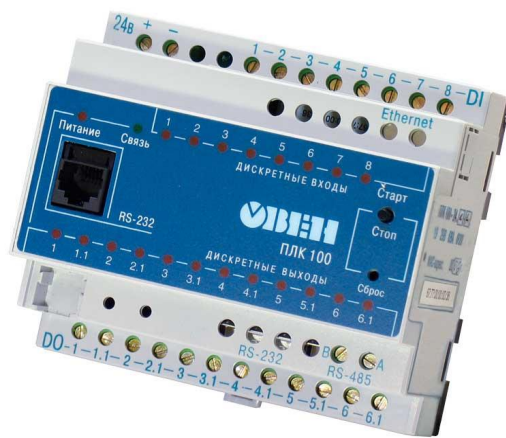




# Программирование промышленных контроллеров ОВЕН ПЛК в пакете CoDeSys V2.3



# Архитектура промышленных контроллеров

Контроллером в системах автоматизации называют устройство, выполняющее управление физическими процессами по заданному в нем алгоритму с использованием информации, получаемой от датчиков и выводимой в исполнительные устройства.



Под архитектурой контроллеров понимают совокупность общих структурных и логических подходов к созданию аппаратных средств, программного обеспечения и принципов организации взаимосвязанной работы их компонентов.

**Архитектура = Взаимодействие\*(железо + системное ПО + прикладное ПО)**

# Аппаратные средства

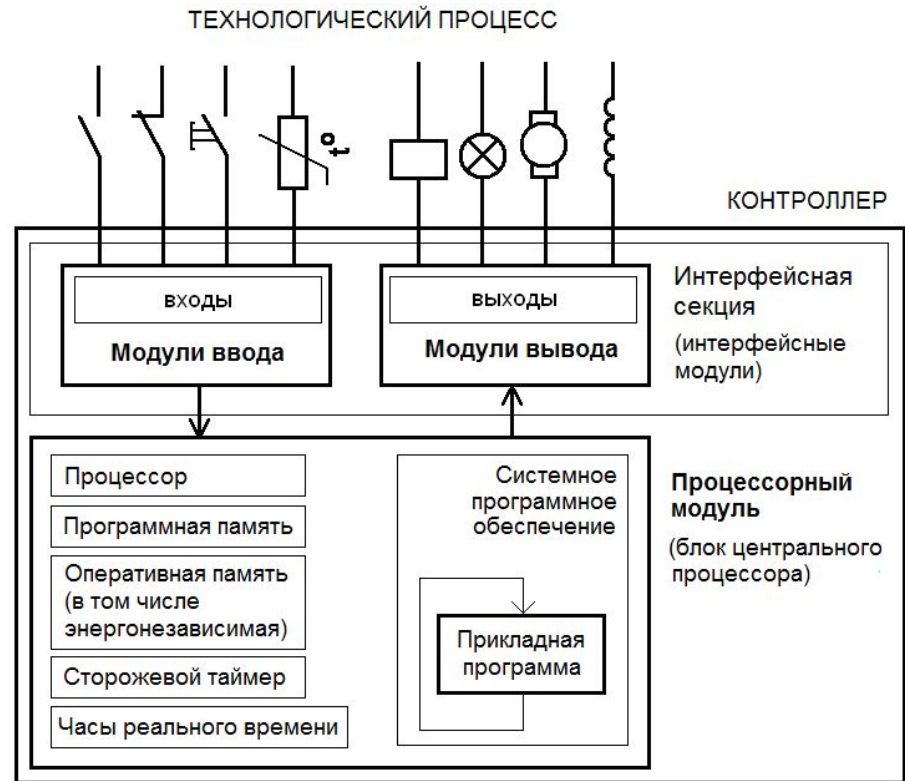
Процессорный модуль включает:

- Процессор
- Память
- Сторожевой таймер
- Часы реального времени

Периферийные модули:

- Дискретные модули ввода
- Дискретные модули вывода
- Аналоговые модули ввода
- Аналоговые модули вывода
- Коммуникационные модули

Способы объединения модулей  
(системная шина, сети)



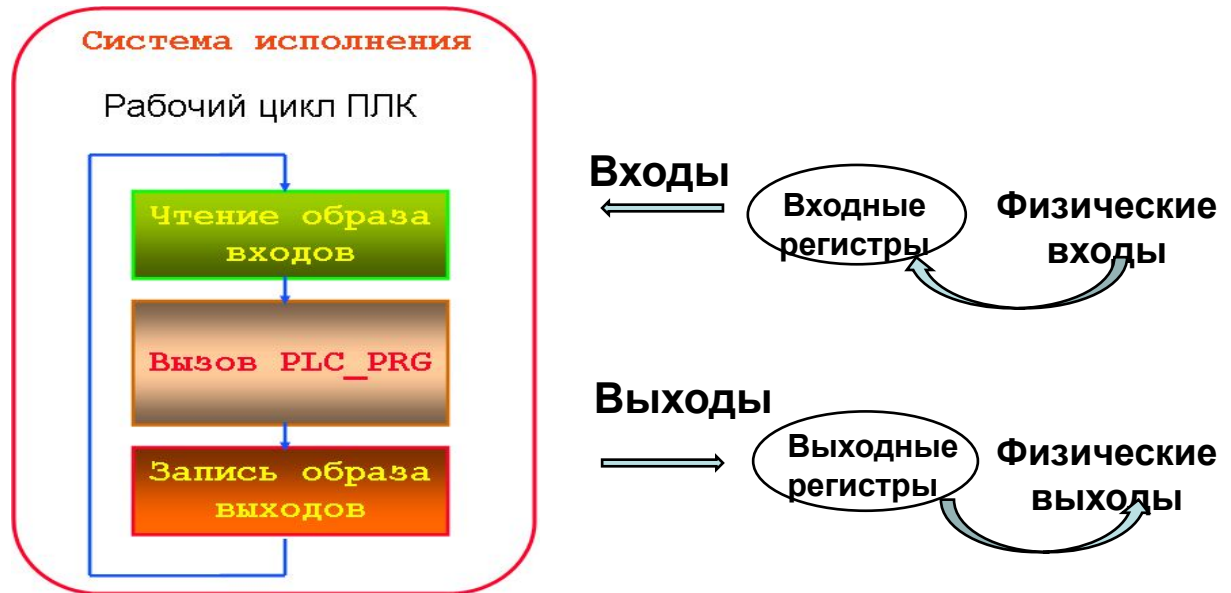
# Программные средства

- Прикладное программное обеспечение
- Системное программное обеспечение

# Взаимодействие программных и аппаратных средств

Прикладная программа выполняется циклически:

