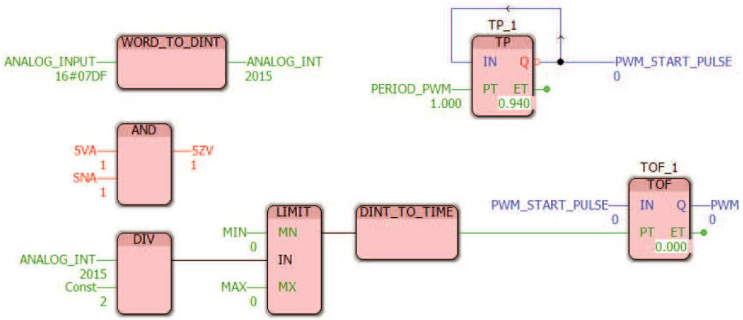
СМАРТ-ТЕХНОЛОГИИ: ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОКОВЫХ ДИАГРАММ

Н.А. Седова
В.А. Седов

СМАРТ-ТЕХНОЛОГИИ

ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОКОВЫХ ДИАГРАММ

Учебное пособие



Министерство образования и науки Российской Федерации

ФГБОУ ВО «Владивостокский государственный университет
экономики и сервиса»

Н.А. Седова
В.А. Седов

СМАРТ-ТЕХНОЛОГИИ

ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОКОВЫХ ДИАГРАММ

Учебное пособие

Владивосток
Издательство ВГУЭС
2017

УДК 62
ББК 32.96
С28

Рецензенты: *Н.В. Сгребнев*, канд. техн. наук, профессор каф. электрооборудования и автоматики судов ФГБОУ ВО «Морской государственный университет им. адм. Г.И. Невельского»;
Н.Л. Халаев, канд. техн. наук, доцент каф. информационных технологий и систем ФГБОУ ВО «Владивостокский государственный университет»

Седова, Н.А.

С28 **Смарт-технологии: язык функциональных блоковых диаграмм** [Текст] : учебное пособие / Н.А. Седова, В.А. Седов ; Владивостокский государственный университет экономики и сервиса. – Владивосток : Изд-во ВГУЭС, 2017. – 220 с.

ISBN 978-5-9736-0406-6

Учебное пособие по дисциплине «Смарт-технологии» составлено в соответствии с требованиями ФГОС ВО для студентов, обучающихся по направлениям подготовки 09.03.02 «Информационные системы и технологии» и 11.03.02 «Инфокоммуникационные технологии и системы связи».

УДК 62
ББК 32.96

ISBN 978-5-9736-0406-6

ВВЕДЕНИЕ

Современные условия развития науки и техники выдвигают повышенные требования к системам автоматизации, и разработка программ для программируемых логических контроллеров (ПЛК) открывает новые возможности для создания новых актуальных устройств и систем.

Учебное пособие предназначено для обучения студентов, обучающихся по направлениям подготовки 09.03.02 «Информационные системы и технологии» и 11.03.02 «Инфокоммуникационные технологии и системы связи», основам программирования ПЛК на языке Function block diagram (FBD), входящем в состав языков программирования, соответствующих стандарту Международной электротехнической комиссии (МЭК) 61131-3. Учебное пособие включает в себя материалы, предназначенные для поэтапного овладения навыками работы с ПЛК на примере программируемых логических контроллеров ILC 131 Starterkit фирмы Phoenix Contact.

Первая глава учебного пособия предназначена для ознакомления с функциональными возможностями стартового комплекта ILC 131 Starterkit, использующегося при выполнении лабораторных работ, и обучения работе в среде программирования PC WorX v.6.30. Последующие главы содержат цикл лабораторных работ на программируемых логических контроллерах на языке FBD. Следует отметить, что каждая тема содержит несколько лабораторных работ для выполнения, например, тема «Арифметические и тригонометрические операции», в предварительной части которой содержатся основные сведения о базовых блоках арифметических и тригонометрических операций, насчитывает четыре лабораторные работы разной степени сложности для обучающихся с различным уровнем знаний. Такая структура учебного пособия позволяет обучающемуся, изучившему теоретическую часть, получить индивидуальный вариант для самостоятельного решения, а преподавателю – сформировать различные индивидуальные задания для контроля усвоения изучаемого раздела. Общее число лабораторных работ, включенных в учебное пособие, равно 35.

Выполнение лабораторных работ, предложенных в учебном пособии, позволяет сформировать у обучающихся основные навыки работы с программируемыми логическими контроллерами, использующимися для разработки систем автоматизации, а также сформировать профессиональные компетенции в области создания проектов на программируемых логических контроллерах с использованием среды PC WorX v.6.30, являющейся унифицированной средой разработки для контроллеров всех классов.

Завершают учебное пособие тестовые вопросы для промежуточного и итогового контроля знаний, которые обучающиеся также могут использовать для контроля знаний при самоподготовке.

Тема 1. АРХИТЕКТУРА КОМПЛЕКТА ILC 131 STARTERKIT, СРЕДА ПРОГРАММИРОВАНИЯ PC WORX

Тема посвящена изучению учебного комплекта ILC 131 Starterkit, а также знакомству со средой программирования PC WorX, с использованием которых выполняются лабораторные работы в рамках дисциплины «Смарт-технологии» для студентов, обучающихся по направлениям подготовки 09.03.02 «Информационные системы и технологии» и 11.03.02 «Инфокоммуникационные технологии и системы связи».

1.1. Описание лабораторного оборудования

Стартовый комплект ILC 131 Starterkit предназначен для ознакомления с функциональными возможностями ПЛК ILC 131 фирмы Phoenix Contact, который обеспечивает возможность создания различных автоматизированных систем.

Стартовый комплект ILC 131 поставляется в собранном виде и подготовленном к работе состоянии. Внешний вид комплекта показан на рис. 1.

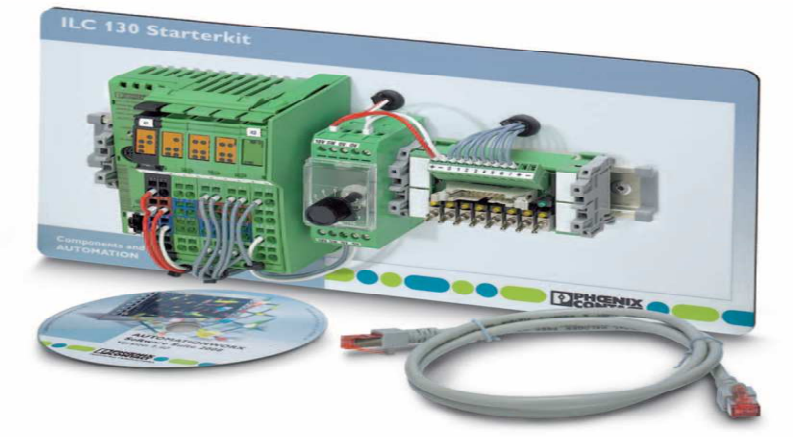


Рис. 1. Стартовый комплект ILC 130 Starterkit

Все компоненты закреплены на рабочей панели на монтажной DIN-рейке и включают в себя Inline-контроллер ILC 131 ETH 2, коммутационную панель 5 (8-канальный имитатор сигналов UM 45-IB-DI/SIM8), блок питания 1, терминал аналогового ввода 3 (клеммный модуль аналогового ввода Inline, модели Inline-ME (Machine Edition) IB IL AI 2/SF-ME) и потенциометр 4. Состав комплекта представлен на рис. 2.

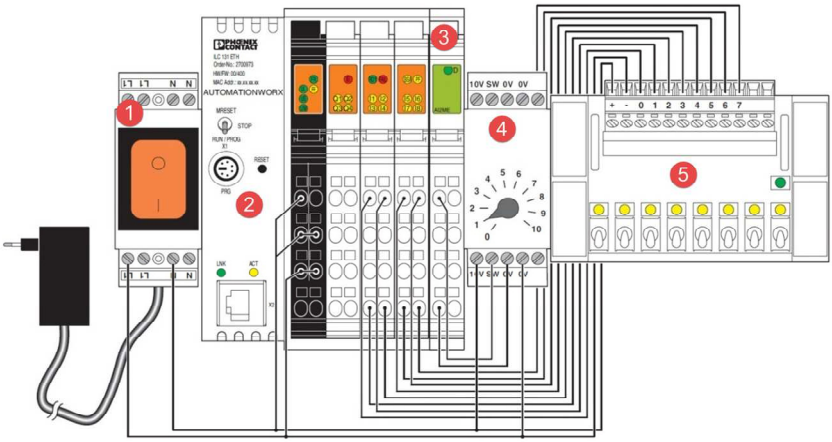


Рис. 2. Комплект ILC 130 Starterkit. 1 – блок питания; 2 – контроллер ILC 131 ETH; 3 – терминал аналогового ввода IB IL AI 2/SF-ME; 4 – потенциометр; 5 – коммутационная панель

На компакт-диске с ПО AutomationSoftwareSuite также содержится среда разработки PC WorX.

1.1.1. Контроллер Inline ILC 131 ETH

Контроллеры Inline – это контроллеры с интегрированной коммуникацией Ethernet и INTERBUS. Семейство контроллеров Phoenix Contact ILC 130 ETH, ILC 150 ETH и ILC 155 ETH имеют идентичный внешний вид и ряд одинаковых функций и характеристик. Основное отличие заключается в объёме памяти. Контроллер ILC 170 ETH 2TX оснащен дополнительным портом Ethernet и слотом для карты памяти формата SD.

Все контроллеры Inline одинаково параметризуются и программируются согласно МЭК 61131, используя пакет программного обеспечения PC WorX. Подключение PC WorX к контроллеру производится через сеть Ethernet. Процессор программируется на пяти языках согласно МЭК 61131 и обеспечивают выполнение задач управления.

Контроллеры Inline состоят из следующих элементов (рис. 3):
1 – процессорный модуль;
2 – переключатель режима работы;
3 – кнопка сброса;
4 – интерфейс V.24 (RS-232);
5 – Ethernet порт;
6 – штекер 1: Клеммы для подключения питания;
7 – штекер 2: Клеммы дискретных выходов;
8 – штекеры 3 и 4: Клеммы дискретных входов;
9 – индикаторы;
10 – оконечная пластина.

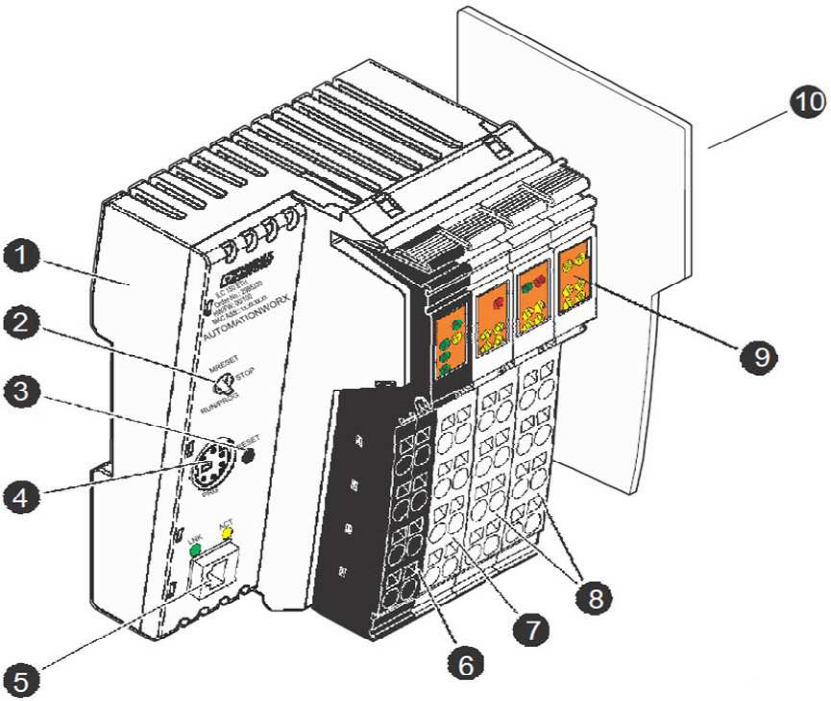


Рис. 3. Элементы контроллеров Inline (ILC 130 ETH, ILC 150 ETH, ILC 155 ETH)

Встроенный порт Ethernet (кабель «витая пара») используется для подключения в сеть. Через сеть Ethernet контроллеры Inline могут быть доступны по протоколам TCP/IP или UDP/IP. Сеть Ethernet предоставляет универсальный канал для связи с контроллером. Используя функциональные блоки приема и отправки, контроллеры Inline могут обмениваться

данными и переменными через сеть Ethernet. Это позволяет создать распределенную, модульную систему автоматизации.

Интерфейс V.24 (RS-232) позволяет назначать контроллеру IP адрес или подключаться пакетом диагностики Diag+, либо осуществлять обмен данными с подключенным устройством с помощью встроенных функциональных блоков. Контроллер не имеет возможности программирования через интерфейс V.24 (RS-232).

INTERBUS. Локальная шина Inline и шина распределенного ввода-вывода INTERBUS подключаются к контроллеру через соответствующий интерфейс. Таким образом, можно создать полноценную систему INTERBUS (с максимум четырьмя уровнями шины). Система ввода-вывода, подключаемая к контролеру Inline, основана на INTERBUS.

При сборке рабочей станции модули системы INTERBUS Inline защелкиваются в ряд подобно обычным шинным клеммам. При этом периферийные модули автоматически соединяются между собой системной шиной, создавая общую структуру всех электрических связей рабочей станции. Модернизация такой системы производится добавлением или изъятием определенных модулей.

В распределенных системах точки ввода и вывода данных иногда сильно рассеяны по объекту управления. При параллельном соединении всех точек ввода-вывода чрезмерно много средств затрачивается на кабели и их прокладку. Существует простое и экономичное решение в виде промышленного мультиплексора, который передает поток данных по двухпроводной шине. Пользователю нет необходимости вдаваться в подробности теории локальных сетей: всё возложено на мультиплексор. В этом случае на объекте устанавливаются датчики и исполнительные устройства, их подключают к интерфейсным модулям INTERBUS Inline (модули собираются на DIN-рейке в ряд вместе с мультиплексором) и получившуюся станцию связывают с аналогичной станцией по двухпроводной линии.

Контроллеры серии ILC 150 ETH/ILC 155 ETH/ILC 170 ETH 2TX также могут быть использованы как распределенная система управления на базе системы Inline, подключаемая в сеть Ethernet. Локальная шина Inline, а также полноценная шина INTERBUS с максимально четырьмя уровнями (рис. 4) подключаются к контроллеру. Удаленная шина подключается с помощью модулей IBS IL 24 RB-T или IBS IL 24 RB-T-PAC. Первый модуль удаленной шины устанавливается первым в сборке Inline (сразу после контроллера). В терминах топологии он открывает ветвь удаленной шины. Если необходимо реализовать несколько ветвей удаленной шины, то следующий модуль удаленной шины устанавливается сразу после первого. К контроллеру можно подключить до трех модулей удаленной шины.

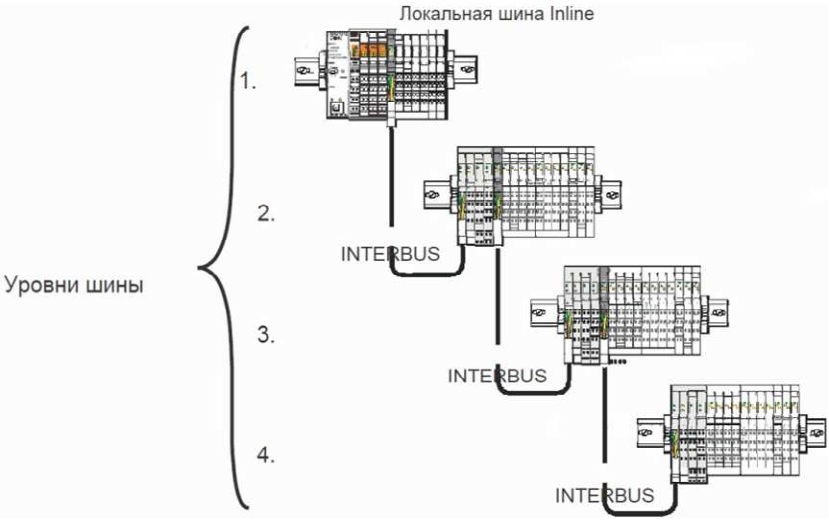


Рис. 4. Пример подключения модулей удаленной шины

Светодиодные индикаторы на панели контроллера (рис. 5) предназначены для быстрого определения статуса и ошибок контроллера (для всех контроллеров серии ILC одинаково). Данные по индикаторам статуса и ошибок сведены в табл. 1.

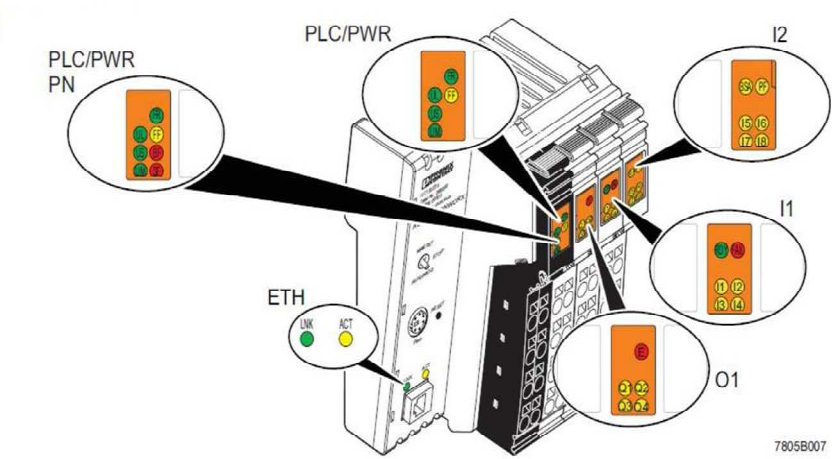


Рис. 5. Индикаторы статуса и ошибок контроллеров Inline

Таблица 1

Индикаторы статуса и ошибок

Наименование	Цвет	Статус	Значение
1	2	3	4
ETH: Статус интерфейса Ethernet (для ILC 170 ETH 2TX, применимо для обоих портов (X2, 1, X2.2))			
LNK	Зелёный	ВЫКЛ	Нет подключения
		ВКЛ	Подключение установлено (link): Контроллер готов осуществлять обмен
ACT	Жёлтый	ВЫКЛ	Передача данных не ведётся
		ВКЛ	Передача данных активна: приём или передача
Диагностика состояния контроллера			
		Контроллер Inline работает	
FR	Зелёный	ВЫКЛ	Среда исполнения МЭК 61131 не готова к работе
		Мигает	Среда исполнения МЭК 61131 успешно запущена. Контроллер в состоянии READY/STOP, программа не запущена
		ВКЛ	Среда исполнения МЭК 61131 успешно запущена: программа контроллера исполняется. Контроллер в состоянии RUN
		Сбой	
FF	Жёлтый	ВЫКЛ	Среда исполнения МЭК 61131 работает без ошибок
		ВКЛ	Ошибка среды исполнения МЭК 61131 (сбой микропрограммного обеспечения)
FR+FF		Мигает	Применимо только для ILC 170 ETH 2TX: Отсутствует карта памяти
I/O: Дискретные входы и выходы на борту контроллера			
		Входы 1–8	
I1-I8	Жёлтый	ВЫКЛ	Вход не активирован
		ВКЛ	Вход активирован
		Ошибка	

Продолжение табл. 1

1	2	3	4
E	Жёлтый	ВЫКЛ	Короткое замыкание / перегрузка выходов 1-4 не обнаружено
		ВКЛ	Короткое замыкание / перегрузка выходов
		Выходы 1-4	
Q1-Q4	Жёлтый	ВЫКЛ	Выход не активирован
		ВКЛ	Выход активирован
PWR: Электропитание			
		Питание U_{ILC} 24 В для питания U_L и U_{ANA}	
UL	Зелёный	ВЫКЛ	Электропитание не подключено
		ВКЛ	Электропитание подключено (включено при наличие напряжения 24 В на U_{ILC}
		Питание 24 В для сегментной цепи	
US	Зелёный	ВЫКЛ	Электропитание не подключено
		ВКЛ	Электропитание подключено
		Питание 24 V для основной цепи питания	
UM	Зелёный	ВЫКЛ	Электропитание не подключено
		ВКЛ	Электропитание подключено
IL: диагностика INTERBUS			
		***INTERBUS мастер готов к работе / передача данных активна* (*INTERBUS ready/running*)**	
RDY	Зелёный	ВЫКЛ	INTERBUS мастер не готов к работе
		Мигает	INTERBUS мастер в состоянии READY или ACTIVE (готов или активен)
		ВКЛ	INTERBUS мастер в состоянии RUN (запущен)
		Сбой	
FAIL	Красный	ВЫКЛ	Система работает без ошибок
		ВКЛ	Обнаружена одна из следующих ошибок: – Ошибка на шине (удаленная шина или локальная) – Ошибка контроллера

Окончание табл. 1

1	2	3	4
		Сегмент шины отключен	
BSA	Жёлтый	ВЫКЛ	Все сегменты шины включены
		ВКЛ	Один или более сегментов шины отключены
		Периферийная ошибка (*не критическая ошибка на шине*)	
PF	Жёлтый	ВЫКЛ	Все устройства работают нормально
		ВКЛ	Периферийная ошибка на локальной или удаленной шине

Переключатель режима работы (рис. 6) предназначен для установки режима работы программы контроллера. Положения RUN/PROG и STOP фиксируются. Положение MRESET не фиксируется. После нажатия переключателя в положение MRESET переключатель возвращается в положение STOP.

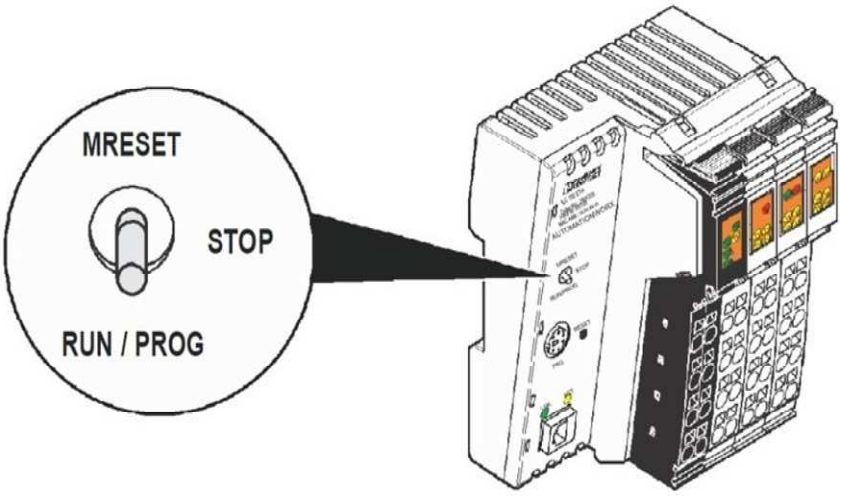


Рис. 6. Переключатель режима работы контроллера Inline

Таблица 2

Режимы работы

Режим	Расшифровка
RUN/PROG	Программа контроллера запущенна (состояние RUN). Возможно программирование контроллера и изменение. Возможен режим отладки и мониторинга конфигурации.
STOP	Программа контроллера остановлена (состояние STOP).
MRESET	Очистка энергонезависимой памяти и удаление программы контроллера. Для удаления программы и очистки энергонезависимой памяти следует выполнить следующие действия: a) перевести переключатель в режим MRESET на 3 секунды; b) отпустить переключатель на 3 секунды; c) перевести переключатель в режим MRESET на 3 секунды.

Кнопка сброса (см. рис. 3) защищена от случайного нажатия. Нажатие кнопки возможно только с помощью тонкого предмета (например, скрепки). Контроллер возвращается к заводским настройкам при одновременном нажатии кнопки сброса и отключении питания контроллера с последующим включением питания. Кнопку сброса следует отпустить только при попеременном мигании светодиодов FR (Зеленый) и FF (Желтый).

Контроллер успешно загрузился с заводскими настройками, если светодиоды FR (Зеленый) и RDY (Зеленый) начали мигать. Когда контроллер находится в режиме READY/STOP, программа не исполняется. Данная процедура занимает примерно одну минуту.

Подключение питания. Электропитание контроллера производится с помощью внешнего источника питания 24 V постоянного тока. Допустимый диапазон от 19,2 V DC до 30 V постоянного тока. Потребляемая мощность при напряжении 24 V составляет 4.8 W (при отсутствии подключенных к контроллеру модулей).

Процедура включения электропитания. Подключить источник питания к штекеру питания контроллера, согласно рис. 7 (разводка электропитания представлена в табл. 3), присоединить штекер питания к контроллеру и включить источник питания. Светодиоды U_L, U_M и U_S включатся, светодиоды FR и RDY начнут мигать. Контроллер Inline запущен. Если светодиоды не зажглись или не начали мигать, то в контроллере произошел внутренний сбой.

Сегментное и основное питание имеют общий опорный потенциал. Гальваническая развязка невозможна.

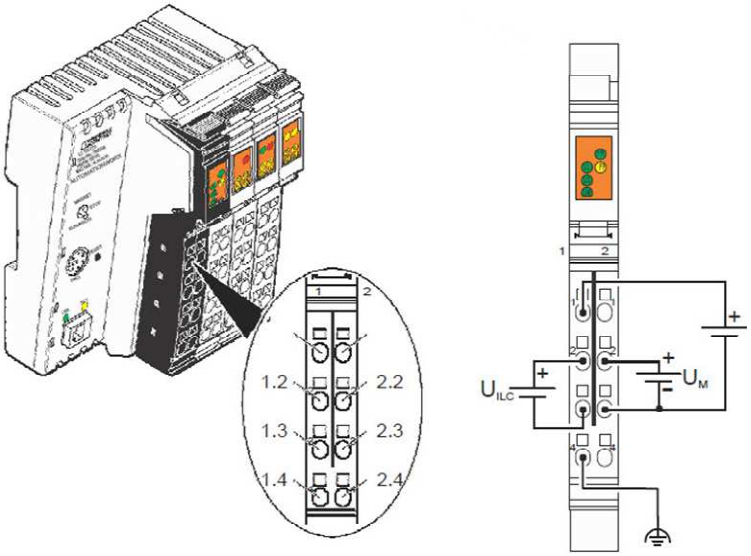


Рис. 7. Электропитание контроллера Inline

Таблица 3

Разводка электропитания контроллера

Клемма	Назначение		Примечание
Штекер 1	Штекер питания		
1	2	3	4
1.1	24 V DC (U_S)	24 V питание сегмента	Подключенное питание напрямую заведено на внутреннюю шину. ***Замечание***: необходимо защитить цепь питания предохранителем номиналом максимально 8 А. Удостоверьтесь, что предохранитель сработает в любом случае.
1.2	24 V DC (U_{ILC})	24 V	Питание связи модулей 7.5 V (U_L) и питание подключенных устройств генерируется из данной цепи. Питание 24 V аналоговых цепей (U_{ANA}) так же генерируется из этой цепи. ***Замечание***: Необходимо защитить цепь питания предохранителем номиналом максимально 2 А. Удостоверьтесь, что предохранитель сработает в любом случае.

Окончание табл. 3

1	2	3	4
2.1, 2.2	24 V DC (U_M)	24 V питание основной цепи	Питание основной цепи проходит через все устройства Inline через внутреннюю шину. ***Замечание***: Необходимо защитить цепь питания предохранителем номиналом максимально 8 А. Удостоверьтесь, что предохранитель сработает в любом случае.
1.3	LGND	Опорный потенциал логики	Опорный потенциал цепи питания логики.
2.3	SGND	Опорный потенциал сегмента	Опорный потенциал заводится непосредственно на внутреннюю шину питания и является опорным для основной и сегментной цепей.
1.4, 2.4	FE	Защитное заземление (FE)	Защитное заземление подключается через источник питания. Контакт соединен с внутренним распределителем и контактом на стороне крепления на рейку. Контроллер заземляется при установке на монтажную рейку. Подключение к заземлению служит для подавления помех.

Есть несколько способов подключения сегментного питания через штекер 1:

1) сегментное питание подключается раздельно через клеммы 1.1 и 2.3 (GND) (рис. 7);

2) возможно установить перемычку между клеммами 1.1 и 2.1 (или 2.2) для подключения питания через основную цепь;

3) можно создать отключаемый сегмент, установив переключатель между клеммами 1.1 и 2.1 (или 2.2).

Замечание: Максимальный допустимый ток через внутреннюю шину питания 8 А.

Дискретные входы и выходы. Описание дискретных входов и выходов применимо для всей серии контроллеров ILC. На контроллерах ILC установлено восемь входов 24 V DC и четыре выхода 24 V DC (табл. 4–6).

Таблица 4

Описание дискретных входов и выходов

Клемма	Наименование	Примечание
1	2	3
Штекер 4	Клеммы входов	
3.1	I5	Вход 5
4.1	I6	Вход 6

Окончание табл. 4

1	2	3
3.2, 4.2	24 V	Цепь питания U_M для двух- или трёхпроводного подключения
3.3, 4.3	GND	Общая точка для трёхпроводного подключения
3.4	I7	Вход 7
4.4	I8	Вход 8

Замечание: Питание входов 24 V DC осуществляется от основной цепи (U_M).

Таблица 5

Описание дискретных входов и выходов

Клемма	Наименование	Примечание
Штекер 2	Клеммы выходов	
1.1	Q1	Выход 1
2.1	Q2	Выход 2
1.2, 2.2	GND	Контакт общей точки для двух- или трёхпроводного подключения
1.3, 2.3	FE	Защитное заземление для трёхпроводного подключения
1.4	Q3	Выход 3
2.4	Q4	Выход 4

Замечание: Выходы питаются от сегментной цепи (U_S).

Таблица 6

Описание дискретных входов и выходов

Клемма	Наименование	Примечание
1	2	3
Штекер 3	Клеммы входов	
1.1	I1	Вход 1
2.1	I2	Вход 2
1.2, 2.2	24 V	Цепь питания U_M для для двух- или трёхпроводного подключения

Окончание табл. 6

1	2	3
1.3, 2.3	GND	Общая точка для трёхпроводного подключения
1.4	I3	Вход 3
2.4	I4	Вход 4

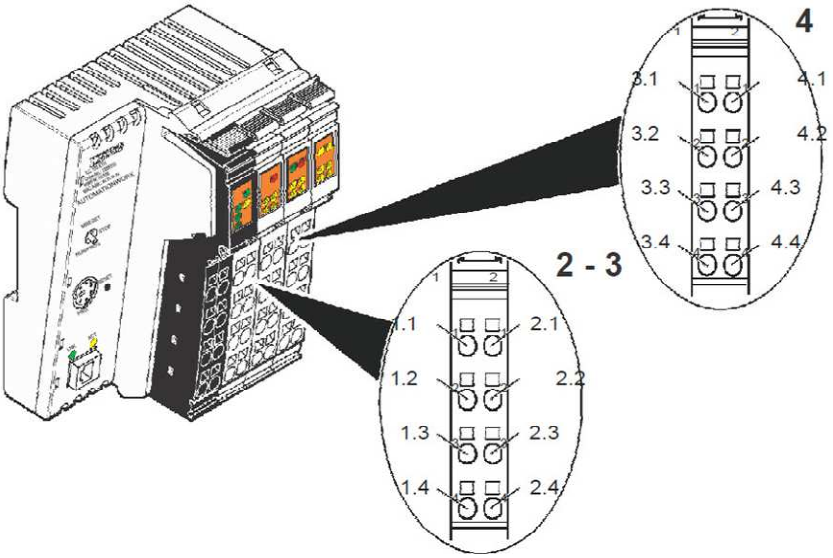


Рис. 8. Дискретные входы и выходы

1.1.2. Терминал аналогового ввода IB IL AI 2/SF-ME

Терминал аналогового ввода разработан в виде блока, интегрирующегося в контроллер ILC 131, и предназначен для измерения аналогового напряжения или тока. Входящий в стартовый комплект терминал IB IL AI 2/SF-ME (рис. 9) имеет два независимых 12-битных аналоговых входа с возможностью измерения напряжения в диапазоне от 0 V до 10 V либо ±10 V или тока в диапазонах от 0 mA до 20 mA, ±20 mA и от 4 mA до 20 mA. Время обработки данных с двух каналов не более 1.5 ms.

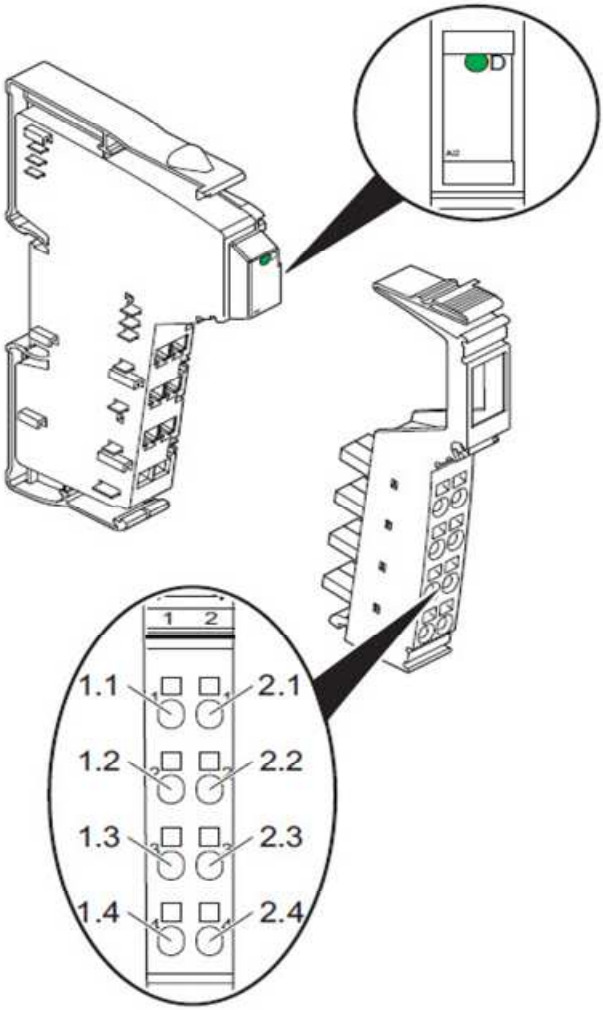


Рис. 9. Терминал IB IL AI 2/SF-ME

Таблица 7

Статусный индикатор

Индикатор	Цвет	Назначение
D	Зеленый	Диагностика

Таблица 8

Назначение выводов

Вывод	Сигнал	Назначение
1.1	+U1	Входное напряжение, канал 1
2.1	+U2	Входное напряжение, канал 2
1.2	+I1	Входной ток, канал 1
2.2	+I2	Входной ток, канал 2
1.3	- U1, – I1	Общий для канала 1
2.3	- U2, – I2	Общий для канала 2
1.4, 2.4	GND	Земля (экранирование)

Подключение коммутационной панели и аналогового терминала к контроллеру показано на рис. 2, для упрощения понимания эти данные сведены в табл. 9.

Таблица 9

Подключение коммутационной панели к контроллеру

Устройство	Вход	Описание	Переменная
ILC 131	Вход I1	штекер 1.1	ONBOARD_INPUT_BIT0
	Вход I2	штекер 2.1	ONBOARD_INPUT_BIT1
	Вход I3	штекер 1.4	ONBOARD_INPUT_BIT2
	Вход I4	штекер 2.4	ONBOARD_INPUT_BIT3
	Вход I5	штекер 3.1	ONBOARD_INPUT_BIT4
	Вход I6	штекер 4.1	ONBOARD_INPUT_BIT5
	Вход I7	штекер 3.4	ONBOARD_INPUT_BIT6
	Вход I8	штекер 4.4	ONBOARD_INPUT_BIT7
IB IL AI 2/SF-ME	Вход I9	штекер 1.1	Input_Analog

Как видно из табл. 9, нумерация системных переменных, использующихся для чтения состояния локальных дискретных входов, и нумерация непосредственно дискретных входов не совпадают. Таким образом, тумблерам на коммутационной панели соответствуют входы и переменные, как показано в табл. 10.

Таблица 10

Соответствие нумерации тумблеров коммутационной панели и системных переменных

Вход	Описание	Переменная
Тумблер 0	Вход I1 (штекер 1.1)	ONBOARD_INPUT_BIT0
Тумблер 1	Вход I2 (штекер 2.1)	ONBOARD_INPUT_BIT1
Тумблер 2	Вход I5 (штекер 3.1)	ONBOARD_INPUT_BIT4
Тумблер 3	Вход I6 (штекер 4.1)	ONBOARD_INPUT_BIT5
Тумблер 4	Вход I3 (штекер 1.4)	ONBOARD_INPUT_BIT2
Тумблер 5	Вход I4 (штекер 2.4)	ONBOARD_INPUT_BIT3
Тумблер 6	Вход I7 (штекер 3.4)	ONBOARD_INPUT_BIT6
Тумблер 7	Вход I8 (штекер 4.4)	ONBOARD_INPUT_BIT7

1.2. Описание программного обеспечения

Для создания и отладки проектов на стартовом комплекте ILC 131 Starterkit фирмы Phoenix Contact предлагается использовать среду программирования PC WorX, поставляемую в комплекте.

PC WorX является частью программного комплекса AUTOMATIONWORX (AUTOMATIONWORX Software Suite). Программный комплекс AUTOMATIONWORX включает в себя следующие программы:

1. CONFIG +

Программы для настройки и эксплуатации сетей INTERBUS.

2. Diag +

Сетевая диагностика при подключении и работе ПЛК.

Diag + NetScan

Программы мониторинга нескольких сетей INTERBUS.

PC WorX

Единая среда программирования на языках МЭК 61131 для всех контроллеров Phoenix Contact.

PC WorX Express

Простая в использовании версия PC WorX для упрощенного обучения программирования МЭК 61131 для ПЛК Phoenix Contact ILC1XX.

6. AX OPC-сервер

Программное обеспечение для обмена данными между распределенными сетями INTERBUS и системой визуализации.

7. WebVisit

Инструмент для создания графических пользовательский интерфейсов на основе технологии web.

1.2.1. Установка программного обеспечения

Запуск программы установки AUTOMATIONWORX Software Suite происходит автоматически после ввода диска в дисковод, либо через запуск файла Setup.exe. Таким образом, вызывается Мастер инсталляции, следуя инструкциям которого устанавливается необходимое для работы ПО. Для выполнения лабораторных работ требуется установить PC WorX PLC Programming v. 6. 0 и WebVisit 6.x.

1.2.2. Описание среды программирования PC WorX

Программирование всех контроллеров Phoenix Contact, в том числе и серии Inline, возможно в универсальной среде PC WorX, известной также как Multiprog, компании KW_Software GmbH. Указанная компания не занимается выпуском ПЛК, а её основная цель заключается в создании универсальной среды разработки, соответствующей стандарту IEC 61131-3, и её адаптация к максимально большому числу производителей ПЛК [2].

Поскольку при разработке ПО PC WorX внимание было сфокусировано именно на создании унифицированной среды разработки для контроллеров всех классов, то наряду с удобством для пользователя в PC WorX предусмотрена возможность повторного использования различных программ и функций. Интерфейс PC WorX обеспечивает простой и быстрый обзор проекта. Использование окон упрощено за счет их отображения в виде рабочих книг. Прикрепляемые и открепляемые рабочие панели и конфигурируемые строки меню, которые можно настроить под требования любого пользователя, увеличивают эффективность процесса программирования. Среда разработки PC WorX поддерживает языки программирования, соответствующие МЭК 61131-3, причём программы на основных языках МЭК 61131, к которым относятся LD, FBD и IL, могут свободно переписываться с одного языка на другой. Для ускорения и упрощения редактирования в программах-редакторах предусмотрены специальные функции и мастера, контролирующие тип вводимых данных, функциональные блоки, операторы и объявления переменных. Для текстовых редакторов дополнительно предусмотрена программа-мастер для задания ключевых слов и форматов команд [2].

В PC WorX встроен конфигуратор шины, позволяющий спроектировать структуру сети. Поддерживаются сети Interbus и Profinet IO. Системы на базе полевых шин другого типа могут быть подключены через прокси-сервер, и проектирование осуществляется на основе файла описания уст-

ройства. Каталог конфигуратора предоставляет все необходимые компоненты, разбитые по разделам, перетягиваемые мышью в окно аппаратной конфигурации. В режиме просмотра связей с помощью перетаскивания мышью можно связать различные программные переменные с входами/выходами компонентов. При попытке создания ошибочной связи на дисплей выводится соответствующее окно с предупреждением. Адресация переменных в памяти устройства управления производится автоматически. Все имеющиеся в программе комментарии можно экспортировать, перевести и затем обратно импортировать в программу. Таким образом, можно создавать проекты с комментариями на различных языках программистов и пользователей [2].

Встроенная система защиты по паролю поддерживает различные функции, например, возможна защита как всего проекта в целом, так и защита от чтения или записи отдельных программных блоков, а также запрещение включения/отключения контроллеров [2].

Для тестирования программ с целью нахождения ошибок для всех совместимых с INTEL® устройств управления поставляются имитаторы, которые позволяют произвести проверку программы на компьютере, что позволяет сократить продолжительность последующего ввода системы в эксплуатацию. Логический анализатор обеспечивает сбор значений всех переменных в режиме реального времени, а также элементов структур и массивов, которые вводятся непосредственно из рабочих расчетных таблиц. Возможна запись на жесткий диск хронограмм [2].

Установленную среду программирования PC WorX можно запустить через главное меню: при помощи двойного щелчка по значку программы или заранее созданному ярлыку на рабочем столе Windows, либо щелчком по значку программы в Панели задач Windows.

Интерфейс. Интерфейс PC WorX состоит из следующих основных компонент – панель меню (*menu bar*), панель инструментов (*tool bar*), главное окно (*main window*) и панель статуса (*status bar*), см. рисунок 10. Содержимое окон зависит от выбранного рабочего пространства.

1.2.3. Панель инструментов

Программа содержит несколько панелей инструментов с набором значков для быстрого доступа к часто используемым операциям. Помимо значков требуемые операции могут быть вызваны через элементы меню или предустановленные ярлыки. По умолчанию открыты все панели инструментов. Чтобы скрыть или отобразить определенную панель инструментов, можно использовать диалоговое окно <*Extras → Options → Toolbars*>.

При наведении указателя мыши на значок (без нажатия) у указателя появляется всплывающая подсказка – имя текущего значка, и в строке состояний появляется краткое описание функции. Если подсказки не ото-

бражаются, эта функция может быть активирована в диалоговом окне <*Extras → Options → Toolbars*>.

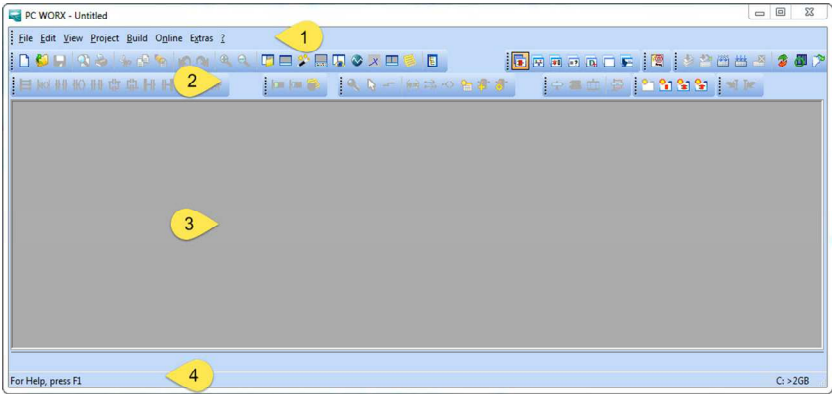


Рис. 10. Пользовательский интерфейс PC WorX. 1 – панель меню, 2 – панель инструментов, 3 – главное окно, 4 – панель статуса

Рабочее пространство. Панель инструментов для выбора рабочего пространства (*Workspaces*) состоит из семи значков для выбора семи конфигурируемых рабочих пространств (вызываются через вкладку <View>), отображающих соответствующие комбинации рабочих окон, которые могут быть, при необходимости, показаны/скрыты.

Каждым из рабочих пространств учитывается доступ к основным функциям PC WorX:

1) IEC Programming Workspace – программирование;
2) Bus Configuration Workspace – конфигурация шины;
3) Process Data Workspace – пространство данных;
4) Project Comparison Result Workspace – сравнение проектов;
5) Workspace Diagnostic – диагностика;
6) Workspace User Defined – пользовательское рабочее пространство;
7) FDT (Field Device Tool) workspace – инструментальное рабочее пространство.

Настройки рабочих пространств могут быть сброшены через <*Extras → Options → General → Reset Workspaces*>.

Основными используемыми рабочими пространствами являются: IEC Programming workspace, Bus Configuration Workspace и Process Data Workspace.

IEC Programming workspace – пространство для программирования ПЛК на языках МЭК (программы, функции и функциональные блоки), объявления определяемых пользователем типов данных, интеграции биб-

лиотек, создания пользовательских функций и функциональных блоков (рис. 11). Пространство состоит из дерева проекта (*Project Tree*) и мастера редактирования (*Edit Wizard*). Помимо полного отображения проекта дерево проекта предлагает определенное отображение для организационных модулей программы (program organization units – POU), библиотек, оборудования и задач.

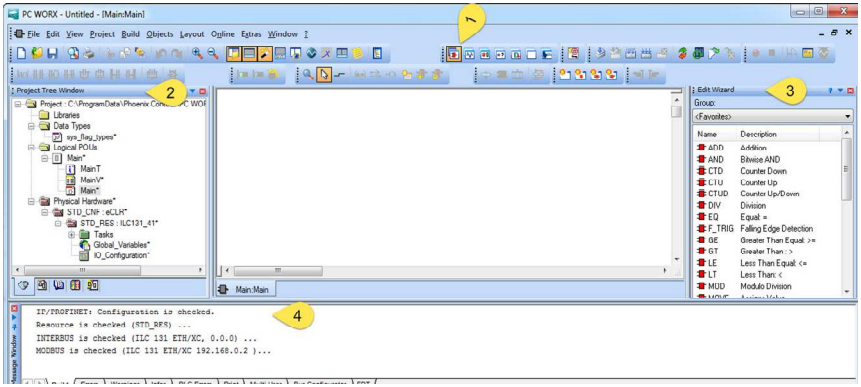


Рис. 11. Рабочее пространство IEC Programming: 1 – значок вызова рабочего пространства IEC Programming на панели инструментов, 2 – дерево проекта, 3 – мастер редактирования, 4 – информационное окно

Мастер редактирования *Edit Wizard* является контекстно-зависимым, т.е. в зависимости от того, с каким элементом работаем, мастер предлагает помощь в создании пользовательских типов данных или программы. Рабочая область используется для создания рабочих страниц (*worksheets*) (с рабочим кодом и таблицами переменных). Для увеличения пространства рабочих страниц дерево проекта мастер редактирования и окно сообщений можно скрыть (они автоматически всплывут во время процесса компиляции).

Bus Configuration Workspace – рабочее пространство, предназначеное для создания шин, поддерживаемых контроллером (INTERBUS и PROFINET), редактирования и общего управления устройствами (рис. 12). Рабочее пространство Bus Configuration Workspace используется для начальной конфигурации оборудования при создании проекта. По умолчанию отображаются следующие окна – структура шины (*Bus configuration*), детализация устройства (*Device details*) и каталог устройств (*Device catalog*).

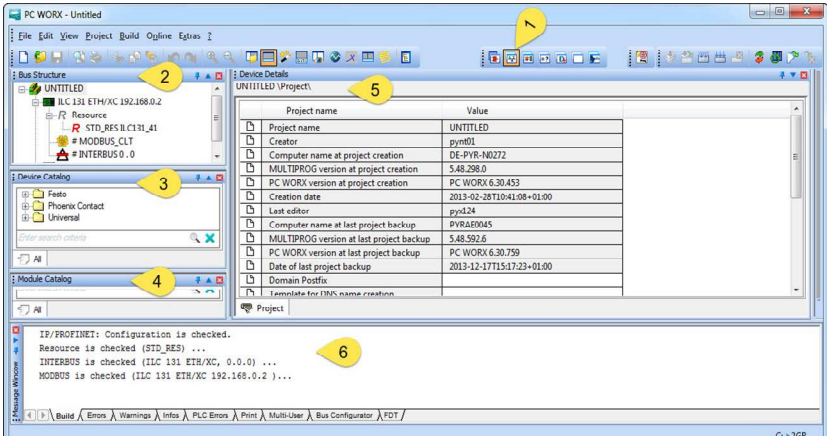


Рис. 12. Рабочее пространство Bus Configuration Workspace: 1 – значок вызова рабочего пространства Bus Configuration Workspace на панели инструментов, 2 – структура шины, 3 – каталог устройств, 4 – каталог модулей, 5 – детализация устройства, 6 – информационное окно

Process Data Workspace – рабочее пространство, предназначенное для установления связи доступных по шине модулей с программными глобальными переменными, а также для создания стандартизированных глобальных переменных для объектов (рис. 13).

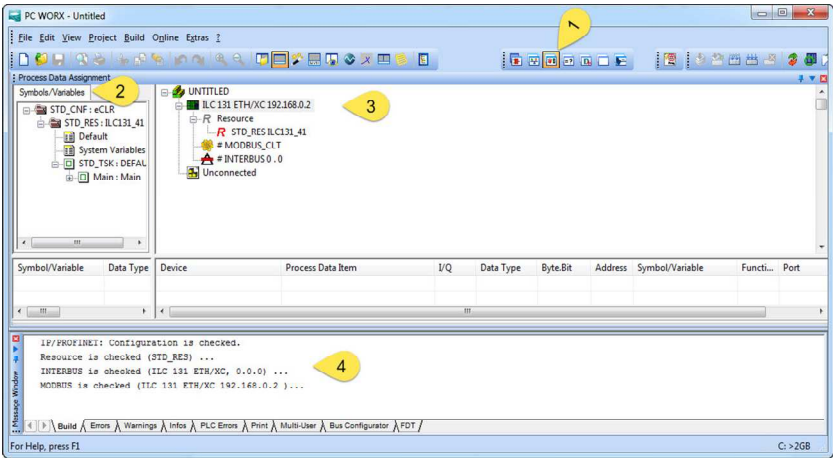


Рис. 13. Рабочее пространство Process Data Workspace. 1 – значок вызова рабочего пространства Process Data Workspace на панели инструментов, 2 – Структура ПЛК и каталог переменных, 3 – структура шины, 4– информационное окно

Рабочее пространство Process Data структурно состоит из двух окон, каждое из которых разделено на два сегмента. Окно слева (маркер **2** на рис. 13) показывает структуру ПЛК, и, в зависимости от выбранного элемента верхнего сегмента окна, в нижнем сегменте отображаются соответствующие глобальные переменные.

Окно справа (маркер **3** на рис. 13) отображает оборудование, подключенное к системной шине (верхний сегмент окна), при выборе элемента в нижнем сегменте появляется детализированное описание – для одиночных устройств приводятся объекты выбранного устройства, для шинных терминалов приводятся доступные данные процесса, для систем управления перечисляются все с ними связанные объекты.

1.2.4. Основные окна вкладки <View>

Помимо переключения между рабочими пространствами вкладка <*View*> позволяет также открывать различные сервисные окна (рис. 14.1).

Окно Bus Structure. Используется для отображения и редакции шинной топологии проекта (рис. 14.2). Отдельные функции устройства в окне Bus Structure обозначены определенными значками, значения которых детализируются в табл. 11.

