УТВЕРЖДАЮ

Директор ООО «Континент ЭТС»

_____А.А. Алексеев

«_26_» ___02___2025 г.

Интегрированная система разработки и исполнения алгоритмов для программируемых логических контроллеров «ПРОКОНТ»

Руководство пользователя по установке и эксплуатации

СОГЛАСОВАНО:

Начальник отдела АСУТП

_____А.Н. Вовк

«_26_» ___02___2025 г.

Зам. начальника отдела АСУТП

_____ А.С. Морозов

«_26_» ___02___2025 г.

2025 г

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящее руководство пользователя предназначено для ознакомления пользователя с указаниями о правильной установке и эксплуатации программного обеспечения «Интегрированная система разработки и исполнения алгоритмов для программируемых логических контроллеров (ПРОКОНТ)».

1 Запуск и установка ПО

1.1 Для установки ПО «ПРОКОНТ» требуется ПЭВМ архитектуры x86-64 с ОС Астра 1.8.

1.2 Установить следующие пакеты из состава операционной системы:

- cmake;

- gss;

-g++.

1.3 Создать пользователя ambitecs.

1.4 С правами пользователя ambitecs скопировать и распаковать архив Qt.tar.gz в домашнюю директорию пользователя ambitecs (/home/ambitecs).

1.5 Выполнить команду с правами администратора:

sudo chmod –R +r/home/ambitecs

1.6 Скачать инсталлятор ПО «ПРОКОНТ» ProcontInstaller.bin по предоставленной ссылке.

1.7 Дать инсталлятору разрешение на исполнение и запустить его (*рисунок 1.1*).

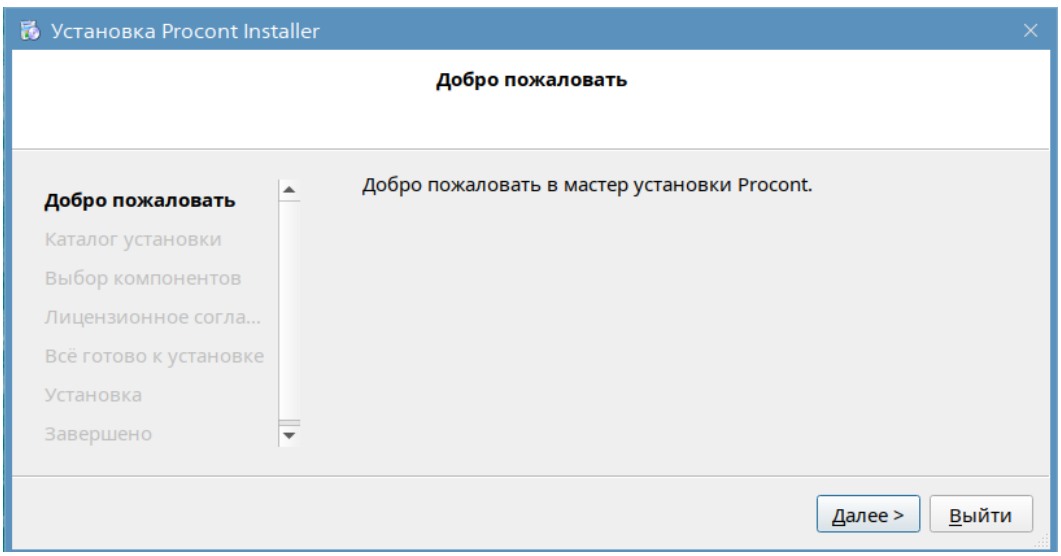

Рисунок 1.1 - Стартовое окно инсталлятора

1.8 Далее нужно указать место установки (*рисунок 1.2*). Если в выбранном каталоге уже есть установленный экземпляр программы, появляется сообщение с предложением выбрать другой каталог (*рисунок 1.3*), если каталог не пустой – сообщение с предупреждением (*рисунок 1.4*).

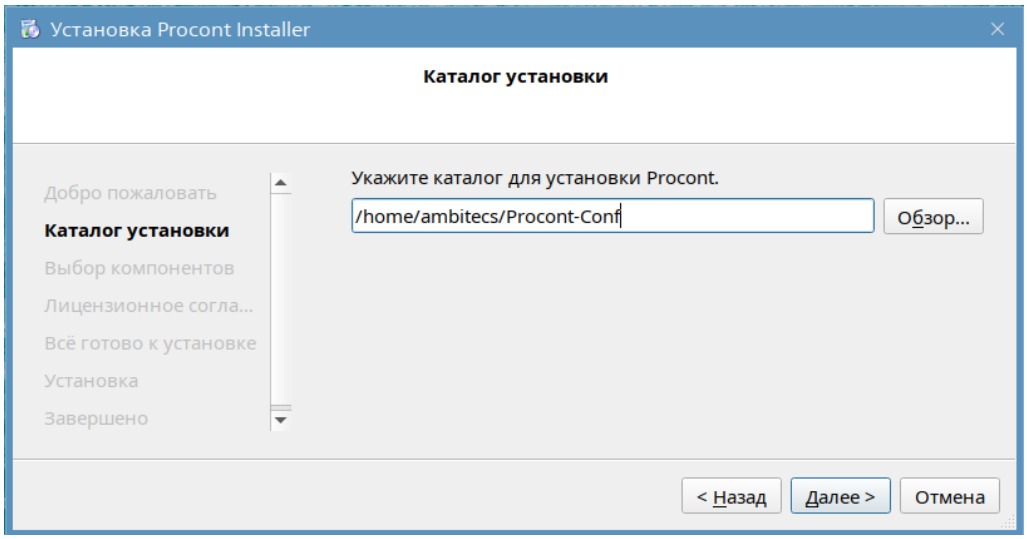

Рисунок 1.2 – Выбор каталога установки

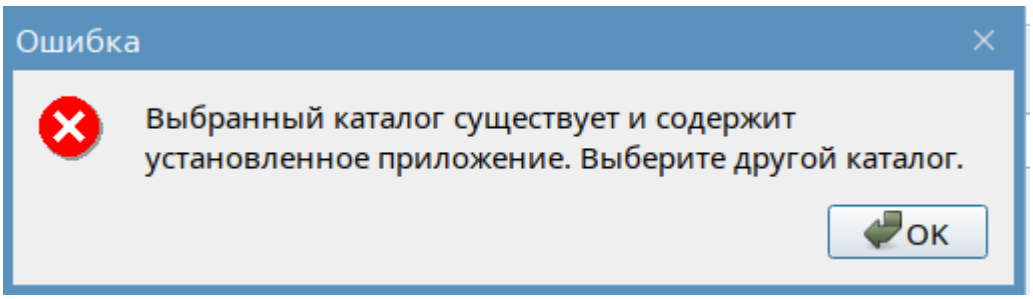

Рисунок 1.3 – Сообщение о наличии установленного приложения в каталоге

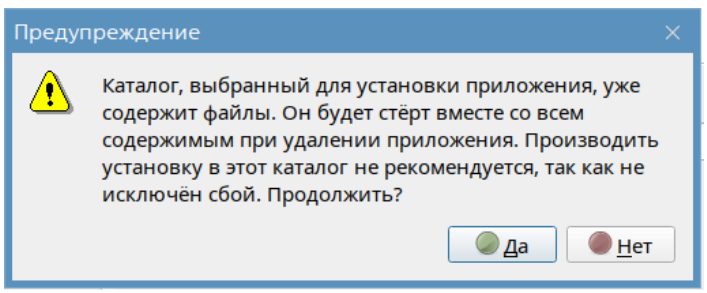

Рисунок 1.4 – Предупреждение о не пустом каталоге установки

1.9 Выбрать устанавливаемые компоненты (*рисунок 1.5*).