рабочий цикл контроллера  
время реакции

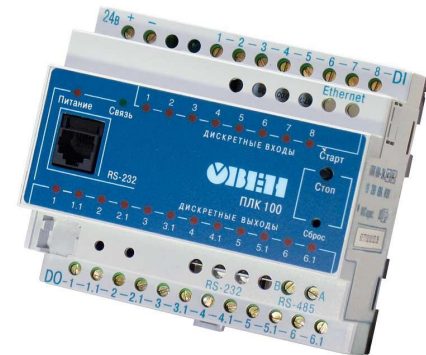


## Функции системного программного обеспечения

1. Чтение состояния входов.
2. Выполнение кода программы пользователя.
3. Запись состояния выходов.
4. Обслуживание - аппаратных ресурсов ПЛК (диагностика),
  - ресурсов прикладной программы (таймеры, счетчики),
  - средств обмена по сети.
5. Монитор системы исполнения.
6. Контроль времени цикла.

# Программно-технический комплекс компании ОВЕН

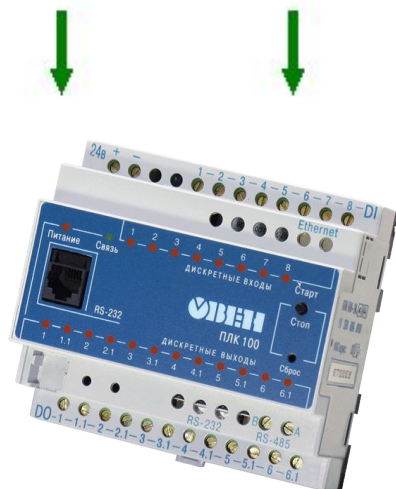
- Контроллеры
- Операторные панели
- Модули ввода-вывода
- Сети



# Контроллер ПЛК150-220.У-М

6 дискр  
входов

4 аналог  
входа



4 реле

2 аналог  
выхода

интерфейсы:

RS-232

RS-485

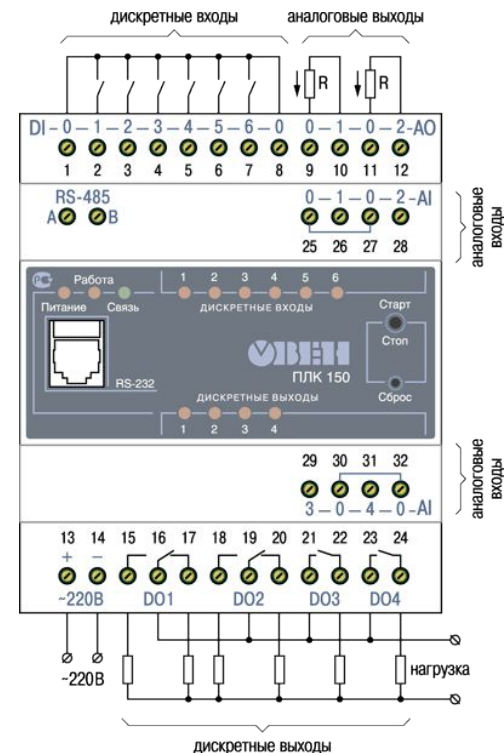
Ethernet

Протоколы:

ModBus (ASCII / RTU / TCP)

DCON

ОВЕН



220 - питание ~220 В

I - аналоговые выходы ПЛК150/154 ЦАП  $I_{\text{вых}} = 4 \dots 20$  мА

У - аналоговые выходы ПЛК150/154 ЦАП  $U_{\text{вых}} = 0 \dots 10$  В

A - аналоговые выходы ПЛК150/154 универсальные ЦАП  $4 \dots 20$  мА /  $0 \dots 10$  В

L - объем памяти ввода/вывода 360 байт (Low license)

M - объем памяти ввода/вывода не ограничен (Med. license)



# Состав учебного стенда



Пакет CoDeSys  
Пакет «Конфигуратор ИП320»