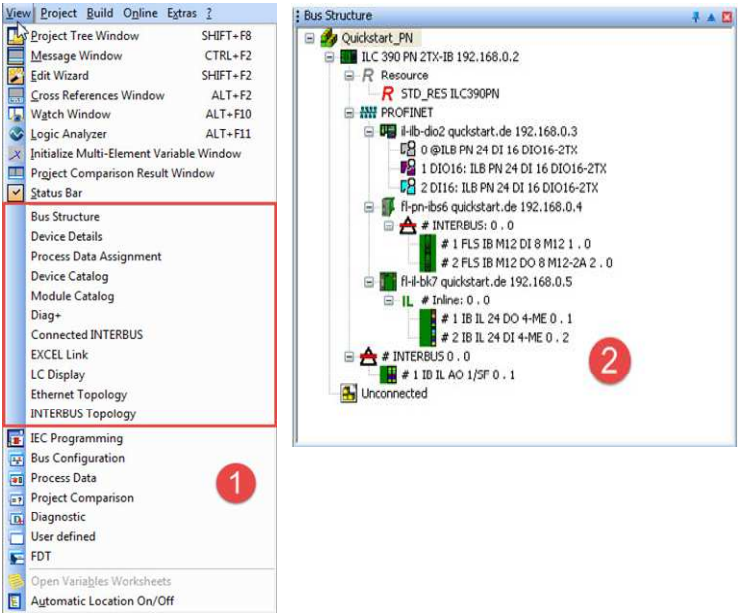


Рис. 14.1 – Вкладка <*View*>; 2 – окно Bus Structure

Таблица 11

Значок	Описание
	**Ресурсы контроллера.** При создании проекта ресурсы контроллера отображаются ниже этого значка.
	**Контроллер PROFINET IO**. Все устройства PROFINET IO вставляются ниже этого значка.
	**Мастер INTERBUS**. Ниже этого значка отображаются все устройства, подключенные по шине INTERBUS.
	**Inline.** Ниже этого значка отображаются терминалы Inline, подключенные к локальной шине.
	**Устройства PROFINET IO**: Интерфейс шины (прокси) и модули. Для устройств PROFINET IO отображается интерфейс шины (прокси, обозначение начинается с «@») и его модули. Номер слота указывается после графического отображения шины или модулей. Значки для шинного интерфейса устройства и его модулей и непосредственно модули хранятся в устройстве и могут варьироваться для разных производителей Устройства PROFINET IO фирмы Phoenix Contact Значки по умолчанию

Окно Bus Structure можно настроить, кликнув правой кнопкой мыши по устройству в окне и выбрав в контекстном меню *<Edit Device Representation>*.

Окно EXCEL Link. Позволяет импортировать/экспортировать данные в файл Excel, а именно МЭК переменные, связанные данные и текст. Таким образом, данные для всех МЭК переменных могут быть отредактированы в страницах Excel, а не в диалоговых окнах PC WorX.

Окно Diag+. Программа Diag+ представляет собой инструмент для диагностики шин INTERBUS и PROFINET. Diag+ интегрирована в PC WorX, устанавливается автоматически и вызывается через окно Diag+.

Окно Connected INTERBUS. Позволяет активировать для чтения устройства, подключенные по шине INTERBUS. Для имеющихся устройств в списке производится сравнение данных о физическом подключении и текущей конфигурации, при этом они подсвечиваются определенным цветом в зависимости от своего состояния (рис. 15): зеленым – идентификатор устройства и длина данных для устройства идентичны; красным – идентификатор устройства и длина данных для устройства различаются; синим – устройство недоступно для выбранной конфигурации шины; серым – для

устройства сравнение не проводилось. Например, для активации модуля IB IL AI 2/SF-ME (модуля аналогового ввода для ILC 131 Starterkit) следует проделать следующую процедуру. Необходимо зайти в окно «*Connected INTERBUS*», открыв выпадающее меню по кнопке «*View*» и выбрав пункт «Connected INTERBUS» (рис. 15).

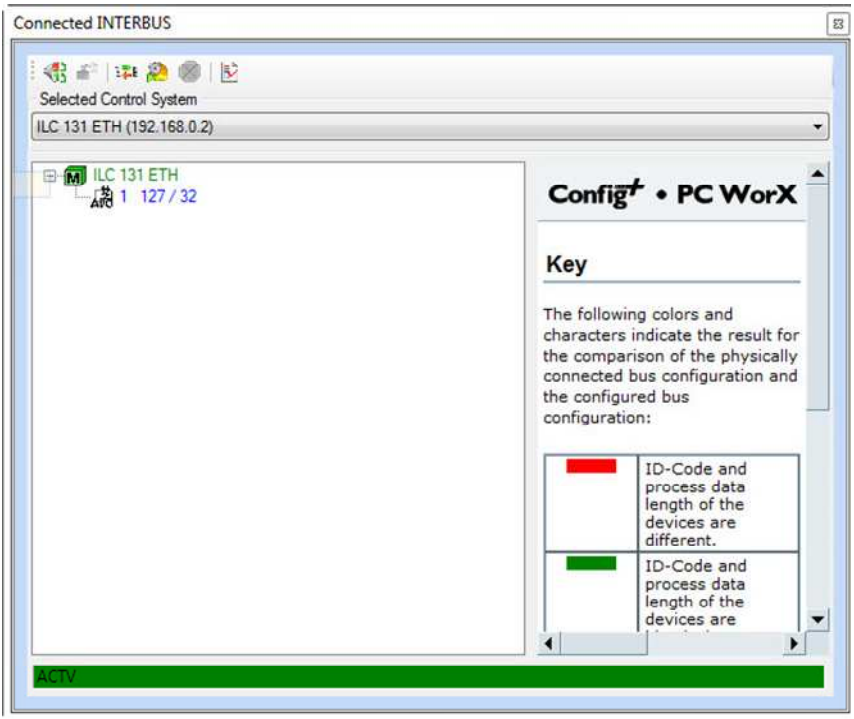


Рис. 15. Окно Connected INTERBUS

В меню «*Selected Control System*» необходимо выбрать ПЛК (ILC 131 ETH). К контроллеру по шине INTERBUS подключен один модуль с идентификационным кодом 127 и строкой данных 32 бит (рис. 16), который активируется кнопкой «*Apply Device/Segment*». После нажатия кнопки «*Apply Device/Segment*» появляется окно «*Select Device*», предлагающее выбрать тип устройства из предлагаемой группы. Далее необходимо выбрать IB IL AI 2/SF-ME, как показано на рисунке 5, нажать «*ок*». Требуется убедиться, что модуль подключен правильно (зеленая полоса и строка RUN или ACTV в нижней части экрана). Далее необходимо закрыть окно «*Connected INTERBUS*».

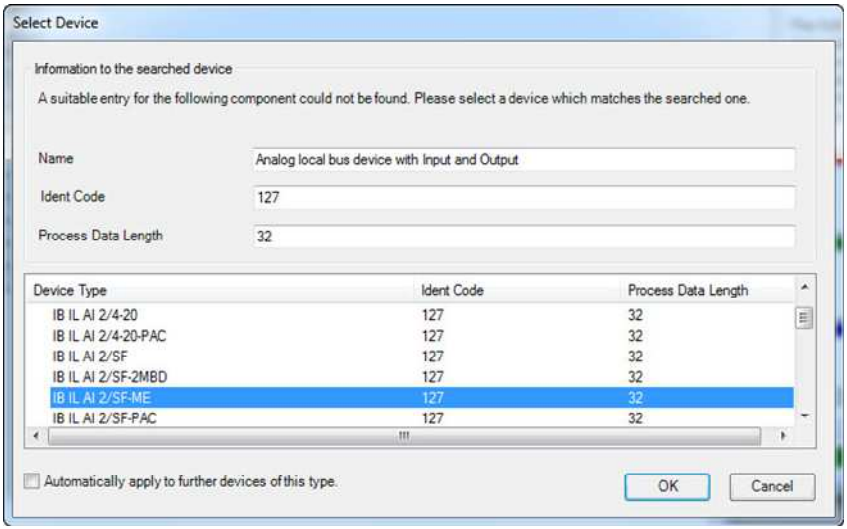


Рис. 16. Окно Select Device

Окно Device Catalog. Требуется для конфигурации магистральных систем в режиме оффлайн. Также необходимо для корректировки неправильно выбранных устройств в режиме онлайн.

1.2.5. Архитектура аппаратных средств

Операционная система. Операционная система ProConOS (**Pro**grammable **Con**troller **O**perating **S**ystem) применяется для обеспечения системных услуг ПЛК на специальных или стандартных аппаратных платформах. Они включают загрузку и обработку внешне создаваемых ПЛК-программ, а также обеспечение функций отладки при программировании, установке и обслуживании управляемых ПЛК машин и систем.

ProConOS используется в качестве операционной системы для большинства систем управления PhoenixContact, обеспечивая таким образом возможность параметризации и программирования систем управления в одной и той же среде разработки PC WorX.

Многозадачность. ProConOS основана на стандартных многозадачных операционных системах, которые управляют посредством приоритетов задачи. В ProConOS организованы три различных уровня приоритетов:

1) приоритетный уровень для пользовательских задач;
2) приоритетный уровень для задач ProConOS;
3) приоритетный уровень для системных задач.

Эта классификация гарантирует, что процессор всегда распределяет доступное ПЛК вычислительное время в пользу важных для приложения задач, и другие системные задачи получают необходимое компьютерное время, если позволяет тайминг. Использование многозадачности с разделением задач по времени в архитектуре системы ProConOS позволяет вычислить время отклика, оптимизировать производительность, т.е. уменьшить время отклика и реагировать на ошибки во время выполнения задачи (*runtime errors*).

Пользовательские задачи включают циклические задачи (*cyclic task*), событийные задачи (*event task*) и задачи по умолчанию (*default task*). В пределах установленного временного интервала циклические задачи выполняются циклически согласно приоритету, определяемому пользователем.

Событийные задачи – реакция на нециклически происходящие события, например, аппаратное прерывание.

Задача по умолчанию – это задача с самым низким приоритетом, выполняемая, если нет активных задач. Задача по умолчанию является циклической.

1.2.6. Каналы связи

В зависимости от типа применяемой системы управления доступны различные каналы связи. Для сетевой связи используется регистрация IP-адресов для систем управления, например, посредством службы BootP.

Последовательный интерфейс. Обеспечивает связь для соединения программатора и системы управления. Выбирается на вкладке <*Communication*> после открытия элемента <*Control System*> (В ILC131 Starterkit не используется).

Ethernet. В дополнение к последовательному интерфейсу большинство систем управления обеспечивает возможность организации связи через TCP/IP. Для установления связи по Ethernet системе управления должен быть назначен IP-адрес, который транслируется в PC WorX (рис. 17), который также выбирается на вкладке <Communication> после открытия элемента <Control System> (маркер 2 рис. 17). Нажатие кнопки <Test> активирует установление соединения, которое отображается в строке состояния (маркер 3 рис. 17). Кнопкой <Apply> определяется канал связи для текущей системы управления и текущего рабочего проекта. Используя эти настройки, другие окна получают доступ к системе управления.

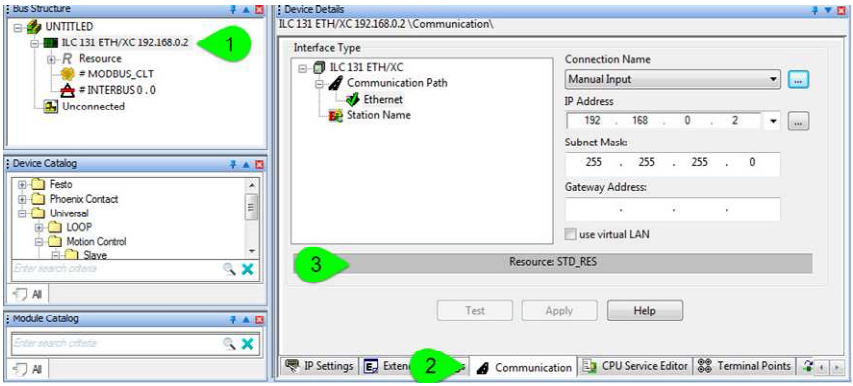


Рис. 17. Настройка связи по Ethernet

Назначение IP-адреса вручную. Для систем управления с часами реального времени IP-адрес назначается в окне на вкладке <*Extended Settings*> после выбора элемента <*Control System*>. После нажатия кнопки <Read> (маркер 4 рис. 18) в секторе <*Network Settings*> (маркер 1 рис. 18) отображается сетевая конфигурация системы управления. Требуется выбрать значение <*Manual definition of the TCP/IP setting*s> и ввести IP адрес и маску подсети (значение по умолчанию: использовать BootP сервер).

При необходимости возможна подстройка часов реального времени системы управления кнопкой <*System Time*> (маркер 3 рис. 18). Проверить соответствие внесенных изменений можно после рестарта контроллера (маркер 5 рис. 18). У систем управления с успешно сконфигурированной сетевой связью организуется сервер FTP, доступ к которому можно получить, нажав кнопку <*Open FTP Folder on Device*>. При этом отобразится содержимое карты CompactFlash, отведенное под FTP.

Назначение IP-адреса через BootP сервер. Если после запуска сервера BootP (например, PC WorX) системе управления требуется получить свой IP-адрес, то в настройках следует отметить опцию <*Usage of BootP server*> (маркер 2 рис. 18). После передачи нового статического IP-адреса новая конфигурация должна быть отправлена и затем проверена сбросом контроллера.

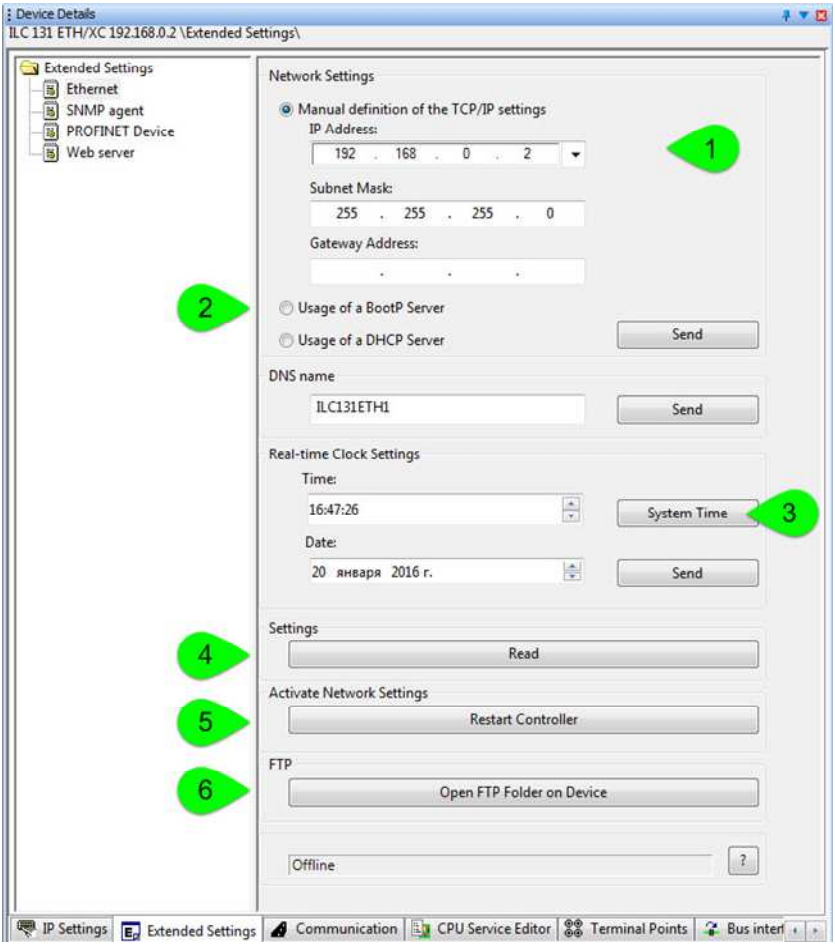


Рис. 18. Назначение IP-адреса

Для использования опции BootP для системы управления посредством интегрированного в PC WorX сервера BootP последний необходимо предварительно активировать, что можно сделать через меню по вкладке *Extras → BootP/SNMP/TFTP-Configuration*, как показано на рис. 19.

Для присвоения MAC-адреса системе управления, его можно ввести в ячейке <*MAC-address*> во вкладке <*IP-settings*>.

Выбор имеющегося сетевого соединения. Если контроллер был сконфигурирован с допустимым IP-адресом, этот адрес необходимо сделать доступным для PC WorX.

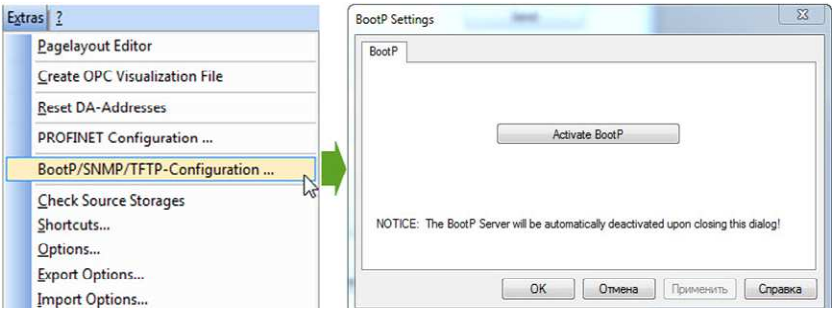


Рис. 19. Активация BootP сервера

Есть два способа выбрать канал связи TCP/IP (Ethernet/Localhost).

1. Выбрать элемент <*Manual Input*> и ввести требуемый IP-адрес, как изображено на рис. 17.

2. Выбрать станцию из выпадающего списка имен подключений в *Interface Type* → *Communication Path*. Именам станций поставлены в соответствие требуемые IP-адреса.

1.2.7. Шина INTERBUS

Шина распределенного ввода/вывода INTERBUS разрабатывалась как шина данных между датчиками и исполнительными механизмами для повышения производительности и сокращения расходов машин и установок. На сегодняшний день технология INTERBUS стандартизирована IEC 61158 и IEC 61784. Благодаря специальной технологии организации обмена данных и кольцевой топологии, система INTERBUS обеспечивает скоростную и циклическую передачу данных, проста в эксплуатации и настройке, имеет комплексные диагностические функции для минимизации времени простоя и высокую помехозащищенность. Именно за эти характеристики в сочетании с недорогой организацией связи между датчиками и исполнительными механизмами шина INTERBUS пользуется заслуженной популярностью. В ILC 131 Starterkit шина INTERBUS служит для связи ПЛК с широкой номенклатурой подключаемых к ПЛК модулей, а также связи между несколькими рабочими станциями (см. пример рис. 4).

Конфигурация INTERBUS в режиме онлайн. В случае, если готовая к работе система соединена с системой управления, может использоваться конфигурация шины INTERBUS в режиме онлайн (online).

Для вывода на экран конфигурации в режиме online используются вкладки меню *View* → *Connected INTERBUS* (рис. 15). Чтобы сохранить используемое в настоящий момент рабочее пространство (обычно рабочее пространство *Bus Configuration*) в доступном режиме, оно должно быть переключено в режим оффлайн (offline) и скрыто снова после ввода данных.

Для чтения шины INTERBUS необходимо выбрать выполняющуюся в текущий момент систему управления из списка <*Selected Control System*>. В этом случае PC WorX выведет сохраненный в системе управления фрейм конфигурации INTERBUS или, в случае, если система используется в состоянии *Ready*, создаст новый фрейм конфигурации.

Если система не обеспечивает готовность к работе или находится в неисправном состоянии, фрейм конфигурации не появится. В этом случае для диагностики следует использовать окно Diag+.

Добавление обнаруженных по шине INTERBUS устройств. В зависимости от типа подключенного в текущий момент устройства на экране в диалоге меню <*Select Device*> доступен список файлов-дескрипторов, описывающих устройства. Для выбора текущего устройства следует уточнить его название и комбинацию идентификационного кода и длины данных процесса (рисунок 16). Если выбор устройства был ошибочным, процесс выбора прерывать не следует, необходимо продолжить выбор с учетом ошибки. В большинстве случаев последующее исправление через каталог устройства займет меньше времени, чем повторный ввод данных в систему.

Конфигурация в режиме оффлайн. Для конфигурации INTERBUS в режиме offline должно быть активировано окно <*Device Catalog*> (<*View*> → <*Device Catalog*>). Для удобства предлагается расположить это окно между окнами <*Bus configuration*> и <*Device Details*>. Функция каталога относительно конфигурации шины ограничена тремя следующими действиями.

1. Добавление устройства на тот же самый уровень (удаленное устройство шины за удаленным устройством шины или устройство локальной шины за устройством локальной шины). Выполняется буксировкой мыши с зажатой левой клавишей выбранного устройства из <*Device* Catalog> в <Bus Structure>.

2. Добавление устройства на более низкий уровень (ответвление) (удаленное устройство шины в удаленной шине или устройство локальной шины позади терминала локальной шины).

3. Замена устройства (возможна, только если устройство на замену с аналогичным интерфейсом). Устройство на замену должно быть скопировано из <Device Catalog> через контекстное меню правой кнопки мыши и вставлено через операцию Replase в контекстном меню на заменяемом устройстве.

Шина PROFINET. PROFINET является открытым промышленным стандартом для автоматизации Ethernet PROFIBUS & PROFINET International (PI). PROFINET использует TCP/IP и IT стандарты и режим реального времени Ethernet. В ILC 131 Starterkit шина PROFINET не применяется.

1.3. Создание проекта в PC WorX

Создание проекта. Для создания заготовки проекта необходимо выбрать тип контроллера из пункта меню <*File*> → <*New_Project*> (рис. 20).

На стадии разработки проекта не принципиально, какой тип контроллера выбирается, т.к. в процессе адаптации программы разработчик может выбрать любой из доступных типов контроллеров. При этом необходимо иметь в виду, что контроллер можно поменять в любое время, и что режим симуляции возможен на архитектуре RFC_400 и S_MAX (тип контроллеров IPC) (см. табл. 12). Возможность работы в режиме симуляции позволяет на стадии проектирования отладить локальные алгоритмы и определиться с конечным типом контроллера с учетом потребностей в быстродействии и числом каналов ввода/вывода [2].

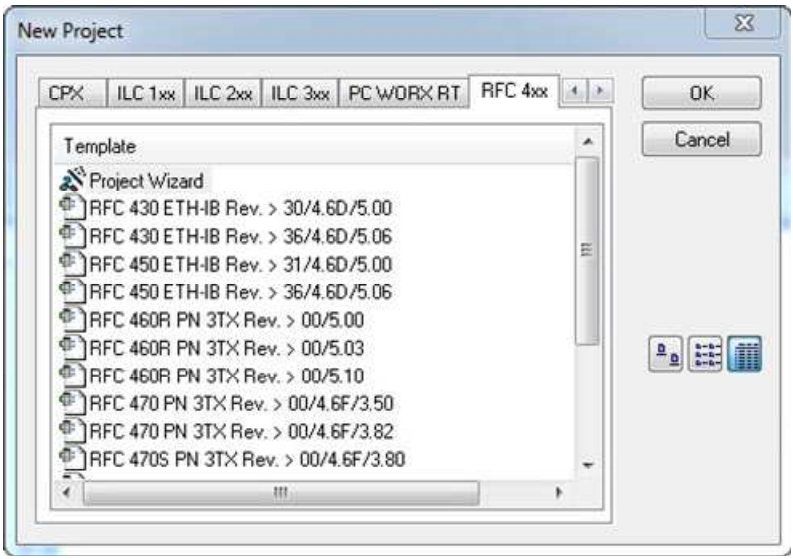


Рис. 20. Выбор типа контроллера при создании нового проекта

Проект можно создать, воспользовавшись подпунктом <*Project Wizard*> на любой вкладке меню <*File*> → <*New Project*>. В этом случае предстоит выбрать базовый язык программы, название проекта, тип процессора и тип контроллера. Для правильного выбора необходимо знать соответствие архитектуры ЦПУ типу контроллера, указанному в табл. 12. Каждый тип контроллера обладает определенными возможностями применения. Система PC WorX автоматически «соберет» конфигурацию ПЛК в соответствии с выбранным контроллером [2].

Следующим шагом станет уточнение информации проекта – выбор коммуникации контроллера в среде программирования PC WorX. Для этого в меню <*View*> необходимо поставить галочки: <*Bus Structure*> и <*Device Details*>, и в нижней части рабочей области появятся окна с соот-

ветствующими названиями. Окно в рабочей области <*Bus Structure*> позволяет выбирать конкретные устройства, которые присоединены к контроллеру по шине INTERBUS, а также сам контроллер [2].

Для работы программы в режиме эмуляции необходимо выделить мышкой тип выбранного контроллера (в примере – RFC 430), при этом в окне <*Device Details*> в закладке <*Communication*> в меню <*Serial port*> поставить галочку напротив режима <*Simulation*>, после выбрать пункт меню <*Simulation1*> и нажать кнопку <*Apply*>. При работе с реальным контроллером необходимо выбрать те коммуникации, которые используются для связи с ним (для ILC 131 Starterkit – Ethernet) [2].

В окне <*Device Details*> можно просмотреть текущие параметры, а также изменить конфигурацию сетевого контроллера, установить время, дату, открыть или запретить доступ к встроенным FTP и WWW серверам (если такая возможность имеется в конкретной модели контроллера). Для написания программы-примера эти опции неактуальны, поэтому их временно опустим [2].

В окне <*Bus Structure*> можно добавлять модули ввода/вывода, присоединенные к шине INTERBUS. По каждому присоединенному модулю можно также просмотреть его свойства (название модуля, его каталожный номер, типы и адреса переменных каналов ввода/вывода, оставить комментарии к ним, техническое описание этого модуля) [2].

Таблица 12

№ п/п	ЦПУ	Тип ПЛК	Возможность симуляции
1	M68_32	FC 200 PCL	нет
		ILC 200 IB	
2	IPC_40	RFC 430 ETH	да
		RFC 450 ETH	
		S-MAX	
3	IPC_32	RFC 430 ETH	
		RFC 450 ETH	
		S-MAX	
4	ARM_L_40	CP 3xx	нет
		FC 350 PCI	
		ILC 3xx	
5	ARM_L_32	FC 350 PCI	
		ILC 350 ETH	
		ILC 350 PN	
		ILC 370 PN	
6	eCLR	ILC 1xx	

Программирование. Для создания программы необходимо выбрать рабочее пространство IEC Programming Workspace. Для появления рабочего поля в окне <*Project Tree Window*> в папке <*Logical POUs*> необходимо выбрать рабочий лист <*Main*>.

Компиляция и отладка проекта. После создания программы в рабочем пространстве IEC Programming Workspace для обнаружения ошибок и предупреждений проект следует скомпилировать.

При первой компиляции проекта необходимо выбрать команду <*Rebuild Project*> в меню <*Build*>. Для последующих процессов компиляции используется команда <*Make*> в меню <*Build*>. *Е*сли при компиляции возникают ошибки, их следует исправить и повторить процесс компиляции, пока он не будет успешно завершен. Для успешной компиляции сообщения об ошибках должны быть удалены (предупреждения убирать не обязательно). Результаты процесса компиляции отображаются в окне сообщений с перечнем и расшифровкой ошибок и предупреждений.

Отладка проекта. Правильность функционирования программы можно отслеживать в режиме отладки. Для запуска режима отладки проект предварительно должен быть откомпилирован. Для перехода в режим отладки в IEC Programming Workspace в меню <*Online*> необходимо выбрать строку <*Debug*> (также вызов режима отладки возможен клавишей F10).

Программное состояние элементов POU в режиме отладки отображается в рабочем листе <*Main*>. Состояние локальных переменных, используемых в программе, отображается в листе переменных < *MainV*> в дереве проекта. Состояние глобальных переменных отображается на вкладке <*Global Variables*> дерева проекта.

Тема 2. АРИФМЕТИЧЕСКИЕ И ТРИГОНОМЕТРИЧЕСКИЕ ОПЕРАЦИИ

Базовые блоки, реализующие арифметические операции на языке Function Block Diagram в PC WorX, показаны на рисунке 21 и представляют собой операции сложения (ADD), вычитания (SUB), умножения (MUL) и деления (DIV) соответственно (рис. 21).

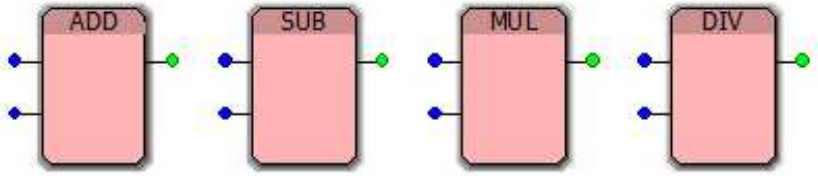


Рис. 21. Базовые блоки арифметических операций

По умолчанию реализованы элементы, имеющие два входа IN1 и IN2. Увеличение числа входных переменных возможно дублированием входа IN2 в свойствах элемента.

Важно понимать, что выполнение арифметических операций требует назначения одинакового типа данных всем входным переменным. Для назначения типа данных входным переменным требуется выбрать соответствующий тип в выпадающем меню «Data type» в окне свойств переменных (рис. 22).

По умолчанию входные переменные имеют начальное значение, равное нулю (независимо от выбранного типа данных). Поэтому для реализации операции деления (элемент DIV) во избежание ошибки, связанной с делением на ноль, требуется входу делителя IN2 присваивать начальное значение, отличное от нуля. Это также можно сделать в окне свойств переменных, задав числовое значение в поле данных «Initial value» (рис. 22).

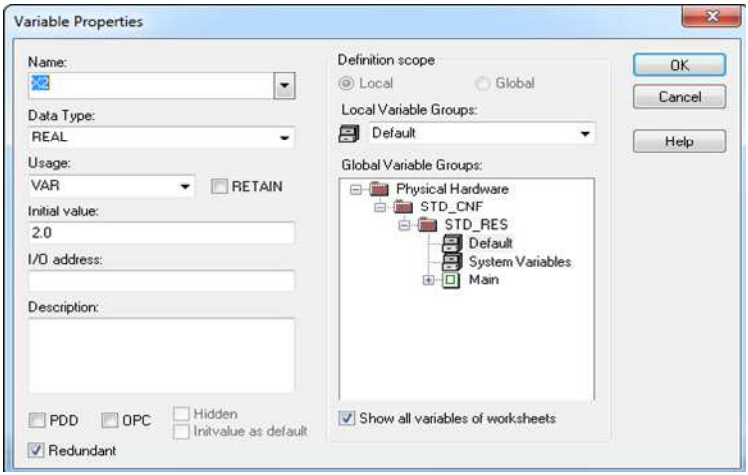


Рис. 22. Окно свойств переменных

Задание. Исследуйте базовые арифметические элементы ADD, SUB, MUL и DIV. Для этого переместите их на рабочее поле IEC Programming Workspace и задайте произвольные входные переменные. В режиме отладки присвойте этим переменным числовые значения и убедитесь в правильности выдачи результата. Пример описанной процедуры показан на рис. 23.

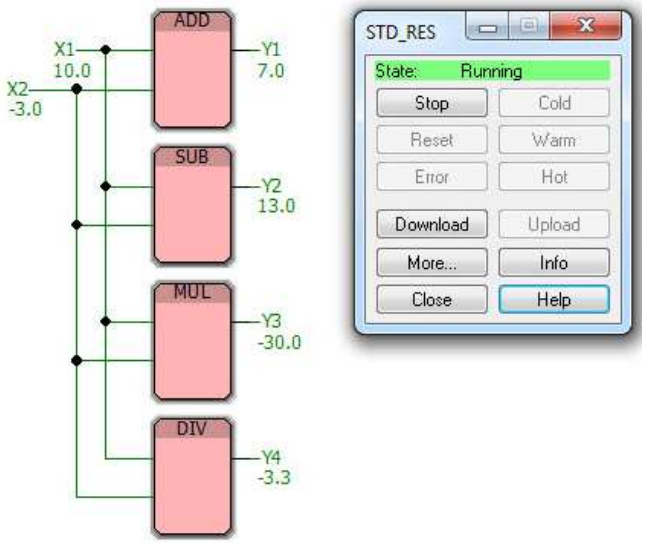


Рис. 23. Исследование базовых арифметических элементов ADD, SUB, MUL и DIV

Поработайте с разными типами данных. Заметим, что при арифметических операциях с десятичной точкой требуется выбрать целочисленный тип данных REAL. При этом по умолчанию число будет представлено в формате IEEE (в виде 1.2345678 ± 10^9). Для перевода в более удобный формат представления в окне ввода числовых данных для переменной (в режиме отладки) следует отказаться от формата IEEE, сняв соответствующую галочку (рис. 24). Также в поле «Precision» можно задать требуемое число знаков после запятой.

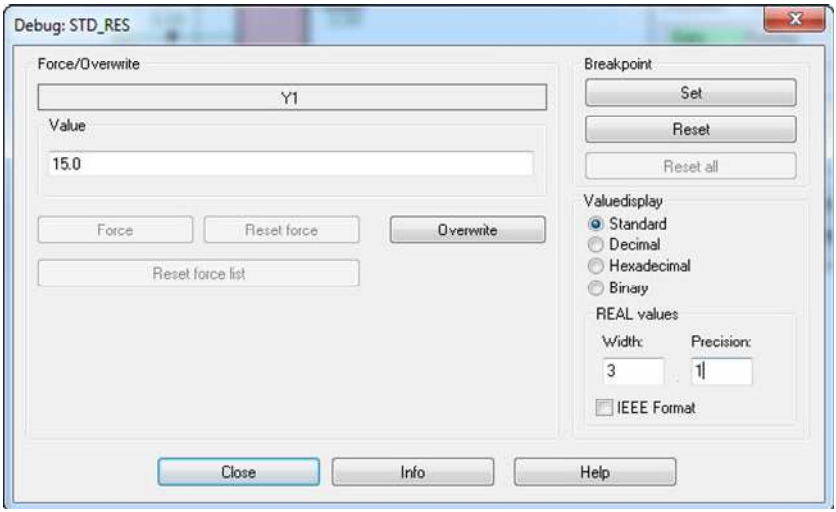


Рис. 24. Окно ввода числовых значений для переменной

2.1. Лабораторная работа «Арифметические выражения с тремя параметрами»

Цель лабораторной работы: научиться реализовывать в среде программирования PC WorX v.6.30 арифметические выражения с тремя параметрами, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки арифметических функций.

Порядок выполнения лабораторной работы

Необходимо разработать приложение на языке Function Block Diagram для реализации следующего арифметического выражения $y = \alpha x_1 + \beta x_2 + \gamma x_3$.

Таблица 13

Значения коэффициентов арифметического выражения

	I				II		
	α	β	γ		α	β	γ
	-18	-6	12		0,54	0,03	-0,18
	15	13	5		-0,13	-0,74	-0,65
	-15	1	-5		0,5	-0,15	-0,37
	7	6	-6		0,38	0,86	-0,09
	3	-9	-14		0,98	-0,38	-0,58
	-14	-1	6		-0,32	0,24	-0,28
	11	-8	2		0,56	0,78	-0,43
	-3	15	10		0,53	-0,63	-0,55
	4	1	-2		0,92	-0,65	0,92
	17	-9	10		0,01	0,92	0,63
	-6	17	-7		-0,4	-0,24	0,05
	-4	-14	-3		0,3	-0,04	0,27
	-15	6	7		0,79	-0,75	0,21
	20	8	-6		-0,07	-0,06	0,14
	4	-9	-16		0,83	0,77	-0,31
	14	-1	-2		0,03	-0,61	0,5
	-2	2	-8		0,54	-0,68	0,73
	10	-3	16		-0,16	0,65	-0,88
	19	-9	-2		0,3	-0,38	-0,38
	-13	-6	-6		-0,94	0,96	0,73
	16	14	17		-0,39	-0,6	0,4
	-14	5	8		0,43	-0,46	-0,26
	11	-14	5		0,05	-0,79	-0,16
	5	8	2		0,16	0,63	-0,78
	-15	12	-19		0,23	-0,6	0,78
	-3	-20	16		0,29	0,87	0,67
	8	-1	-19		-0,82	0,96	0,15
	8	-18	-16		-0,27	-0,88	-0,83
	9	-6	-11		0,31	-0,55	0,75
	-5	-2	14		-0,06	0,27	0,79

Заметим, что для вариантов группы I табл. 13 коэффициенты α, β, γ принадлежат множеству целых чисел, для вариантов группы II – множеству действительных чисел.

Составьте проверочную таблицу (табл. 14), состоящую из не менее 3 проверочных примеров, с помощью которой произведите отладку разработанной программы.

Таблица 14

Проверочная таблица

α	x_1	β	x_2	γ	x_3	y
…	…	…	…	…	…	…
…	…	…	…	…	…	…

Пример выполнения лабораторной работы 2.1.

Составим арифметическое выражение, соответствующее варианту № 30/I. Подставляем в арифметическое выражение $y = \alpha x_1 + \beta x_2 + \gamma x_3$ коэффициенты из таблицы 13, получаем $y = -5x_1 - 2x_2 + 14x_3$.

Пример реализации в PC WorX программы показан на рис. 25.

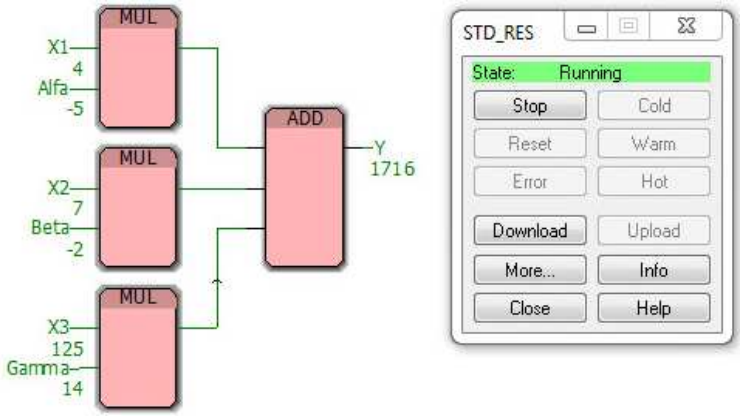


Рис. 25. Пример выполнения лабораторной работы 2.1, варианта 30/1

Проверим правильность выхода, выдаваемого программой: $y = -5x_1 - 2x_2 + 14x_3 = -5 \cdot 4 - 2 \cdot 7 + 14 \cdot 125 = 1716$, занесём информацию в проверочную таблицу (табл. 15).

Таблица 15

Фрагмент проверочной таблицы для выражения $y = -5x_1 - 2x_2 + 14x_3$

α	x_1	β	x_2	γ	x_3	y
-5	4	-2	7	14	125	1716
…						

На рисунке 26 представлена программа в режиме отладки для арифметического выражения, соответствующего варианту № 30/II. Подставив соответствующие коэффициенты из табл. 13, получаем выражение $y = -0{,}06x_1 + 0{,}27x_2 + 0{,}79x_3$, в соответствующую проверочную таблицу заносим информацию о нескольких тестовых примерах (табл. 16), например, $y = -0{,}06x_1 + 0{,}27x_2 + 0{,}79x_3 = -0{,}06 \cdot 1 + 0{,}27 \cdot (-2{,}01) + 0{,}79 \cdot (-0{,}85) = -1{,}2742$.

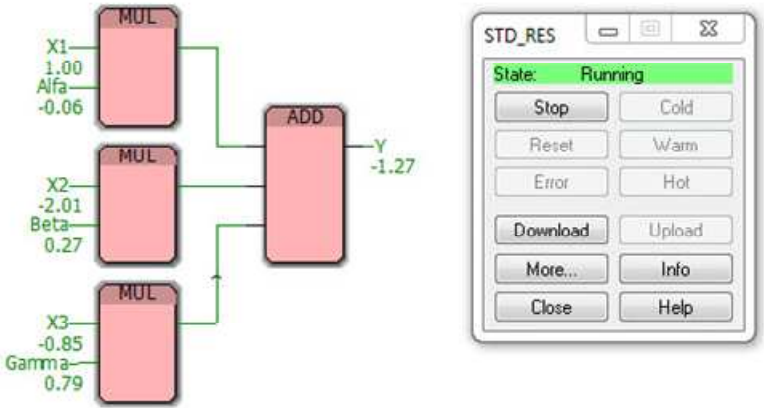


Рис. 26. Пример выполнения лабораторной работы 2.1, варианта 30/II

Таблица 16

Фрагмент проверочной таблицы для выражения
$y = -0{,}06x_1 + 0{,}27x_2 + 0{,}79x_3$

α	x_1	β	x_2	γ	x_3	y
-0,06	1	0,27	-2,01	0,79	-0,85	-1,2742
…						

2.2. Лабораторная работа «Арифметические выражения с четырьмя параметрами»

Цель лабораторной работы: научиться реализовывать в среде программирования PC WorX v.6.30 арифметические выражения с четырьмя параметрами, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки арифметических функций.

Порядок выполнения лабораторной работы

Необходимо разработать приложение на языке Function Block Diagram для реализации следующего арифметического выражения $y = x_4 + \alpha x_3 + \beta x_2 + \gamma x_1 + \delta$ (коэффициенты представлены в табл. 17).

Таблица 17

	I					II			
	α	β	γ	δ		α	β	γ	δ
1	2	3	4	5	6	7	8	9	10
	-7	-19	-5	-13		0,02	-0,43	0,23	-0,73
	4	-4	4	7		-0,98	0,64	0,66	-0,5
	9	-1	6	-18		-0,5	0,67	-0,25	0,18
	-13	19	-11	4		-0,76	0,19	0,09	-0,59
	-16	19	0	6		0,23	0,88	-0,81	0,13
	-9	15	17	9		-0,14	0,42	-0,35	0,67
	9	-12	6	-7		-0,54	0,79	-0,39	0,16
	9	-16	-16	12		0,92	0,28	-0,43	-0,26
	-15	16	-6	10		0,1	-0,44	0,15	0,8
	-4	-15	1	-11		-0,31	0,48	-0,79	0,2
	14	15	-17	15		-0,57	0,78	0,35	-0,97
	-20	-11	8	0		-0,67	0,23	-0,25	-0,68
	1	13	0	-16		-0,12	-0,44	0,79	0,17
	-14	5	-13	1		0,92	0,86	-0,81	-0,52
	-18	14	-6	1		-0,44	-0,03	0,06	0,34
	-11	-5	9	2		0,17	0,04	0,75	0,73
	-17	3	-9	-12		-0,71	0,96	-0,83	0,66
	-2	12	-11	4		-0,92	-0,5	0,75	-0,52

Окончание табл. 17

1	2	3	4	5	6	7	8	9	10
	-8	-20	13	-14		0,38	-0,82	0,48	0,62
	-11	9	0	-19		0,63	-0,29	0,06	0,47
	1	-5	-20	-6		0,51	-0,39	0,66	-0,67
	-7	-18	-9	9		0,35	-0,19	0,79	-0,38
	-17	12	-15	3		0,75	0,55	-0,1	0,17
	16	15	-10	0		0,74	-0,68	0,96	-0,97
	11	2	12	-7		0,39	0,02	-0,67	-0,32
	4	1	-3	12		0,52	-0,2	-1,01	-0,26
	14	6	1	19		0,62	0,73	-0,17	-0,64
	5	3	-3	5		0,01	0,2	-0,97	-0,26
	8	14	10	-11		0,72	0,28	-0,84	0,79
	-16	11	-15	5		0,58	-0,14	-0,04	0,17

Заметим, что для вариантов группы I таблицы 13 коэффициенты α, β, γ, δ принадлежат множеству целых чисел, для вариантов группы II – множеству действительных чисел.

Составьте проверочную таблицу (табл. 18), состоящую из не менее 3 проверочных примеров, с помощью которой произведите отладку разработанной программы.

Таблица 18

Проверочная таблица

x_4	α	x_3	β	x_2	γ	x_1	δ	y
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

Пример выполнения лабораторной работы 2.2

Рассмотрим выражение с коэффициентами из варианта № 30/I. Подставляем в арифметическое выражение $y = x_4 + \alpha x_3 + \beta x_2 + \gamma x_1 + \delta$ коэффициенты из табл. 17, получаем $y = x_4 - 16x_3 + 11x_2 - 15x_1 + 5$. Пример реализации указанного арифметического выражения показан на рис. 27.

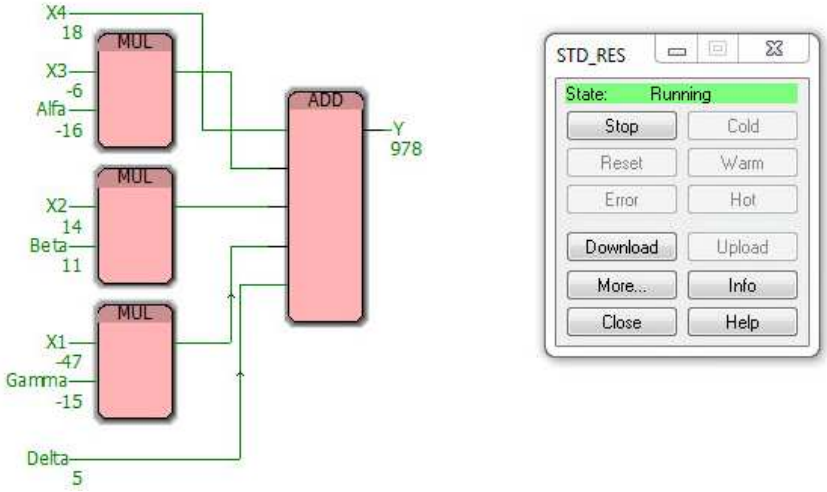


Рис. 27. Пример выполнения лабораторной работы 2.2. варианта 30/I

Проверим правильность выхода, выдаваемого программой: $y = x_4 - 16x_3 + 11x_2 - 15x_1 + 5 = 18 - 16 \cdot (-6) + 11 \cdot 14 - 15 \cdot (-47) + 5$, занесём информацию в проверочную таблицу (табл. 19).

Таблица 19

Фрагмент проверочной таблицы для выражения

$y = x_4 - 16x_3 + 11x_2 - 15x_1 + 5$

x_4	α	x_3	β	x_2	γ	x_1	δ	y
18	-16	-6	11	14	-15	-47	5	978
…								

Возьмём для арифметического выражения $y = x_4 + \alpha x_3 + \beta x_2 + \gamma x_1 + \delta$ коэффициенты группы II из табл. 17, получаем $y = x_4 + 0{,}58x_3 - 0{,}14x_2 - 0{,}04x_1 + 0{,}17$. Реализованная программа в режиме отладки показана на рис. 28.

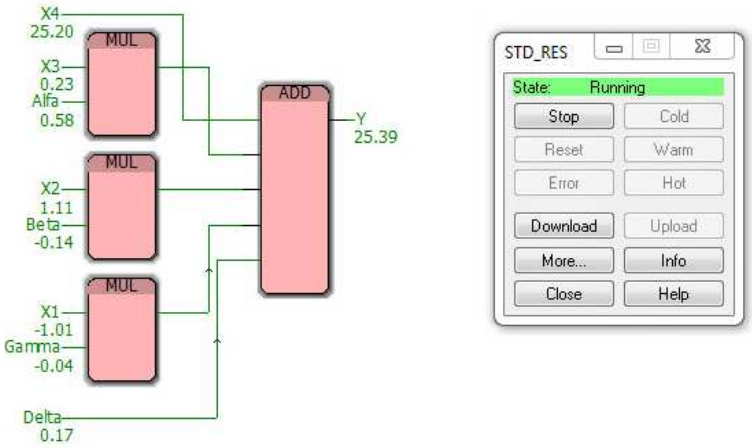


Рис. 28. Пример выполнения лабораторной работы 2.2. варианта 30/II

Проверим правильность выхода, выдаваемого программой:

$$y = x_4 + 0{,}58x_3 - 0{,}14x_2 - 0{,}04x_1 + 0{,}17 =$$
$$= 25{,}2 + 0{,}58 \cdot 0{,}23 + 0{,}14 \cdot 1{,}11 - 0{,}04 \cdot (-1{,}01) + 0{,}17 = 25{,}3884,$$

занесём информацию в проверочную таблицу (табл. 20).

Таблица 20

Фрагмент проверочной таблицы для выражения
$y = x_4 - 16x_3 + 11x_2 - 15x_1 + 5$

x_4	α	x_3	β	x_2	γ	x_1	δ	y
25,2	0,58	0,23	-0,14	1,11	-0,04	-1,01	0,17	25,3884
…								

2.3. Лабораторная работа «Дерево арифметического выражения»

Цель лабораторной работы: научиться в среде программирования PC WorX v.6.30 разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки арифметических функций, а также строить для арифметического выражения соответствующее дерево.

Порядок выполнения лабораторной работы

Для следующего арифметического выражения $a\,\alpha\,b\,\beta\,c\,\gamma\,d\,\delta\,e\,\varepsilon\,f$, где a, b, c, d, e, f – переменные, а α, β, γ, δ, ε – арифметические операции, заданные в табл. 21 (через • обозначение умножение, через / – деление), необходимо изобразить дерево и разработать приложение на языке Function Block Diagram. Для удобства можно расставить скобки.

Таблица 21

Символы операций арифметического выражения

№ п/п	α	β	γ	δ	ε
1	2	3	4	5	6
1	–	/	–	•	+
2	+	+	–	/	–
3	/	+	•	+	–
4	+	•	–	•	/
5	–	+	+	/	–
6	+	•	–	/	/
7	•	–	+	•	+
8	/	–	•	+	+
9	/	•	–	•	/
10	+	+	/	•	–
11	/	•	+	–	/
12	–	+	+	•	/
13	•	+	•	/	+
14	–	•	+	+	/
15	/	+	–	•	/
16	/	/	•	+	+
17	+	•	–	+	•
18	•	+	•	/	+
19	/	–	/	•	+
20	+	•	/	+	+
21	–	/	•	–	•
22	+	/	+	•	–
23	•	–	•	+	/
24	+	/	–	•	+
25	+	•	/	–	–
26	/	+	–	•	/

Окончание табл. 21

1	2	3	4	5	6
27	–	–	•	+	–
28	/	+	–	/	/
29	–	•	/	+	+
30	–	•	/	+	/

Составьте проверочную таблицу (табл. 22), состоящую из не менее 5 проверочных примеров, с помощью которой произведите отладку разработанной программы.

Таблица 22

Проверочная таблица

a	*b*	*c*	*d*	*e*	*f*	*y*
…	…	…	…	…	…	…
…	…	…	…	…	…	…

Пример выполнения лабораторной работы 2.3

Подставляем в арифметическое выражение $a\,\alpha\,b\,\beta\,c\,\gamma\,d\,\delta\,e\,\varepsilon\,f$ арифметические операции из табл. 21, получаем арифметическое выражение $a - b \bullet c / d + e / f$, поставим для удобства скобки следующим образом: $a - b \bullet (c / d + e / f)$, для этого арифметического выражение дерево представлено на рис. 29.

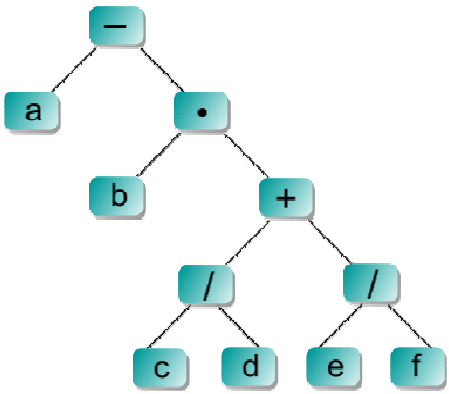


Рис. 29. Дерево арифметического выражения a – b • (c / d + e / f)

Реализованная программа для выражения $a - b \bullet (c / d + e / f)$ показана на рис. 30.

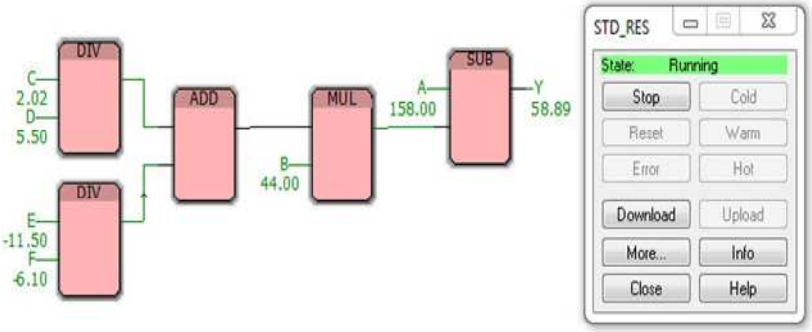


Рис. 30. Пример выполнения лабораторной работы 2.3

Покажем фрагмент проверочной таблицы (табл. 23) для примера из этого рисунка:

$$y = a - b \cdot (c \setminus d + e \setminus f) = 158 - 44 \cdot (2{,}02 \setminus 5{,}5 + (-11{,}5) \setminus (-6{,}1)) = 58{,}88918 .$$

Таблица 23

Фрагмент проверочной таблицы для выражения $y = a - b \cdot (c \setminus d + e \setminus f)$

a	*b*	*c*	*d*	*e*	*f*	*y*
158	44	2,02	5,5	-11,5	-6,1	58,88918
…	…	…	…	…	…	…

2.4. Лабораторная работа «Тригонометрические выражения»

Цель лабораторной работы: научиться реализовывать в среде программирования PC WorX v.6.30 тригонометрические выражения, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки тригонометрических функций.

