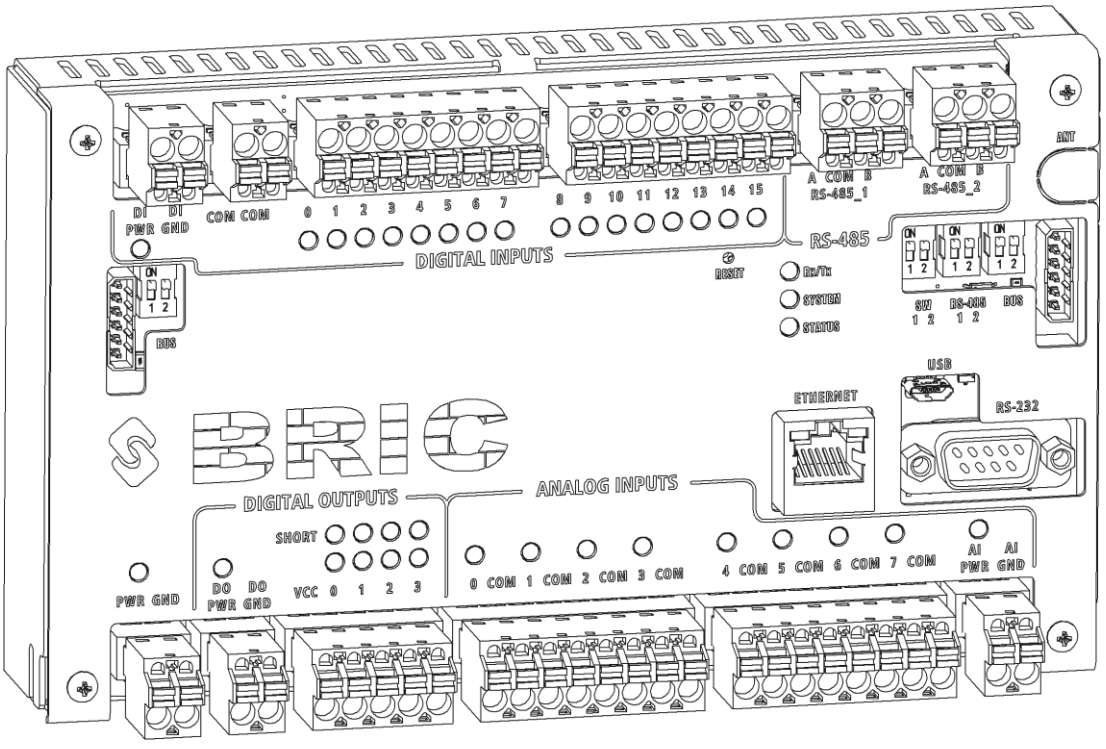
ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«СНЭМА–СЕРВИС»

ПРОГРАММИРУЕМЫЙ ЛОГИЧЕСКИЙ КОНТРОЛЛЕР BRIC

РУКОВОДСТВО ПО ПРОГРАМИРОВАНИЮ ПЛК BRIC В СРЕДЕ РАЗРАБОТКИ BEREMIZ

СНС 1.001.001 ИС(ИНСРУКЦИЯ ЭКСПЛУАТАЦИОННАЯ СПЕЦИАЛЬНАЯ)

Издание 1



УФА 2020

Оглавление

1 КРАТКОЕ ПРЕДСТАВЛЕНИЕ О ИСР BEREMIZ

Данная версия интегрированной среды разработки (ИСР) Beremiz, поставляемая с ПЛК BRIC, предназначена для создания и отладки прикладных программ на языках стандарта IEC 61131–3 для целевых устройств ПЛК BRIC. В качестве языков описания алгоритмов и логики работы программ выступают текстовые Structured Text (далее ST) и Instruction List (далее IL), графические Function Block Diagram (далее FBD), Ladder Diagram (далее LD), Sequential Function Chart (далее SFC).

ИСР Beremiz может выполняться на операционных системах Windows. Необходимые библиотеки языков Python, C, C++ прилагаются вместе с ИСР Beremiz.

Техническими требованиями для работы ИСР Beremiz является персональный компьютер, для которого минимальными требованиями являются:

- Тактовая частота процессора 1000 МГц (32–битные/64–битные);
- 1 Гб оперативной памяти;
- Операционная система Windows XP/Vista/7/10;
- Монитор (для оптимальной работы не меньше 17 дюймов);
- Клавиатура, мышь (или устройства, полноценно заменяющего мышь).

2 ЗАПУСК ИНТЕГРИРОВАННОЙ СРЕДЫ РАЗРАБОТКИ BEREMIZ И ОБЗОР ГЛАВНОЙ СТРАНИЦЫ

Для создания нового проекта для ПЛК BRIC необходимо зайти в ИСР Beremiz, обозначенную значком, указанным на рисунке 1.

Рисунок 1 – Иконка ИСР Beremiz

ИСР Beremiz имеет несколько областей на своем главном окне:

- Панель инструментов;
- Дерево проекта;
- Панель переменных и констант;
- Панель библиотеки функций и функциональных блоков (ФБ);
- Панель экземпляров проекта;
- Отладочная панель (подобнее указано в Приложении А).

3 СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОЙ ПРОГРАММЫ

ШАГ 1. СОЗДАНИЕ НОВОГО ПРОЕКТА

Для создания проекта в ИСР Beremiz необходимо нажать на кнопку «Создать новый проект» на панели инструментов указанную на рисунке 2:

Рисунок 2 – Кнопка «Создать новый проект»

При создании папки нового проекта необходимо знать, что ее наименование должно прописываться буквами латинского алфавита без наличия букв кириллического алфавита и пробелов. Путь расположения папки не должен иметь буквы латинского алфавита. Пример такого использования указан на рисунке 3.

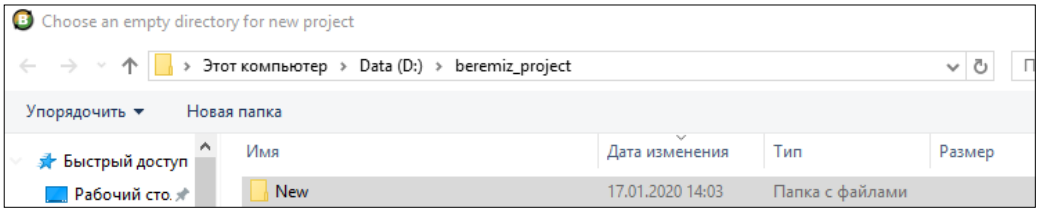


Рисунок 3 – Создание папки нового проекта и указание его расположения

ШАГ 2. КОНФИГУРИРОВАНИЕ ПРОЕКТА

После указания расположения проекта во всплывающем окне, необходимо сконфигурировать основные параметры программы проекта (см. рисунок 4):

- Имя программного компонента;
- Тип программного компонента (в первом программном компоненте выбор ограничен одним вариантом: программа);
- Используемый язык программирования (стандарта IEC 61131–3: FBD, IL, ST, SFC, LD).

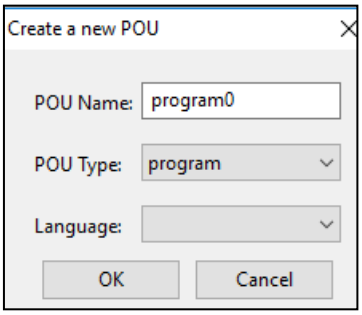


Рисунок 4 – Кнопка «Создать новый программный компонент»

После настройки программного компонента необходимо перейти к главной странице проекта как показано на рисунке 5.

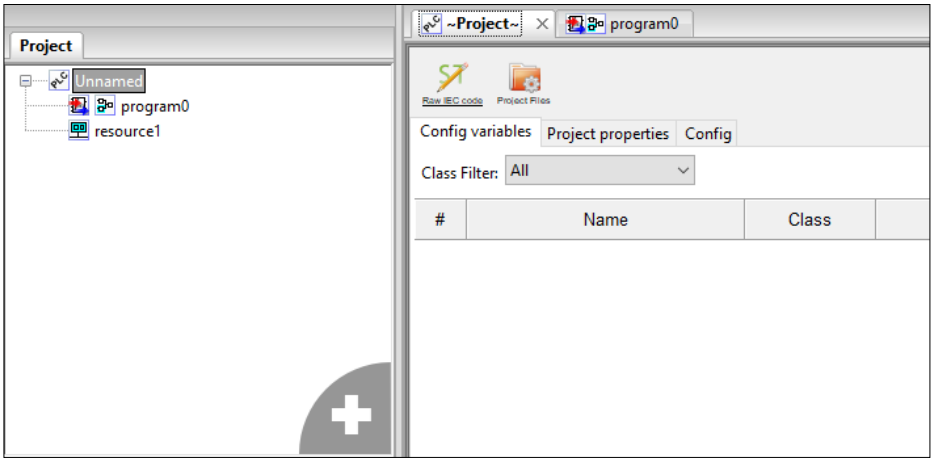


Рисунок 5 – Переход на главную страницу проекта

Далее необходимо настроить тип платформы. ПЛК BRIC и модули расширения имеют архитектуру целевой платформы «Sofi», поэтому во вкладке «Config» в разделе TargetType требуется установить целевую платформу «Sofi» (см. рисунок 6).

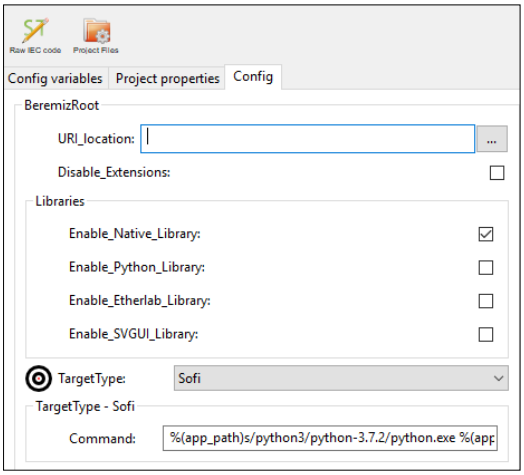


Рисунок 6 – Переход в конфигурацию проекта

ШАГ 3. СБОРКА ПРОЕКТА

Для добавления нового элемента в дерево проекта необходимо щелкнуть левой клавишей мыши в окне дерева проекта под перечисленным списком дерева проекта и выбирать один из следующих пунктов (см. рисунок 7):

- «Data Type» (Типы данных)
- «Function» (Функции)
- «Function Block» (ФБ)
- «Program» (Программы)
- «Resource» (Ресурсы)
- «Modbus support» (Поддержка Modbus)
- «Modules support» (Поддержка Modules)

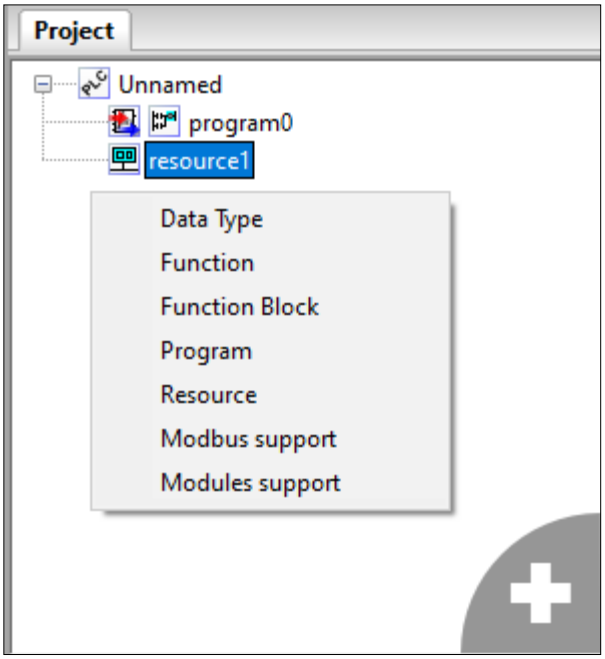


Рисунок 7 – Добавление нового элемента в дерево проекта

Далее прописывается код алгоритма проекта. Для переменных ФБ и функций необходимо выбрать параметры переменных (см. Приложение Г).

Все глобальные переменные, используемые в проекте, должны быть указаны на главной странице проекта. В параметрах проекта (см. рисунок 8) вносятся следующие изменения:

- Все используемые элементы типа «Program» должны быть указаны в экземплярах (Instances);
- В задачах (Tasks) прописываются экземпляры, и они должны иметь:
    - тип исполнения (Interrupt, Cyclic);
    - время (не менее 1 мс), необходимое на исполнение (Interval).

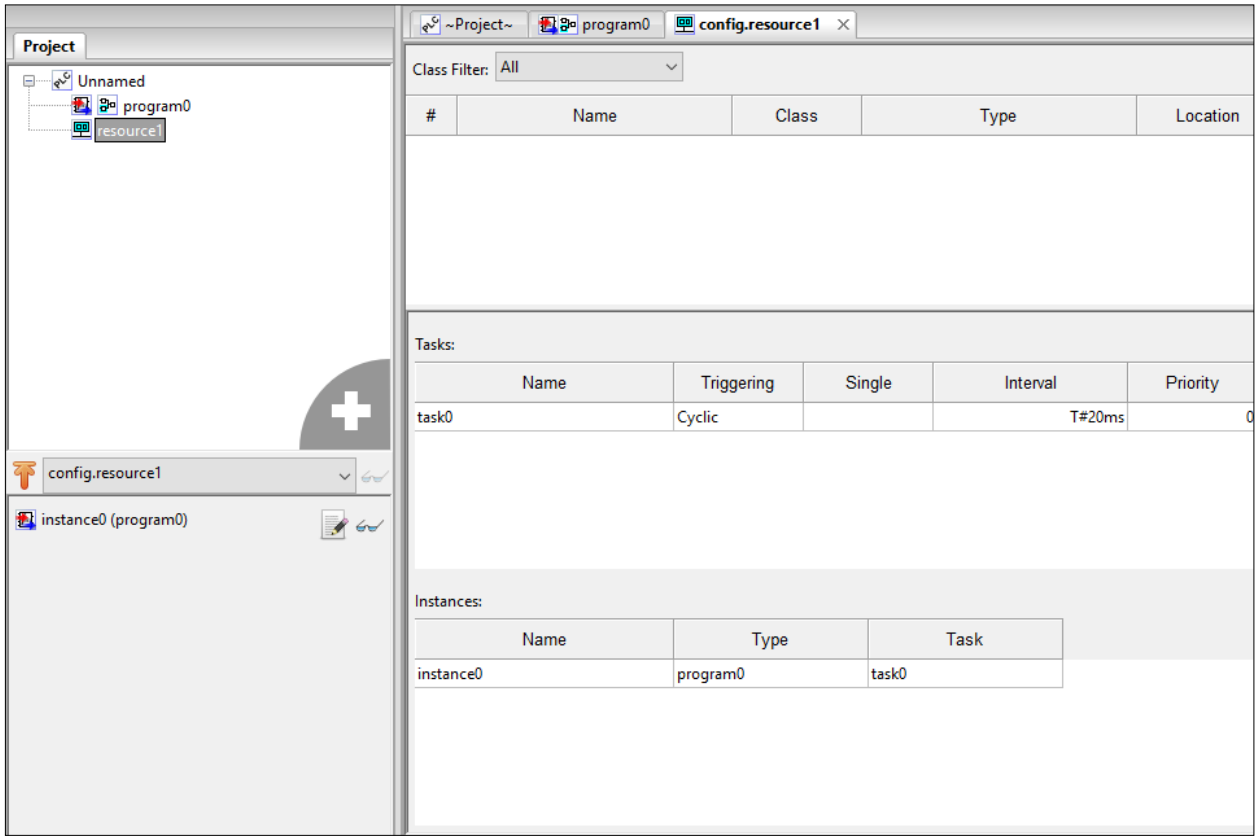


Рисунок 8 – Переход в параметры проекта

Далее необходимо собрать проект. Для этого нажимаем на кнопку «Собрать проект» на панели инструментов указанную на рисунке 9.

Рисунок 9 – Кнопка «Собрать проект»

Результат сборки проекта указывается на отладочной панели. Если в выведенном сообщении имеются строки, подчеркнутые красной линией как на рисунке 10, то проект считается готовым для загрузки в ПЛК.

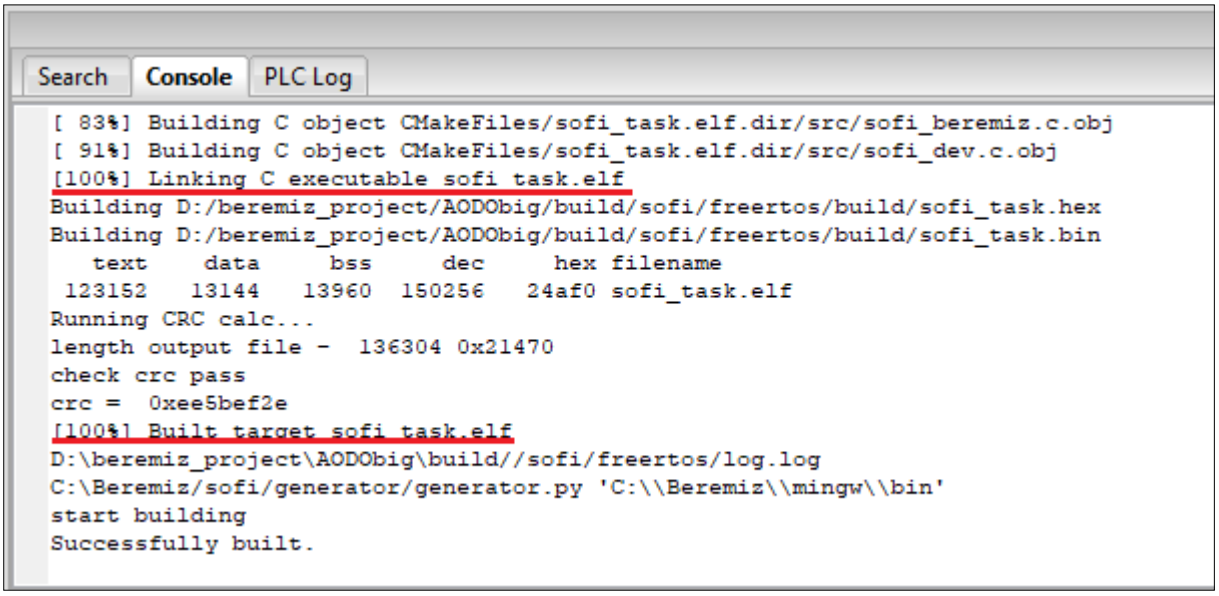


Рисунок 10 – Сообщение об успешной сборке проекта

Итоговый бинарный файл, который будет загружен в ПЛК, находится в папке проекта с названием *sofi_task_crc.bin*. На рисунке 11 подчеркнут путь в папке проекта файла и название файла.

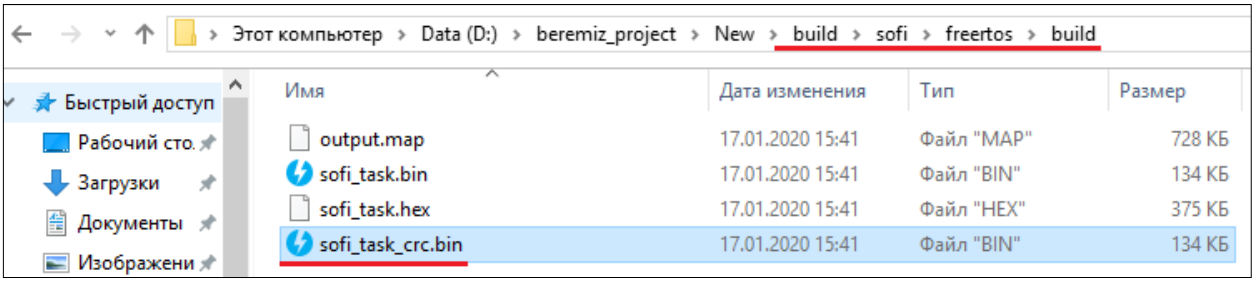


Рисунок 11 – Расположение итогового файла проекта

4 СТАНДАРТНЫЕ СРЕДСТВА КОНФИГУРИРОВАНИЯ (MODBUS)

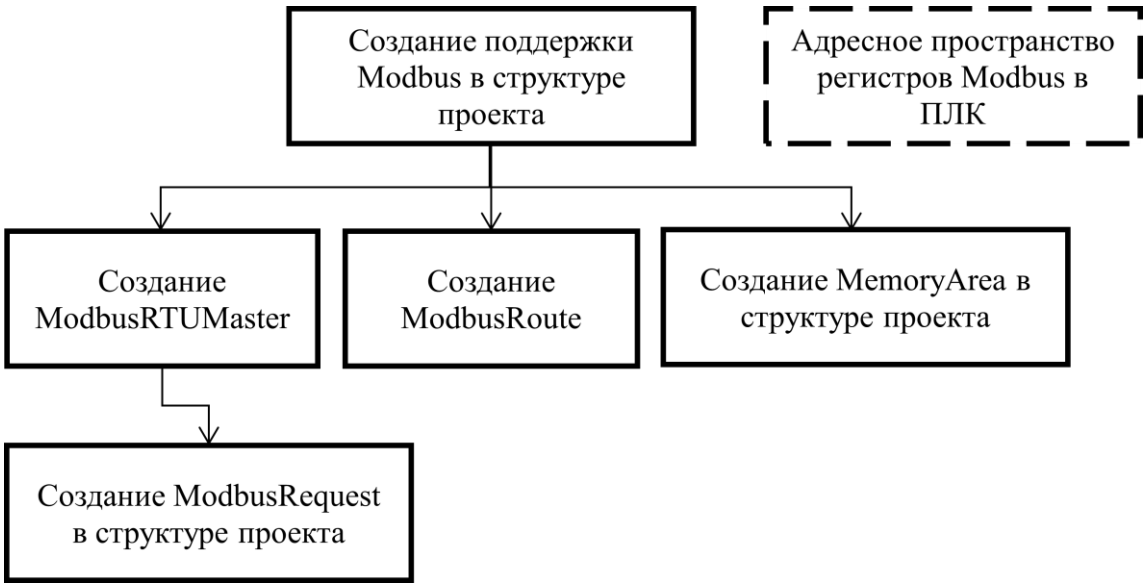
Для ПЛК BRIC имеется возможность опроса данных по протоколу Modbus, ретрансляции и расширения адресного пространства. Для этого необходимо заложить в структуру проекта элемент «Поддержка Modbus» (см. раздел 3) и элемент для выполнения необходимого функционала:

- ModbusRTUMaster (фоновый опрос устройств для чтения и записи данных по протоколу Modbus);
- ModbusRoute (ретрансляция пакетов из одного канала в другой);
- ModbusArea (расширение адресного пространства).

По умолчанию, Modbus адресное пространство имеет зарезервированную область адресов:

0x30000–0x39999	Область адресов, выделенных под чтение и запись массивов пользовательской программы
0x40000–0x49999	Область адресов, выделенных под чтение и запись регистров пользовательской программы
0x60000–0x69999	Область адресов, выделенных под системные регистры ПЛК

Настройка интерфейсов и протоколов обмена в ИСР Beremiz имеет следующую последовательность действий:



4.1 Создание подмодуля ModbusRTUMaster

Обмен данными со сторонними устройствами по Modbus осуществляет элемент «ModbusRTUMaster». Для подключения необходимо добавить его в элемент «Поддержка Modbus», щелкнув правой клавишей мыши и выбрав пункт «Добавить ModbusRTUMaster» (см. рисунок 12). Затем в окне конфигурации (см. рисунок 13) настроить канал опроса. Для каждого физического канала может быть не более одного элемента «ModbusRTUMaster».

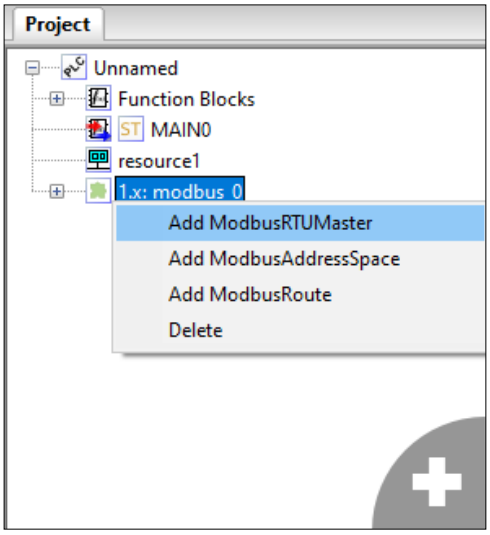


Рисунок 12 – Добавление в структуру проекта ModbusRTUMaster

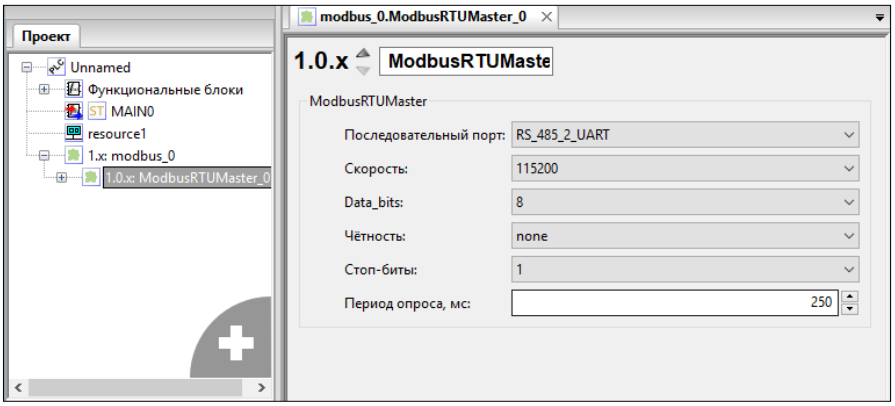


Рисунок 13 – Выбор параметров ModbusRTUMaster

Установка параметров ModbusRTUMaster включает следующие позиции:

- Последовательный порт (выбор порта из перечня: RS_232_UART, RS_485_1_UART, RS_485_2_UART);
- Скорость передачи данных (выбор производится из установленного ряда: 1200, 2400, 4800, 9600,14400, 19200, 28800, 38400, 56000 57600, 76800, 115200 бит/с.);
- Количество информационных битов;
- Тип чётности байта при её наличии (выбор производится из установленного ряда: even (чётный), none (без чётности), odd (нечётный));

- Количество Стоп–битов;
- Период опроса.

Далее необходимо добавить элементы «ModbusRequest» (см. рисунок 14) и настроить их согласно рисунку 15. Разрешается добавлять несколько элементов с разными параметрами, тогда они будут встроены в порядок опроса последовательно.

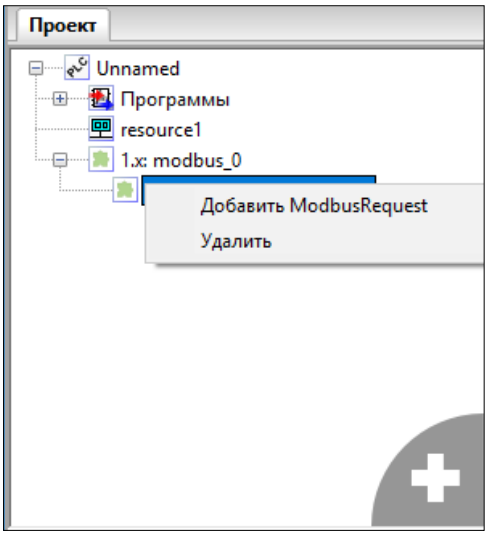


Рисунок 14 – Добавление в структуру проекта ModbusRequest

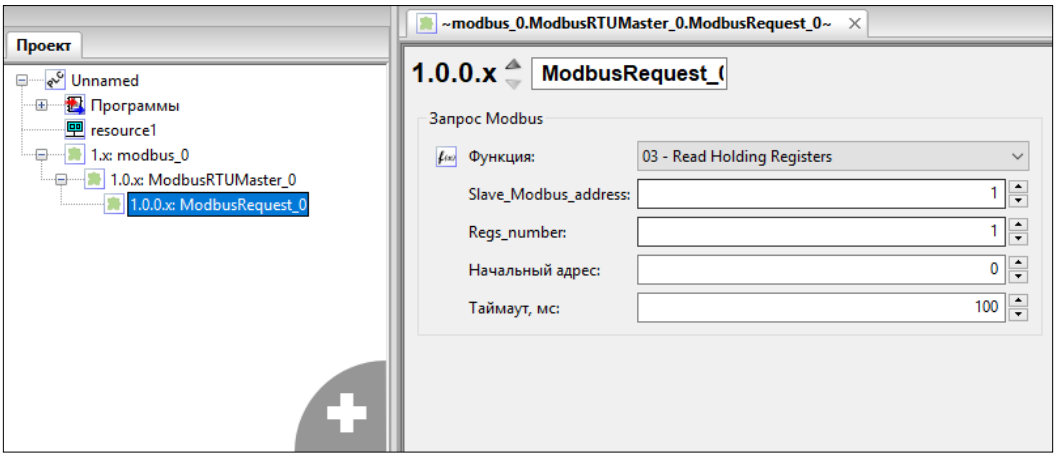
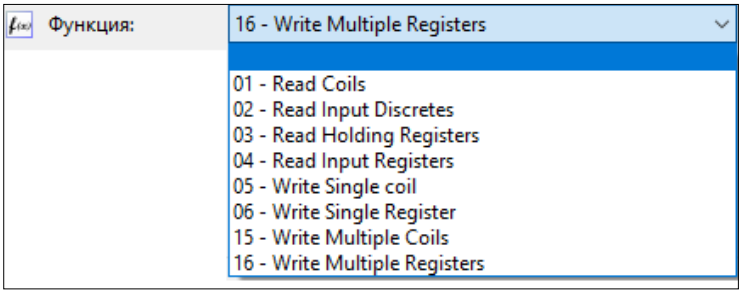


Рисунок 15 – Выбор параметров ModbusRequest

Установка параметров ModbusRequest включает следующие позиции:

- Выбор команды (описание команд указано в Приложении Б);



- Адрес slave–устройства с которым производится обмен данными по Modbus протоколу;
- Число Reg/Coil в одном пакете (для Reg до 120);
- Адрес первого Reg/Coil в пакете согласно адресного пространства slave–устройства;
- Таймаут в мс (не должен превышать период опроса).

После установления необходимых параметров «ModbusRequest» необходимо обозначить глобальные переменные, которые используются при передаче данных по протоколу Modbus. Для этого в панели переменных и констант необходимо для переменной в ячейке «Location» записать ссылку на адрес. Подробное описание конфигурирования см. в разделе 4.4

4.2 Создание ModbusRoute

ПЛК BRIC имеет возможность ретранслировать пакеты из одного канала в другой. Также есть возможность приёма-передачи пакетов Modbus TCP в Modbus RTU и Modbus RTU – Modbus RTU. Для подключения подмодуля «ModbusRoute» необходимо подвести курсор к созданной ветке «Поддержка Modbus», щелкнуть правой клавишей мыши и выбрать пункт «Добавить ModbusRoute» (см. рисунок 16).

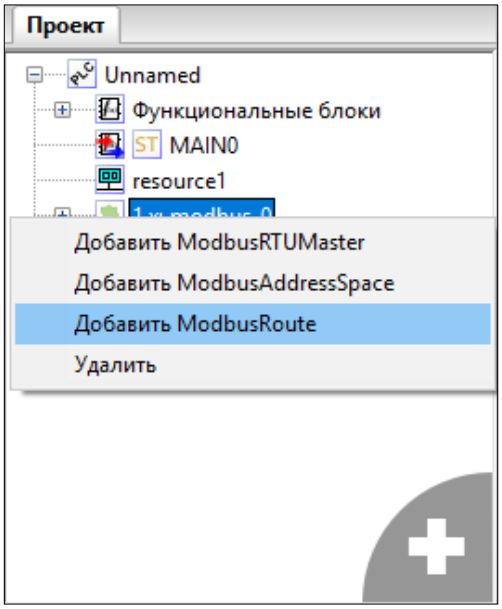


Рисунок 16 – Добавление в структуру проекта ModbusRoute

Окно конфигурирования ModbusRoute представлено на рисунке 17.