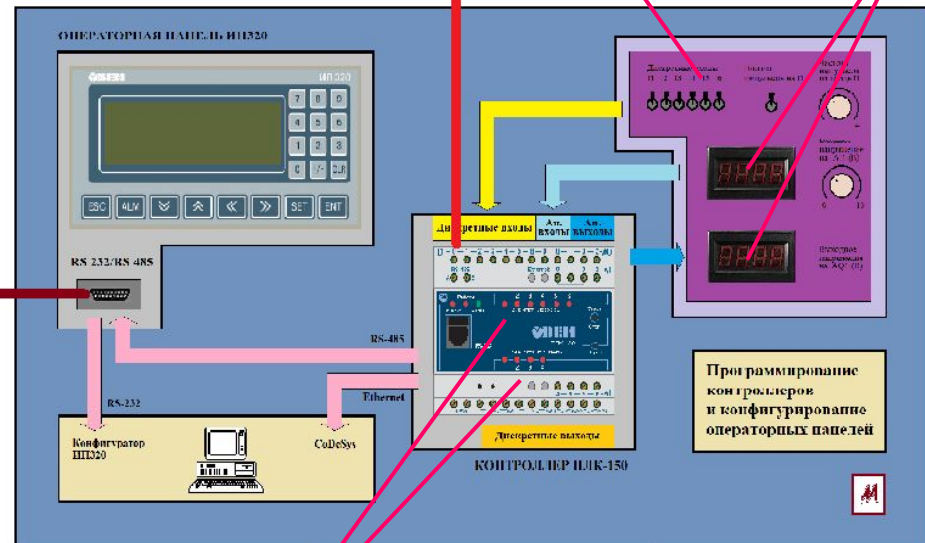
USB/RS-232

Ethernet

USB

Задание значений  
на дискретных входах

Задание значений входного  
и измерение выходного  
напряжения



Контроль значений сигналов  
на дискретных входах и выходах

# Стандарт МЭК 61131 Программируемые контроллеры (Programmable Controllers)

принят IEC (МЭК) в 1992 г. в пяти частях:

1. Общая информация.

2. Требования к оборудованию и испытаниям.

- электрические, механические и функциональные требования для промышленного контроллера и связанных с ними периферийных устройств;
- условия эксплуатации, хранения и транспортирования;
- методы испытаний и процедуры, которые должны использоваться для проверки соответствия характеристик промышленного контроллера и связанных с ним периферийных устройств установленным требованиям.

3. Языки программирования.

4. Руководства пользователя.

5. Разработка сообщений.

ГОСТ Р 51840–2001 (МЭК 61131–1–92) «Программируемые контроллеры. Общие положения и функциональные характеристики».

ГОСТ Р 51841–2001 - адаптированная часть стандарта МЭК 61131–2.

В настоящее время службами Госстандарта России проводятся работы по адаптации остальных разделов МЭК 61131.

В Украине в 2002 г. был принят стандарт ДСТУ 4108–2002,  
в республике Беларусь – СТБ IEC 61131–2–2010.



# Стандарт МЭК 61131-3 Programming languages

Общие требования стандарта к языкам.

В настоящее время стандарт МЭК 61131-3 поддерживает пять языков технологического программирования:

- Ladder Diagrams (LD) – язык релейных диаграмм;
- Function Block Diagram (FBD) – язык функциональных блоковых диаграмм;
- Statement List (ST) – язык структурированного текста;
- Instruction List (IL) – язык инструкций.
- Sequential Function Chart (SFC) – язык последовательных функциональных блоков;

На чем можно программировать еще?

С

Continuous Flow Chart (CFC) – язык непрерывной потоковой схемы

...

# Общие элементы языков

- Система команд
- Форматы данных (элементарные или базовые типы)

Ключевое слово	Диапазон	Пример
BOOL	0, 1	FALSE, TRUE, 0, 1
SINT, INT, DINT	-128 .. 127, -32768 .. 32767, -2147483648 .. 2147483647	0, 24453 -38099887
BYTE, WORD, DWORD	0 .. 255, 0 .. 65535, 0 .. 4294967295	8450 16#2102
REAL, LREAL	$-1.2 \times 10^{-38} \dots 3.4 \times 10^{38}$ $-2.3 \times 10^{-308} \dots 1.7 \times 10^{308}$	1.34996 2.8377E-15
STRING	1 .. 255 символов	`Emergency Stop`

- Организационные блоки программы (POU):
  1. Программы
  2. Функции
  3. Функциональные блоки

# Пакеты по созданию проектов

## Что такое проект?

Хранится в одном файле Name.pro

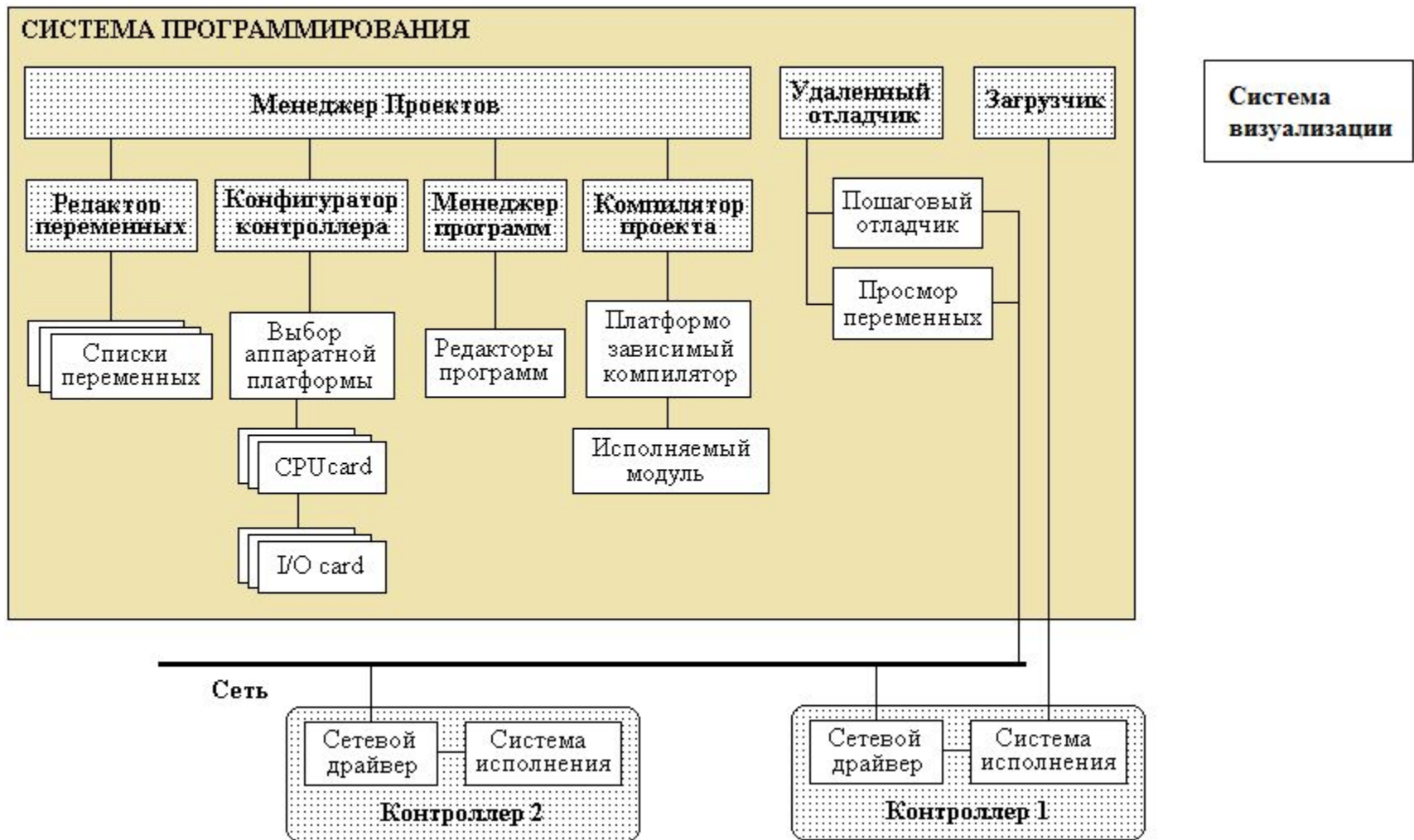
Содержит программные компоненты, ресурсы, визуализации...