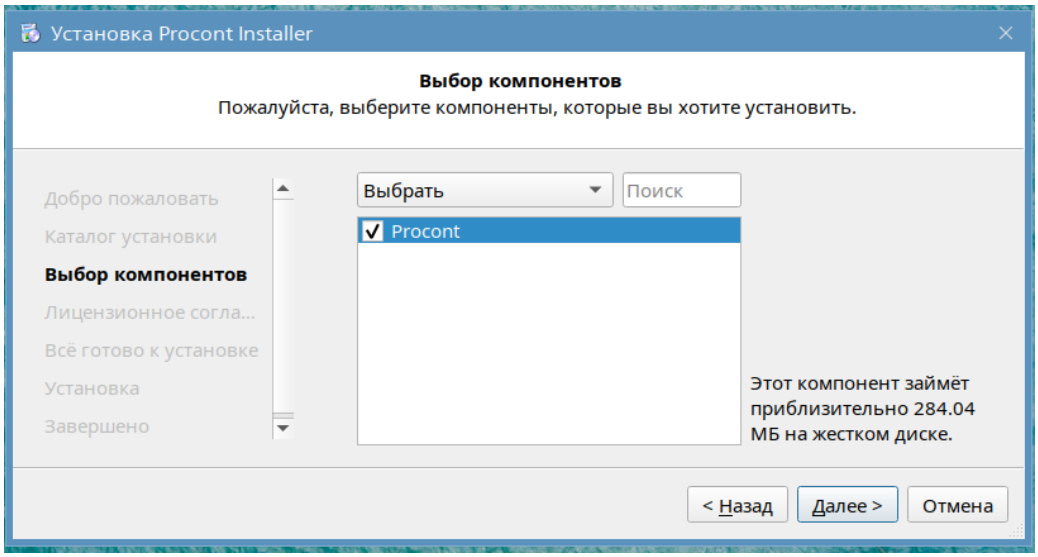

Рисунок 1.5 – Окно выбора компонентов установки

1.10 Подтвердить согласие с лицензионным соглашением (*рисунок 1.6*) и нажать кнопку «Установить» (*рисунок 1.7*), после установки нажать кнопку «Завершить» (*Рисунок 1.8*).