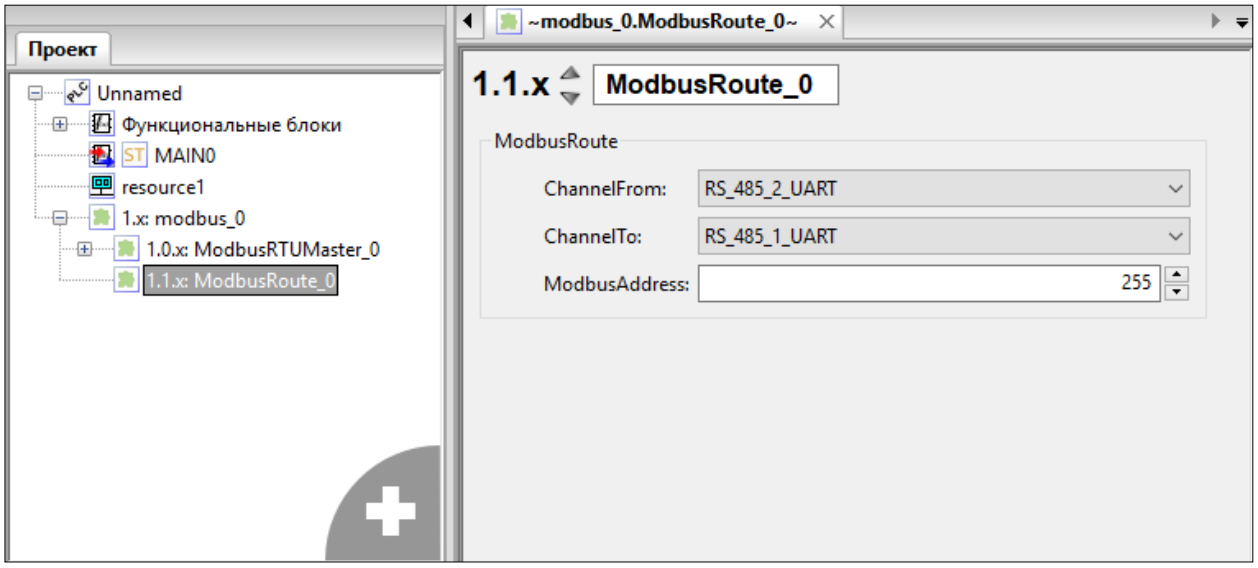
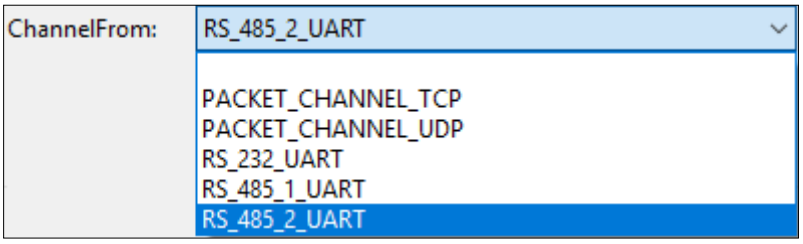


Рисунок 17 – Выбор параметров ModbusRoute

Установка параметров ModbusRoute включает следующие позиции:

- Задействованные каналы (выбрать порт из перечня: PACKET_CHANNEL_TCP, RS_232_UART, RS_485_1_UART, RS_485_2_UART, PACKET_CHANNEL_UDP)[1];



- ModbusAddress (Modbus адрес устройства, для которого производится ретрансляция из одного канала в другой)[2].

4.3 Создание ModbusArea

ПЛК BRIC имеет возможность увеличить адресное пространство для каждого типа регистров (Coils, Input Discrete, Input Registers, Holding Registers) при помощи подключения подмодуля «ModbusArea». Для подключения «ModbusArea» необходимо подвести курсор к созданной ветке «Поддержка Modbus», щелкнуть правой клавишей мыши и выбрать пункт «Добавить ModbusArea» (см. рисунок 18).

[1] PACKET_CHANNEL_TCP, PACKET_CHANNEL_UDP данные передаваемые протоколами (TCP, UDP) через канал связи Ethernet.
[2] При указании адреса 255 ретранслирует все пакеты полученные с канала «извлечения» в канал «записи».

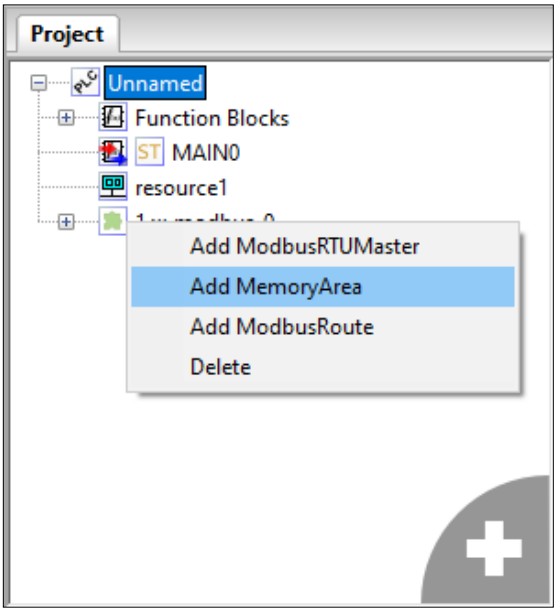


Рисунок 18 – Добавление в структуру проекта ModbusArea

Окно конфигурирования ModbusArea показано на рисунке 19.

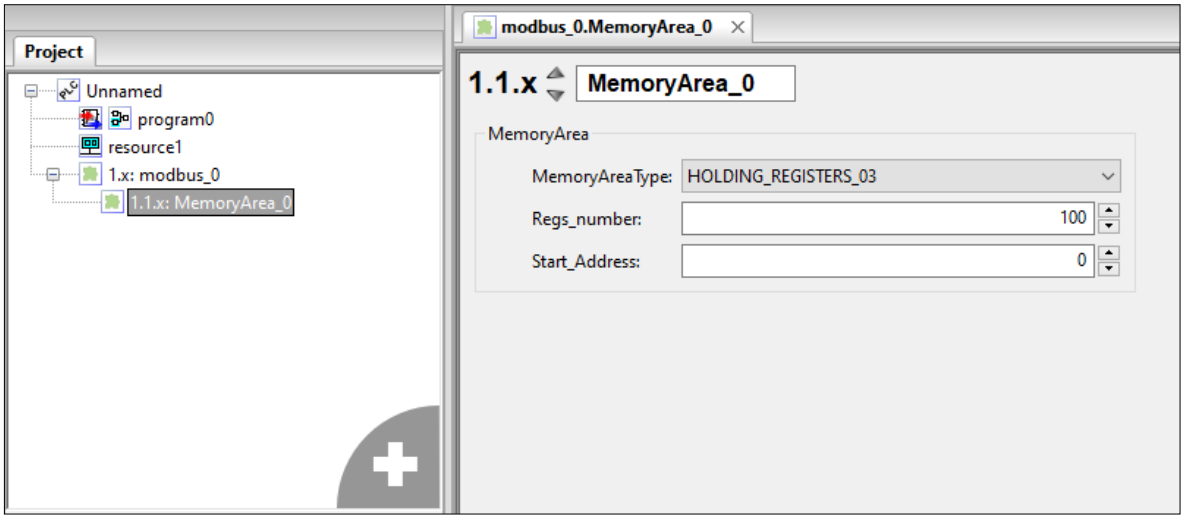
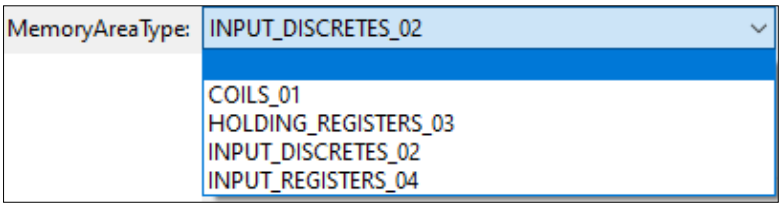


Рисунок 19 – Выбор параметров ModbusArea

Установка параметров ModbusArea включает следующие позиции:

- Выбор типа регистра адресного пространства (Coils, Input Discrete, Input Registers, Holding Registers);



- Regs_number – количество регистров в адресном пространстве[3]
- Start_Address – первый Modbus-адрес адресного пространства.

Расположение разных типов регистров независимо, поэтому номера регистров разных типов могут иметь одинаковое значение. Ограничением по количеству регистров является:

- Размер итогового файла сборки проекта, указываемого в отладочной панели под элементом «dec» строки (text data bss dec hexfilename), составляющий не более «197524»;
- Количество регистров в адресном пространстве (65530).

4.4 Привязка глобальным переменным Modbus адреса

Для запроса с ПЛК информации хранящейся в пользовательских регистрах, необходимо интересуемым переменным присвоить Modbus адреса. Для этого в панели переменных и констант для необходимой переменной в ячейке «Location» требуется записать Modbus адрес (см. рисунок 20)[4].

[3] Modbus-адреса задаваемые в разных адресных пространствах для одинаковых типов регистров не должны повторяться

[4] Для устройств, опрашивающих ПЛК BRIC имеется возможность как использовать команды чтения, так и записи регистров.

#	Name	Class	Type	Location	Initial Value
1	REG1	Global	UINT	%MW0.3	
2	REG2	Global	UINT	%MW1.1.0	
3	REG4	Global	UINT	%IW1.0.0.0	
4	REG5	Global	UINT	%QW1.0.1.0	
5	REG3	Global	ARRAY [0..4] OF UINT		[0, 4, 98, 45, 9]

Рисунок 20 – Пример создания переменных с Modbus адресами

Структура записи адреса приведена ниже:

1 2 3 4
%[Форма][Размер][Идентификатор].[Номер]

1. Форма переменной.

Тип формы	Описание
Q	Данная глобальная переменная используется для записи Reg/Coil slave–устройства (WriteSingleCoil, WriteSingleRegister, WriteMultipleCoils, WriteMultipleRegisters)
I	Данная глобальная переменная используется для чтения Reg/Coil slave–устройства (ReadCoils, ReadInputDiscretes, ReadHoldingRegisters, ReadInputRegisters)
M	Данная глобальная переменная используется для записи чтения

2. Размер переменной.

Размер	Количество байтов	Тип данных
D	4	DINT, REAL, UDINT, DWORD
L	8	LINT, ULINT, LREAL, LWORD
B	1	BYTE, USINT, SINT
X	1	BOOL
W	2	WORD, INT, UINT

3. Идентификатор элемента.

Структура	Предназначение
X.X.X	Для ModbusRequest
X.X	Для ModbusArea
X	Для остальных, не входящих в модуль расширения

4. Номер регистра.

Номер выставляется согласно номеру в выборке ModbusRequest, при этом номер первого регистра равен 0.

5 ДОБАВЛЕНИЕ МОДУЛЕЙ В ПЛК BRIC

В ПЛК BRIC реализовано подключение нескольких устройств линейки BRIC по межмодульной шине. Для этого необходимо добавить в структуру проекта, созданного в ИСР Beremiz, «Поддержка modules» (см. раздел 3). Для добавления нового устройства необходимо подвести курсор к созданной ветке «Поддержка modules», щелкнуть левой клавишей мыши и выбрать пункт «Добавить AO/DO/DI/AI/BRIC», как показано на рисунке 21.

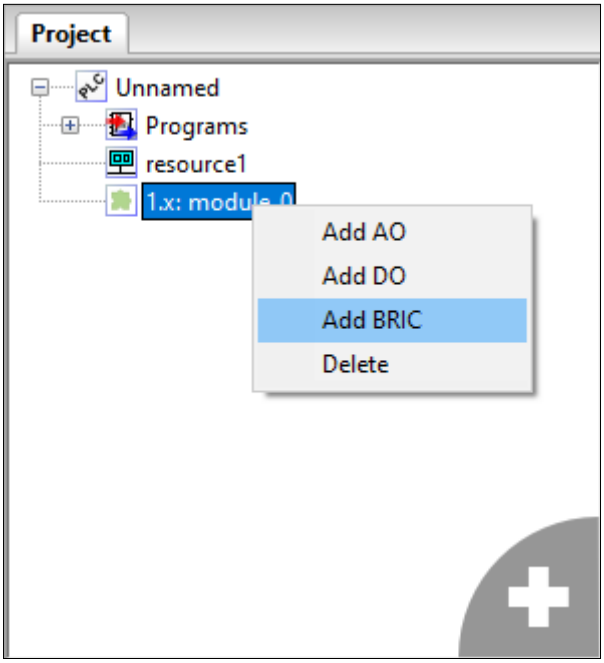
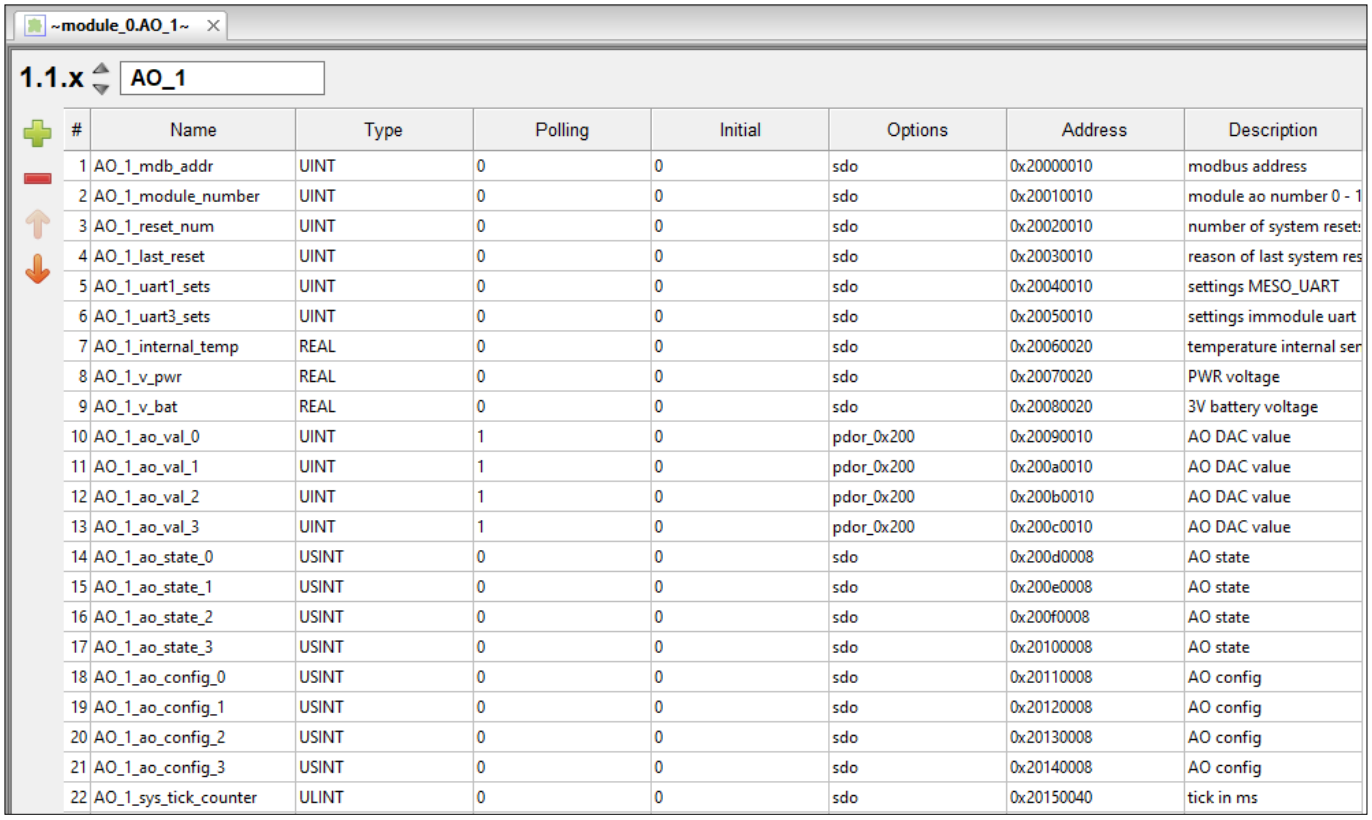


Рисунок 21 – Добавление в структуру проекта модуля расширения/ПЛК по межмодульной шине

В окне добавленного модуля автоматически присваивается номер модуля расширения/ПЛК в межмодульной шине, пользователь имеет возможность изменить номер (верхний левый угол окна) (см. рисунок 22).



#	Name	Type	Polling	Initial	Options	Address	Description
1	AO_1_mdb_addr	UINT	0	0	sdo	0x20000010	modbus address
2	AO_1_module_number	UINT	0	0	sdo	0x20010010	module ao number 0 - 1
3	AO_1_reset_num	UINT	0	0	sdo	0x20020010	number of system reset
4	AO_1_last_reset	UINT	0	0	sdo	0x20030010	reason of last system res
5	AO_1_uart1_sets	UINT	0	0	sdo	0x20040010	settings MESO_UART
6	AO_1_uart3_sets	UINT	0	0	sdo	0x20050010	settings immodule uart
7	AO_1_internal_temp	REAL	0	0	sdo	0x20060020	temperature internal ser
8	AO_1_v_pwr	REAL	0	0	sdo	0x20070020	PWR voltage
9	AO_1_v_bat	REAL	0	0	sdo	0x20080020	3V battery voltage
10	AO_1_ao_val_0	UINT	1	0	pdor_0x200	0x20090010	AO DAC value
11	AO_1_ao_val_1	UINT	1	0	pdor_0x200	0x200a0010	AO DAC value
12	AO_1_ao_val_2	UINT	1	0	pdor_0x200	0x200b0010	AO DAC value
13	AO_1_ao_val_3	UINT	1	0	pdor_0x200	0x200c0010	AO DAC value
14	AO_1_ao_state_0	USINT	0	0	sdo	0x200d0008	AO state
15	AO_1_ao_state_1	USINT	0	0	sdo	0x200e0008	AO state
16	AO_1_ao_state_2	USINT	0	0	sdo	0x200f0008	AO state
17	AO_1_ao_state_3	USINT	0	0	sdo	0x20100008	AO state
18	AO_1_ao_config_0	USINT	0	0	sdo	0x20110008	AO config
19	AO_1_ao_config_1	USINT	0	0	sdo	0x20120008	AO config
20	AO_1_ao_config_2	USINT	0	0	sdo	0x20130008	AO config
21	AO_1_ao_config_3	USINT	0	0	sdo	0x20140008	AO config
22	AO_1_sys_tick_counter	ULINT	0	0	sdo	0x20150040	tick in ms

Рисунок 22 – Окно модуля расширения/ПЛК по межмодульной шине

Область регистров модулей расширения/ПЛК делятся на 2 типа:

- PDOR – регистры, автоматически опрашиваемые ведущим ПЛК);
- SDO – регистры, опрашиваемые ведущим ПЛК при указании пользователем.

Для работы с переменными SDO необходимо в окне модуля расширения/ПЛК в ячейке «Polling» указать действие, которое необходимо совершить с переменной (**«read»** – чтение, **«write»** – запись). После этого их можно использовать в функциях, ФБ и программах.

6 ОПИСАНИЕ БИБЛИОТЕКИ ФУНКЦИЙ И ФБ

Форма записи языке ST и изображение в FBD встроенных ФБ указано в Приложении Д.

6.1 Стандартные ФБ

SR–триггер Данный ФБ представляет собой бистабильный SR–триггер, с доминирующим входом S (Set). Выход Q1 становится "1", когда вход S1 становится "1". Это состояние сохраняется, даже если S1 возвращается обратно в "0". Выход Q1 возвращается в "0", когда вход R становится "1". Если входы S1 и R находятся в "1" одновременно, доминирующий вход S1 установит выход Q1 в "1". Когда ФБ вызывается первый раз, начальное состояние Q1 это "0".

RS–триггер Данный ФБ представляет собой бистабильный RS–триггер, с доминирующим входом R (Reset). Выход Q1 становится "1", когда вход S становится "1". Это состояние сохраняется, даже если S возвращается обратно в "0". Выход Q1 возвращается в "0", когда вход R1 становится "1". Если входы S и R1 находятся в "1" одновременно, доминирующий вход R1 установит выход Q1 в "0". Когда ФБ вызывается первый раз, начальное состояние Q1 это "0".

SEMA Данный ФБ представляет собой семафор, определяющий механизм, позволяющий элементам программы иметь

взаимоисключающий доступ к определенным ресурсам.

R_TRIG Данный ФБ представляет собой индикатор нарастания фронта, который генерирует на выходе одиночный импульс при нарастании фронта сигнала. Выход Q становится "1", если происходит переход из "0" в "1" на входе CLK. Выход остается в состоянии "1" от одного выполнения блока до следующего (один цикл); затем выход возвращается в "0".

F_TRIG Данный ФБ представляет собой индикатор спада фронта, который генерирует на выходе одиночный импульс при спаде фронта сигнала. Выход Q становится "1", если происходит переход из "1" в "0" на входе CLK. Выход будет оставаться в состоянии "1" от одного выполнения блока до следующего; затем выход возвращается в "0".

CTU Данный ФБ представляет собой инкрементный счётчик. Сигнал "1" на входе R вызывает присваивание значения "0" выходу CV. При каждом переходе из "0" в "1" на входе CU значение CV увеличивается на 1. Когда CV >= PV, выход Q устанавливается в "1".

Примечание: *Счетчик работает только до достижения максимального значения используемого типа данных. Переполнения не происходит. Входы CU, RESET и выход Q типа BOOL, вход PV и выход CV типа WORD. По каждому фронту на входе CU (переход из FALSE в TRUE) выход CV увеличивается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет значения заданного PV. Счетчик CV сбрасывается в 0 по входу RESET = TRUE.*

CTD Данный ФБ представляет собой декрементный счётчик.

Сигнал "1" на входе LD вызывает присваивание значения на входе PV выходу CV. При каждом переходе из "0" в "1" на входе CD значение CV уменьшается на 1.Когда CV <= 0, выход Q принимает значение "1".

Примечание: *Счетчик работает только до достижения минимального значения используемого типа данных. Переполнения не происходит.*

CTUD Данный ФБ представляет собой реверсивный счётчик. Сигнал "1" на входе R вызывает присваивание значения "0" выходу CV. Сигнал "1" на входе LD вызывает присваивание значения на входе PV выходу CV. При каждом переходе из "0" в "1" на входе CU значение CV увеличивается на 1. При каждом переходе из "0" в "1" на входе CD значение CV уменьшается на 1. Если сигнал "1" приходит одновременно на входы R и LD, вход R обрабатывается первым. Когда CV >= PV, выход QU имеет значение "1". Когда CV <= 0, выход QD принимает значение "1".

Примечание: *Вычитающий счетчик работает только до достижения минимального значения используемого типа данных, суммирующий счетчик работает только до достижения максимального значения используемого типа данных. Переполнения не происходит.*

TP Данный ФБ представляет собой повторитель импульсов и используется для генерирования импульса с заданной продолжительностью. Если IN становится "1", Q становится "1", и начинается отсчет внутреннего времени (ET). Если внутреннее время достигает значения PT, Q становится "0" (независимо от IN). Отсчет внутреннего времени останавливается/сбрасывается, если IN становится "0". Если внутреннее время не достигло значения PT, импульс IN не

влияет на внутреннее время. Если внутреннее время достигло значения PT, и IN равен "0", отсчет внутреннего времени останавливается/сбрасывается, и Q становится "0".

TON Данный ФБ представляет собой таймер с задержкой включения. Он запускается, когда состояние сигнала на входе меняется от 0 к 1 и устанавливает на выходе 1 по истечении заданного времени. Если IN становится "1", запускается отсчет внутреннего времени (ET). Если внутреннее время достигает значения PT, Q становится "1". Если IN становится "0", Q становится "0", а подсчет внутреннего времени останавливается/сбрасывается. Если IN становится "0" до того, как внутреннее время достигло значения PT, подсчет внутреннего времени останавливается/сбрасывается, а выход Q не устанавливается в "0".

TOF Данный ФБ представляет собой таймер с задержкой отключения. Он запускается, когда состояние сигнала на входе меняется от 1 к 0 и устанавливает на выходе 0 по истечении заданного времени. Если IN становится "1", Q становится "1". Если IN становится "0", запускается отсчет внутреннего времени (ET). Если внутреннее время достигает значения PT, Q становится "0". Если IN становится "1", Q становится "1", а подсчет внутреннего времени останавливается/сбрасывается. Если IN становится "1" до того, как внутреннее время достигло значения PT, подсчет внутреннего времени останавливается/сбрасывается, а выход Q не устанавливается в "0".

6.2 Дополнительные функциональные блоки

RTC Данный ФБ представляет собой часы реального времени и имеет много вариантов использования, включая добавление временных отметок, для установки даты и времени в формируемых отчетах, в аварийных сообщениях и т.д. Вход PDT (Preset DT) предназначен для установки времени. Часы начинают отсчет времени от значения PDT. Выход Q (BOOL) повторят значение EN. Выход CDT (Current DT) дает текущее значение даты и времени.

INTEGRAL ФБ интеграл интегрирует входное значение XIN по времени.

DERIVATIVE ФБ производная выдаёт значение XOUT пропорционально скорости изменения входного параметра XIN.

PID Данный ФБ представляет собой устройство в цепи обратной связи, используемое в системах автоматического управления для формирования управляющего сигнала. ПИД–регулятор формирует управляющий сигнал, являющийся суммой трёх слагаемых, первое из которых пропорционально входному сигналу, второе – интеграл входного сигнала, третье – производная входного сигнала.

HYSTERESIS ФБ гистерезис предоставляет выходное гистерезисное булевское значение, которое определяется разницей вводных параметров XIN1 и XIN2 (типа REAL с плавающей точкой).

READ_DI	ФБ READ_DI предоставляет данные с дискретных входных каналов ПЛК записанного в одну переменную типа (UDINT), при этом 0 бит соответствует DI_0, а 15 бит – DI_15.
READ_DI_CNT	ФБ READ_DI_CNT предоставляет значение счетчика дискретного входного канала ПЛК номер, которого прописан на входе блока.
READ_DI_FREQ	ФБ READ_DI_FREQ предоставляет значение частоты дискретного входного канала ПЛК, номер которого прописан на входе блока.
READ_AI	ФБ READ_AI указывает значение аналогового входного канала ПЛК номер, которого прописан на входе блока.
READ_DO	ФБ READ_DO предоставляет информацию о состоянии дискретных выходов.
READ_RESET	ФБ READ_RESET предоставляет информацию о количестве перезагрузок с момента обновления ОС ПЛК и причине последней перезагрузки.
READ_PWR	ФБ READ_PWR предоставляет информацию о входном напряжении питания и напряжение батареи ПЛК.
READ_INTERNAL_TEMP	ФБ READ_INTERNAL_TEMP предоставляет информацию температуре микропроцессора ПЛК BRIC.
READ_SYS_TICK_COUNTER	ФБ READ_SYS_TICK_COUNTER предоставляет информацию о времени работы ПЛК с момента последней перезагрузки, указанной в мс.

WRITE_MDB_ADRESS ФБ WRITE_MDB_ADRESS позволяет установить Modbus адрес ПЛК (от 1 до 247).

WRITE_UART_SETS ФБ WRITE_UART_SETS предоставляет изменять параметры UART каналов, записанных в формате U16 на рисунке 23 в двоичной форме указан формат подбора настройки UART канала.

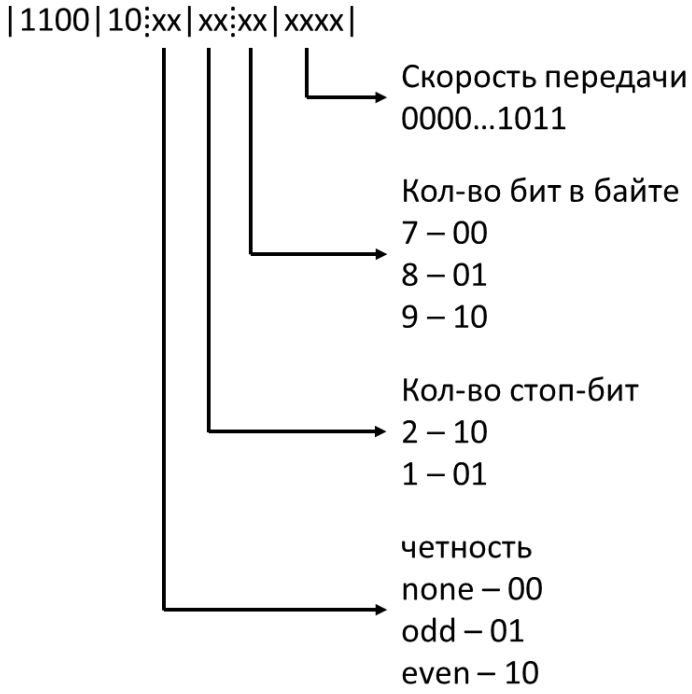


Рисунок 23 – Структура изменения параметров UART каналов

WRITE_CH_TIMEOUT ФБ WRITE_CH_TIMEOUT изменяет время задержки UART каналов, наименование каналов и соответствующие им номера указаны в таблице:

Номер канала	Наименование UART канала
0	MESO_UART_UART
1	RS_485_2_UART
2	RS_232_UART
4	RS_485_1_UART
5	RS_485_IMMO_UART
6	HART_UART

WRITE_DI_NOISE_FLTR_10US ФБ WRITE_DI_NOISE_FLTR_10US для указанного дискретного входа задается период нечувствительности импульса в диапазоне от 0 до 65512, при этом считается в десятках мс.

WRITE_DI_PULSELESS ФБ WRITE_DI_PULSELESS для указанного дискретного входа задает период, за который ведется подсчет импульсов для расчета частоты.

WRITE_DI_MODE ФБ WRITE_DI_MODE для указанного дискретного входа обозначает подключенные опции (0– не подключены, 1– подключен счетчик импульсов, 2– подключен расчет частоты дискретного входа, 3– подключен счетчик импульсов и расчет частоты дискретного входа).

WRITE_DO	ФБ WRITE_DO подает напряжение, согласно маске DO_MASK, на соответствующие дискретные выхода DO_VALUE (выхода прописываются побитово).
WRITE_DO_SC	ФБ WRITE_DO_SC устанавливает номера дискретных выходов, на которых необходимо включить программную защиту от КЗ и дискретные выхода, на которых сработала программная защита от КЗ.
WRITE_DO_PWM_FREQ	ФБ WRITE_DO_PWM устанавливает ШИМ дискретных выходов в диапазоне от 0 до 10000 Гц.
WRITE_DO_PWM_CTRL	ФБ WRITE_DO_PWM_CTRL устанавливает для выбранного дискретного выхода скважность сигнала. Указывается диапазон от 0 до 127, который соответствует диапазону от 0 до 100%.
STRUCT_REAL_TIME	ФБ STRUCT_REAL_TIME позволяет считать время с ПЛК.
UNIX_TIME	ФБ UNIX_TIME устанавливает время на ПЛК (время указывается в мс.).
WRITE_STRUCT_TIME	ФБ WRITE_STRUCT_TIME устанавливает время на ПЛК, выхода функционального блока используются для проверки записанного времени.

6.3 Преобразования типов

Данный набор функций предназначен для всех возможных и корректных, согласно стандарта IEC 61131–3, преобразований между типами данных.

6.4 Числовые операции

ABS Данная функция возвращает в OUT модуль входного числа IN.

SQRT Данная функция возвращает в OUT квадратный корень входного числа IN.

LN Данная функция возвращает в OUT значение натурального логарифма от IN.

LOG Данная функция возвращает в OUT значение логарифма по основанию 10 от IN.

EXP Данная функция возвращает в OUT значение экспоненты, возведённой в степень IN.

SIN Данная функция возвращает в OUT значение синуса IN.

COS Данная функция возвращает в OUT значение косинуса IN.

TAN Данная функция возвращает в OUT значение тангенс IN.

ASIN Данная функция возвращает в OUT значение арксинус IN.

ACOS Данная функция возвращает в OUT значение арккосинус IN.

ATAN Данная функция возвращает в OUT значение арктангенс IN.

6.5 Арифметические операции

ADD Данная функция возвращает в OUT результат сложения IN1 и IN2.

MUL Данная функция возвращает в OUT результат умножения IN1 и IN2.

SUB Данная функция возвращает в OUT результат вычитания из IN1 значения IN2.

DIV Данная функция возвращает в OUT результат деления IN1 на IN2.

MOD Данная функция возвращает в OUT остаток от деления IN1 на IN2.

EXPT Данная функция возвращает в OUT значение IN1 возведённое в степень IN2.

MOVE Данная функция возвращает в OUT значение IN.