## Два подхода к созданию пакетов:

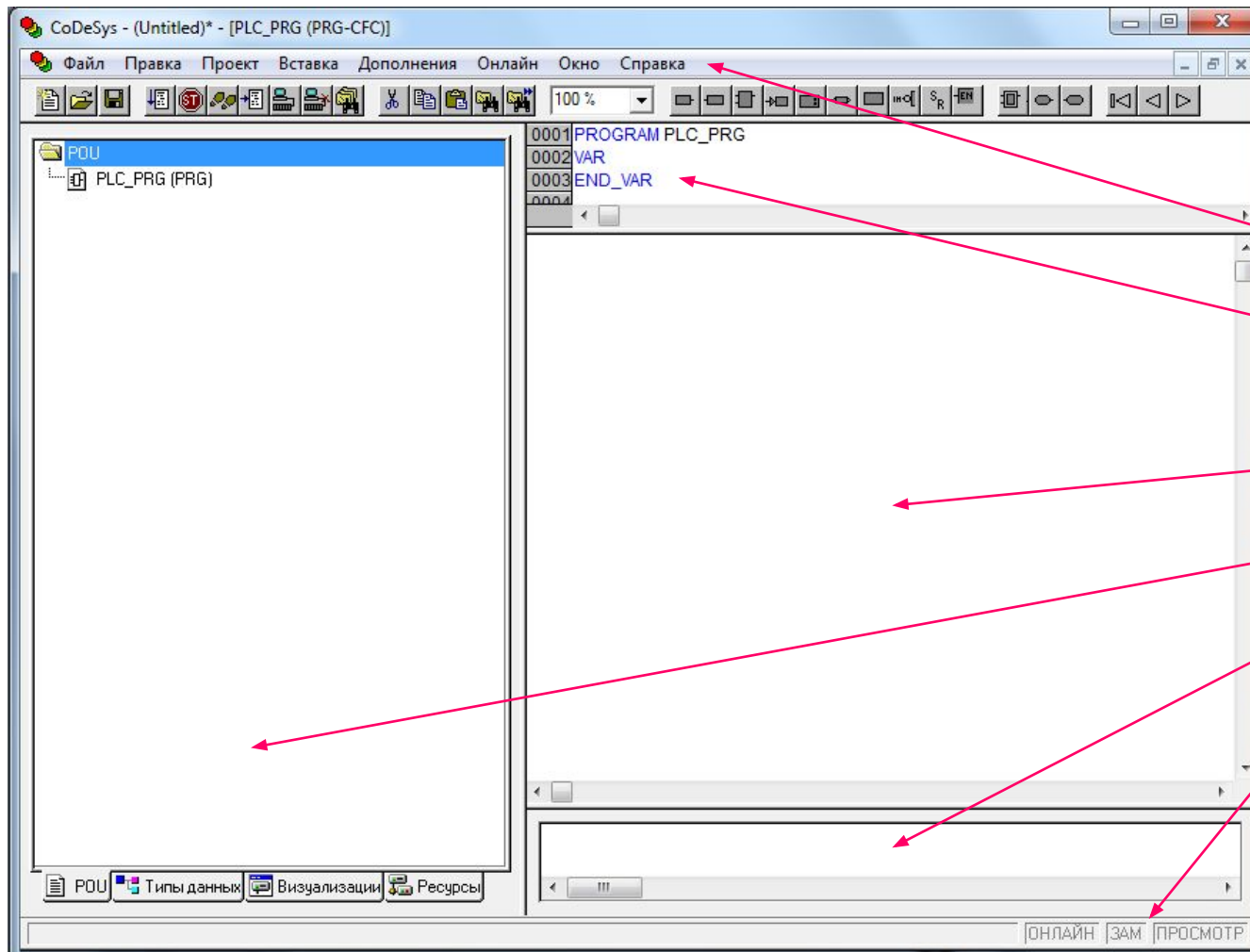
- «Фирменные» пакеты от производителей контроллеров для программирования только их контроллеров  
TIA Portal (Siemens)  
CX (Omron)
- Пакеты, в которых можно программировать контроллеры различных производителей  
CoDeSys (Smart Software Solutions - 3S)

Что такое Target Files (файлы целевой конфигурации)?