Порядок выполнения лабораторной работы

Для следующего тригонометрического выражения $a\alpha(b x_1) + c\beta(d x_2)$, где *a*, *b*, *c*, *d* – числовые коэффициенты, а α, β – тригонометрические функции, заданные в табл. 24 реализуйте приложение

на языке Function Block Diagram. В PC WorX имеются тригонометрические функции, представленные на рис. 31, причем входное значение функций синус (SIN), косинус (COS) и тангенс (TAN) должно быть в радианах, а для функций арксинус (ASIN) и арккосинус (ACOS) область определения находится в диапазоне [-1; 1] (для арктангенса (ATAN) – любое число).

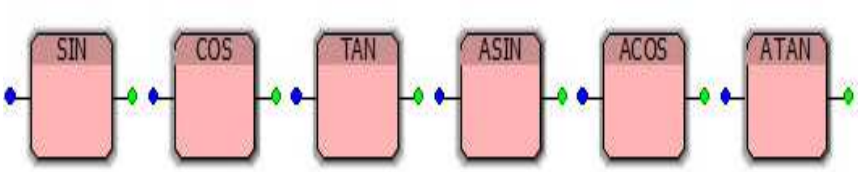


Рис. 31. Элементы, реализующие тригонометрические преобразования

На рисунке 32 продемонстрированы две программы, реализующие основные тригонометрические функции.

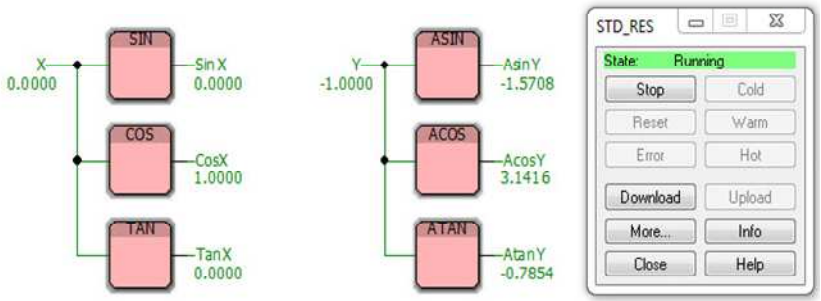


Рис. 32. Пример тригонометрических преобразований

Задание. Исследуйте вышеуказанные тригонометрические функции SIN, COS, TAN, ASIN, ACOS и ATAN. Для этого переместите их на рабочее поле IEC Programming Workspace, в режиме отладки задайте входные переменные из допустимых диапазонов и убедитесь в правильности выдачи результатов.

Таблица 24

Параметры тригонометрического выражения

№ п/п	*a*	α	*b*	*c*	β	*d*
1	5	tg	1	4	arctg	4
2	5	cos	6	4	sin	3
3	2	tg	6	4	arcsin	5
4	5	sin	4	2	tg	5
5	1	arctg	5	3	cos	2
6	5	arctg	6	1	sin	5
7	6	cos	3	5	arccos	4
8	2	tg	6	6	sin	2
9	2	sin	5	3	cos	5
10	5	arcsin	5	2	tg	4
11	4	cos	4	5	sin	1
12	5	sin	2	5	arctg	5
13	4	sin	3	3	cos	1
14	4	arccos	1	1	arcsin	3
15	6	cos	2	4	sin	4
16	2	sin	3	5	arccos	6
17	2	tg	1	5	cos	6
18	3	sin	4	1	sin	6
19	5	arcsin	4	1	cos	4
20	2	sin	4	1	arctg	6
21	3	cos	3	3	sin	2
22	5	arccos	4	6	sin	1
23	5	sin	3	6	tg	5
24	6	tg	1	5	cos	4
25	4	sin	1	3	cos	3
26	6	cos	2	5	tg	1
27	2	arcsin	6	1	sin	1
28	1	arccos	3	4	tg	3
29	3	sin	3	3	cos	3
30	6	sin	2	4	cos	5

Составьте проверочную таблицу (табл. 25), состоящую из не менее 5 проверочных примеров, с помощью которой произведите отладку разработанной программы.

Таблица 25

Проверочная таблица

a	α	b	x_1	c	β	d	x_2	y
6	sin	2	…	4	cos	5	…	…
…	…	…	…	…	…	…	…	…

Пример выполнения лабораторной работы 2.4

Реализуем в PC WorX программу для тригонометрического выражения варианта № 30. Для этого сначала подставим в тригонометрическое выражение $a\,\alpha\,(b\,x_1) + c\,\beta\,(d\,x_2)$ соответствующие варианту № 30 числовые коэффициенты и тригонометрические функции из табл. 24, получим тригонометрическое выражение $6\sin(2\,x_1) + 4\cos(5\,x_2)$. Представим полученное тригонометрическое выражение в виде программы, показанной на рис. 33.

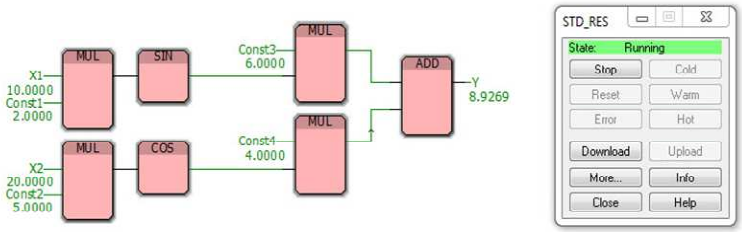


Рис. 33. Пример выполнения лабораторной работы 2.4

Проверим программу для тригонометрического выражения, указав результат проверочного примера в табл. 26:

$$y = 6\sin(2\,x_1) + 4\cos(5\,x_2) = 6\sin(2\cdot 10) + 4\cos(5\cdot 20) = 6\cdot 0{,}912945 + \\ + 4\cdot 0{,}862319 = 8{,}926947$$

Таблица 26

Фрагмент проверочной таблицы для выражения
$y = 6\sin(2\,x_1) + 4\cos(5\,x_2)$

a	α	b	x_1	c	β	d	x_2	y
6	sin	2	10	4	cos	5	20	8,926947
…	…	…	…	…	…	…	…	…

Тема 3. ЛОГИЧЕСКИЕ ВЫРАЖЕНИЯ

Базовые блоки, реализующие функции булевой алгебры (булевы функции, логические функции, переключательные функции) на языке Function Block Diagram в PC WorX показаны на рисунке 34 и представляют собой логические элементы конъюнкцию (логическое умножение) И (AND), дизъюнкцию (логическое сложение) ИЛИ (OR), сумму по модулю 2 (кольцевую сумму) «исключающее ИЛИ» (XOR), инверсию (отрицание) НЕ (NOT) слева направо соответственно.

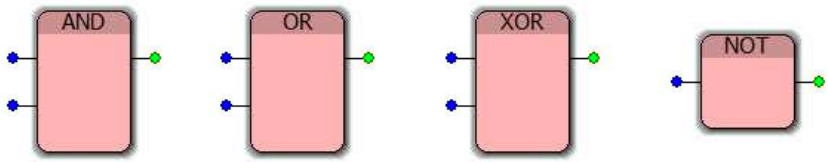


Рис. 34. Базовые блоки булевых функций

По умолчанию реализованы элементы, имеющие два входа (кроме элемента НЕ, соответствующего унарной операции инверсии). Увеличение числа входных переменных возможно дублированием входа IN2 в свойствах элемента, как показано на рис. 35. Для этого переместите выбранный элемент на рабочее поле IEC Programming Workspace, двойным нажатием на элемент левой кнопкой мыши войдите в свойства блока (Function/Function Block Properties) и, выбрав в параметрах вход IN2, нажмите кнопку «Duplicate».

Также можно реализовать инверсию (отрицание) по входу элемента, выбрав в свойствах этого элемента необходимый вход и выставив напротив него в поле «Negated» галочку (рис. 36).

Задание. Запишите таблицы истинности для каждого элемента и исследуйте перебором всех входных значений базовые логические элементы И, ИЛИ, «Исключающее ИЛИ», НЕ в режиме отладки программы.

Для этого переместите на рабочее поле IEC Programming Workspace необходимые логические элементы и двойным нажатием левой кнопкой мыши по входу или выходу элемента задайте его имя и тип (bool). После этого скомпилируйте проект нажатием кнопки «Rebuild Project» и загрузи-

те его в ПЛК нажатием кнопки «Download changes». Перейдите в режим отладки нажатием кнопки «Debug on/off». В режиме отладки выберите вход элемента и задайте его логическое значение TRUE или FALSE, как показано на рис. 37.

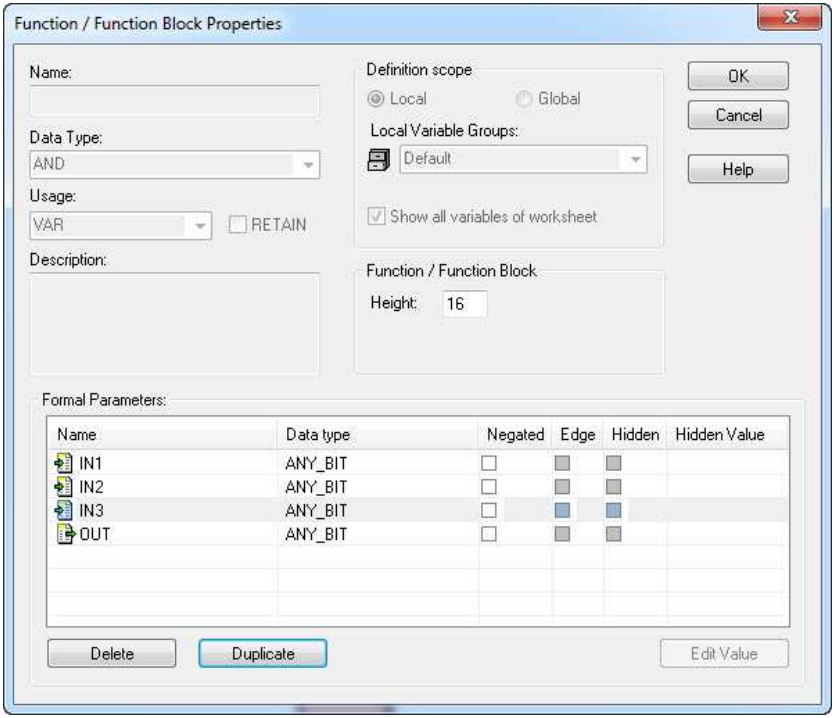


Рис. 35. Свойства логического элемента

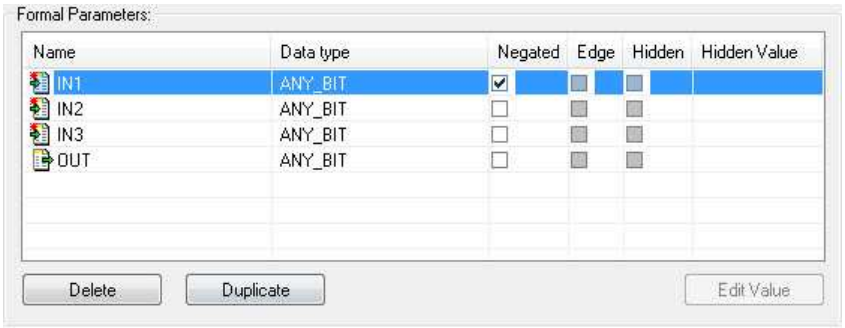


Рис. 36. Инверсия по входу IN1 логического элемента

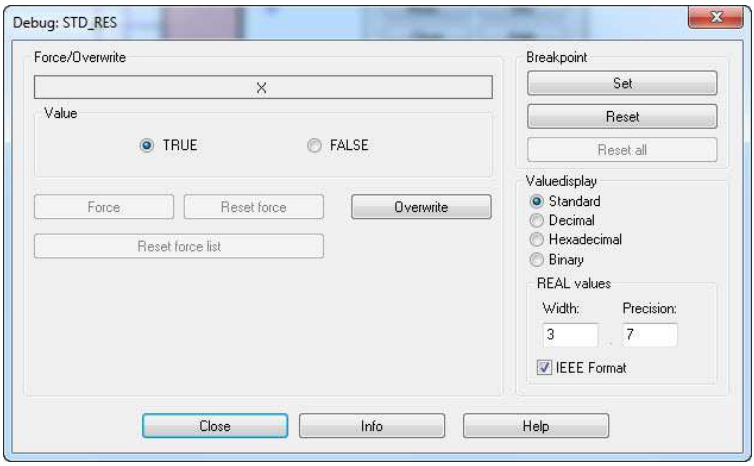


Рис. 37. Установка логического значения входа элемента

Полным перебором всех входных значений получите соответствующие им наборы выходных значений, проверьте правильность выдачи результата по соответствующей логическому элементу таблице истинности. Пример описанной процедуры показан на рис. 38.

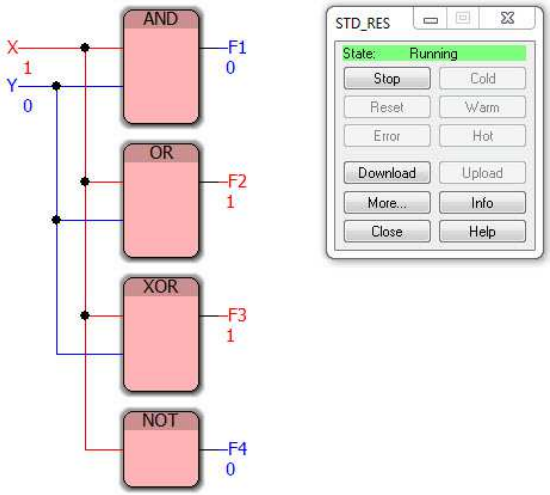


Рис. 38. Пример процедуры перебора входных значений логических элементов

3.1. Лабораторная работа «Совершенная дизъюнктивная нормальная форма функций булевой алгебры»

Цель лабораторной работы*:* научиться реализовывать в среде программирования PC WorX v.6.30 логические функции булевой алгебры в виде совершенной дизъюнктивной нормальной формы, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки логических функций.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX функцию булевой алгебры в виде совершенной дизъюнктивной нормальной формы (СДНФ) из табл. 27.

Таблица 27

Варианты функций булевой алгебры

№ п/п	Функция булевой алгебры
1	2
1	$F = \bar{x}\bar{y}z \vee x\bar{y}z \vee xy\bar{z} \vee xyz$
2	$F = \bar{x}yz \vee \bar{x}\bar{y}z \vee xy\bar{z} \vee xyz$
3	$F = \bar{x}yz \vee x\bar{y}z \vee \bar{x}y\bar{z} \vee \bar{x}\bar{y}\bar{z}$
4	$F = \bar{x}\bar{y}z \vee x\bar{y}z \vee xy\bar{z} \vee xyz$
5	$F = \bar{x}\bar{y}z \vee \bar{x}\bar{y}z \vee x\bar{y}z \vee xyz$
6	$F = \bar{x}\bar{y}z \vee \bar{x}\bar{y}z \vee xy\bar{z} \vee xyz$
7	$F = \bar{x}\bar{y}z \vee \bar{x}\bar{y}z \vee x\bar{y}z \vee xyz$
8	$F = \bar{x}\bar{y}z \vee x\bar{y}z \vee \bar{x}\bar{y}z \vee xyz$
9	$F = \bar{x}\bar{y}z \vee x\bar{y}z \vee \bar{x}\bar{y}z \vee xyz$
10	$F = \bar{x}\bar{y}z \vee x\bar{y}z \vee \bar{x}\bar{y}z \vee xyz$
11	$F = \bar{x}\bar{y}z \vee xyz \vee x\bar{y}z \vee xyz$ 12
12	$F = \bar{x}\bar{y}z \vee xyz \vee \bar{x}\bar{y}z \vee xyz$

Окончание табл. 27

1	2
13	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee \overline{x}yz$
14	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee x\overline{y}z$
15	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee \overline{xyz}$
16	$F = \overline{xy}z \vee \overline{x}yz \vee \overline{\overline{x}}yz \vee \overline{\overline{x}}yz$
17	$F = \overline{xy}z \vee \overline{x}yz \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee \overline{xyz}$
18	$F = \overline{xy}z \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee xyz \vee \overline{xyz}$
19	$F = \overline{xy}z \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee xyz \vee \overline{xyz}$
20	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee xyz \vee \overline{xyz}$
21	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee \overline{xyz}$
22	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee \overline{xyz}$
23	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee \overline{xyz}$
24	$F = \overline{\overline{x}}yz \vee \overline{x}\overline{y}z \vee \overline{\overline{x}}yz \vee \overline{xyz} \vee \overline{xyz}$
25	$F = \overline{xy}z \vee \overline{x}yz \vee \overline{xy}z \vee xyz$
26	$F = \overline{\overline{x}}yz \vee \overline{x}yz \vee \overline{xy}z \vee xyz$
27	$F = \overline{xy}z \vee \overline{\overline{x}}yz \vee \overline{xy}z \vee xyz$
28	$F = \overline{xy}z \vee \overline{x}yz \vee \overline{\overline{x}}yz \vee xyz$
29	$F = \overline{xy}z \vee \overline{x}yz \vee \overline{xy}z \vee \overline{xyz}$
30	$F = \overline{xy}z \vee \overline{x}yz \vee \overline{xy}z \vee \overline{x}\overline{y}z \vee \overline{xyz}$

Для каждой функции булевой алгебры необходимо составить таблицу истинности и с её помощью проверить составленную в PC WorX программу в режиме отладки.

Пример выполнения лабораторной работы 3.1

Задание выполняется в два этапа. На первом этапе необходимо представить элементарные конъюнкции СДНФ функции F в виде отдельных логических элементов И, для которых задается необходимое число входов и, при необходимости, ставится инверсия требуемых входов. На втором этапе выходы конъюнктивных фрагментов являются входами дизъюнктивного элемента ИЛИ. Полным перебором всех входных значений получите соответствующий им набор выходных значений. Пример реализации функции для варианта № 30 представлен на рис. 39.

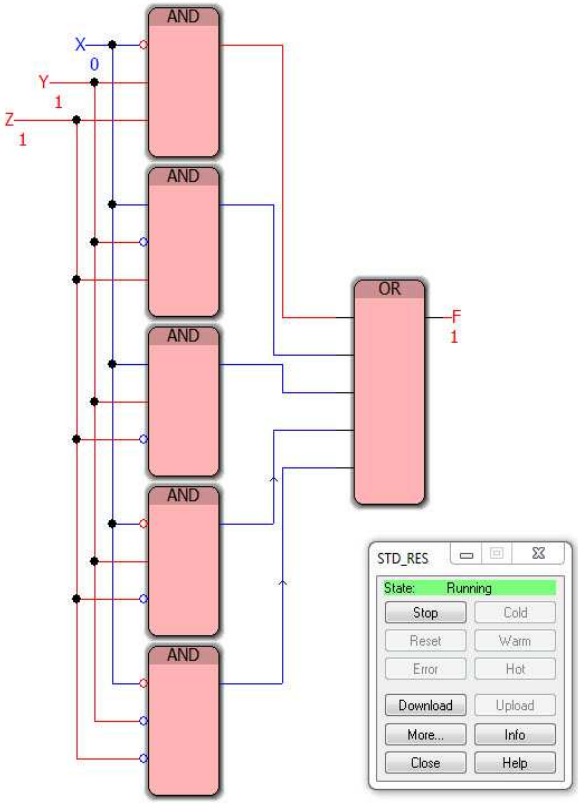


Рис. 39. Пример выполнения лабораторной работы 3.1

Проверим составленную программу. Для этого составим для функции булевой алгебры $F = \bar{x}yz \vee x\bar{y}z \vee xy\bar{z} \vee \bar{x}y\bar{z} \vee \bar{x}\bar{y}\bar{z}$ таблицу истинности (табл. 28).

Таблица 28

Таблица истинности для $F = \bar{x}yz \vee x\bar{y}z \vee xy\bar{z} \vee \bar{x}\bar{y}z \vee \bar{x}\bar{y}\bar{z}$

x	y	z	$\bar{x}$	$\bar{y}$	$\bar{z}$	$\bar{x} \wedge y$	$\bar{x} \wedge y \wedge z$	$x \wedge \bar{y}$	$x \wedge \bar{y} \wedge z$
0	0	0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0
0	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
1	0	0	0	1	1	0	0	1	0
1	0	1	0	1	0	0	0	1	1
1	1	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0

Продолжение табл. 28

x	y	z	$x \wedge y$	$x \wedge y \wedge \bar{z}$	$\bar{x} \wedge y \wedge \bar{z}$	$\bar{x} \wedge \bar{y}$	$\bar{x} \wedge \bar{y} \wedge \bar{z}$
0	0	0	0	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	**1**	**1**	**0**	**0**	**0**	**0**	**0**
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	1	0	0	0
1	1	1	1	0	0	0	0

Окончание таблицы 28

x	y	z	$\bar{x} \wedge y \wedge z \vee x \wedge \bar{y} \wedge z$	$x \wedge y \wedge \bar{z} \vee \bar{x} \wedge y \wedge \bar{z}$	F
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	1	1
0	**1**	**1**	**1**	**0**	**1**
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	0	0

Обратим внимание, что на рис. 39 представлен набор входов (0, 1, 1), согласно таблице истинности (строка 4) значение функции *F* должно быть равным 1 (табл. 28), такой выход и получили (см. рис. 39). Аналогично проверяются и значения функции булевой алгебры на остальных наборах.

3.2. Лабораторная работа «Совершенная конъюнктивная нормальная форма функции булевой алгебры»

Цель лабораторной работы*:* научиться реализовывать в среде программирования PC WorX v.6.30 логические функции булевой алгебры в виде совершенной конъюнктивной нормальной формы, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки логических функций.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX функцию булевой алгебры в виде совершенной конъюнктивной нормальной формы (СКНФ) из табл. 29.

Таблица 29

Варианты функций булевой алгебры

№ п/п	Функция булевой алгебры
1	2
1	$F = \left(\bar{x} \vee \bar{y} \vee z\right) \wedge \left(\bar{x} \vee y \vee \bar{z}\right) \wedge \left(x \vee \bar{y} \vee \bar{z}\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(\bar{x} \vee \bar{y} \vee \bar{z}\right)$
2	$F = \left(\bar{x} \vee \bar{y} \vee z\right) \wedge \left(\bar{x} \vee y \vee \bar{z}\right) \wedge \left(x \vee \bar{y} \vee \bar{z}\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(\bar{x} \vee \bar{y} \vee \bar{z}\right)$
3	$F = \left(\bar{x} \vee \bar{y} \vee z\right) \wedge \left(\bar{x} \vee y \vee \bar{z}\right) \wedge \left(x \vee \bar{y} \vee \bar{z}\right) \wedge \left(\bar{x} \vee \bar{y} \vee \bar{z}\right) \wedge \left(x \vee y \vee \bar{z}\right)$
4	$F = \left(\bar{x} \vee y \vee z\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(\bar{x} \vee y \vee \bar{z}\right) \wedge \left(\bar{x} \vee \bar{y} \vee \bar{z}\right)$
5	$F = \left(\bar{x} \vee y \vee z\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(x \vee y \vee z\right)$
6	$F = \left(\bar{x} \vee \bar{y} \vee z\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(x \vee y \vee z\right)$
7	$F = \left(\bar{x} \vee y \vee z\right) \wedge \left(\bar{x} \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(x \vee y \vee z\right)$
8	$F = \left(\bar{x} \vee y \vee z\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(\bar{x} \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee z\right)$
9	$F = \left(\bar{x} \vee y \vee z\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(\bar{x} \vee \bar{y} \vee \bar{z}\right)$
10	$F = \left(\bar{x} \vee y \vee \bar{z}\right) \wedge \left(x \vee \bar{y} \vee z\right) \wedge \left(x \vee y \vee \bar{z}\right) \wedge \left(x \vee y \vee z\right)$

Окончание табл. 29

1	2
11	$F = (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee y \vee \bar{z}) \wedge (x \vee y \vee z)$
12	$F = (\bar{x} \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee y \vee \bar{z})$
13	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee \bar{z}) \wedge (x \vee y \vee z)$
14	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee z) \wedge (x \vee y \vee z)$
15	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z)$
16	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee \bar{z}) \wedge (x \vee y \vee z)$
17	$F = (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (x \vee y \vee z)$
18	$F = (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z)$
19	$F = (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee y \vee \bar{z}) \wedge (x \vee y \vee z)$
20	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee z)$
21	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee z)$
22	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee \bar{y} \vee z)$
23	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee \bar{z})$
24	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
25	$F = (\bar{x} \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
26	$F = (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
27	$F = (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee y \vee \bar{z}) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
28	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
29	$F = (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
30	$F = (x \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z)$

Для каждой функции булевой алгебры необходимо составить таблицу истинности и с её помощью проверить составленную в PC WorX программу в режиме отладки.

Пример выполнения лабораторной работы 3.2

На первом этапе необходимо представить элементарные дизъюнкции СКНФ функции *F* в виде отдельных логических элементов ИЛИ, для которых задается необходимое число входов и, при необходимости, ставится инверсия требуемых входов. На втором этапе выходы дизъюнктивных фрагментов становятся входами конъюнктивного элемента И. Полным перебором всех входных значений получите соответствующий им набор выходных значений. Пример реализации функции для варианта № 30 представлен на рис. 40.

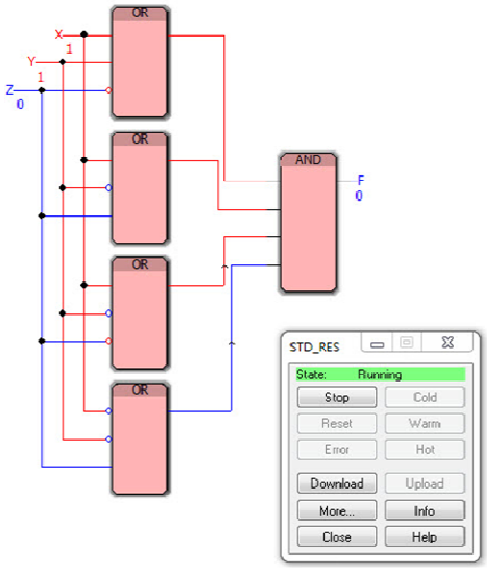


Рис. 40. Пример выполнения лабораторной работы 3.2

Проверим составленную программу. Для этого составим для функции булевой алгебры $F=\left(x\vee y\vee\overline{z}\right)\wedge\left(x\vee\overline{y}\vee z\right)\wedge\left(x\vee\overline{y}\vee\overline{z}\right)\wedge\left(\overline{x}\vee\overline{y}\vee z\right)$ таблицу истинности (табл. 30).

Таблица 30

Таблица истинности
для $F = (x \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z)$

x	y	z	$\bar{x}$	$\bar{y}$	$\bar{z}$	$(x \vee y \vee \bar{z})$ *	$(x \vee \bar{y} \vee z)$ **	$(x \vee \bar{y} \vee \bar{z})$ ***
0	0	0	1	1	1	1	1	1
0	0	1	1	1	0	0	1	1
0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	1	1	0
1	0	0	0	1	1	1	1	1
1	0	1	0	1	0	1	1	1
1	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
1	1	1	0	0	0	1	1	1

Окончание таблицы 30

x	y	z	$(\bar{x} \vee \bar{y} \vee z)$ ****	* ∧ **	*** ∧ ****	F
0	0	0	1	1	1	1
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	1	1	0	0
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	**1**	**0**	**0**	**1**	**0**	**0**
1	1	1	1	1	1	1

Набор, указанный в строке 7 таблицы истинности (табл. 30), соответствует входам, заданным на рис. 40, при этом видно, что выход, равный 0, в программе (рис. 40) совпадает со значением функции *F* (табл. 30). Аналогично проверяются остальные семь наборов.

3.3. Лабораторная работа «Сложные функции булевой алгебры»

Цель лабораторной работы: научиться разрабатывать приложения на языке программирования Function Block Diagram сложных функций булевой алгебры.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX функцию булевой алгебры из табл. 31.

Таблица 31

Варианты формул булевой алгебры

№ п/п	Формула булевой алгебры
1	2
1	$\bar{x}\bar{y}z \vee \overline{x \vee \bar{y}} \vee \overline{x \vee \bar{y} \vee z} \vee \bar{x}yz$
2	$\bar{x}yz \vee \overline{x \vee \bar{y}}$
3	$xyz \vee \bar{y}z \vee \bar{x}z \vee \overline{x \vee y \vee z}$
4	$\bar{x}yz \vee \overline{\bar{x} \vee y} \vee xyz \vee \overline{\bar{x} \vee \bar{y} \vee z}$
5	$x\bar{y}z \vee yz \vee \bar{x}y \vee \overline{x \vee \bar{y}}$
6	$\bar{x}yz \vee \overline{x \vee \bar{y}} \vee z \vee yz \vee \bar{x}y\bar{z}$
7	$\bar{x}\bar{y}z \vee \bar{x}y \vee yz \vee \overline{x \vee y \vee z}$
8	$\bar{y}z \vee \bar{x}yz \vee \overline{x \vee y \vee z} \vee xyz$
9	$\bar{x}z \vee \bar{x}\bar{y}z \vee \overline{x \vee y \vee z} \vee \bar{x}y\bar{z} \vee \bar{y}z$
10	$\bar{x}\bar{y}z \vee \overline{y \vee \bar{z}} \vee \bar{x}yz \vee \overline{x \vee y \vee \bar{z}}$
11	$\bar{x}yz \vee \bar{x}\bar{y}z \vee \overline{\bar{y} \vee z} \vee \bar{x}yz$
12	$xyz \vee \bar{x}z \vee \bar{x}y \vee \overline{x \vee y \vee z}$
13	$\bar{x}yz \vee \overline{\bar{y} \vee z} \vee xyz \vee \vee \overline{x \vee \bar{y} \vee \bar{z}}$

Окончание табл. 31

1	2
14	$\bar{x}yz \vee xz \vee y\bar{z} \vee \overline{y \vee z}$
15	$\bar{x}yz \vee \overline{x \vee y \vee z} \vee \bar{x}z \vee \bar{x}\bar{y}z$
16	$\overline{x \vee y \vee z} \vee \bar{x}\bar{y}z \vee xyz \vee \overline{x \vee y \vee z}$
17	$\bar{x}\bar{z} \vee \bar{x}yz \vee xyz \vee \overline{x \vee y \vee z}$
18	$\bar{x}\bar{y}\bar{z} \vee \vee x \vee z \vee xyz \vee \overline{\overline{x \vee y \vee z}}$
19	$\bar{x}y \vee \bar{x}\bar{y}z \vee \bar{x}\bar{y}z \vee \bar{x}z \vee \overline{x \vee y \vee z}$
20	$\bar{x}yz \vee \overline{x \vee z} \vee xy\bar{z} \vee \overline{x \vee y \vee z}$
21	$\bar{x}yz \vee \bar{x}yz \vee \overline{x \vee y \vee z} \vee xy\bar{z}$
22	$xyz \vee \bar{x}y \vee \overline{x \vee y \vee z} \vee y\bar{z}$
23	$\bar{x}yz \vee \overline{x \vee z} \vee \overline{x \vee y \vee z} \vee xyz$
24	$\bar{x}yz \vee xy \vee \overline{x \vee z} \vee \bar{x}z$
25	$\bar{x}yz \vee \overline{x \vee y \vee z} \vee x\bar{y} \vee x\bar{y}z$
26	$xy\bar{z} \vee yz \vee \bar{x} \vee \overline{x \vee y \vee z}$
27	$\bar{x}yz \vee \bar{x}y \vee \overline{x \vee y \vee z} \vee xyz$
28	$\bar{x}\bar{y}\bar{z} \vee \overline{x \vee y} \vee \bar{x}\bar{y}z \vee \overline{x \vee y \vee z}$
29	$x \wedge (\bar{y} \vee x \wedge z)$
30	$\overline{x \vee \bar{y} \wedge z} \vee \bar{x} \vee y \wedge z$

Для каждой функции булевой алгебры необходимо составить таблицу истинности и с её помощью проверить составленную в PC WorX программу в режиме отладки.

Пример выполнения лабораторной работы 3.3

Составим программу в PC WorX, соответствующую формуле варианта № 29 табл.31 (рис. 41).

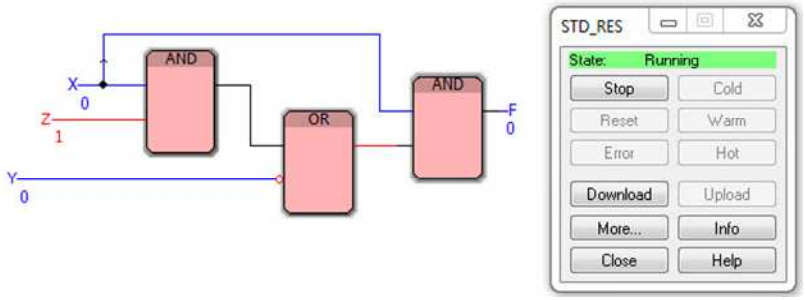


Рис. 41. Пример выполнения варианта № 29 лабораторной работы 3.3

Для проверки правильности реализации программы построим таблицу истинности (табл. 32) формулы $x \wedge (\overline{y} \vee x \wedge z)$, при этом особое внимание обратим на приоритет выполнения операций, в частности, на то, что логическая операция «конъюнкция» $\wedge$ имеет приоритет над логической операцией «дизъюнкция» $\vee$, а, следовательно, должна определяться перед дизъюнкцией.

Таблица 32

Таблица истинности для $F = x \wedge (\overline{y} \vee x \wedge z)$

x	y	z	$\overline{y}$	$x \wedge z$	$\overline{y} \vee x \wedge z$	$x \wedge (\overline{y} \vee x \wedge z)$
0	0	0	1	0	1	0
0	**0**	**1**	**1**	**0**	**1**	**0**
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	1	1

Последовательное применение всевозможных наборов в качестве входов схемы подтверждает правильность её составления, в частности, набор (0, 0, 1), соответствующий 2 строке таблицы истинности (табл. 32), имеет значение функции булевой алгебры $x \wedge (\overline{y} \vee x \wedge z)$, равное 0, это же значение показывает выход на рис. 41.

Пример выполнения № 2

Составим программу в PC WorX, соответствующую формуле варианта № 30 табл. 31 (рис. 42).

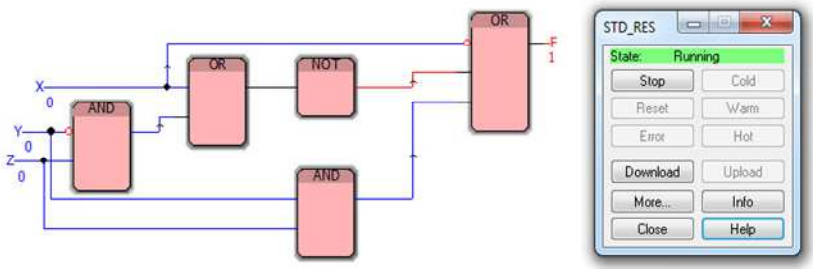


Рис. 42. Реализация тестового примера булевой алгебры для практического задания № 4 вариант № 30/II

Для проверки правильности реализации программы построим таблицу истинности (табл. 33) формулы $\overline{x \vee \overline{y} \wedge z} \vee \overline{x} \vee y \wedge z$, учитывая приоритет выполнения операций.

Таблица 33

Таблица истинности для $F = \overline{x \vee \overline{y} \wedge z} \vee \overline{x} \vee y \wedge z$

x	y	z	$\overline{x}$	$\overline{y}$	$\overline{y} \wedge z$	$x \vee \overline{y} \wedge z$	$\overline{x \vee \overline{y} \wedge z}$	$y \wedge z$
0	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**
0	0	1	1	1	1	1	0	0
0	1	0	1	0	0	0	1	0
0	1	1	1	0	0	0	1	1
1	0	0	0	1	0	1	0	0
1	0	1	0	1	1	1	0	0
1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	1	0	1

Окончание табл. 33

x	y	z	$\overline{x \vee \bar{y} \wedge z} \vee \bar{x} \vee y \wedge z$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Последовательное применение всевозможных наборов в качестве входов схемы подтверждает правильность её составления, в частности, набор (0, 0, 0), соответствующий 1 строке таблицы истинности (табл. 33), имеет значение функции булевой алгебры $\overline{x \vee \bar{y} \wedge z} \vee \bar{x} \vee y \wedge z$, равное 1, это же значение показывает выход на рис. 42.

3.4. Лабораторная работа «Сокращённая дизъюнктивная нормальная форма для заданных вектором значений функций булевой алгебры»

Цель лабораторной работы*:* научиться реализовывать в среде программирования PC WorX v.6.30 логические функции булевой алгебры в виде совершенной дизъюнктивной нормальной формы, заданной вектором значений булевой функции, а также разрабатывать соответствующие приложения на языке программирования Function Block Diagram.

Порядок выполнения лабораторной работы

Для функций булевой алгебры (табл. 34), заданных вектором значений, запишите СДНФ (СКНФ), определите сокращённую дизъюнктивную нормальную форму (сокр. ДНФ). Реализуйте в PC WorX СДНФ (СКНФ) и сокр. ДНФ функции булевой алгебры *F*.

Таблица 34

Варианты функций булевой алгебры

№ п/п	Функция булевой алгебры	№ п/п	Функция булевой алгебры
1	*F* = (10110011)	16	*F* = (01111001)
2	*F* = (01101101)	17	*F* = (10110101)
3	*F* = (11010011)	18	*F* = (01010101)
4	*F* = (11101010)	19	*F* = (11000101)
5	*F* = (10110110)	20	*F* = (01111000)
6	*F* = (10001010)	21	*F* = (11100110)
7	*F* = (10010011)	22	*F* = (10101001)
8	*F* = (01010110)	23	*F* = (00011110)
9	*F* = (00101110)	24	*F* = (11010100)
10	*F* = (10001101)	25	*F* = (10010010)
11	*F* = (01010110)	26	*F* = (01001001)
12	*F* = (10100100)	27	*F* = (00101100)
13	*F* = (00001101)	28	*F* = (11001000)
14	*F* = (00010101)	29	*F* = (10110000)
15	*F* = (01010010)	30	*F* = (01100100)

Проверьте составленную в PC WorX программу в режиме отладки.

Пример выполнения лабораторной работы 3.4

Решение варианта № 30. Таблица истинности, соответствующая функции булевой алгебры *F* = (01100100), представляет собой таблицу следующего вида (табл. 35).

Таблица 35

Таблица истинности F = (01100100)

№ п/п	*x*	*y*	*z*	*F*(*x*, *y*, *z*)
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	0
5	1	0	0	0
6	**1**	**0**	**1**	**1**
7	1	1	0	0
8	1	1	1	0

Для составления формулы в виде СДНФ для функции булевой алгебры, не соответствующей тождественно ложной (противоречию) булевой функции, необходимо выбрать выражения, соответствующие строкам, где функции булевой алгебры истинна, и соединить их операцией $\vee$, при этом если в какой-либо строке логическая переменная имеет ложное значение, то в соответствующем выражении она используется с инверсией (в табл. 36 показаны всевозможные «навешивания» инверсии над логическими переменными для построения СДНФ).

Таблица 36

Таблица истинности для СДНФ

№ п/п	x	y	z	
1	0	0	0	$\bar{x} \wedge \bar{y} \wedge \bar{z}$
2	**0**	**0**	**1**	$\bar{x} \wedge \bar{y} \wedge z$
3	**0**	**1**	**0**	$\bar{x} \wedge y \wedge \bar{z}$
4	0	1	1	$\bar{x} \wedge y \wedge z$
5	1	0	0	$x \wedge \bar{y} \wedge \bar{z}$
6	**1**	**0**	**1**	$x \wedge \bar{y} \wedge z$
7	1	1	0	$x \wedge y \wedge \bar{z}$
8	1	1	1	$x \wedge y \wedge z$

Составим СДНФ функции булевой алгебры F, заданной вектором значений (0**11**00**1**00): для этого обратим внимание, что эта функция истинна (содержит единицы) в строках под номерами 2, 3 и 6, поэтому выражения из таблицы 36 $\bar{x} \wedge \bar{y} \wedge z$, $\bar{x} \wedge y \wedge \bar{z}$ и $x \wedge \bar{y} \wedge z$ соединяем операцией $\vee$:

$$\left(\bar{x} \wedge \bar{y} \wedge z\right) \vee \left(\bar{x} \wedge y \wedge \bar{z}\right) \vee \left(x \wedge \bar{y} \wedge z\right).$$

Обозначим полученную СДНФ через $F1$.

Для составления формулы в виде СКНФ для функции булевой алгебры, не соответствующей тождественно истинной (тавтологии) булевой функции, необходимо выбрать выражения, соответствующие строкам, где функции булевой алгебры ложна, и соединить их операцией $\wedge$, при этом если в какой-либо строке логическая переменная имеет истинное значение, то в соответствующем выражении она используется с инверсией (в таблице 37 показаны всевозможные выражения для построения СКНФ).

Таблица 37

Таблица истинности для СКНФ

№ п/п	x	y	z	
1	0	0	0	$x \vee y \vee z$
2	0	0	1	$x \vee y \vee \bar{z}$
3	0	1	0	$x \vee \bar{y} \vee z$
4	0	1	1	$x \vee \bar{y} \vee \bar{z}$
5	1	0	0	$\bar{x} \vee y \vee z$
6	1	0	1	$\bar{x} \vee y \vee \bar{z}$
7	1	1	0	$\bar{x} \vee \bar{y} \vee z$
8	1	1	1	$\bar{x} \vee \bar{y} \vee \bar{z}$

Составим СКНФ функции булевой алгебры F, заданной вектором значений (**0**11**00**1**00**): для этого обратим внимание, что эта функция ложна (содержит нули) в строках под номерами 1, 4, 5, 7 и 8, поэтому выражения из таблицы 37 $x \vee y \vee z$, $x \vee \bar{y} \vee \bar{z}$, $\bar{x} \vee y \vee z$, $\bar{x} \vee \bar{y} \vee z$ и $\bar{x} \vee \bar{y} \vee \bar{z}$ соединяем операцией $\wedge$:

$$(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}).$$

Обозначим полученную СКНФ через $F2$.

Для построения сокращённой ДНФ, которую обозначим через $F3$, раскрываем скобки в СКНФ $F2$, начиная перемножение со скобок, которые отличаются всего одной переменной (например, z и $\bar{z}$).

Поменяв местами вторую и третью скобки и используя законы коммутативности, получим:

$$(x \vee y \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}).$$

Перемножим первую и вторую скобки, а также четвёртую и пятую скобки.

$$(x\bar{x} \vee xy \vee xz \vee y\bar{x} \vee yy \vee yz \vee z\bar{x} \vee zy \vee zz) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge$$
$$\wedge (\bar{x}\bar{x} \vee \bar{x}\bar{y} \vee \bar{x}\bar{z} \vee \bar{y}\bar{x} \vee \bar{y}\bar{y} \vee \bar{y}\bar{z} \vee \bar{z}\bar{x} \vee \bar{z}\bar{y} \vee \bar{z}\bar{z}).$$

Воспользуемся законами идемпотентности и $x\bar{x}=0$.

Получим: $\left(0\vee xy\vee xz\vee y\bar{x}\vee y\vee yz\vee z\bar{x}\vee zy\vee z\right)\wedge\left(x\vee\bar{y}\vee\bar{z}\right)\wedge$

$$\wedge\left(\bar{x}\vee\bar{x}\bar{y}\vee\bar{x}\bar{z}\vee\bar{y}\bar{x}\vee\bar{y}\vee\bar{y}\bar{z}\vee\bar{z}\bar{x}\vee\bar{z}\bar{y}\vee 0\right).$$

В первой скобке слагаемое y поглощает все слагаемые, содержащие y, а слагаемое z поглощает все слагаемые, содержащие z. В третьей скобке слагаемое $\bar{x}$ поглощает все слагаемые, содержащие $\bar{x}$, а слагаемое $\bar{y}$ поглощает все слагаемые, содержащие $\bar{y}$. Получим $\left(y\vee z\right)\wedge\left(x\vee\bar{y}\vee\bar{z}\right)\wedge\left(\bar{x}\vee\bar{y}\right)$.

Перемножим вторую и третью скобки: $\left(y\vee z\right)\wedge\left(x\bar{x}\vee\bar{y}x\vee\bar{z}x\vee x\bar{y}\vee\bar{y}\bar{y}\vee z\bar{y}\right)$.

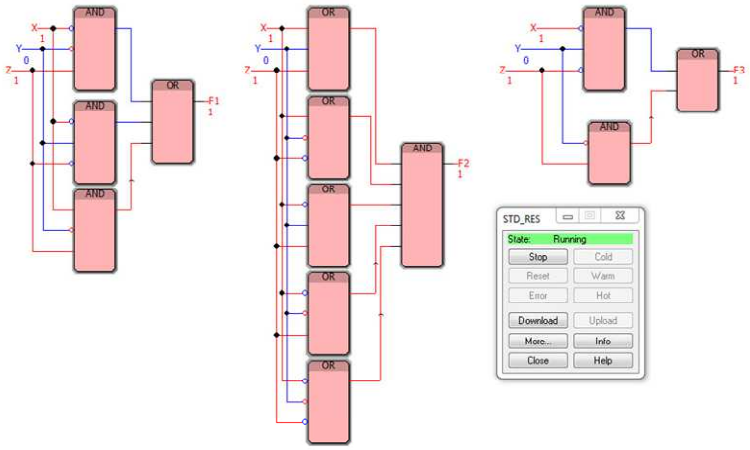
Снова воспользуемся законами идемпотентности, дополнения и поглощения. Получим: $\left(y\vee z\right)\wedge\left(0\vee\bar{y}x\vee\bar{z}x\vee x\bar{y}\vee\bar{y}\vee z\bar{y}\right)\equiv\left(y\vee z\right)\wedge\left(\bar{z}x\vee\bar{y}\right)$.

Перемножаем оставшиеся скобки:

$$\left(y\vee z\right)\wedge\left(\bar{z}x\vee\bar{y}\right)\equiv y\bar{z}x\vee z\bar{z}x\vee y\bar{y}\vee z\bar{y}\equiv y\bar{z}x\vee 0\vee 0\vee z\bar{y}\equiv y\bar{z}x\vee z\bar{y}.$$

Получена сокращенная ДНФ $F3=\bar{x}y\bar{z}\vee\bar{y}z$ функции булевой алгебры F.

На рис. 43 представлены примеры построения СДНФ $F1$, СКНФ $F2$ и сокр. ДНФ $F3$ для варианта задания № 30 из табл. 34.



а б в

Рис. 43. Пример выполнения лабораторной работы 3.4:
а) СДНФ, б) СКНФ, в) сокр. ДНФ

Для проверки правильности реализации программ с выходными переменными *F*1, *F*2 и *F*3 используем таблицу истинности (табл. 35): при предъявлении набора входов выходы со всех функций *F*1, *F*2 и *F*3 будут одинаковые, например, набор (1, 0, 1), соответствующий 6 строке таблицы истинности, имеет значение, равное 1, введя указанный набор на входы *F*1, *F*2 и *F*3, получаем также 1. Аналогично проверяются оставшиеся наборы входов. Таким образом, можно сделать вывод о том, что правильно определена сокращенная ДНФ *F*3 и реализованы программы для СДНФ *F*1, СКНФ *F*2 и сокр. ДНФ *F*3.

3.5. Лабораторная работа «Сложные функции булевой алгебры, содержащие кольцевую сумму»

Цель лабораторной работы*:* научиться реализовывать в среде программирования PC WorX v.6.30 сложные функции булевой алгебры, содержащие кольцевую сумму, а также разрабатывать соответствующие приложения на языке программирования Function Block Diagram.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX функцию булевой алгебры из табл. 38 (обратите особое внимание на приоритет выполнения операций).

Таблица 38

Варианты формул булевой алгебры

№ п/п	Формула I	Формула II
1	2	3
1	$\overline{A} \vee B \oplus B \oplus AB$	$\overline{AB \vee CA} \oplus \overline{ACD \vee C} \wedge \left(AD \oplus AB \vee \overline{B}\right)$
2	$A \wedge \overline{B} \oplus A \vee B$	$\overline{AC \oplus B} \vee \left(CD \oplus \overline{D \wedge AC}\right)A \oplus BD \vee \overline{D}$
3	$AC \oplus A \vee \overline{CA}$	$XYZ \vee XZ \oplus \left(\overline{X \vee YW} \oplus WY\right) \wedge YX \vee Z\overline{W}$
4	$\overline{B} \oplus CB \vee \left(\overline{C} \oplus B\right)$	$\overline{B}F \vee \left(\overline{DF \oplus DC} \vee F\right) \wedge DBC \oplus \overline{\overline{C} \vee DF}$
5	$\overline{C \vee \overline{CA}} \oplus \overline{A}$	$A \vee \overline{BC} \oplus \left(AC \vee \overline{BD \oplus C}\right) \wedge D\overline{C} \vee ABC$
6	$\overline{AB} \vee A \oplus B \vee \overline{A}$	$\overline{D}C \oplus \overline{B \vee E} \vee \overline{DE \oplus B} \oplus DB \wedge \left(C \vee \overline{D}\right)$
7	$BA \vee \left(\overline{A} \oplus B \oplus A\right)$	$FB \vee \overline{E\overline{D} \oplus EF} \oplus \overline{FEB \vee F} \oplus \left(E \vee FB\right)$

Продолжение табл. 38

1	2	3
8	$\overline{BC} \oplus \overline{CB \oplus B \vee \overline{C}}$	$SDG \vee \left(S \oplus \overline{DG}\right) \vee T \oplus \overline{STD \vee G} \oplus S\overline{D}$
9	$AC \oplus A \vee C \oplus \overline{AC}$	$\overline{AB \oplus A} \oplus \overline{ACD \vee C} \wedge \left(A \oplus \overline{ACD \vee \overline{C}}\right)$
10	$\overline{\overline{A} \vee B} \oplus B \oplus AB$	$X\overline{W} \vee \left(Y \oplus XYW\right) \oplus \overline{ZX} \wedge W \oplus X \vee Z$
11	$AC \vee A \oplus \overline{C \oplus A} \vee \overline{C}$	$\overline{SK \vee T \oplus L} \wedge \left(ST \oplus L\right) \oplus \overline{S\overline{L} \vee TSL} \wedge K$
12	$CB \vee \overline{BC} \oplus \overline{BC} \vee \overline{C}$	$\overline{AB \vee D} \wedge CD \oplus BC \wedge \overline{\left(AB \oplus C\right) \vee DC}$
13	$AB \vee \overline{A \oplus AB}$	$S\overline{DG} \vee S \oplus \overline{DG}\Big) \vee T \oplus \overline{STD \vee G} \oplus S\overline{D}$
14	$\overline{\overline{BC} \oplus B \vee C} \oplus C$	$FB \vee \left(\overline{E\overline{D} \oplus EF} \oplus \overline{FEB \vee F}\right) \oplus \left(E \vee FB\right)$
15	$A\overline{B} \oplus \overline{A} \oplus \overline{A}$	$\overline{DC} \oplus \left(\overline{B \vee E} \vee \overline{DE \oplus B}\right) \oplus DB \wedge \left(C \vee \overline{D}\right)$
16	$B\overline{C} \vee \overline{\overline{B} \oplus CB \oplus B}$	$\left(A \vee \overline{BC} \oplus AC \vee \overline{BD \oplus C}\right) \wedge D\overline{C} \vee ABC$
17	$\overline{\overline{AB} \vee A \oplus B} \vee \overline{A}$	$X\left(YZ \vee XZ \oplus \overline{X \vee YW} \oplus WY\right) \wedge YX \vee Z\overline{W}$
18	$\overline{A \wedge \overline{B}} \oplus A \vee \overline{BA}$	$\left(\overline{DF \oplus DC} \vee B\right)F \vee DBC \oplus \overline{\overline{C} \vee DF} \oplus B$
19	$\overline{A \vee \overline{C}A} \oplus \overline{\overline{A}C \oplus C}$	$S \vee \overline{DG}\left(S \vee \overline{DG}\right) \oplus T \oplus \overline{ST \vee G} \oplus S\overline{D} \vee S$
20	$\overline{A} \vee B \oplus \left(B \oplus \overline{AB}\right)$	$\left(X\overline{W} \vee Y \oplus XYW \oplus \overline{ZXW \oplus X}\right) \wedge Z$
21	$AC \oplus \overline{A\overline{C} \oplus AC}$	$\overline{AB \oplus A} \oplus \left(\overline{ACD \vee C} \wedge A \oplus \overline{ABD \vee \overline{C}}\right)$
22	$A\overline{C} \oplus AC \vee \overline{A \oplus C}$	$\overline{ST \vee K \oplus KL}\left(ST \oplus L \oplus \overline{S\overline{L} \vee TSL}\right) \wedge \overline{K}$
23	$BC \oplus \overline{B\overline{C}} \wedge B$	$\overline{A \vee BD} \wedge C \oplus \left(BC \wedge \overline{AB \oplus C \vee DA} \vee A\right)$
24	$CD \oplus \overline{D \wedge CD}$	$ED \oplus \overline{E\overline{D} \oplus EF} \oplus \left(\overline{FEB \oplus F} \oplus E \vee \overline{FB}\right)$
25	$\overline{X \vee XW} \oplus XW$	$\overline{D}A\left(\overline{B \vee E} \vee \overline{DE \oplus B}\right) \oplus DB \oplus \left(\overline{A \vee \overline{D}E}\right)$
26	$\overline{E \oplus EF} \oplus \overline{FE \vee F}$	$\left(\overline{YZ} \vee X\overline{Z} \oplus \overline{X \vee YW} \oplus \overline{WY}\right) \oplus Y \vee Z\overline{YW}$

1	2	3
27	$YZ \vee Z \oplus \overline{YZ} \oplus Y$	$\overline{B \oplus CA} \oplus A\overline{D} \wedge \left(\overline{D \oplus AB} \vee \overline{B}\right) \oplus CB \vee \overline{CD}$
28	$\overline{YZ} \vee \overline{Z} \oplus \overline{Y} \oplus \overline{Y}$	$\overline{A\overline{C} \vee B} \oplus \left(\overline{CD} \oplus \overline{\overline{D} \wedge BC}\right) \oplus BAD \vee \overline{AD}$
29	$A \wedge \overline{DA} \oplus \overline{A \vee D}$	$F \vee \left(A \wedge \overline{DA} \oplus \overline{A \vee DF}\right) \oplus AF \vee A \oplus \overline{DF}$
30	$\left(\overline{X} \oplus Y\right) \oplus (Y \vee X)$	$((X \oplus Y) \vee Z) \oplus \left(Y \wedge \overline{Z} \vee \overline{X}\right)$

Для каждой функции булевой алгебры составьте таблицу истинности и с её помощью проверьте программу в режиме отладки.