6.6 Операции смещения бит

SHL Данная функция возвращает в OUT арифметический сдвиг аргумента IN на N бит влево с заполнением битов справа нулями.

SHR Данная функция возвращает в OUT арифметический сдвиг аргумента IN на N бит вправо с заполнением битов слева нулями.

ROR Данная функция возвращает в OUT циклический сдвиг аргумента IN на N бит влево.

ROL Данная функция возвращает в OUT циклический сдвиг аргумента IN на N бит вправо.

6.7 Побитовые операции

AND Данный ФБ представляет собой организацию «логического И» для всех входных аргументов $IN_1 \ldots IN_n$.

OR Данная функция представляет собой организацию «логического ИЛИ» для всех входных аргументов $IN_1 \ldots IN_n$.

XOR Данная функция представляет собой организацию «логического исключающего ИЛИ» для всех входных аргументов $IN_1 \ldots IN_n$.

NOT Данная функция представляет собой организацию «логической инверсии» для входного аргумента IN.

6.8 Операции выбора

SEL Данная функция возвращает в OUT один из двух аргументов IN1 или IN2 в зависимости от значения аргумента G. Если G = 0, то OUT равно X1, иначе – OUT равно X2.

MAX Данная функция возвращает в OUT максимум из входных аргументов IN1 и IN2.

MIN Данная функция возвращает в OUT минимум из входных аргументов IN1 и IN2.

LIMIT Данная функция возвращает в OUT значение входного аргумента IN, в случае превышения им значения MX – в OUT возвращается MX, в случае если IN меньше MN – в OUT возвращается MN.

MUX Данная функция возвращает в OUT значение на входе IN(K), в зависимости от входного K. Количество входов $IN_{(n)}$ изменяемое – от 2 до 20. По умолчанию 2.

6.9 Операции сравнения

GT Данная функция сравнивает все входные аргументы и выдаёт на выходе OUT значение True, если выполнится следующее условие: (IN1 > IN2) & (IN2 > IN3) & … (INn–1 > INn), в противном случае в OUT выдаётся False. Количество входов IN(n) изменяемое – от 2 до 20. По умолчанию 2.

GE Данная функция сравнивает все входные аргументы и выдаёт на выходе OUT значение True, если выполнится следующее условие: (IN1 >= IN2) & (IN2 >= IN3) &… (IN_{n-1} >= IN_n), в противном случае в OUT выдаётся False. Количество входов IN(n) изменяемое – от 2 до 20. По умолчанию 2

EQ Данная функция сравнивает все входные аргументы и выдаёт на выходе OUT значение True, если выполнится следующее условие: (IN1 = IN2) & (IN2 = IN3) & … (IN_{n-1} = IN_n), в противном случае в OUT выдаётся False. Количество входов IN(n) изменяемое – от 2 до 20. По умолчанию 2.

LT Данная функция сравнивает все входные аргументы и выдаёт на выходе OUT значение True, если выполнится следующее условие: (IN1 < IN2) & (IN2 < IN3) & … (IN_{n-1} < IN_n), в противном случае в OUT выдаётся False. Количество входов IN(n) изменяемое – от 2 до 20. По умолчанию 2.

LE Данная функция сравнивает все входные аргументы и выдаёт на выходе OUT значение True, если выполнится следующее условие: (IN1 <= IN2) & (IN2 <= IN3) & … (IN_{n-1} <= IN_n), в противном случае в OUT выдаётся False. Количество входов IN(n) изменяемое – от 2 до 20. По умолчанию 2.

NE Данная функция сравнивает все входные аргументы и выдаёт на выходе OUT значение True, если выполнится следующее условие: (IN1 <> IN2) & (IN2 <> IN3) &… (IN_{n-1} <> IN_n), в противном случае в OUT выдаётся False. Количество входов $IN_{(n)}$ изменяемое – от 2 до 20. По умолчанию 2.

7 ОПИСАНИЕ РАБОТЫ С ЯЗЫКАМИ МЭК 61131-3

МЭК 61131-3 – раздел международного стандарта МЭК 61131, описывающий языки программирования для программируемых логических контроллеров.

Стандарт устанавливает пять языков программирования со следующими названиями:

- Структурированный текст (ST – Structured Text);
- Последовательные функциональные схемы (SFC – "Sequential Function Chart");
- Диаграммы функциональных блоков (FBD – Function Block Diagram);
- Релейно-контактные схемы, или релейные диаграммы (LD – Ladder Diagram);
- Список инструкций (IL – Instruction List).

Графическими языками являются SFC, FBD, LD. Языки IL и ST являются текстовыми.

7.1 Описание работ с текстовыми редакторами языков ST и IL

Текстовый редактор языков ST и IL (см. рисунок 24) позволяет создавать и редактировать алгоритмы и логику выполнения программных модулей на языках ST и IL.

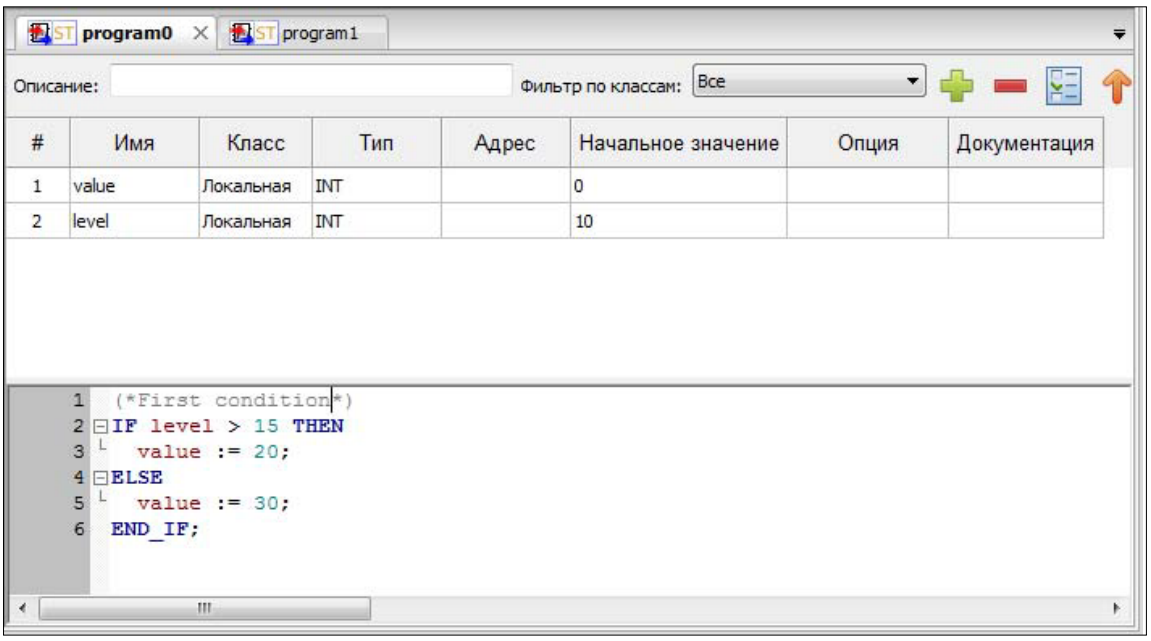


Рисунок 24 – Редактор языков ST и IL

Он обеспечивают следующие возможности:

- Подсветку синтаксиса кода, написанного пользователем, т.е. выделения особыми параметрами шрифта ключевых слов данных языков;
- Нумерации строк, что может быть полезным при возникновении ошибок в программе, т.к. транслятор кода ST в C выдаёт номер строки, в которой найдена ошибка;
- Сворачивание кода структурных элементов языка: определения функции, определение типа и т.д.

Увеличение или уменьшение размера шрифт выполняется с помощью Ctrl + <колёсико мыши>.

7.1.1 Конструкции языка ST

К конструкциям языка ST относятся:

1. Арифметические операции;
2. Логические (побитовые) операции;
3. Операции сравнения;
4. Операция присвоения;
5. Конструкция IF – ELSE;
6. Цикл FOR;
7. Цикл WHILE;
8. Конструкция CASE.

1. Арифметические операции.

К арифметическим операциям относятся:

«+» – Сложение;

«–» – Вычитание;

«*» – Умножение;

«/» – Деление;

«MOD» – Остаток от целочисленного деления.

2. Логические (побитовые) операции.

К данным операциям относятся:

«OR» – Логическое (побитовое) сложение;

«AND» – Логическое (побитовое) умножение;

«XOR» – Логическое (побитовое) «исключающее ИЛИ»;

«NOT» – Логическое (побитовое) отрицание.

3. Операции сравнения

Поддерживаются следующие операции сравнения:

«=» – Сравнение на «равенство»;

«<>» – Сравнение на «неравенство»;

«>» – Сравнение на «больше»;

«>=» – Сравнение на «не меньше»;

«<» – Сравнение на «меньше»;

«<=» – Сравнение на «не больше».

В качестве результата сравнения всегда используется значение типа BOOL.

4. Операция присвоения.

Для обозначения присвоения используется парный знак «:=». В правой и левой части выражения должны быть операнды одного типа (автоматического приведения типов не предусмотрено). В левой части выражения (принимающая сторона) может быть использована только переменная. Правая часть может содержать выражение или константу.

5. Конструкция *IF – ELSE*.

Конструкция IF–ELSE имеет следующий формат:

IF<логическое выражение>THEN<выполняемое выражение>
[ELSE<выполняемое выражение>]
END_IF;

Например:

```
IF Vary<> 0
THEN Var4 := 1
ELSE Var4 := 10;
END_IF;
```

Конструкция допускает вложенность, т.е. внутри одного IF может быть еще один и т.д.

Например:

```
IF Var5 > 10 THEN
IF Var5 < Var2 + 1
THEN Var5 := 10;
ELSE Var5 := 0;
END_IF;
END_IF;
```

6. Цикл *FOR*

Служит для задания цикла с фиксированным количеством итераций.

Формат конструкции следующий:

FOR<переменная управления> := <выражение1>[5] TO <выражение2>
[BY< выражение3>] DO
<выполняемое выражение>
END_FOR;

При задании условий цикла считается, что <переменная управления>, <выражение1> …<выражение3> имеют тип INT. Выход из цикла будет произведен в том случае, если значение переменной цикла превысит значение <выражение3>.

Например:

```
FOR i := 1 TO 10 BY 2 DO
k := k * 2;
END_FOR;
```

[5] Выражения <выражение1> … <выражение1> вычисляются до входа в цикл, поэтому изменения значений переменных, входящих в любое из этих выражений, не приведет к изменению числа итераций.

Оператор BY задает приращение переменной цикла (в данном случае i будет увеличиваться на 2 при каждом проходе по циклу). Если оператор BY не указан, то приращение равно 1.

Например:

```
FOR i := 1 TO k / 2 DO
var3 := var3 + k;
k := k – 1;
END_FOR;
```

Внутри цикла могут использоваться другие циклы, операторы IF и CASE. Для выхода из цикла (любого типа) может использоваться оператор EXIT.

Например:

```
FOR i := 1 TO 10 BY 2 DO
k := k * 2;
IF k > 20 THEN
EXIT;
END_IF;
END_FOR;
```

Например:

```
k := 10;
FOR i := 1 TO k / 2 DO
k := 20;
END_FOR;
```

На строках между DO и END_FOR производится изменение переменной k, при этом эти строки выполнятся пять раз.

Например:

```
FOR i := 1 TO 5 DO
i := 55;
END_FOR;
```

При первом проходе значение i будет равно 1, потом в теле цикла изменится на 55, но на втором проходе значение i станет равно 2 – следующему значению по условиям цикла.

7. Цикл *WHILE*

Служит для определения цикла с предусловием. Цикл будет исполняться до тех пор, пока выражение в предложении WHILE возвращает TRUE.

Формат конструкции следующий:

WHILE<Boolean–Expression>DO<выполняемое выражение>

END_WHILE;

Значение <Boolean–Expression> проверяется на каждой итерации. Завершение цикла произойдет, если выражение <Boolean–Expression> вернет FALSE.

Например:

```
k := 10;
WHILE k > 0 DO
i := i + k;
k := k – 1;
END_WHILE;
```

Внутри цикла могут использоваться другие циклы, операторы IF и CASE. Для досрочного завершения цикла используется оператор EXIT.

8. Конструкция *CASE*

Данная конструкция служит для организации выбора из диапазона значений.

Формат конструкции следующий:

CASE<выражение>OF

CASE_ELEMENT {CASE_ELEMENT}

[ELSE<выполняемое выражение>]

END_CASE;

CASE_ELEMENT – это список значений, перечисленных через запятую. Элементом списка может быть целое число или диапазон целых чисел. Диапазон задается следующим образом BEGIN_VAL.. END_VAL.

Если текущее значение <выражение> не попало ни в один CASE_ELEMENT, то управление будет передано на предложение ELSE. Если предложение ELSE не указано, то никаких действий выполнено не будет.

Значение <выражение> может быть только целым.

Например:

```
CASE k OF
1:
k := k * 10;
2..5:
k := k * 5;
i := 0;
6, 9..20:
k := k - 1;
ELSE
k := 0;
i := 1;
END_CASE;
```

При задании списка значений необходимо выполнять следующие условия:

– наборы значений внутри одного CASE не должны пересекаться;

– при указании диапазона значений начало диапазона должно быть меньше его конца.

Действия, предусмотренные для обработки каждого из случаев CASE, могут использовать циклы, операторы IF и CASE.

7.1.2 Конструкции языка IL

Основа языка программирования IL, как и в случае Assembler, это переходы по меткам и аккумулятор. В аккумулятор загружается значения переменной, а дальнейшее выполнение алгоритма представляет собой извлечение значения из аккумулятора и совершение над ним операций. В таблице 1 приведены операторы языка IL. Пример программы, выполненной на IL приведен на рисунке 25.

Таблица 1 – Операторы языка IL

Оператор	Описание
LD	Загрузить значение операнда в аккумулятор
LDN	Загрузить обратное значение операнда в аккумулятор
ST	Присвоить значение аккумулятора операнду
STN	Присвоить обратное значение аккумулятора операнду
S	Если значение аккумулятора TRUE, установить логический операнд
R	Если значение аккумулятора FALSE, сбросить логический операнд
AND	«Поразрядное И» аккумулятора и операнда
ANDN	«Поразрядное И» аккумулятора и обратного операнда
OR	«Поразрядное ИЛИ» аккумулятора и операнда
ORN	«Поразрядное ИЛИ» аккумулятора и обратного операнда
XOR	«Поразрядное разделительное ИЛИ» аккумулятора и операнда
XORN	«Поразрядное разделительное ИЛИ» аккумулятора и обратного операнда
ADD	Сложение аккумулятора и операнда, результат записывается в аккумулятор

SUB	Вычитание операнда из аккумулятора, результат записывается в аккумулятор
MUL	Умножение аккумулятора на операнд, результат записывается в аккумулятор
DIV	Деление аккумулятора на операнд, результат записывается в аккумулятор
GT	Значение аккумулятора сравнивается со значением операнда(>(greater than)). Значение (TRUE или FALSE) записывается в аккумулятор
GE	Значение аккумулятора сравнивается со значением операнда (>=greater than or equal). Значение (TRUE или FALSE) записывается в аккумулятор
EQ	Значение аккумулятора сравнивается со значением операнда (=(equal)). Значение (TRUE или FALSE) записывается в аккумулятор
NE	Значение аккумулятора сравнивается со значением операнда <>(not equal). Значение (TRUE или FALSE) записывается в аккумулятор
LE	Значение аккумулятора сравнивается со значением операнда (<=(less than or equal to)). Значение (TRUE или FALSE) записывается в аккумулятор
LT	Значение аккумулятора сравнивается со значением операнда (<(less than)). Значение (TRUE или FALSE) записывается в аккумулятор
JMP	Переход к метке
JMPC	Переход к метке при условии, что значение аккумулятора TRUE
JMPCN	Переход к метке при условии, что значение аккумулятора FALSE

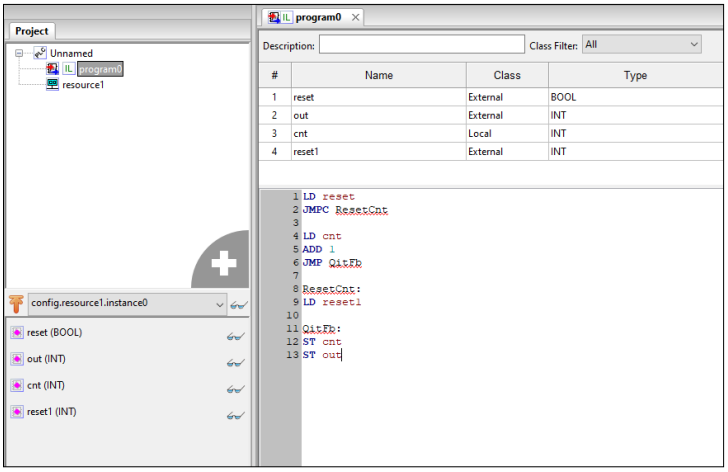


Рисунок 25 – Пример программы на языке IL

7.2 Описание работ с текстовыми редакторами языков ST и IL

Данные редакторы позволяют создавать и редактировать алгоритмы и логику выполнения программных модулей, написанных на языках FBD, SFC и LD.

7.2.1 Конструкции языка FBD

Основными элементами языка FBD являются: переменные, функциональные блоки и соединения. При редактировании FBD диаграммы, в панели инструментов появляется следующая панель (см. рисунок 26).

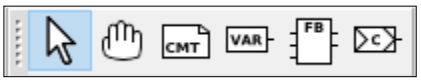


Рисунок 26 – Панель редактирования FBD диаграмм

С помощью данной панели можно добавить все элементы языка FBD (назначение каждой кнопки описано в таблице 2).

Таблица 2 – Кнопки панели редактирования FBD диаграммы

Внешний вид кнопки	Наименование кнопки	Функции кнопки
	Выделение объектов на диаграмме	Перевод указателя мыши в состояние, при котором можно осуществлять выделение объектов в редакторе
	Перемещение диаграммы	Перевод указателя мыши в состояние, при котором можно изменять размеры редактора
	Создать новый комментарий	Вызов диалога создания комментария
	Добавить переменную	Вызов диалога добавления переменной

	Добавить ФБ	Вызов диалога добавления функционального блока
	Добавить соединение	Вызов диалога добавления соединения

Для этого необходимо указателем мыши выбрать необходимую кнопку и нажать на свободное место в области редактирования FBD диаграммы. В зависимости от выбранного элемента появятся определённые диалоги добавления данного элемента.

Аналогичные действия можно выполнить с помощью всплывающего меню в области редактирования FBD диаграмм. Вызов данного меню происходит при помощи нажатия правой клавишей мыши и выбора пункта «Add» (Добавить), в котором будет: «Block» (Блок), «Variable» (Переменная), «Connection» (Соединение), «Comment» (Комментарий) (см. рисунок 27).

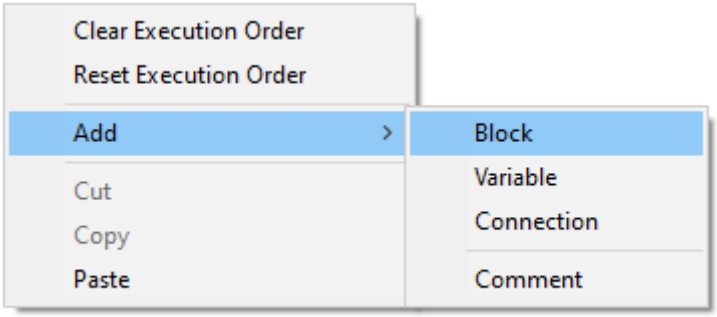


Рисунок 27 – Всплывающее меню редактора языка FBD

При добавлении функционального блока одним из описанных выше способов, появится диалог «Block Properties» (Свойства блока) (см. рисунок 28).

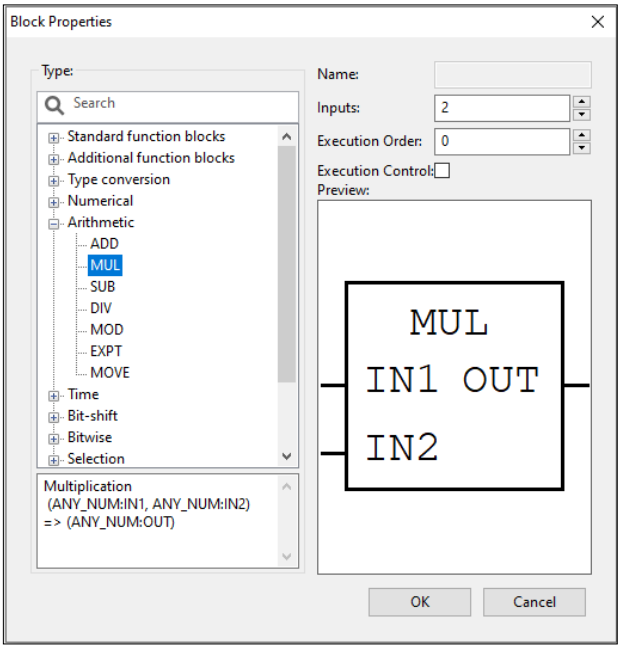


Рисунок 28 – Свойства функционального блока

В данном диалоге приведено краткое описание функционального блока в нижнем левом окне и предоставлена возможность задать некоторые свойства (имя, количество входов, порядок выполнения).

Добавление блока происходит путем перетаскивания необходимой функции из панели библиотеки функций и функциональных блоков, через окно «Block Properties» или путем копирования существующего блока.

Переменные добавляются из панели переменных и констант с помощью перетаскивания левой клавишей мыши за область (см. на рисунок 29) в область редактирования FBD диаграмм или через диалог «Variable Properties» (Свойства переменной) (см. на рисунок 30), вызванный через всплывающего меню редактора языка FBD.

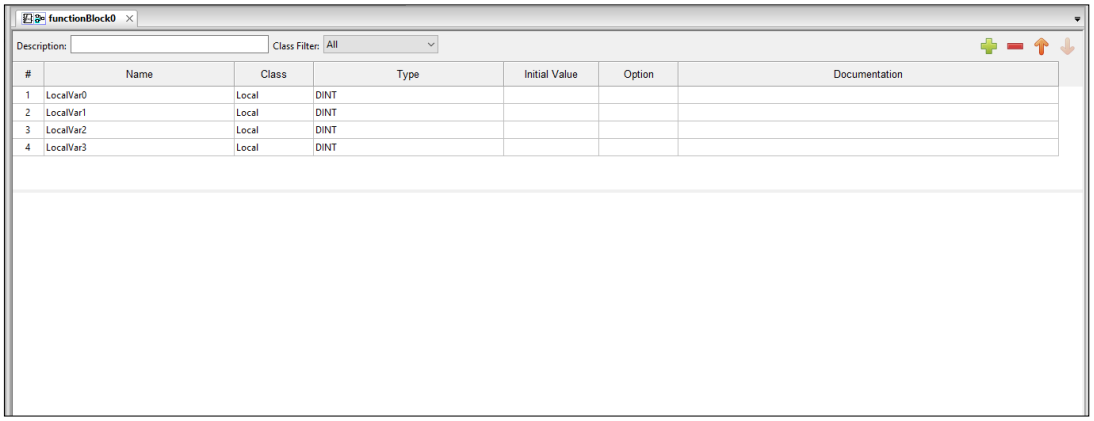


Рисунок 29 – Добавление переменной из панели переменных и констант

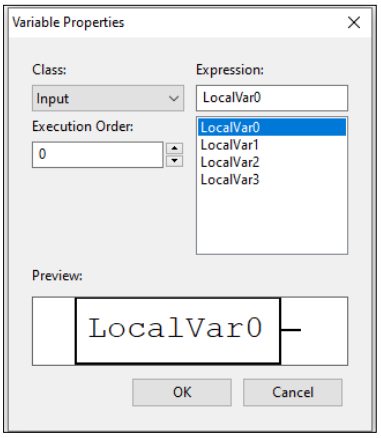


Рисунок 30 – Свойства переменной

В данном диалоге можно задать порядок выполнения переменной и изменить её класс («Input» (Входная), «Output» (Выходная), «InOut» (Входная/Выходная)).

Когда необходимо передать выходное значение одного функционального блока на один из входов другого для удобства можно использовать элемент «Connection» (Соединение). На схемах с большим количеством функциональных блоков элемент «Connection» позволяет избежать пересечения прямых соединений, которые приводит к тому, что схема становится менее понятной.

После выбора добавления элемента «Connection» появится диалог «Connection Properties» (Свойства соединения) (см. рисунок 31).

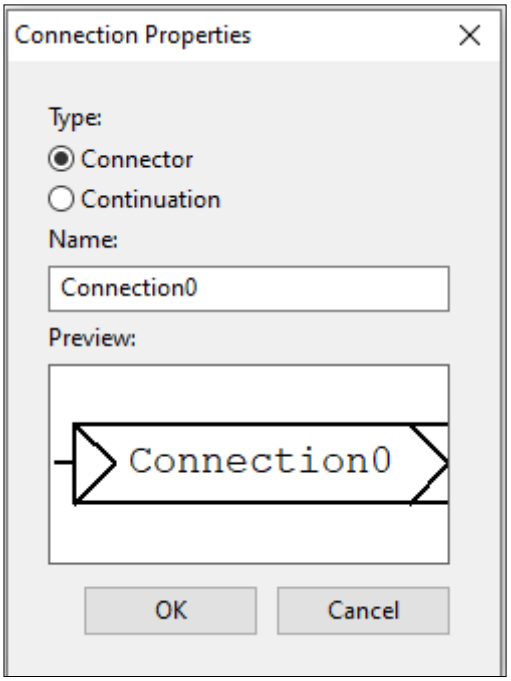


Рисунок 31 – Диалог добавления соединения для FBD

В данном диалоге можно выбрать тип соединения: «Connector» (Выходное соединение) – для выходного значения, «Continuation» (Входное соединение) – для входного значения, а также необходимо указать имя данного соединения. На рисунке 32 представлен пример использования соединений.

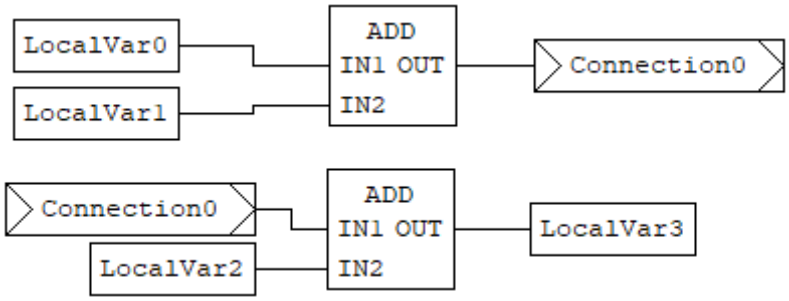


Рисунок 32 – Пример FBD диаграммы с использованием соединений

Редактор FBD диаграмм позволяют добавлять комментарии на диаграмму. После выбора на панели редактирования комментария и добавления его в область редактирования появится диалог (см. рисунок 33) для ввода текста комментария.

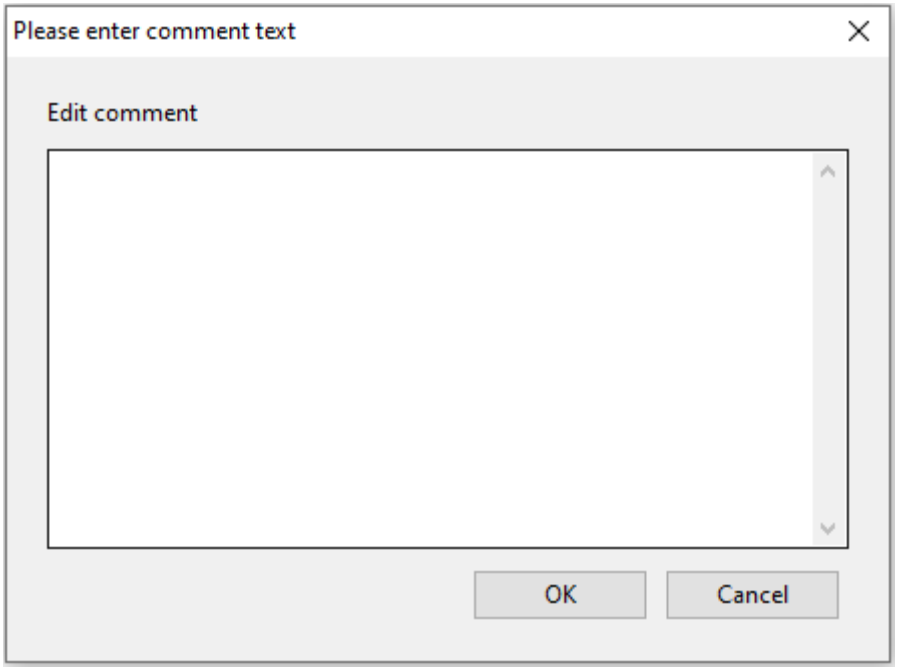


Рисунок 33 – Диалог добавления комментария

Последовательность исполнения функций и функциональных блоков определяется порядком их выполнения. Автоматически он регламентируется следующим образом: чем выше и левее расположен верхний левый угол, описывающего функцию или ФБ прямоугольника, тем раньше данная функция или функциональный будет выполнен. Порядок выполнения может быть изменён вручную с помощью диалога свойств опцией «Execution Order» (Порядок выполнения).

7.2.2 Конструкции языка LD

Язык LD представляет собой графическую форму записи логических выражений в виде контактов и катушек реле.

Как только вкладка с редактированием LD диаграммы становится активной, в панели инструментов появляется панель (см. рисунок 34) с элементами языка LD.

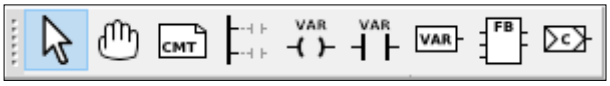


Рисунок 34 – Панель редактирования LD диаграмм

С помощью данной панели можно добавить все элементы языка LD. В таблице 3 приведено описание кнопок данной панели.

Таблица 3 – Кнопки панели редактирования LD диаграммы

Внешний вид кнопки	Наименование кнопки	Функции кнопки
	Выделение объектов на диаграмме	Перевод указателя мыши в состояние, при котором можно осуществлять выделение объектов редакторе одного из графических языков
	Перемещение диаграммы	Перевод указателя мыши в состояние, при котором можно изменять размеры редактора одного из графических языков, с помощью его перемещения
	Создать новую шину питания	Вызов диалога создания новой шины питания
	Создать новую катушку	Вызов диалога создания новой катушки[6]
	Создать новый контакт	Вызов диалога создания нового контакта[7]
	Создать новый комментарий	Вызов диалога создания комментария
	Добавить переменную	Вызов диалога добавления переменной
	Добавить ФБ	Вызов диалога добавления функционального блока
	Добавить соединение	Вызов диалога добавления соединения

[6] Устанавливает соответствующий битовый объект в значение, равное результату, полученному в проверочной зоне (имеется инверсный элемент).

[7] Контакт замкнут, когда битовая переменная, которая управляет им, равна 1 (имеется инверсный элемент)

При добавлении шины питания появится диалог «Power Rail Properties» (Свойства шины питания) (см. рисунок 35).

В данном диалоге указываются следующие свойства:

- «Type» (тип шины питания) («Left PowerRail» (шина питания слева), «Right PowerRail» (шина питания справа));
- «Pin number» (количество контактов на добавляемой шине питания).

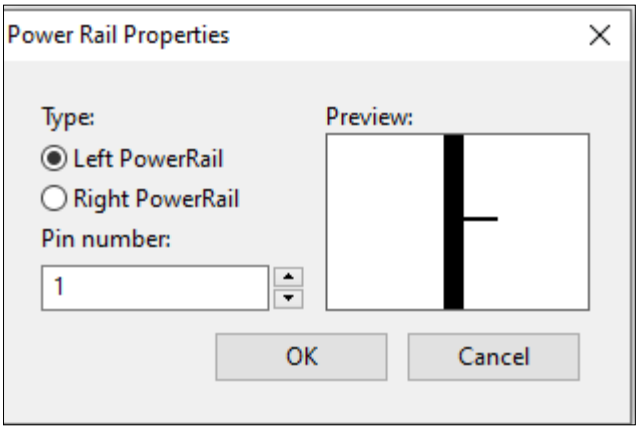


Рисунок 35 – Свойство шины питания

При добавлении контакта на LD диаграмму появится диалог «Edit Contact Values» (Редактирование значения контакта) (см. рисунок 36).