# Типовая структура пакетов программирования



# Главное окно CoDeSys



Главное меню  
и панель инструментов

Область  
определения  
переменных

Редактор программы

Менеджер объектов

Окно состояний

Строка статуса

# Связь с ПК

1

2

3

Контроллеры на производстве получают IP-адрес 10.0.6.10

# Настройки ПК для связи с контроллером

The image shows a Windows XP desktop environment with the Network Connections control panel open. The main window displays the 'Подключение по локальной сети' (Local Area Connection) for the 'Realtek PCIe GBE Family Controller'. A secondary window titled 'Подключение по локальной сети - свойства' (Local Area Connection - Properties) is open, showing the 'Сеть' (Networking) tab. In this window, the 'Подключение через:' (Connect through) field is set to 'Realtek PCIe GBE Family Controller'. Under 'Отмеченные компоненты используются этим подключением:' (Components that use this connection), the 'Протокол Интернета версии 4 (TCP/IPv4)' (Internet Protocol Version 4 (TCP/IPv4)) is selected and highlighted in blue. A third window, 'Свойства: Протокол Интернета версии 4 (TCP/IPv4)' (Properties: Internet Protocol Version 4 (TCP/IPv4)), is open, showing the 'Общие' (General) tab. The 'Использовать следующий IP-адрес:' (Use the following IP address) radio button is selected. The IP address is set to '10 . 0 . 6 . 11', the subnet mask to '255 . 255 . 0 . 0', and the primary gateway to '10 . 0 . 6 . 1'. A red arrow points from the text 'Работа с контроллером в одной подсети' to the IP address field. The background window also shows the 'Состояние - Подключение по локальной сети' (Status - Local Area Connection) window, which displays connection details like 'Подключение: IPv4-подключение: IPv6-подключение: Состояние среды: Длительность: Скорость: Сведения...' and 'Активность' (Activity) section with 'Отправлено' (Sent) and 'Байт:' (Bytes) information.

Подключение по локальной сети  
Сеть 3  
Realtek PCIe GBE Family Controller

Состояние - Подключение по локальной сети

Общие

Подключение

IPv4-подключение:  
IPv6-подключение:  
Состояние среды:  
Длительность:  
Скорость:  
Сведения...

Активность

Отправлено

Байт: 1 125 957

Свойства Отключить

Подключение по локальной сети - свойства

Сеть

Подключение через:  
Realtek PCIe GBE Family Controller  
Настроить...

Отмеченные компоненты используются этим подключением:

- Клиент для сетей Microsoft
- Kaspersky Anti-Virus NDIS 6 Filter
- Планировщик пакетов QoS
- Служба доступа к файлам и принтерам сетей Micro...
- Протокол Интернета версии 6 (TCP/IPv6)
- Протокол Интернета версии 4 (TCP/IPv4)
- Драйвер в/в топология канального уровня
- Ответчик обнаружения топологии канального уровня

Установить... Удалить Свойства

Описание  
Протокол TCP/IP - стандартный протокол глобальных сетей, обеспечивающий связь между различными взаимодействующими сетями.

Свойства: Протокол Интернета версии 4 (TCP/IPv4)

Общие

Параметры IP могут назначаться автоматически, если сеть поддерживает эту возможность. В противном случае параметры IP можно получить у сетевого администратора.

Получить IP-адрес автоматически

Использовать следующий IP-адрес:

IP-адрес: 10 . 0 . 6 . 11

Маска подсети: 255 . 255 . 0 . 0

Основной шлюз: 10 . 0 . 6 . 1

Получить адрес DNS-сервера автоматически

Использовать следующие адреса DNS-серверов:

Предпочитаемый DNS-сервер: . . .

Альтернативный DNS-сервер: . . .

Подтвердить параметры при выходе

Дополнительно...