Пример выполнения лабораторной работы 3.5

Изобразим в PC WorX программу (рис. 44), соответствующую функции булевой алгебры, состоящей из двух входных логических переменных, $\left(\overline{X} \oplus Y\right) \oplus (Y \vee X)$.

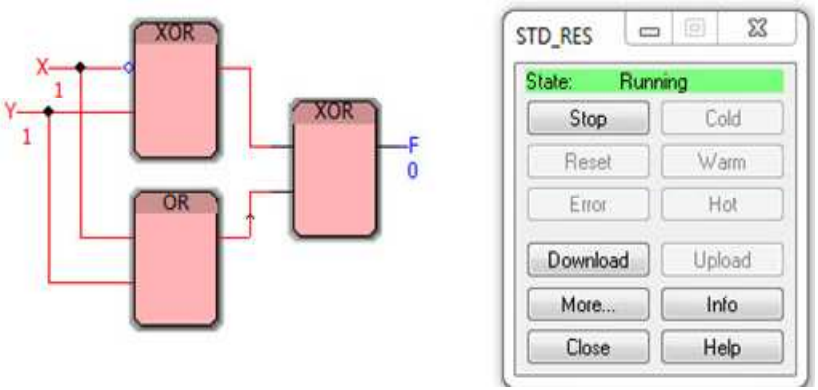


Рис. 44. Пример выполнения варианта № 30/I лабораторной работы 3.5

Для проверки программы составим таблицу истинности (табл. 39), для этого выделим следующие подформулы: $F_1 = \overline{X}$, $F_2 = F_1 \oplus Y$, $F_3 = Y \vee X$, $F_4 = F_2 \oplus F_3$.

Таблица 39

Таблица истинности для $\left(\overline{X} \oplus Y\right) \oplus \left(Y \vee X\right)$

№ п/п	X	Y	F_1	F_2	F_3	F_4
1	0	0	1	1	0	1
2	0	1	1	0	1	1
3	1	0	0	0	1	1
4	1	1	0	1	1	0

Анализ результатов отладки программы для функции булевой алгебры $\left(\overline{X} \oplus Y\right) \oplus \left(Y \vee X\right)$ (пример для набора входных логических переменных (1, 1) показан на рисунке 44) позволяет сделать вывод о её адекватности тестовым примерам.

Изобразим в PC WorX программу (рис. 45), соответствующую функции булевой алгебры, состоящей из трёх входных логических переменных, $\left((X \oplus Y) \vee Z\right) \oplus \left(Y \wedge \overline{Z} \vee \overline{X}\right)$.

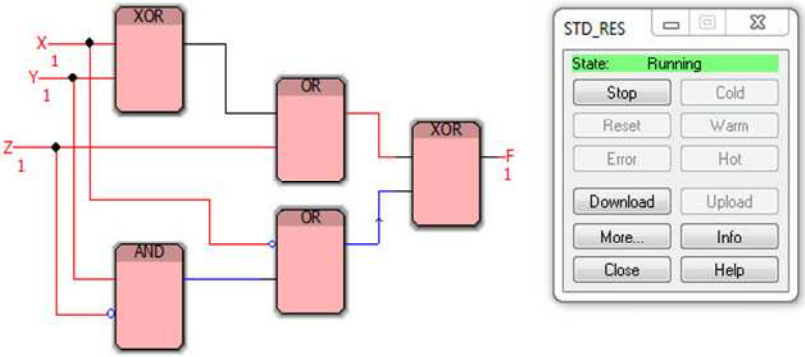


Рис. 45. Пример выполнения варианта № 30/II лабораторной работы 3.5

Для составления таблицы истинности (табл. 40), использующейся для проверки реализованной программы для функции булевой алгебры $\left((X \oplus Y) \vee Z\right) \oplus \left(Y \wedge \overline{Z} \vee \overline{X}\right)$, необходимо выделить следующие подформулы: $F_1 = X \oplus Y$, $F_2 = F_1 \vee Z$, $F_3 = \overline{Z}$, $F_4 = \overline{X}$, $F_5 = Y \wedge F_3$, $F_6 = F_5 \vee F_4$, $F_7 = F_2 \oplus F_6$.

Таблица 40

Таблица истинности для $((X \oplus Y) \vee Z) \oplus (Y \wedge \overline{Z} \vee \overline{X})$

№ п/п	X	Y	Z	F_1	F_2	F_3	F_4	F_5	F_6	F_7
1	0	0	0	0	0	1	1	0	1	1
2	0	0	1	0	1	0	1	0	1	0
3	0	1	0	1	1	1	1	1	1	0
4	0	1	1	1	1	0	1	0	1	0
5	1	0	0	1	1	1	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1
7	1	1	0	0	0	1	0	1	1	1
8	1	1	1	0	1	0	0	0	0	1

Анализ результатов отладки реализованной в PC WorX программы для функции булевой алгебры $((X \oplus Y) \vee Z) \oplus (Y \wedge \overline{Z} \vee \overline{X})$ (пример для набора входных логических переменных (1, 1, 1) показан на рис. 45) позволяет сделать вывод о её адекватности тестовым примерам.

3.6. Лабораторная работа «Полином Жегалкина для функций булевой алгебры»

Цель лабораторной работы*:* научиться реализовывать в среде программирования PC WorX v.6.30 логические функции булевой алгебры в виде полинома Жегалкина, а также разрабатывать соответствующие приложения на языке программирования Function Block Diagram.

Порядок выполнения лабораторной работы

Для функции булевой алгебры из табл. 34 определите СДНФ (СКНФ) и полином Жегалкина. Проверьте правильность полученного полинома Жегалкина построением таблицы истинности. Реализуйте в PC WorX функцию булевой алгебры в виде СДНФ (СКНФ) и полинома Жегалкина, а также проверьте составленные программы в режиме отладки.

Пример выполнения лабораторной работы 3.6

Составленные и реализованные в PC WorX функции булевой алгебры варианта № 30 в виде СДНФ (СКНФ) представлены на рис. 43.

Построим полином Жегалкина для функции булевой алгебры варианта № 30 (см. табл. 34).

Заполним таблицу (табл. 41) по следующим правилам:

1. Первые четыре столбца взяты из условия (заполняются стандартным образом).

2. В пятом столбце строится треугольник Паскаля: верхняя строка такого треугольника представляет собой строку значений исходной функции булевой алгебры; в каждой строке, начиная со второй, любой элемент треугольника равен сумме по модулю 2 (XOR) двух элементов предыдущей строки, расположенной над ним, т.е. элементы второй и последующих строк определяются по следующим правилам: 0 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 1 + 1 = 0, 1 + 0 = 1, 0 + 0 = 0, 0 + 1 = 1.

3. В шестом столбце указаны конъюнкции переменных, значения которых в одном из первых трёх столбцов равны 1 (заметим, что набору (0, 0, 0) соответствует слагаемое 1).

Таблица 41

Таблица для построения полинома Жегалкина

x	y	z	F	Треугольник Паскаля	Слагаемые
0	0	0	0	**0**1100100	1
0	0	1	1	**1**010110	*z*
0	1	0	1	**1**11101	*y*
0	1	1	0	**0**0011	*yz*
1	0	0	0	**0**010	*x*
1	0	1	1	**0**11	*xz*
1	1	0	0	**1**0	*xy*
1	1	1	0	**1**	*xyz*

4. Левая сторона треугольника Паскаля, рассчитанного в столбце 5 табл. 41, равна 01100011. Единицам этой строки соответствуют слагаемые *z*, *y*, *xy* и *xyz* из шестого столбца. Поэтому полином Жегалкина функции булевой алгебры *F* = (01100100) равен $z \oplus y \oplus xy \oplus xyz$.

Проверим правильность полученного полинома Жегалкина построением таблицы истинности (табл. 42).

Таблица 42

Таблица истинности для $F = z \oplus y \oplus xy \oplus xyz$

x	y	z	x ∧ y	x ∧ y ∧ z	z⊕y	z⊕y⊕xy	z⊕y⊕xy⊕xyz
0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1
0	1	0	0	0	1	1	1
0	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0
1	**0**	**1**	**0**	**0**	**1**	**1**	**1**
1	1	0	1	0	1	0	0
1	1	1	1	1	0	1	0

Результат табл. 42 показывает, что полином Жегалкина для функции булевой алгебры F = (01100100) определён верно.

Реализованный в PC WorX полином Жегалкина, соответствующий функции булевой алгебры F = (01100100), показан на рис. 46, правильность составленной программы подтверждается адекватными результатами тестовых примеров, например, в режиме отладки при вводе в качестве входных логических элементов набора (1, 0, 1), соответствующего 6 строке табл. 42, получаем необходимое значение, равное 1.

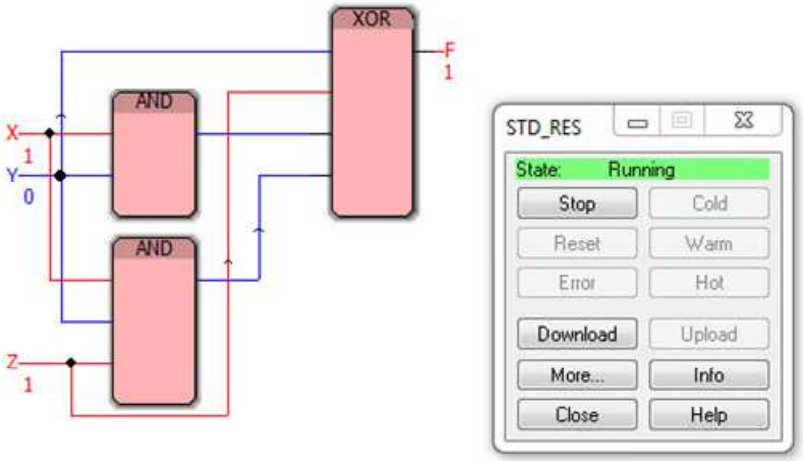


Рис. 46. Реализация тестового примера булевой алгебры
для практического задания № 7

Тема 4. РАБОТА С БИТОВОЙ СТРОКОЙ

Под битовой строкой (bit string) будем понимать данные, представленные в виде двоичной последовательности. Помимо арифметических и булевых функций, для работы с битовой строкой используются операторы сдвига. Программные блоки для операций сдвига на языке Function Block Diagram представлены элементами ROL, ROR, SHL и SHR, вид которых изображен на рис. 47.

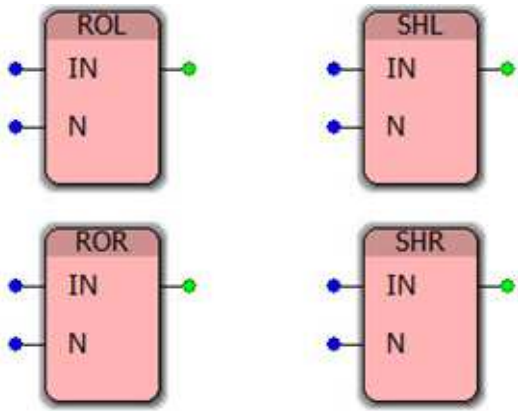


Рис. 47. Программные блоки для операций сдвига языка Function Block Diagram

Результатом операции ROL (циклический сдвиг влево) будет битовая строка, полученная перемещением *N* бит из начала строки в ее конец.

Результатом операции ROR (циклический сдвиг вправо) будет битовая строка, полученная перемещением *N* бит из конца строки в ее начало.

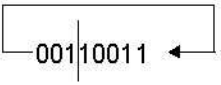
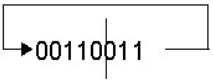
Результатом операции SHL (сдвиг влево) будет битовая строка, полученная сдвигом влево на *N* бит фрагмента строки, начиная с *N*+1. Оставшиеся битовые позиции справа заполняются нулями.

Результатом операции SHR (сдвиг вправо) будет битовая строка, полученная сдвигом вправо на *N* бит фрагмента строки, начиная с *N*+1. Оставшиеся битовые позиции слева заполняются нулями.

Если число *N* задано меньше нуля, то функции будут возвращать значение равное нулю, так как параметр *N* по физическому смыслу предполагается только беззнаковым целым. Недопустимо задавать вход *N* типом bool, это вызовет ошибку ПЛК.

Примеры использования описанных операторов сдвига приведены в табл. 43.

Таблица 43

Вход	*N*	Преобразование	Результат
ROL			
00110011	3		10011001
ROR			
00110011	3		01100110
SHL			
00110011	3		10011000
SHR			
00110011	3		00000110

Исследуйте блоки ROL, ROR, SHL и SHR.

Для этого переместите их на рабочее поле IEC Programming Workspace и задайте произвольные входные переменные с типом DWORD, WORD или BYTE. В режиме отладки присвойте входным переменным значение битовых строк в двоичном виде (например, для типа BYTE – 2#11100011), входной переменной *N* – целочисленное значение, меньшее длины строки, и убедитесь в правильности выдачи результата. Пример описанной процедуры показан на рис. 48.

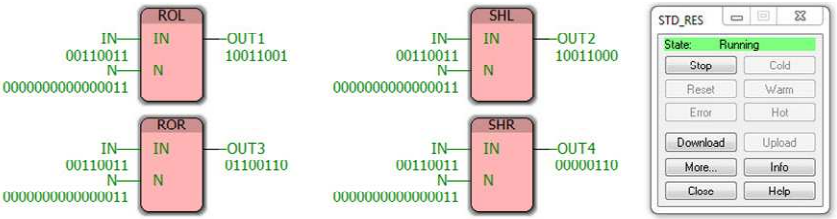


Рис. 48. Исследование блоков ROL, ROR, SHL и SHR

Для вывода результата в виде битовой строки в режиме отладки при вводе значений любой переменной в окне «Debug: STD_RES» установите значение «Binary» в меню «Valuedisplay» (рис. 49).

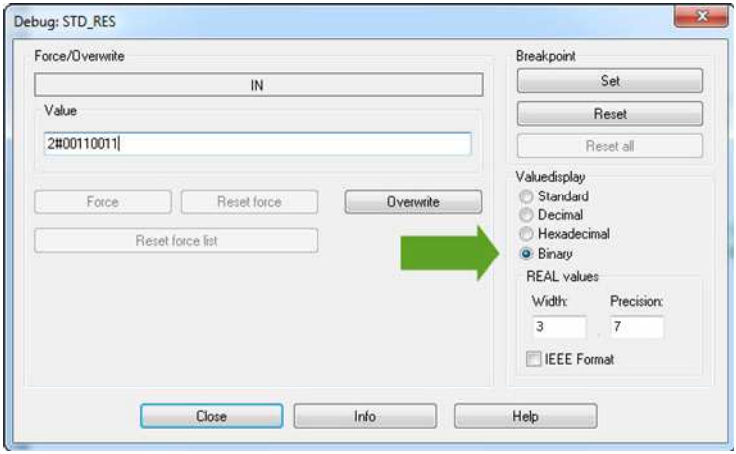


Рис. 49. Окно «Debug:STD_RES»

4.1. Лабораторная работа «Операции сдвига над битовыми строками»

Цель лабораторной работы: научиться работать с битовыми строками в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки функций сдвига.

Порядок выполнения лабораторной работы

Получите результат операций сдвига на *N* бит операторами ROL, ROR, SHL и SHR для битовой строки, представленной в табл. 44.

Таблица 44

Варианты битовых строк

№ п/п	*N*	Битовая строка	№ п.п.	*N*	Битовая строка
1	2	3	4	5	6
1	8	(10001010)	16	3	(10110011)
2	8	(10010011)	17	7	(01101101)

Окончание табл. 44

1	2	3	4	5	6
3	3	(01010110)	18	6	(11010011)
4	2	(00101110)	19	1	(11101010)
5	2	(10001101)	20	7	(10110110)
6	8	(01010110)	21	1	(01111001)
7	1	(10100100)	22	6	(10110101)
8	4	(00001101)	23	1	(01010101)
9	8	(00010101)	24	3	(11000101)
10	1	(01010010)	25	1	(01111000)
11	2	(10010010)	26	3	(11100110)
12	3	(01001001)	27	3	(10101001)
13	4	(00101100)	28	3	(00011110)
14	2	(11001000)	29	8	(11010100)
13	1	(10110000)	30	8	(01100100)

Пример выполнения лабораторной работы 4.1

Для решения поставленной задачи переместите элементы ROL, ROR, SHL и SHR на рабочее поле IEC Programming Workspace и задайте одинаковые входные переменные с типом BYTE. Задайте одинаковую переменную для входа *N* типа INT. В режиме отладки присвойте входным переменным табличное значение битовых строк (например, для варианта № 30 это 2#01100100), входной переменной *N* – табличное целочисленное значение (для варианта № 30 *N* = 8), и получите результат побитовых операций сдвига. Пример описанной процедуры для варианта № 30 показан на рис. 50.

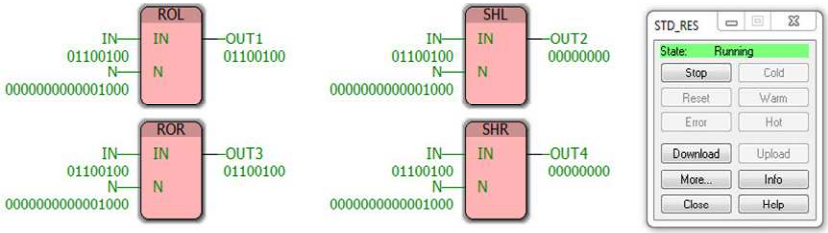


Рис. 50. Пример выполнения лабораторной работы 4.1.

4.2. Лабораторная работа «Получение определённой битовой строки из заданной битовой строки»

Цель лабораторной работы: научиться работать с битовыми строками в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, получающие определённые битовые строки из заданных.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX программу, позволяющую из битовой строки табл. 44 получить битовую строку табл. 45. При необходимости используйте логические операции AND, OR, NOT, XOR.

Таблица 45

Варианты битовых строк

№ п/п	Битовая строка	№ п.п.	Битовая строка
1	(01010010)	16	(10110000)
2	(10110011)	17	(10001010)
3	(01101101)	18	(10010011)
4	(11010011)	19	(01010110)
5	(11101010)	20	(00101110)
6	(01100100)	21	(01010110)
7	(11100110)	22	(10100100)
8	(10101001)	23	(00001101)
9	(00011110)	24	(00010101)
10	(11010110)	25	(10110110)
11	(10010010)	26	(01111001)
12	(01001001)	27	(10110101)
13	(00101100)	28	(01110101)
14	(11001000)	29	(11000101)
15	(10001101)	30	(01011001)

Пример выполнения лабораторной работы 4.2

Разберем решение задачи на примере варианта № 30. Исходная битовая строка (01100100), из которой в результате преобразований необходимо получить строку (01011001). Находим общий для двух строк фрагмент, для нашего случая это часть (011001), которую можно получить из исходной сдвигом вправо на 2 бита, в результате чего получим строку (00011001). Для получения необходимого результата применим дизъюнкцию со строкой (01000000), которую получим из исходной сдвигом влево на 4 бита. Пример программы на Function Block Diagram показан на рис. 51.

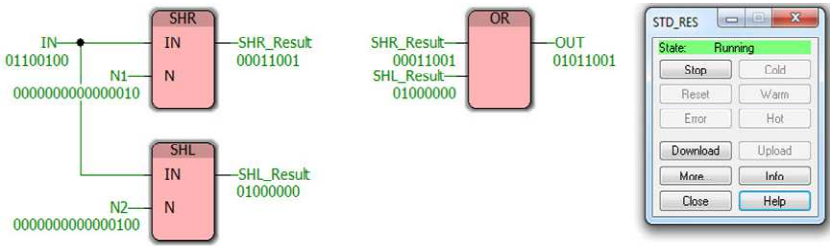


Рис. 51. Пример выполнения лабораторной работы 4.2

4.3. Лабораторная работа «Подсчёт единичных битов в восьмибитовой строке»

Цель лабораторной работы: научиться составлять на языке Function Block Diagram в среде программирования PC WorX v.6.30 приложения по подсчёту единичных битов в строке.

Порядок выполнения лабораторной работы

Составьте в PC WorX программу подсчета единичных битов (степень заполнения) в строке. Подсчитайте число битов в строке для строк, представленных в двоичном и шестнадцатеричном форматах для переменных, указанных в табл. 46 вариант I, алгоритмизировав на языке Function Block Diagram метод подсчета степени заполнения строки.

Пример выполнения лабораторной работы 4.3.

Алгоритм подробно обсуждается в [16] и проиллюстрирован на рис. 52. Так, исходная строка делится на 2-разрядные поля, и в каждое поле помещается число имевшихся в нем единичных бит. Затем значения, содержащиеся в соседних 2-разрядных полях, складываются, и результат помещается в 4-разрядное поле, и т.д. Строка, в которой подсчитывается число единичных бит, находится в первом ряду (маркер 1), в последнем

нижнем ряду (маркер 4) содержится результат в двоичной системе счисления. Выполнение подзадач на самом нижнем уровне (суммирование значений соседних бит) может осуществляться параллельно, как и суммирование значений в соседних полях, которое можно выполнить параллельно за фиксированное число шагов на любом этапе (для 32-битной строки за $\log_2(32) = 5$ шагов, для 8 битной – за $\log_2(38) = 3$ шага).

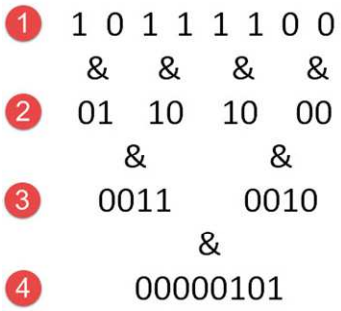


Рис. 52. Алгоритм подсчета степени заполнения строки

На языке С вышеописанный метод может быть выражен следующим образом (рис. 53).

```
1  x = (x & 0x55) + ((x >> 1) & 0x55);
2  x = (x & 0x33) + ((x >> 2) & 0x33);
3  x = (x & 0x0F) + ((x >> 4) & 0x0F);
```

Рис. 53. Программа на языке С для подсчета степени заполнения строки

4.4. Лабораторная работа «Подсчёт единичных битов в шестнадцатибитной строке»

Цель лабораторной работы: научиться составлять на языке Function Block Diagram в среде программирования PC WorX v.6.30 приложения по подсчёту единичных битов в шестнадцатибитной строке.

Порядок выполнения лабораторной работы

Модернизируйте программу для нахождения числа бит в 16-битной строке. Подсчитайте число битов в строке для строк, представленных в двоичном и шестнадцатеричном форматах для переменных, указанных в табл. 46 вариант II.

Таблица 46

№ п/п	**Вариант I**		**Вариант II**	
	Строка 2#	Строка 16#	Строка 2#	Строка 16#
1	10110000	89	0101001010001010	1112
2	11010110	A2	1011001110010011	1019
3	01110110	67	0110110101010110	0D0E
4	01010110	B3	1101001100101110	0716
5	11101010	6F	1110101010001101	0911
6	10001000	A6	0110010001010110	0D1C
7	11100001	A0	1110011010100100	000F
8	10100011	9F	1010100100001101	0C0E
9	00011100	96	0001111000010101	0610
10	01010010	96	1101011001010010	101E
11	11100111	C4	1001001010110011	121B
12	01001000	C0	0100100101101101	081A
13	01001000	AF	0010110011010011	090F
14	10101100	A4	1100100011101010	0809
15	11100011	85	1000110110110110	080B
16	00111001	C1	0101011001111001	0510
17	00111001	8C	1010010010110101	0C14
18	11000000	74	0000110101010101	1203
19	00011101	9B	0001010111000101	0B0E
20	00000001	9E	1011011001111000	080D
21	01111101	72	0111100111100110	090B
22	00000100	B6	1011010110101001	0012
23	01100001	A8	0111010100011110	110C
24	00000111	AA	1100010111010100	1117
25	00011110	B8	1011000010010010	010A
26	00010000	65	1000101001001001	0F19
27	01001011	A7	1001001100101100	0919
28	11011100	7F	0101011011001000	120E
29	00000001	C3	0010111010110000	070E
30	10111000	69	0101100101100100	1410

Пример выполнения лабораторной работы 4.4

Разберем решение задания 1 лабораторной работы 4.4. на примере варианта № 30/I. Составим программу для нахождения числа бит в 8-битной строке. Для этого организуем три последовательных преобразования входной строки, как показано на рис. 53. Маркером 1 на рис. 54, представляющем пример программы на языке Function Block Diagram, отмечен фрагмент программы, выполняющий первое преобразование, аналогичное приведенному в строке 1 рис. 53. В блоке AND реализована операция (x & 0x55), блоком SHR сдвигаем входную строку на 1 бит вправо (x >> 1), после результат сдвига логически умножен на константу, в результате на первый вход блока ADD подается результат преобразования (x & 0x55), а на второй вход – результат ((x >> 1) & 0x55).

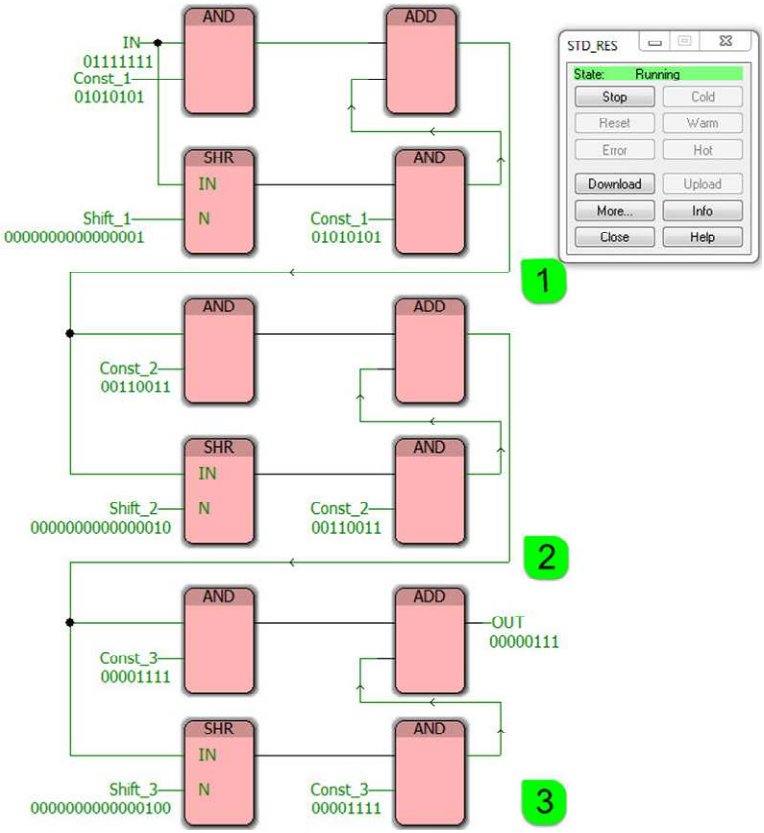


Рис. 54. Пример выполнения лабораторной работы 4.4

Аналогичным образом реализованы два последующих преобразования входной строки, результат представлен на выходе OUT также в виде битовой строки.

Исходная битовая строка в двоичном коде 2#01101001. Результат подсчета числа единичных бит 2#00000100 = 4 шт.

Исходная битовая строка для заданного в шестнадцатеричном коде числа 16#69 выглядит как 2#01101001. Результат подсчета числа единичных бит 2#00000100 = 4 шт.

Тема 5. ИСПОЛЬЗОВАНИЕ ОПЕРАЦИЙ СРАВНЕНИЯ

Операции сравнения представлены на языке Function Block Diagram в PC WorX блоками, изображенными на рис. 55, и соответствуют (слева направо сверху вниз) операциям «Равно» (EQ), «Больше или равно» (GE), «Больше» (GT), «Меньше или равно» (LE), «Меньше» (LT) и «Не равно» (NE).

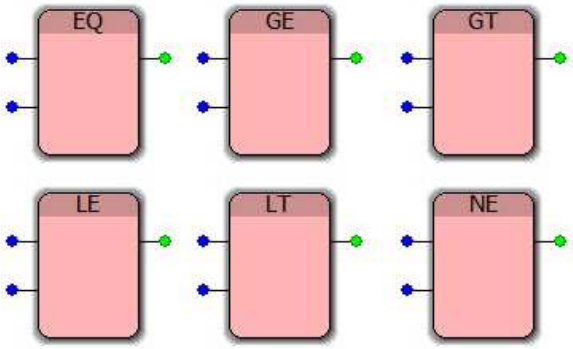


Рис. 55. Базовые блоки для операций сравнения

Все вышеуказанные элементы имеют два входа IN1 и IN2, на которые следует подавать данные для сравнения одинакового типа. Выход элементов имеет тип bool, его можно инвертировать, поставив галочку «negated» в окне свойств элемента. В табл. 47 описаны входные и выходные параметры для всех операций сравнения.

Таблица 47

Операции сравнения

Параметр	Тип данных	Описание
1	2	3
Входные параметры		
IN1	ELEMENTARY	Первое входное значение
IN2	ELEMENTARY	Второе входное значение

Окончание табл. 47

1	2	3
EQ		
OUT	BOOL	TRUE, если входные значения равны. FALSE, если входные значения не равны.
GE		
OUT	BOOL	TRUE, если IN1 больше или равно IN2. FALSE, если IN1 меньше чем IN2.
GT		
OUT	BOOL	TRUE, если IN1 больше чем IN2. FALSE, если IN1 равен или меньше чем IN2.
LE		
OUT	BOOL	TRUE, если IN1 меньше или равно IN2. FALSE, если IN1 больше чем IN2.
LT		
OUT	BOOL	TRUE, если IN1 меньше чем IN2. FALSE, если IN1 больше или равен IN2.
NE		
OUT	BOOL	TRUE, если IN1 и IN2 не равны. FALSE, если IN1 и IN2 равны.

Задание. Исследуйте операции сравнения EQ, GE, GT, LE, LT и NE на примере булевых входных параметров. Для этого переместите исследуемые элементы на рабочее поле IEC Programming Workspace и запишите таблицу истинности для каждого элемента и для всех возможных комбинаций входных параметров. Пример описанной процедуры показан на рис. 56.

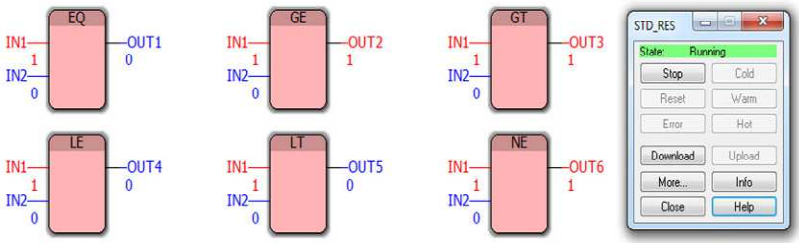
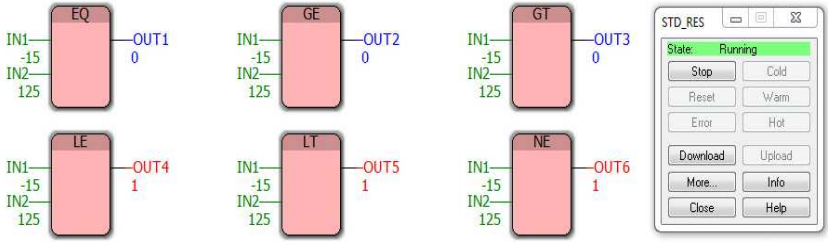
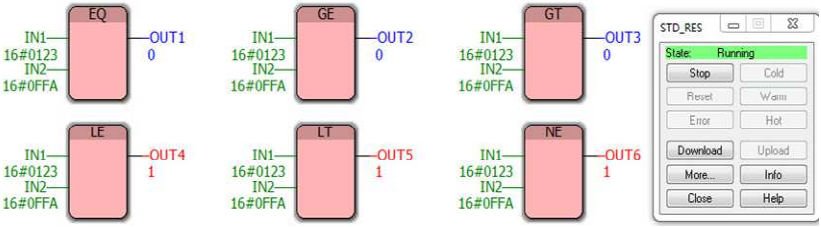


Рис. 56. Исследование операций сравнения. Формат данных BOOL

Поработайте с форматом входных данных INT, WORD. Убедитесь в правильности выдачи результатов (рис. 57).



Формат данных INT



Формат данных WORD

Рис. 57. Исследование операций сравнения. Форматы данных INT, WORD

5.1. Лабораторная работа «Операции сравнения для аналогового сигнала»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки операций сравнения.

Порядок выполнения лабораторной работы

Разработайте программу для реализации операции сравнения, заданной в табл. 48 (столбец «Тип сравнения»), аналогового сигнала с потенциометра и заданного числа. Заданное число рассчитайте как $k \cdot AIN_{max}$, где AIN_{max} максимальное значение потенциометра, k – коэффициент из табл. 48.

Запрограммируйте аппаратный выход ПЛК (Порты OUTPUT0 – OUTPUT3, один из которых необходимо выбрать из табл. 48, столбец «№ порта») для сигнализации выполнения условия сравнения.

Таблица 48

	Вариант I			**Вариант II**		
№ п/п	№ порта OUTPUT	Коэффициент *k*	Тип сравнения	№ порта	Коэффициент	Тип сравнения
1	4	1	NE	3	0,1	LE
2	1	0	EQ	3	0,9	GT
3	1	0,5	GE	3	0,5	LT
4	2	0,4	LE	2	0	NE
5	4	0,4	GT	2	1	EQ
6	4	0,9	LT	2	0,9	GE
7	1	0	NE	4	0,9	LE
8	3	0	EQ	2	0,8	GT
9	1	0,1	GE	2	0,0	LT
10	1	0,3	LE	3	1	NE
11	1	0,5	GT	3	0	EQ
12	2	0,8	LT	4	0,1	GE
13	4	1	NE	1	0,5	LE
14	1	0	EQ	4	0,1	GT
15	4	0,9	GE	1	0,4	LT
16	2	0,6	LE	4	1	NE
17	3	0,8	GT	1	1	EQ
18	1	0,2	LT	2	0,1	GE
19	4	0	NE	3	0,5	LE
20	4	1	EQ	2	0,4	GT
21	4	0,4	GE	2	0,2	LT
22	4	0,6	LE	2	1	NE
23	2	0,5	GT	4	0	EQ
24	1	0,7	LT	3	0,9	GE
25	4	1	EQ	1	0,1	GT
26	1	0,9	GE	3	0,8	LT
27	3	0,6	LE	2	0	NE
28	2	0,7	GT	4	0	EQ
29	2	0,8	LT	3	0,5	GE
30	3	1	NE	4	0,2	LE

Для получения аналогового сигнала с потенциометра следует подключить модуль аналогового ввода. Для этого зайдите в окно «Connected INTERBUS», открыв выпадающее меню по кнопке «View» и выбрав пункт «Connected INTERBUS» (рис. 58). В меню «Selected Control System» выберите ПЛК (ILC 131 ETH). К контроллеру по шине INTERBUS подключен один модуль с идентификационным кодом 127 и строкой данных 32 бит (рис. 58), активируйте его кнопкой «Apply Device/Segment».

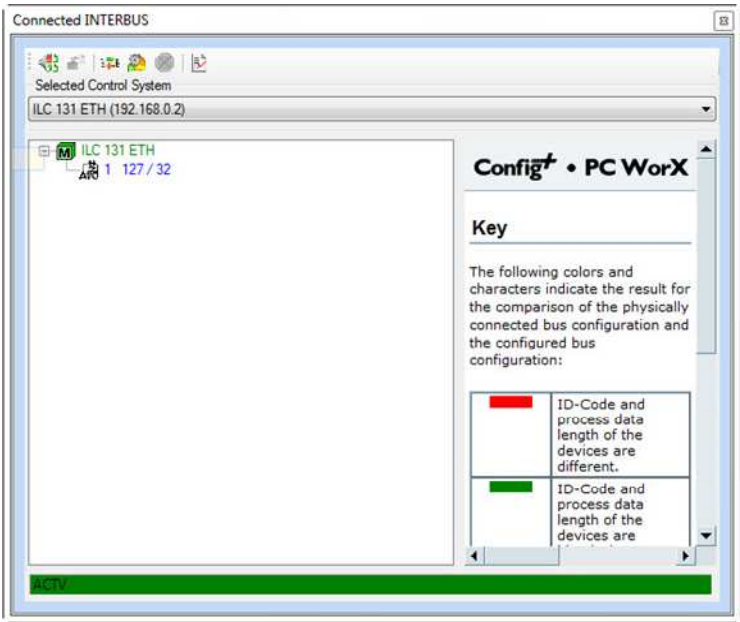


Рис. 58. Окно «Connected INTERBUS»

После нажатия кнопки «Apply Device/Segment» появляется окно «Select Device», предлагающее выбрать тип устройства из предлагаемой группы. Выберите IB IL AI 2/SF-ME, как показано на рис. 59, нажмите «OK». Убедитесь, что модуль подключен правильно (зеленая полоса и строка RUN или ACTV в нижней части экрана). Закройте окно «Connected INTERBUS».

Следующим действием будет создание переменной, которой поставим в соответствие сигнал с модуля аналогового ввода. Для этого в окне «IEC Programming Workspace» создайте глобальную переменную IN типа WORD и локальную переменную OUT типа WORD и соедините их действием «Connect», как показано на рис. 60.

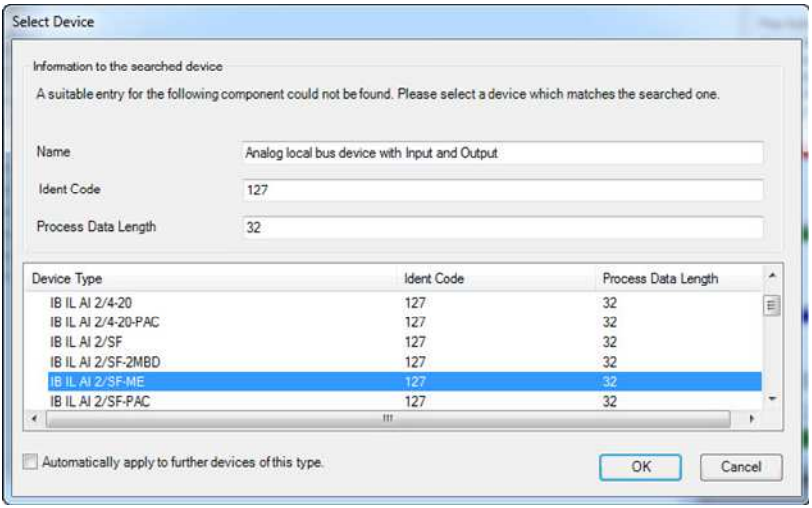


Рис. 59. Окно «Select Device»

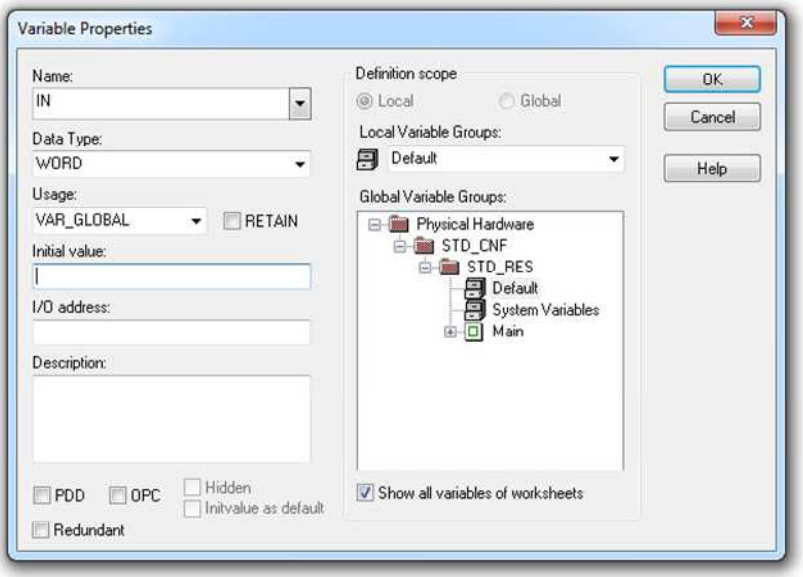


Рис. 60. Окно «Variable Properties»

Нажатием на кнопку «Process Data Workspace» перейдите в окно «Process Data Assignment», в котором в правой части окна видим данные по подключенному модулю аналогового ввода (доступные функции затенены синим цветом), а в левой (выделив папку «STD_RES») – по созданным глобальным переменным. Найдите в правой части напряжение на канале 1 модуля аналогового ввода (AI 1 Voltage), и буксировкой левой кнопкой мыши на переменную IN назначьте переменной IN требуемый аналоговый вход (рис. 61).

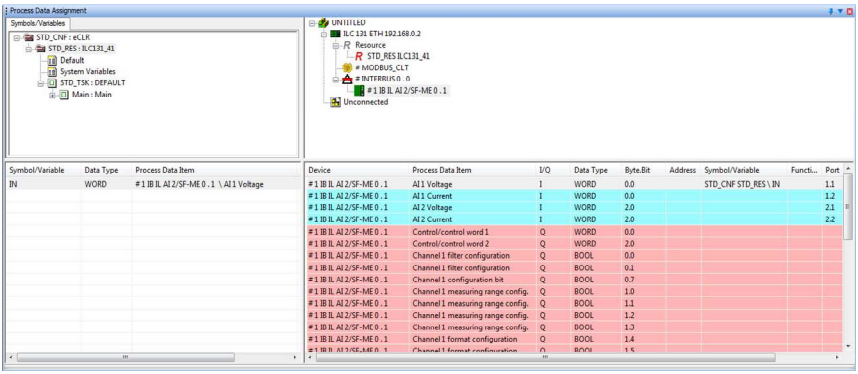


Рис. 61. Окно «Process Data Assignment»

Скомпилируйте проект кнопкой «Make», загрузите его в контроллер и в режиме отладки убедитесь, что значение на выходе OUT меняется в зависимости от положения потенциометра (рис. 62).

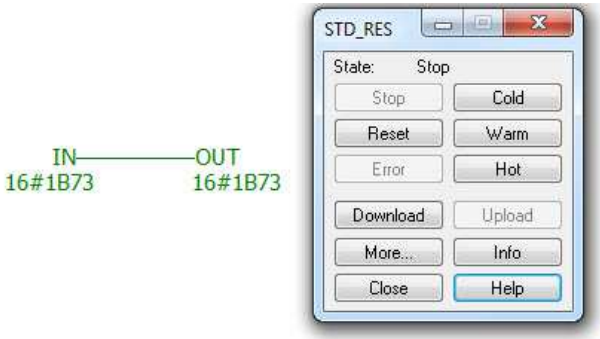


Рис. 62. Пример работы аналогового входа

Пример выполнения лабораторной работы 5.1

Составим программу, соответствующую варианту № 30/I. Предварительно рассчитаем число, с которым будем сравнивать аналоговые данные. Экспериментально определим значение $AIN_{max} = 16\#768C$, т.к. $k = 1$, то требуемое число в формате WORD равно 16#768C. Для сигнализации выполнения условия сравнения зададим выход OUTPUT3, назначив выводу элемента сравнения переменную ONBOARD_OUTPUT_BIT2. Поставим в соответствие глобальной переменной IN вход 1 канала модуля аналогового ввода, как описано выше. Пример реализации такой программы в PC WorX на языке Function Block Diagram показан на рис. 63.

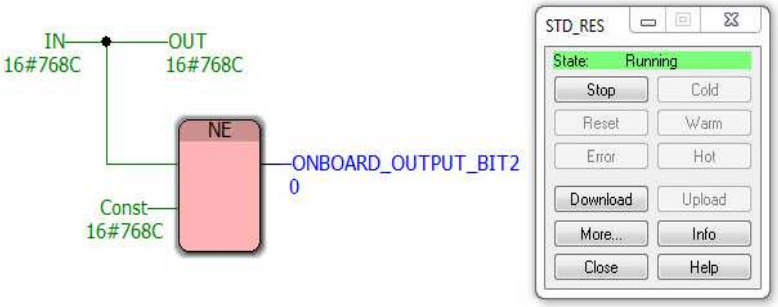


Рис. 63. Пример выполнения варианта 30/I лабораторной работы

Рассмотрим выполнение варианта № 30/II, в целом, аналогичного № 30/I. Константа для сравнения 16#17B5. Сигнал TRUE на выход OUTPUT4 должен подаваться при условии, что на аналоговом входе сигнал меньше либо равен константе 16#17B5. Пример программы приведен на рис. 64.

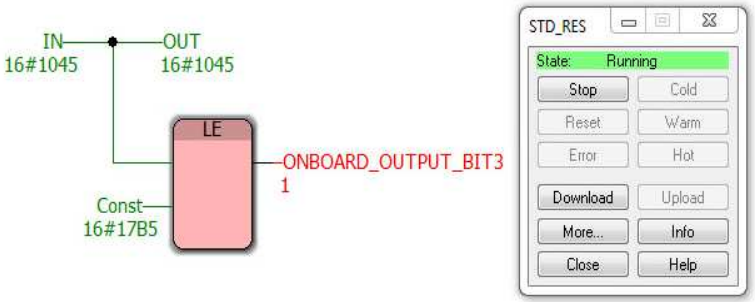


Рис. 64. Пример выполнения варианта 30/II лабораторной работы 5.1

5.2. Лабораторная работа «Реализация неравенств»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие неравенства.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX выражения, представленные в табл. 49.

Таблица 49

Варианты неравенств

№ п/п	Выражение	№ п/п	Выражение
1	$\|x\| + \|y\| < 4$	16	$\sqrt{x^2 + y^2} \geq 3$
2	$\max\{\|x\|, \|y\|\} \leq 2$	17	$\|x\| < 1, \|y\| \geq 5$
3	$\sqrt{(x-2)^2 + (y-2)^2} < 2$	18	$\sqrt{(x-2)^2 + (y-2)^2} \geq 8$
4	$\|x\| + \|y\| > 12$	19	$x^2 + y^2 \geq 9$
5	$x^2 + y^2 < 4$	20	$\sqrt{x^2 + y^2} > 6$
6	$\min\{\|x\|, \|y\|\} \leq 7$	21	$\max\{\|x\|, \|y\|\} \geq 3$
7	$\sqrt{x^2 + y^2} \leq 8$	22	$\min\{\|x\|, y^2\} \geq 2$
8	$\max\{\|x\|, \|y\|\} < 4$	23	$\min\{\|x + 2\|, \|y + 2\|\} \leq 4$
9	$\max\{\|x + 3\|, \|y + 3\|\} \leq 3$	23	$\sqrt{(x-4)^2 + (y-4)^2} > 2$
10	$\|x\| + \|y\| \geq 5$	25	$\min\{\|x\|, \|y\|\} < 2$
11	$\|x\| \leq 4, \|y\| \leq 4$	26	$\max\{\|x + 1\|, \|y + 1\|\} < 2$
12	$x^2 + y^2 \leq 25$	27	$\sqrt{(x-3)^2 + (y-3)^2} \leq 9$
13	$\|x\| \geq 2, \|y\| \geq 5$	28	$\min\{\|x - 2\|, \|y - 2\|\} \geq 2$
14	$\|x\| + \|y\| \leq 5$	29	$\max\{\|x\|, \|y\|\} > 5$
15	$\sqrt{x^2 + y^2} < 4$	30	$x^2 + y^2 > 1$

Составьте проверочную таблицу истинности, состоящую из не менее 5 проверочных примеров, в виде табл. 50.

Таблица 50

№ п/п	*x*	*y*	Левая часть выражения (Result)	{0, 1}
	…	…	…	…
	…			

Пример выполнения лабораторной работы 5.2

Программа, реализующая преобразование $x^2 + y^2 > 1$, показана на рис. 65.

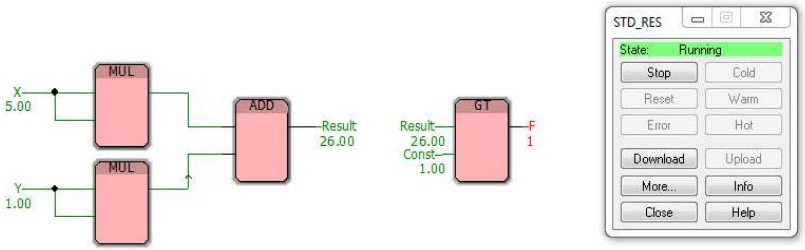


Рис. 65. Пример выполнения варианта 30/I лабораторной работы 5.2.

Промежуточный расчет левой части представлен переменной «Result», и необходим для подготовки проверочной таблицы истинности. Выходная переменная *F* имеет тип bool и принимает значение TRUE, если подставленные числовые значения для переменных *X* и *Y* удовлетворяют неравенству $x^2 + y^2 > 1$.

Результат подстановки пяти значений переменных *X* и *Y*, промежуточный расчет левой части «Result» и значения выходной переменной *F* сведены в проверочную табл. 51.

Таблица 51

№ п/п	*x*	*y*	Левая часть выражения (Result)	{0, 1}
1	0	0	0	0
2	0	1	1	0
3	0.5	0.5	0.5	0
4	5	1	26	1
5	-10	-15	325	1

5.3. Лабораторная работа «Операции сравнения и логические операции»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие выражения, содержащие операции сравнения и логические операции.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX возможность сравнения нескольких величин: $OUT = (x *_1 y) \circ_1 (x *_2 z) \circ_2 (y *_3 z)$, где $*_1$, $*_2$ и $*_3$ – операторы сравнения, $\circ_1$ и $\circ_2$ – логические операторы из табл. 52.