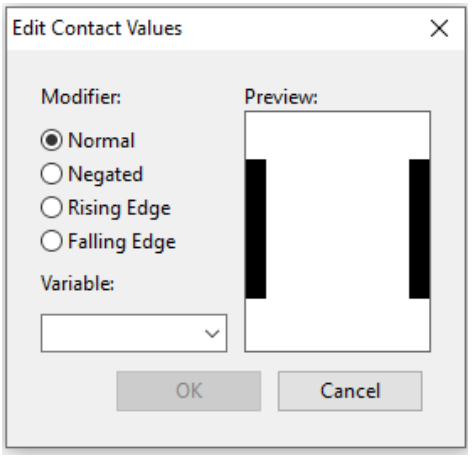


Рисунок 36 – Редактирование контакта

Данный диалог позволяет определить модификатор данного контакта:

- «Normal» (Нормальный);
- «Negated» (Инверсный);
- «Rising Edge» (Нарастание фронта);
- «Falling Edge» (Спад фронта).

Диалог позволяет выбрать из списка переменную, к которой он связан. Переменные должны быть определены в панели переменных и констант и иметь тип переменной BOOL.

Еще одним способом добавления контакта на диаграмму перетаскивание из панели переменных и констант переменной типа BOOL и класса: «Входная», «Входная/Выходная», «Внешняя», «Локальная», «Временная». Для этого необходимо зажать левой кнопкой мыши за первый столбец (который имеет заголовок #) переменную, удовлетворяющую описанным выше критериям и перенести в область редактирования диаграммы.

При добавлении катушки на LD диаграмму появится диалог «Edit Coil Values» (Редактирование значения катушки) (см. рисунок 37).

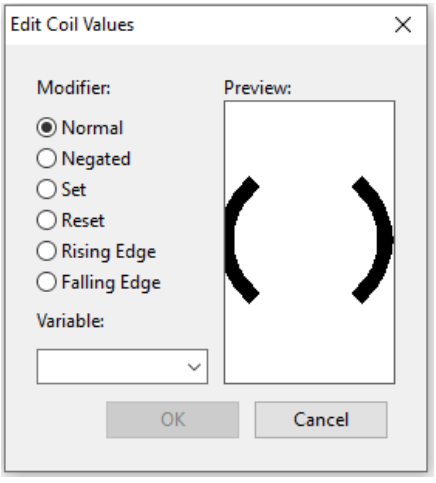


Рисунок 37 – Редактирование катушки

В данном диалоге можно определить модификатор данного контакта:

- «Normal» (Нормальный);
- «Negated» (Инверсный);
- «Set» (Установка);
- «Reset» (Сброс);
- «Rising Edge» (Нарастание фронта);
- «Falling Edge» (Спад фронта).

7.2.3 Конструкции языка SFC

Основными элементами языка SFC являются: начальный шаг, шаг, переход, блок действий, дивергенции, «прыжок». Программа на языке SFC состоит из набора шагов, связанных переходами.

Как только вкладка с редактированием SFC диаграммы становится активной, в панели инструментов появляется следующая панель (см. рисунок 38). В таблице 4 приведено описание кнопок данной панели.

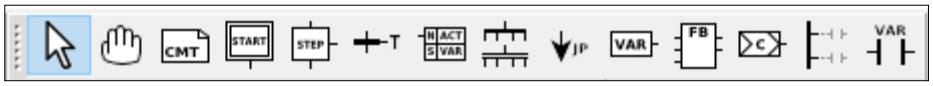


Рисунок 38 – Панель редактирования SFC диаграмм

Таблица 4 – Кнопки панели редактирования SFC диаграммы

Внешний вид кнопки	Наименование кнопки	Функции кнопки
	Выделение объектов на диаграмме	Перевод указателя мыши в состояние, при котором можно осуществлять выделение объектов редакторе одного из графических языков
	Перемещение диаграммы	Перевод указателя мыши в состояние, при котором можно изменять размеры редактора одного из графических языков,

		с помощью его перемещения
	Создать новый комментарий	Вызов диалога создания комментария
	Создать новый начальный шаг	Вызов диалога редактирования шага
	Создать новый шаг	Вызов диалога редактирования шага
	Создать новый переход	Вызов диалога редактирования перехода
	Создать новый блок действий	Вызов диалога редактирования блока действий
	Создать новую дивергенцию	Вызов диалога создания новой дивергенции и конвергенции
	Создать новый «прыжок»	Вызов диалога создания«прыжка»
	Добавить переменную	Вызов диалога добавления переменной
	Добавить ФБ	Вызов диалога добавления функционального блока
	Добавить соединение	Вызов диалога добавления соединения
	Создать новую шину питания	Вызов диалога создания новой шины питания
	Создать новый контакт	Вызов диалога создания нового контакта

Процедура добавления шага инициализации и обычного шага ничем не отличается. В обоих случаях вызывается диалог «Edit Step» (Редактировать шаг) (см. рисунок 39).

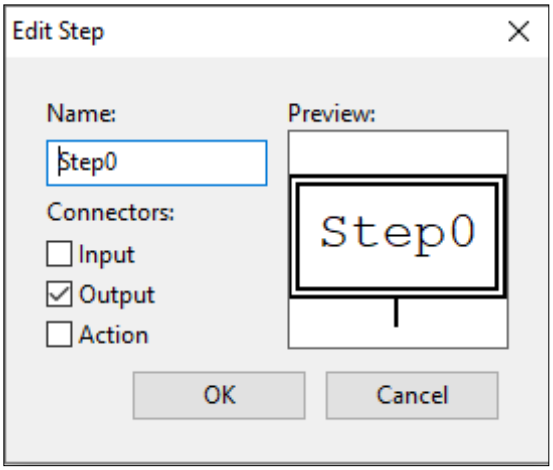


Рисунок 39 – Диалоги редактирования шага инициализации и обычного шага SFC диаграммы

Согласно стандарта IEC 61131-3, на SFC диаграмме должен быть один шаг инициализации, который характеризует начальное состояние SFC-диаграммы и отображается со сдвоенными линиями на границах.

При добавлении на SFC диаграмму перехода, появится диалог «Edit transition» (Редактировать переход) (см. рисунок 40).

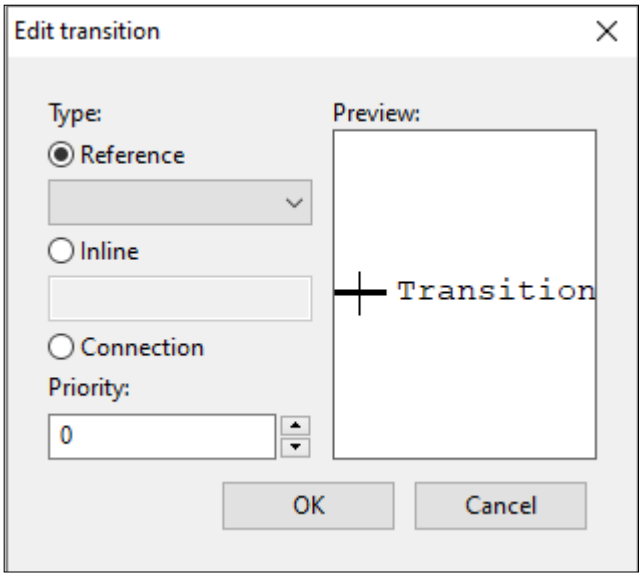


Рисунок 40 – Диалог редактирования перехода для SFC диаграммы

В данном диалоге необходимо выбрать тип перехода и его приоритет. Тип перехода может быть:

- «Reference» (Ссылка);
- «Inline» (Встроенный код);
- «Connection» (Соединение).

При выборе типа перехода «Ссылка» в открывающемся списке будут доступны переходы, предопределённые в дереве проекта для данного программного модуля, написанного на языке SFC. Добавление предопределённого перехода описывается ниже после описания всех добавляемых элементов языка SFC.

При выборе типа перехода «Inline» (см. рисунок 41), условие перехода можно написать в виде выражения на языке ST.

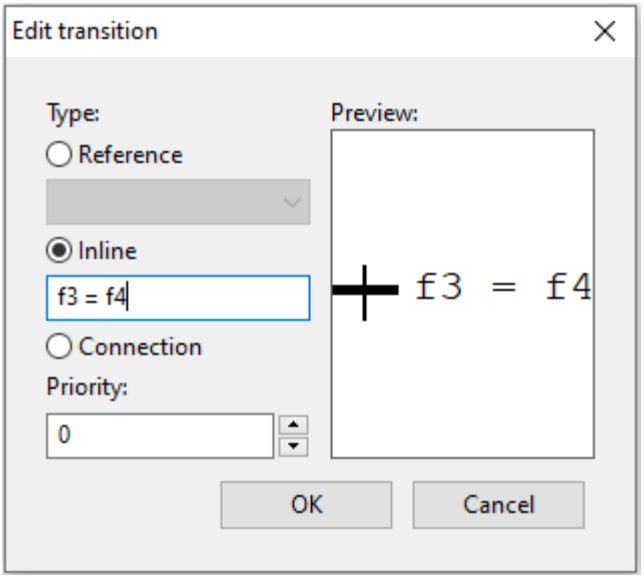


Рисунок 41 – Условие перехода в виде встроенного кода, написанного на языке ST

Реализация перехода таким способом удобна в случае, когда необходимо короткое условие, как например переменные «f3» и «f4» типа INT равны. Встроенный код для такого условия выглядит следующим образом:

```
f3 = f4
```

Так же, например, можно в качестве условия просто указать переменную. В случае её значения равного 0 – будет означать FALSE, все остальные значения – TRUE.

При выборе типа перехода «Connection», в качестве условия перехода можно использовать выходные значения элементов языка FBD или LD.

При выборе типа перехода «Connection», у добавленного перехода появится слева контакт, который необходимо соединить с выходным значением, например, функционального блока языка FBD или катушки LD диаграммы. Стоит отметить, что данное выходное значение должно быть типа BOOL.

При добавлении блока действий на диаграмму появится диалог «Edit action block properties» (Редактировать свойство блока действий) (см. рисунок 42).

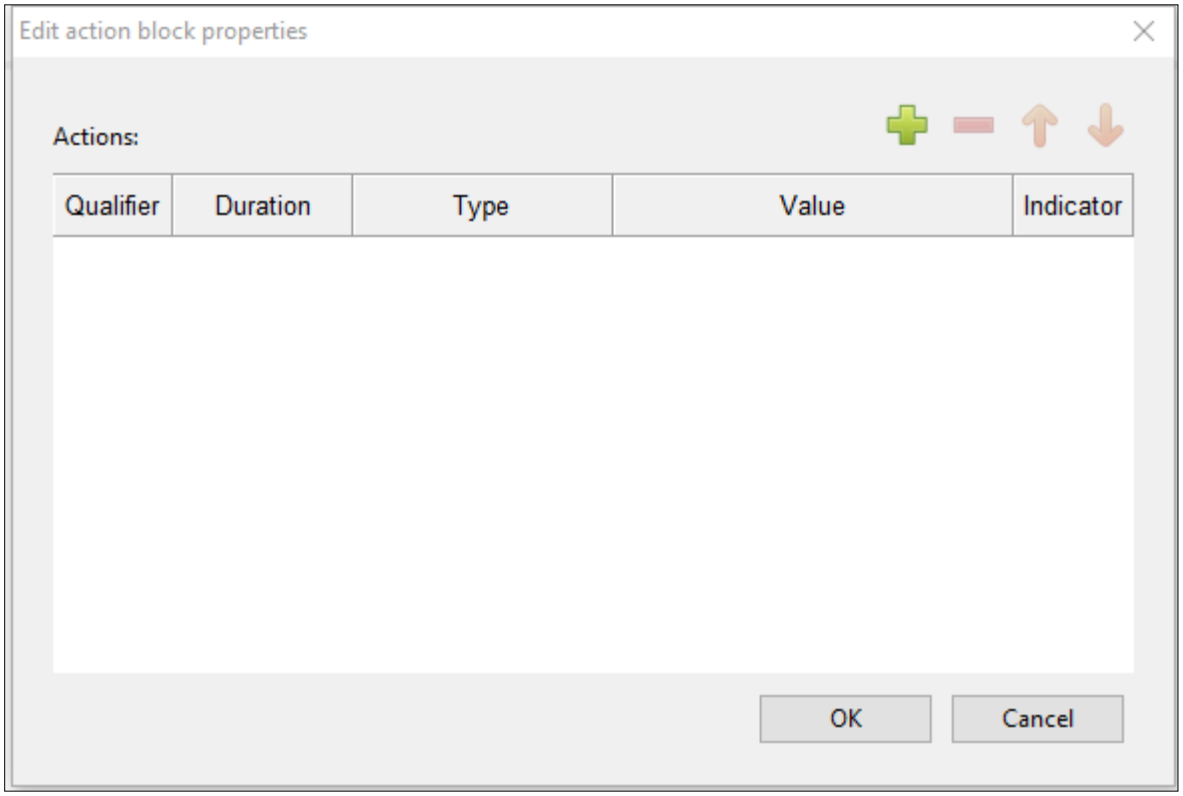


Рисунок 42 – Диалог «Редактировать свойство блока действий»

Данный блок действий может содержать набор действий. Добавить новое действие можно нажав кнопку «Добавить» и установив необходимые параметры:

- «Qualifier» (Квалификатор);
- «Duration» (Продолжительность);
- «Type» (Тип): «Action» (Действие), «Variable» (Переменная), «Inline» (Встроенный код);
- «Value» (Значение);
- «Indicator» (Индикатор).

Поле «Qualifier» определяет момент времени, когда действие начинается, сколько времени продолжается и когда заканчивается. Выбрать квалификатор можно из списка.

Подробное описание квалификаторов, которые выбираются из предлагаемого списка при добавлении действия приведено в таблице 5.

Таблица 5 – Квалификаторы действий SFC диаграммы

Имя квалификатора действия	Поведение блока
D	Действие начинает выполняться через некоторое заданное время (если шаг еще активен) и выполняется до тех пор, пока данный шаг активен
L	Действие выполняется в течение некоторого заданного интервала времени, после чего выполнение действия останавливается
N	Действие выполняется, пока данный шаг активен
P	Действие выполняется один раз, как только шаг стал активен
S	Действие активируется и остается активным пока SFC диаграмма выполняется
R	Действие выполняется, когда диаграмма деактивизируется
DS	Действие начинается выполняться через некоторое заданное время, только в том случае если шаг еще активен
SL	Действие активно в течении некоторого, заданного интервала
SD	Действие начинается выполняться через некоторое время, даже в том случае если шаг уже не активен

Поле «Duration» необходимо для установки интервала времени необходимого для некоторых квалификаторов, описанных выше в таблице 5.

«Type» определяет код или конкретную манипуляцию, которая будет выполняться во время активации действия. В случае выбора «Действия» появляется возможность, как и в случае с переходом, использовать предопределённые действия в дереве проекта для данного программного модуля, написанного на языке SFC.

Элемент «Jump» (прыжок) на SFC диаграмме подобен выполнению оператора GOTO при переходе на определённую метку в коде в различных языках программирования. После выбора добавления «прыжка» на SFC диаграмму, появится диалог (см. рисунок 43), в котором необходимо выбрать из списка шаг, к которому будет происходить «прыжок» – переход от одного шага SFC диаграммы к другому.

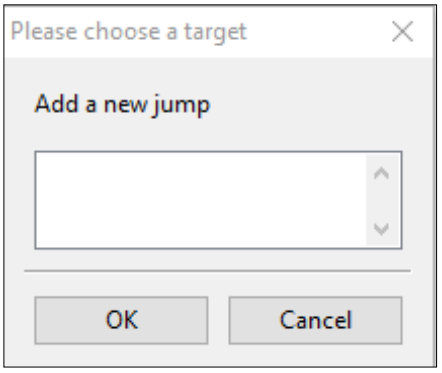


Рисунок 43 – Диалог добавления «прыжка»

В данном диалоге также присутствует и шаг инициализации (начальный шаг). После выбора шага и нажатия кнопки OK. На SFC диаграмме появится стрелочка, которую нужно соединить с переходом. Справа от стрелочки находится имя шага, к которому осуществляется переход в случае выполнения условия перехода, находящегося выше и соединённого с ней.

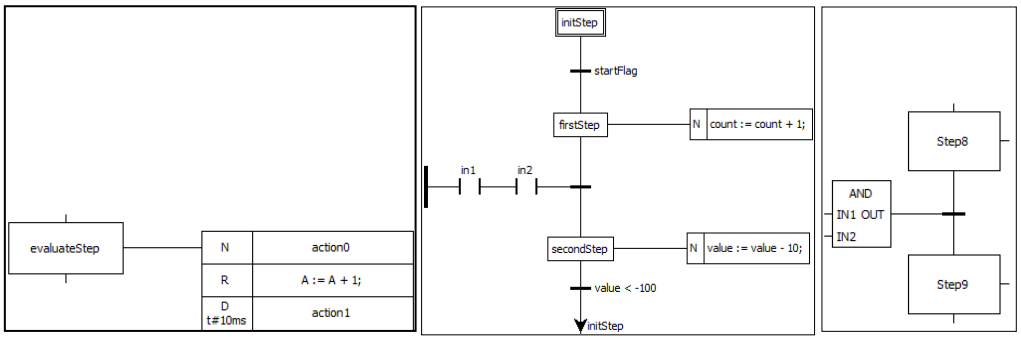


Рисунок 44 – Примеры алгоритмов, написанных на SFC

ПРИЛОЖЕНИЕ А. ДЕТАЛЬНОЕ ОПИСАНИЕ ПАНЕЛЕЙ ГЛАВНОГО ОКНА

Рисунок А.1 – Панель инструментов

Таблица А.1 –Кнопки панели инструментов

	Новый проект	Создать новый проект
	Открыть проект	Открыть существующий проект
	Сохранить проект	Сохранить текущий проект
	Сохранить проект как	Сохранить текущий проект в определённую папку
	Печать	Печать на принтере текущей программы
	Проверка версии	Проверка версии
	Отменить	Отмена последнего действия в среде разработки
	Повторить	Повтор отменённого действия в среде разработки
	Вырезать	Удалить в буфер обмена выделенные элементы в редакторе
	Копировать	Копировать в буфер обмена выделенные элементы в редакторе
	Вставить	Вставить из буфера обмена находящиеся там элементы в редактор
	Поиск в проекте	Вызов диалога поиска данных в проекте
	Развернуть/свернуть окно	Развернуть/свернуть окно на полный экран

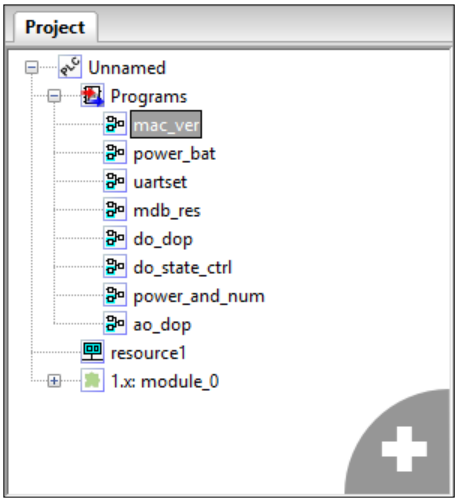
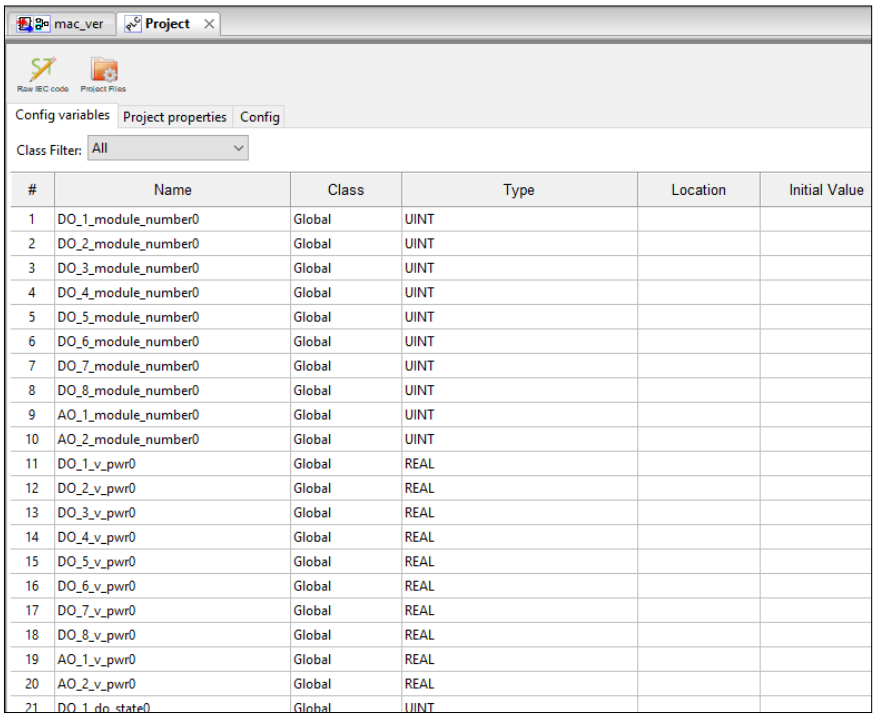


Рисунок А.2 – Дерево проекта

Дерево проекта имеет следующие опции:

переходить в структуру элемента проекта, а также добавлять и удалять элементы такие как: функция, ФБ, программа, модуль ввода/вывода, modbus поддержка, конфигурация проекта.



#	Name	Class	Type	Location	Initial Value
1	DO_1_module_number0	Global	UINT		
2	DO_2_module_number0	Global	UINT		
3	DO_3_module_number0	Global	UINT		
4	DO_4_module_number0	Global	UINT		
5	DO_5_module_number0	Global	UINT		
6	DO_6_module_number0	Global	UINT		
7	DO_7_module_number0	Global	UINT		
8	DO_8_module_number0	Global	UINT		
9	AO_1_module_number0	Global	UINT		
10	AO_2_module_number0	Global	UINT		
11	DO_1_v_pwr0	Global	REAL		
12	DO_2_v_pwr0	Global	REAL		
13	DO_3_v_pwr0	Global	REAL		
14	DO_4_v_pwr0	Global	REAL		
15	DO_5_v_pwr0	Global	REAL		
16	DO_6_v_pwr0	Global	REAL		
17	DO_7_v_pwr0	Global	REAL		
18	DO_8_v_pwr0	Global	REAL		
19	AO_1_v_pwr0	Global	REAL		
20	AO_2_v_pwr0	Global	REAL		
21	DO_1_do_state0	Global	UINT		

Рисунок А.3 – Панель переменных и констант

Панель переменных и констант отображает с помощью таблицы все глобальные переменные и константы проекта, указанные пользователем.

Каждая переменная имеет следующие параметры:

- Имя переменной, представляющее собой уникальный идентификатор переменной в пределах её области видимости и действия;
- Класс переменной, указывающий на ее роль в структуре проекта:
    - «Глобальная» (только этот тип, если указывается в главном окне проекта);
    - «Входная» (указывает, что данная переменная зависит от значения переменной, подаваемой на вход данного ФБ, функции);
    - «Выходная» (указывает, что от данной переменной зависит значение переменной, выходящей из выхода данного ФБ, функции);
    - «Входная/Выходная», «Локальная» (используется только в данном ФБ, функции и удаляется по окончании работы ФБ, функции);
    - «Внешняя» (возможно использовать любой программой/ФБ/функцией проекта);
    - «Временная».
- Тип, определяющий тип переменной и может принадлежать базовому типу (в соответствии со стандартом IEC 61131–3: BOOL, SINT, INT, LINT, DINT, USINT, UINT, ULINT, UDINT, REAL, LREAL, BYTE, STRING, WORD, LLWORD, DWORD, TIME, DAT, TOD, DT (последние 4 могут используются только в качестве внутренних переменных)), пользовательскому типу (ФБ, массиву);
- Адрес – идентификатор, необходимый для связывания данной переменной с Modbus–переменной;
- Начальное значение – инициализация переменной некоторым начальным значением;

- Опция – задание константности, реманентности (сохранение её значения в энергонезависимой памяти) и неремanентности переменной;
- Документация – комментарий к назначению данной переменной или константы.

Первый символ имени переменной или константы должен быть буквой, или символом (_), далее могут следовать цифры, буквы латинского алфавита и символы подчеркивания.

При выборе типа переменной «Array» (Массив) (см. рисунок А.4) появится окно «Edit array type properties» (изменение свойств массива).

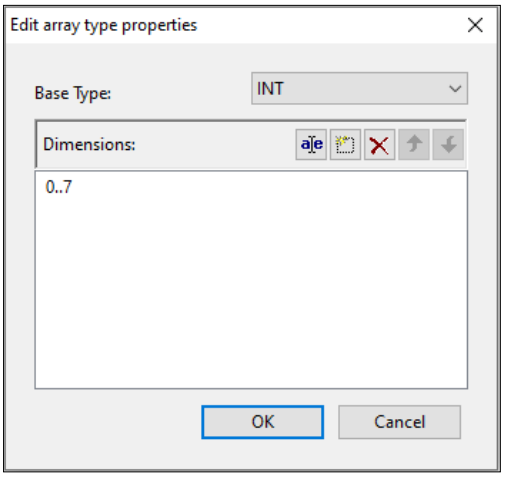


Рисунок А.4 – Редактирование свойств массива

Свойство «Base Type» определяет какому типу будут принадлежать элементы массива. Номера элементов массива при помощи кнопки «Edit item» Пример удачного создания массива приведен на рисунке А.5

#	Name	Class	Type
1	LocalVar0	Global	ARRAY [0..7] OF INT

Рисунок А.5 – созданный массив на панели переменных и констант

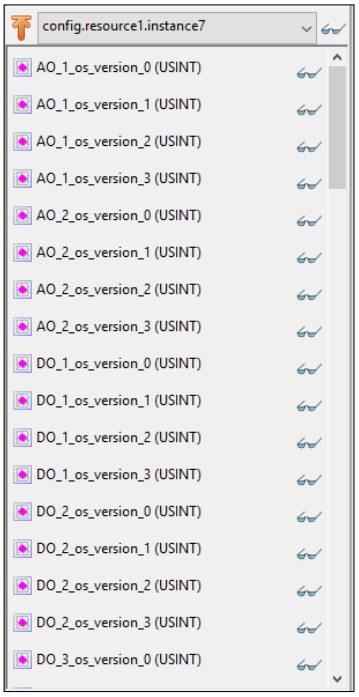


Рисунок А.6 – Панель экземпляров проекта

При выборе в дереве проекта элемента, соответствующего ресурсу, в панели экземпляров проекта будут отображены экземпляры, определённые в данном ресурсе, а также глобальные переменные ресурса.

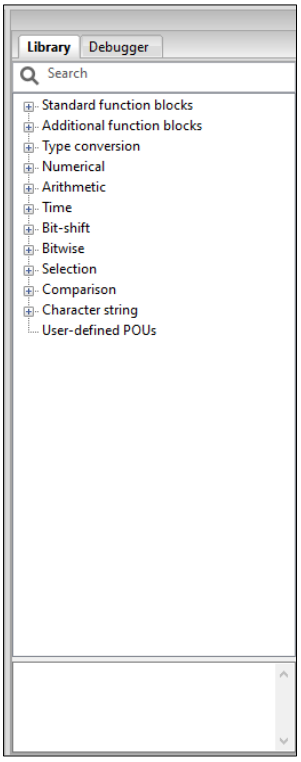


Рисунок А.7 – Панель библиотеки функций и функциональных блоков

Панель библиотеки функций и функциональных блоков содержит коллекцию стандартных функций и функциональных блоков, разделённых по разделам в соответствии с их назначением, которые доступны при написании алгоритмов и логики работы программных модулей. Выделены следующие разделы для функций и функциональных блоков: стандартные, дополнительные, преобразования типов данных, операций с числовыми данными, арифметических операций, временных операций, побитовых и смещения бит, операций выбора, операций сравнения, строковых операций. Помимо стандартных функций и функциональных блоков, данная панель содержит раздел «пользовательские программные модули». В него попадают функции и функциональные блоки, добавленные в конкретный проект, т.е. содержащиеся в дереве проекта. Использование данных функций и функциональных блоков осуществляется перетаскиванием необходимого блока с помощью зажатой левой кнопки мыши в область редактирования: либо текстовый редактор (ST), либо графический редактор (FBD). Имеется специальное поле поиска функционального блока по имени.

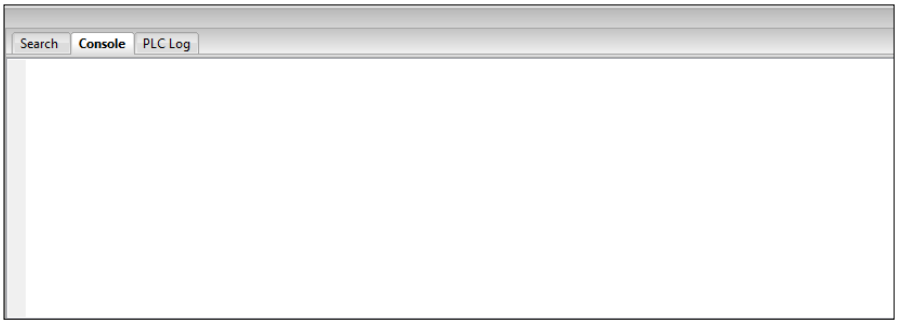


Рисунок А.8 – Отладочная панель

Отладочная панель служит для отображения в виде текстовых сообщений:

- Результатов генерации ST и C кода;
- Результатов компиляции и компоновки прикладной программы;
- Процесса соединения и передачи прикладной программы на целевое устройство;
- Различных промежуточных манипуляций в процессы создания прикладной программы.

В случае, если необходимо вывести предупреждения ИСР Beremiz или ошибки компиляторов (MatIEC или C кода) во время их работы цвет вывода текстовых сообщений становится красным (исключением составляет: CMake Warning: Manually–specified variables were not used by the project: CMAKE_SH). Критические ошибки также выделяется красным цветом, но при этом еще желтым фоном.

ПРИЛОЖЕНИЕ Б. ТЕОРИТИЧЕСКИЕ ОСНОВЫ ПО РАБОТЕ С MODBUS

Последовательный порт — наименование канала, по которому осуществляется связь по протоколу.

Скорость — скорость передачи данных по каналу.

Data_bits (выбор зависит от скорости передачи данных по каналу и четности).

Чётность — выбор режима (проверка на четность/проверка на нечетность/без проверки).

Стоп–биты — количество битов, оповещающих об окончании сообщения.

Период опроса — время между повторным опросом.

Адрес устройства — указание идентифицируемого номера устройства с кем осуществляется обмен данными

Число каналов — сколько регистров/реле используются в записи/чтении (для функций 05, 06 всегда равно 1).

Начальный адрес — номер первого регистра/реле используемого в записи/чтении.

Таймаут— время необходимое для ответа slave–устройству.

а) Функция 01 (ReadCoils) — Команда используется для получения состояний определенного количества реле, начиная с указанного в запросе. Состояние одного реле при этом передается одним битом. Если бит установлен в 1 – реле включено, если 0 – реле отключено.

б) Функция 02 (ReadInputDiscretes) — Команда используется для получения состояний определенного количества дискретных входов, начиная с указанного в запросе. Состояние одного входа при этом передается одним битом. Если бит установлен в 1 – вход замкнут, если 0 – вход разомкнут.

в) Функция 03 (ReadHoldingRegisters) — Команда используется для чтения указанного количества 2–Байтных регистров.

г) Функция 04 (ReadInputRegisters) — Команда используется для получения состояний определенного количества 2–Байтных регистров, хранящих состояние

дискретных входов, начиная с указанного в запросе. Значение одного регистра передается двумя байтами.

д) Функция 05 (WriteSingleCoil) — Команда используется для включения/отключения одного реле. Требуемое состояние реле передается двумя байтами.

е) Функция 06 (WriteSingleRegister) — Команда выполняет запись нового значения в указанный регистр.

ж) Функция 15 (WriteMultipleCoils) — Команда используется для групповой установки состояний определенного количества реле, начиная с указанного. Состояние одного реле при этом передается одним битом. Если бит установлен в 0 – реле отключено, если 1 – реле включено.

з) Функция 16 (WriteMultipleRegisters) — Команда выполняет запись новых значений в указанные регистры.