OK Отмена

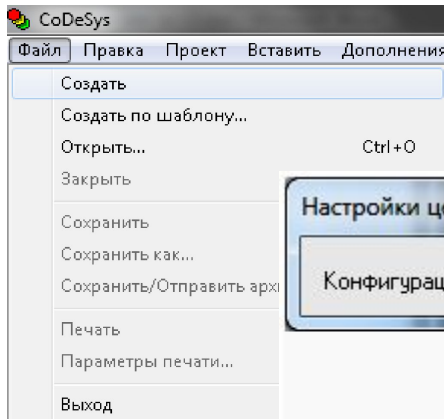
Работа с контроллером в одной подсети



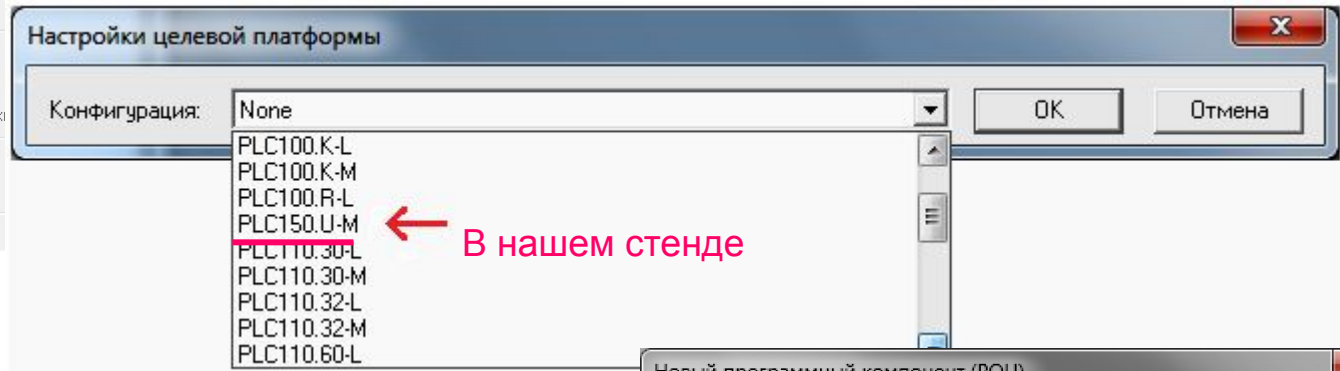


# Начало работы в CoDeSys

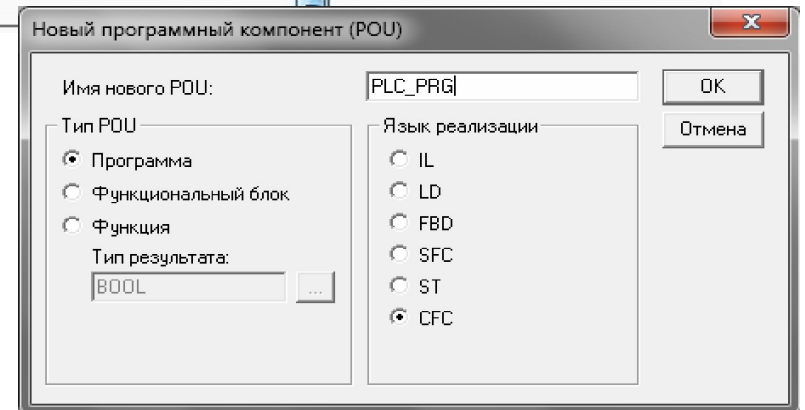
## 1. Создайте новый проект



## 2. Выберите целевую платформы



3. Задайте имя POU  
(основная программа в CoDeSys  
по умолчанию всегда PLC\_PRG)  
и выберите язык



## 4. Сохраните проект

# Задание времени цикла

Задание длительности минимального и максимального цикла контроллера

The screenshot displays the SIMATIC Manager interface. On the left, a tree view shows the project structure with 'Конфигурация ПЛК' (PLC Configuration) selected. The main window shows the 'PLC160' configuration page, specifically the 'Параметры модуля' (Module Parameters) tab. A table lists the parameters for the module:

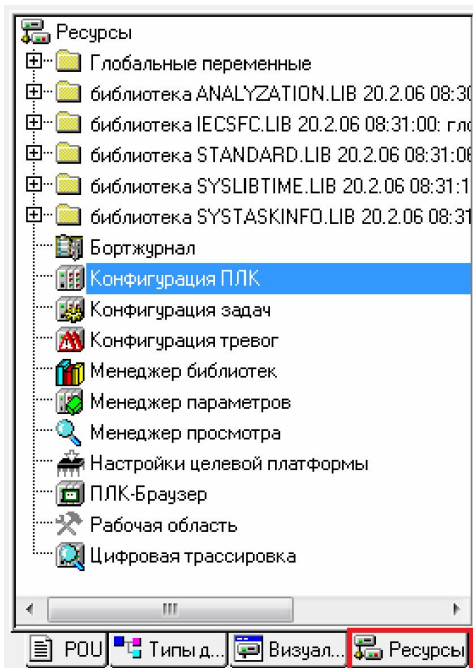
Индекс	Имя	Знач...	По ум...	Мин.	Макс.
1	MinCycleLength ms	1	1	0	50
2	MaxCycleLength ms	1000	1000	1000	10000
3	back-up working time	6	6	2	30



# Конфигурирование контроллера

Это задание состава и определение способов использования средств контроллера:

- соотнесение физических входов-выходов и имен переменных в программе (автоматически устанавливается формат вводимых в программу переменных);
- настройка параметров входов и выходов;
- включение в состав используемых средств сетевых компонентов и задание их параметров;
- определение и описание сетевых переменных.



```
└─ PLC 150 U
  └─ Discrete input 6 bit[FIX]
  └─ Discrete output 4 bit[FIX]
  └─ Special output[FIX]
  └─ Unified signal sensor[SLOT]
      └─ In_a AT %ID3.0: REAL; (* Value *) [CHANNEL (I)]
      └─ AT %IW3.1: WORD; (* Circular time *) [CHANNEL (I)]
      └─ Analog Input[FIX]
  └─ Unified signal sensor[SLOT]
  └─ Unified signal sensor[SLOT]
  └─ Unified signal sensor[SLOT]
  └─ Analog output[FIX]
      └─ Out_A AT %QD7.0: REAL; (* Value *) [CHANNEL (Q)]
  └─ Analog output[FIX]
  └─ ModBus (slave)[VAR]
      └─ Modbus[FIX]
          └─ RS-485-1[VAR]
      └─ 2 byte[VAR]
          └─ Disp_Volt AT %QW9.1.0: WORD; (* *) [CHANNEL (Q)]
      └─ 2 byte[VAR]
          └─ Disp_Temp AT %QW9.2.0: WORD; (* *) [CHANNEL (Q)]
```

Глобальные, сетевые и локальные переменные

Все переменные, определенные при конфигурировании контроллера, автоматически становятся глобальными

# Правила задания имен переменных

- Буквы и цифры английского языка.
- Имя должно начинаться с буквы.
- Только одинарные подчеркивания.
- Без пробелов.
- Нельзя использовать зарезервированные слова МЭК и операторы (AND, VAR, WORD, PROGRAM и т.д.).
- Регистр букв не различается.