Таблица 52

№ п/п	$*_1$	$\circ_1$	$*_2$	$\circ_2$	$*_3$
1	2	3	4	5	6
1	GE	AND	GE	AND	GE
2	LE	OR	LE	OR	LE
3	GT	XOR	GT	XOR	GT
4	LT	AND	LT	AND	LT
5	NE	OR	NE	OR	NE
6	EQ	XOR	EQ	XOR	EQ
7	GE	AND	GE	AND	GE
8	LE	OR	LE	OR	LE
9	GT	XOR	GT	XOR	GT
10	LT	AND	LT	AND	LT
11	NE	OR	NE	OR	NE
12	EQ	XOR	EQ	XOR	EQ
13	GE	AND	GE	AND	GE
14	LE	OR	LE	OR	LE
15	GT	XOR	GT	XOR	GT
16	LT	AND	LT	AND	LT
17	NE	OR	NE	OR	NE

Окончание табл. 52

1	2	3	4	5	6
18	EQ	XOR	EQ	XOR	EQ
19	GE	AND	GE	AND	GE
20	LE	OR	LE	OR	LE
21	GT	XOR	GT	XOR	GT
22	LT	AND	LT	AND	LT
23	NE	OR	NE	OR	NE
24	EQ	XOR	EQ	XOR	EQ
25	GE	AND	GE	AND	GE
26	LE	OR	LE	OR	LE
27	GT	XOR	GT	XOR	GT
28	LT	AND	LT	AND	LT
29	NE	OR	NE	OR	NE
30	EQ	XOR	EQ	XOR	EQ

Постройте таблицу истинности (табл. 53), состоящую из не менее 5 проверочных примеров, в следующем виде

Таблица 53

№ п/п	x	y	z	$*_1$	{0, 1}	$*_2$	{0, 1}	$*_3$	{0, 1}	$\circ_1$	$\circ_2$	OUT
	…	…	…	…	…	…	…	…	…	…	…	
	…											

Пример выполнения лабораторной работы 5.3

Рассмотрим вариант задания № 30/I. Согласно заданию, предлагается запрограммировать функцию вида $OUT = (x *_1 y) \circ_1 (x *_2 z) \circ_2 (y *_3 z)$, подставив в которую табличные данные, получим $OUT = (x \text{ EQ } y)\text{XOR}(x \text{ EQ } z)\text{XOR}(y \text{ EQ } z)$. Рис. 66 иллюстрирует пример выполнения этого задания.

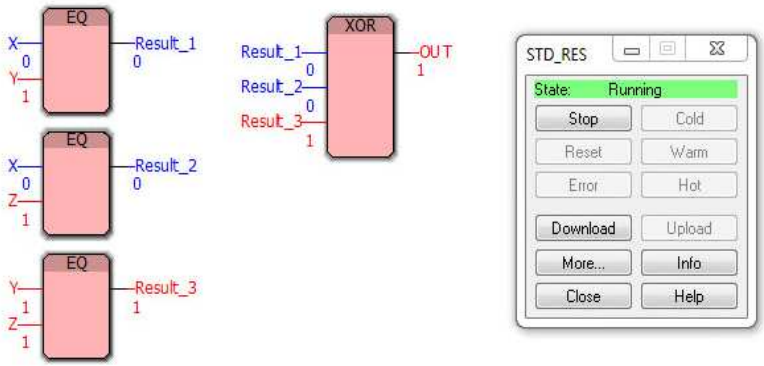


Рис. 66. Пример выполнения варианта 30/I лабораторной работы 5.3

Как указано в задании, строим таблицу истинности из пяти проверочных примеров (табл. 54). Для построения таблицы истинности с промежуточными результатами в программу введены переменные Result_1, Result_2 и Result_3.

Таблица 54

№ п/п	*x*	*y*	*z*	*1	{0, 1}	*2	{0, 1}	*3	{0, 1}	∘1	OUT
	0	0	0	EQ	1	EQ	1	EQ	1	XOR	1
	1	0	0	EQ	0	EQ	0	EQ	1	XOR	1
	0	1	0	EQ	0	EQ	1	EQ	0	XOR	1
	0	0	1	EQ	1	EQ	0	EQ	0	XOR	1
	0	1	1	EQ	0	EQ	0	EQ	1	XOR	1

5.4. Лабораторная работа «Неравенства и логические операции»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие неравенства, содержащие операции сравнения и логические операции.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX выражения: $OUT = \alpha \circ \beta$, где $\circ$ – логический оператор из табл. 55.

Таблица 55

№ п/п	α	β	$\circ$
1	2	3	4
1	$y - x^2 - 1 \le 0$	$y - x^2 + 3 \ge 0$	XOR
2	$y - \frac{4}{x} \le 0$	$y + \frac{4}{x} \ge 0$	AND
3	$x^2 + y^2 - 4x \le 0$	$x^2 + y^2 + 4x \le 0$	OR
4	$y - x^4 - 2 \le 0$	$0 \le y \le \sqrt{x}$	XOR
5	$y + x^2 - 5 \le 0$	$y + x^2 - 6y \le 0$	AND
6	$x^2 + y^2 - 9 \le 0$	$\lvert y \rvert \le 4;\ -6 \le x \le 1$	OR
7	$x - y + 2 > 0$	$x + y < 0$	XOR
8	$y + x^2 - 6 \le 0$	$\lvert x \rvert > 2;\ \lvert y \rvert > 2$	AND
9	$y \le \sin x$	$y > 0{,}5$	OR
10	$x < y + 3$	$x > y - 3$	XOR
11	$y - \frac{5}{x} \le 0$	$y + \frac{2}{x} \ge 0$	AND
12	$x^2 + y^2 + 6y \le 0$	$y + x + 1 \ge 0$	OR
13	$x^2 + y^2 - 25 \le 0$	$y - \frac{4}{x} \le 0$	XOR
14	$0 < y < \sqrt{x}$	$2 \le x \le 6;\ -3 \le y \le 1$	AND
15	$\lvert x \rvert \le 5;\ \lvert y \rvert \le 1$	$y + \frac{5}{x} \ge 0$	OR
16	$x^2 - y - 2 \ge 0$	$x^2 - y + 4 \ge 0$	XOR
17	$\lvert x \rvert \le 2;\ \lvert y \rvert \le 5$	$y + \frac{4}{x} \ge 4$	AND
18	$y + x^2 - 5 \le 0$	$y^2 + x^2 - 6y \le 0$	OR
19	$x^2 - y + 2 \ge 0$	$x + y \ge 0$	XOR

Окончание табл. 55

1	2	3	4
20	$y + x^2 - 6 \le 0$	$x^2 + y^2 \le 4$	AND
21	$\lvert x \rvert \le 4;\ \lvert y \rvert \le 4$	$x^2 + y^2 \le 25$	OR
22	$x \ge \cos y$	$x < 0{,}5$	XOR
23	$y + x^2 - 5 \le 0$	$\lvert x \rvert \le 2;\ -4 \le y \le 0$	AND
24	$y - x^2 - 2 \le 0$	$y - x^2 + 3 \ge 0$	OR
25	$x^2 + y^2 - 6y \le 0$	$y + x^2 + 1 \ge 0$	AND
26	$y - \frac{2}{x} \le 0$	$x^2 + y^2 - 25 \le 0$	OR
27	$0 \le y \le \sqrt{x}$	$2 \le x \le 6;\ -2 < y \le 1$	XOR
28	$\lvert x \rvert \le 6;\ -3 \le y \le -2$	$x^2 + y^2 - 18x \le 0$	AND
29	$\lvert x \rvert \le 5;\ \lvert y \rvert \le 1$	$\lvert x \rvert \le 1;\ \lvert y \rvert \le 5$	OR
30	$x + 2 > y$	$x^2 + y^2 \le 4$	XOR

Составьте проверочную таблицу истинности (табл. 56), состоящую из не менее 5 проверочных примеров.

Таблица 56

№ п/п	x	y	α	{0, 1}	β	{0, 1}	○	OUT
1	…	…	…	…	…	…	…	
2	…							

Пример выполнения лабораторной работы 5.4

Составим программу на языке Function Block Diagram для варианта задания № 30. Согласно заданию, функция имеет вид $(x + 2 > y)\,\mathrm{XOR}\left(x^2 + y^2 \le 4\right)$. На рис. 67 показан пример выполнения этого задания.

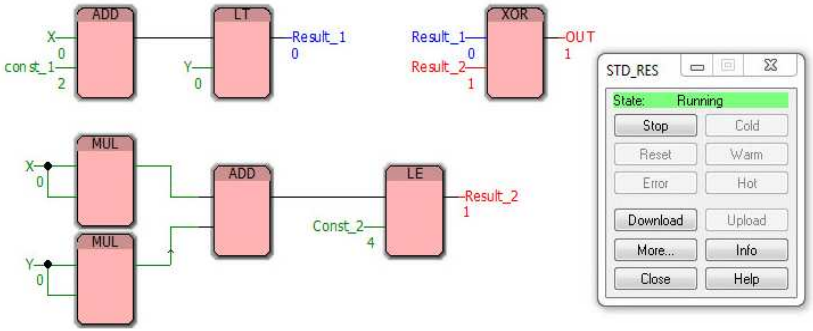


Рис. 67. Пример выполнения варианта 30/I лабораторной работы 5.4

В режиме отладки необходимо несколько раз вводить переменные *X* и *Y*, регистрируя результат преобразований. Переменные Result_1 и Result_2 введены для контроля промежуточных вычислений, их значения также заносятся в проверочную таблицу истинности. Таблица истинности для примера на рис. 67 показана в табл. 57.

Таблица 57

№ п/п	*x*	*y*	*α*	{0, 1}	*β*	{0, 1}	○	OUT
1	0	0	LT	0	LE	1	XOR	1
2	3	0	LT	0	LE	0	XOR	0
3	3	10	LT	1	LE	0	XOR	1
4	10	10	LT	0	LE	0	XOR	0
5	-10	20	LT	1	LE	0	XOR	1

5.5. Лабораторная работа «Сложные арифметические выражения»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие сложные арифметические выражения.

Порядок выполнения лабораторной работы

Разработать приложение на языке Function Block Diagram, реализующее вычисление следующих арифметических выражений (табл. 58).

Таблица 58

№ п/п	Выражение
1	2
1	$y(x_1, x_2, x_3) = 5x_1x_2 - 1 + 2x_1^2 - 2x_1^2x_3^2 + e^{3x_1 + x_2^2}$
2	$y(x_1, x_2, x_3) = \ln\lvert 10 - x_1^2 - x_2 - x_3 \rvert$
3	$y(x_1, x_2, x_3) = \begin{cases} -x_3, \text{если } x_2 \le 0 \\ tg(x_1), \text{если } 0 < x_2 < \dfrac{\pi}{4} \\ 2, \text{если } x_2 \ge \dfrac{\pi}{4} \end{cases}$
4	$y(x_1, x_2, x_3) = \dfrac{\cos(x_1) + 4x_2}{1 + x_3^2}$
5	$y(x_1, x_2, x_3) = \begin{cases} (x_1 + x_2)^2, \text{если } x_2 < x_3 \\ \dfrac{1}{x_1 + x_2}, \text{если } x_2 > x_3 \\ \cos(x_1), \text{если } x_2 = x_3 \end{cases}$
6	$y(x_1, x_2, x_3) = 4 + 7x_1x_2 - 5x_3^2 + x_1^2x_2^4 - e^{6x_3^2 - x_1}$
7	$y(x_1, x_2, x_3) = \ln\lvert 12 - x_1^2 - x_2^2 + x_3 \rvert$
8	$y(x_1, x_2, x_3) = \begin{cases} 2x_1^2 + 1, \text{если } x_3 \le 0 \\ \cos(2x_1), \text{если } 0 < x_3 < \dfrac{\pi}{2} \\ x_2 + 3, \text{если } x_3 \ge \dfrac{\pi}{2} \end{cases}$
9	$y(x_1, x_2, x_3) = \dfrac{2x_1 - \sqrt{4x_2}}{(1 - x_3)^2}$
10	$y(x_1, x_2, x_3) = \begin{cases} \arcsin\left(\dfrac{x_1^2}{x_3}\right), \text{если } x_2 < x_3 \\ \operatorname{arctg}\left(3x_1^2x_2\right), \text{если } x_2 \ge x_3 \end{cases}$

Продолжение табл. 58

1	2
11	$y(x_1, x_2, x_3) = 5 + 3x_2x_3 - 7x_2^2 + x_1^2x_2^2 + e^{3x_1 + x_2^2}$
12	$y(x_1, x_2, x_3) = x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2$
13	$y(x_1, x_2, x_3) = \begin{cases} \cos\left(\frac{\pi x_1}{2}\right), \text{если } \lvert x_2\rvert \le 1 \\ \lvert 1 - x_3\rvert + \ln(x_1), \text{если } \lvert x_2\rvert > 1 \end{cases}$
14	$y(x_1, x_2, x_3) = \sqrt{x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3}$
15	$y(x_1, x_2, x_3) = \begin{cases} \lvert x_1 - x_3\rvert, \text{если } x_1 < 0 \\ -\lvert x_1 - x_3\rvert, \text{если } x_1 > 0 \\ (x_1 - x_3)^2, \text{если } x_1 = 0 \end{cases}$
16	$y(x_1, x_2, x_3) = 2 + 8x_1x_2 - 5x_1^2 + x_2^2x_3 - \sin(4x_1^2 + 3x_2)$
17	$y(x_1, x_2, x_3) = \sqrt{x_1^2 + x_2^2 + x_3^2}$
18	$y(x_1, x_2, x_3) = \begin{cases} \sin(2x_3), \text{если } x_1 \le \frac{\pi}{4} \\ \operatorname{ctg}(x_3), \text{если } \frac{\pi}{4} < x_1 \le \frac{\pi}{2} \\ x_2 - 1, \text{если } x_1 > \frac{\pi}{2} \end{cases}$
19	$y(x_1, x_2, x_3) = \begin{cases} x_1 + \sqrt{x_3}, \text{если } x_1 < x_2 \\ \sqrt{x_1 + x_3}, \text{если } x_1 > x_2 \\ \sqrt{x_1} \cdot \sqrt{x_3}, \text{если } x_1 = x_2 \end{cases}$
20	$y(x_1, x_2, x_3) = \frac{x_1}{\sqrt{x_1 + x_2 + x_3}}$
21	$y(x_1, x_2, x_3) = \ln\lvert 3x_1 - 2x_2 + x_3\rvert$
22	$y(x_1, x_2, x_3) = \begin{cases} x_1 - x_2, \text{если } x_2 < x_3 \\ x_1 + x_2, \text{если } x_2 > x_3 \\ \sin(x_1), \text{если } x_2 = x_3 \end{cases}$
23	$y(x_1, x_2, x_3) = \frac{x_1 - 1{,}7x_2}{x_3^2 + 1}$

1	2
24	$y(x_1, x_2, x_3) = 4{,}5x_1 + 1{,}2x_2x_3 - 0{,}5x_3$
25	$y(x_1, x_2, x_3) = x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3$
26	$y(x_1, x_2, x_3) = \ln(x_1) + \ln(x_2) + \ln(x_3)$
27	$y(x_1, x_2, x_3) = \begin{cases} -x_2, \text{если } x_1 \le 0 \\ \sin(x_3), \text{если } 0 < x_1 \le \pi \\ x_2 - 2, \text{если } x_1 > \pi \end{cases}$
28	$y(x_1, x_2, x_3) = \dfrac{2x_1 + x_2^2}{5 + \ln(x_3)}$
29	$y(x_1, x_2, x_3) = \begin{cases} 2\left(x_1 + x_2^2\right), \text{если } x_3 < 0 \\ \dfrac{1}{2\left(x_1 + x_2^2\right)}, \text{если } x_3 > 0 \\ 0, \text{если } x_3 = 0 \end{cases}$
30	$y(x_1, x_2, x_3) = \begin{cases} \lvert x_1 + x_2 \rvert, \text{если } x_3 < 0 \\ -\lvert x_1 + x_2 \rvert, \text{если } x_3 > 0 \\ 0, \text{если } x_3 = 0 \end{cases}$

Пример выполнения лабораторной работы 5.5

В предлагаемых примерах выходное значение переменной задается функцией, вид которой определяется условиями задачи. Для реализации варианта задания № 30 для операции выбора по условию используем программный блок SEL (рис. 68), значение на выходе (OUT) которого передается со входа IN1, если на входе *G* = FALSE, и выходной сигнал OUT передается со входа IN2, если *G* = TRUE. Входные переменные IN1 и IN2 должны быть одного типа, вход *G* имеет тип bool и может быть инвертирован. Увеличить число входных переменных в блоке SEL невозможно.

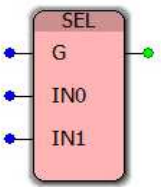


Рис. 68. Программный блок SEL

Пример программы на языке Function Block Diagram, соответствующей варианту № 30/I, показан на рис. 69.

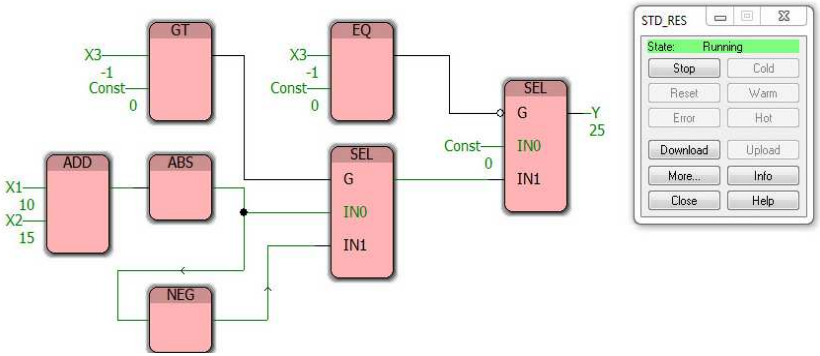


Рис. 69. Пример выполнения лабораторной работы 5.5

Блоки ADD, ABS и NEG осуществляют заданные арифметические преобразования, а блоками GT и EQ производится задание условий перехода выполненных преобразований на выход программных блоков SEL.

5.6. Лабораторная работа «Аналитические выражения с двумя параметрами»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие сложные аналитические выражения с двумя параметрами.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX вычисление следующих аналитических выражений (табл. 59).

Таблица 59

№ п/п	a	b	f	№ п/п	a	b	f
1	2	3	4	5	6	7	8
1	4	9	$f_{S_3}(x;a,b)$		4	8	$f_{S_2}(x;a,b)$
2	4	8	$f_{S_4}(x;a,b)$		5	6	$f_{S_3}(x;a,b)$

Окончание табл. 59

1	2	3	4	5	6	7	8
3	3	10	$f_{\Pi_1}(x;\sigma,c)$		2	10	$f_{S_4}(x;a,b)$
4	4	9	$f_{Z_1}(x;a,b)$		3	6	$f_{\Pi_1}(x;\sigma,c)$
5	4	8	$f_{Z_2}(x;a,b)$		4	7	$f_{Z_1}(x;a,b)$
6	1	7	$f_{Z_3}(x;a,b)$		4	8	$f_{Z_2}(x;a,b)$
7	3	8	$f_{S_1}(x;a,b)$		5	9	$f_{Z_3}(x;a,b)$
8	4	6	$f_{S_2}(x;a,b)$		2	10	$f_{S_1}(x;a,b)$
9	5	7	$f_{S_3}(x;a,b)$		2	9	$f_{S_2}(x;a,b)$
10	3	7	$f_{Z_1}(x;a,b)$		5	8	$f_{\Pi_1}(x;\sigma,c)$
11	3	6	$f_{Z_2}(x;a,b)$		3	6	$f_{Z_1}(x;a,b)$
12	3	8	$f_{Z_3}(x;a,b)$		5	9	$f_{Z_2}(x;a,b)$
13	3	7	$f_{S_1}(x;a,b)$		4	9	$f_{Z_3}(x;a,b)$
14	2	8	$f_{S_2}(x;a,b)$		5	9	$f_{S_1}(x;a,b)$
15	3	6	$f_{S_4}(x;a,b)$		5	7	$f_{S_3}(x;a,b)$

где

$$f_{Z_1}(x;a,b)=\begin{cases}1, \text{ если } x<a\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-a}{b-a}\right)\pi, \text{ если } a\le x\le b\\ 0, \text{ если } x>b\end{cases},$$

$$f_{Z_2}(x;a,b)=\begin{cases}1, \text{ если } x\le a\\ 1-2\left(\frac{x-a}{b-a}\right)^2, \text{ если } a<x\le\frac{a+b}{2}\\ 2\left(\frac{b-x}{b-a}\right)^2, \text{ если } \frac{a+b}{2}<x<b\\ 0, \text{ если } x\ge b\end{cases},$$

$$f_{Z_3}(x;a,b)=\begin{cases}1, \text{ если } x\le a\\ \dfrac{b-x}{b-a}, \text{ если } a<x<b\\ 0, \text{ если } x\ge b\end{cases},$$

$$f_{S_1}(x;a,b)=\begin{cases}0, \text{ если } x<a\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left(\dfrac{x-b}{b-a}\right)\pi, \text{ если } a\le x\le b\\ 1, \text{ если } x>b\end{cases},$$

$$f_{S_2}(x;a,b)=\begin{cases}0, \text{ если } x\le a\\ 2\left(\dfrac{x-a}{b-a}\right)^2, \text{ если } a<x\le\dfrac{a+b}{2}\\ 1-2\left(\dfrac{b-x}{b-a}\right)^2, \text{ если } \dfrac{a+b}{2}<x<b\\ 1, \text{ если } x\ge b\end{cases},$$

$$f_{S_3}(x;a,b)=\begin{cases}0, \text{ если } x\le a\\ \dfrac{x-a}{b-a}, \text{ если } a<x<b\\ 1, \text{ если } x\ge b\end{cases},$$

$$f_{S_4}(x;a,b)=\frac{1}{1+e^{-a(x-b)}},$$

где e – основание натурального логарифма, a и b – числовые параметры из множества действительных чисел, удовлетворяющие условию $a<b$;

$$f_{\Pi_1}(x;a,b)=e^{\frac{-(x-b)^2}{2a^2}},$$

причем a и b – числовые параметры, при этом квадрат первого в теории вероятностей называется дисперсией распределения, b – математическим ожиданием.

Составьте проверочную таблицу (на примере табл. 60), содержащую не менее 10 пар векторов x, $f(x)$, причем значения переменной x рекомендуется взять с некоторым постоянным шагом, а в проверочной таблице указывать их по возрастанию, изобразите по проверочной таблице график функции.

Таблица 60

№ п/п	*x*	*f*(*x*)
1	…	…
2	…	

Пример выполнения лабораторной работы 5.6

Получим решение для варианта № 30. Согласно заданию, функция имеет вид $f_{S3}(x;a,b)=\begin{cases}0, & \text{если } x\le a\\ \dfrac{x-a}{b-a}, & \text{если } a<x<b\\ 1, & \text{если } x\ge b\end{cases}$, где a = 5, b = 7. Программа, выполняющая заданные преобразования на языке Function Block Diagram представлена на рис. 70.

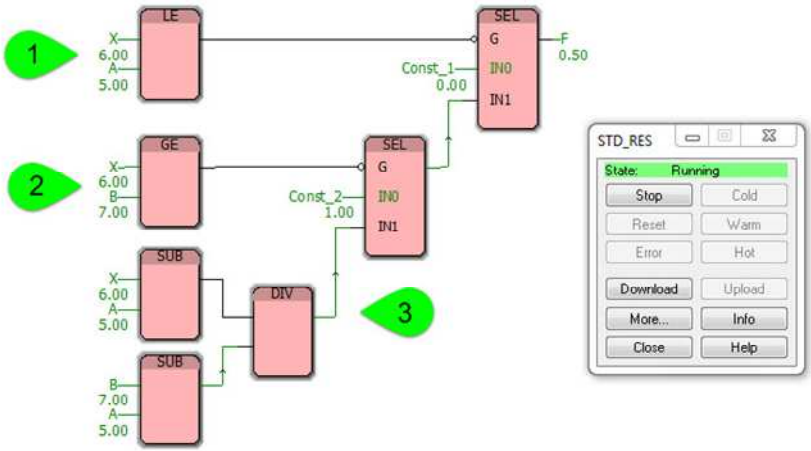


Рис. 70. Пример выполнения лабораторной работы 5.6

На рисунке 70 элемент LE, помеченный маркером 1, производит проверку условия $x\le a$, элемент GE (маркер 2) проверяет на условие $x\ge b$. Третье условие ($a<x<b$) не проверяется, так как оно выполняется всегда, когда не выполняются два вышеуказанных условия. Элементы SUB и DIV (маркер 3) выполняют математическую операцию $\dfrac{x-a}{b-a}$, которая трансли-

руется на выход *F*, если на выходах блоков LE и GE сигнал FALSE. Таблица истинности для примера на рисунке 70 показана в табл. 61.

Таблица 61

№ п/п	*x*	*f*(*x*)
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0.5
7	7	1
8	8	1
9	9	1
10	10	1

На рисунке 71 показан график аналитического выражения, соответствующего вариант № 30 лабораторной работы 5.6.

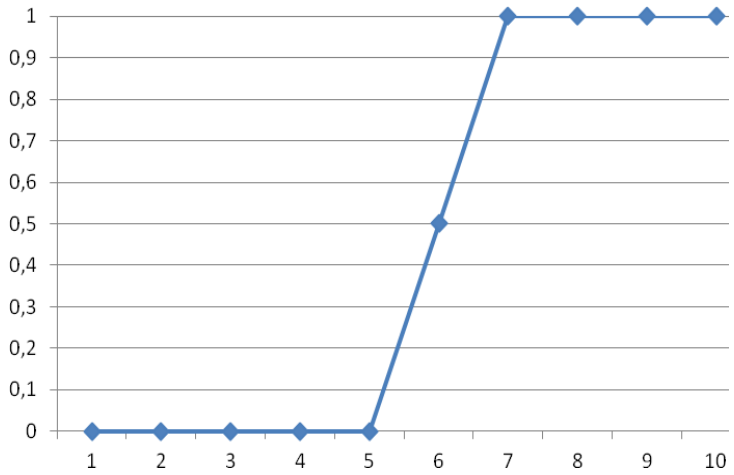


Рис. 71. График функции для варианта № 30 лабораторной работы 5.6

5.7. Лабораторная работа «Аналитические выражения с тремя параметрами»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие сложные аналитические выражения с тремя параметрами.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX вычисление следующих аналитических выражений (табл. 62).

Таблица 62

№ п/п	a	b	c	f	№ п/п	a	b	c	f
1	1	7	12	$f_{\Pi_2}(x;a,b,c)$		5	6	15	$f_{\Delta_1}(x;a,b,c)$
2	4	10	11	$f_{\Delta_1}(x;a,b,c)$		3	6	13	$f_{\Pi_2}(x;a,b,c)$
3	4	9	12	$f_{\Delta_2}(x;a,b,c)$		5	9	11	$f_{\Delta_2}(x;a,b,c)$
4	3	10	14	$f_{\Pi_2}(x;a,b,c)$		2	9	15	$f_{\Delta_1}(x;a,b,c)$
5	4	10	14	$f_{\Delta_1}(x;a,b,c)$		2	10	15	$f_{\Pi_2}(x;a,b,c)$
6	2	9	15	$f_{\Delta_2}(x;a,b,c)$		1	7	14	$f_{\Delta_2}(x;a,b,c)$
7	4	7	12	$f_{\Pi_2}(x;a,b,c)$		4	9	15	$f_{\Delta_1}(x;a,b,c)$
8	1	6	14	$f_{\Pi_2}(x;a,b,c)$		3	10	15	$f_{\Pi_2}(x;a,b,c)$
9	3	8	13	$f_{\Delta_2}(x;a,b,c)$		5	8	14	$f_{\Delta_2}(x;a,b,c)$
10	5	6	11	$f_{\Pi_2}(x;a,b,c)$		1	9	15	$f_{\Delta_1}(x;a,b,c)$
11	5	6	15	$f_{\Delta_2}(x;a,b,c)$		1	7	13	$f_{\Delta_2}(x;a,b,c)$
12	4	7	14	$f_{\Delta_1}(x;a,b,c)$		4	9	13	$f_{\Pi_2}(x;a,b,c)$
13	3	6	14	$f_{\Pi_2}(x;a,b,c)$		1	6	14	$f_{\Delta_1}(x;a,b,c)$
14	3	7	12	$f_{\Delta_2}(x;a,b,c)$		3	10	14	$f_{\Delta_2}(x;a,b,c)$
15	1	7	15	$f_{\Delta_1}(x;a,b,c)$		2	8	11	$f_{\Delta_1}(x;a,b,c)$

где

$$f_{\Delta 1}(x;a,b,c)=\begin{cases}0, & \text{если } x\le a\\ \dfrac{x-a}{b-a}, & \text{если } a\le x\le b\\ \dfrac{c-x}{c-b}, & \text{если } b\le x\le c\\ 0, & \text{если } c\le x\end{cases},$$

$$f_{\Delta 2}(x;a,b,c)=\min\{f_{S3}(x;a,b);\, f_{Z3}(x;b,c)\},$$

$$f_{\Pi 2}(x;a,b,c)=\frac{1}{1+\left|\dfrac{x-c}{a}\right|^{2b}},$$

причем a, b и c – числовые параметры из множества действительных чисел, удовлетворяющие условиям $a<b<c$ и $b>0$.

Составьте проверочную таблицу (табл. 63), содержащую не менее 10 пар векторов x, $f(x)$, причем значения переменной x рекомендуется взять с некоторым постоянным шагом, а в проверочной таблице указывать их по возрастанию, изобразите по проверочной таблице график функции.

Таблица 63

№ п/п	x	$f(x)$
1	…	…
2	…	

Пример выполнения лабораторной работы 5.7

Подставим для варианта № 30 параметры из таблицы 62 в аналитическое выражение

$$f_{\Delta 1}(x;a,b,c)=\begin{cases}0, & \text{если } x\le a\\ \dfrac{x-a}{b-a}, & \text{если } a\le x\le b\\ \dfrac{c-x}{c-b}, & \text{если } b\le x\le c\\ 0, & \text{если } c\le x\end{cases},$$

получим выражение вида:

$$f_{\Delta_1}(x;2,8,11)=\begin{cases}0, \text{ если } x\le 2\\ \dfrac{x-a}{b-a}, \text{ если } 2\le x\le 8\\ \dfrac{c-x}{c-b}, \text{ если } 8\le x\le 11\\ 0, \text{ если } 11\le x\end{cases}.$$

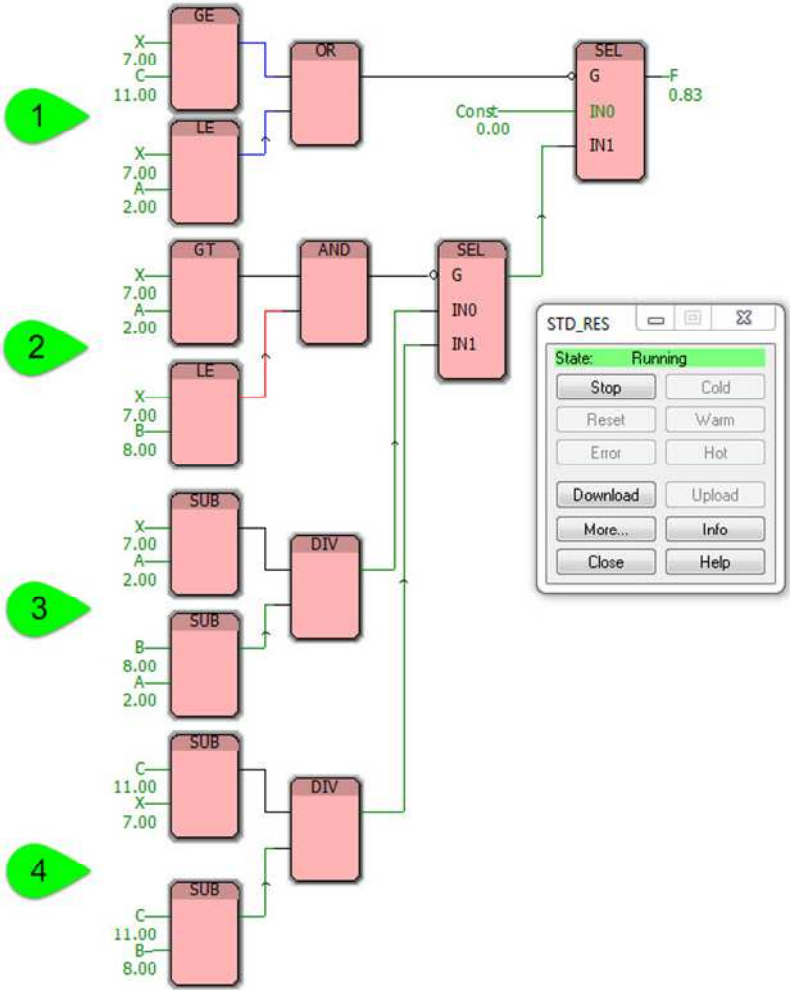


Рис. 72. Пример выполнения лабораторной работы 5.7

Представленный на рис. 72 пример реализует указанное аналитическое выражение из табл. 62 варианта № 30. Блоками GE, LE и OR (маркер 1) проверяются условия $x \le a$ и $c \le x$. При выполнении этих условий выходной функции ставится в соответствие переменная *Const* = 0. Блоками GT, LE и AND (маркер 2) проверяется принадлежность переменной *X* диапазону $a \le x \le b$, при выполнении этого условия на выход *Y* (маркер 5) подается результат выполнения операции $\frac{x-a}{b-a}$ (маркер 3), если условие не выполняется, то на выход *Y* подается результат математической операции $\frac{c-x}{c-b}$ (маркер 4).

Составим для указанного аналитического выражения проверочную табл. 64.

Таблица 64

№ п/п	*x*	*f*(*x*)
1	1	0
2	2	0
3	3	0,166667
4	4	0,333333
5	5	0,5
6	6	0,666667
7	7	0,833333
8	8	1
9	9	0,666667
10	10	0,333333
11	11	0
12	12	0

Изобразим рисунок, соответствующий аналитическому выражению варианта № 30, составленный по табл. 64.

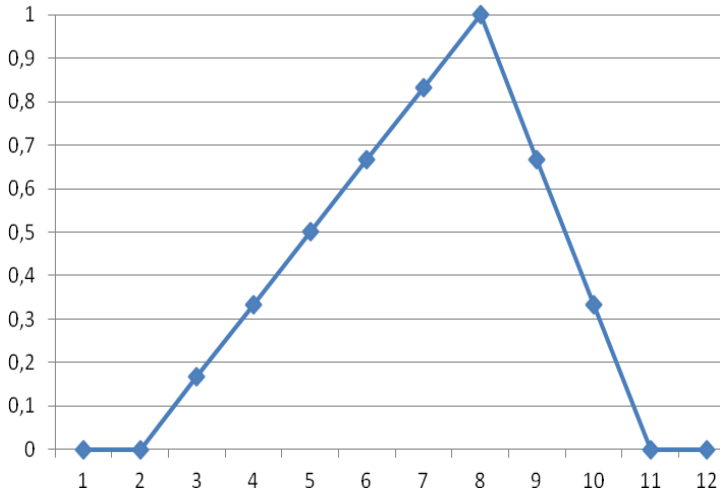


Рис. 73. График функции для варианта № 30 лабораторной работы 5.7

5.8. Лабораторная работа «Аналитические выражения с четырьмя параметрами»

Цель лабораторной работы: научиться работать с операциями сравнения в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, реализующие сложные аналитические выражения с четырьмя параметрами.

Порядок выполнения лабораторной работы

Реализуйте в PC WorX вычисление следующих аналитических выражений (табл. 65).

Таблица 65

№ п/п	*a*	*b*	*c*	*d*	*f*
1	2	3	4	5	6
1	4	10	14	17	$f_{T_1}(x;a,b,c,d)$
2	4	10	15	20	$f_{T_2}(x;a,b,c,d)$
3	4	7	11	19	$f_{П_3}(x;a,b,c,d)$

Продолжение табл. 65

1	2	3	4	5	6
4	3	10	15	20	$f_{\text{П}4}(x; a, b, c, d)$
5	5	7	13	17	$f_{\text{П}5}(x; a, b, c, d)$
6	3	8	12	16	$f_{\text{Т}1}(x; a, b, c, d)$
7	2	6	15	20	$f_{\text{Т}2}(x; a, b, c, d)$
8	3	7	15	20	$f_{\text{П}3}(x; a, b, c, d)$
9	5	6	14	18	$f_{\text{П}4}(x; a, b, c, d)$
10	5	6	13	19	$f_{\text{П}5}(x; a, b, c, d)$
11	5	10	15	19	$f_{\text{Т}1}(x; a, b, c, d)$
12	1	6	12	19	$f_{\text{Т}2}(x; a, b, c, d)$
13	1	10	15	19	$f_{\text{П}3}(x; a, b, c, d)$
14	5	10	12	19	$f_{\text{П}4}(x; a, b, c, d)$
15	2	10	15	17	$f_{\text{П}5}(x; a, b, c, d)$
16	2	6	12	16	$f_{\text{Т}1}(x; a, b, c, d)$
17	2	9	11	16	$f_{\text{Т}2}(x; a, b, c, d)$
18	5	10	11	17	$f_{\text{П}3}(x; a, b, c, d)$
19	1	7	11	16	$f_{\text{П}4}(x; a, b, c, d)$
20	2	7	12	20	$f_{\text{П}5}(x; a, b, c, d)$
21	2	8	14	16	$f_{\text{Т}1}(x; a, b, c, d)$
22	4	7	14	20	$f_{\text{Т}2}(x; a, b, c, d)$
23	3	6	14	20	$f_{\text{П}3}(x; a, b, c, d)$
24	4	10	12	16	$f_{\text{П}4}(x; a, b, c, d)$
25	3	8	15	20	$f_{\text{Т}1}(x; a, b, c, d)$
26	4	6	13	18	$f_{\text{Т}2}(x; a, b, c, d)$

Окончание табл. 65

1	2	3	4	5	6
27	4	7	12	20	$f_{\Pi_3}(x;a,b,c,d)$
28	4	8	11	18	$f_{\Pi_4}(x;a,b,c,d)$
29	4	9	13	17	$f_{\Pi_5}(x;a,b,c,d)$
30	2	8	14	17	$f_{\Pi_5}(x;a,b,c,d)$

где

$$f_{\mathrm{T}_1}(x;a,b,c,d)=\begin{cases}0, & \text{если } x\le a\\ \dfrac{x-a}{b-a}, & \text{если } a\le x\le b\\ 1, & \text{если } b\le x\le c\\ \dfrac{d-x}{d-c}, & \text{если } c\le x\le d\\ 0, & \text{если } d\le x\end{cases},$$

$$f_{\mathrm{T}_2}(x;a,b,c,d)=\min\{f_{S_3}(x;a,b);f_{Z_3}(x;c,d)\},$$

$$f_{\Pi_3}(x;a,b,c,d)=f_{S_1}(x;a,b)\cdot f_{Z_1}(x;c,d),$$

$$f_{\Pi_4}(x;a,b,c,d)=f_{S_2}(x;a,b)\cdot f_{Z_2}(x;c,d),$$

$$f_{\Pi_5}(x;a,b,c,d)=f_{S_3}(x;a,b)\cdot f_{Z_3}(x;c,d),$$

причем a, b, c и d – числовые параметры из множества действительных чисел, удовлетворяющие условию $a\le b<c\le d$;

Составьте проверочную таблицу (табл. 66), содержащую не менее 10 пар векторов x, $f(x)$, причем значения переменной x рекомендуется взять с некоторым постоянным шагом, а в проверочной таблице указывать их по возрастанию, изобразите по проверочной таблице график функции.

Таблица 66

№ п/п	x	$f(x)$
1	…	…
2	…	

Пример выполнения лабораторной работы 5.8

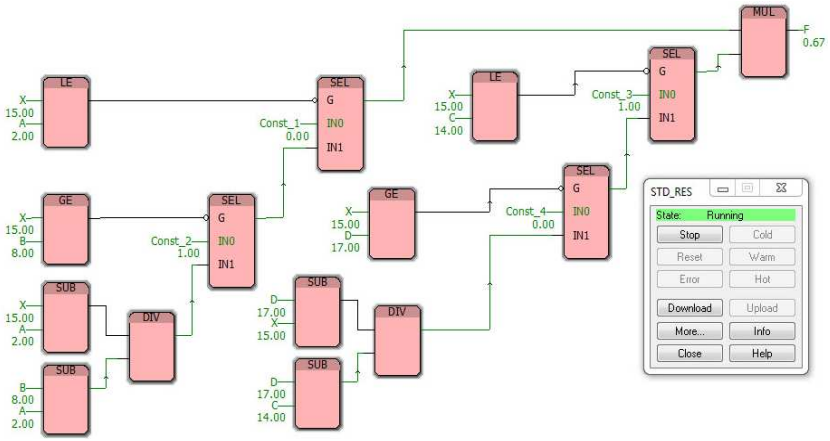


Рис. 74. Пример выполнения лабораторной работы 5.8

Представленный на рис. 74 пример реализует аналитическое выражение из табл. 65 варианта № 30. На рисунке можно выделить два фрагмента программы, аналогичных описанному в примере лабораторной работы 5.8., где сигналы с выхода перемножаются в блоке MUL. Проверочная таблица представлена ниже (табл. 67).

Таблица 67

№ п/п	x	f_{S_3}	f_{Z_3}	$f(x)$
1	1	0	1	0
2	3	0,166667	1	0,166667
3	5	0,5	1	0,5
4	7	0,833333	1	0,833333
5	9	1	1	1
6	11	1	1	1
7	13	1	1	1
8	15	1	0,666667	0,666667
9	17	1	0	0
10	19	1	0	0
11	21	1	0	0

На рисунке 75 показан результирующий график аналитического выражения $f\Pi_5$ варианта № 30 при параметрах, заданных в табл. 65.

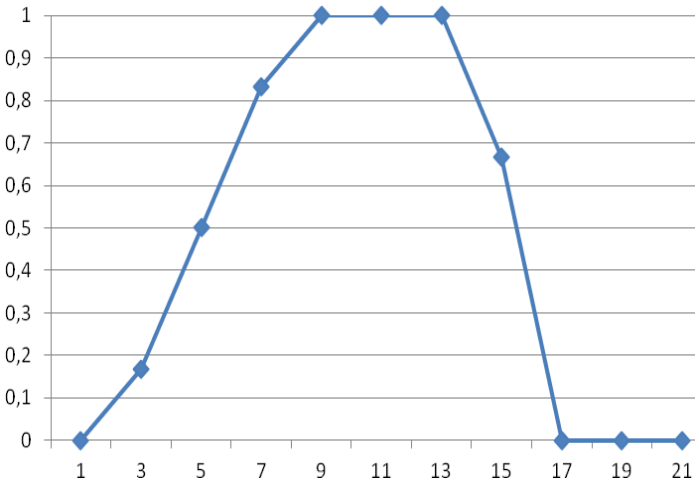


Рис. 75. График функции для варианта № 30 лабораторной работы 5.8

Тема 6. РАБОТА С ТЕКСТОВОЙ СТРОКОЙ

Последовательность символов алфавита, заключенная в кавычки, представляет собой символьную строку. Кавычки в строку не входят. Например, 'Это строка символов'. Первая кавычка указывает на начало строки, последняя – на ее окончание.

Функции для операций с символьными строками на языке Function Block Diagram представлены элементами CONCAT, DELETE, EQ_STRING NE_STRING GE_STRING, GT_STRING, LE_STRING, LT_STRING, FIND, INSERT, MID, REPLACE, LEFT, RIGHT и LEN, вид которых изображен на рис. 76.

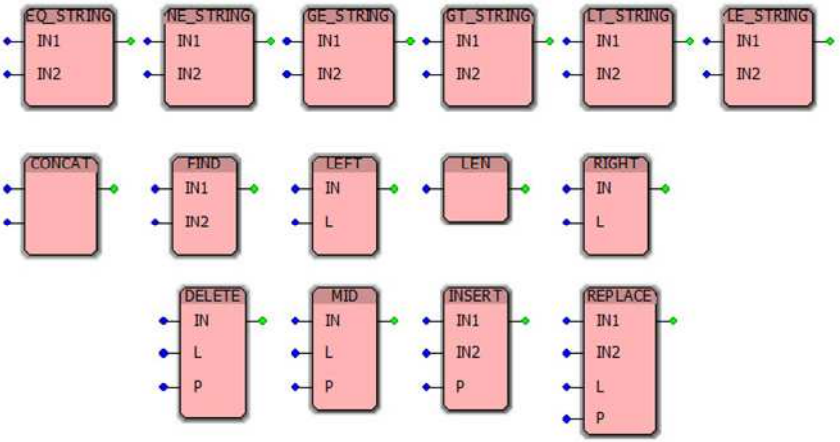


Рис. 76. Программные блоки для работы с символьной строкой

Функция CONCAT объединяет на выходе OUT (тип данных STRING) две символьные строки путем добавления символьной строки на входе IN2 (тип данных STRING) в конец символьной строки на входе IN1 (тип данных STRING).

Функция DELETE. Эта функция удаляет из символьной строки IN (тип данных STRING) фрагмент с числом символов L (тип данных ANY_INT) начиная с позиции P (тип данных ANY_INT). Тип данных на

выходе OUT – STRING. В случае неверного ввода данных результирующая переменная возвращает пустую строку, и ПЛК выдает ошибку.

Функция EQ_STRING сравнивает символьную строку с входа IN1 (тип данных STRING) с символьной строкой с входа IN2 (тип данных STRING) и устанавливает на выходе OUT (тип данных BOOL) значение TRUE, если строки IN1 и IN2 равны, и значение FALSE, если не равны.

Функция NE_STRING сравнивает символьную строку с входа IN1 (тип данных STRING) с символьной строкой с входа IN2 (тип данных STRING) и устанавливает на выходе OUT (тип данных BOOL) значение TRUE, если строки IN1 и IN2 не равны, и значение FALSE, если они равны.

Функция GE_STRING сравнивает символьную строку с входа IN1 (тип данных STRING) с символьной строкой с входа IN2 (тип данных STRING) и устанавливает на выходе OUT (тип данных BOOL) значение TRUE, если строка IN1 больше или равна строке IN2, в противном случае устанавливается значение FALSE.

Функция GT_STRING сравнивает символьную строку с входа IN1 (тип данных STRING) с символьной строкой с входа IN2 (тип данных STRING) и устанавливает на выходе OUT (тип данных BOOL) значение TRUE, если строка IN1 больше строки IN2, в противном случае устанавливается значение FALSE.

Функция LE_STRING сравнивает символьную строку с входа IN1 (тип данных STRING) с символьной строкой с входа IN2 (тип данных STRING) и устанавливает на выходе OUT (тип данных BOOL) значение TRUE, если строка IN1 меньше или равна строке IN2, в противном случае устанавливается значение FALSE.

Функция LT_STRING сравнивает символьную строку с входа IN1 (тип данных STRING) с символьной строкой с входа IN2 (тип данных STRING) и устанавливает на выходе OUT (тип данных BOOL) значение TRUE, если строка IN1 меньше строки IN2, в противном случае устанавливается значение FALSE.

Для функций GE_STRING, GT_STRING, LE_STRING и LT_STRING необходимо помнить, что сравнение осуществляется слева направо. Символ 'Z' больше, чем символ 'A', поэтому строка 'Z' больше, чем строка 'AZ' и 'AZ' больше строки 'ABC'. При сравнении двух строк разной длины, более короткая строка должна быть продлена справа до величины более длинной строки символами с нулевым значением. Выход OUT можно инвертировать.

Функция FIND определяет положение строки на входе IN2 (тип данных STRING) в строке на входе IN1. Позиция символа первого вхождения строки на входе IN2 в строку IN1 передается на выход OUT (тип данных INT). Если совпадений не найдено, то выход OUT = 0.

Функция INSERT. Эта функция символьной строки вставляет строку с входа IN2 (тип данных STRING) в строку на входе IN1 (тип данных STRING) после символа на позиции P (тип данных INT).

Функция MID извлекает фрагмент символьной строки, поданной на вход IN (тип данных STRING), начиная с позиции P (тип данных ANY_INT). Число извлекаемых символов определяется переменной L (тип данных ANY_INT).

Функция REPLACE заменяет фрагмент символьной строки с входа IN1 на строку с входа IN2, начиная с позиции P (тип данных ANY_INT). Число символов, которые будут заменены, определяется переменной L (тип данных ANY_INT).

Необходимо понимать, что для функций INSERT, MID и REPLACE значение переменной на входе P не может быть равно 0. Первая позиция в строке представляет собой всегда 1. Также невозможно использовать одну и ту же строку в качестве входной и выходной строки. В этом случае следует использовать промежуточную переменную на выходе, которой затем назначается значение входной переменной. Если требуется вставить строку одну перед другой, следует использовать не функцию INSERT, а функцию CONCAT.

Функция LEFT извлекает L левых символов строки с входа IN. Тип данных для входа IN и выхода OUT – STRING, для входа L – ANY_INT.

Функция RIGHT извлекает справа L символов строки с входа IN. Тип данных для входа IN и выхода OUT – STRING, для входа L – ANY_INT.

Функция LEN определяет длину символьной строки. Тип данных для входа IN – STRING, для выхода OUT – INT.

Задание. Исследуйте блоки CONCAT, DELETE, EQ_STRING NE_STRING, GE_STRING, GT_STRING, LE_STRING, LT_STRING, FIND, INSERT, MID, REPLACE, LEFT, RIGHT и LEN. Для этого переместите их на рабочее поле IEC Programming Workspace и задайте произвольные входные переменные с типом данных STRING. В режиме отладки присвойте входным переменным значения символьных строк и убедитесь в правильности выдачи результатов. Пример описанной процедуры показан на рис. 77.

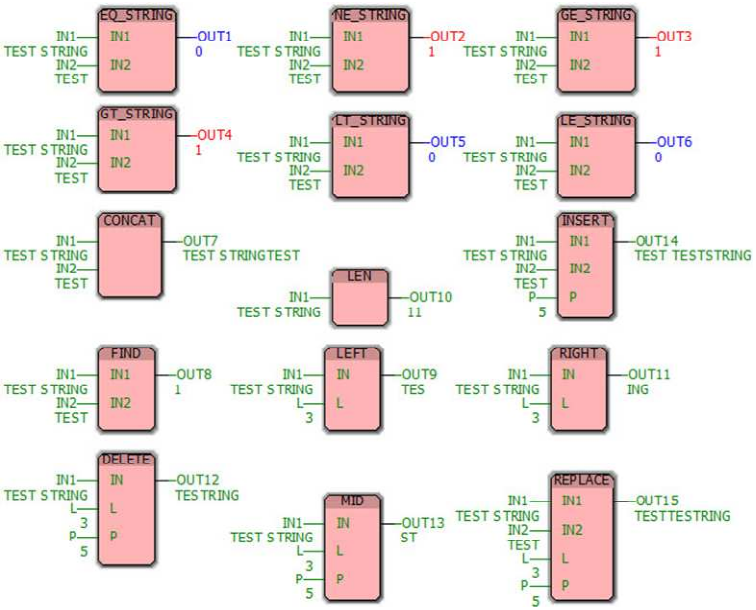


Рис. 77. Исследование программных блоков для работы с символьной строкой

6.1. Лабораторная работа «Проверка символьных строк на логическое условие»

Цель лабораторной работы: научиться работать с текстовыми строками в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки операций с символьными строками.

Порядок выполнения лабораторной работы

Проверьте символьную строку на условие, заданное в табл. 68. Результат проверки представьте в виде логически обоснованной символьной строки.