ПРИЛОЖЕНИЕ В. АДРЕСНОЕ ПОРСТРАНСТВО ПЛК BRIC

MB адрес диапазона от 0x60000	Тип переменной	Название регистра	Описания регистра
0	U16	mdb_addr	modbus address
1	U8	mdb_revers	reverse 3 and 4 function
1	U8	mdb_shift	shift start address regs from 0 to 1
2	U8	ip	IP address
4	U8	netmask	netmask address
6	U8	gateaway	gateaway address
8	U8	eth_speed	speed10–100mb
8	U8	eth_duplex	duplex full or half
9	U16	reset_num	number of system resets
10	U16	last_reset	reason of last system reset
11	U16	user_task_state	user task current state
12	U16	user_task_config	user task config
13	U16	uart1_sets	settings MESO_UART
14	U16	uart2_sets	settings RS_485_2
15	U16	uart3_sets	settings RS_232
16	U16	uart5_sets	settings RS_485_1
17	U16	uart6_sets	settings RS_485_IMMO
18	U16	uart7_sets	settings HART
19	U32	channels_timeout	time outs for channel use for retranslations
33	U8	do_state	state of digital output
33	U8	do_sc_ctrl	DO short circuit control
34	U16	do_ctrl	control digital output
35	U16	do_pwm_freq	PWM frequency Hz
36	U16	do_pwm_ctrl	PWM control
40	U16	di_noise_fltr_us	digital inputs noise filter in us (x10)

56	U32	di_pulseless_ms	digital inputs pulseless time in ms
88	U16	di_mode	digital inputs mode
104	U32	di_state	digital inputs state
106	U64	di_cnt	digital inputs cnt values
170	FLOAT	di_freq	digital inputs frequency values
202	U16	ai_unit	14 bit capacity or 12 bit capacity for analog inputs
210	U16	ai_state	ai_state
211	U16	ai_internal	12 bit capacity internal analog inputs
219	U16	ai_external	14 bit capacity external analog inputs
227	FLOAT	internal_temp	temperature internal sense value
229	FLOAT	external_temp	temperature ADC sense value
231	FLOAT	v_pwr	PWR voltage
233	FLOAT	v_bat	3V battery voltage
235	U64	sys_tick_counter	tick in ms
239	U64	tick100us	tick counter in 100us time
243	U8	time_hms	struct for real time
248	S32	unix_time_sec	unix_time_sec
250	U8	os_version	os_version
252	U8	mac_addr	ethernet mac address
255	U32	flash_err_cnt	flash_err cnt
257	U32	flags_task	check for task created
259	U64	counter_task	struct counter tasks
275	U32	flags_init_passed	inited modules
277	U32	flags_succ_init	success inited modules
279	U16	isol_pwr_state	isolated power state
280	U32	internal_task	internal_flash_dinamic_task
282	U32	external_task	external_flash_dinamic_task
284	U32	di_test_result	di_test result
286	U16	do_test_result	do test result
287	U16	ai_test_result	ai test result
288	U32	sofi_test_result	sofi_test blocks results
290	U32	sofi_test_blocks	sofi test blocks
292	U16	run_test	running tests

293	U32	cur_free_heap	cur_free_heap
295	U32	min_free_heap	min_free_heap
297	U8	debug_info	reserved use for debug
301	U32	rs_485_1_sends	RS–485_1 send num
303	U32	rs_485_1_errors	RS–485_1 errors
305	U32	rs_485_2_sends	RS–485_2 send num
307	U32	rs_485_2_errors	RS–485_2 errors
309	U32	rs_232_sends	RS–232 send num
311	U32	rs_232_errors	RS–232 errors
313	U16	command	Command register
314	U16	num_of_vars	num_of_vars
315	U16	current_os	using os 1 or 2
316	U16	module_number	module ao number 0 – 127
317	U32	can_sdo_error	can_sdo_error
319	U8	local_ip	ip address for local net
321	U8	local_netmask	netmask address for local net
323	U8	local_gateaway	gateaway address for local net
325	U32	monitor_period	sofi_monitor period in ms
327	FLOAT	total_tasks_time	total_tasks_time
329	U8	task0	task0
343	U8	task1	task1
357	U8	task2	task2
371	U8	task3	task3
385	U8	task4	task4
399	U8	task5	task5
413	U8	task6	task6
427	U8	task7	task7
441	U8	task8	task8
455	U8	task9	task9
469	U8	task10	task10
483	U8	task11	task11
497	U8	task12	task12
511	U8	task13	task13

525	U8	task14	task14
539	U8	task15	task15
553	U8	task16	task16
567	U8	task17	task17
581	U8	task18	task18
595	U8	task19	task19
609	U8	task20	task20
623	U8	task21	task21
637	U8	task22	task22
651	U8	task23	task23
665	U8	task24	task24
679	U8	task25	task25
693	U8	task26	task26
707	U8	task27	task27
721	U8	task28	task28
735	U8	task29	task29
749	U8	task30	task30
763	U8	task31	task31
777	U16	link	link
778	U16	eth_arp	eth_arp
779	U16	ip_frag	ip_frag
780	U16	ip_proto	ip_proto
781	U16	icmp	icmp
782	U16	udp	udp
783	U16	tcp	tcp
784	U16	mem_heap	mem_heap
785	U16	memp_udp_pool	memp_udp_pool
786	U16	memp_tcp_pool	memp_tcp_pool
787	U16	memp_listen_tcp	memp_listen_tcp
788	U16	memp_seg_tcp	memp_seg_tcp
789	U16	memp_altcp	memp_altcp
790	U16	memp_reassdata	memp_reassdata
791	U16	memp_frag_pbuf	memp_frag_pbuf

792	U16	memp_net_buf	memp_net_buf
793	U16	memp_net_conn	memp_net_conn
794	U16	memp_tcpip_api	memp_tcpip_api
795	U16	memp_tcpip_input	memp_tcpip_input
796	U16	memp_sys_timeout	memp_sys_timeout
797	U16	memp_pbuf_ref	memp_pbuf_ref
798	U16	memp_pbuf_pool	memp_pbuf_pool
799	U16	lwip_sys	lwip_sys

ПРИЛОЖЕНИЕ Г. ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ ИСР BEREMIZ

Целевое устройство – аппаратное средство с определённой архитектурой процессора, на котором могут исполняться различные исполняемые файлы, обращающиеся с помощью него к модулям устройства связи с объектом (УСО).

Прикладная программа (исполняемый файл) для целевого устройства – скомпилированный и скомпонованный so–файл, который будет выполняться на целевом устройстве.

Плагин для модуля УСО – интерфейс, состоящий из специальных драйверов и элементов пользовательского интерфейса для ИСР Beremiz, позволяющий связывать переменные модулей УСО с переменными программных модулей, из которых состоит проект.

Проект – совокупность программных модулей (программ, функциональных блоков, функций), плагинов внешних модулей УСО, ресурсов, пользовательских типов данных, сборка (компиляция и компоновка) которых, представляет собой прикладную программу для целевого устройства. Каждый проект сохраняется в отдельном файле.

Переменная – область памяти, в которой находятся данные, с которыми оперирует программный модуль.

Ресурс – элемент, отвечающий за конфигурацию проекта: глобальные переменные и экземпляры проекта, связываемыми с программными модулями типа «Программа» и задачами.

Программный модуль – элемент, представляющий собой функцию, ФБ или программу. Каждый программный модуль состоит из раздела объявлений и кода. Для написания всего кода программного используется только один из языков программирования стандарта IEC 61131–3.

Функция – программный модуль, который возвращает только единственное значение, которое может состоять из одного и нескольких элементов (если это битовое поле или структура).

Функциональный блок – программный модуль, который принимает и возвращает произвольное число значений, а также позволяет сохранять своё состояние (подобно классу в различных объектно–ориентированных языках). В отличие от функции ФБ не формирует возвращаемое значение.

Программа – программный модуль, представляющий собой единицу исполнения, как правило, связывается (ассоциируется) с задачей.

Задача – элемент представляющий время и приоритет выполнения программного модуля типа «Программа» в рамках экземпляра проекта.

Экземпляр – представляет собой программу, как единицу исполнения, связанную (ассоциированную) с определённой задачей. Так же, как экземпляр, рассматриваются переменные, определённые в программных модулях: программа и ФБ.

Пользовательский тип данных – тип данных, добавленный в проект и представляющий собой: псевдоним существующего типа, под диапазон существующего типа, перечисление, массив или структуру.

Класс переменной – тип использования переменной:

- Локальная (появляется при работе ФБ где фигурирует);
- Вход (локальная переменная, требующая подключения внешней переменной на вход ФБ/функции, где она фигурирует);
- Выход (локальная переменная, требующая подключения внешней переменной на выход из ФБ/функции, где она фигурирует);
- Вход/Выход (локальная переменная, требующая подключения внешней переменной на выход и вход ФБ/функции, где она фигурирует),
- Внешняя[8] (сохраняется в адресном пространстве ПЛК).

Исходное значение – значение переменной на момент запуска программы на ПЛК.

[8] Не поддерживаются типы переменных TIME, DATE, TOD, DT, STRING

Настройка переменной – возможность изменение переменной:

- constant (неизменная);
- retain (сохранение значения при перезагрузке ПЛК);
- non-retain (сброс значения при перезагрузке).

ПРИЛОЖЕНИЕ Д. ОПИСАНИЕ ФБ БИБЛИОТЕКИ ИСР BEREMIZ ДЛЯ ПЛК BRIC

Стандартные:

1 Блок SR

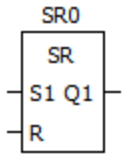


Рисунок Д.1 – FBD

Таблица Д.1 – Переменные блока SR

Входа	**Тип**	**Описание**
S1	BOOL	Вход (доминирующий)
R	BOOL	Сброс
Выхода	**Тип**	**Описание**
Q	BOOL	Выход становится "1", когда на вход S1 приходит "1". При переходе S1 в "0" сохраняется состояние. Выход Q1 возвращается в "0", когда вход R становится"1"

Форма записи на языке ST:

FB(S1 := *BOOL*, R := *BOOL*);

BOOL := *FB*.Q1;

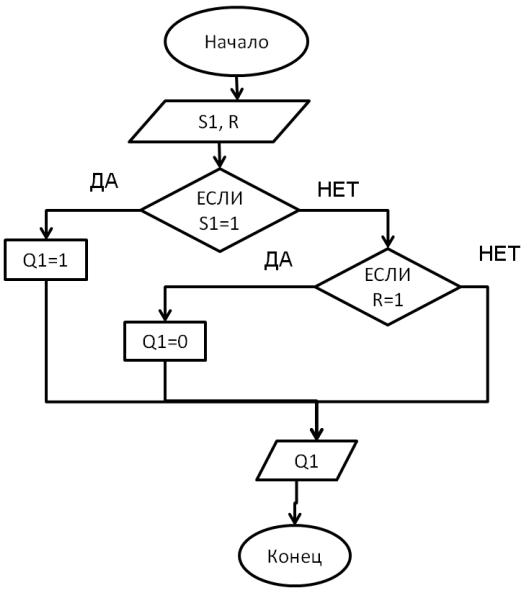


Рисунок Д.2 – Блок–схема[9]

[9] Для использования стандартных и дополнительных функциональных блоков необходимо перетащить необходимый из библиотеки в зону перечисления переменных и дать наименование

2 Блок RS

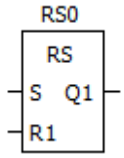


Рисунок Д.3 – FBD

Таблица Д.2 – Переменные блока RS

Входа	**Тип**	**Описание**
S	BOOL	Вход
R1	BOOL	Сброс (доминирующий)
Выхода	**Тип**	**Описание**
Q	BOOL	Выход становится "1", когда вход R1 становится "0". При переходе R1 в "0" сохраняется состояние. Выход Q1 возвра–щается в "1", когда вход S становится "1"

Форма записи на языке ST:

FB (S := *BOOL*, R1 := *BOOL*);

BOOL := *FB*.Q1;

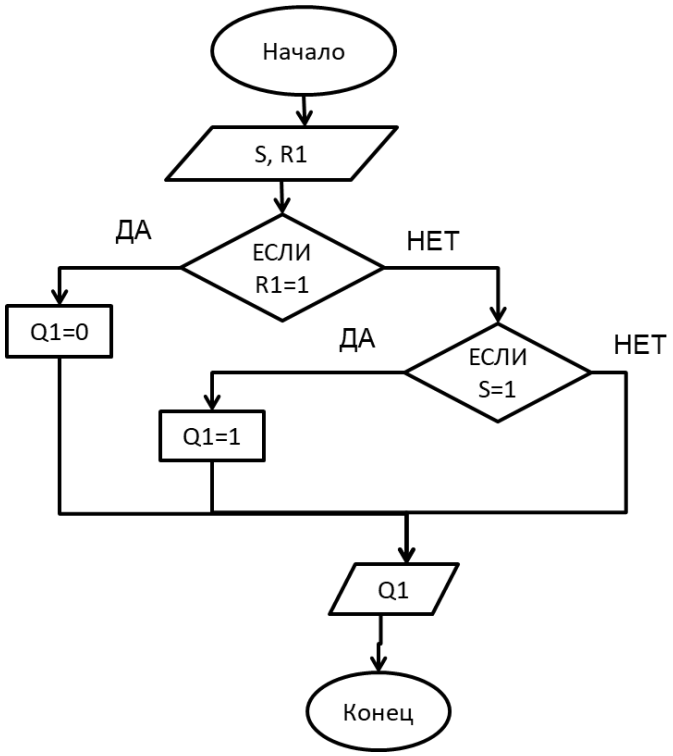


Рисунок Д.4 – Блок–схема

3 Блок SEMA

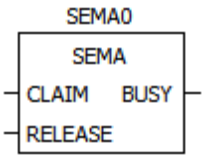


Рисунок Д.5 – FBD

Таблица Д.3 – Переменные блока SEMA

Входа	**Тип**	**Описание**
CLAIM	BOOL	Вход(доминирующий)
RELEASE	BOOL	Сброс
Выхода	**Тип**	**Описание**
BUSY	BOOL	Выход активируется, при CLAIM=1 и деактивируется, при RELEASE=1.

Форма записи на языке ST:

FB(CLAIM := *BOOL*);

BOOL := *FB*.RELEASE;

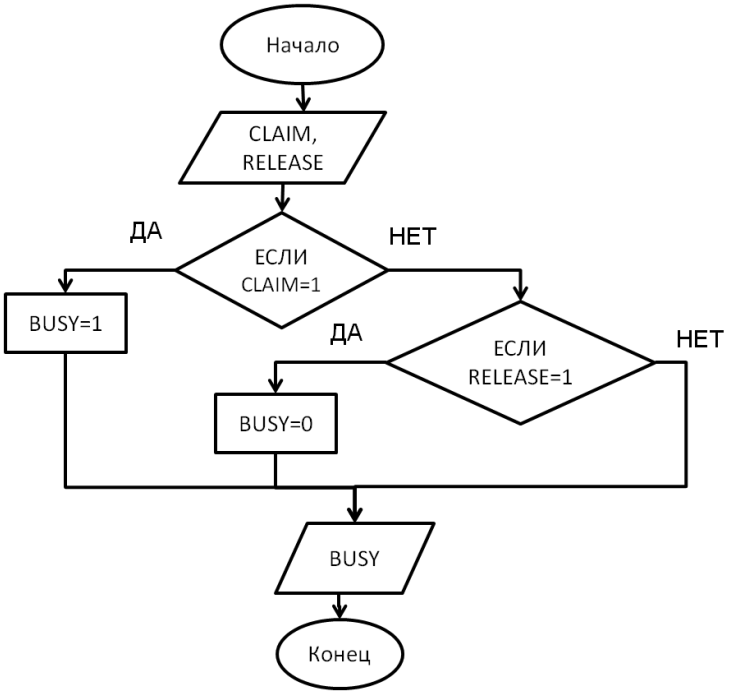


Рисунок Д.6 – Блок–схема

3 Блок R–TRIG

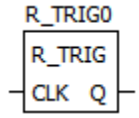


Рисунок Д.7 – FBD

Таблица Д. 4 – Переменные блока R–TRIG

Входа	**Тип**	**Описание**
CLK	BOOL	Вход
Выхода	**Тип**	**Описание**
Q	BOOL	Выход генерирует единичный импульс, если на входе передний фронт

Форма записи на языке ST:

FB(CLK := *BOOL*);

BOOL := *FB*.Q;

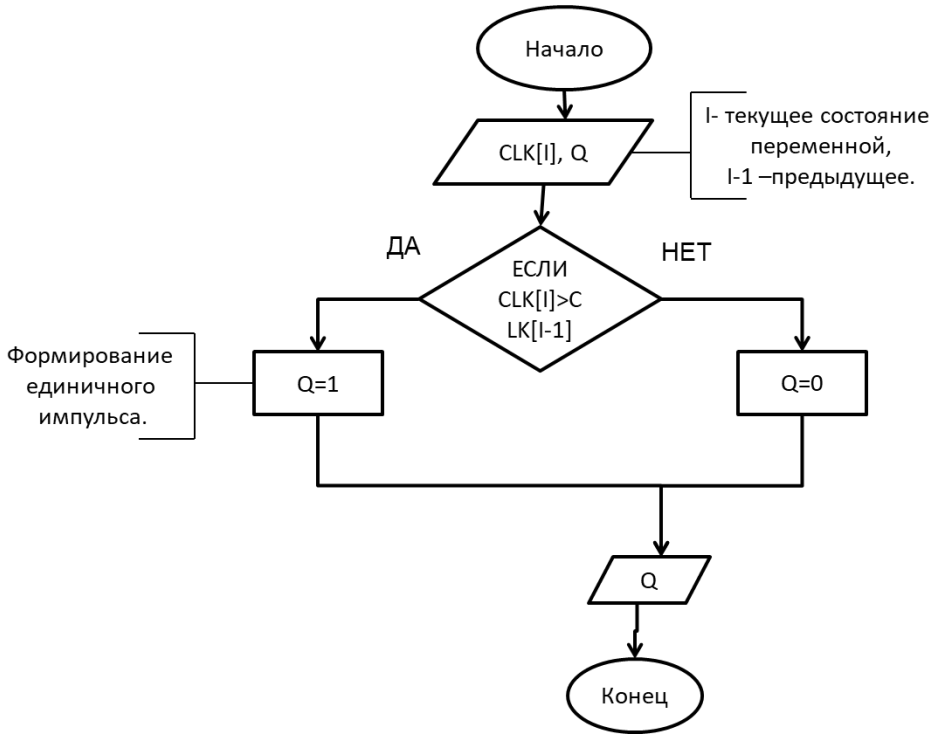


Рисунок Д.8 – Блок–схема

4 Блок F–TRIG

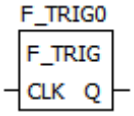


Рисунок Д.9 – FBD

Таблица Д. 5 – Переменные блока F–TRIG

Входа	**Тип**	**Описание**
CLK	BOOL	Вход
Выхода	**Тип**	**Описание**
Q	BOOL	Выход генерирует единичный импульс, если на входе задний фронт

Форма записи на языке ST:

FB(CLK := *BOOL*);

BOOL := *FB*.Q;

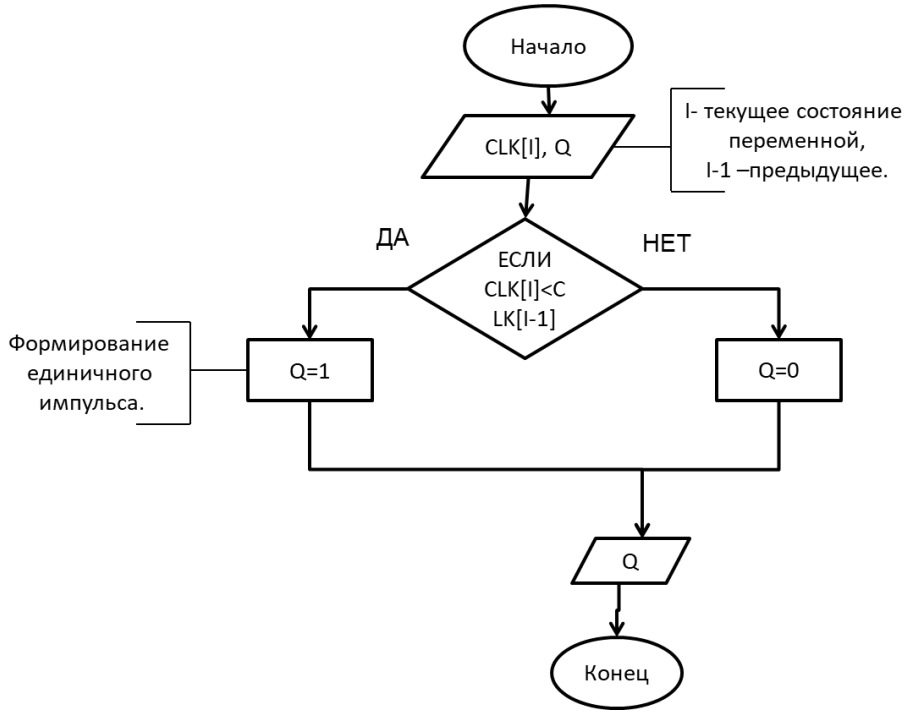


Рисунок Д.10 – Блок–схема

5 Блок CTU/ CTU_DINT …

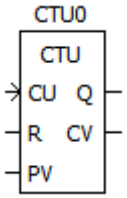


Рисунок Д.11 – FBD

Таблица Д.6 – Переменные блока CTU

Входа	**Тип**	**Описание**
CU	BOOL	Подача импульса
R	BOOL	Сброс
PV	ANY_INT	Предел счета
Выхода	**Тип**	**Описание**
Q	BOOL	Принимает значение "TRUE" когда CV>=PV
CV	ANY_INT	Считает количество импульсов(CV=CV+1) пока Q=0

Форма записи на языке ST:

FB(CU := *BOOL*, R := *BOOL*, PV := *ANY_INT*);

BOOL := *FB*.Q;

ANY_INT := *FB*.CV;

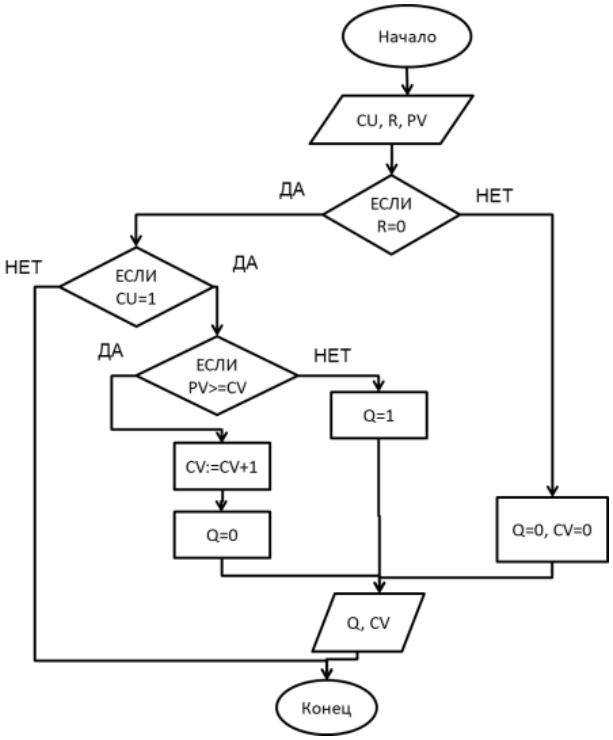


Рисунок Д.12 – Блок–схема[10]

[10] PV не включает тип данных USINT, SINT

6 Блок CTD/ CTD_DINT …

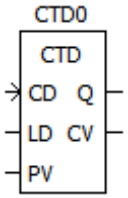


Рисунок Д.13 – FBD

Таблица Д.7 – Переменные блока CTD

Входа	**Тип**	**Описание**
CD	BOOL	Подача импульса
LD	BOOL	Сброс
PV	ANY_INT	Предел импульсов
Выхода	**Тип**	**Описание**
Q	BOOL	Принимает значение "TRUE" когда CV=0
CV	ANY_INT	Считает количество импульсов(CV=CV–1) пока Q=0

Форма записи на языке ST:

FB(CD := *BOOL*, LD := *BOOL*, PV := *ANY_INT*);

BOOL := *FB*.Q;

ANY_INT := *FB*.CV;

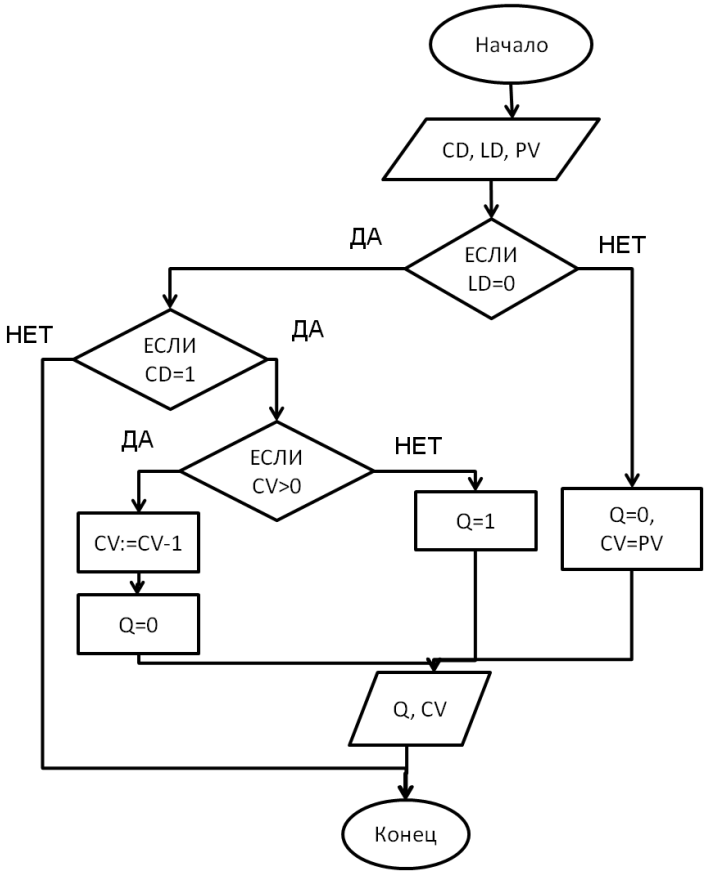


Рисунок Д.14 – Блок–схема

7 Блок CTUD/ CTUD_DINT

…

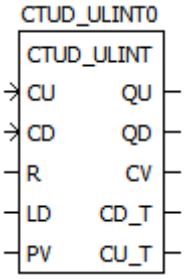


Рисунок Д.15 – FBD

Таблица Д.8 – Переменные блока CTUD

Входа	**Тип**	**Описание**
CD	BOOL	Подача импульса
LD	BOOL	Подача импульса
R	BOOL	Сброс до 0
LD	BOOL	Сброс до PV
PV	ANY_INT	Верхний предел импульсов
Выхода	**Тип**	**Описание**
QU	BOOL	Принимает значение "TRUE" когда CV>=PV
QD	BOOL	Принимает значение "TRUE" когда CV=0
CV	ANY_INT	Считает количество импульсов(CV=CV–1, если CD=1,CV=CV+1, если CU=1) пока Q=0
CD_T	R_TRIG	(в существующей версии не применяется)
CU_T	R_TRIG	(в существующей версии не применяется)

Форма записи на языке ST:

FB (CU := *BOOL*, CD := *BOOL*, R := *BOOL*, LD := *BOOL*,

PV := *ANY_INT*);

BOOL := *FB*.QU;

BOOL := *FB*.QD;

ANY_INT := *FB*.CV;

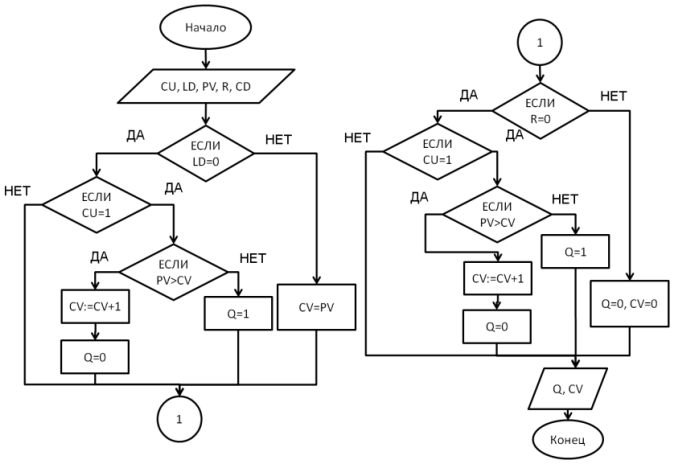


Рисунок Д.16 – Блок–схема

8 Блок TP

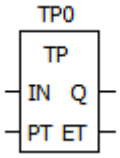


Рисунок Д.17 – FBD

Таблица Д.9 – Переменные блока TP

Входа	**Тип**	**Описание**
IN	BOOL	Подача импульса
PT	TIME	Время одного импульса
Выхода	**Тип**	**Описание**
Q	BOOL	Выход (пока ET<PT, Q="TRUE")
ET	TIME	Пока IN=1 и ET<PT, счет времени ET

Форма записи на языке ST:

FB(IN := *BOOL*, PT := *TIME*);

BOOL := *FB*.Q;

TIME := *FB*.ET;

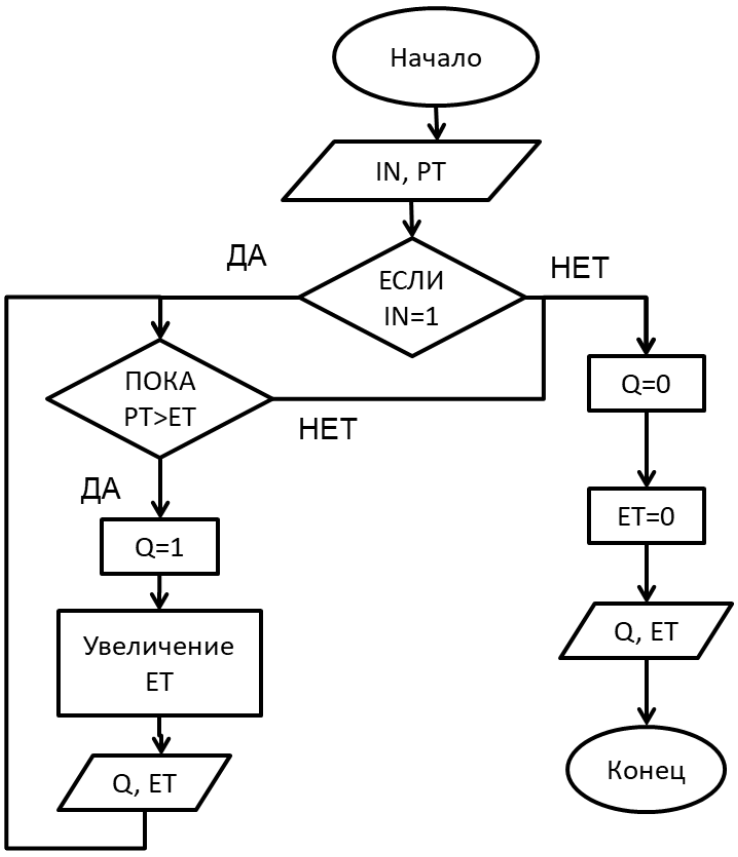


Рисунок Д.18 – Блок–схема

9 Блок TON

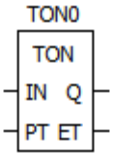


Рисунок Д.19 – FBD

Таблица Д.10 – Переменные блока TON

Входа	**Тип**	**Описание**
IN	BOOL	Вход
PT	TIME	Время задержки
Выхода	**Тип**	**Описание**
Q	BOOL	Выход. Если ET=PV и IN=1 то Q=1, иначе Q=0
ET	TIME	Счетчик времени считает, пока ET<PV и IN=1

Форма записи на языке ST:

FB(IN := *BOOL*, PT := *TIME*);

BOOL := *FB*.Q;

TIME := *FB*.ET;