**In1, Lamp, Datchik,  
Temp\_dvig**

Так можно

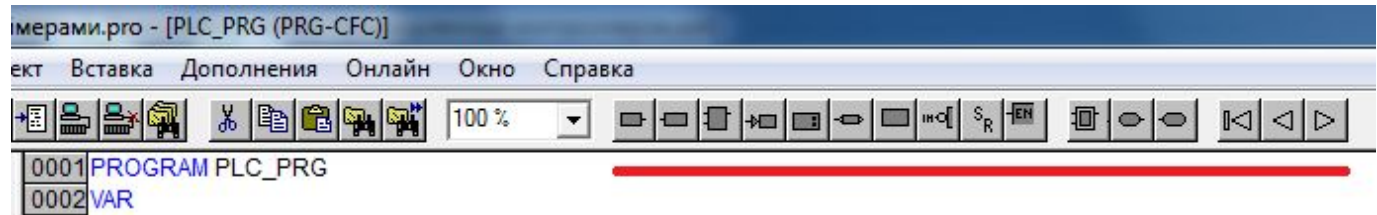
**In1, in1, IN1**

Воспринимается  
как одна переменная

~~**TON, 1\_step**~~

Так нельзя

# Базовые операции языка CFC



Вход – вызов переменной



Выход – запись переменной



Элемент – вызов оператора



Инверсия



Set/Reset – операции установки / сброса значения


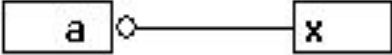
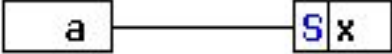



EN/ENO – добавление разрешающего входа

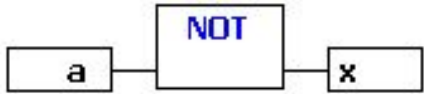
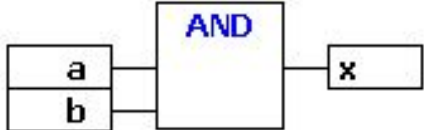
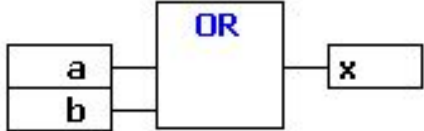



Комментарий

# Операторы присваивания

Оператор	CFC	ST
LD / ST		<code>X := A;</code>
LDN / ST		<code>X := NOT (A) ;</code>
LD / S		<code>IF A THEN X := TRUE; END_IF</code>
LD / R		<code>IF A THEN X := FALSE; END_IF</code>

# Логические операторы

Оператор	CFC	ST
NOT		$X := \text{NOT}(A);$
AND		$X := A \text{ AND } B;$
OR		$X := A \text{ OR } B;$
XOR		$X := A \text{ XOR } B;$

Для набора программы с операторами (функциональными блоками)

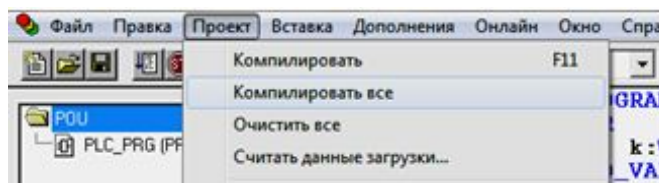
вставьте элемент  (по умолчанию AND) и исправьте функцию на нужную.



# Отладка и запуск проекта

## 1 Компиляция



можно использовать  
клавишу F11



## 2 Режим эмуляции -

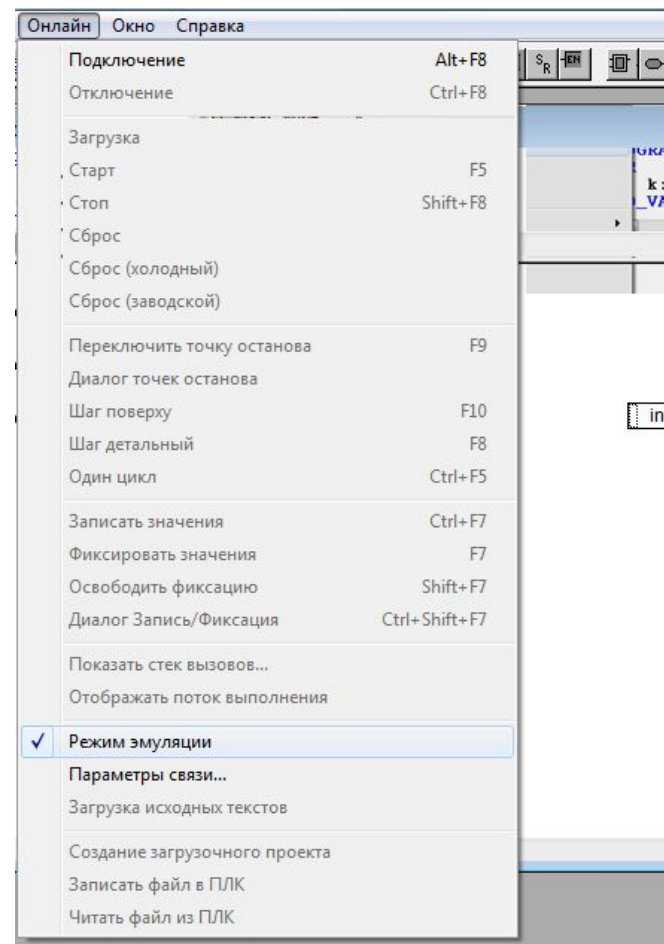
зайти в меню «Онлайн»  
и поставить галочку в пункте «Режим эмуляции».

## 3 Запуск проекта на исполнение

- «Онлайн» – «Подключение» или **Alt+F8** или иконка   
Зайти в меню «Онлайн» и выбрать пункт «Старт»  
(использовать клавишу **F5** или ).