Таблица 68

№ п/п	Символьная строка	Присутствуют элементы
1	2	3
1	'let the punishment fit the crime'	не 'a' или 'b' и 'c'
2	'opportunity only knocks once'	(не 'a' или 'b') и 'c'

Окончание табл. 68

1	2	3
3	'a chain is only as strong as its weakest link'	'a' или не 'b' или 'c'
4	'a change is as good as a rest'	'a' и не 'b' и 'c'
5	'a good man is hard to find'	'a' или не 'b' и 'c'
6	'a house divided against itself cannot stand'	('a' или не 'b') и 'c'
7	'a bad workman blames his tools'	'a' или 'b' или не 'c'
8	'a bird may be known by its song'	'a' и 'b' и не 'c'
9	'a broken friendship may be soldered, but will never be sound'	'a' или 'b' и не 'c'
10	'a friend in need is a friend indeed'	('a' или 'b') и не 'c'
11	'a house is not a home'	либо 'a', либо 'b' и 'c'
12	'a journey of a thousand miles begins with a single step'	'a' или (либо 'b', либо 'c')
13	'a leopard cannot change its spots'	(либо 'a', либо 'b') и 'c'
14	'a little knowledge is a dangerous thing'	(либо 'a', либо 'b') или 'c'
15	'a little of what you fancy does you good'	'a' и (либо 'b', либо 'c')
16	'a new broom sweeps clean'	либо (либо 'a', либо 'b'), либо 'c'
17	'a penny saved is a penny earned'	либо 'a', либо (либо 'b', либо 'c')
18	'enough is enough'	либо не 'a', либо 'b' и 'c'
19	'finders keepers, losers weepers'	не 'a' или (либо 'b', либо 'c')
20	'first things first'	(либо не 'a', либо 'b') и 'c'
21	'keep your chin up'	(либо не 'a', либо 'b') или 'c'
22	'keep your powder dry'	не 'a' и (либо 'b', либо 'c')
23	'let bygones be bygones'	либо (либо не 'a', либо 'b'), либо 'c'
24	'let sleeping dogs lie'	'a' и 'b' или 'c'
25	'a bad compromise is better than a good lawsuit'	'a' и 'b' и 'c'
26	'absence makes the heart grow fonder'	'a' или 'b' и 'c'
27	'a cat in gloves catches no mice'	('a' или 'b') и 'c'
28	'boys will be boys'	не 'a' или 'b' или 'c'
29	'brevity is the soul of wit'	не 'a' и 'b' и 'c'
30	'test character string'	'a' или 'b' или 'c'

Пример выполнения лабораторной работы 6.1

Рассмотрим пример решения для варианта № 30. По условию задачи символьная строка проверяется на наличие латинских символов 'A', 'B' и 'C'.

Для проверки каждого условия воспользуемся функцией FIND, для которой по первому входу будем вводить исследуемую символьную строку, а по второму – условия совпадения. При совпадении символов на выходе функции FIND появляется число символов в формате ANY_INT. Для организации перехода по условию воспользуемся блоком SEL, как показано на рис. 78. Для согласования типов данных применим функцию преобразования типов INT_TO_BOOL, выдающую значение TRUE при любом INT, не равном нулю.

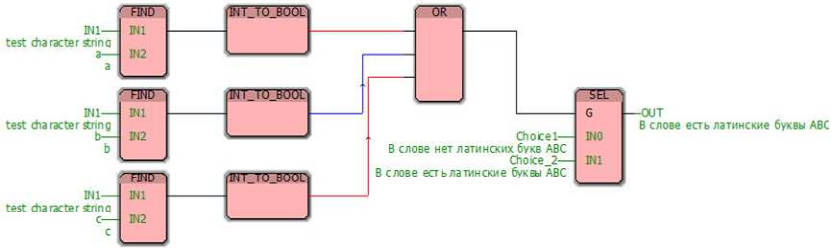


Рис. 78. Пример выполнения лабораторной работы 6.1

6.2. Лабораторная работа «Сложные условия для символьных строк»

Цель лабораторной работы: научиться работать с текстовыми строками в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram для символьных строк со сложными условиями.

Порядок выполнения лабораторной работы

Для символьной строки (если не указано условие на символьную строку, то она считается произвольной, состоящей из латинских букв с допущением пробелов) написать программу в PC WorX, чтобы производилась символьная строка, удовлетворяющая одному из условий, представленных ниже. Для отладки программы рекомендуется воспользоваться символьными строками, представленными в табл. 68, или состоящими из полного имени, отчества и фамилии студента).

Перечень условий:

1) символы *a* и *b* не стоят рядом и разделяются символами *out*;
2) символы *oo* не стоят рядом;

3) запрещено символосочетание *let*;

4) слова символьной строки, начинающиеся с гласного символа, размещаются в лексикографическом порядке;

5) между символами *a* стоит блок из четырёх символов;

6) не больше одной пары одинаковых символов стоят рядом;

7) не встречается символосочетание *ab*;

8) одинаковые символы не идут друг за другом;

9) все символы *a* и *b* заменяются на *c* и *d* соответственно;

10) производится подсчёт гласных символов;

11) производится подсчёт согласных символов от *b* до *m*;

12) производится подсчёт согласных символов от *n* до *z*;

13) производится подсчёт случаев, когда символ слева меньше символа, стоящего справа;

14) производится подсчёт случаев, когда символ справа меньше символа, стоящего слева;

15) в символьной строке каждое слово имеет чётное число символов;

16) в символьной строке каждое слово имеет нечётное число символов;

17) из символьной строки удалите все символы, которые являются; одной из четырёх первых букв латинского алфавита;

18) каждый гласный символ удваивается;

19) каждый согласный символ от *b* до *m* удваивается;

20) каждый согласный символ от *n* до *z* удваивается;

21) в символьной строке замените все символосочетания «qu» на «qu-qurequ»;

22) из символьной строки удалите среднюю букву, если длина слова нечетная;

23) из символьной строки удалите первый символ каждого слова;

24) в символьной строке слова располагаются по возрастанию числа символов в словах;

25) символ *d* идёт непосредственно после *a*;

26) символ *d* идёт непосредственно перед *a*;

27) в начале символьной строки стоит символ *a*;

28) в конце символьной строки стоит символ *a*;

29) на 2 месте в символьной строке стоит символ *a*;

30) первоначальная символьная строка состоит из полного имени, отчества и фамилии, получаемая символьная строка должна быть в виде Фамилия И.О. (например, символьная строка 'Виктор Николаевич Семенов' должна преобразоваться в символьную строку 'Семенов В.Н.').

Пример выполнения лабораторной работы 6.2

Рассмотрим пример решения для варианта № 30. Согласно заданию, последовательность написания имени, отчества и фамилии жестко задана,

и по умолчанию они разделены пустой строкой (пробелом). Поэтому логика преобразований строится следующим образом:

1. Для преобразования «Имя Отчество Фамилия» в «И.Отчество Фамилия» находим первую пустую строку, что дает возможность подсчитать число букв в имени (Функция FIND в линии с маркером 1). Удалим лишние буквы, заменив их одной точкой (Функция REPLACE в линии с маркером 1) (рис. 79).

2. Аналогичным образом преобразуем полученный результат в «И.О.Фамилия» (элементы в линии, отмеченной маркером 2).

3. Далее по заданию требуется поменять местами строки «И.О.» и «Фамилия» и отделить их пробелом. Для этого предварительно определяем число символов в фамилии (число символов в строке «И.О.» известно), затем функциями LEFT и RIGHT разделяем символьную строку на две.

4. Функцией CONCAT сливаем строки в обратном порядке и дополняем пробелом при помощи функции INSERT (линия маркера 3).

Реализация задачи на языке Function Block Diagram показана на рис. 79.

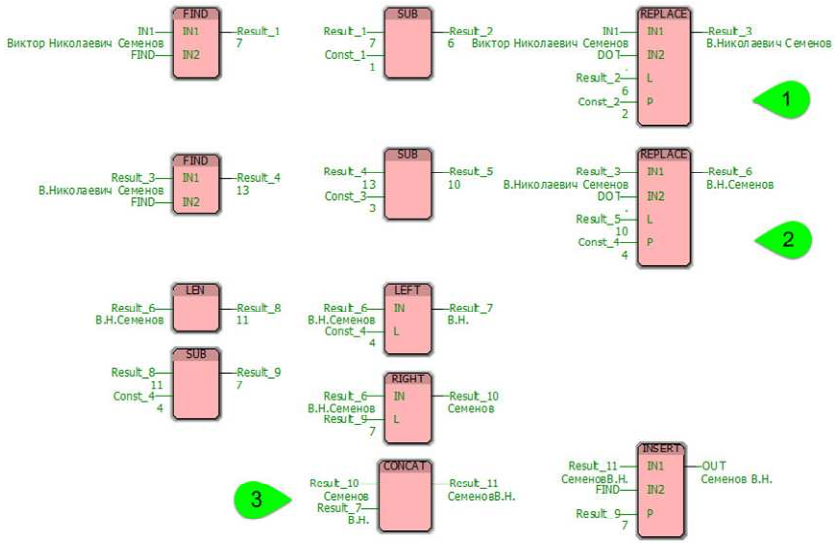


Рис. 79. Пример выполнения лабораторной работы 6.2

Тема 7. СЧЁТЧИКИ

Счетчики – это элементы, ведущие счет импульсов или циклов. С их помощью также можно организовать циклы, аналогичные циклам ***for...next***, ***while...do***, ***repeat***.

К стандартным счетчикам относятся счетчик вверх CTU, счетчик вниз CTD и реверсивный счетчик CTUD (рис. 80).

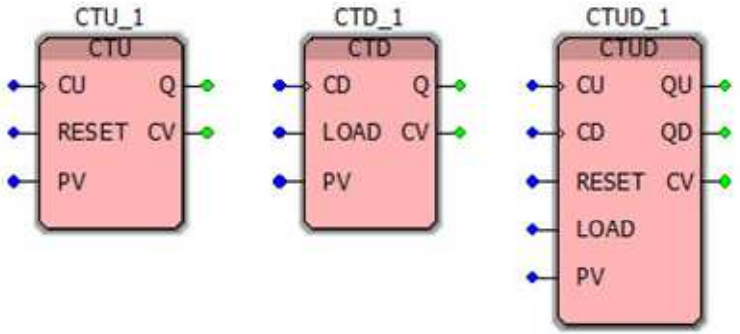


Рис. 80. Счетчики на языке Function Block Diagram в PC WorX

Счетчик CTU считает входные импульсы по переднему фронту. При появлении на входе CU переднего фронта сигнала, и если RESET = FALSE, значение CV увеличивается на 1. На входе PV задается значение счета, при достижении которого выход Q устанавливает значение TRUE, и подсчет импульсов прекращается. Если RESET = TRUE, счетчик инициализируется с нулевым значением CV. Выход Q может быть инвертирован.

Счетчик CTD производит обратный отсчет входных импульсов по переднему фронту. При появлении на входе CD переднего фронта сигнала, и если LOAD = FALSE, значение CV уменьшается на 1. По входу PV устанавливается финишное значение счета, при достижении которого выход Q устанавливает значение TRUE, и подсчет импульсов прекращается. Если LOAD = TRUE, счетчик инициализируется со значением CV, равным PV. Выход Q может быть инвертирован.

Счетчик CTUD – реверсивный. Выход CV инкрементируется на единицу по приходу переднего фронта на входе CU, и декрементируется на

единицу по переднему фронту на входе CD. Если CV = PV, выход QU устанавливается как TRUE, если CV = 0, выход QD устанавливается в TRUE. Если RESET = TRUE, счетчик инициализируется со значением 0, если LOAD = TRUE, счетчик инициализируется со значением PV. Для запуска процесса счета входы RESET и LOAD должны быть установлены как FALSE. Выходы QU и QD можно инвертировать.

В таблице 69 представлено описание входных и выходных параметров для счетчиков.

Таблица 69

Параметр	Тип данных	Описание
1	2	3
Блок CTU		
CU	bool	Если обнаружено появление переднего фронта, CV увеличивается на 1
RESET	bool	Если TRUE, счетчик устанавливается в 0. Если FALSE, подсчет импульсов разрешен.
PV	int	Предельное значение счета
Q	bool	TRUE, если CV = PV
CV	int	Значение счета
Блок CTD		
CD	bool	Если обнаружено появление переднего фронта, CV уменьшается на 1
LOAD	bool	Если TRUE, счетчик инициализируется в PV. Если FALSE, подсчет импульсов разрешен.
PV	int	Предельное значение счета
Q	bool	TRUE, если CV = 0
CV	int	Значение счета
Блок CTUD		
CU	bool	Если обнаружено появление переднего фронта, CV увеличивается на 1
CD	bool	Если обнаружено появление переднего фронта, CV уменьшается на 1
RESET	bool	Если TRUE, счетчик устанавливается в 0. Если FALSE, подсчет импульсов разрешен.

Окончание табл. 69

1	2	3
LOAD	bool	Если TRUE, счетчик инициализируется в PV. Если FALSE, подсчет импульсов разрешен.
PV	int	Предельное значение счета
QU	bool	TRUE, если CV = PV
QD	bool	TRUE, если CV = 0
CV	int	Значение счета

Задание. Исследуйте счетчики CTU, CTD, CTUD. Для этого переместите их на рабочее поле IEC Programming Workspace и задайте произвольные входные переменные. Поместите на рабочее поле таймер TP и создайте на его основе генератор импульсов. Период импульсов задайте произвольно. В режиме отладки исследуйте логику работы счетчиков. Пример описанной процедуры показан на рис. 81.

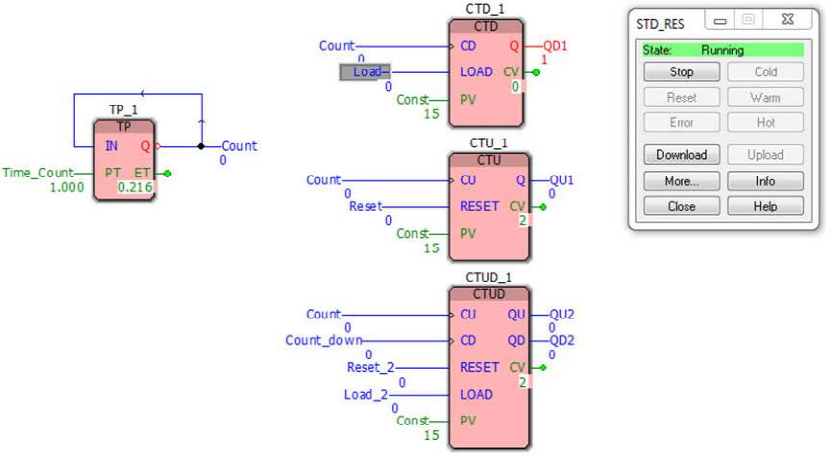


Рис. 81. Исследование счетчиков в PC WorX

7.1. Лабораторная работа «Реализация циклов с использованием счётчиков»

Цель лабораторной работы: научиться реализовывать циклы, разрабатывать приложения на языке программирования Function Block Diagram, использующие функциональные блоки счётчиков.

Порядок выполнения лабораторной работы

Разработайте при помощи счетчиков на языке Function Block Diagram цикл с параметрами, заданными в табл. 70. Выведите входные и выходные параметры в окно логического анализатора (программного модуля среды программирования PCWorx, предназначенного для записи и отображения последовательности цифровых сигналов).

Таблица 70

№ п/п	*k*	*l*	*m*	*n*	*p*	*Задание*
1	2	3	4	5	6	7
1	2	5	8	3	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, или выключен тумблер ***n***
2	3	6	7	4	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, и выключен тумблер ***n***
3	4	7	6	5	5	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, и тумблер ***n*** переключился ***p*** раз
4	1	0	5	6	9	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, или тумблер ***n*** переключился ***p*** раз
5	2	1	4	7	8	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, или тумблер ***n*** переключился ***p*** раз
6	3	2	3	–	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз
7	4	3	4	–	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз
8	1	4	5	7	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, и если тумблер ***n*** выключен
9	2	5	6	6	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, и если тумблер ***n*** включен
10	3	6	7	5	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** и тумблер ***n*** переключились ***m*** раз
11	4	7	8	4	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, или выключен тумблер ***n***

Продолжение табл. 70

1	2	3	4	5	6	7
12	1	0	9	3	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, и выключен тумблер ***n***
13	2	1	8	2	4	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, и тумблер ***n*** переключился ***p*** раз
14	3	2	7	1	6	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, или тумблер ***n*** переключился ***p*** раз
15	4	3	6	0	7	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, или тумблер ***n*** переключился ***p*** раз
16	1	4	5	–	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз
17	2	5	4	–	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз.
18	3	6	3	0	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, и если тумблер ***n*** выключен
19	4	7	4	1	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, и если тумблер ***n*** включен
20	1	0	5	2	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** и тумблер ***n*** переключились ***m*** раз
21	2	1	6	3	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, или выключен тумблер ***n***
22	3	2	7	4	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, и выключен тумблер ***n***
23	4	3	8	5	7	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, и тумблер ***n*** переключился ***p*** раз
24	1	4	9	6	6	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз, или тумблер ***n*** переключился ***p*** раз
25	1	0	5	–	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз

Окончание табл. 70

1	2	3	4	5	6	7
26	2	1	6	–	–	Подайте на выход OUTPUT***k*** сигнал TRUE, и перевести его в состояние FALSE, если тумблер ***l*** переключился ***m*** раз
27	3	2	7	0	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, и если тумблер ***n*** выключен
28	4	3	8	1	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, и если тумблер ***n*** включен
29	1	4	9	2	–	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** и тумблер ***n*** переключились ***m*** раз
30	2	5	8	7	5	Подайте на выход OUTPUT***k*** сигнал TRUE, если тумблер ***l*** переключился ***m*** раз, или тумблер ***n*** переключился ***p*** раз

Пример выполнения лабораторной работы 7.1

В качестве примера разберем вариант № 30 лабораторной работы 7.1. (рис. 82). Подставив числовые значения из табл. 70, задание формализуется как «Подайте на выход OUTPUT2 сигнал TRUE, если тумблер 5 переключился 8 раз, или тумблер 7 переключился 5 раз».

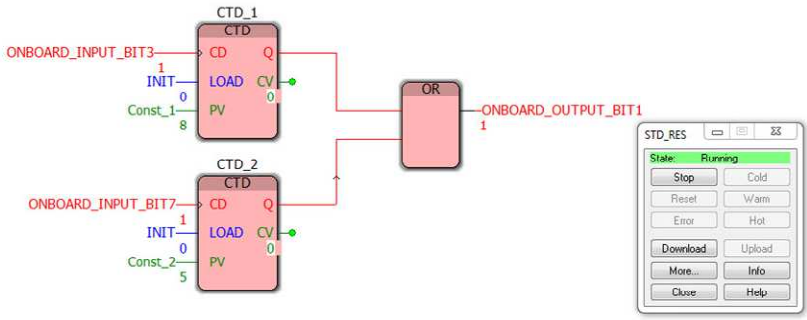


Рис. 82. Пример выполнения лабораторной работы 7.1

Для проверки двух условий используем два CTD счетчика, счетные входы которых зададим переменными ONBOARD_INPUT_BIT3 (соответствует тумблеру 5) и ONBOARD_INPUT_BIT7 (соответствует тумблеру 7). Константами Const_1 и Const_2 зададим требуемое число переключений для каждого тумблера. Выходы счетчиков объединим условием ИЛИ (элемент OR), выход которого зададим переменной ONBOARD_OUTPUT_BIT1, что

соответствует физическому выходу ПЛК OUTPUT2. Перед началом работы счетчики инициализируем (переменная INIT).

Для работы с логическим анализатором в режиме отладки поместим в окно Logic Analyser переменные ONBOARD_INPUT_BIT3, ONBOARD_INPUT_BIT7 и ONBOARD_OUTPUT_BIT1. В меню Trigger Configuration необходимо установить режим Continuous recording, что позволит работать с анализатором в режиме реального времени. Кнопкой Start recording запускаем работу логического анализатора, и проверяем срабатывание по заданному условию (на рис. 83 маркером 8 отмечено восьмое переключение тумблера 5, в то время, как тумблер 7 ещё не переключился 5 раз).

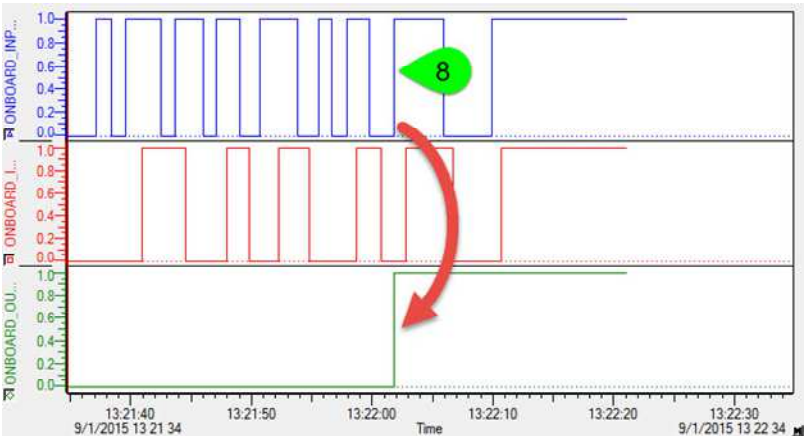


Рис. 83. Окно логического анализатора для примера решения варианта № 30 лабораторной работы 7.1

7.2. Лабораторная работа «Решение задачи освещения и подсчета занятых мест в гараже с использованием счётчиков»

Цель лабораторной работы: научиться разрабатывать приложения на языке программирования Function Block Diagram на примере решения задачи освещения и подсчёта занятых мест в гараже.

Порядок выполнения лабораторной работы

Рассмотрим задачу освещения и подсчета занятых мест в гараже. В качестве исходных данных примем наличие двух датчиков фиксации проезда автомобиля – одного снаружи ворот и одного внутри ворот. Предполагается, что при въезде автомобиля в гараж сначала срабатывает внешний датчик, а затем внутренний, а при выезде наоборот. Режим движения

автомобилей таков, что датчики срабатывают с интервалом 0,5 секунд и не должны срабатывать одновременно. Число машино-мест в гараже – 15, освещение включено, если в гараже есть машины, и выключается через 5 секунд после выезда последней машины.

Пример выполнения лабораторной работы 7.2

Рассмотрим выполнение этого задания на языке Function Block Diagram (рис. 84).

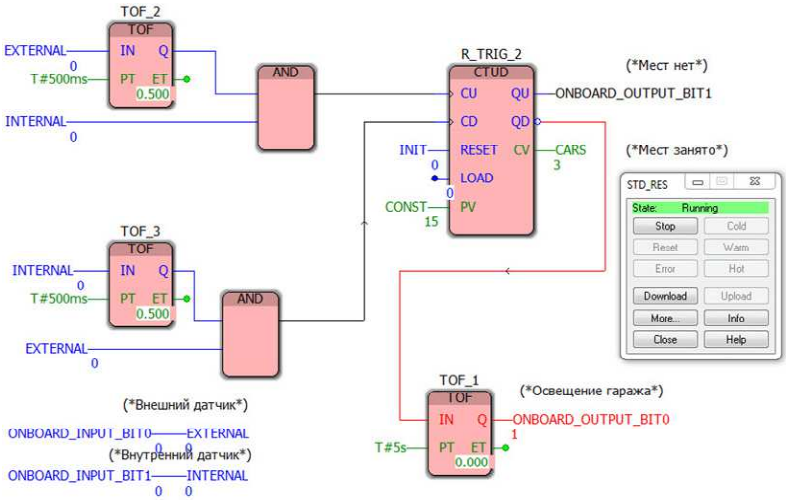


Рис. 84. Пример выполнения лабораторной работы 7.2

На рисунке при помощи таймеров TOF_2, TOF_3 и логических элементов AND организовано разделение машин на въезжающие и выезжающие. Так на вход таймера CU подается логическая единица, в случае срабатывания внешнего датчика, а затем в промежутке времени 0,5 секунд – внутреннего (для входа CD – наоборот, TRUE, если сначала срабатывает внутренний, а затем внешний датчики). Для удобства управления, сигналы датчиков EXTERNAL и INTERNAL заведены на тумблеры 1 и 2 наборного поля. В счетчике CTUD подсчитывается число машин в гараже, при этом на выход OUTPUT1 подается сигнал об отсутствии мест, а на выход OUTPUT2 – сигнал освещения гаража.

Задание. Внесите изменения в разобранную в примере программу подсчета машино-мест и управления гаражным освещением для гаража с двумя воротами (и, соответственно, с четырьмя датчиками) для въезда и выезда автотранспорта.

ТЕМА 8. ТРИГГЕРЫ

Триггер – это цифровой элемент, обладающий способностью длительно находиться в одном из двух устойчивых состояний и чередовать эти состояния под воздействием внешних сигналов. Отличительной особенностью триггера как функционального устройства является свойство запоминания двоичной информации. Под памятью триггера подразумевают способность оставаться в одном из двух состояний и после прекращения действия переключающего сигнала. Приняв одно из состояний за «1», а другое за «0», можно считать, что триггер хранит (запоминает) один разряд числа, записанного в двоичном коде.

В PC WorX триггеры представлены двумя бистабильными асинхронными RS-триггерами с приоритетом – программными блоками RS и SR (рис. 85).

В программных блоках SR и RS выход Q1 устанавливается входом SET1 и сохраняет свое значение даже в случае сброса входа SET1 в состояние FALSE. Выход Q1 = FALSE, если вход RESET = TRUE. Отличие блоков заключается в том, что блок SR имеет приоритет установки, и если оба входа (SET1 и RESET) установлены в TRUE, выход Q1 устанавливается входом SET1 в состояние TRUE; а блок RS имеет приоритет сброса, и если SET = RESET1 = TRUE, то Q1 устанавливается входом RESET1 в состояние FALSE. Все параметры блоков RS и SR можно инвертировать.

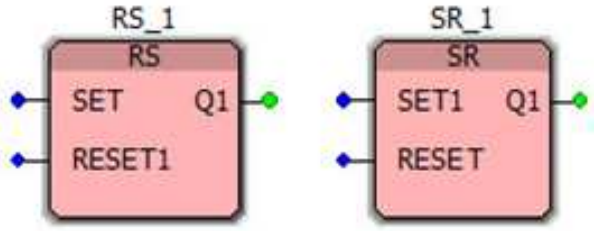


Рис. 85. Триггеры на языке Function Block Diagram в PC WorX

Блокам SR, RS необходимо объявлять уникальные имена в листе переменных POU, где блок будет использоваться.

В таблице 71 представлено описание входных и выходных параметров для триггеров.

Таблица 71

Параметр	Тип данных	Описание
Блок SR		
SET1	bool	Если TRUE, Q1 = 1 (с приоритетом установки)
RESET	bool	Если TRUE, Q1 = 0
Q1	bool	Выходное значение
Блок RS		
SET	bool	Если TRUE, Q1 = 1
RESET1	bool	Если TRUE, Q1 = 0 (с приоритетом сброса)
Q1	bool	Выходное значение

Задание. Исследуйте триггеры SR и RS. Для этого переместите их на рабочее поле IEC Programming Workspace и задайте входные переменные типа BOOL. Изменяя в режиме отладки входные параметры, постройте таблицу истинности для исследуемых триггеров. Пример описанной процедуры показан на рис. 86.

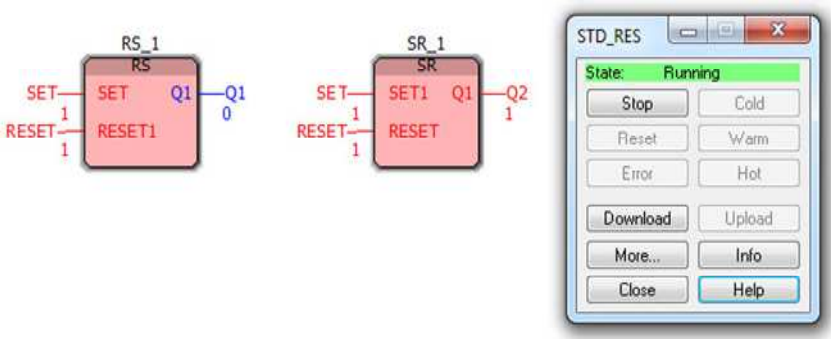


Рис. 86. Исследование триггеров в PC WorX

8.1. Лабораторная работа «Аналог синхронного RS-триггера»

Цель лабораторной работы: научиться реализовывать в среде программирования PC WorX v.6.30 триггеры, разрабатывать приложения на языке программирования Function Block Diagram, использующие программные блоки триггеров.

Порядок выполнения лабораторной работы

При помощи логических элементов и асинхронного RS триггера создайте аналог синхронного RS-триггера. Таблица истинности синхронного RS-триггера приведена в табл. 72.

Таблица 72

Таблица истинности синхронного RS-триггера

C	R	S	Q(t)	Q(t+1)	Пояснения
0	*x*	*x*	0	0	Режим хранения информации
0	*x*	*x*	1	1	
1	0	0	0	0	Режим хранения информации
1	0	0	1	1	
1	0	1	0	1	Режим установки единицы $S = 1$
1	0	1	1	1	
1	1	0	0	0	Режим записи нуля $R = 1$
1	1	0	1	0	
1	1	1	0	*	$R = S = 1$ (в RS-триггере запрещенная комбинация).
1	1	1	1	*	

8.2. Лабораторная работа «Делитель частоты»

Цель лабораторной работы*:* научиться реализовывать в среде программирования PC WorX v.6.30 схемы делителей частоты на триггерах и счетчиках.

Порядок выполнения лабораторной работы

Разработайте при помощи триггеров и/или счетчиков на языке Function Block Diagram программу для делителя частоты. Тактовая частота $F1$, результирующая частота $F2$ и фронт срабатывания заданы в табл. 73, форма сигнала задающего генератора и форма выходного сигнала – меандр. В качестве задающего генератора предлагается использовать импульсный таймер. Продемонстрируйте работу программы на логическом анализаторе.

Таблица 73

№ п/п	F1, Гц	F2, Гц	фронт	№ п/п	F1, Гц	F2, Гц	фронт
1	16	4	передний	16	1,6	0,4	задний
2	2,5	1,25	задний	17	10	5	передний
3	5	1,25	передний	18	6	1,5	задний
4	10	2,5	задний	19	20	5	передний
5	12	6	передний	20	18	4,5	задний
6	10	5	задний	21	8	2	передний
7	1	0,25	передний	22	1	0,5	задний
8	0,5	0,125	задний	23	0,8	0,2	передний
9	1,2	0,3	передний	24	14	3,5	задний
10	12	3	задний	25	0,6	0,15	передний
11	3	1,5	передний	26	0,4	0,1	задний
12	6	1,5	задний	27	12	6	передний
13	4	1	передний	28	1,2	0,6	задний
14	3	1,5	задний	29	12	3	передний
15	0,3	0,15	задний	30	2,5	1,25	передний

Для работы над заданием необходимо дополнительно ознакомиться с логикой действия функциональных блоков R_TRIG и F_TRIG (табл. 74).

Таблица 74

Параметр	Тип данных	Описание
Блок F_TRIG		
CLK	bool	Детектирует задний фронт
Q	bool	Если обнаружено появление заднего фронта, Q меняет значение с FALSE на TRUE.
Блок R_TRIG		
CLK	bool	Детектирует передний фронт
Q	bool	Если обнаружено появление переднего фронта, Q меняет значение с FALSE на TRUE.

Функциональные блоки R_TRIG и F_TRIG детектируют фронт входного сигнала. В блоке R_TRIG выход Q меняет свое значение с FALSE на TRUE по переднему фронту сигнала на входе CLK. При первом вызове блока выход Q = FALSE до появления первого переднего фронта. В блоке F_TRIG выход Q меняет свое значение с FALSE на TRUE по заднему фронту сигнала на входе CLK. При первом вызове блока выход Q = FALSE до появления первого заднего фронта.

Пример выполнения лабораторной работы 8.2

Рассмотрим выполнение лабораторной работы 8.2. на примере варианта № 30 (рис. 87).

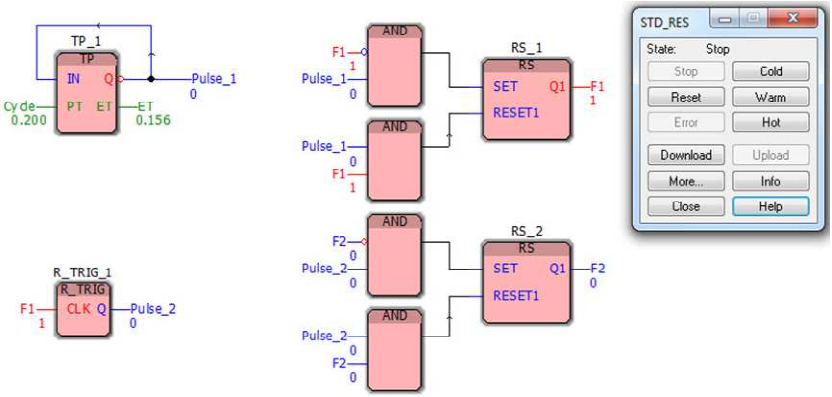


Рис. 87. Пример выполнения лабораторной работы 8.2

На импульсном таймере TP_1 при помощи введения обратной связи с инверсного выхода реализован генератор единичных импульсов с периодом следования, задаваемым переменной *Cycle*. Для формирования меандра используем триггер – блок SR. На выходе триггера получаем задающую частоту $F1 = \frac{1}{2T}$, где T – период следования импульсов таймера TP_1 (в примере – переменная *Cycle*).

Согласно заданию, частота следования импульсов на выходе должна быть вдвое меньше входной, фронт срабатывания – передний. Для этого на элементах R_TRIG_1, RS_2 и двух элементах AND сформирован асинхронный *T*-триггер, переключающийся по каждому импульсу на входе *F*1. Работу описанного логического устройства удобно и наглядно можно представить в графическом виде в окне графического анализатора (рис. 88). Маркером 1 на рис. 88 отмечен выход *Pulse*_1, маркером 2 – *F*1, 3 – *Pulse*_2, маркером 4 – выход *F*2.

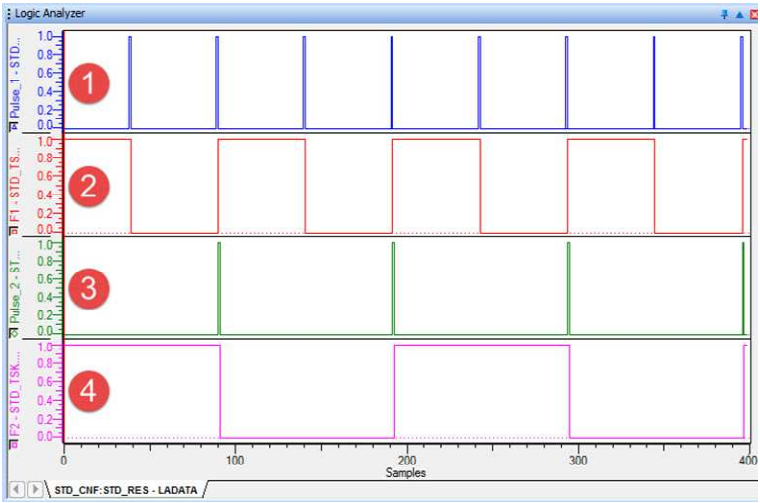


Рис. 88. Переменные Pulse_1, F1, Pulse_2 и F2 в окне логического анализатора

Для работы с логическим анализатором из рабочего поля IEC Programming Workspace в режиме отладки буксировкой левой кнопкой мыши поместите переменные, необходимые для анализа, в окно Logic Analyser в правой части рабочего поля (маркер 1 рис. 89). Открыть окно Logic Analyser можно из меню View → Logic Analyser или сочетанием клавиш ALT+F11. Для запуска логического анализатора нажмите кнопку Start Recording (маркер 2 рис. 89), либо кнопку F11. Изменять параметры процесса регистрации сигналов можно в меню Trigger Configuration (маркер 3 рис. 89) или сочетанием клавиш SHIFT+F11.

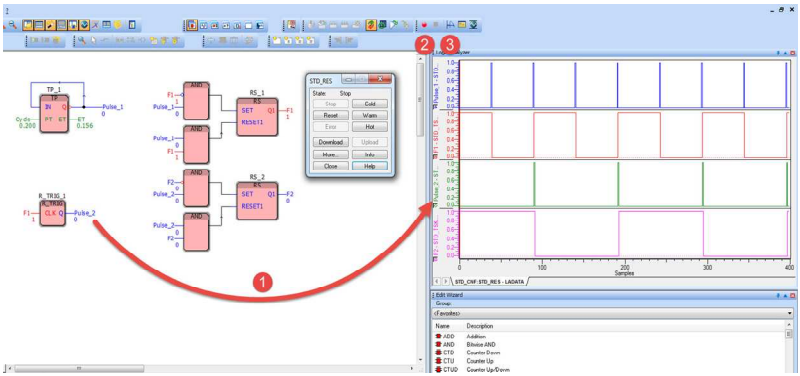


Рис. 89. Назначение переменных в окне логического анализатора

ТЕМА 9. ТАЙМЕРЫ

Под таймером будем понимать средство обеспечения задержек и измерения времени программными средствами. Базовые блоки, реализующие такие задержки времени на языке Function Block Diagram в PC WorX, показаны на рис. 90 и состоят из трех элементов: таймер с задержкой по включению TON (Timer_on_delay), таймер с задержкой по выключению TOF (Timer_off_delay) и импульсный таймер TP (Timer_pulse) (рис. 90).

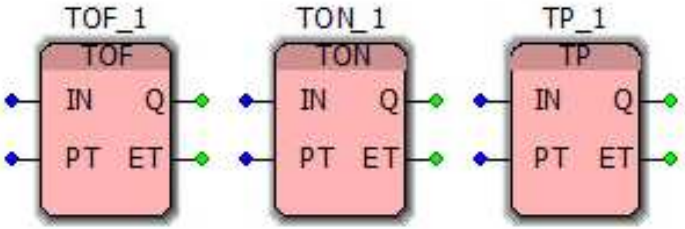


Рис. 90. Таймеры на языке Function Block Diagram в PC WorX

Таймер TON обеспечивает задержку по включению. Если на входе IN сигнал меняет свое значение с 0 (False) на 1 (True), переключение сигнала на выходе Q задерживается на время, заданное на входе PT. Прошедшее с момента изменения сигнала на входе время отображается на выходе ET. Вход IN и выход Q можно инвертировать.

Таймер TOF реализует задержку по выключению. Если на входе IN сигнал меняет свое значение с 1 (True) на 0 (False), выключение сигнала на выходе Q задерживается на время, заданное на входе PT. После истечения времени, заданного на PT, сигнал на выходе Q меняет свое значение с 1 (True) на 0 (False). Прошедшее время отображается на выходе ET. Вход IN и выход Q также можно инвертировать.

Таймер TP формирует импульс на выходе Q с длительностью, заданной по входу PT, как только на входе IN сигнал меняет свое значение с 0 (False) на 1 (True). При этом, даже если сигнал 1 (True) на входе IN длится дольше, чем время на PT, на длительность выходного импульса это не повлияет. Также как и для других таймеров, прошедшее время отображается на выходе ET, а вход IN и выход Q также можно инвертировать.

Всё вышесказанное формализовано в табличном виде и представлено в табл. 75.

Таблица 75

Параметр	Тип данных	Описание
TOF		
IN	bool	Старт задержки времени по заднему фронту сигнала
PT	time	Предустановка временного интервала для задержки
Q	bool	TRUE if IN = TRUE and ET < PT. FALSE if IN = FALSE and ET >= PT.
ET	time	Индикация прошедшего времени
TON		
IN	bool	Старт задержки времени по переднему фронту сигнала
PT	time	Предустановка временного интервала для задержки
Q	bool	TRUE if IN = TRUE and ET >= PT. FALSE if IN = FALSE or ET < PT.
ET	time	Индикация прошедшего времени
TP		
IN	bool	Старт импульса по переднему фронту сигнала
PT	time	Предустановка временного интервала для импульса
Q	bool	TRUE if IN = TRUE and ET < PT. FALSE if IN = FALSE and ET >= PT.
ET	time	Индикация прошедшего времени

Задание. Исследуйте таймеры TON, TOF и TP. Для этого переместите их на рабочее поле IEC Programming Workspace и задайте произвольные входные переменные. В режиме отладки присвойте этим переменным числовые значения и убедитесь в правильности выдачи результата. Пример описанной процедуры показан на рис. 91.

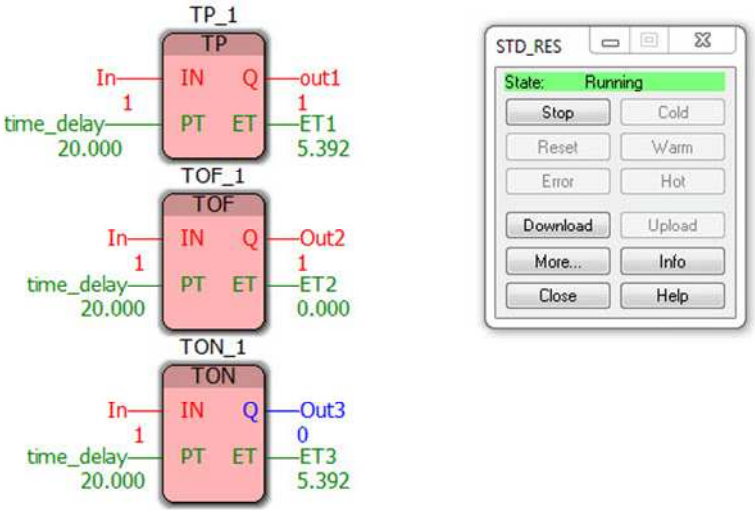


Рис. 91. Исследование таймеров в PC WorX

9.1. Лабораторная работа «Система управления пешеходным светофором»

Цель лабораторной работы: научиться реализовывать в среде программирования PC WorX v.6.30 систему управления пешеходным светофором, разрабатывать приложения на языке программирования Function Block Diagram, использующие функциональные блоки таймеров.

Порядок выполнения лабораторной работы

Разработайте при помощи таймеров на языке Function Block Diagram схему управления пешеходным светофором со следующими параметрами, заданными в таблице: длительность красного сигнала *n* секунд, длительность зелёного сигнала *m* секунд. Сигналы сменяются поочередно и никогда не горят одновременно. Задействуйте для красного и зелёного сигналов физические выходы ПЛК с номерами *k* и *l* соответственно, заданными в табл. 76.

Таблица 76

№ п/п	*n*	*m*	*k*	*l*	№ п/п	*n*	*m*	*k*	*l*
1	2	3	4	5	6	7	8	9	10
1	14	4	1	3	16	15	4	4	2

Окончание табл. 76

1	2	3	4	5	6	7	8	9	10
2	13	9	3	2	17	10	3	2	4
3	25	3	2	3	18	12	3	3	4
4	24	5	1	4	19	13	8	4	1
5	24	8	2	3	20	14	8	1	4
6	10	9	4	3	21	10	7	3	2
7	12	10	1	4	22	17	8	1	2
8	11	9	3	1	23	8	4	2	4
9	21	9	2	4	24	11	7	4	1
10	8	8	3	2	25	15	10	4	1
11	17	3	4	2	26	17	4	2	1
12	20	4	1	2	27	16	7	1	2
13	12	7	3	1	28	14	8	4	3
14	12	10	2	3	29	11	9	2	4
15	11	8	1	3	30	15	5	3	2

Предложите свое техническое решение задания, отличное от предложенного в примере (например, реализуйте схему без таймера TON, без таймера TOF, схему с двумя таймерами, и т.д.).

Пример выполнения лабораторной работы 9.1

Один из примеров реализации лабораторной работы 9.1. показан на рис. 92.

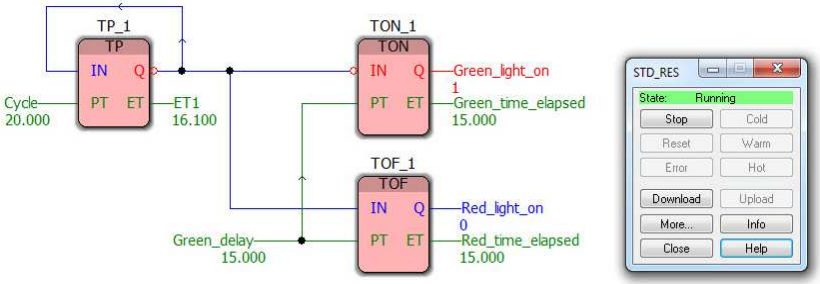


Рис. 92. Пример выполнения лабораторной работы 9.1

На импульсном таймере TP_1 при помощи введения обратной связи с инверсного выхода реализован генератор единичных импульсов с перио-

дом следования, задаваемым переменной Cycle. В примере переменная Cycle задает длительность одного рабочего цикла светофора, т.е. длительность свечения красного сигнала плюс длительность свечения зеленого. Единичный импульс, поступая на вход таймера TOF_1, формирует на выходе Q (переменная Red_light_on, которая соответствует работе красного сигнала светофора) логическую единицу на период времени, задаваемый переменной Green_delay. По истечении этого времени Red_light_on сбрасывается в 0, одновременно с этим срабатывает выход Q таймера TON_1 (отсрочка срабатывания задана той же переменной Green_delay), устанавливая на выходе логическую единицу до прихода следующего единичного импульса с таймера TP_1.

Для того, чтобы назначить выходным переменным физические выходы ПЛК (Output1 – Output 4), в свойствах переменной в выпадающем меню имени выберите ONBOARD_OUTPUT_BIT0 – ONBOARD_OUTPUT_BIT3 (выберите требуемый).

9.2. Лабораторная работа «Модификация системы управления пешеходным светофором»

Цель лабораторной работы: научиться основам программирования в среде программирования PC WorX v.6.30 на примере системы управления пешеходным светофором, разрабатывать приложения на языке программирования Function Block Diagram, использующие таймеы.

Порядок выполнения лабораторной работы

Модернизируйте ранее разработанную схему управления пешеходным светофором. Длительность непрерывного свечения красного сигнала *n* секунд, длительность непрерывного свечения зелёного сигнала *m* секунд, длительность мигания зеленого сигнала светофора *x* секунд с частотой мигания *f* герц. Сигналы сменяются поочередно в следующей последовательности: длительный красный, длительный зелёный, мигающий зелёный, и никогда не горят одновременно. Задействуйте для красного и зелёного сигналов физические выходы ПЛК с номерами *k* и *l* соответственно, заданными в табл. 77.

Таблица 77

№ п/п	*n*	*m*	*x*	*f*	*k*	*l*	№ п/п	*n*	*m*	*x*	*f*	*k*	*l*
1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	40	14	3	1	1	2	16	34	15	6	0,5	3	4
2	23	7	5	1,5	1	4	17	27	8	5	1	4	1

Окончание табл. 77

1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	12	9	2	2	1	2	18	21	8	5	1,5	2	1
4	19	6	6	0,5	4	2	19	35	9	4	2	4	3
5	13	7	3	1	1	3	20	21	6	3	0,5	2	1
6	16	8	4	1,5	2	4	21	27	14	6	1	3	1
7	24	13	6	2	1	2	22	25	3	3	1,5	2	4
8	25	9	5	0,5	4	2	23	30	9	4	2	1	3
9	10	6	2	1	2	4	24	44	17	5	0,5	4	2
10	24	7	3	0,5	4	2	25	15	10	4	2	4	3
11	20	8	3	1	3	2	26	15	8	4	0,5	4	2
12	43	15	5	1,5	3	4	27	18	9	5	1	3	4
13	48	17	6	2	1	3	28	16	6	3	1,5	2	1
14	15	7	3	0,5	1	2	29	18	5	2	2	1	4
15	25	9	6	1,5	3	4	30	40	17	3	1	3	4

Пример выполнения лабораторной работы 9.2

Один из примеров реализации в PC WorX программы лабораторной работы 9.2. показан на рис. 93.

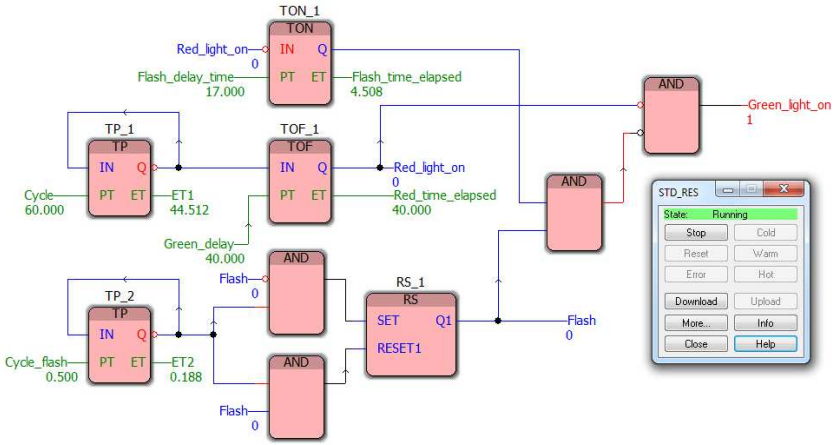


Рис. 93. Пример выполнения лабораторной работы 9.2

Аналогично предыдущему заданию, на импульсном таймере TP_1 при помощи введения обратной связи с инверсного выхода реализован генератор единичных импульсов с периодом следования, задаваемым переменной Cycle. В примере переменная Cycle задает длительность одного рабочего цикла светофора.

Единичный импульс, поступая на вход таймера TOF_1, формирует на выходе Q (переменная Red_light_on, которая соответствует работе красного сигнала светофора) логическую единицу на период времени, задаваемый переменной Green_delay. По истечении этого времени Red_light_on сбрасывается в 0, одновременно с этим запуская таймер TON_1 (отсрочка срабатывания таймера TON_1 обозначена переменной Flash_delay_time, которая задаёт длительность непрерывного свечения зелёного сигнала светофора), устанавливая по истечении времени Flash_delay_time на выходе логическую единицу.

Таймером TP_2, триггером RS_1 и двумя логическими элементами AND формируется меандр с периодом, равным удвоенному значению переменной Cycle_flash. Для приведённого примера Cycle_flash = 0.5, что соответствует частоте 1 Гц.

Логическими элементами AND, формирующими выход Green_light_on, задаются условия, разрешающие мигание зелёного сигнала светофора: отсутствие сигнала на Red_light_on и окончание времени действия таймера TON_1.

Для того, чтобы назначить выходным переменным физические выходы ПЛК (Output1 – Output 4), в свойствах переменной в выпадающем меню имени выберите ONBOARD_OUTPUT_BIT0 – ONBOARD_OUTPUT_BIT3 (выберите требуемый).

9.3. Лабораторная работа «Реализация ШИМ-сигналов с использованием таймеров»

Цель лабораторной работы: научиться реализовывать в среде программирования PC WorX v.6.30 приложения на языке программирования Function Block Diagram для ШИМ-сигналов.

Порядок выполнения лабораторной работы

Сформируйте при помощи таймеров ШИМ-сигнал, управляемый потенциометром ILC 131 Starterkit. При изменении сигнала на аналоговом входе необходимо изменять коэффициент заполнения выходных импульсов в диапазоне от ***n*** % до ***m*** % (табл. 78). Период ШИМ равен ***k*** секунд. Диапазон регулирования потенциометра – ***l***. Для отображения процесса управления используйте логический анализатор.

Таблица 78

№ п/п	*n*	*m*	*k*	*l*	№ п/п	*n*	*m*	*k*	*l*
1	20	55	8	4 – 9	16	55	90	0,3	3 – 8
2	20	75	0,6	1 – 7	17	35	45	8	4 – 7
3	25	60	6	2 – 10	18	25	60	0,4	2 – 8
4	55	60	0,1	0 – 4	19	55	80	7	1 – 6
5	10	75	3	2 – 6	20	10	55	0,5	5 – 10
6	45	95	0,5	5 – 10	21	55	70	5	0 – 5
7	25	50	5	0 – 7	22	10	65	0,7	4 – 10
8	30	65	0,4	2 – 9	23	35	95	1	1 – 5
9	15	80	6	5 – 8	24	10	60	0,5	2 – 8
10	30	55	0,1	2 – 7	25	50	75	1	0 – 3
11	55	95	4	1 – 9	26	10	55	0,4	3 – 8
12	40	95	0,7	2 – 7	27	10	85	2	3 – 9
13	45	85	4	2 – 10	28	5	75	0,3	0 – 5
14	50	95	0,2	0 – 8	29	25	50	3	5 – 6
15	35	50	0,1	3 – 9	30	1	100	3	0 – 10

Пример выполнения лабораторной работы 9.3

Рассмотрим выполнение варианта № 30 лабораторной работы 9.3. (рис. 94). На таймере TP собран генератор единичных импульсов, задающий период (*T*) следования импульсов ШИМ (переменная PERIOD_PWM) на таймере TOF_1. Вторая переменная на входе PT таймера TOF_1 непосредственно задает длительность импульса *t*, управляя которой можно регулировать коэффициент заполнения ШИМ $D = \frac{t}{T}$.

Для удобства аналоговый сигнал с потенциометра, представленный переменной ANALOG_INPUT, конвертируем из типа WORD в DINT, получая диапазон изменений переменной ANALOG_INT от 0 до 30320 при изменении диапазона регулирования потенциометра от 0 до 10.

Так как на входе PT таймера TOF_1 требуется тип данных TIME, используем блок DINT_TO_TIME, преобразующий числовую переменную в формат TIME, представленный в миллисекундах. Так, например, числовой

переменной DINT 100 в формате TIME будет соответствовать выражение T#100ms.