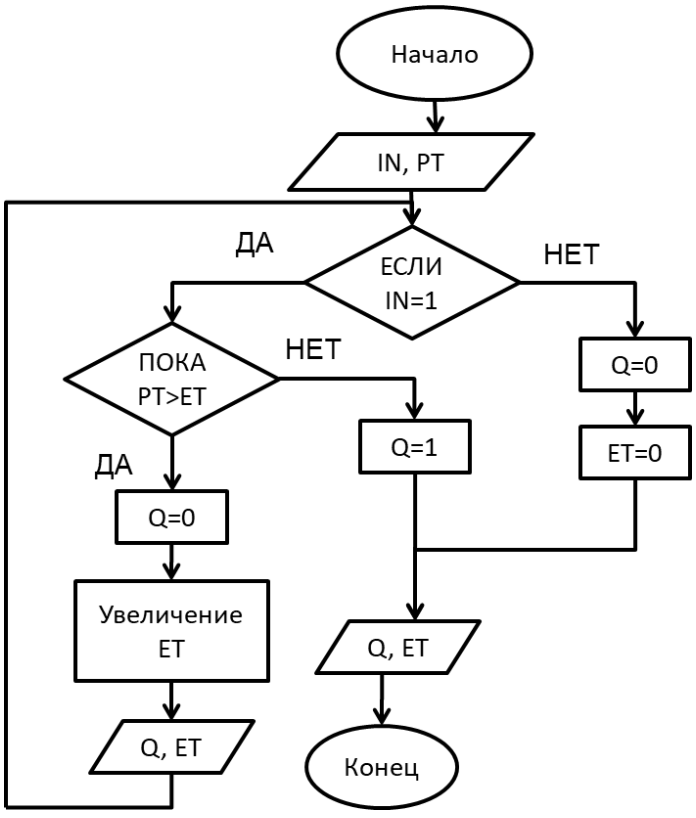


Рисунок Д.20 – Блок–схема

10 Блок TOF

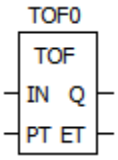


Рисунок Д.21 – FBD

Таблица Д.11 – Переменные блока TOF

Входа	**Тип**	**Описание**
IN	BOOL	Вход
PT	TIME	Сколько времени будет задержка
Выхода	**Тип**	**Описание**
Q	BOOL	Выход. Если ET=PV и IN=1то Q=1, иначе Q=0
ET	TIME	Счетчик времени считает, пока ET<PV и IN=1

Форма записи на языке ST:

FB(IN := *BOOL*, PT := *TIME*);

BOOL := *FB*.Q;

TIME := *FB*.ET;

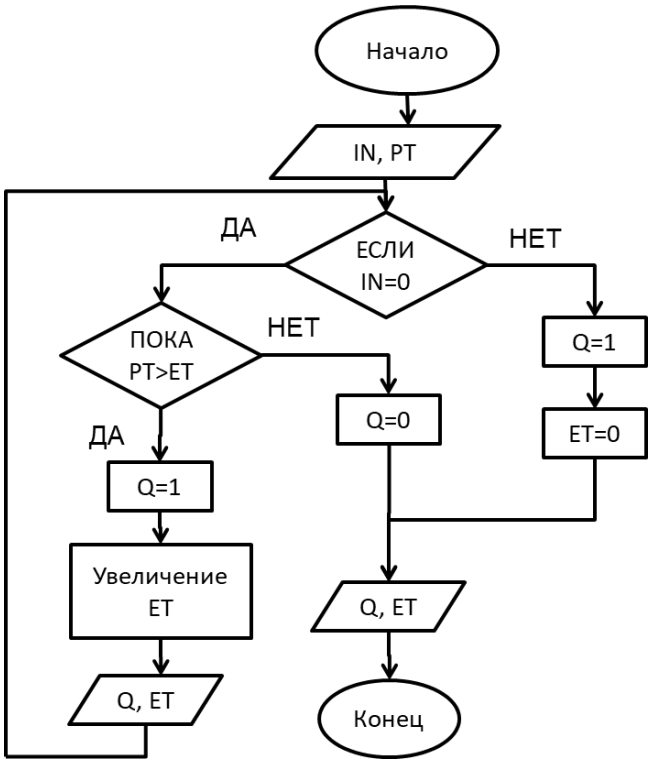


Рисунок Д.22 – Блок–схема

Дополнительные:

1 Блок RTC

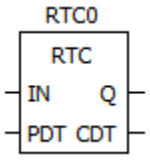


Рисунок Д.23 – FBD

Таблица Д.12 – Переменные блока RTC

Входа	**Тип**	**Описание**
IN	BOOL	Переключение режима
PDT	DT	Время начала работы
Выхода	**Тип**	**Описание**
Q	BOOL	Индикация режима
CDT	DT	Вывод времени

Форма записи на языке ST:

FB(IN := *BOOL*, PDT := *DT*);

BOOL := *FB*.Q;

DT := *FB*.CDT;

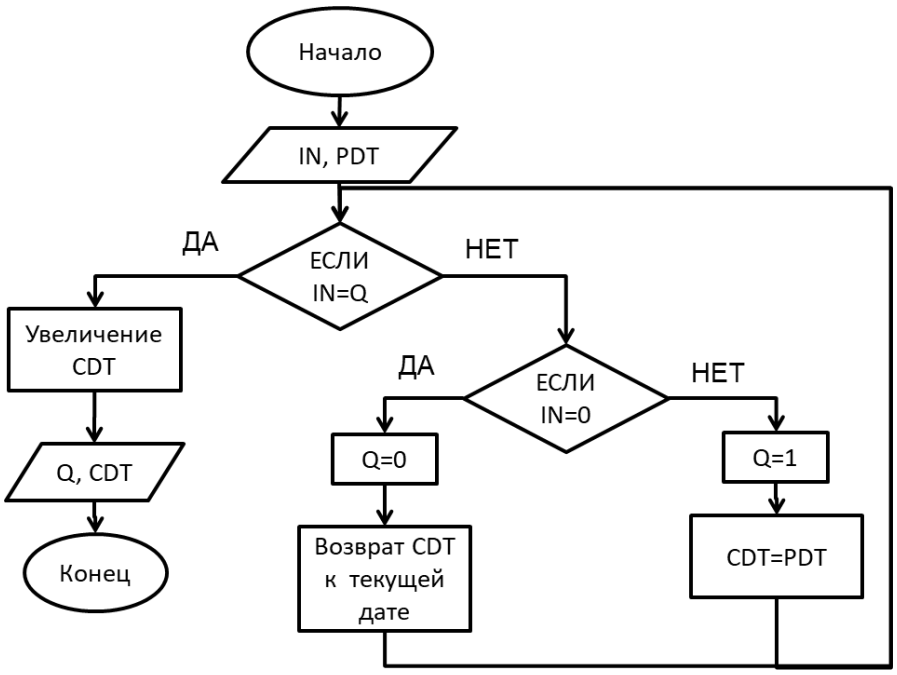


Рисунок Д.24 – Блок–схема

2 Блок INTEGRAL

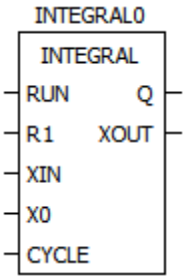


Рисунок Д.25 – FBD

Таблица Д.13 – Переменные блока INTEGRAL

Входа	**Тип**	**Описание**
RUN	BOOL	Включение блока
R1	BOOL	Сброс (XOUT=X0)
XIN	REAL	Интегрируемое значение
X0	REAL	Начальное значение XOUT
CYCLE	TIME	Время интегрирования
Выхода	**Тип**	**Описание**
Q	BOOL	Q=1, если R1=0
XOUT	REAL	Выход (интегрирует входное значение XIN во времени) (XOUT=X0*CYCLE*(количество тактов при RUN=1))

Форма записи на языке ST:

FB(RUN := *BOOL*, R1 := *BOOL*, XIN := *REAL*, X0 := *REAL*,

CYCLE := *TIME*);

BOOL := *FB*.Q;

REAL := *FB*.XOUT;

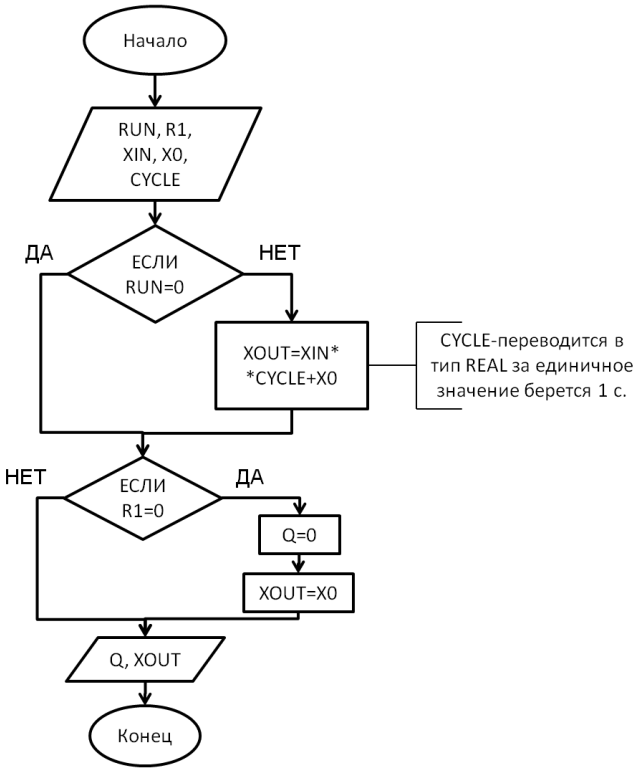


Рисунок Д.26 – Блок–схема

3 Блок DERIVATIVE

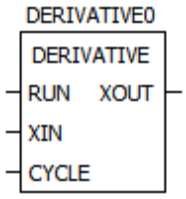


Рисунок Д.27 – FBD

Таблица Д.14 – Переменные блока DERIVATIVE

Входа	**Тип**	**Описание**
RUN	BOOL	Включение блока
XIN	REAL	Вход
CYCLE	TIME	Время дифференцирования
Выхода	**Тип**	**Описание**
XOUT	REAL	Формирование сигнала пропорционально частоте изменения входа XIN

Форма записи на языке ST:

FB(RUN := *BOOL*, XIN := *REAL*, CYCLE := *TIME*);

REAL := *FB*.XOUT;

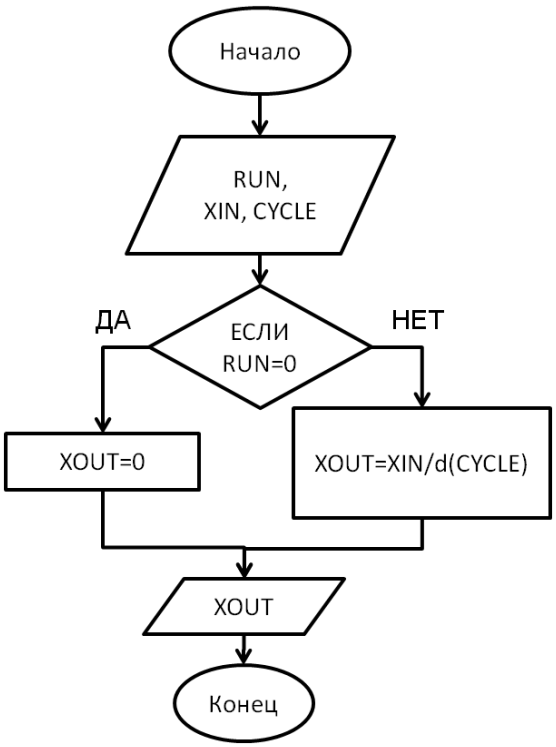


Рисунок Д.28 – Блок–схема

4 БлокPID

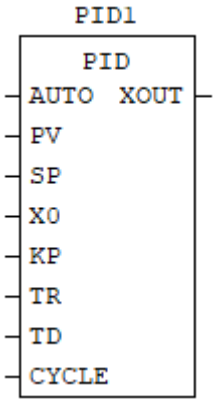


Рисунок Д.29 – FBD

Таблица Д.15 – Переменные блока PID

Входа	**Тип**	**Описание**
AUTO	BOOL	Включение блока(переход от ручного режима к автоматическому)
PV	REAL	Задание(автоматическое упр.)
SP	REAL	Присваиваемое значение
X0	REAL	Задание (ручное упр.)
KP	REAL	П – составляющая
TR	REAL	И – составляющая
TD	REAL	Д – составляющая
CYCLE	TIME	Время цикла
Выхода	**Тип**	**Описание**
XOUT	REAL	Выход

Форма записи на языке ST:

FB(AUTO := *BOOL*, PV := *REAL*, SP := *REAL*, X0 := *REAL*,

KP := *REAL*, TR := *REAL*, TD := *REAL*, CYCLE := *TIME*);

REAL:= *FB*.XOUT;

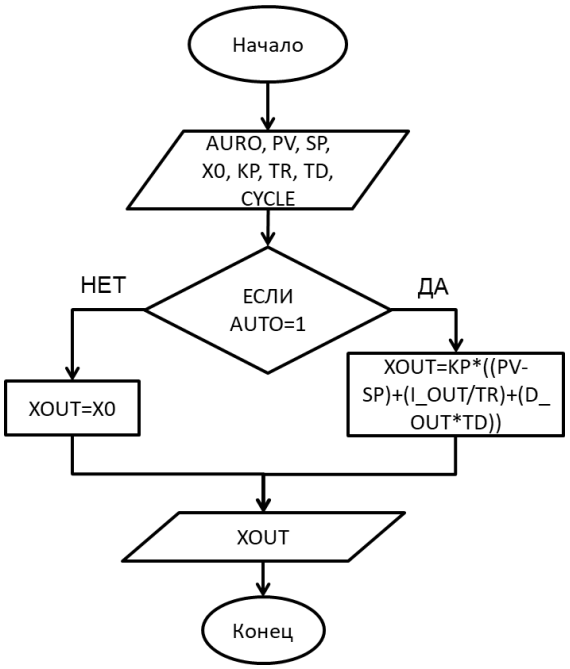


Рисунок Д.30 – Блок–схема

5 Блок RAMP

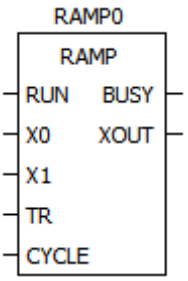


Рисунок Д.31 – FBD

Таблица Д.16 – Переменные блока RAMP

Входа	**Тип**	**Описание**
RUN	BOOL	Включение блока
X0	REAL	Начало отсчета
X1	REAL	Конечное значение
TR	TIME	Время перехода
CYCLE	TIME	Периодичность повтора
Выхода	**Тип**	**Описание**
BUSY	BOOL	BUSY=1, если XOUT изменяется
XOUT	REAL	Если RUN=1, то пока XOUT<X1 XOUT=X1*CYCLE*t/TR+X0 иначе XOUT=X0

Форма записи на языке ST:

FB(RUN := *BOOL*, X0 := *REAL*, X1 := *REAL*, TR := *TIME*,

CYCLE := *TIME*);

BOOL := *FB*.BUSY;

REAL := *FB*.XOUT;

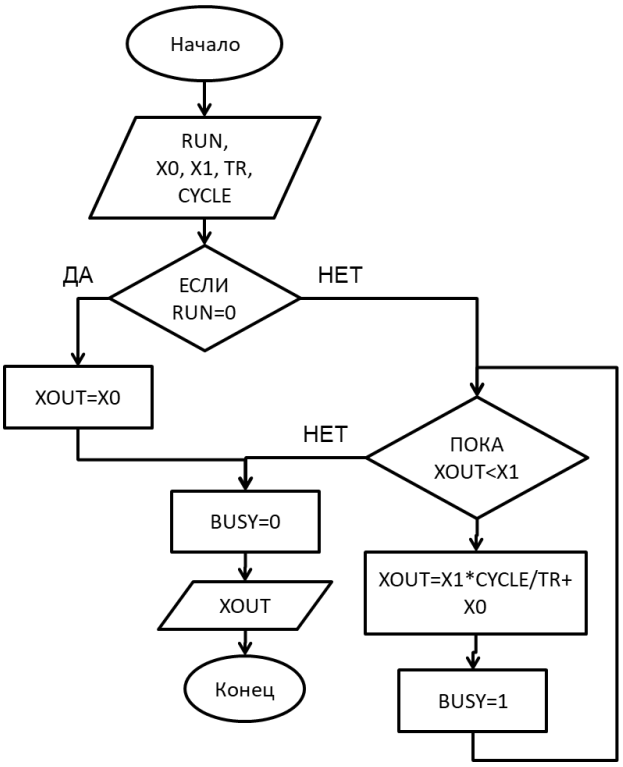


Рисунок Д.32 – Блок–схема

6 Блок READ_DI

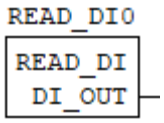


Рисунок Д.33 – FBD

Таблица Д. 17 – Переменные блока READ_DI

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
DI_OUT	UDINT	Состояние дискретных входов ПЛКBRIC

Форма записи на языке ST:

FB();

UDINT:= *FB*.DI_OUT;

7 Блок READ_ DI_CNT

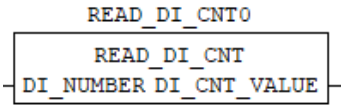


Рисунок Д.34 – FBD

Таблица Д. 18 – Переменные блока READ_DI_CNT

Входа	**Тип**	**Описание**
DI_NUMBER	UINT	Номер исследуемого дискретного входа
Выхода	**Тип**	**Описание**
DI_CNT_VALUE	ULINT	Значение счетчика дискретного входа

Форма записи на языке ST:

FB (DI_NUMBER := *UINT*);

ULINT := *FB*.DI_CNT_VALUE;

8 Блок READ_ DI_FREQ

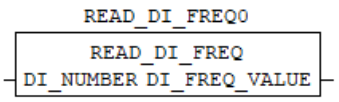


Рисунок Д.35 – FBD

Таблица Д. 19 – Переменные блока READ_DI_FREQ

Входа	**Тип**	**Описание**
DI_NUMBER	UINT	Номер исследуемого дискретного входа
Выхода	**Тип**	**Описание**
DI_FREQ_VALUE	REAL	Значение частоты переключения дискретного входа

Форма записи на языке ST:

FB(DI_NUMBER := *UINT*);

REAL := *FB*.DI_FREQ_VALUE;

9 Блок READ_AI

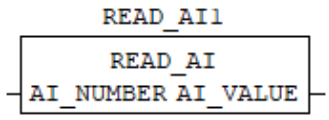


Рисунок Д.36 – FBD

Таблица Д. 20 – Переменные блока READ_AI

Входа	**Тип**	**Описание**
AI_NUMBER	UINT	Номер канала
Выхода	**Тип**	**Описание**
AI_VALUE	UINT	Показание аналогового канала по шкале 0–16383, которая соответствует шкале по принимаемого унифицированного сигнала

Форма записи на языке ST:

FB(AI_NUMBER := * UINT*);

* UINT* := *FB*.AI_VALUE;

10 Блок READ_ DO_SC

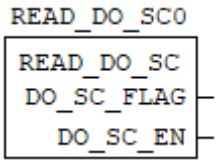


Рисунок Д.37 – FBD

Таблица Д. 21 – Переменные блока READ_DO_SC

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
DO_SC_FLAG	USINT	Дискретные выхода для которых включена программная защита от короткого замыкания (младший бит соответствует DO_3)
DO_SC_EN	USINT	Дискретные выхода на которых сработала программная защита от короткого замыкания (младший бит соответствует DO_3)

Форма записи на языке ST:

FB();

USINT := *FB*.DO_SC_FLAG;

USINT := *FB*.DO_SC_EN;

11 Блок READ_DO

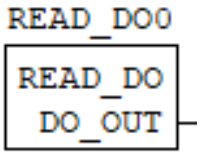


Таблица Д. 22 – Переменные блока READ_DO

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
DO_OUT	USINT	Чтение значения состояния дискретных выходов

Рисунок Д.38 – FBD

Форма записи на языке ST:

FB();

USINT := *FB*.DO_OUT;

12 Блок READ_ RESET

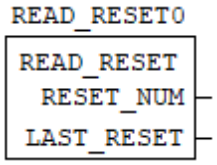


Рисунок Д.39 – FBD

Таблица Д.23 – Переменные блока READ_RESET

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
RESET_NUM	UINT	Количество перезагрузок с момента обновления ОС ПЛК
LAST_RESET	UINT	Код причины последней перезагрузки

Форма записи на языке ST:

FB();

UINT := *FB*.RESET_NUM;

UINT := *FB*.LAST_RESET;

13 Блок READ_ PWR

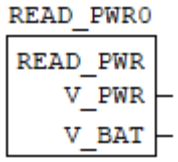


Рисунок Д.40 – FBD

Таблица Д.24 – Переменные блока READ_PWR

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
V_PWR	REAL	Входное напряжение ПЛК
V_BAT	REAL	Напряжение батареи ПЛК

Форма записи на языке ST:

FB();

REAL := *FB*.V_PWR;

REAL *FB*.V_BAT;

14 Блок READ_INTERNAL_TEMP

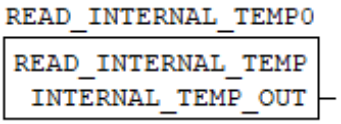


Рисунок Д.41 – FBD

Таблица Д.25 – Переменные блока READ_INTERNAL_TEMP

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
INTERNAL_TEMP_OUT	REAL	Показание температуры микропроцессора в ПЛК

Форма записи на языке ST:

FB();

REAL := *FB*.INTERNAL_TEMP_OUT;

15 Блок READ_ SYS_TICK_ COUNTER

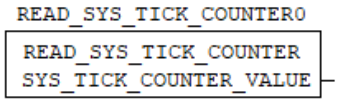


Рисунок Д.42 – FBD

Таблица Д. 26 – Переменные блока READ_SYS_ TICK_COUNTER

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
SYS_TICK_COUNTER_VALUE	ULINT	Чтение времени выполнения с момента последнего сброса в мс.

Форма записи на языке ST:

FB();

ULINT:= *FB*.SYS_TICK_COUNTER_VALUE;

16 Блок WRITE_ MDB_ADRESS

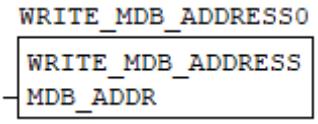


Рисунок Д.43 – FBD

Таблица Д. 27 – Переменные блока WRITE_MDB_ ADRESS

Входа	**Тип**	**Описание**
MDB_ADDR	UINT	Установление адреса по протоколу Modbus в ПЛК
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(MDB_ADDR := *UINT*);

17 Блок WRITE_ UART_SETS

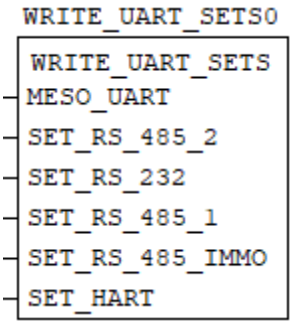


Рисунок Д.44 – FBD

Таблица Д. 28 – Переменные блока WRITE_UART_SETS

Входа	**Тип**	**Описание**
MESO_UART	UINT	Запись параметров настройки интерфейса на канале Mesonin (доступ к порту только при вскрытии крышки ПЛК)
SET_RS_485_2	UINT	Запись параметров настройки интерфейса на канале RS–485–1
SET_RS_232	UINT	Запись параметров настройки интерфейса на канале RS–232
SET_RS_485_1	UINT	Запись параметров настройки интерфейса на канале RS–485–1
SET_RS_485_IMMO	UINT	Запись параметров настройки интерфейса на межмодульной шине
SET_HART	UINT	Запись параметров HART интерфейса на каналах AI
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(MESO_UART:= *UINT*, SET_RS_485_2:= *UINT*, SET_RS_232:= *UINT*, SET_RS_485_1:= *UINT*, SET_RS_485_IMMO:= *UINT*, SET_HART:= *UINT*);

18 Блок WRITE_ CH_TIMEOUT

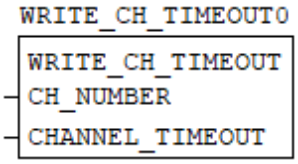


Рисунок Д.45 – FBD

Таблица Д. 29 – Переменные блока WRITE_ CH_TIMEOUT

Входа	**Тип**	**Описание**
CH_NUMBER	USINT	Номер канала
CHANNEL_TIMEOUT	UDINT	Задержка
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(CH_NUMBER:= *USINT*, CHANNEL_TIMEOUT:= *UDINT*);

19 Блок WRITE_ DI_NOISE_ FLTR_10US

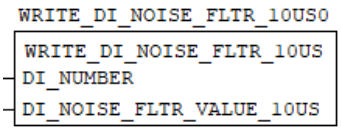


Рисунок Д.46 – FBD

Таблица Д.30 – Переменные блока WRITE_ DI_NOISE_FLTR_10US

Входа	**Тип**	**Описание**
DI_NUMBER	UINT	Номер канала дискретного входа ПЛКBRIC
DI_NOISE_FLTR_ VALUE_10US	UINT	Период, за который счетчик обнаруживает не более 1 импульса
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(DI_NUMBER := *UINT*, DI_NOISE_FLTR_VALUE_10US := *UINT*);

20 Блок WRITE_DI_ PULSELESS

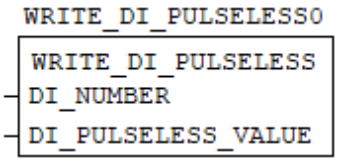


Рисунок Д.47 – FBD

Таблица Д. 31 – Переменные блока WRITE_DI_ PULSELESS

Входа	**Тип**	**Описание**
DI_NUMBER	UINT	Номер канала дискретного входа ПЛКBRIC
DI_PULSELESS_ VALUE	UDINT	Период, относительно которого рассчитывается частота канала
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(DI_NUMBER:= *UINT*, DI_PULSELESS_VALUE:= *UDINT*);

21 Блок WRITE_ DI_MODE

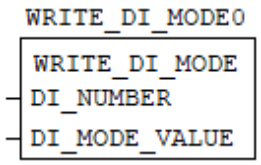


Рисунок Д.48 – FBD

Таблица Д. 32 – Переменные блока WRITE_DI_ MODE

Входа	**Тип**	**Описание**
DI_NUMBER	UINT	Номер канала дискретного входа ПЛКBRIC
DI_MODE_VALUE	UINT	Код подключенных функций данного канала
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(DI_NUMBER := *UINT*, DI_MODE_VALUE := *UINT*);

22 Блок WRITE_DO

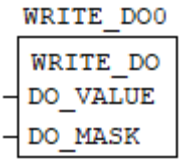


Рисунок Д.49 – FBD

Таблица Д.33 – Переменные блока WRITE_DO

Входа	**Тип**	**Описание**
DO_VALUE	UINT	Запись значения состояния дискретных выходов
DO_MASK	UINT	Запись маски, разрешающей изменять состояние дискретных выходов
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(DO_VALUE := *UINT*, DO_MASK := *UINT*);

23 Блок WRITE_ DO_SC

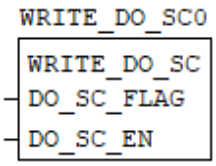


Рисунок Д.50 – FBD

Таблица Д.34 – Переменные блока WRITE_DO_SC

Входа	**Тип**	**Описание**
DO_SC_FLAG	UINT	Запись перечисления дискретных выходов, на которых сработала аппаратная защита от короткого замыкания (использовать только возможность сброса)
DO_SC_EN	UINT	Запись перечисления дискретных выходов, на которых сработала программная защита от короткого замыкания (использовать только возможность сброса)
Выхода	**Тип**	**Описание**

Форма записи на языке ST:

FB(DO_SC_FLAG := *UINT*, DO_SC_EN := *UINT*);

24 Блок WRITE_ DO_PWM_ FREQ

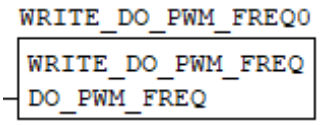


Таблица Д.35 – Переменные блока WRITE_DO_ PWM_FREQ

Входа	**Тип**	**Описание**
DO_PWM_FREQ	UINT	частота ШИМ подаваемая на каналы дискретных выходов, Гц
Выхода	**Тип**	**Описание**

Рисунок Д.51 – FBD

Форма записи на языке ST:

FB(DO_PWM_FREQ := *UINT*);

25 Блок WRITE_ DO_PWM_ CTRL

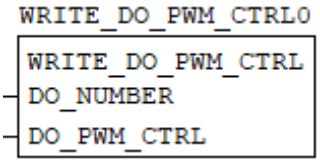


Таблица Д. 36 – Переменные блока WRITE_DO_ PWM_CTRL

Входа	**Тип**	**Описание**
DO_NUMBER	USINT	Номер канала дискретного выхода ПЛКBRIC
DO_PWM_CTRL	UINT	Указание скважности канала
Выхода	**Тип**	**Описание**

Рисунок Д.52 – FBD

Форма записи на языке ST:

FB(DO_NUMBER := *USINT*, DO_PWM_CTRL := *UINT*);

26 Блок STRUCT_ REAL_TIME

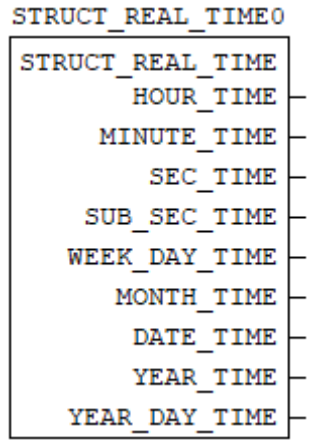


Рисунок Д.53 – FBD

Таблица Д.37 – Переменные блока STRUCT_REAL_ TIME

Входа	**Тип**	**Описание**
Выхода	**Тип**	**Описание**
HOUR_TIME	USINT	Текущий час реального времени внутреннего определения времени ПЛК
MINUTE_TIME	USINT	Текущая минута
SEC_TIME	USINT	Текущая секунда
SUB_SEC_TIME	USINT	Текущая миллисекунда
WEEK_DAY_TIME	USINT	Текущий день недели
MONTH_TIME	USINT	Текущий месяц
DATE_TIME	USINT	Текущий день месяца
YEAR_TIME	USINT	Текущий год
YEAR_DAY_TIME	UINT	Текущий день года

Форма записи на языке ST:

```
*FB*();
*USINT*:= *FB*.HOUR_TIME;
*USINT*:= *FB*.MINUTE_TIME;
*USINT*:= *FB*.SEC_TIME;
*USINT*:= *FB*.SUB_SEC_TIME;
*USINT*:= *FB*.WEEK_DAY_TIME;
*USINT*:= *FB*.MONTH_TIME;
*USINT*:= *FB*.DATE_TIME;
*USINT*:= *FB*.YEAR_TIME;
*UINT*:= *FB*.YEAR_DAY_TIME;
```

27 Блок UNIX_ TIME

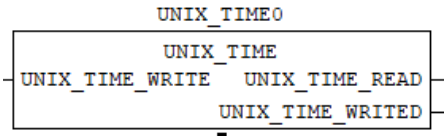


Рисунок Д.54 – FBD

Таблица Д.38 – Переменные блока UNIX_TIME

Входа	**Тип**	**Описание**
UNIX_TIME_WRITE	UDINT	Запись времени на ПЛК
Выхода	**Тип**	**Описание**
UNIX_TIME_READ	UDINT	Время до изменения
UNIX_TIME_WRITED	UDINT	Время после изменения

Форма записи на языке ST:

FB(UNIX_TIME_WRITE := *UDINT*);

UDINT:= *FB*.UNIX_TIME_READ;

UDINT:= *FB*.UNIX_TIME_WRITED;

28 Блок WRITE_ STRUCT_TIME

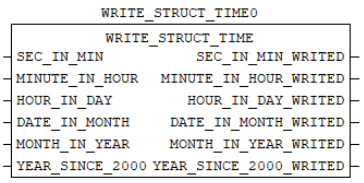


Рисунок Д.55 – FBD

Таблица Д.39 – Переменные блока UNIX_TIME

Входа	**Тип**	**Описание**
SEC_IN_MIN	USINT	Ввод секунд
MINUTE_IN_HOUR	USINT	Ввод минут
HOUR_IN_DAY	USINT	Ввод часов
DATE_IN_MONTH	USINT	Ввод дня месяца
MONTH_IN_YEAR	USINT	Ввод месяца года
YEAR_SINCE_2000	USINT	Ввод года
Выхода	**Тип**	**Описание**
SEC_IN_MIN_WRITED	USINT	Вывод присвоенных значений
MINUTE_IN_HOUR_WRITED	USINT	
HOUR_IN_DAY_WRITED	USINT	
DATE_IN_MONTH_WRITED	USINT	
MONTH_IN_YEAR_WRITED	USINT	
YEAR_SINCE_2000_WRITED	USINT	