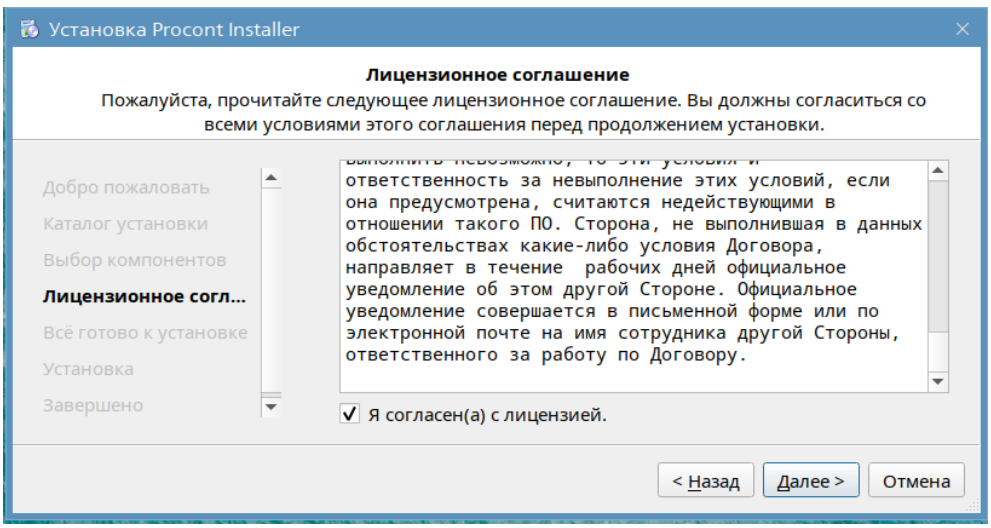

Рисунок 1.6 – Согласие с лицензией

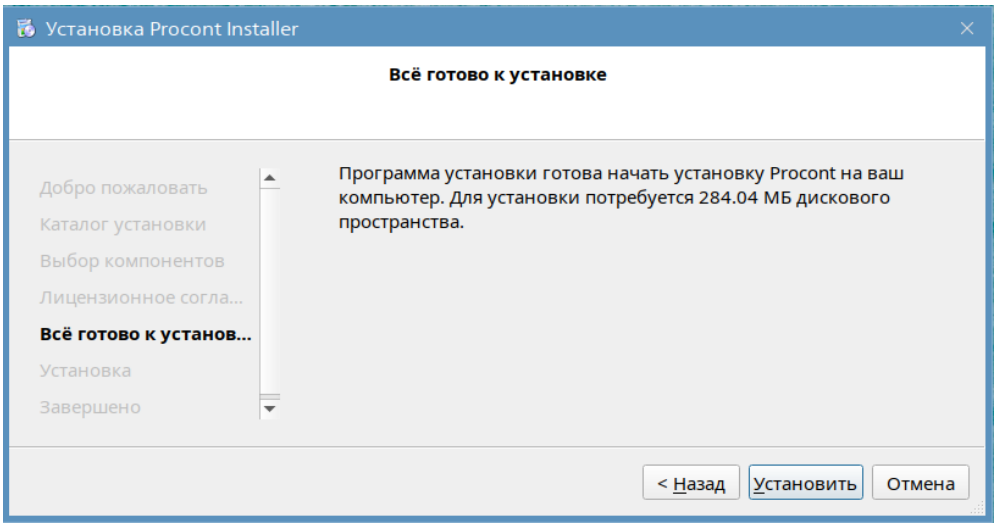

Рисунок 1.7 – Старт установки

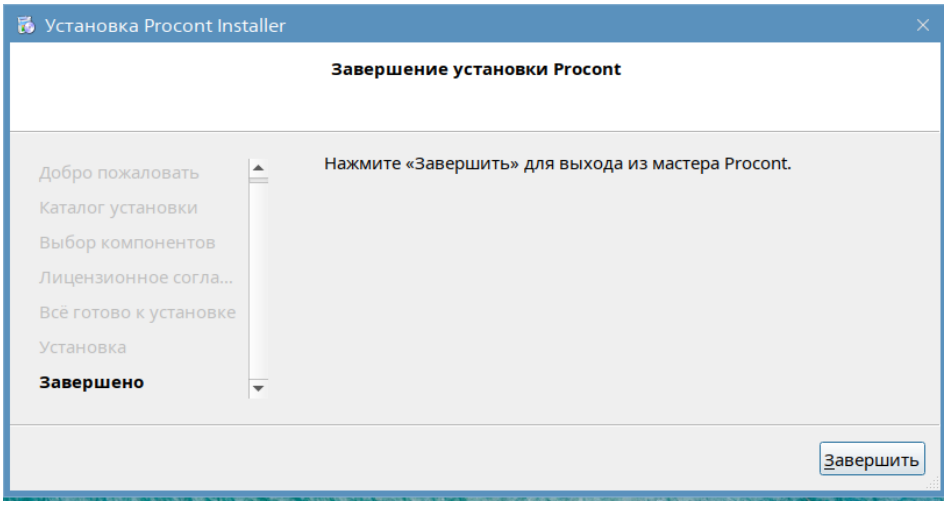

Рисунок 1.8 – Окно завершения установки

1.11 Запуск procont-conf-app можно осуществить двумя способами:

- с помощью ярлыка procont-conf-app на рабочем столе;
- в консоли ввести команду /Каталог установки/bin/procont-conf-app.

2 Библиотеки стандартных POU

В данной главе приведены функции и функциональные блоки стандартной библиотеки.

2.1 Библиотека математических функций

Таблица. 2.1 Библиотека математических функций

№	Имя	Сигнатура	Описание
1	ABS	ABS(ANY_NUM) => ANY_NUM	Абсолютное число
2	SQRT	SQRT(ANY_REAL) => ANY_REAL	Квадратный корень
3	LN	LN(ANY_REAL) => ANY_REAL	Натуральный логарифм
4	LOG	LOG(ANY_REAL) => ANY_REAL	Логарифм десятичный
5	EXP	EXP(ANY_REAL) => ANY_REAL	Возведение в степень
6	SIN	SIN(ANY_REAL) => ANY_REAL	Синус
7	COS	COS(ANY_REAL) => ANY_REAL	Косинус
8	TAN	TAN(ANY_REAL) => ANY_REAL	Касательный
9	ASIN	ASIN(ANY_REAL) => ANY_REAL	Синус дуги
10	ACOS	ACOS(ANY_REAL) => ANY_REAL	Косинус дуги
11	ATAN	ATAN(ANY_REAL) => ANY_REAL	Тангенс дуги
12	ADD	ADD(ANY_NUM, ANY_NUM) => ANY_NUM	Дополнение
13	MUL	MUL(ANY_NUM, ANY_NUM) => ANY_NUM	Умножение
14	SUB	SUB(ANY_NUM, ANY_NUM) => ANY_NUM	Вычитание
15	DIV	DIV(ANY_NUM, ANY_NUM) => ANY_NUM	Деление
16	MOD	MOD(ANY_INT, ANY_INT) => ANY_INT	Остаток (по модулю)
17	EXPT	EXPT(ANY_REAL, ANY_NUM) => ANY_REAL	Показатель степени
18	MOVE	MOVE(ANY) => ANY	Назначение

2.2 Библиотека логических (битовых) функций

Таблица 2.2 Библиотека битовых функций

№	Имя	Сигнатура	Описание
1	SHL	SHL(ANY_BIT, ANY_INT) => ANY_BIT	Сдвиг влево
2	SHR	SHR(ANY_BIT, ANY_INT) => ANY_BIT	Сдвиг вправо
3	ROR	ROR(ANY_NBIT, ANY_INT) => ANY_NBIT	Поворот вправо
4	ROL	ROL(ANY_NBIT, ANY_INT) => ANY_NBIT	Поворот влево
5	AND	AND(ANY_BIT, ANY_BIT) => ANY_BIT	Побитовое И
6	OR	OR(ANY_BIT, ANY_BIT) => ANY_BIT	Побитовое ИЛИ
7	XOR	XOR(ANY_BIT, ANY_BIT) => ANY_BIT	Побитовое ИСКЛЮЧЕНИЕ
8	NOT	NOT(ANY_BIT) => ANY_BIT	Побитовое инвертирование

2.3 Библиотека функций времени

Таблица 2.3 Библиотека функций времени

№	Имя	Сигнатура	Описание
1	ADD_TIME	ADD_TIME(TIME, TIME) => TIME	Сложение по времени

2	ADD_TOD_TIME	ADD_TOD_TIME(TOD, TIME) => TOD	Сложение по времени суток
3	ADD_DT_TIME	ADD_DT_TIME(DT, TIME) => DT	Сложение по дате
4	MULTIME	MULTIME(TIME, ANY_NUM) => TIME	Умножение по времени
5	SUB_TIME	SUB_TIME(TIME, TIME) => TIME	Вычитание по времени
6	SUB_DATE_DATE	SUB_DATE_DATE(DATE, DATE) => TIME	Вычитание по дате
7	SUB_TOD_TIME	SUB_TOD_TIME(TOD, TIME) => TOD	Вычитание из времени суток
8	SUB_TOD_TOD	SUB_TOD_TOD(TOD, TOD) => TIME	Вычитание по времени суток
9	SUB_DT_TIME	SUB_DT_TIME(DT, TIME) => DT	Вычитание времени
10	SUB_DT_DT	SUB_DT_DT(DT, DT) => TIME	Вычитание по дате и времени
11	DIVTIME	DIVTIME(TIME, ANY_NUM) => TIME	Деление по времени

2.4 Библиотека строковых функций

Таблица 2.4 Библиотека строковых функций

№	Имя	Сигнатура	Описание
1	LEN	LEN(STRING) => INT	Длина строки
2	LEFT	LEFT(STRING, ANY_INT) => STRING	строка слева от
3	RIGHT	RIGHT(STRING, ANY_INT) => STRING	строки справа от
4	MID	MID(STRING, ANY_INT, ANY_INT) => STRING	строки от середины
5	CONCAT	CONCAT(STRING, STRING) => STRING	Сцепление
6	INSERT	INSERT(STRING, STRING, ANY_INT) => STRING	Вставка (в)
7	DELETE	DELETE(STRING, ANY_INT, ANY_INT) => STRING	Удаление (внутри)
8	REPLACE	REPLACE(STRING, STRING, ANY_INT, ANY_INT) => STRING	Замена (внутри)
9	FIND	FIND(STRING, STRING) => INT	Найти позицию