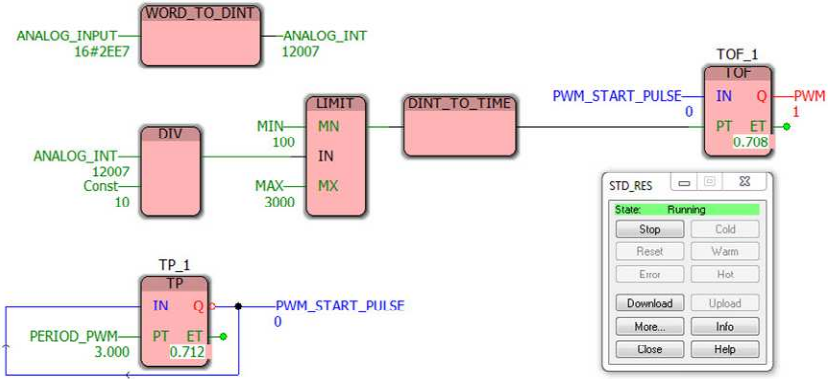


Рис. 94. Пример выполнения лабораторной работы 9.3

Диапазон регулирования скважности задаем блоком LIMIT, ограничивающим свое выходное значение диапазоном от MN до MX. Для вмещения всего диапазона регулирования потенциометра в требуемый диапазон скважности применяем дополнительно элемент DIV, нормирующий значение переменной ANALOG_INT.

Для работы с логическим анализатором из рабочего поля IEC Programming Workspace в режиме отладки поместим переменные PWM_START_PULSE и PWM в окно Logic Analyser (рис. 95).

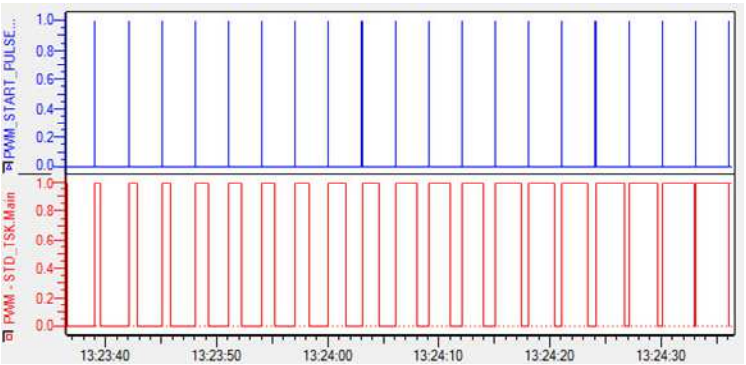


Рис. 95. Пример регулирование сигнала ШИМ для лабораторной работы 9.3

Тема 10. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ И ФУНКЦИОНАЛЬНЫХ БЛОКОВ

Для вставки пользовательской функции или функционального блока в дерево проекта необходимо выбрать папку «Logical POUs» и в контекстном меню по нажатию правую кнопку мыши выбрать Insert → Function либо Insert → Function block (как показано на рис. 96).

При создании функции необходимо помнить, что имя, выбранное для блока функции, будет также использоваться в качестве возвращаемого (и единственного выходного) значения этой функции, и тип выходного значения функции определяется через тип возвращаемого значения. Таким образом, имя функции должно соответствовать правилам создания имени переменной.

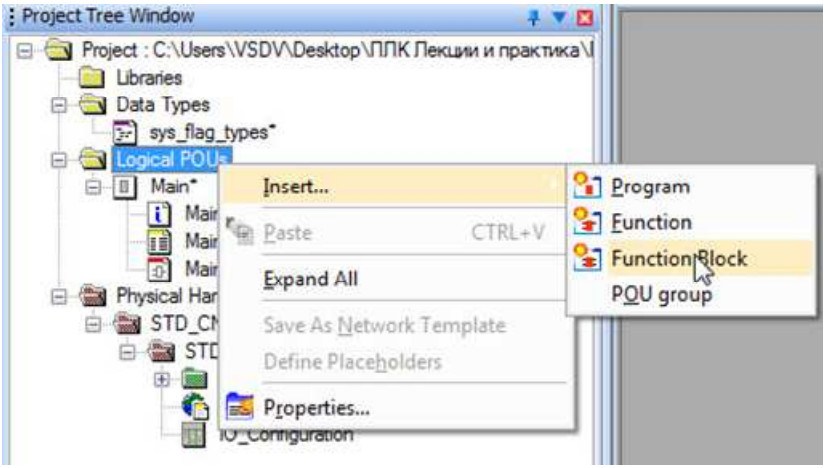


Рис. 96. Добавление функции или функционального блока в дерево проекта

После того, как функция была вставлена в дерево проекта, она перечисляется в папке «Logical POUs» с выбранным именем (например «User_Func_Sample»), при этом она имеет три основанных на имени вложенных элемента (рис. 97):

User_Func_SampleT – лист комментариев. Все записанное здесь появится в новом окне «Help of POU» при выборе в меню «Help on

FB/FU» по щелчку правой кнопкой мыши на созданном элементе в окне «Edit wizard»;

User_Func_SampleV – таблица переменных создаваемой функции;

User_Func_Sample – первый рабочий лист кода.

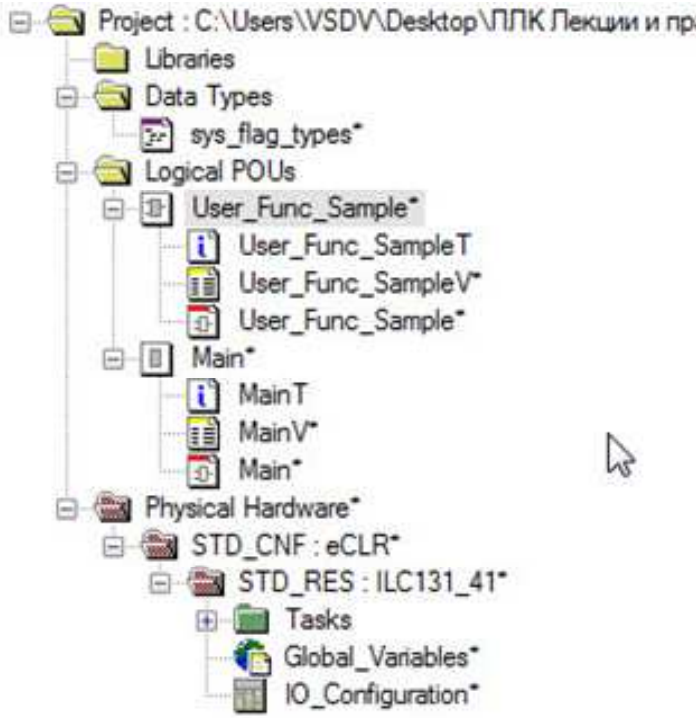


Рис. 97. Дерево проекта

Звездочки возле таблицы переменных рабочего листа показывают, что элементы не были скомпилированы. Созданная функция появляется в окне «Edit Wizard» в группе <all FUs and FBs> и группе <UserFunc>.

Сама функция редактируется в рабочем листе согласно стандартной процедуре программирования. При этом:

a. должна быть объявлена, по крайней мере, одна входная переменная вида Var_Input;
b. чтобы возвращаемое функцией значение было доступно для проекта, выходная переменная должна быть названа именем функции.

Таблица переменных компилируется при закрытии окна. Рабочие листы компилируются при закрытии либо при использовании соответст-

вующей кнопки. После успешной компиляции таблицы переменных функция может быть вызвана в другом POU.

Добавление в дерево проекта функционального блока осуществляется аналогично добавлению функции (рис. 96). При вставке функционального блока тип данных возвращаемых значений может быть не определен. После добавления функционального блока в дерево проекта он вносится в папку «Logical POUs» с выбранным именем (например «User_Block_Sample»), причём также имеет три основанных на имени вложенных элемента:

1) **User_ Block _SampleT** – лист комментариев;
2) **User_ Block_SampleV** – таблица переменных создаваемого функционального блока;
3) **User_ Block _Sample** – первый рабочий лист кода функционального блока.

Созданный функциональный блок появляется в окне «Edit Wizard» в группе <all FUs and FBs> и группе <UserBlock>.

Отличительной особенностью редактирования функционального блока, в отличие от редактирования функции, является более высокая свобода относительно проекта. Так, входные и выходные параметры могут быть объявлены в любом числе, более того, функциональные блоки учитывают доступ к глобальным переменным. Однако следует учитывать, что имя функционального блока не должно использоваться в качестве имени переменной.

Таблица переменных функционального блока компилируется при закрытии окна. Рабочие листы функционального блока компилируются при закрытии либо при использовании соответствующей кнопки. После успешной компиляции таблицы переменных функциональный блок может быть вызван в другом POU.

10.1. Лабораторная работа «Создание пользовательской функции»

Цель лабораторной работы: научиться создавать пользовательские функции в среде программирования PC WorX v.6.30, разрабатывать приложения на языке программирования Function Block Diagram, использующие пользовательские функции.

Порядок выполнения лабораторной работы

Создайте в PC WorX функцию, выполняющую функциональное преобразование из табл. 79.

Таблица 79

Варианты функционального преобразования

№ п/п	Вид функционального преобразования
1	2
1	$\phi = \left\| \frac{\cos^2 t}{1+\sin t} + \sin t \right\|^2$
2	$\varphi = \frac{2\sin^2 t}{1-\cos t} - 2\cos t$
3	$\gamma = \left(\frac{1}{1+\cos t} + \frac{1}{1-\cos t} \right)^2$
4	$\eta = \left\| \frac{1-\sin t}{\cos t} - \frac{\cos t}{1+\sin t} \right\|$
5	$\iota = \left(\frac{\cos t}{1-\sin t} - \frac{\cos t}{1+\sin t} \right)^3$
6	$\kappa = \frac{\cos^2 t - \mathrm{tg}^2 t}{\sin^2 t - \mathrm{tg}^2 t}$
7	$\lambda = \frac{2\sin t \cos t - \cos t}{1 - \sin t + \sin^2 t - \cos^2 t}$
8	$\mu = \cos^2 t - \left(\mathrm{tg}^2 t + 1 \right) \sin^2 t$
9	$\nu = \left(\cos^2 t \, \mathrm{tg}^2 t + \cos^2 t \right) \mathrm{tg}\, t$
10	$o = \frac{1 - (\cos 3t - \sin 3t)^2}{2\cos 3t \ \mathrm{tg} 3t}$
11	$\pi = \left\| \frac{\sin t + \mathrm{ctg}\, t}{1 + \sin t \ \mathrm{tg}\, t} - 2 \right\|$
12	$\theta = \frac{1+\sin t}{\cos t} \cdot \frac{1-\sin t}{\cos t} + 1$
13	$\rho = (1 + \mathrm{tg}\, t)^2 + (1 - \mathrm{tg}\, t)^2 - 1$
14	$\sigma = (\mathrm{tg}\, t + \cos t)^2 - (\mathrm{tg}\, t - \sin t)^2$
15	$\varsigma = \left(\frac{1+\sin t}{\cos t} - \mathrm{tg}\, t \right) \frac{\sin 2t}{1+\cos 2t}$

Окончание табл. 79

1	2
16	$\tau = \operatorname{tg}^2 t + \dfrac{1-\cos^2 t}{\cos^2 t \cdot \operatorname{tg}^2 t}$
17	$\upsilon = \dfrac{1-2\sin^2 t}{\sin t - \cos t} + \cos t \cos t$
18	$\omega = \dfrac{\cos t \operatorname{tg} t}{\sin^2 t} - 2\operatorname{tg} t \cos t$
19	$\xi = \operatorname{ctg}^2 t - \dfrac{1-2\cos^2 t}{1-2\sin^2 t}$
20	$\psi = \left\lvert \dfrac{\cos t \operatorname{tg} t}{\sin t - \sin^3 t} - 1 \right\rvert^3$
21	$a = \dfrac{(\sin 2t + \cos 2t)^2 - 1}{2 \sin 2t \operatorname{ctg} 2t}$
22	$b = \sin t + \dfrac{2\cos^2 t - 1}{\sin t + \cos t}$
23	$c = 2\sin t - \dfrac{\cos t - \cos^3 t}{\sin t \cos t}$
24	$d = \dfrac{1}{1+\operatorname{tg}^2 t} + \dfrac{1}{1+\cos^2 t}$
25	$\alpha = \dfrac{(\sin t + \cos t)^2 - 1}{\operatorname{tg} t - \sin t \cos t}$
26	$\beta = \dfrac{\operatorname{tg} t + 1}{1+2^t} + \dfrac{1-\cos 2t}{\sin 2t}$
27	$\chi = \dfrac{\cos t}{1+\sin t} + \operatorname{tg} t$
28	$\delta = \dfrac{\operatorname{tg} t}{\operatorname{tg} t + \operatorname{tg} t} - \dfrac{1+\cos 2t}{1-\cos 2t}$
29	$\varepsilon = \dfrac{\operatorname{tg} t}{\operatorname{tg}^2 t - 1} \cdot \dfrac{1-\cos^2 t}{t}$
30	$i = \mathrm{Im} \cdot \sin(\Omega \cdot t + \varphi)$

Таблица 80

Варианты функционального преобразования

№ п/п	Вид функционального преобразования
1	2
1	$AB \oplus \overline{B\overline{AC} \oplus BC \oplus A\overline{\overline{C}}} \vee BA$
2	$BA \vee CAB \vee \overline{\overline{A \oplus B\overline{C}} \oplus (AB \vee C)} \oplus \overline{CB}$
3	$\left(A\overline{BC} \oplus BC \vee \overline{A \oplus B}\right)A \vee \overline{BC \oplus A}$
4	$\overline{A\overline{B}} \oplus BC \vee \left(\overline{AC \oplus B\overline{\overline{C}}} \wedge B\right) \oplus A \vee B\overline{C}$
5	$B\overline{C} \vee \overline{AB} \oplus \left(AB \oplus \overline{B}C\right) \vee \overline{\overline{AC \oplus CB} \wedge AB}$
6	$A\overline{BC \vee AC} \oplus \overline{(AB \oplus C)} \vee \overline{AC} \vee B \oplus CA$
7	$\overline{AB \vee CA} \oplus \overline{ACB \oplus C} \wedge \left(AC \oplus AB \vee \overline{B}\right)$
8	$A \vee \overline{BC} \oplus \left(AC \vee \overline{BA \oplus C}\right) \oplus A\overline{C} \vee ABC$
9	$\overline{AB \oplus A} \oplus \overline{ACB \vee C} \wedge \left(A \oplus \overline{ACB \vee \overline{\overline{C}}}\right)$
10	$\overline{AB \vee C} \oplus CB \oplus BC \wedge \overline{(AB \oplus C)} \vee \overline{AC}$
11	$\left(A \vee \overline{BC} \oplus AC \vee \overline{BA \oplus C}\right) \oplus A\overline{C} \vee ABC$
12	$\overline{AB \oplus A} \oplus \left(\overline{ACB \vee C} \wedge A \oplus \overline{ABC \vee \overline{\overline{C}}}\right)$
13	$\overline{A \vee BC} \wedge C \oplus \left(BC \wedge \overline{AB \oplus C \vee CA} \oplus A\right)$
14	$\overline{B \oplus CA} \wedge \left(\overline{C \oplus AB} \vee \overline{B}\right) \oplus CB \vee \overline{CA}$
15	$\overline{A\overline{C} \vee B} \oplus \left(\overline{\overline{A} \oplus BC}\right) \oplus BAC$
16	$A \vee \overline{\overline{B}C} \oplus \left(C \vee \overline{B \oplus AC}\right) \oplus \overline{AB \vee C}$
17	$AB \vee \overline{C \oplus CA} \oplus \overline{ACB \vee A} \oplus (C \vee AB)$
18	$B\overline{AC} \vee AC \oplus B\overline{C} \oplus \overline{A \vee BA \oplus B\overline{C}A}$
19	$BA \oplus \left(AB \oplus \overline{B \vee AC}\right)A \vee \overline{BC \oplus AC}$
20	$\left(AC \vee \overline{C \oplus AB}\right)B \vee A\overline{AC \oplus C \vee BC} \oplus B$

Окончание табл. 80

1	2
21	$\left(\overline{AC \vee BA} \oplus \left(\overline{AB \oplus AC}\right) \vee AB \oplus C\right.$
22	$\overline{AC \oplus B} \vee \left(CA \oplus \overline{B \wedge AC}\right) A \oplus BC$
23	$\overline{B}A \vee \left(\overline{CA \oplus AC} \vee A\right) \oplus ABC \oplus \overline{\overline{C} \vee BA}$
24	$\overline{A}C \oplus \overline{B \vee C} \vee \overline{AC \oplus B} \oplus AB \wedge \left(C \vee \overline{A}\right)$
25	$\overline{AB} \oplus \left(CBA \wedge \overline{A\overline{C} \oplus BC}\right) \oplus \overline{A}BC \vee A$
26	$\left(ABC \oplus \overline{AB \vee CB}\right) \vee C \oplus AB \oplus \overline{B}C$
27	$CB \vee \overline{AB \oplus \left(\overline{BC}A \oplus AB\right)} \oplus CA \vee \overline{A}$
28	$\overline{AB}\left(\overline{CBA} \oplus \overline{A\overline{C} \oplus AC}\right) \oplus \overline{BC}A$
29	$CA \oplus \overline{CAB \vee \left(\overline{B}A \oplus CB\right)} \oplus CA$
30	$AB \vee \overline{C\overline{A} \oplus CA} \oplus \overline{ACB \vee A} \oplus \left(C \vee AB\right)$

Пример выполнения лабораторной работы 10.1

Разберем выполнение задания на примере варианта № 30 из табл. 79. Добавим функцию в дерево проекта, как показано на рисунке 96. В появившемся окне «Insert» запишем имя функции (User_Func_Sample**)**, выберем тип возвращаемых данных и язык программирования (рис. 98).

В рабочем листе для кода «User_Func_Sample» создадим необходимые переменные и функциональные преобразования для них (рис. 99), а в листе комментариев «User_Func_SampleT» пометим, какие параметры функция принимает, а какие – возвращает. Скомпилируем проект, найдем новую функцию «User_Func_Sample» в окне «Edit Wizard» в группе <UserFunc> и поместим её в лист для кода «Main». Исследовав функцию, как показано на рисунке 100, т.е. задавая в режиме отладки входные параметры и проверяя полученный результат, можно убедиться в её адекватности тестовым входным параметрам. Выбрав по щелчку правой кнопки мыши созданную функцию в окне «Edit Wizard» из выпадающего меню строку «Help on FB/FU», можно вызвать окно с ранее записанными комментариями «Help of POU User_Func_Sample» (рис. 101).

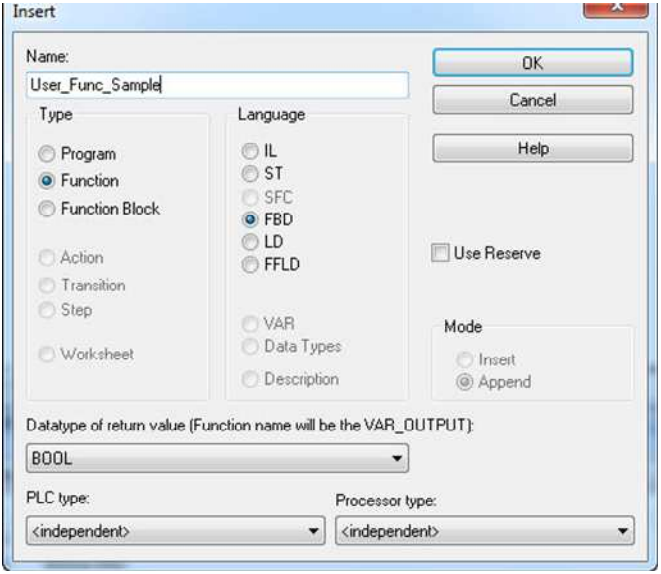


Рис. 98. Окно «Insert»

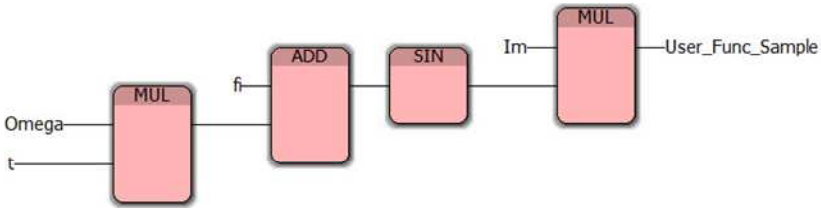


Рис. 99. Лист для кода «User_Func_Sample»

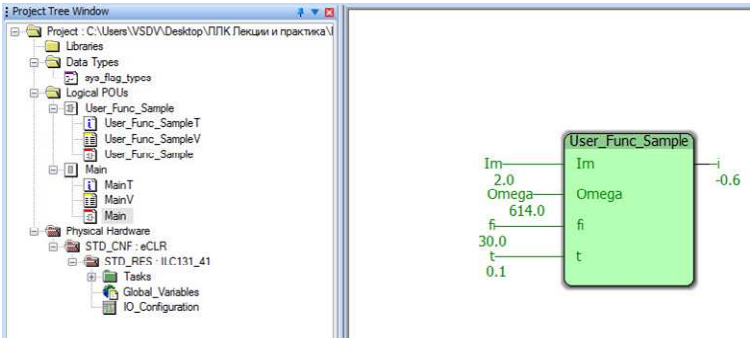


Рис. 100. Тестирование созданной функции

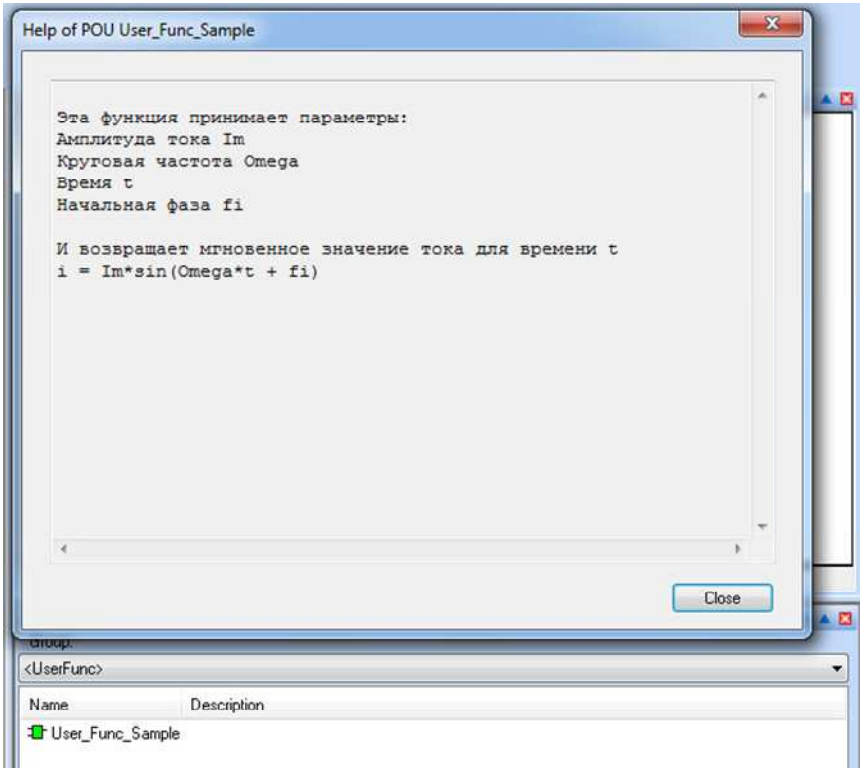


Рис. 101. Окно «Help of POU User_Func_Sample»

10.2. Лабораторная работа «Создание функционального блока»

Цель лабораторной работы: научиться создавать функциональные блоки в среде программирования PC WorX v.6.30.

Порядок выполнения лабораторной работы

Создайте в PC WorX функциональный блок, выполняющий действия, заданные в табл. 81. Протестировать созданные функциональные блоки.

Таблица 81

Варианты задания функционального блока

№ п/п	Вид функционального блока
1	2
1	Создать блок, выполняющий функцию неполного цифрового мультиплексора с двумя адресными (A0, A1) и тремя информационными (D0, D1, D2) входами и с дополнительным управляющим входом CE, разрешающим прохождение сигнала на выход Y. При запрещении работы на выходе мультиплексора Y = 0.
2	Создать блок, выполняющий функцию полного цифрового демультиплексора с двумя адресными входами (A0, A1), одним информационным входом (D) и четырьмя информационными выходами (D0, D1, D2, D3).
3	Создать блок, выполняющий функцию полного цифрового демультиплексора с двумя адресными входами (A0, A1), одним информационным входом (D) и четырьмя информационными выходами (D0, D1, D2, D3). Также создать дополнительный управляющий вход CE, разрешающий прохождение сигнала со входа на выход. При запрещении работы на выходах мультиплексора установить 0.
4	Создать блок, выполняющий функцию полного цифрового демультиплексора с двумя адресными входами (A0, A1), одним информационным входом (D) и четырьмя информационными выходами (D0, D1, D2, D3). Также создать дополнительный управляющий вход CE, разрешающий прохождение сигнала со входа на выход. При запрещении работы на выходах мультиплексора установить 1.
5	Создать блок, выполняющий функцию неполного цифрового демультиплексора с двумя адресными входами (A0, A1), одним информационным входом (D) и тремя информационными выходами (D0, D1, D2).
6	Создать блок, выполняющий функцию бинарного двоичного одноединичного дешифратора.
7	Создать блок, выполняющий функцию бинарного двоичного одноединичного дешифратора. Также создать дополнительный управляющий вход CE, разрешающий работу дешифратора. При запрещении работы на всех выходах дешифратора установить 1.
8	Создать блок, выполняющий функцию бинарного двоичного одноединичного дешифратора. Также создать дополнительный управляющий вход CE, разрешающий работу дешифратора. При запрещении работы на всех выходах дешифратора установить 0.
9	Создать блок, выполняющий функцию трёхвходового двоичного одноединичного дешифратора.
10	Создать блок, выполняющий функцию трёхвходового двоичного одноединичного дешифратора. Также создать дополнительный управляющий вход CE, разрешающий работу дешифратора. При запрещении работы на всех выходах дешифратора установить 0.

Продолжение табл. 81

1	2
11	Создать блок, выполняющий функцию трёхвходового двоичного одноединичного дешифратора. Также создать дополнительный управляющий вход CE, разрешающий работу дешифратора. При запрещении работы на всех выходах дешифратора установить 1.
12	Создать блок, выполняющий функцию четырёхвходового шифратора.
13	Создать блок, выполняющий функцию четырёхвходового шифратора. Также создать дополнительный управляющий вход CE, разрешающий работу шифратора. При запрещении работы на всех выходах шифратора установить 0.
14	Создать блок, выполняющий функцию четырёхвходового шифратора. Также создать дополнительный управляющий вход CE, разрешающий работу шифратора. При запрещении работы на всех выходах шифратора установить 1.
15	Создать блок, выполняющий функцию восьмивходового шифратора.
16	Создать блок, устраняющий эффект дребезга контактов по одному входу. Считать время завершения переходного процесса равным 200 мс.
17	Создать блок, регулирующий длительностью нажатия кнопки T коэффициент заполнения D ШИМ – сигнала на выходе. Создать зависимость вида D [%] = $10 \cdot T$ [с].
18	Создать блок, регулирующий длительностью нажатия кнопки T коэффициент заполнения D ШИМ – сигнала на выходе. Создать зависимость вида D [%] = $2 \cdot T^2$ [с].
19	Создать блок, регулирующий длительностью нажатия кнопки T коэффициент заполнения D ШИМ – сигнала на выходе. Создать зависимость вида D [%] = $1 – 10 \cdot T$ [с].
20	Создать блок, регулирующий длительностью нажатия кнопки T коэффициент заполнения D ШИМ – сигнала на выходе. Создать зависимость вида D [%] = $1 – T^2$ [с].
21	Создать блок, задающий коэффициент заполнения D ШИМ – сигнала на выходе цифровым кодом на входе. Используйте двухбитное регулирование.
22	Создать блок, задающий коэффициент заполнения D ШИМ – сигнала на выходе цифровым кодом на входе. Используйте трёхбитное регулирование.
23	Создать блок, выполняющий функцию задающего генератора. На входе блока требуется задавать время полупериода выходного колебания, а на выходе – формировать меандр соответствующей частоты.
24	Создать блок, выполняющий функцию задающего генератора. На входе блока требуется задавать время полупериода выходного колебания, а на выходах – формировать меандр соответствующей частоты f и частоты f /2.
25	Создать блок, выполняющий функцию полного цифрового мультиплексора с двумя адресными (A0, A1) и четырьмя информационными (D0, D1, D2, D3) входами и одним информационным выходом Y.

Окончание табл. 81

1	2
26	Создать блок, выполняющий функцию полного цифрового мультиплексора с двумя адресными (A0, A1) и четырьмя информационными (D0, D1, D2, D3) входами с дополнительным управляющим входом CE, разрешающим прохождение сигнала на выход Y. При запрещении работы на выходе мультиплексора Y = 1.
27	Создать блок, выполняющий функцию полного цифрового мультиплексора с двумя адресными (A0, A1) и четырьмя информационными (D0, D1, D2, D3) входами с дополнительным управляющим входом CE, разрешающим прохождение сигнала на выход Y. При запрещении работы на выходе мультиплексора Y = 0.
28	Создать блок, выполняющий функцию неполного цифрового мультиплексора с двумя адресными (A0, A1) и тремя информационными (D0, D1, D2) входами и одним информационным выходом Y.
29	Создать блок, выполняющий функцию неполного цифрового мультиплексора с двумя адресными (A0, A1) и тремя информационными (D0, D1, D2) входами и с дополнительным управляющим входом CE, разрешающим прохождение сигнала на выход Y. При запрещении работы на выходе мультиплексора Y = 1.
30	Создать блок, выполняющий функцию задающего генератора. На входе блока требуется задавать время полупериода выходного колебания, а на выходах – формировать меандр соответствующей частоты *f*, частоты *f* /2 и частоты *f* /4.

Пример выполнения лабораторной работы 10.2.

Разберем создание функционального блока на примере варианта № 30. Согласно заданию, требуется получить три выходные переменные, формирующие меандр. Меандр для первой выходной переменной – заданной частоты в зависимости от входного параметра, для второй выходной переменной – меандр с половиной от заданной частоты, и для третьей выходной переменной – меандр с четвертью от заданной частоты.

Добавим функциональный блок в дерево проекта, как описано выше. В появившемся окне «Insert» запишем имя функционального блока (User_Block_Samples**)**, выберем тип «Function Block» и язык программирования «Function Block Diagram». В рабочем листе для кода «User_Block_Samples» создадим необходимые переменные и функциональные преобразования для них, а в листе комментариев «User_Block_Samples*T*» пометим, какие параметры функциональный блок принимает, а какие – возвращает (см. рис. 102). Скомпилируем проект, найдем новый функциональный блок «User_Block_Samples» в окне «Edit Wizard» в группе <User Block> и поместим его в лист для кода «Main». Исследуем его, убедимся в адекватности выходных данных на логическом анализаторе (рисунок 103). Выбрав по щелчку правой кнопки мыши созданный функциональный блок в окне «Edit Wizard» из выпадающего меню

строку «Help on FB/FU», можно вызвать окно с ранее записанными комментариями «Help o*f* POU User_Block_Samples».

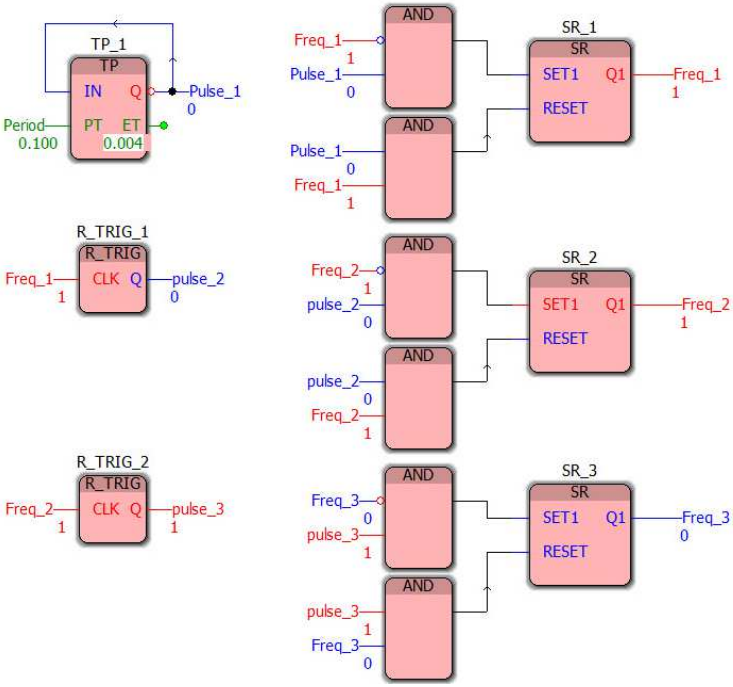


Рис. 102. Лист «User_Block_Samples»

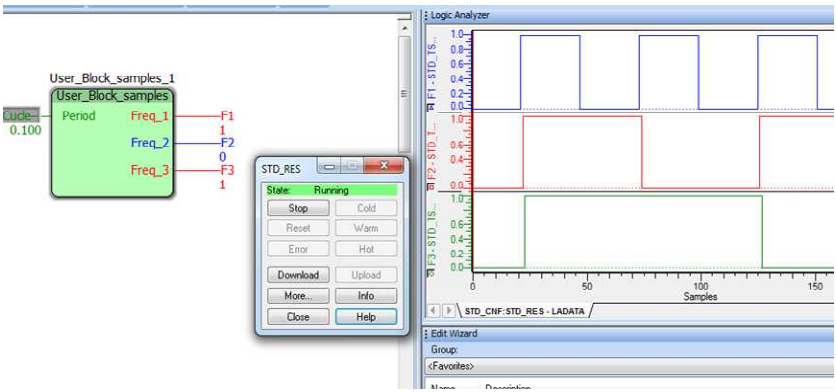


Рис. 103. Тестирование созданного функционального блока

Тема 11. РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА УПРАВЛЕНИЯ ПРОГРАММОЙ ПРОГРАММИРУЕМОГО ЛОГИЧЕСКОГО КОНТРОЛЛЕРА В РЕДАКТОРЕ WEBVISIT

Редактор WebVisit входит в комплект программ AutomationWORX Software Suit компании Phoenix Contact, и предназначен для разработки графического веб-интерфейса управления (Human machine interface – HMI – интерфейс человек-машина) программой ПЛК. Редактор WebVisit основан на открытых стандартах Java и предполагает управление проектами через веб-браузер (с поддержкой Java Virtual Machine) по протоколу TCP/IP.

В редакторе создается только графический дизайн HMI, транслируемый затем на ПЛК, при этом связь между браузером и веб-сервером устанавливается автоматически. Навыков программирования при создании дизайна HMI в редакторе WebVisit не требуется. Рабочий процесс создания HMI состоит из двух этапов:

1) создание дизайна HMI с помощью основных графических символов и объектов (рисунок 104) и создание связей для этих элементов с выбранными переменными проекта;

2) транслирование созданного дизайна HMI на веб-сервер через встроенный FTP клиент.

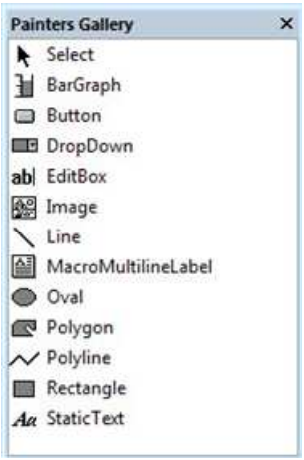


Рис. 104. Основные графические символы и объекты редактора WebVisit

До создания дизайна графического веб-интерфейса в основном проекте PC WorX требуется обозначить переменные, которые будут связаны с графическими объектами редактора WebVisit. Для этого в свойствах переменной требуется отметить галочкой пункт PDD, таким образом помещая переменную в Process Data Directory и открывая к ней доступ при обращении к ПЛК (рис. 105).

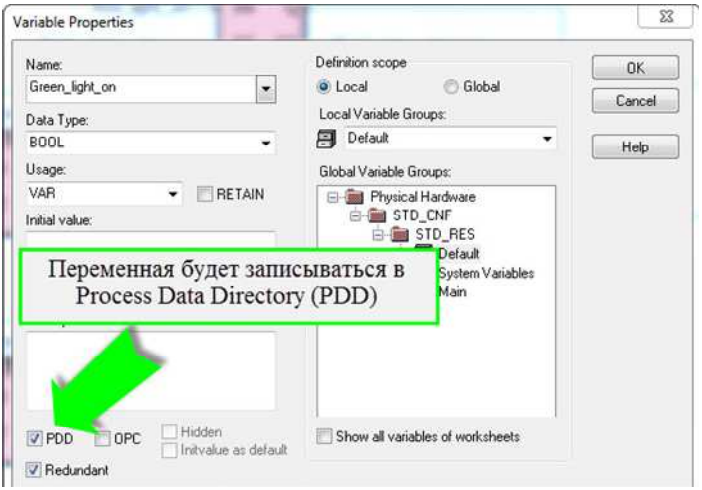


Рис. 105. Свойства переменной в PC WorX

Далее запускаем редактор WebVisit, и создаем новый проект, следуя указаниям мастера создания дизайна. Мастер создания дизайна предлагает последовательно к заполнению 7 форм, позволяя произвести начальные настройки создаваемого проекта HMI.

1. Предлагается выбрать профиль с проекта с различными предустановками (по умолчанию – веб-интерфейс на ПЛК), ввести имя создаваемого проекта и выбрать директорию размещения файлов проекта.
2. Задаётся разрешение для создаваемой формы (по умолчанию 640×480) и период опроса формы (по умолчанию 1000 ms).
3. Выбирается стиль элементов из 4 предустановленных вариантов.
4. Выбирается число окон для создаваемого проекта HMI.
5. Предлагается установить послойное отображение окон дизайна HMI, вводя, помимо рабочего окна, дополнительно окно заднего фона (background) и переднего фона (foreground).
6. Шестой экран мастера предлагает внедрить в проект дополнительные языки. В качестве предустановленных предлагаются английский, французский, немецкий и испанский.

7. Добавляется механизм авторизации в виде пароля на открытие дизайна HMI.

После завершения работы мастера на листах проекта размещаются графические элементы (см. рис. 104), которые можно превратить в интерактивные. Для этого в свойствах этих элементов выбираются действия и переменные PPO, и задается проверка условий для выбранных действий.

Для передачи в редактор WebVisit переменных, выбранных в проекте PC WorX, в редакторе в меню «Project» открываем окно «Project configuration» и выбираем вкладку Project Advanced (рис. 106).

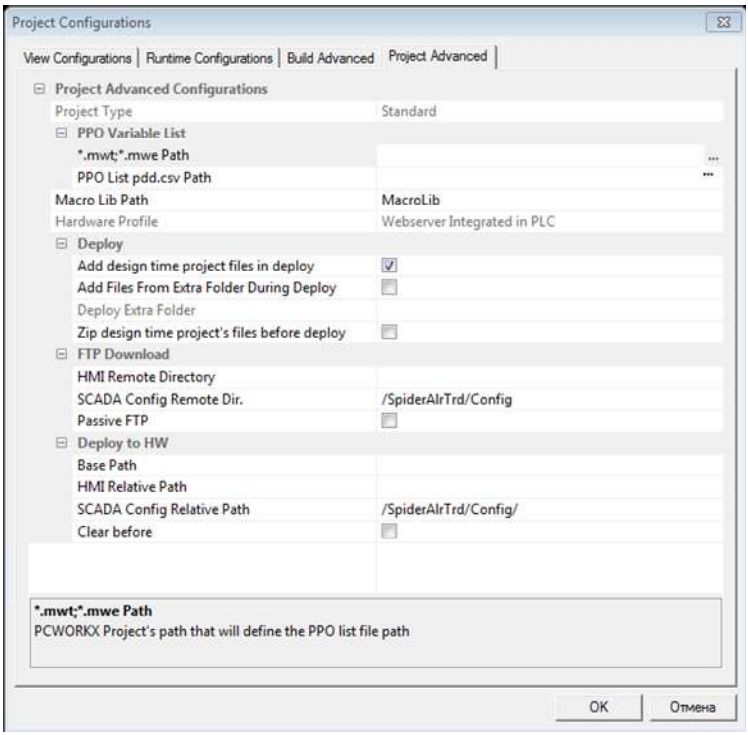


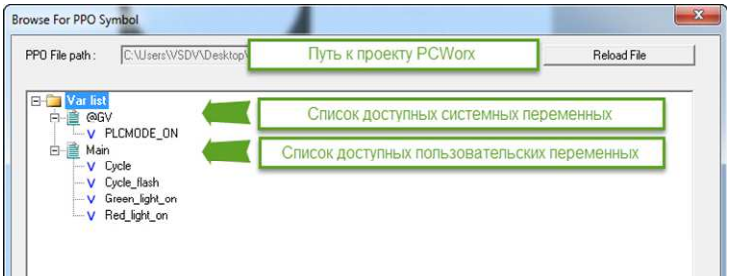
Рис. 106. Окно «Project configuration» в WebVisit

В появившемся окне в строке «*.mwt, *.mwe Path» необходимо выбрать путь к проекту PC WorX. В этом случае все переменные проекта PC WorX, отмеченные галочкой PDD, будут транслированы в редактор WebVisit.

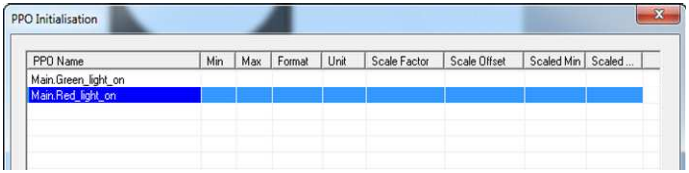
Список доступных переменных будет отображен при обращении к нему из меню свойств графических объектов. Пример такого списка пере-

менных изображен на рис. 107-*а*. Для изменения диапазона отображаемых данных необходимые элементы доступны в окне «PPO Initialisation» (Project → Init PPO) (рис. 107-*б*).

При создании проекта HMI в процессе работы доступна симуляция создаваемого дизайна кнопкой F5 либо в меню Project → Simulate HMI. После создания дизайна проекта HMI его необходимо скомпилировать (Project → Build All или быстрая клавиша F7) и загрузить на FTP-сервер, роль которого по умолчанию выполняет ПЛК. В меню «Project» выбрать «Download Project» и в появившемся окне прописать IP-адрес ПЛК (по умолчанию IP 192.168.0.2), нажать «Connect» и убедиться в корректности загрузки данных (рисунок 108). После загрузки проект будет доступен через веб-браузер по адресу ftp://192.168.0.2.



а) Фрагмент окна списка переменных PPO



б) Фрагмент окна инициализации переменных PPO

Рис. 107. Переменные PPO

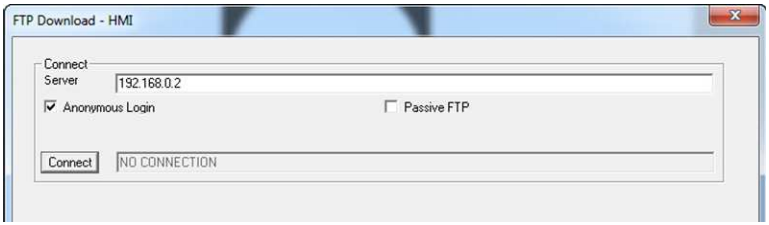


Рис. 108. Фрагмент окна FTP Download

11.1. Лабораторная работа «Веб-интерфейс для отображения показаний аналогового потенциометра ILC131 Starterkit»

Цель лабораторной работы: научиться в редакторе WebVisit среды программирования PC WorX v.6.30 разрабатывать графический пользовательский интерфейс для отображения показаний аналогового потенциометра.

Порядок выполнения лабораторной работы

Отобразить показания аналогового потенциометра в виде линейной шкалы в диапазоне от 0 до 10 (согласно отградуированной на потенциометре шкале), размер которой соответствует текущему значению потенциометра. В указанном диапазоне задавать аварийный уровень, при превышении которого линейная шкала изменит цвет с зеленого на красный.

Пример выполнения лабораторной работы 11.1

Разработаем программу отображения показаний потенциометра в PC WorX, взяв за основу программу, разобранную в лабораторной работе 5.1. темы «Операции сравнения». В программе (рис. 109) переменной *Analog_OUT* соответствует сигнал положения потенциометра в шестнадцатеричном коде. Для преобразования шестнадцатеричного сигнала в десятичный и получения требуемого диапазона использованы блоки WORD_TO_INT и DIV. Блоком GT задаем уровень, при превышении которого появляется сигнал *Alarm*. Для отображения работы программы через веб-интерфейс отметим галочку PDD в свойствах переменных *OUT* и *Level*.

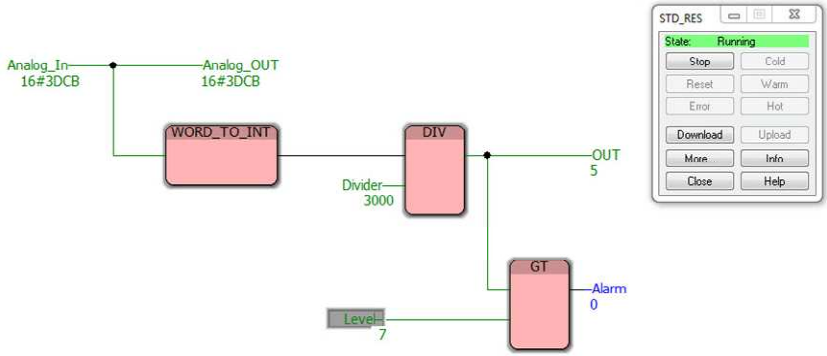


Рис. 109. Пример выполнения лабораторной работы 11.1

Создадим проект дизайна HMI в WebVisit, воспользовавшись работой мастера создания, выбрав все значения по умолчанию. На рабочую форму

поместим элемент *Bargraph*, который будет отображать линейную шкалу (маркер 1 рис. 110), и пять элементов *StaticText* (маркеры 2–6 рис.а 110).

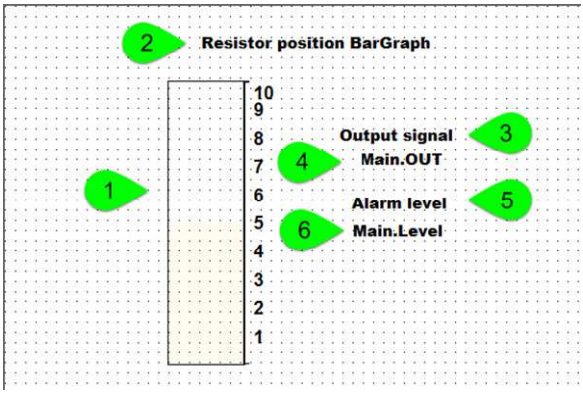


Рис. 110. Рабочее поле создаваемого дизайна

Элементы *StaticText*, обозначенные маркерами 2, 3 и 5, предназначены только для создания поясняющих надписей и в процессе работы меняться не будут. Ввести поясняющий текст для каждой надписи можно в свойствах выбранного элемента (маркер 1 рис. 111) в выпадающем меню General → Repaints → Edit 1 → Name (рис. 111, маркер 2).

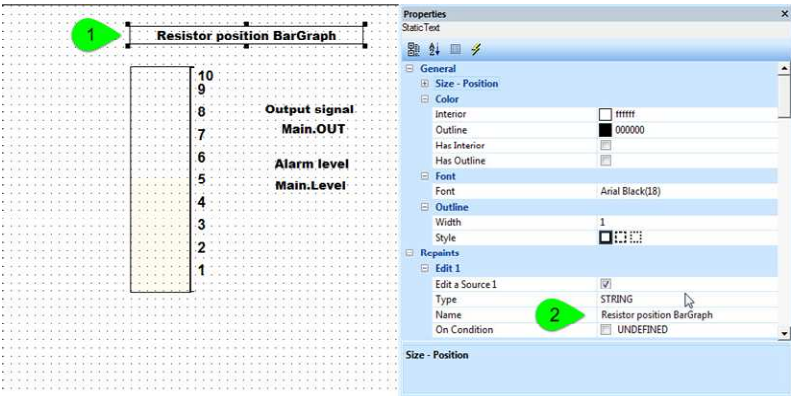


Рис. 111. Меню «Properties» второго графического элемента StaticText

Под надписью «Output signal» требуется отображать значение выходного сигнала. Для этого поместим в требуемом месте элемент StaticText, в свойствах которого в меню General → Repaints → Edit 1 → Type выберем

из выпадающего меню тип PPO (рис. 112, маркер 1), а в General → Repaints → Edit 1 → Name из выпадающего меню (рис. 112, маркер 2) выберем в появившемся окне (рис. 107-а) переменную OUT, содержащую информацию об уровне выходного сигнала. Аналогичным образом сопоставим шестому элементу StaticText значение переменной *Level*.

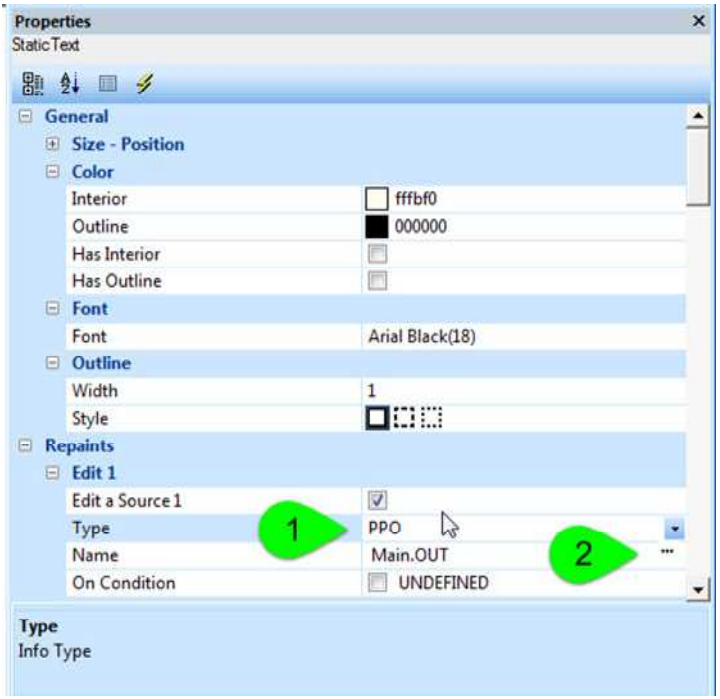


Рис. 112. Меню «Properties» пятого графического элемента StaticText

Для элемента BarGraph в его свойствах в меню General → Border Advanced → Border Colors выберем два цвета (рис. 113, маркер 1), ниже в меню *Colors* пометим галочками и выберем цвет для элементов меню *Use Interior Color 1* и *2* (маркеры 2 и 3 рис. 113). Теперь зададим условие изменения цвета с первого на второй. В меню General → Border Advanced → Border Colors → Conditional Interior Color поставим галочку на опции *Has Interior On Condition* (заливка по условию) и выбираем условие «больше либо равно» (рис. 113 маркер 4). Далее для *Condition Info 1* выбираем значение переменной OUT, а для *Condition Info 2* – значение переменной *Level* (маркеры 5 и 6 рис. 113). Таким образом, получаем условие смены цвета *Main.OUT >= Main.Level.*

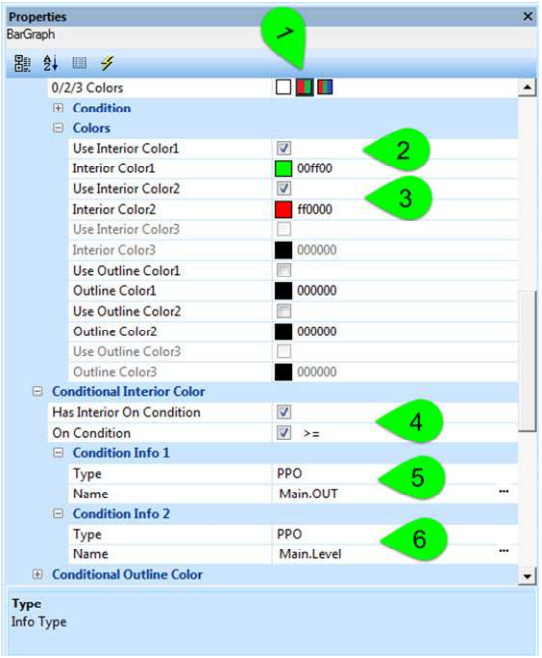
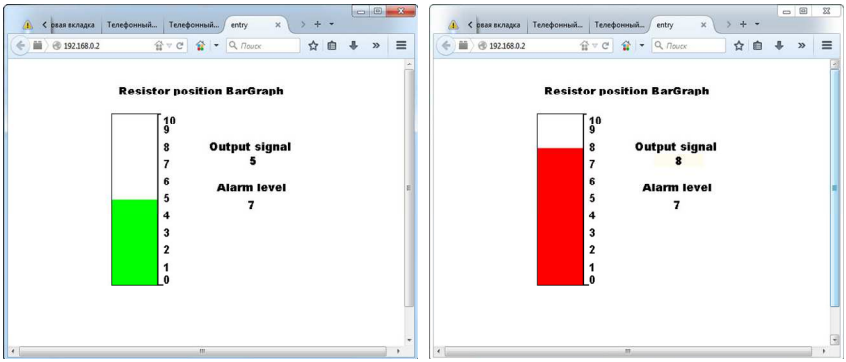


Рис. 113. Меню «Properties» первого графического элемента BarGraph

Компилируем проект и загружаем его на FTP-сервер. В окне браузера набираем IP-адрес FTP-сервера и разрешаем выполнение Java-апплета. После загрузки апплета открывается разработанный web-интерфейс (рис. 114).