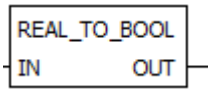
Форма записи на языке ST:

```
ADDFB_25(SEC_IN_MIN := *USINT*, MINUTE_IN_HOUR := *USINT*,
HOUR_IN_DAY := *USINT*, DATE_IN_MONTH := *USINT*,
MONTH_IN_YEAR := *USINT*, YEAR_SINCE_2000 := *USINT*);
*USINT*:= *FB*.SEC_IN_MIN_WRITED;
*USINT*:= *FB*.MINUTE_IN_HOUR_WRITED;
*USINT*:= *FB*.HOUR_IN_DAY_WRITED;
*USINT*:= *FB*.DATE_IN_MONTH_WRITED;
*USINT*:= *FB*.MONTH_IN_YEAR_WRITED;
*USINT*:= *FB*.YEAR_SINCE_2000_WRITED;
```

Преобразование типов:

1 Блок*__*_ТО_*__*

Таблица Д.40 – Переменные блока *__*_ТО_*__*



Входа	Тип	Описание
IN	…	Преобразуемая информация
Выхода	**Тип**	**Описание**
OUT	…	Преобразованная информация

Рисунок Д.56 – FBD

Форма записи на языке ST:

— := *_*_ТО_*_*(*—*);

Рисунок Д.57 – Блок–схема

Математические функции:

1 Блок ABS

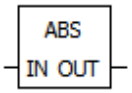


Рисунок Д.58 – FBD

Таблица Д.41 – Переменные блока ABS

Входа	**Тип**	**Описание**
IN	ANY_NUM	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_NUM	Выход (модуль числа)

Форма записи на языке ST:

*****ANY_NUM*****:= ABS(*****ANY_NUM*****);

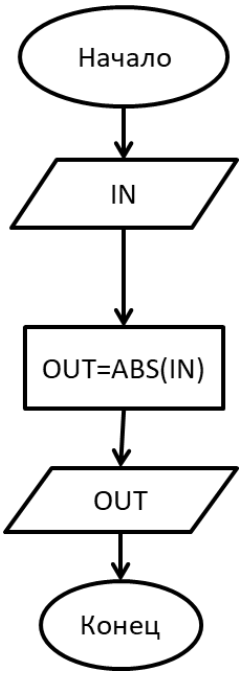


Рисунок Д.59 – Блок–схема

2 Блок SQRT

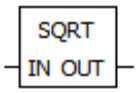


Рисунок Д.60 – FBD

Таблица Д.42 – Переменные блока SQRT

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (корень числа)

Форма записи на языке ST:

ANY_REAL:= SQRT(*ANY_REAL*);

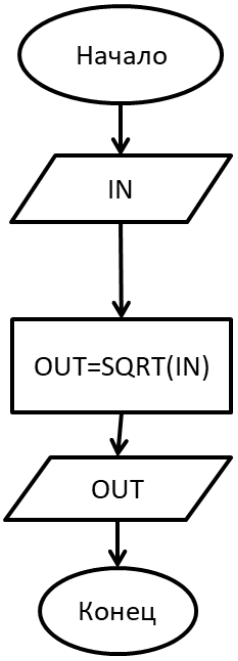


Рисунок Д.61 – Блок–схема

3 Блок LN

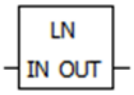


Рисунок Д.62 – FBD

Таблица Д. 43 – Переменные блока LN

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (натуральный логарифм числа)

Форма записи на языке ST:

*****ANY_REAL*****:= LN(*****ANY_REAL*****);

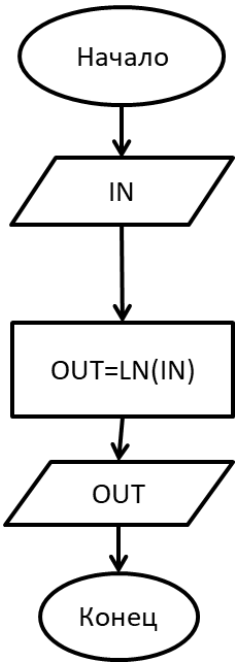


Рисунок Д.63 – Блок–схема

4 Блок LOG

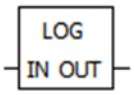


Рисунок Д.64 – FBD

Таблица Д.44 – Переменные блока LOG

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (логарифм числа)

Форма записи на языке ST:

ANY_REAL:= LOG(*ANY_REAL*);

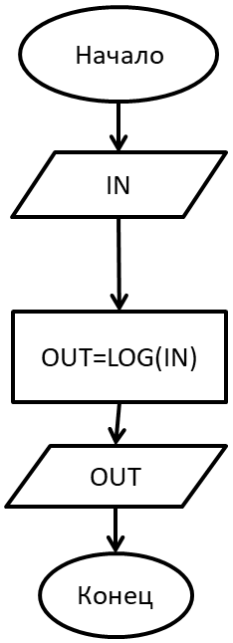


Рисунок Д.65 – Блок–схема

5 Блок EXP

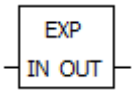


Рисунок Д.66 – FBD

Таблица Д.45 – Переменные блока EXP

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (экспонента числа)

Форма записи на языке ST:

ANY_REAL:= EXP(*ANY_REAL*);

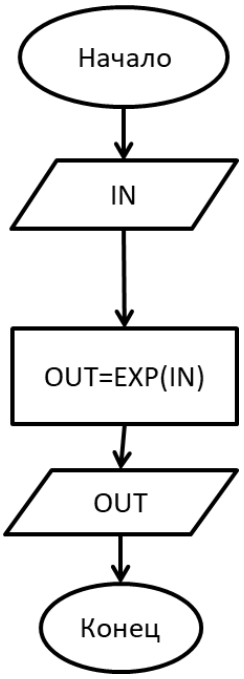


Рисунок Д.67 – Блок–схема

6 Блок SIN

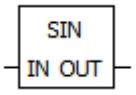


Рисунок Д.68 – FBD

Таблица Д.46 – Переменные блока SIN

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (синус числа)

Форма записи на языке ST:

ANY_REAL:= SIN(*ANY_REAL*);

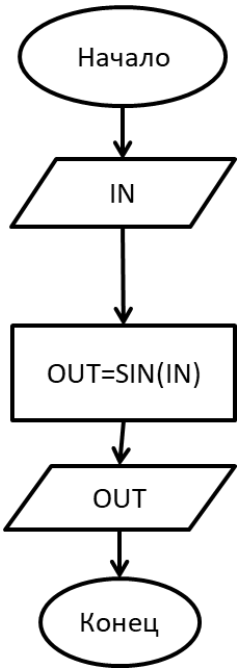


Рисунок Д.69 – Блок–схема

7 Блок COS

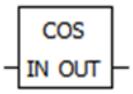


Рисунок Д.70 – FBD

Таблица Д.47 – Переменные блока COS

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (косинус числа)

Форма записи на языке ST:

ANY_REAL:= COS(*ANY_REAL*);

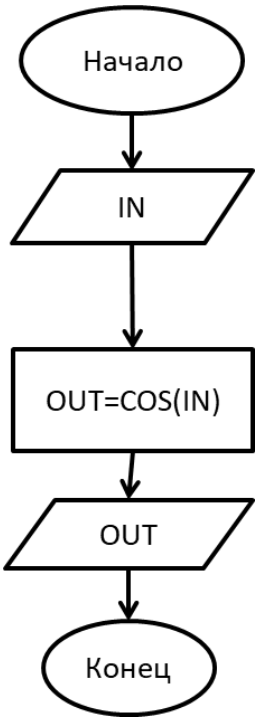


Рисунок Д.71 – Блок–схема

8 Блок TAN

Рисунок Д.72 – FBD

Таблица Д.48 – Переменные блока TAN

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (тангенс числа)

Форма записи на языке ST:

ANY_REAL:= TAN(*ANY_REAL*);

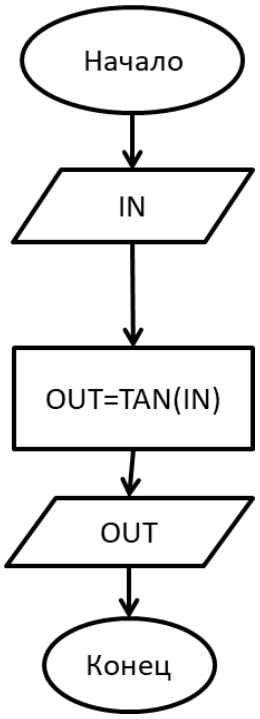


Рисунок Д.73 – Блок–схема

9 Блок ASIN

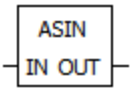


Рисунок Д.74 – FBD

Таблица Д.49 – Переменные блока ASIN

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (арксинус числа)

Форма записи на языке ST:

ANY_REAL:= ASIN(*ANY_REAL*);

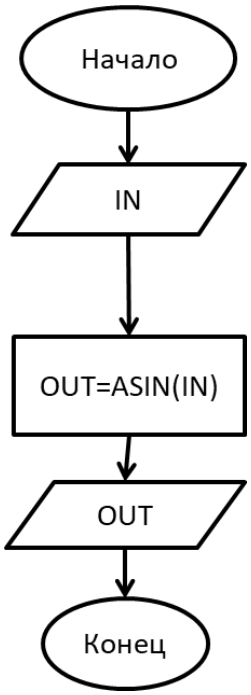


Рисунок Д.75 – Блок–схема

10 Блок ACOS

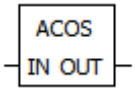


Рисунок Д.76 – FBD

Таблица Д.50 – Переменные блока ACOS

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (арккосинус числа)

Форма записи на языке ST:

ANY_REAL:= ACOS(*ANY_REAL*);

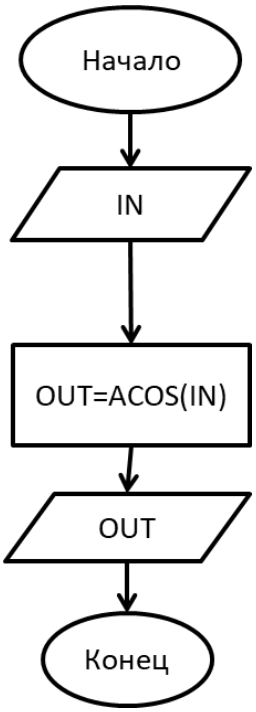


Рисунок Д.77 – Блок–схема

11 Блок ATAN

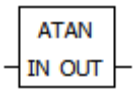


Рисунок Д.78 – FBD

Таблица Д.51 – Переменные блока ATAN

Входа	**Тип**	**Описание**
IN	ANY_REAL	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_REAL	Выход (арктангенс числа)

Форма записи на языке ST:

ANY_REAL:= ATAN(*ANY_REAL*);

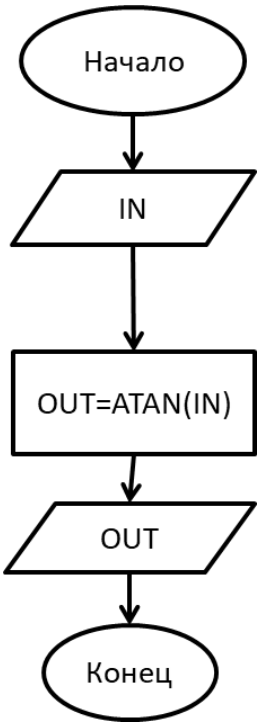


Рисунок Д.79 – Блок–схема

Математика:

1 Блок ADD

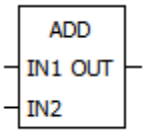


Рисунок Д.80 – FBD

Таблица Д.52 – Переменные блока ADD

Входа	**Тип**	**Описание**
IN1	ANY_NUM	Вход
IN2	ANY_NUM	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_NUM	Выход(сложение)

Форма записи на языке ST:

ANY_NUM:= ADD(*ANY_NUM*, *ANY_NUM*);

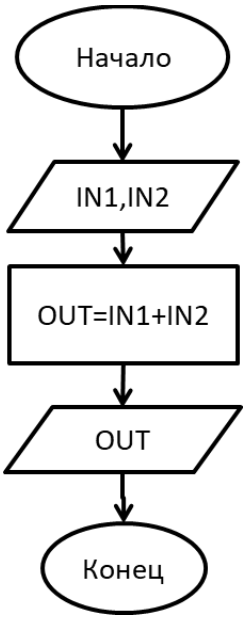


Рисунок Д.81 – Блок–схема

2 Блок MUL

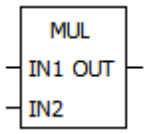


Рисунок Д.82 – FBD

Таблица Д.53 – Переменные блока MUL

Входа	**Тип**	**Описание**
IN1	ANY_NUM	Вход
IN2	ANY_NUM	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_NUM	Выход(умножение)

Форма записи на языке ST:
ANY_NUM:= MUL(*ANY_NUM*, *ANY_NUM*);

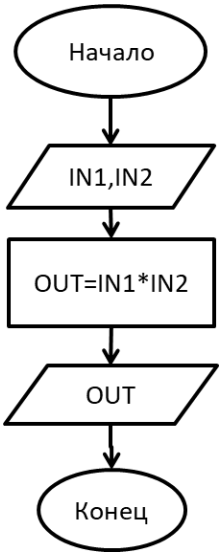


Рисунок Д.83 – Блок–схема

3 Блок SUB

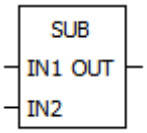


Рисунок Д.84 – FBD

Таблица Д.54 – Переменные блока SUB

Входа	**Тип**	**Описание**
IN1	ANY_NUM	Вход
IN2	ANY_NUM	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_NUM	Выход(разность)

Форма записи на языке ST:
ANY_NUM:= SUB (*ANY_NUM*, *ANY_NUM*);

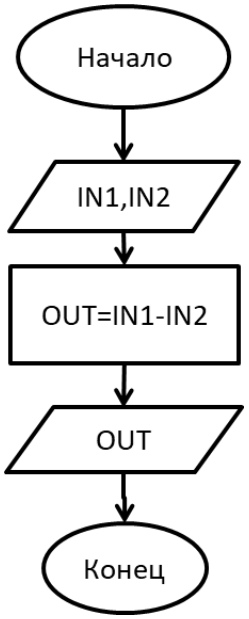


Рисунок Д.85 – Блок–схема

4 Блок DIV

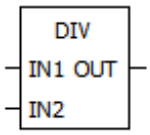


Рисунок Д.86 – FBD

Таблица Д.55 – Переменные блока DIV

Входа	**Тип**	**Описание**
IN1	ANY_NUM	Вход (делитель)
IN2	ANY_NUM	Вход(делитель)
Выхода	**Тип**	**Описание**
OUT	ANY_NUM	Выход (деление)

Форма записи на языке ST:
ANY_NUM:= DIV (*ANY_NUM*, *ANY_NUM*);

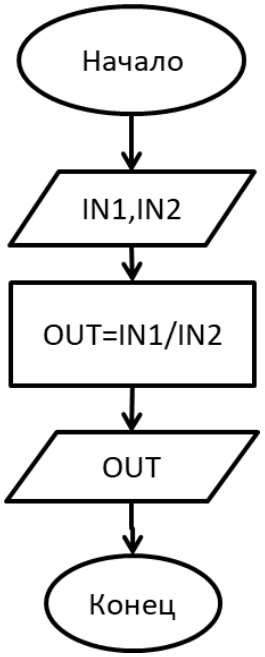


Рисунок Д.87 – Блок–схема

5 Блок MOD

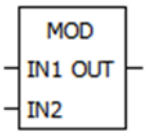


Рисунок Д.88 – FBD

Таблица Д.56 – Переменные блока MOD

Входа	**Тип**	**Описание**
IN1	ANY_INT	Вход (делимое)
IN2	ANY_INT	Вход(делитель)
Выхода	**Тип**	**Описание**
OUT	ANY_INT	Выход (остаток от деления)

Форма записи на языке ST:
ANY_INT:= MOD (*ANY_INT*, *ANY_INT*);

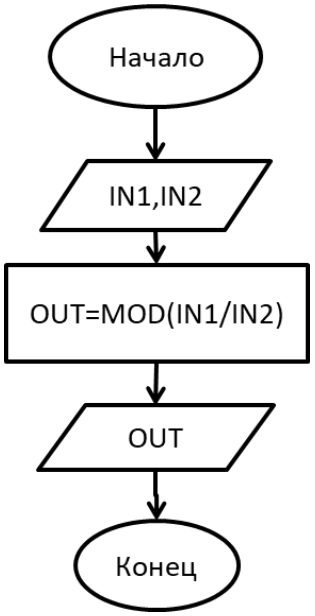


Рисунок Д.89 – Блок–схема

6 Блок EXPT

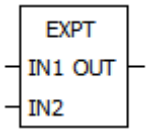


Рисунок Д.90 – FBD

Таблица Д.57 – Переменные блока EXPT

Входа	**Тип**	**Описание**
IN1	ANY_REAL	Вход
IN2	ANY_NUM	Вход(степень)
Выхода	**Тип**	**Описание**
OUT	ANY_REAL[1]	Выход (число в степени)

Форма записи на языке ST:
ANY_REAL:= EXPT(*ANY_REAL*,* ANY_NUM*);

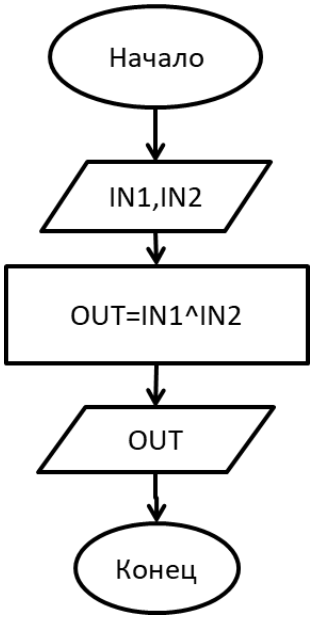


Рисунок Д.91– Блок–схема[11]

[11] OUT пип переменной Выхода должен соответствовать типу переменной входа

7 Блок MOVE

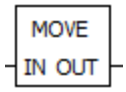


Рисунок Д.92 – FBD

Таблица Д. 58 – Переменные блока MOVE

Входа	**Тип**	**Описание**
IN1	ANY	Вход
Выхода	**Тип**	**Описание**
OUT	ANY	Выход (присваивание)

Форма записи на языке ST:

ANY:= MOVE(*ANY*);

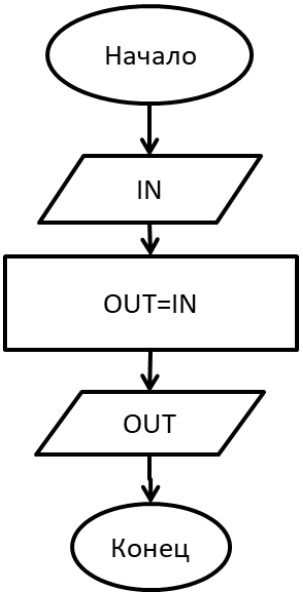


Рисунок Д.93 – Блок–схема

Сдвиговые операции:

1 Блок SHL

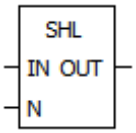


Рисунок Д.94 – FBD

Таблица Д.59 – Переменные блока SHL

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_INT	Количество сдвигов влево
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= SHL(*ANY_BIT*, *ANY_INT*);

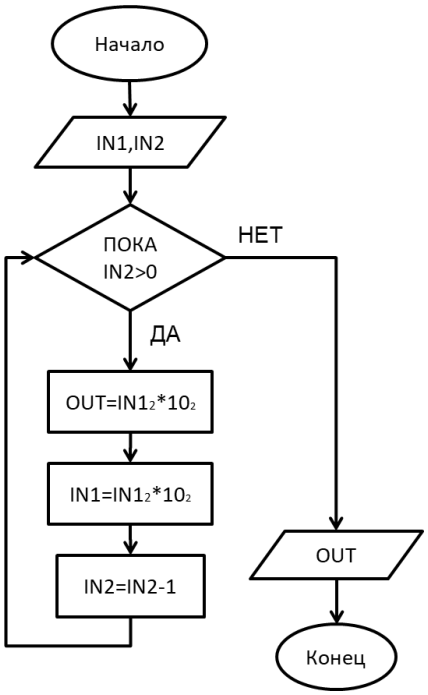


Рисунок Д.95 – Блок–схема[12]

[12]IN1 тип переменной Выхода должен соответствовать типу переменной входа;
OUT нет возможности использования переменных типа BOOL

2 Блок SHR

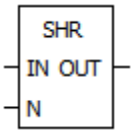


Рисунок Д.96 – FBD

Таблица Д. 60 – Переменные блока SHR

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_INT	Количество сдвигов вправо
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= SHR(*ANY_BIT*, *ANY_INT*);

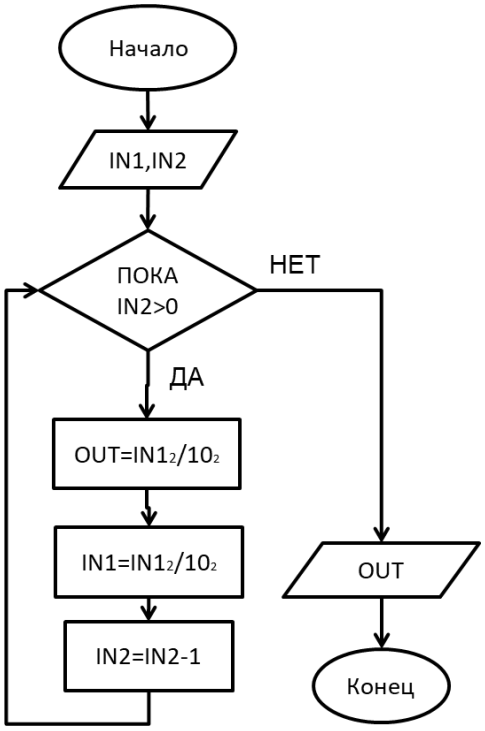


Рисунок Д.97 – Блок–схема[13]

[13] IN1 тип переменной Выхода должен соответствовать типу переменной входа;
OUT нет возможности использования переменных типа BOOL

3 Блок ROR

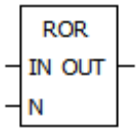


Рисунок Д.98 – FBD

Таблица Д. 61 – Переменные блока ROR

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_INT	Количество сдвигов вправо
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= ROR(*ANY_BIT*, *ANY_INT*);

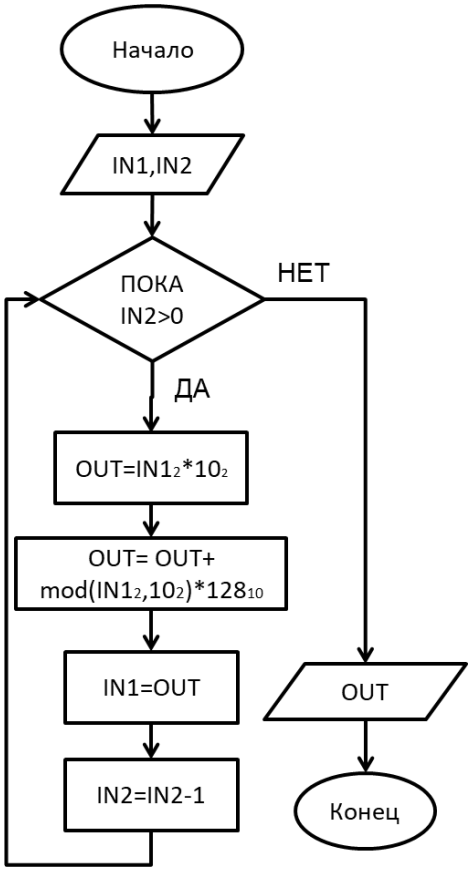


Рисунок Д.99 – Блок–схема[14]

[14] IN1 пип переменной Выхода должен соответствовать типу переменной входа;
OUT нет возможности использования переменных типа BOOL

4 Блок ROL

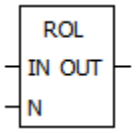


Рисунок Д.100 – FBD

Таблица Д.62 – Переменные блока ROL

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_INT	Количество сдвигов влево
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= ROL(*ANY_BIT*, *ANY_INT*);

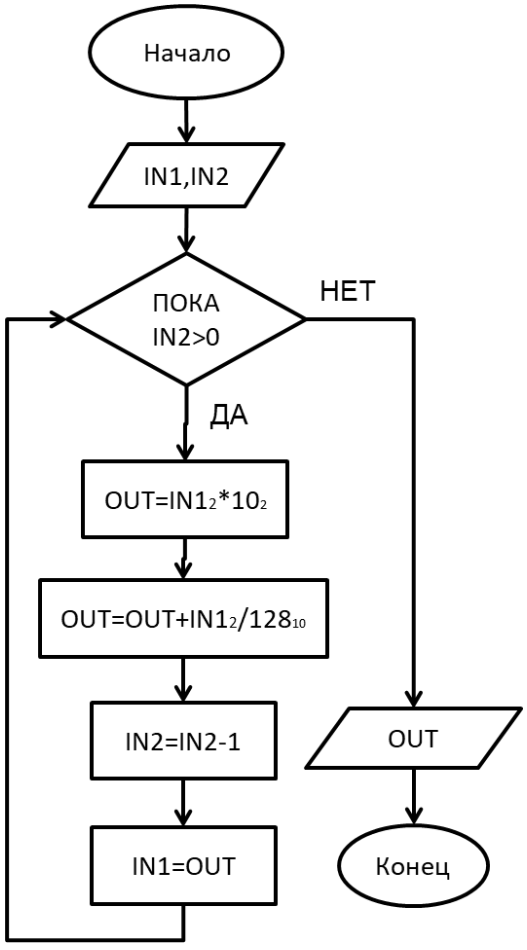


Рисунок Д.101 – Блок–схема[15]

[15] IN1 тип переменной Выхода должен соответствовать типу переменной входа;
OUT нет возможности использования переменных типа BOOL

Битовые операции:

1 Блок AND

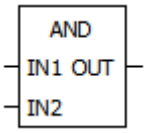


Рисунок Д.102 – FBD

Таблица Д. 63 – Переменные блока AND

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_BIT	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= AND(*ANY_BIT*, *ANY_BIT*);

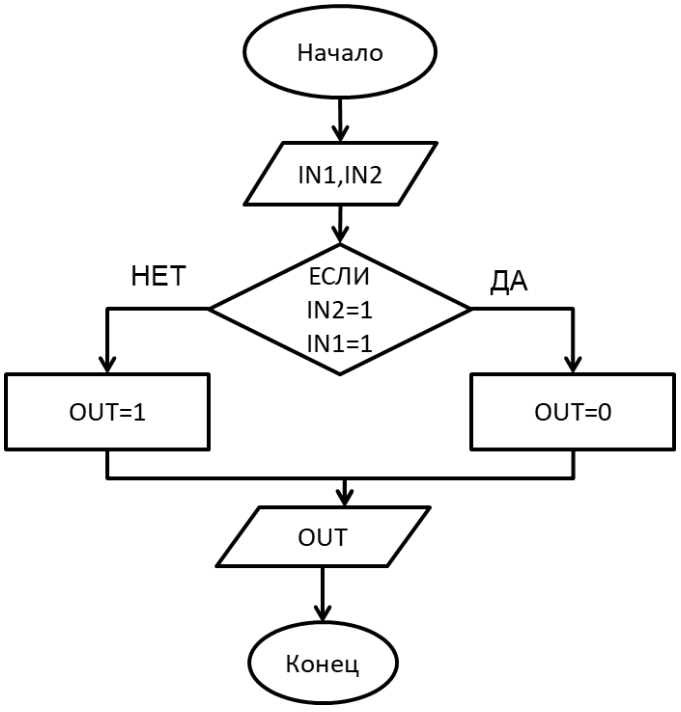


Рисунок Д.103 – Блок–схема

2 Блок OR

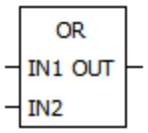


Рисунок Д.104 – FBD

Таблица Д. 64 – Переменные блока OR

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_BIT	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= OR(*ANY_BIT*, *ANY_BIT*);

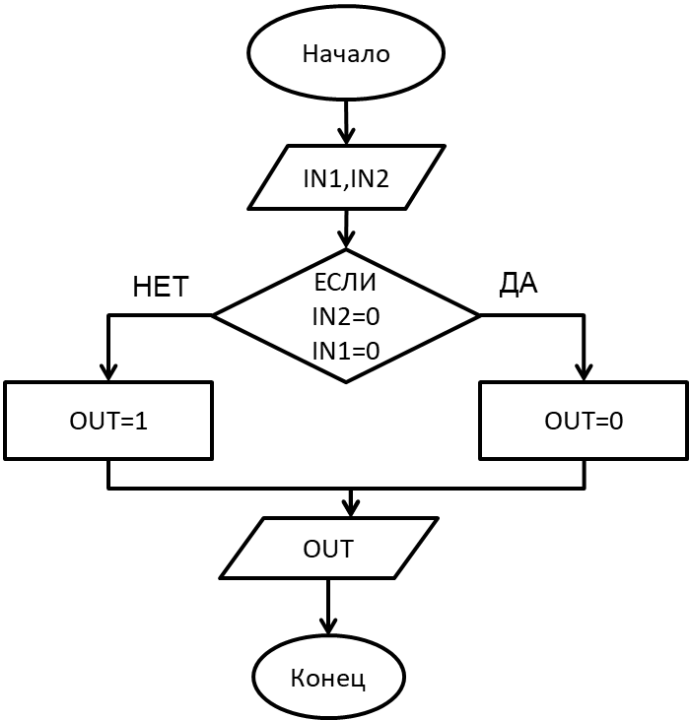


Рисунок Д.105 – Блок–схема

3 Блок XOR

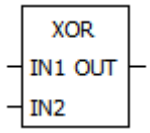


Рисунок Д.106 – FBD

Таблица Д. 65 – Переменные блока XOR

Входа	**Тип**	**Описание**
IN1	ANY_BIT	Вход
IN2	ANY_BIT	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход

Форма записи на языке ST:

ANY_BIT:= XOR(*ANY_BIT*, *ANY_BIT*);

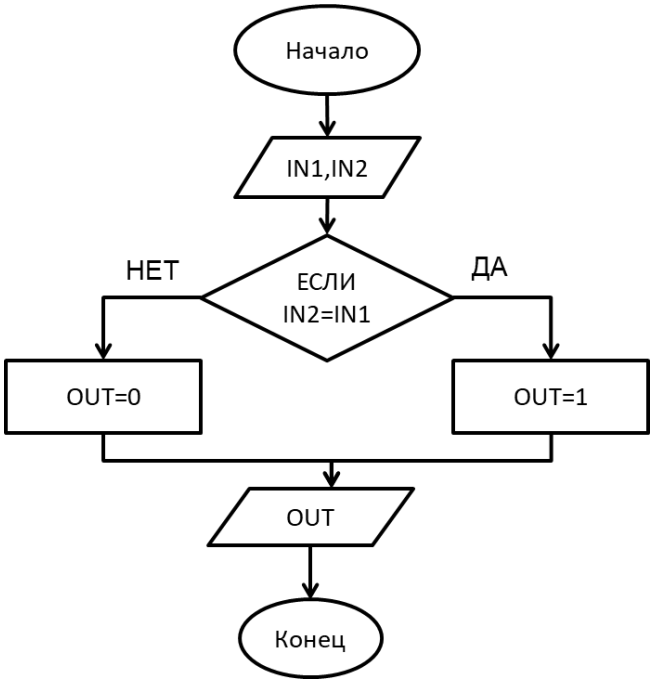


Рисунок Д.107 – Блок–схема

4 Блок NOT

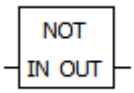


Рисунок Д.108 – FBD

Таблица Д. 66 – Переменные блока NOT

Входа	**Тип**	**Описание**
IN	ANY_BIT	Вход
Выхода	**Тип**	**Описание**
OUT	ANY_BIT	Выход (инверсия)

Форма записи на языке ST:
ANY_BIT:= NOT(*ANY_BIT*, *ANY_BIT*);