2.5 Библиотека функций выбора

Таблица 2.5 Библиотека функций выбора

№	Имя	Сигнатура	Описание
1	SEL	SEL(BOOL, ANY, ANY) => ANY	Бинарный выбор (1 из 2)
2	MAX	MAX(ANY, ANY) => ANY	Максимальный
3	MIN	MIN(ANY, ANY) => ANY	Минимальный
4	LIMIT	LIMIT(ANY, ANY, ANY) => ANY	Ограничение
5	MUX	MUX(ANY_INT, ANY, ANY) => ANY	Мультиплексор (1 из N)

2.6 Библиотека функций сравнения

Таблица 2.6 Библиотека функций сравнения

№	Имя	Сигнатура	Описание
1	GT	GT(ANY, ANY) => BOOL	Больше, чем
2	GE	GE(ANY, ANY) => BOOL	Больше или равно
3	EQ	EQ(ANY, ANY) => BOOL	Равно
4	LT	LT(ANY, ANY) => BOOL	Меньше, чем
5	LE	LE(ANY, ANY) => BOOL	Меньше или равно
6	NE	NE(ANY, ANY) => BOOL	Не равно

2.7 Библиотека функций преобразования

Таблица 2.7 Библиотека функций преобразования

№	Имя	Сигнатура	Описание
1	TRUNC	TRUNC(ANY_REAL) => ANY_INT	Округление
2	DATE_AND_TIME_ TO_TIME_OF_DAY	DATE_AND_TIME_TO_TIME_OF_DAY(DT) => TOD	Пересчет в зависимости от времени суток
3	DATE_AND_TIME_ TO_DATE	DATE_AND_TIME_TO_DATE(DT) => DATE	Пересчет в дату

2.8 Библиотека стандартных функциональных блоков

Таблица 2.8 Библиотека стандартных функциональных блоков

№	Имя	Сигнатура	Описание
1	RTC	RTC(BOOL, DT)	Часы реального времени
2	INTEGRAL	INTEGRAL(BOOL, BOOL, REAL, REAL, TIME)	Интеграл входа по времени
3	DERIVATIVE	DERIVATIVE(BOOL, REAL, TIME)	Производная от входа
4	PID	PID(BOOL, REAL, REAL, REAL, REAL, REAL, REAL, TIME)	PID регулятор с замкнутым контуром
5	RAMP	RAMP(BOOL, REAL, REAL, TIME, TIME)	RAMP (изменение входа)
6	HYSTERESIS	HYSTERESIS(REAL, REAL, REAL)	Гистерезис
7	SR	SR(BOOL, BOOL)	Память SR (Set)
8	RS	RS(BOOL, BOOL)	Память RS (Reset)
9	SEMA	SEMA(BOOL, BOOL)	Семафор
10	R_TRIG	R_TRIG(BOOL)	Восходящий фронт
11	F_TRIG	F_TRIG(BOOL)	Нисходящий фронт
12	CTU	CTU(BOOL, BOOL, INT)	Сигнал верхнего значения
13	CTU_DINT	CTU_DINT(BOOL, BOOL, DINT)	Сигнал верхнего значения
14	CTU_LINT	CTU_LINT(BOOL, BOOL, LINT)	Сигнал верхнего значения
15	CTU_UDINT	CTU_UDINT(BOOL, BOOL, UDINT)	Сигнал верхнего значения
16	CTU_ULINT	CTU_ULINT(BOOL, BOOL, ULINT)	Сигнал верхнего значения
17	CTD	CTD(BOOL, BOOL, INT)	Сигнал нулевого значения
18	CTD_DINT	CTD_DINT(BOOL, BOOL, DINT)	Сигнал нулевого значения
19	CTD_LINT	CTD_LINT(BOOL, BOOL, LINT)	Сигнал нулевого значения
20	CTD_UDINT	CTD_UDINT(BOOL, BOOL, UDINT)	Сигнал нулевого значения
21	CTD_ULINT	CTD_ULINT(BOOL, BOOL, ULINT)	Сигнал нулевого значения
22	CTUD	CTUD(BOOL, BOOL, BOOL, BOOL, INT)	Сигнал вверх/вниз (CU/CD)
23	CTUD_DINT	CTUD_DINT(BOOL, BOOL, BOOL, BOOL, DINT)	Сигнал вверх/вниз (CU/CD)
24	CTUD_LINT	CTUD_LINT(BOOL, BOOL, BOOL, BOOL, LINT)	Сигнал вверх/вниз (CU/CD)
25	CTUD_UDINT	CTUD_UDINT(BOOL, BOOL, BOOL, BOOL, UDINT)	Сигнал вверх/вниз (CU/CD)
26	CTUD_ULINT	CTUD_ULINT(BOOL, BOOL, BOOL, BOOL, ULINT)	Сигнал вверх/вниз (CU/CD)
27	TP	TP(BOOL, TIME)	Импульсный таймер
28	TON	TON(BOOL, TIME)	Таймер задержки включения
29	TOF	TOF(BOOL, TIME)	Задержка выключения

3 Редактор ST

Основные структуры языка ST (*Рисунок 3.1*):

- POU (program, function, function block);
- Variable (input var, output var, local var, external var, global var);
- Array;
- Struct и др.

```
<dataTypes/>
<pous>
  <pou name="AverageVal" pouType="function">
    <interface>
      <returnType>
        <REAL/>
      </returnType>
      <inputVars>
        <variable name="Cnt1">
          <type>
            <INT/>
          </type>
        </variable>
        <variable name="Cnt2">
          <type>
            <INT/>
          </type>
        </variable>
        <variable name="Cnt3">
          <type>
            <INT/>
          </type>
        </variable>
        <variable name="Cnt4">
          <type>
            <INT/>
          </type>
        </variable>
        <variable name="Cnt5">
          <type>
            <INT/>
          </type>
        </variable>
      </inputVars>
      <localVars>
```

Рисунок 3.1 - Данные в формате XML

Реализованы механизмы импорта и экспорта модуля проекта в отдельный ST документ, с последующим обработкой написанного текста в структуры с данными с возможностью их анализа и перевода в другие форматы данных.