а) б)

Рис. 114. Web-интерфейс к примеру № 1: а) OUT < Level; б) OUT > Level

11.2. Лабораторная работа «Веб-интерфейс для отображения работы пешеходного светофора»

Цель лабораторной работы: научиться в редакторе WebVisit среды программирования PC WorX v.6.30 разрабатывать графический пользовательский интерфейс для отображения работы пешеходного светофора.

Порядок выполнения лабораторной работы

Отобразить работу пешеходного светофора. Исходную программу взять из примера решения задания № 2 в главе «Таймеры».

Пример выполнения лабораторной работы 11.2

Перекомпилируем исходную программу PC WorX, предварительно отметив галочкой PDD в свойствах переменных, отвечающих за отображение красного и зеленого цветов.

Поместим на рабочее поле три графических элемента: два типа *Oval* и один *Rectangle*, – как показано на рис. 115. Прямоугольник закрасим однотонным серым цветом, и наложим на него два круга, получив стилизованное изображение светофора. Верхний круг изображает красный сигнал, поэтому в его свойствах выберем отображение двух цветов – базовый серый, изменяющийся по условию на красный. Условие отображения зададим как Red_light_on > 0. Аналогичные процедуры проделаем и для второго круга, но здесь вторым цветом выберем зеленый (рис. 116 – свойства элементов Oval).

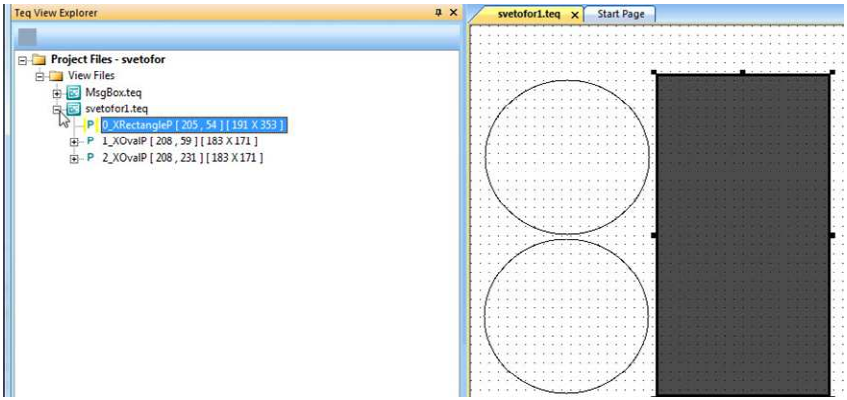
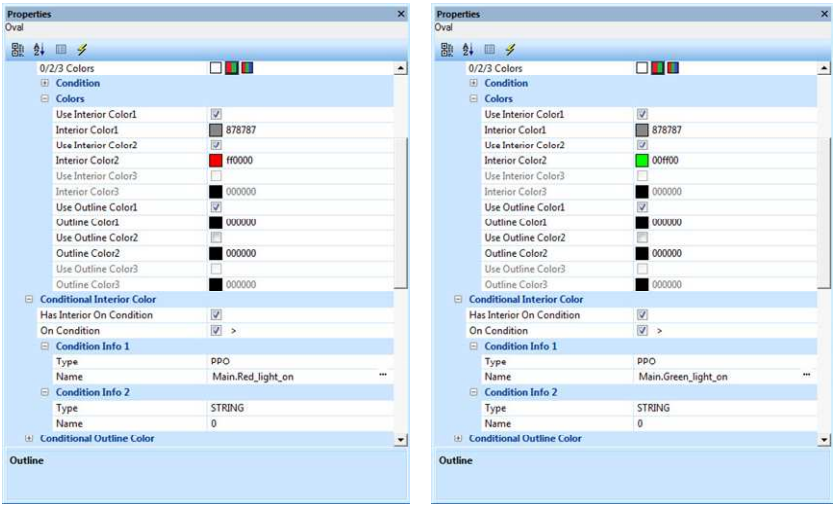


Рис. 115. Графические элементы для лабораторной работы 11.2



а) б)

Рис. 116. Свойства элементов Oval: а) для отображения красного цвета; б) для отображения зеленого цвета

После компиляции и загрузки проекта на сервер проверяем работу полученного интерфейса, сопоставляя с работой программы PC WorX (рис. 117).

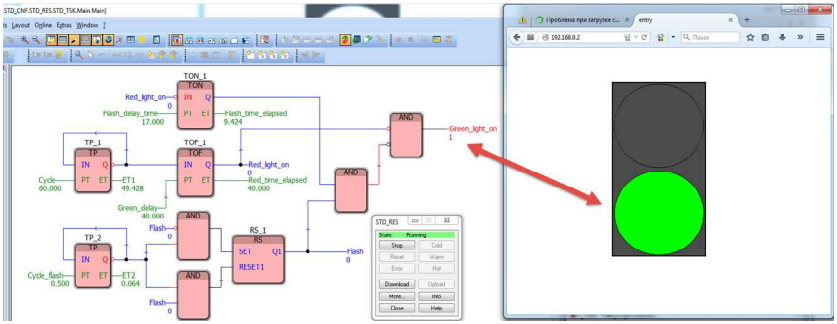


Рис. 117. Проект программы и web-интерфейс примера выполнения лабораторной работы 11.2

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Упражнения повышенной сложности

Управление воздушной заслонкой

На промышленном объекте установлена система вентилирования и кондиционирования. Одним из элементов системы является воздушная заслонка, управляемая напряжением.

Задание. Создать веб-интерфейс работы воздушной заслонки. Создать программу в PC WorX, реализующую нижеприведенный алгоритм работы.

Алгоритм. Программа ПЛК должна обеспечить следующие режимы работы.

1. Реализовать управление воздушной заслонкой с цифрового выхода ПЛК, считая, что заслонка управляется с ПЛК через дифференцирующее звено. Положение заслонки «открыто» – 0В, положение «закрыто» – 10 В.
2. Обеспечить плавное регулирование степени открытия заслонки кнопками веб-интерфейса.
3. Отображать текущее положение заслонки в процентах.
4. Сигнализировать о достижении концевых положений «открыто» и «закрыто».

Автоматический ввод резерва

На промышленном объекте 3 группы насосов, по 2 насоса в каждой группе – один рабочий, второй резервный. Возле каждой группы насосов расположен кнопочный пост с кнопкой «пуск». На объекте имеются четыре технологических датчика (Sensor1, Sensor2, Sensor3, Sensor4).

Задание. Создать веб-интерфейс работы системы автоматического ввода резерва. Создать программу в PC WorX, реализующую нижеприведенный алгоритм работы.

Алгоритм. Программа ПЛК должна обеспечить следующие режимы работы.

1. При нажатии на кнопку «пуск» насос соответствующей группы должен включиться на 15 секунд, затем автоматически отключится.

2. При срабатывании двух датчиков из четырех необходимо переключать в соответствующей группе рабочий насос на резервный до выключения датчиков:

Таблица 82

Sensor1	Sensor2	Sensor3	Sensor4	Группа
on	on	off	off	Группа 1
off	on	on	off	Группа 2
off	off	on	on	Группа 3

3. Необходимо подсчитывать число включений и текущую наработку (в часах и минутах) для каждого насоса.

Управление освещением и подсчетом машиномест гаража

Гараж имеет двое ворот для въезда и выезда автомобилей. В каждых воротах находятся два датчика фиксации проезда автомобиля – один снаружи ворот и одного внутри ворот. Предполагается, что при въезде автомобиля в гараж сначала срабатывает внешний датчик, а затем внутренний, а при выезде наоборот. Режим движения автомобилей таков, что датчики срабатывают с интервалом 0,5 секунд и не должны срабатывать одновременно. Число машино-мест в гараже – 15.

Задание. Создать веб-интерфейс работы системы управления гаражем со схематичным изображением гаража и машиномест. Создать программу в PC WorX, реализующую нижеприведенный алгоритм работы.

Алгоритм. Программа ПЛК должна обеспечить следующие режимы работы.

1. Подсчитывать текущее число машин в гараже. В веб-интерфейсе затенять машиноместа, занятые автомобилями.
2. Выводить данные о текущем количестве свободных мест. При заполнении гаража выводить надпись «Мест нет».
3. Выключать свет, если гараж полностью пустой или если после постановки последней машины прошло 30 секунд.

Система пожарной сигнализации здания

Здание имеет три этажа. На каждом этаже здания установлено 4 пожарных датчика (Sens), кнопка ручного тестирования Alarm_test, кнопка ручного выключения сигнализации Alarm_off, и сигнальная лампа Alarm_lamp.

1 этаж – Sensor11, Sensor12, Sensor13 и Sensor14, Alarm_test1, Alarm_off1, Alarm_lamp1

2 этаж – Sensor21, Sensor22, Sensor23 и Sensor24, Alarm_test2, Alarm_off2, Alarm_lamp2

3 этаж – Sensor31, Sensor32, Sensor33 и Sensor34, Alarm_test3, Alarm_off3, Alarm_lamp3

Сигнализация пожара ALARM общая для всего здания.

Задание. Создать веб-интерфейс работы пожарной сигнализации здания со схематичным изображением этажей здания и указанными выше интерактивными компонентами. Создать программу в PC WorX, реализующую нижеприведенный алгоритм работы.

Алгоритм. Программа ПЛК должна обеспечить следующие режимы работы.

1. Если на этаже срабатывает хотя бы один датчик, загорается сигнальная лампа для соответствующего этажа. Лампа гаснет при отключении датчика.
2. При срабатывании двух и более датчиков на этаже – включается пожарная сигнализация.
3. При срабатывании двух и более сигнальных ламп в здании – включается пожарная сигнализация.
4. Сигнализация отключается кнопкой Alarm_off соответствующего этажа, либо кнопкой Alarm_off1 – первого этажа.
5. Пожарная сигнализация может быть включена кнопкой проверки Alarm_test любого этажа вне зависимости от состояния датчиков. В этом случае загорается и мигает этажная сигнальная лампа.

Система управления аварийным дизель-генератором

Цель лабораторной работы: использовать ранее полученные знания и навыки для разработки в среде программирования PC WorX v.6.30 приложения на языке программирования Function Block Diagram для системы управления аварийным дизель-генератором.

На предприятии существует система аварийного электропитания от дизель-генератора (АДГ). Принципиальная схема подключения питания показана на рис. 118.

Входными сигналами для системы управления служат контакты реле Р5 «РКС» и Р6 «РКГ» (наличие напряжения сети и напряжения с АДГ соответственно). Для коммутации нагрузки на сеть или АДГ служат контактор сети Р1 «КС» и контактор генератора Р2 «КГ», электрически и механически заблокированные от одновременного включения. Управление этими контакторами производится с помощью контактов Р1.1 «КСА» и Р2.1 «КГА» реле Р1 и Р2, находящихся в схеме управления.

При этом предполагается ввод в работу АДГ вручную.

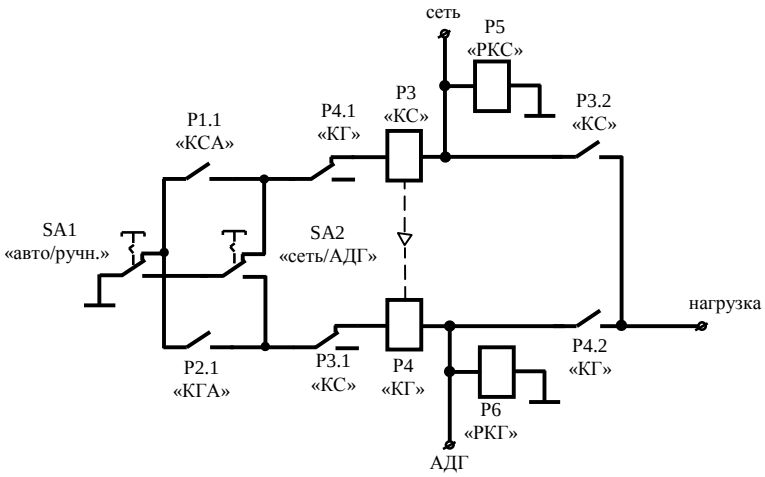


Рис. 118. Принципиальная схема подключения нагрузки к сети или к АДГ

Задание. Автоматизировать запуск АДГ и обеспечить обработку штатных и аварийных режимов работы согласно предложенному алгоритму. Создать веб-интерфейс, на котором показывать на принципиальной схеме текущее состояние контактных групп и режимы работы АДГ.

Алгоритм. Программа ПЛК должна обеспечить следующие режимы работы системы управления.

1. Дежурный режим.

Напряжения с АДГ нет, есть напряжение в сети.

Реле КГА в положении «выкл», реле КСА в положении «вкл» – нагрузка подключена к сети. Мигает светодиод VD1 «режим»

2. Кратковременное пропадание сети (менее 10 секунд).

Нагрузка подключена к сети (реле КГА в положении «выкл», реле КСА в положении «вкл»). Светодиод VD1 «режим» не мигает.

Если сетевое напряжение пропадает на период времени менее, чем 10 секунд, а затем восстанавливается, то система продолжает работать в дежурном режиме.

3. Долговременное пропадание сети (более 10 секунд).

Нагрузка подключена к сети (реле КГА в положении «выкл», реле КСА в положении «вкл»). Светодиод VD1 «режим» не мигает.

При пропадании сетевого напряжения на период времени более, чем 10 секунд, начинается работа в режиме «Запуск дизеля».

4. Режим «Запуск дизеля».

Реле КГА в положении «выкл», реле КСА в положении «выкл» – нагрузка не подключена ни к АДГ, ни к сети.

Последовательно включаются следующие реле:

1) на 2 сек. включается реле Р2 «работа»;
2) на 7 сек. включается реле Р4 «подогрев»;
3) на 5 сек. включается реле Р1 «старт».

После выключения реле Р1 «старт» в течение 15 секунд происходит проверка наличия напряжения с АДГ. Если по каким-то причинам напряжение с АДГ не появилось, то запуск повторяется. Если не появилось напряжение с АДГ после третьего запуска, то выход в режим «сбой».

5. Режим «сбой».

Мигает светодиод VD7 «сбой», реле КГА в положении «выкл», реле КСА в положении «вкл». Система находится в этом режиме до выключения напряжения питания системы управления.

6. Режим «прогрев АДГ».

Если после режима «запуск» появилось напряжение с АДГ, то система переходит в режим «прогрев АДГ». В этом режиме в течение 2 минут включено реле Р2 «работа».

7. Режим «Работа нагрузки от АДГ».

После режима «прогрев» система переходит к режиму «Работа нагрузки от АДГ». В этом режиме включено реле Р2 «работа», реле КГА в положении «вкл», реле КСА в положении «выкл». Мигает светодиод VD1 «режим». В таком состоянии система остается до появления сетевого напряжения либо до пропадания напряжения с АДГ.

Если при работе в этом режиме пропало напряжение с АДГ, то система переходит в режим «сбой».

Если во время работы в режиме «Работа нагрузки от АДГ» появляется напряжение в сети и держится не менее 10 секунд, то система переходит к режиму «останов АДГ».

8. Режим «останов АДГ».

В этом режиме включено реле Р2 «работа», реле КГА в положении «выкл», реле КСА в положении «вкл». В таком состоянии система находится 2 минуты.

Если в течение этих двух минут напряжение в сети не пропадает, то реле Р2 «работа» выключается, и система выходит в дежурный режим.

Если при работе в этом режиме напряжение в сети пропадает более, чем на 10 секунд, система переходит в режим «Работа нагрузки от АДГ».

Тесты для текущего и промежуточного контроля успеваемости

Реализация логических функций для ПЛК

1. Функции, изображенной на рис. 119, соответствует таблица входов-выходов

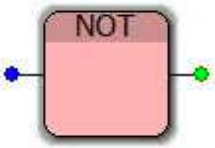


Рис. 119. Функция NOT

а)

Вход	Выход
FALSE	FALSE
TRUE	FALSE

б)

Вход	Выход
FALSE	TRUE
TRUE	TRUE

в)

Вход	Выход
FALSE	FALSE
TRUE	TRUE

г)

Вход	Выход
FALSE	TRUE
TRUE	FALSE

2. Функции, изображенной на рис. 120, соответствует таблица входов-выходов

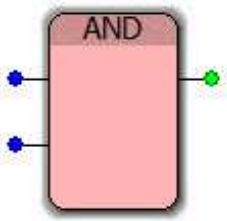


Рис. 120. Функция AND

а)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

б)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

в)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

г)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

3. Функции, изображенной на рис. 121, соответствует таблица входов-выходов

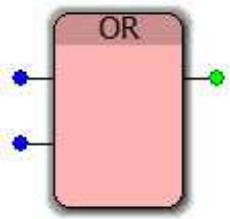


Рис. 121. Функция OR

а)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

б)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

в)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

г)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

4. Функции, изображенной на рис. 122, соответствует таблица входов-выходов

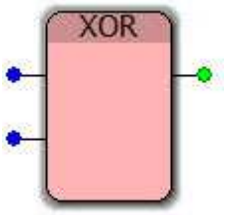


Рис. 122. Функция XOR

а)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

б)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

в)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

г)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

5. Функция, которая выдаёт выход TRUE, если хотя бы один из входов истинен, называется

а) логическое «И» (конъюнкция)

б) логическое «ИЛИ» (дизъюнкция)

в) импликация

г) логическое «И-НЕ» (штрих Шеффера)

6. Функция, которая выдаёт выход TRUE, если истинен только один из входов, называется

а) логическое «И» (конъюнкция)

б) логическое «И-НЕ» (штрих Шеффера)

в) «исключающее ИЛИ»
г) импликация

7. Функция, которая выдаёт выход TRUE, если оба входа истинны, называется
а) логическое «И» (конъюнкция)
б) логическое «ИЛИ» (дизъюнкция)
в) «исключающее ИЛИ»
г) логическое «НЕ» (отрицание)

8. Функция, которая выдаёт выход FALSE, если оба входа совпадают, называется
а) логическое «И» (конъюнкция)
б) логическое «ИЛИ» (дизъюнкция)
в) «исключающее ИЛИ»
г) логическое «НЕ» (отрицание)

9. Функция, которая выдаёт выход FALSE тогда и только тогда, когда если оба входа FALSE, называется
а) логическое «И» (конъюнкция)
б) логическое «ИЛИ» (дизъюнкция)
в) «исключающее ИЛИ»
г) логическое «НЕ» (отрицание)

10. Функция, которая выдаёт выход FALSE во всех случаях, кроме случая, когда оба входа TRUE, называется
а) логическое «И» (конъюнкция)
б) логическое «ИЛИ» (дизъюнкция)
в) «исключающее ИЛИ»
г) логическое «НЕ» (отрицание)

11. Функция, которой соответствует вектор-строка выходов (1100), в PC WorX определяется программным блоком
а)

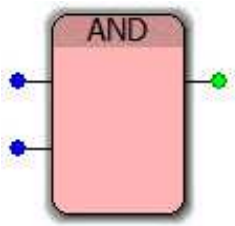


Рис. 123. Функция AND

б)

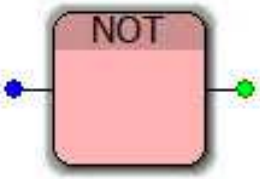


Рис. 124. Функция NOT

в)

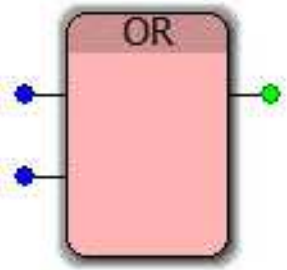


Рис. 125. Функция OR

г)

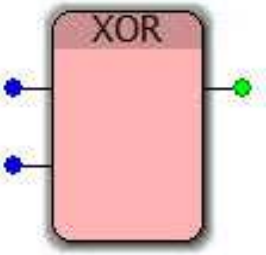


Рис. 126. Функция XOR

12. Функция, которой соответствует вектор строка выходов (0001), в PC WorX определяется программным блоком

а)

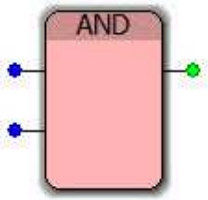


Рис. 127. Функция AND

б)

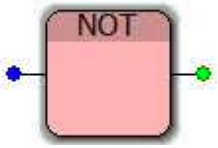


Рис. 128. Функция NOT

в)

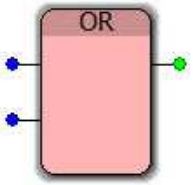


Рис. 129. Функция OR

г)

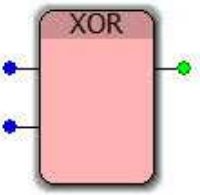


Рис. 130. Функция XOR

13. Функция, которой соответствует вектор строка выходов (0111), в PC WorX определяется программным блоком

а)

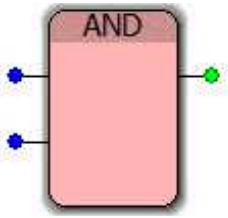


Рис. 131. Функция AND

б)

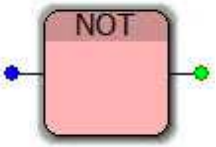


Рис. 132. Функция NOT

в)

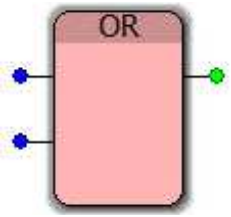


Рис. 133. Функция OR

г)

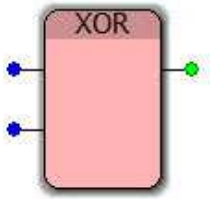


Рис. 134. Функция XOR

14. Функция, которой соответствует вектор строка выходов (0110), в PC WorX определяется программным блоком

а)

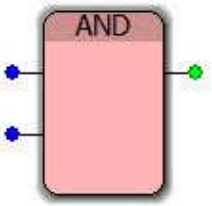


Рис. 135. Функция AND

б)

Рис. 136. Функция NOT

в)

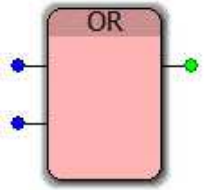


Рис. 137. Функция OR

г)

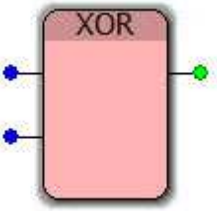


Рис. 138. Функция XOR

15. Для реализации следующей функции булевой алгебры $(\overline{x} \vee \overline{y}) \wedge (\overline{x \vee z}) \vee x \wedge y$ не используется программный блок (функция)

а)

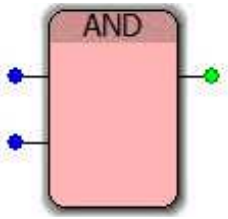


Рис. 139. Функция AND

б)

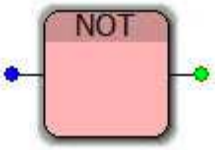


Рис. 140. Функция NOT

в)

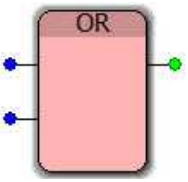


Рис. 141. Функция OR

г)

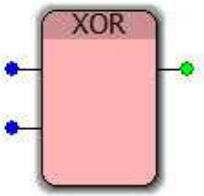


Рис. 142. Функция XOR

16. Для реализации следующей функции булевой алгебры $\overline{\overline{x \oplus y} \wedge \bar{z}} \wedge x \oplus y \oplus z$ не используется программный блок (функция)

а)

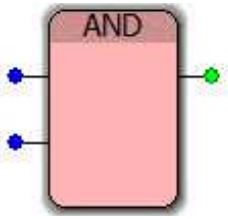


Рис. 143. Функция AND

б)

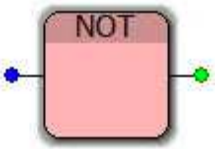


Рис. 144. Функция NOT

в)

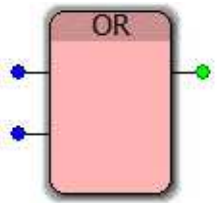


Рис. 145. Функция OR

г)

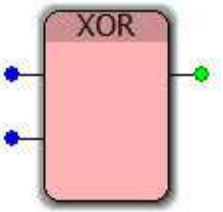


Рис. 146. Функция XOR

17. Для реализации следующей функции булевой алгебры $\overline{x \vee y} \oplus \overline{y \vee z} \oplus \overline{x \vee \overline{z}}$ не используется программный блок (функция)

а)

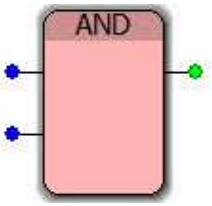


Рис. 147. Функция AND

б)

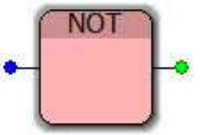


Рис. 148. Функция NOT

в)

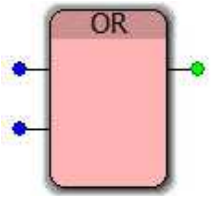


Рис. 149. Функция OR

г)

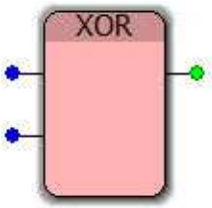


Рис. 150. Функция XOR

18. Для реализации следующей функции булевой алгебры $(x \wedge y \wedge z) \vee (x \oplus y \oplus z)$ не используется программный блок (функция)

а)

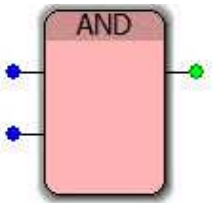


Рис. 151. Функция AND

б)

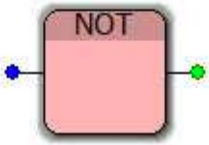


Рис. 152. Функция NOT

в)

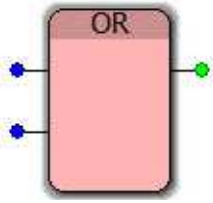


Рис. 153. Функция OR

г)

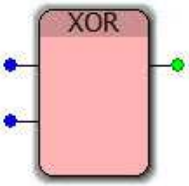


Рис. 154. Функция XOR

19. Тавтологии (тождественно истинной функции) соответствует таблица входов-выходов

а)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

б)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

в)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

г)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

20. Противоречию (тождественно ложной функции) соответствует таблица входов-выходов

а)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

б)

Вход 1	Вход 2	Выход
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

в)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

г)

Вход 1	Вход 2	Выход
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

21. Программные блоки (функции), использующиеся для ДНФ, называются

а) логическое «ИЛИ» (дизъюнкция), логическое «И» (конъюнкция), логическое «НЕ» (отрицание)

б) логическое «ИЛИ» (дизъюнкция), логическое «И» (конъюнкция), «исключающее ИЛИ»

в) логическое «ИЛИ» (дизъюнкция), логическое «НЕ» (отрицание)

г) логическое «И» (конъюнкция), логическое «НЕ» (отрицание)

22. Программные блоки (функции), использующиеся для КНФ, называются

а) логическое «ИЛИ» (дизъюнкция), логическое «И» (конъюнкция), логическое «НЕ» (отрицание)

б) логическое «ИЛИ» (дизъюнкция), логическое «И» (конъюнкция), «исключающее ИЛИ»

в) логическое «ИЛИ» (дизъюнкция), логическое «НЕ» (отрицание)

г) логическое «И» (конъюнкция), логическое «НЕ» (отрицание)

23. Программные блоки (функции), использующиеся для построения для полинома Жегалкина, называются

а) логическое «НЕ» (отрицание), логическое «И» (конъюнкция)

б) логическое «НЕ» (отрицание), логическое «ИЛИ» (дизъюнкция)

в) логическое «И» (конъюнкция), «исключающее ИЛИ»

г) логическое «ИЛИ» (дизъюнкция), «исключающее ИЛИ»

Основные функции и функциональные блоки на языке Function Block Diagram в PC WorX

1. Функция, изображенная на рис. 155, называется

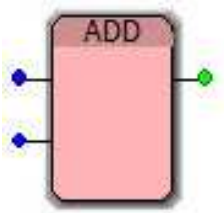


Рис. 155. Функция ADD

а) сложением

б) вычитанием

в) конъюнкцией

г) дизъюнкцией

2. Функция, изображенная на рис. 156, называется

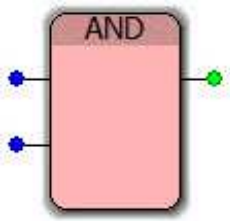


Рис. 156. Функция AND

а) сложением

б) вычитанием

в) конъюнкцией

г) дизъюнкцией

3. Функция, изображенная на рис. 157,

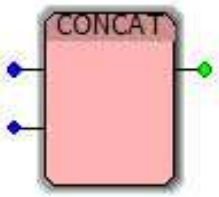


Рис. 157. Функция CONCAT

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2

б) извлекает фрагмент символьной строки, поданной на вход, начиная с некоторой позиции

в) объединяет две символьные строки путем добавления символьной строки в конец первой символьной строки

г) определяет положение строки в строке

4. Функциональный блок, изображенный на рис. 158, соответствует

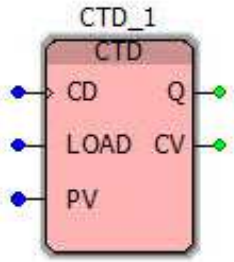


Рис. 158. Функциональный блок CTD

а) реверсивному счётчику

б) конъюнкции

в) счётчику, производящему обратный отсчёт входных импульсов по переднему фронту

г) счётчику, считающему входные импульсы по переднему фронту

5. Функциональный блок, изображенный на рис. 159, соответствует

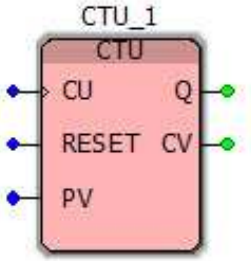


Рис. 159. Функциональный блок CTU

а) реверсивному счётчику
б) конъюнкции
в) счётчику, производящему обратный отсчёт входных импульсов по переднему фронту
г) счётчику, считающему входные импульсы по переднему фронту

6. Функциональный блок, изображенный на рис. 160, соответствует

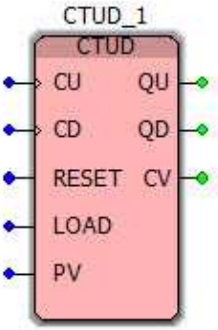


Рис. 160. Функциональный блок CTUD

а) реверсивному счётчику
б) конъюнкции
в) счётчику, производящему обратный отсчёт входных импульсов по переднему фронту
г) счётчику, считающему входные импульсы по переднему фронту

7. Функция, изображенная на рис. 161,

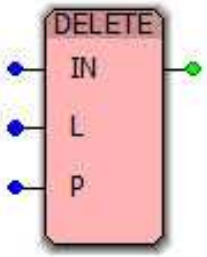


Рис. 161. Функция DELETE

а) делит указанное действительное число на некоторое рациональное число

б) извлекает фрагмент символьной строки, поданной на вход, начиная с некоторой позиции

в) делит указанное целое число на некоторое целое число

г) удаляет из символьной строки фрагмент с указанным числом символов, начиная с некоторой позиции

8. Функция, изображенная на рис. 162, называется

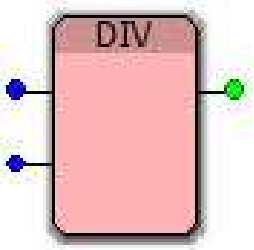


Рис. 162. Функция DIV

а) сложением

б) вычитанием

в) конъюнкцией

г) дизъюнкцией

9. Функция, изображенная на рис. 163, соответствует

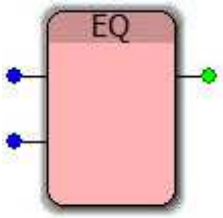


Рис. 163. Функция EQ

а) операции «Равенство»
б) операции «Больше или равно»
в) операции «Меньше или равно»
г) операции «Эквивалентность»

10. Функция, изображенная на рис. 164,

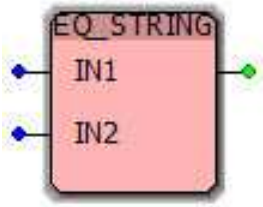


Рис. 164. Функция EQ_STRING

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строки IN1 и IN2 равны, и значение FALSE, если не равны

б) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше или равна строке IN2, в противном случае устанавливается значение FALSE

в) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше строки IN2, в противном случае устанавливается значение FALSE

г) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше строки IN2, в противном случае устанавливается значение FALSE

11. Функция, изображенная на рис. 165,

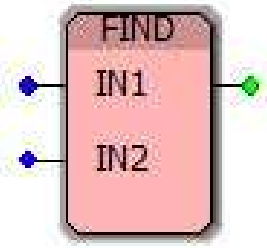


Рис. 165. Функция FIND

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2

б) извлекает фрагмент символьной строки, поданной на вход, начиная с некоторой позиции

в) объединяет две символьные строки путем добавления символьной строки в конец первой символьной строки

г) определяет положение строки в строке

12. Функция, изображенная на рис. 166, соответствует

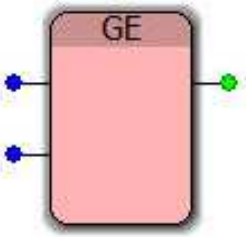


Рис. 166. Функция GE

а) операции «Равенство»

б) операции «Больше или равно»

в) операции «Меньше или равно»

г) операции «Не равно»

13. Функция, изображенная на рис. 167,

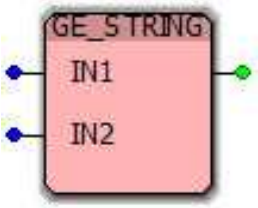


Рис. 167. Функция GE_STRING

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строки IN1 и IN2 равны, и значение FALSE, если не равны

б) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше или равна строке IN2, в противном случае устанавливается значение FALSE

в) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше строки IN2, в противном случае устанавливается значение FALSE

г) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше строки IN2, в противном случае устанавливается значение FALSE

14. Функция, изображенная на рис. 168, соответствует

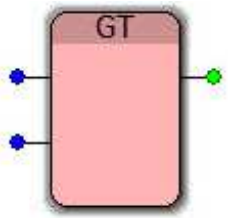


Рис. 168. Функция GT

а) операции «Меньше»
б) операции «Больше или равно»
в) операции «Меньше или равно»
г) операции «Больше»

15. Функция, изображенная на рис. 169,

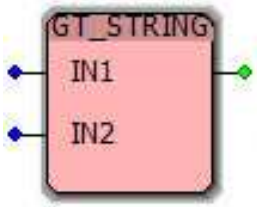


Рис. 169. Функция GT_STRING

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строки IN1 и IN2 равны, и значение FALSE, если не равны

б) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше или равна строке IN2, в противном случае устанавливается значение FALSE

в) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше строки IN2, в противном случае устанавливается значение FALSE

г) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше строки IN2, в противном случае устанавливается значение FALSE.

16. Функция, изображенная на рис. 170,

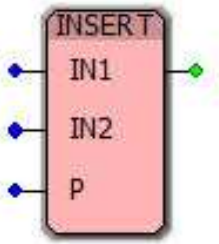


Рис. 170. Функция INSERT

а) вставляет строку с входа IN2 в строку на входе IN1 после символа на некоторой позиции

б) извлекает фрагмент символьной строки, поданной на вход, начиная с некоторой позиции

в) объединяет две символьные строки путем добавления символьной строки в конец первой символьной строки

г) определяет положение строки в строке

17. Функция, изображенная на рис. 171, соответствует

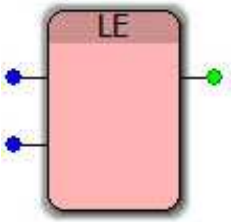


Рис. 171. Функция LE

а) операции «Меньше»
б) операции «Больше или равно»
в) операции «Меньше или равно»
г) операции «Больше»

18. Функция, изображенная на рис. 172,

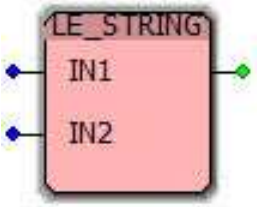


Рис. 172. Функция LE_STRING

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше строки IN2, в противном случае устанавливается значение FALSE.

б) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше строки IN2, в противном случае устанавливается значение FALSE

в) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше или равна строке IN2, в противном случае устанавливается значение FALSE

г) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше или равна строке IN2, в противном случае устанавливается значение FALSE

19. Функция, изображенная на рис. 173,

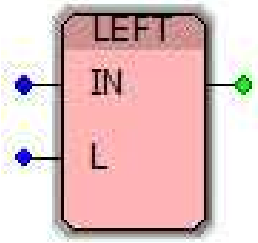


Рис. 173. Функция LEFT

а) извлекает фрагмент символьной строки, поданной на вход IN, начиная с некоторой позиции L слева

б) извлекает L символов слева символьной строки IN

в) извлекает фрагмент символьной строки, поданной на вход IN, начиная с некоторой позиции L справа

г) извлекает L символов справа символьной строки IN

20. Функция, изображенная на рис. 174,

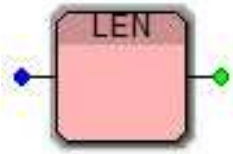


Рис. 174. Функция LEN

а) удаляет символьную строку

б) разделяет символьную строку пополам

в) объединяет две символьные строки в единую символьную строку

г) определяет длину символьной строки

21. Функция, изображенная на рис. 175, соответствует

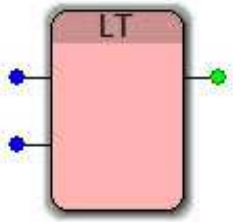


Рис. 175. Функция LT

а) операции «Меньше»
б) операции «Больше или равно»
в) операции «Меньше или равно»
г) операции «Больше»

22. Функция, изображенная на рис. 176,

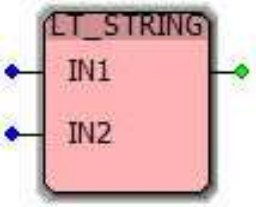


Рис. 176. Функция LT_STRING

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше строки IN2, в противном случае устанавливается значение FALSE.

б) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше строки IN2, в противном случае устанавливается значение FALSE

в) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 меньше или равна строке IN2, в противном случае устанавливается значение FALSE

г) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 больше или равна строке IN2, в противном случае устанавливается значение FALSE

23. Функция, изображенная на рис. 177,

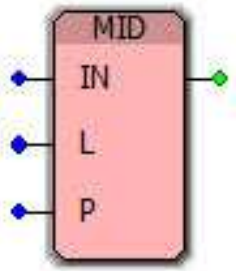


Рис. 177. Функция MID

а) сравнивает символьную строку IN с символьными строками L и P

б) извлекает фрагмент символьной строки, поданной на вход, начиная с некоторой позиции

в) объединяет две символьные строки путем добавления символьной строки в конец первой символьной строки

г) определяет положение строки в строке

24. Функция, изображенная на рис. 178,

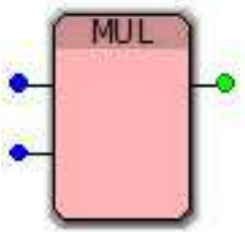


Рис. 178. Функция MUL

а) определяет длину символьной строки

б) определяет операцию деления двух чисел заданного типа данных

в) извлекает фрагмент символьной строки, поданной на вход, начиная с некоторой позиции

г) определяет произведение двух чисел заданного типа данных

25. Функция, изображенная на рис. 179, соответствует

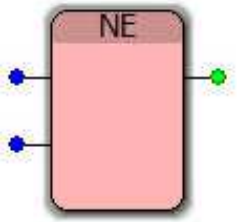


Рис. 179. Функция NE

а) операции «Отрицание»
б) операции «Не больше»
в) операции «Не равно»
г) операции «Не меньше»

26. Функция, изображенная на рис. 180,

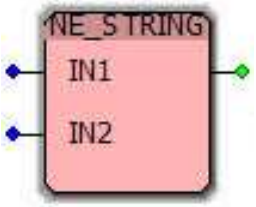


Рис. 180. Функция NE_STRING

а) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строки IN1 и IN2 равны, и значение FALSE, если не равны

б) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 не больше строки IN2, в противном случае устанавливается значение FALSE

в) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строки IN1 и IN2 не равны, и значение FALSE, если равны

г) сравнивает символьную строку с входа IN1 с символьной строкой с входа IN2, устанавливая на выходе значение TRUE, если строка IN1 не меньше строки IN2, в противном случае устанавливается значение FALSE

27. Функция, изображенная на рис. 181,

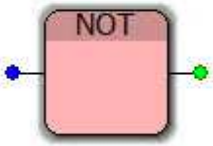


Рис. 181. Функция NOT

а) сравнивает две символьные строки
б) соответствует отрицанию
в) соответствует операции «Не равно»
г) удаляет символьную строку

28. Функция, изображенная на рис. 182,

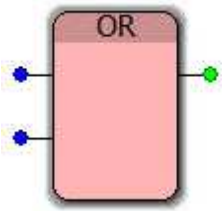


Рис. 182. Функция OR

а) сравнивает две символьные строки
б) соответствует дизъюнкции
в) определяет длину символьной строки
г) соответствует конъюнкции

29. Функция, изображенная на рис. 183,

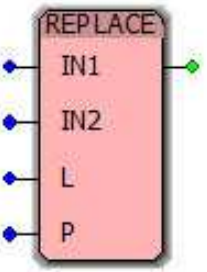


Рис. 183. Функция REPLACE

а) заменяет фрагмент символьной строки с входа IN1 на строку с входа IN2, начиная с некоторой позиции

б) извлекает фрагмент символьной строки, начиная с позиции P

в) вставляет строку с входа IN2 в строку на входе IN1 после символа на позиции P

г) объединяет две символьные строки путем добавления символьной строки на входе IN2 в конец символьной строки на входе IN1

30. Функция, изображенная на рис. 184,

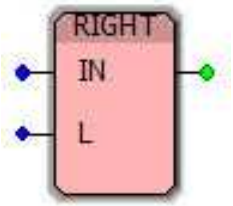


Рис. 184. Функция RIGHT

а) извлекает фрагмент символьной строки, поданной на вход IN, начиная с некоторой позиции L слева

б) извлекает L символов слева символьной строки IN

в) извлекает фрагмент символьной строки, поданной на вход IN, начиная с некоторой позиции L справа

г) извлекает L символов справа символьной строки IN

31. Функция, изображенная на рис. 185, приведёт к образованию битовой строки, полученной

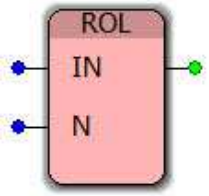


Рис. 185. Функция ROL

а) перемещением *N* бит из конца строки в ее начало

б) сдвигом вправо на *N* бит фрагмента строки, начиная с *N*+1, причём оставшиеся битовые позиции слева заполнятся нулями

в) сдвигом влево на *N* бит фрагмента строки, начиная с *N*+1

г) перемещением *N* бит из начала строки в ее конец

32. Функция, изображенная на рис. 186, приведёт к образованию битовой строки, полученной

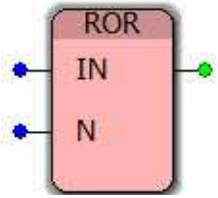


Рис. 186. Функция ROR

а) перемещением *N* бит из конца строки в ее начало
б) сдвигом вправо на *N* бит фрагмента строки, начиная с *N*+1, причём оставшиеся битовые позиции слева заполнятся нулями
в) сдвигом влево на *N* бит фрагмента строки, начиная с *N*+1
г) перемещением *N* бит из начала строки в ее конец

33. Функция, изображенная на рис. 187, соответствует

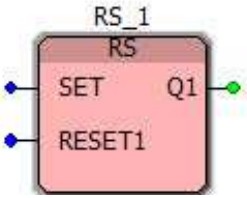


Рис. 187. Функция RS

а) таймеру
б) счётчику
в) триггеру
г) оператору сравнения

34. Функция, изображенная на рис. 188, соответствует триггеру,

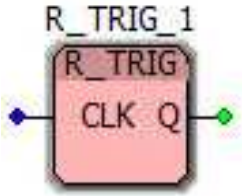


Рис. 188. Функция R_TRIG

а) детектирующему фронт входного сигнала с FALSE на TRUE по переднему фронту

б) детектирующему фронт входного сигнала с TRUE на FALSE по заднему фронту

в) детектирующему фронт входного сигнала с FALSE на TRUE по заднему фронту

г) детектирующему фронт входного сигнала с TRUE на FALSE по переднему фронту

35. Функция, изображенная на рис. 189, приведёт к образованию битовой строки, полученной

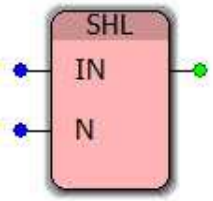


Рис. 189. Функция SHL

а) перемещением *N* бит из конца строки в ее начало

б) сдвигом вправо на *N* бит фрагмента строки, начиная с *N*+1, причём оставшиеся битовые позиции слева заполнятся нулями

в) сдвигом влево на *N* бит фрагмента строки, начиная с *N*+1

г) перемещением *N* бит из начала строки в ее конец

36. Функция, изображенная на рис. 190, приведёт к образованию битовой строки, полученной

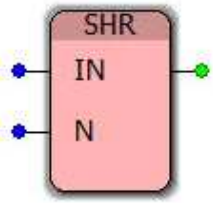


Рис. 190. Функция SHR

а) перемещением *N* бит из конца строки в ее начало

б) сдвигом вправо на *N* бит фрагмента строки, начиная с *N*+1, причём оставшиеся битовые позиции слева заполнятся нулями

в) сдвигом влево на *N* бит фрагмента строки, начиная с *N*+1
г) перемещением *N* бит из начала строки в ее конец

37. Функция, изображенная на рис. 191, соответствует

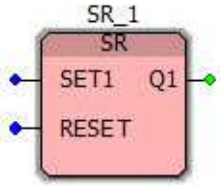


Рис. 191. Функция SR

а) таймеру
б) триггеру
в) счётчику
г) оператору сравнения

38. Функция, изображенная на рис. 192, называется

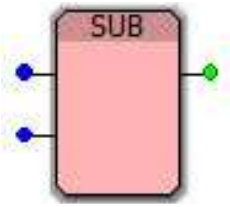


Рис. 192. Функция SUB

а) сложением
б) умножением
в) делением
г) вычитанием

39. Функциональный блок, изображенный на рис. 193, реализует

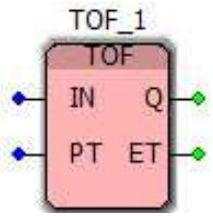


Рис. 193. Функциональный блок TOF

а) импульсный таймер
б) таймер с задержкой по включению
в) таймер с задержкой по выключению
г) триггер

40. Функциональный блок, изображенный на рис. 194, реализует

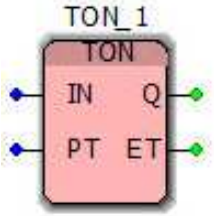


Рис. 194. Функциональный блок TON

а) импульсный таймер
б) таймер с задержкой по выключению
в) таймер с задержкой по включению
г) триггер

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Деменков, Н.П. Языки программирования промышленных контроллеров: учеб. пособие / н.п. Деменков; под ред. К.А. Пупкова. – М.: Изд-во МГТУ им. Н.Э Баумана, 2004. – 172 с.

2. Зайцев, М.В. Маленький контроллер для больших задач или программирование и составление программы ПЛК / М.В. Зайцев // Автоматизация в промышленности. – 2010. – № 12. – С. 15–17.

3. Зайцев, М.В. Маленький контроллер для больших задач или программирование и составление программы ПЛК / М.В. Зайцев // Автоматизация в промышленности. – 2011. – № 1. – С. 29–30.

4. Зайцев, М.В. Маленький контроллер для больших задач или программирование и составление программы ПЛК / М.В. Зайцев // Автоматизация в промышленности. – 2011. – № 2. – С. 13–16.

5. Минаев, И.Г. Программируемые логические контроллеры: практическое руководство для начинающего инженера / И.Г. Минаев, В.В. Самойленко. – Ставрополь: АГРУС, 2009. – 100 с.

6. Митин, Г.П. Системы автоматизации с использованием программируемых логических контроллеров: учеб. пособие / Г.П. Митин, О.В. Хазанова. – М.: ИЦ МГТУ «Станкин», 2005. – 136 с.

7. Мишунин, В.В. Информационно-измерительные и управляющие систем: учебно-методическое пособие / В.В. Мишунин, Е.В. Корсунова, В.И. Ищенко, А.В. Курлов. – Белгород: Изд-во БелГУ, 2010. – 129 с.

8. Парр, Э. Программируемые контроллеры: руководство для инженера / Э. Парр; пер. 3-го англ. изд. – М.: БИНОМ. Лаборатория знаний, 2007. – 516 с.

9. Петров, И.В. Программируемые контроллеры. Стандартные языки и приёмы прикладного проектирования / И.В. Петров; под ред. проф. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2004. – 256 с.

10. Седов, В.А. Задачи булевой алгебры на языке Function block diagram / В.А. Седов, Н.А. Седова // Информационно-телекоммуникационные системы и технологии Всероссийская научно-практическая конференция. – Кемерово, 2015. – С. 245.

11. Седов, В.А. Использование языка Function Block Diagram для реализации треугольных функций принадлежности / В.А. Седов, Н.А. Седова // Информационные технологии. Радиоэлектроника. Телекоммуникации. (ITRT-2016): сб. ст. VI международной заочной научно-технической кон-

ференции. Ч. 2 / Поволжский гос. ун-т сервиса. – Тольятти: Изд-во ПВГУС, 2016. – С. 194–196.

12. Седов, В.А. Программируемые логические контроллеры на языке Function block diagram: практикум / В.А. Седов, Н.А. Седова. – Владивосток: Мор. гос. ун-т, 2016. – 178 с.

13. Седова, Н.А. Функции принадлежности нечётких множеств на языке Function Block Diagram / В.А. Седов, Н.А. Седова // Решение: материалы четвертой Всероссийской научно-практической конференции. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2015. – Т. 1. – С. 249–251.

14. Седова, Н.А. ШИМ-регулирование с аналоговым управлением на ПК ILC 131 / В.А. Седов, Н.А. Седова // Решение: материалы пятой Всероссийской научно-практической конференции. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2016.

15. Технические средства автоматизации: программирование контроллеров в среде ISaGRAF: лабораторные работы / сост.: И.А. Елизаров, А.А. Третьяков, В.Н. Назаров, М.Н. Солуданов. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2008. – 24 с.

16. Уоррен, Генри С. Алгоритмические трюки для программистов / Г.С. Уоррен; пер. с англ. – 2-е изд.– М.: ООО «И. Д. Вильямс», 2014. – 512 с.

ОГЛАВЛЕНИЕ

Учебное издание

Седова Нелли Алексеевна
Седов Виктор Александрович

СМАРТ-ТЕХНОЛОГИИ

ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОКОВЫХ ДИАГРАММ

В авторской редакции
Компьютерная верстка М.А. Портновой

Подписано в печать 21.02.2017. Формат 60×84/16.
Бумага писчая. Печать офсетная. Усл. печ. л. 12,55.
Тираж 200 экз. Заказ

Издательство Владивостокского государственного университета экономики и сервиса
690014, Владивосток, ул. Гоголя, 41
Отпечатано в Множительном участке Издательства ВГУЭС
690014, Владивосток, ул. Гоголя, 41