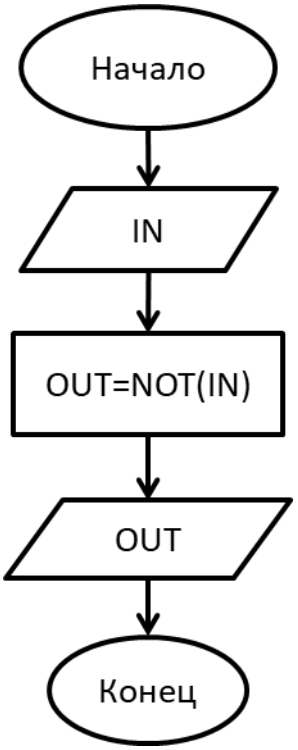


Рисунок Д.109 – Блок–схема

Выбор:

1 Блок SEL

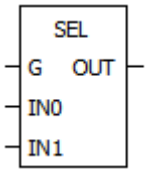


Рисунок Д.110 – FBD

Таблица Д. 67 – Переменные блока SEL

Входа	**Тип**	**Описание**
G	BOOL	Номер входа
IN0	ANY	Вход 1
IN1	ANY	Вход 2
Выхода	**Тип**	**Описание**
OUT	ANY	Выход

Форма записи на языке ST:

ANY:= SEL(*BOOL*, *ANY*, *ANY*);

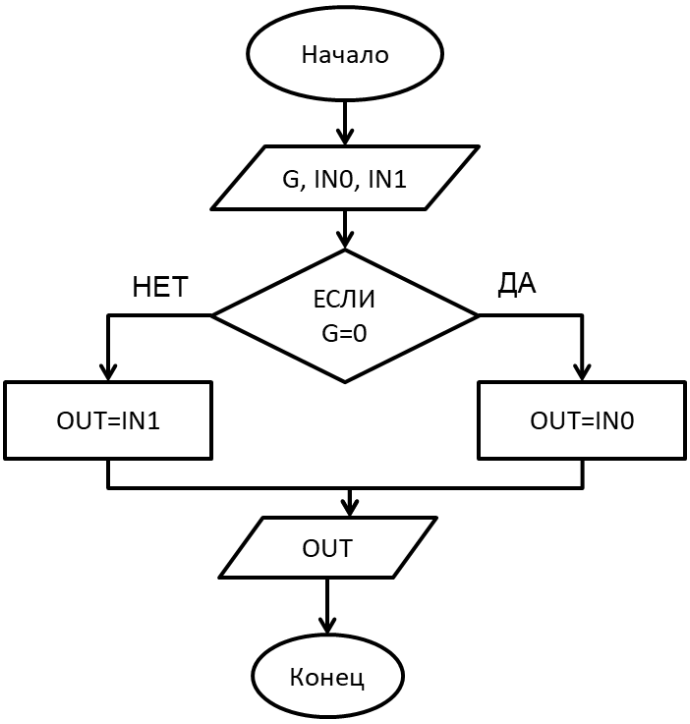


Рисунок Д.111 – Блок–схема

2 Блок MAX

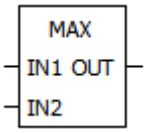


Рисунок Д.112 – FBD

Таблица Д.68 – Переменные блока MAX

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	…	Вход …
Выхода	**Тип**	**Описание**
OUT	ANY	Выход

Форма записи на языке ST:

ANY:= MAX(*ANY*, *ANY*);

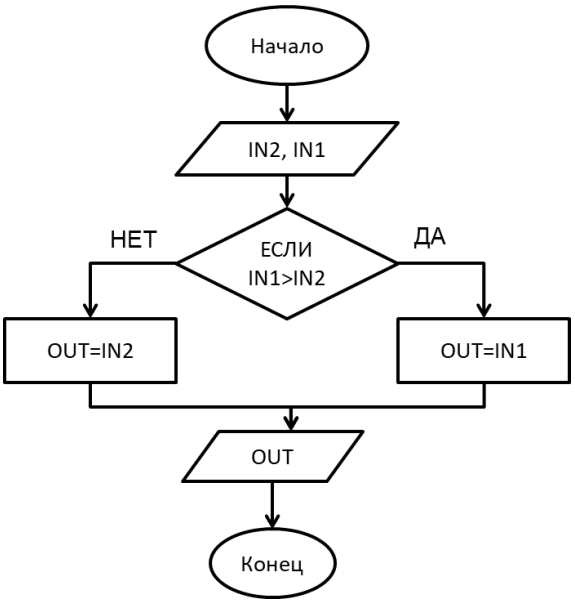


Рисунок Д.113 – Блок–схема

3 БлокMIN

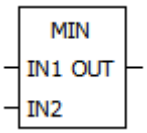


Рисунок Д.114 – FBD

Таблица Д.69 – Переменные блока MIN

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	…	Вход …
Выхода	**Тип**	**Описание**
OUT	ANY	Выход

Форма записи на языке ST:

ANY:= MIN(*ANY*, *ANY*);

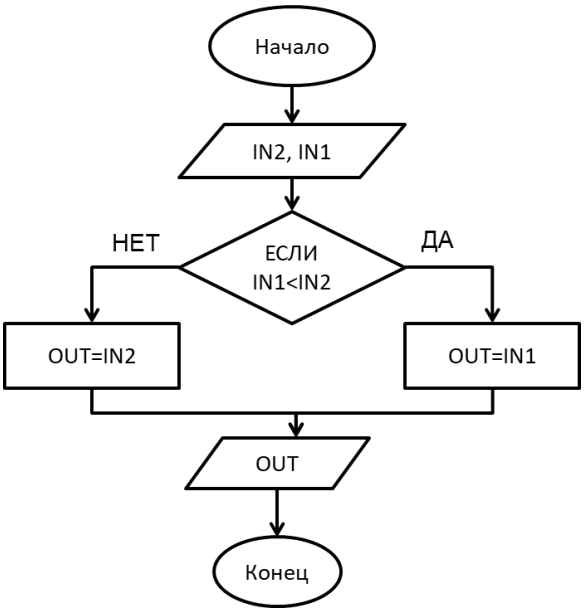


Рисунок Д.115 – Блок–схема

4 БлокLIMIT

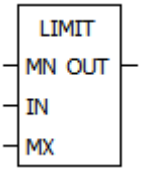


Рисунок Д.116 – FBD

Таблица Д.70 – Переменные блока LMIT

Входа	**Тип**	**Описание**
MN	ANY	Вход (нижний предел)
IN	ANY	Вход
MX	ANY	Вход (верхний предел)
Выхода	**Тип**	**Описание**
OUT	ANY	Выход

Форма записи на языке ST:

ANY:= LIMIT(*ANY*, *ANY*, *ANY*);

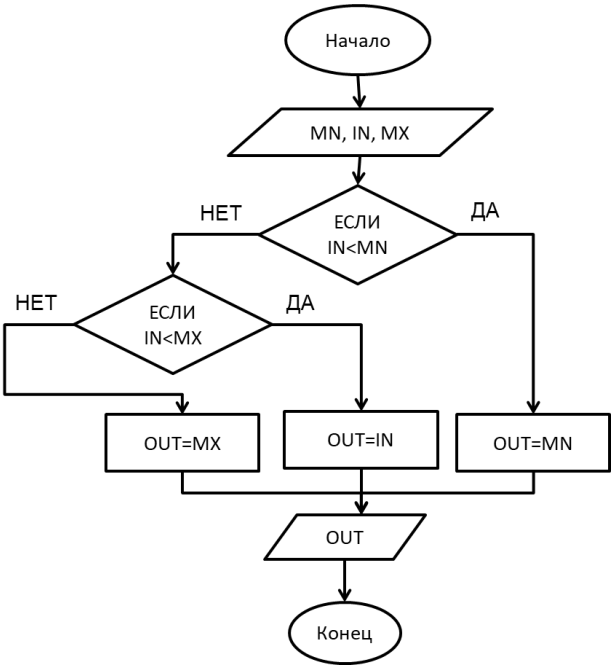


Рисунок Д.117 – Блок–схема

5 БлокMUX

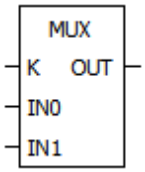


Рисунок Д.118 – FBD

Таблица Д.71 – Переменные блока MUX

Входа	**Тип**	**Описание**
K	ANY_INT	Переключатель
IN0	ANY	Вход 1
IN1	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	ANY	Выход

Форма записи на языке ST:

ANY:= MUX(*ANY*, *ANY*, *ANY*);

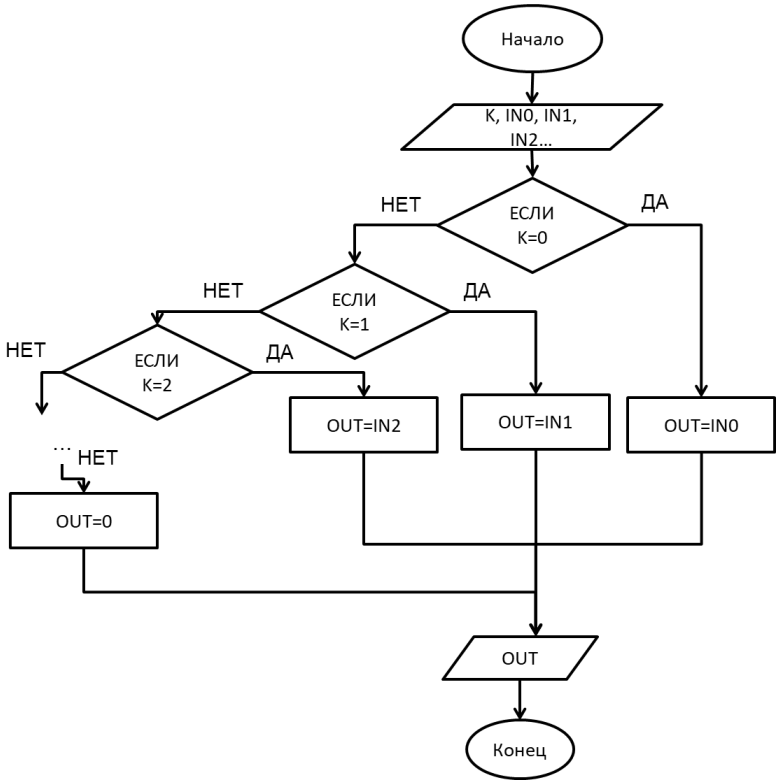


Рисунок Д.119 – Блок–схема

Сравнение:

1 Блок GT

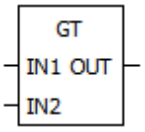


Рисунок Д.120 – FBD

Таблица Д.72 – Переменные блока GT

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	BOOL	Выход

Форма записи на языке ST:

BOOL:= GT(*ANY*, *ANY*);

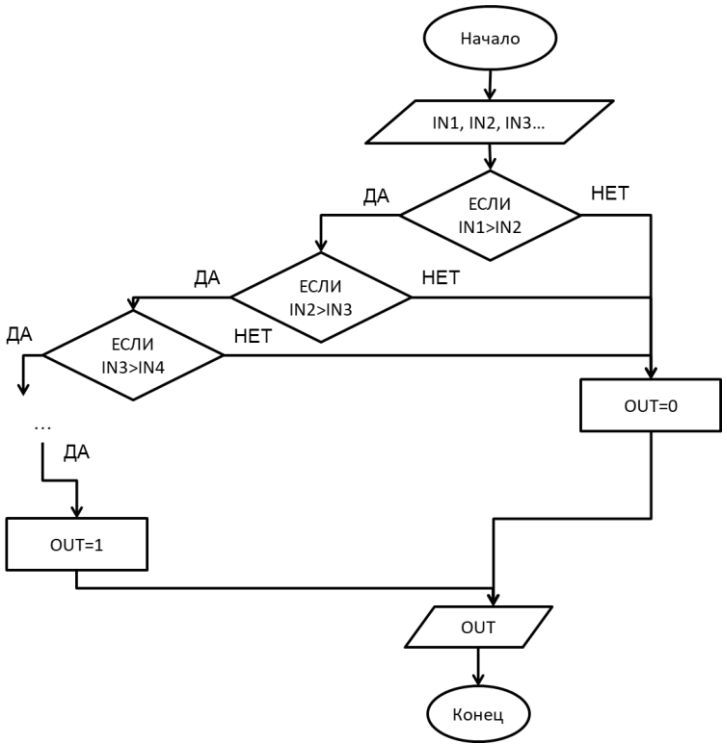


Рисунок Д.121 – Блок–схема

2 Блок GE

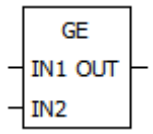


Рисунок Д.122 – FBD

Таблица Д.73 – Переменные блока GE

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	BOOL	Выход

Форма записи на языке ST:

BOOL:= GE(*ANY*, *ANY*);

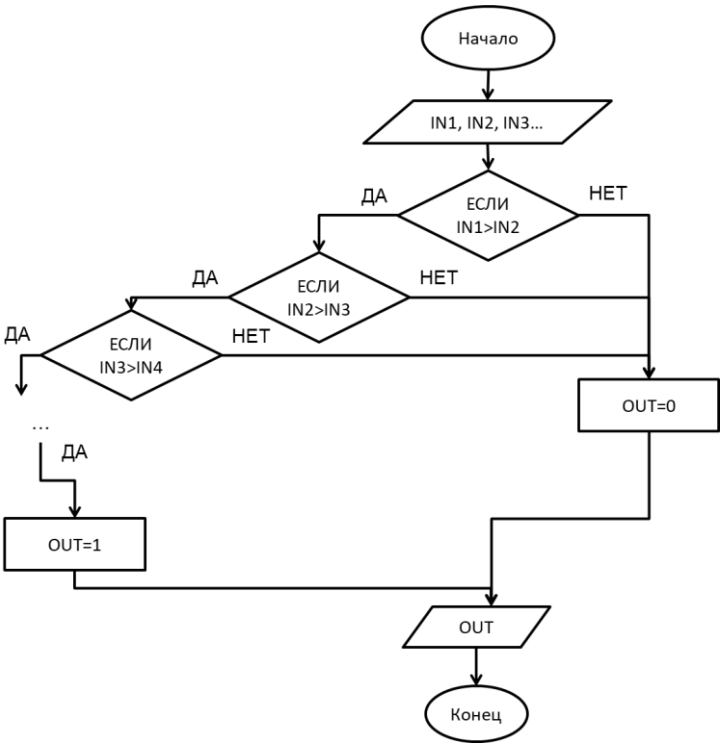


Рисунок Д.123 – Блок–схема

3 Блок EQ

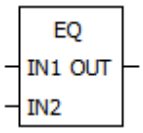


Рисунок Д.124 – FBD

Таблица Д.74 – Переменные блока EQ

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	BOOL	Выход

Форма записи на языке ST:

BOOL:= EQ(*ANY*, *ANY*);

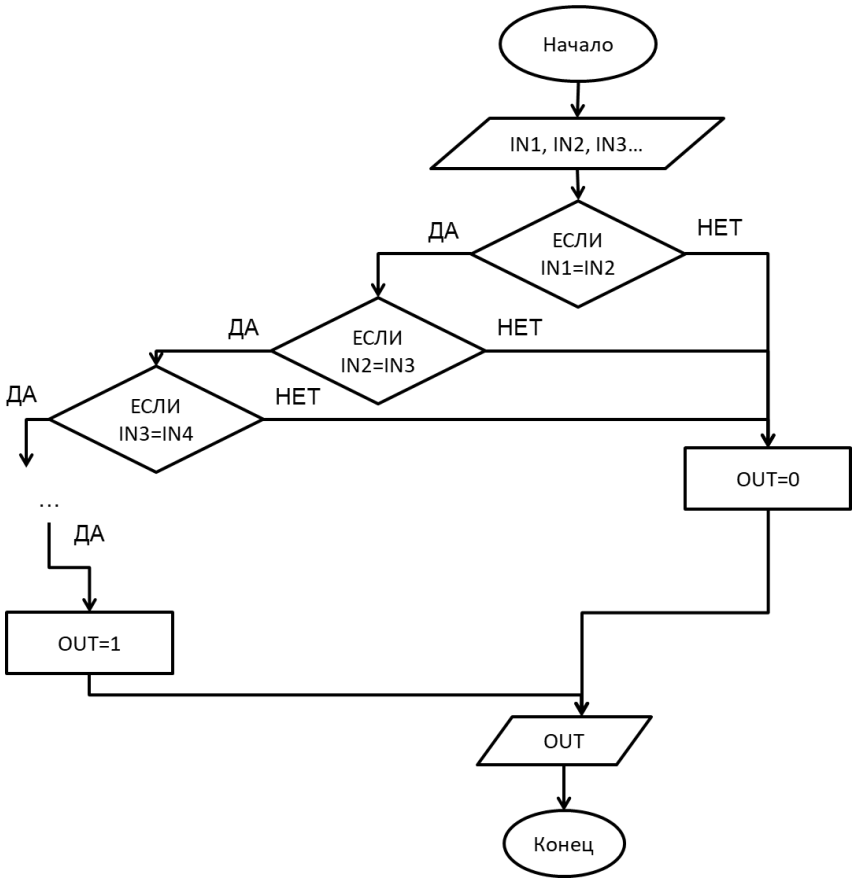


Рисунок Д.125 – Блок–схема

4 Блок LT

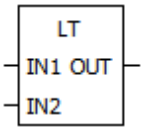


Рисунок Д.126 – FBD

Таблица Д.75 – Переменные блока LT

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	BOOL	Выход

Форма записи на языке ST:

BOOL:= LT(*ANY*, *ANY*);

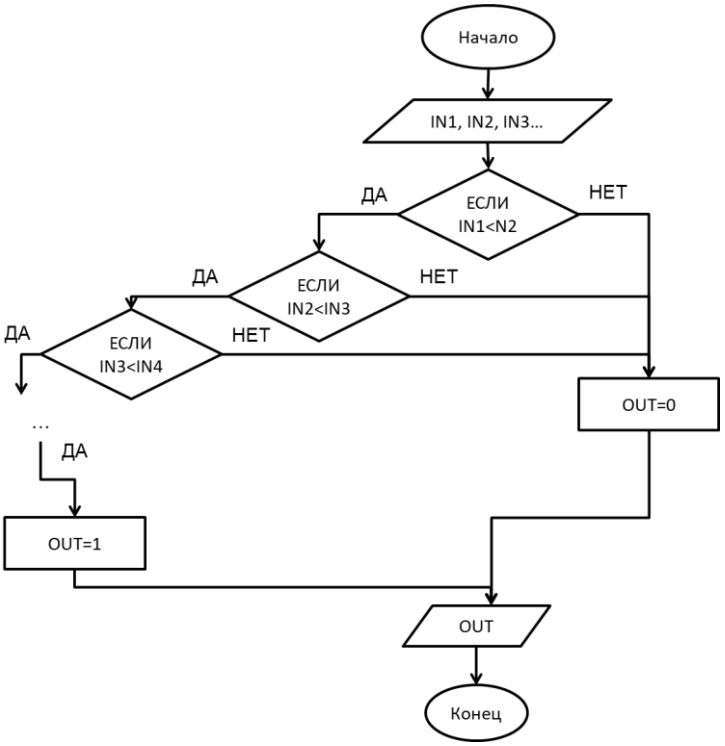


Рисунок Д.127 – Блок–схема

5 Блок LE

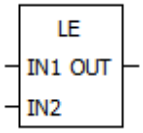


Рисунок Д.128 – FBD

Таблица Д.76 – Переменные блока LE

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	BOOL	Выход

Форма записи на языке ST:

BOOL:= LE(*ANY*, *ANY*);

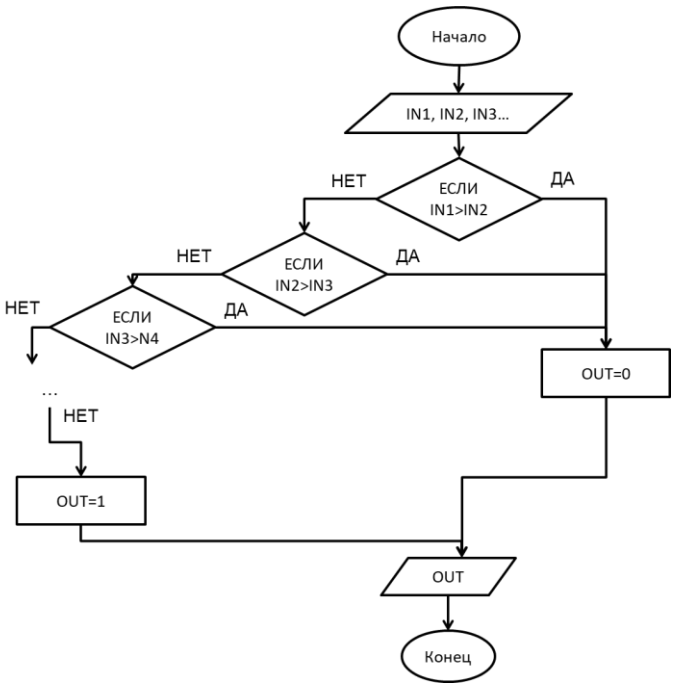


Рисунок Д.129 – Блок–схема

6 Блок NE

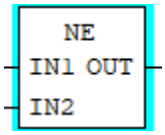


Рисунок Д.130 – FBD

Таблица Д.77 – Переменные блока NE

Входа	**Тип**	**Описание**
IN1	ANY	Вход 1
IN2	ANY	Вход 2
…	ANY	Вход …
Выхода	**Тип**	**Описание**
OUT	BOOL	Выход

Форма записи на языке ST:

BOOL:= NE(*ANY*, *ANY*);

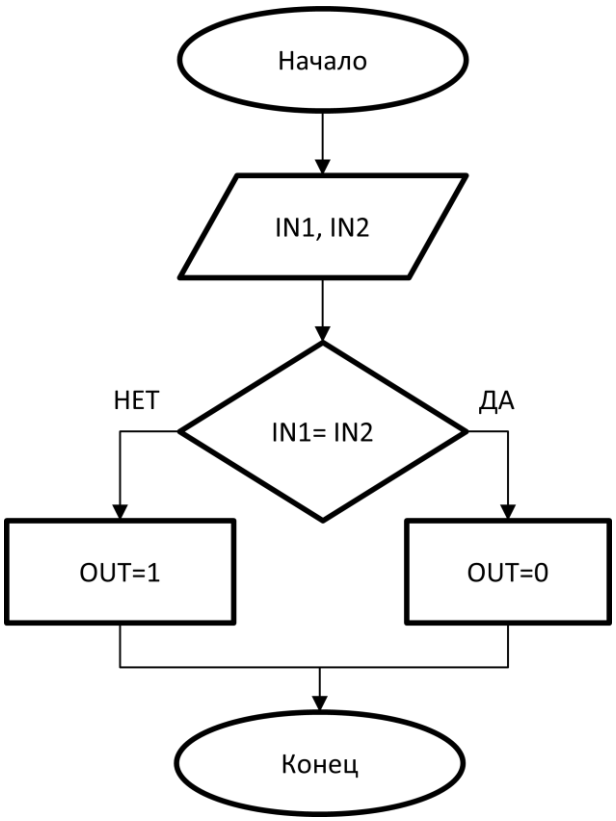


Рисунок Д.131 – Блок–схема

ПРИЛОЖЕНИЕ Е. КОМБИНАЦИИ БЫСТРОГО ВЫЗОВА КОМАНД ИСР BEREMIZ

Часть операций, выполняемых с помощью выбора определённого пункта меню мышью, могут быть исполнены с помощью «горячей клавиши». Далее будет подробно описаны «горячие клавиши».

Меню «Файл» предназначено для работы с проектом и предоставляет следующие пункты:

- «Новый» – создание нового проекта (CTRL + N);
- «Открыть» – открытие существующего проекта (CTRL + O);
- «Сохранить» – сохранение текущего проекта пункт (CTRL + S);
- «Сохранить как» – сохранение текущего проекта в папку отличную от той, в которой он сохранён на данный момент (CTRL + SHIFT + S);
- «Закрыть вкладку» – закрытие активной вкладки для открытого проекта (CTRL + W);
- «Закрыть проект» – закрыть открытый проект (CTRL + SHIFT + W);
- «Параметры страницы» – настройка параметров страницы для печати на принтере активной программы, представленной в виде диаграммы (CTRL + ALT + P);
- «Просмотр» – предварительный просмотр перед печатью на принтере активной программы (CTRL + SHIFT + P);
- «Печать» – печать на принтере активной программы (CTRL + P);
- «Выход» – закрытие текущего проекта и выход из программы ИСР Beremiz (CTRL+ Q);
- «Отменить» – отмена последнего действия в редакторе (CTRL + Z);
- «Повторить» повтор отменённого действия в редакторе (CTRL + Y);
- «Вырезать» – удалить в буфер обмена выделенные элементы в редакторе (CTRL + X);

- «Копировать» – копировать в буфер обмена выделенные элементы в редакторе (CTRL + C);
- «Вставить» – вставить из буфера обмена находящиеся там элементы в редактор (CTRL + V);
- «Поиск в проекте» – вызов диалога поиска данных в проекте (CTRL + SHIFT + F);
- «Выделить всё» – выделение всех элементов в активной вкладке редактора (CTRL +A);
- «Обновить» – обновление данных и снятие выделения в редакторе (CTRL + R);
- «Очистить ошибки» – очистка указателей ошибок в редакторе (CTRL + K).

ПРИЛОЖЕНИЕ Ж. ЧАСТО ВСТРЕЧАЕМЫЕ ОШИБКИ ПРИ НАПИСАНИИ ПРОЕКТА В ИСР BEREMIZ

Причиной возникновения ошибки, указанной на рисунке Ж.1 является использование типа переменной (TIME, DATE, TOD, DT, STRING) не поддерживаемой в классе «внешняя» для платформы «Sofi». Для исправления ошибки необходимо перевести их в локальный класс переменных и при необходимости изменения/считывания их использовать преобразователи типов переменных.

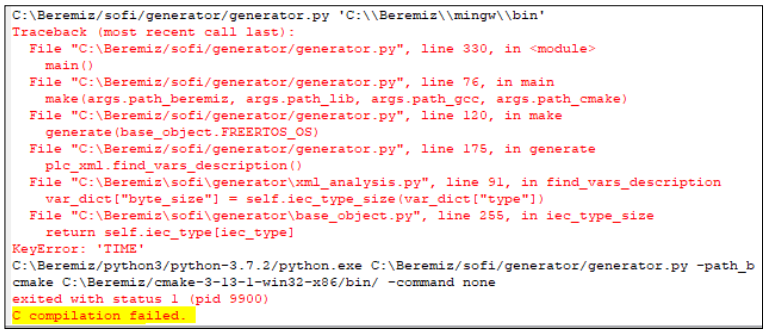


Рисунок Ж.1

Причиной возникновения ошибки, указанной на рисунке Ж.2 является использование в программном модуле переменных внешнего класса не указанных на главной странице.

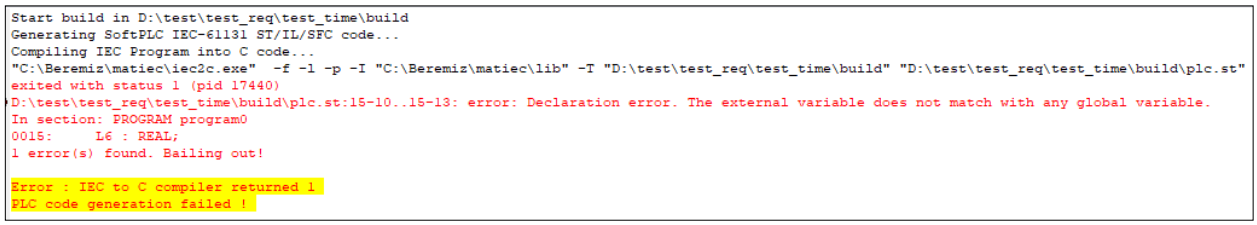


Рисунок Ж.2

Причиной возникновения ошибки, указанной на рисунке Ж.3 является использование функциональных блоков, поддерживаемых только платформой «Sofi». В разделе Config необходимо поменять платформу на «Sofi».

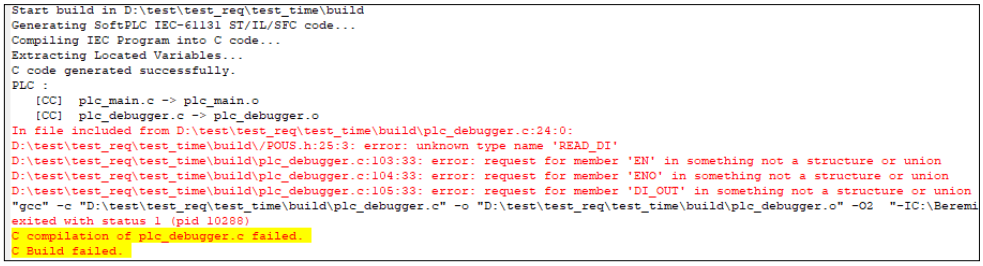


Рисунок Ж.3

Причиной возникновения ошибки, указанной на рисунке Ж.4 является отсутствие подключенной переменной на входе в функциональный блок.

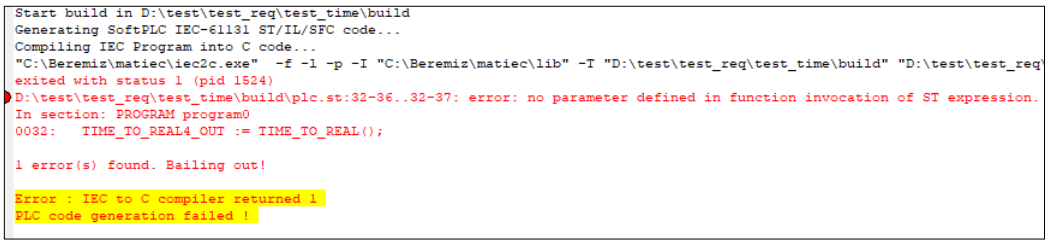


Рисунок Ж.4

Причиной возникновения ошибки, указанной на рисунке Ж.5 является неверный адрес на несуществующий путь в поддержке Modbus или несоответствие типа указанного в адресе переменной.

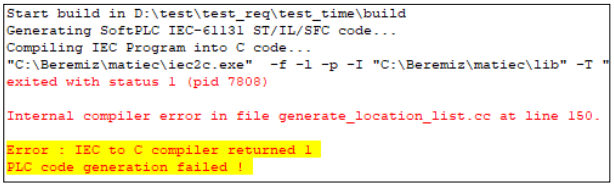


Рисунок Ж.5

Причиной возникновения ошибки, указанной на рисунке Ж.6 является неверное название папки проекта (наличие пробела).

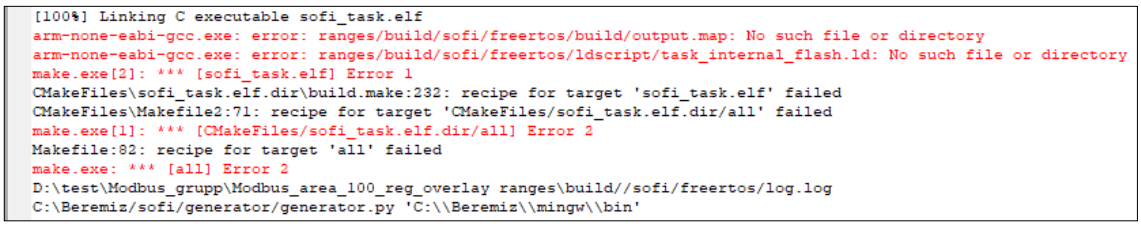


Рисунок Ж.6

Причиной возникновения ошибки, указанной на рисунке Ж.7 является отсутствие task в resource и не все программы проинициализированы в Instances.

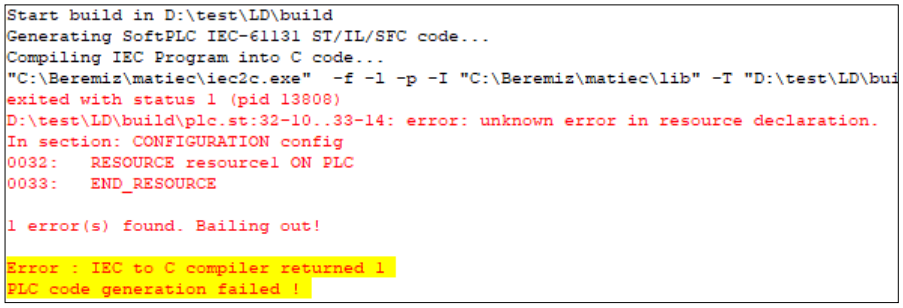


Рисунок Ж.7