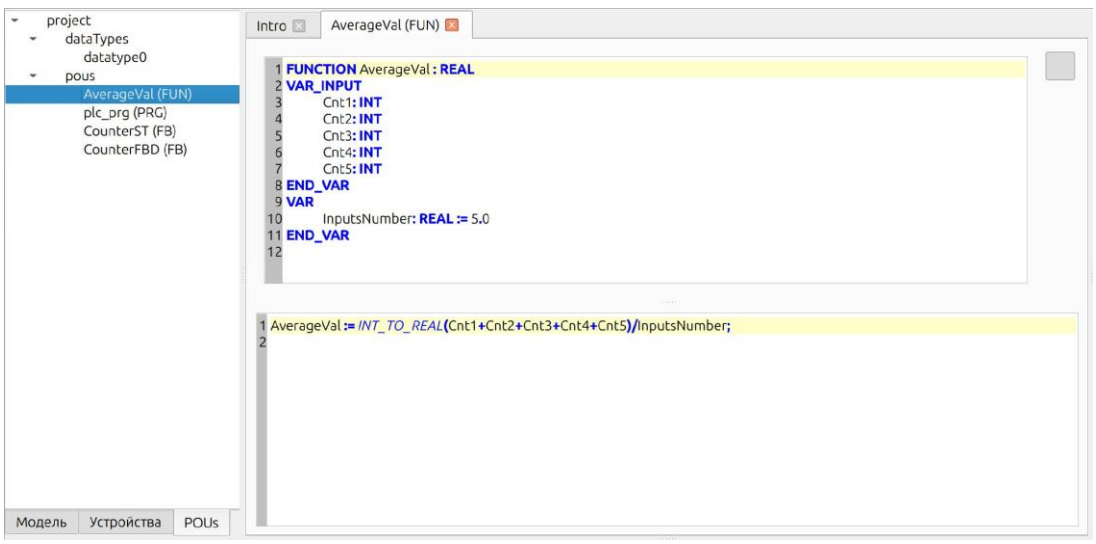

Рисунок 3.2 - Данные в текстовом представлении ST

4 Редактор графических схем FBD

4.1 FBD диаграмма

Основной компонент

```
class CDiagramWidget: public QWidget {}
```

путь: {project_root}/fbd/graphics/cdiagramwidget.h

Дополнительно задействованы визуальные компоненты:

- Заглушка от перезагрузки приложения (см. README.md)

```
class OglWidget: public QOpenGLWidget {}
```

путь: {project_root}/general/forms/main/OglWidget.h

- Дерево компонентов (панель инструментов). В GUI до создания диаграммы.

```
class CTreeObject : public QTreeWidget {}
```

путь: {project_root}/general/forms/derived/ctreeobject.h

Использование

```
explicit CDiagramWidget(

    const QDomNode &pou_node,
    CTreeObject * tree_object ,
    const bool &is_editable = true,
    QWidget *parent = nullptr

);
```

где:

- const QDomNode &pou_node

  содержит POU в формате OpenXML Exchange, типа:

```
<pou name="READ_.." poutype="program">

          <interface>…</interface>
          <fbd>…..</fbd>

</pou>
```

- CTreeObject * tree_object – само дерево компонентов, где диаграмма отстраивает «свои» компоненты
- const bool &is_editable – режим открытия диаграммы

Реализован режим «помощник ввода».

Для сборки приложения (компонента) помимо библиотек Qt необходимы следующие библиотеки (см CMakeLists.txt):

```
/usr/lib/x86_64-linux-gnu/libGLU.so (Linux)

msys64 - C:/msys64/mingw64/lib/libopengl32.a (Windows).
```

4.2 Вызов редактора FBD

Выбрать в проекте POU соответствующего типа (FBD) - *Рисунок 4.1*

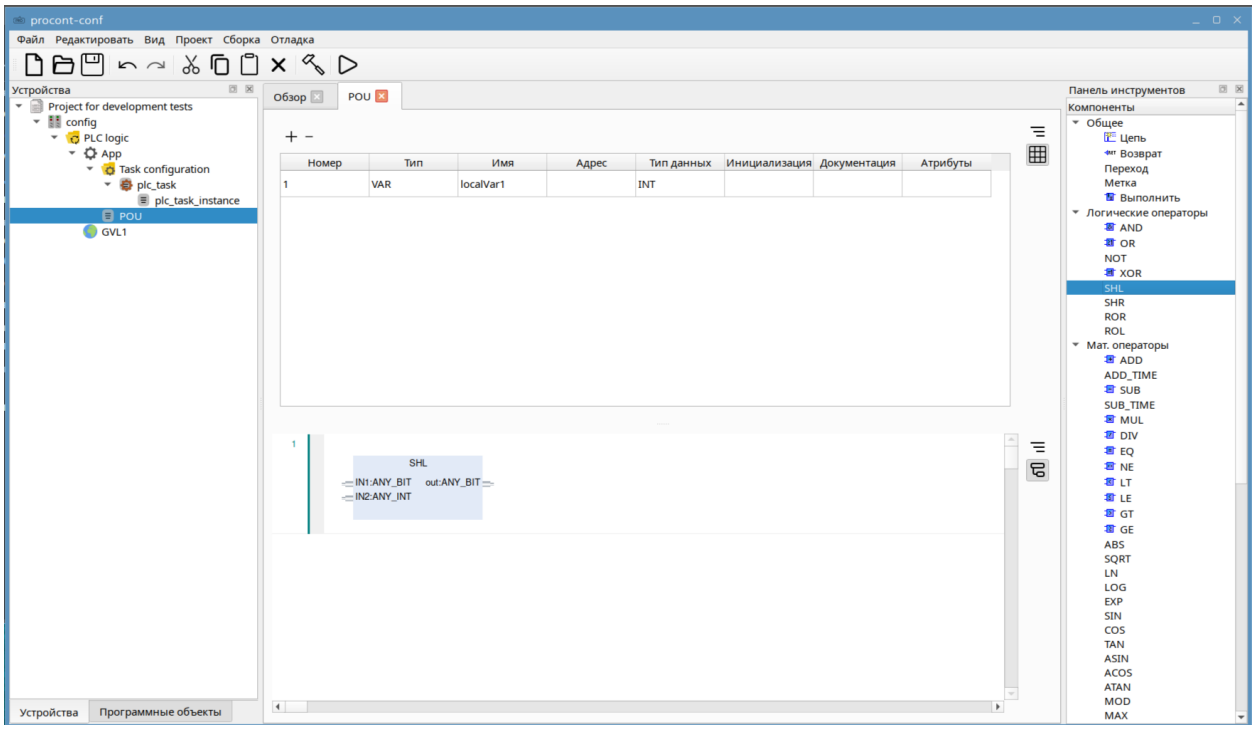

Рисунок 4.1 - Вызов редактора FBD

Поле редактора разделено на «ступени» - горизонтальные разделы пространства редактора. Ступени нумеруются 1, 2, 3… и т.д. Всего – до 100 ступеней.

Первоначально, POU, созданный во внешнем редакторе, например перенесенный из OpenPLC editor, располагается в первой ступени. Это происходит потому, что он не содержит соответствующих метаданных, т.к. в редакторе OpenPLC не предусмотрено разделение поля на ступени.

Пустые ступени используются для дальнейшей разработки и отладки алгоритмов. Например, можно вручную переносить блоки между ступенями и наполнять ступени новым содержимым.

Внешний вид редактора FBD представлен *Рисунке 4.2*

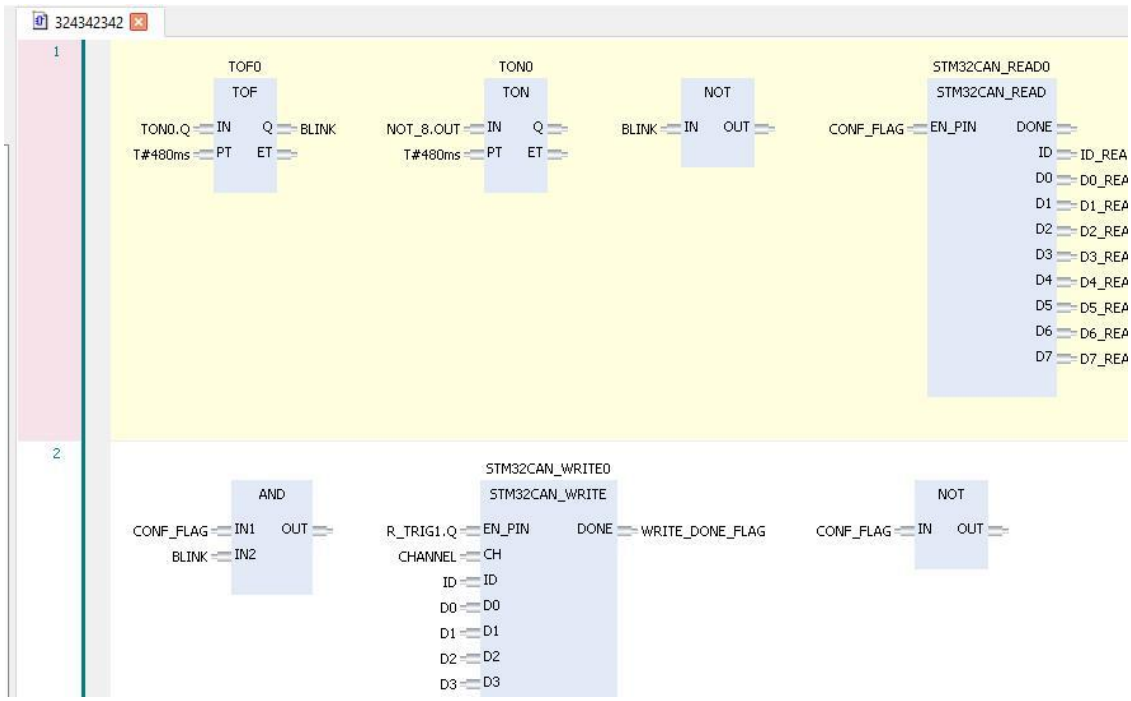

Рисунок 4.2 - Вид редактора FBD

4.3 Функции редактора FBD

4.4 DRAG-DROP с дерева компонентов:

Из дерева (палитры) компонентов можно перенести 4 компонента из группы «логические операторы»: AND, AND3, OR, OR3, а также из группы «общее» - Цепь.

4.4.1 DRAG-DROP в диаграмме:

− Выбор объекта переноса осуществляется стандартным кликом мыши (левая кнопка).

− Выбор только ступени – клик в любой свободной части ступени. −

Выбор компонента – клик по «телу» компонента.

− Выбор *пина* для создания связей клик на *пине*.

Можно не отпуская кнопку мыши двигать выделенный объект по стандартной технологии «Drag’n’drop»

4.4.2 Перемещение ступеней

Если передвигать ступень, то двигая курсор с изображением ступени над другими ступенями будет подсвечена позиция вставки:

Рисунок 4.3 - Перемещение ступеней

4.4.3 Перенос компонентов

При переносе компонента, двигая курсор с изображением компонента над ступенями нужно понимать, что компонент нужно вставлять в подсвеченную область (*Рисунок 4.4*). При перемещении над пустой ступенью – вставка компонента возможна в произвольном месте.

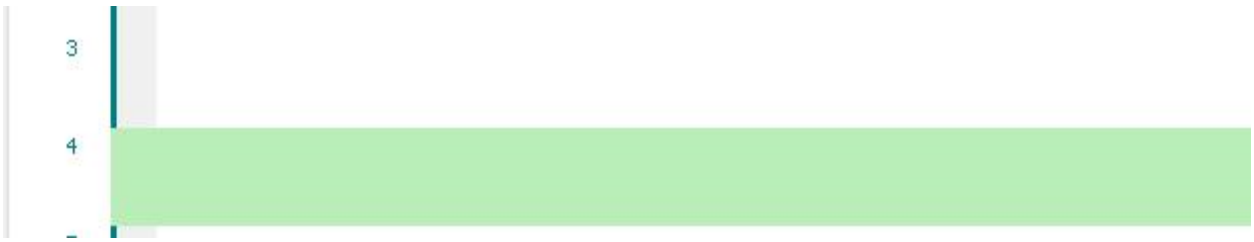

Рисунок 4.4 - Область вставки компонента

В данном примере курсор был над 4-й ступенью.

Если двигать компонент над ступенью с имеющимися компонентами, то нужно понимать, что вставлять можно на компоненты исходя из логики, что передвигаемый объект будет вставлен в ступень ПЕРЕД компонентом, над которым произошла вставка.

Выбранный для вставки компонент будет подсвечен (*Рисунок 4.5*):

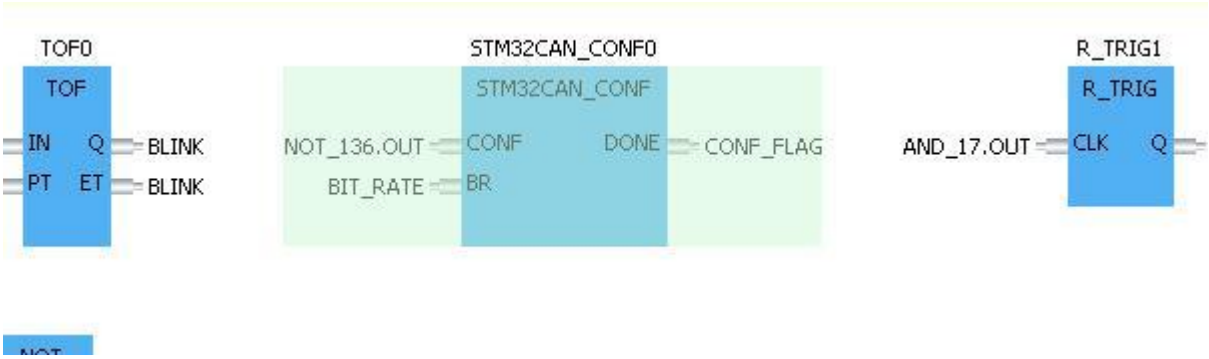

Рисунок 4.5 - Перемещение компонентов

В данном примере курсор с десантом был над «STM32CAN_CONF0».

Если компонент нужно добавить в конец уже имеющихся компонентов, то соответственно и нужно двигать курсор с дропом в ту сторону (*Рисунок 4.6*):

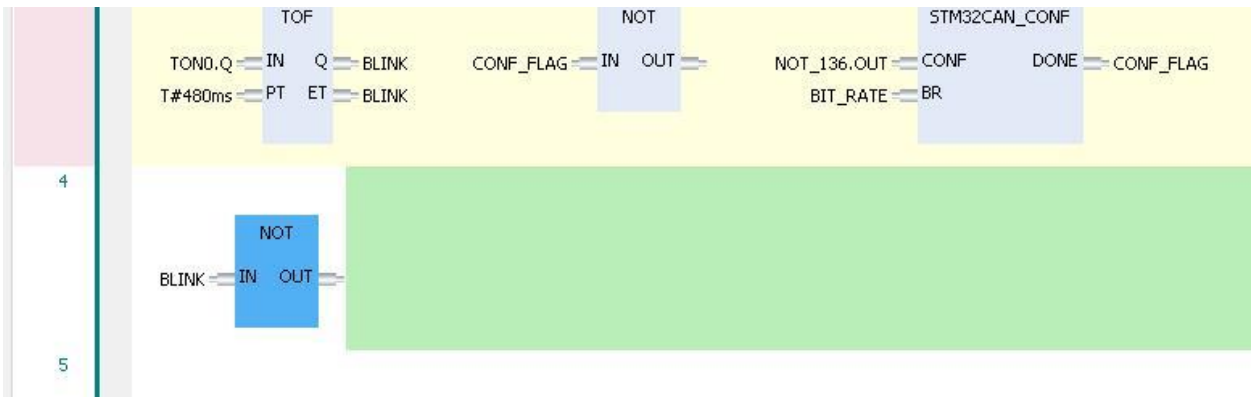

Рисунок 4.6 - Вставка компонентов

В данном примере двигали компонент над 5-й ступенью справа от имеющегося (“NOT”).

Для дропа в области за рамками виджета, необходимо дроп подвести к соответствующему краю, т.е. для автопрокрутки вниз, подвести курсор с десантом к нижнему краю и аналогично для остальных сторон.

При подведении курсора к краю высвечивается странный градиент (будет заменён на серый-прозрачный и увеличен). Вдоль этого градиента и двигать курсор, пока не поле редактора не прокрутится до нужного места.

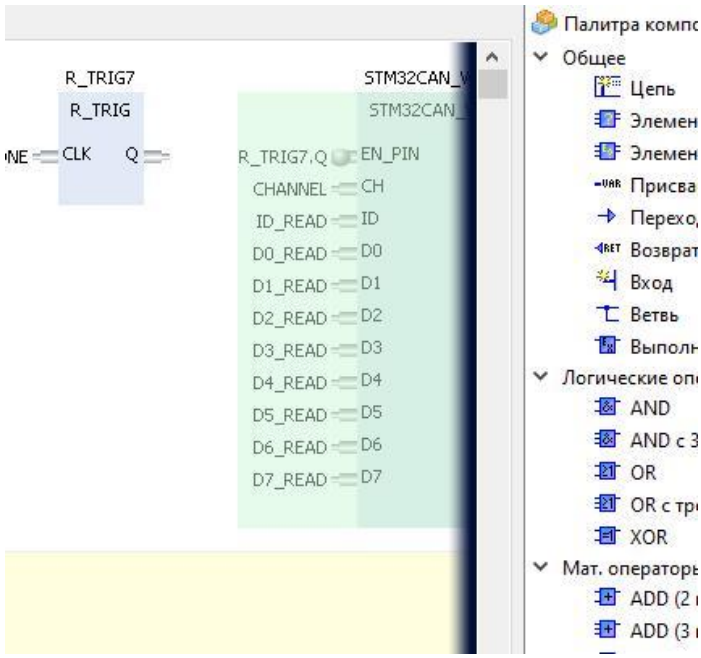

Рисунок 4.7 - Прокрутка поля редактора

В данном примере была зафиксирована попытка скроллинга вправо.

5 Создание простого проекта

Процесс создания пользовательского проекта заключается в выполнении следующих шагов:

5.1 Добавление в проект устройств (*рисунок 5.1*):

- Процессорный модуль (*рисунок 5.2*);
- Модули ввода/вывода (*рисунок 5.3)*.

5.2 Настройка аппаратных компонентов.

5.3 Добавление программных компонент:

- Константы и переменные;
- Функции;
- Функциональные блоки;
- Программные алгоритмические блоки (*рисунок 5.4*).

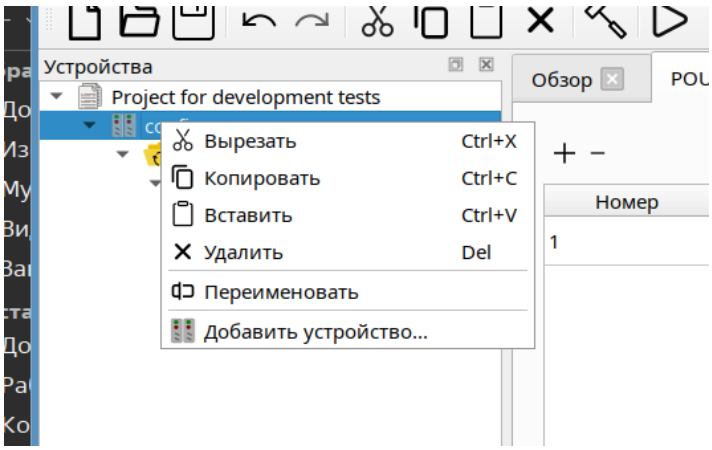

Рисунок 5.1 – Этап добавления устройств

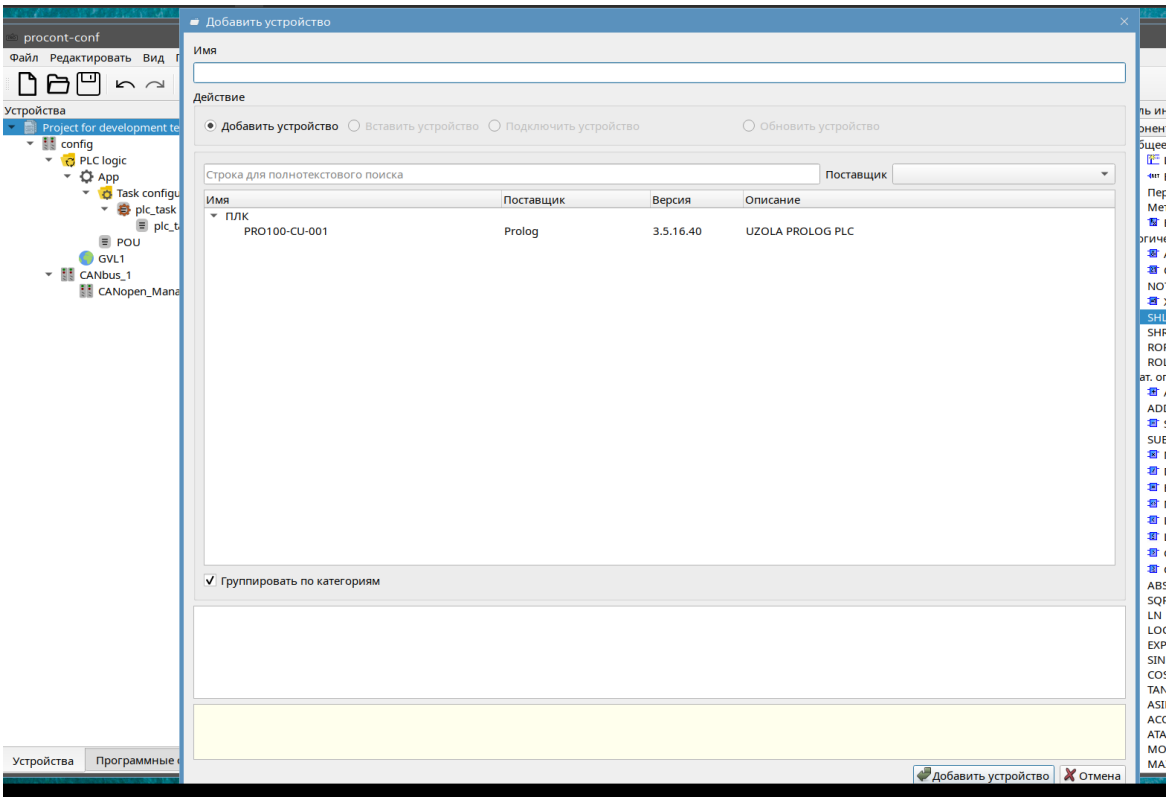

Рисунок 5.2 – Добавление процессорного модуля в проект

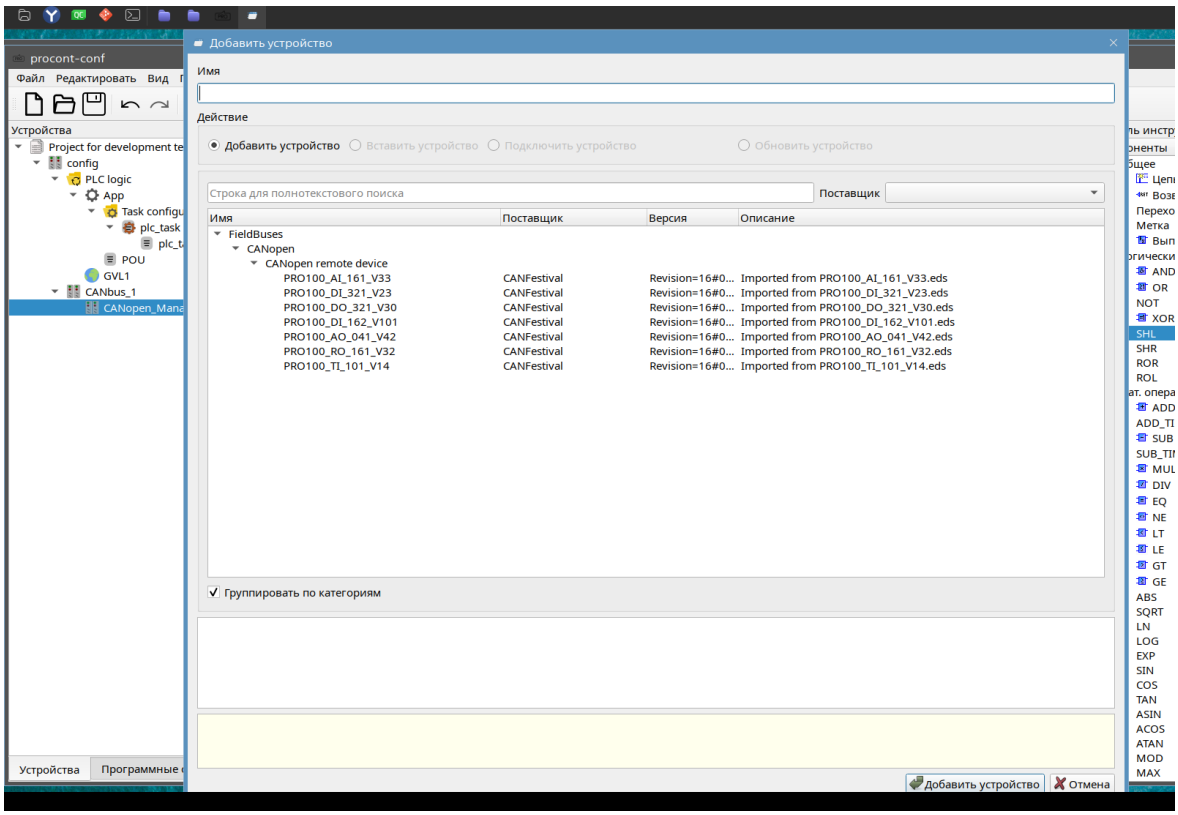

Рисунок 5.3 – Добавление модулей ввода/вывода в проект

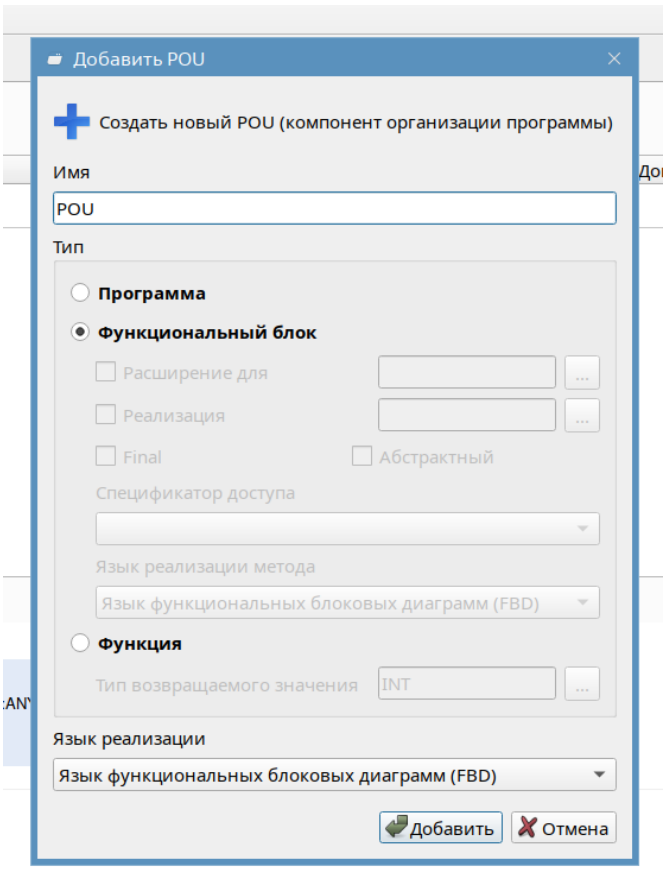

Рисунок 5.4 Добавление программных объектов