### Внимание:

- выполнение операции «Подключение» приводит к автоматическому выполнению компиляции;
- если не установлен режим эмуляции, то проект будет записываться и выполняться в контроллере.
- после подключения все элементы программы выделяются «пожирнее» и появляется команда «RETURN» (ВОЗВРАТ), подчеркивающая, что программа будет выполняться циклически
- появятся отображения текущих значений переменных и значений на выходах функциональных блоков

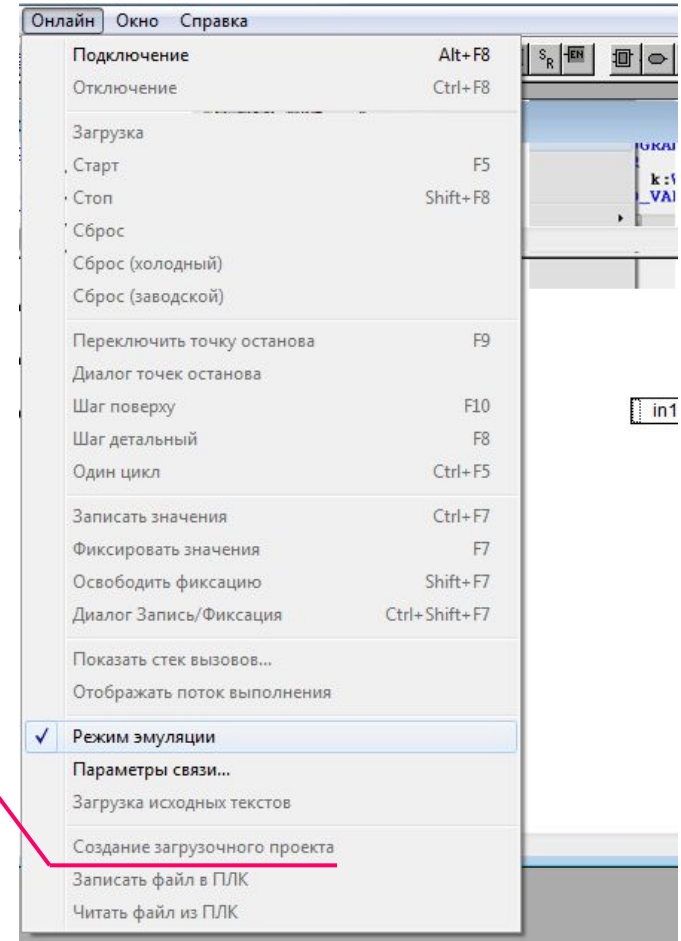


# Создание загрузочного проекта

“Онлайн”/

“Создание загрузочного проекта”

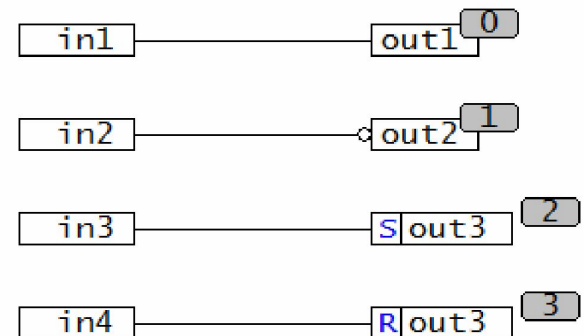
Используется для того, чтобы сделать код проекта автоматически загружаемым при перезапуске ПЛК. При перезапуске (включении) контроллера этот проект будет начинаться выполняться автоматически.



# Первый пример создания программы

```
□---PLC100.R
  □---Discrete input 8 bit[FIX]
    □--- AT %IB0.0: BYTE; (* 8 discrete inputs *) [CHANNEL (I)]
      .....in1 AT %IX0.0.0: BOOL; (* Bit 0 *)
      .....in2 AT %IX0.0.1: BOOL; (* Bit 1 *)
      .....in3 AT %IX0.0.2: BOOL; (* Bit 2 *)
      .....in4 AT %IX0.0.2: BOOL; (* Bit 3 *)
      ..... AT %IX0.0.4: BOOL; (* Bit 4 *)
      ..... AT %IX0.0.5: BOOL; (* Bit 5 *)
      ..... AT %IX0.0.6: BOOL; (* Bit 6 *)
      ..... AT %IX0.0.7: BOOL; (* Bit 7 *)
  □---Discrete output - relay[FIX]
    .....out1 AT %QX1.0: BOOL; (* relay *) [CHANNEL (Q)]
  □---Discrete output - relay[FIX]
    .....out2 AT %QX2.0: BOOL; (* relay *) [CHANNEL (Q)]
  □---Discrete output - relay[FIX]
    .....out3 AT %QX3.0: BOOL; (* relay *) [CHANNEL (Q)]
  □---Discrete output - relay[FIX]
  □---Discrete output - relay[FIX]
  □---Discrete output - relay[FIX]
  □---Special output[FIX]
```

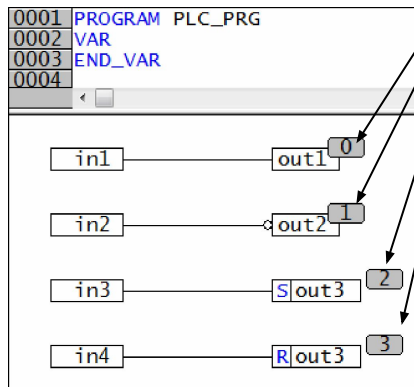
0001	PROGRAM PLC_PRG
0002	VAR
0003	END_VAR
0004	



# Поток данных

Порядок выполнения операций устанавливается сам по мере ввода программы. Если после внесения изменений он у Вас поменялся, то Вы вновь можете устанавливать порядок в соответствии с потоком данных (слева-направо/сверху-вниз)

Порядок выполнения программы  
(поток данных)



SoDeSys - (Untitled)\* - [PLC\_PRG (PRG-CFC)]

Файл Правка Проект Вставка Дополнения Онлайн Окно Справка

100 %

POU  
PLC\_PRG (PRG)

```
0001 PROGRAM PLC_PRG
0002 VAR
0003   a1: BOOL;
0004   a2: BOOL;
0005 END_VAR
0006
```

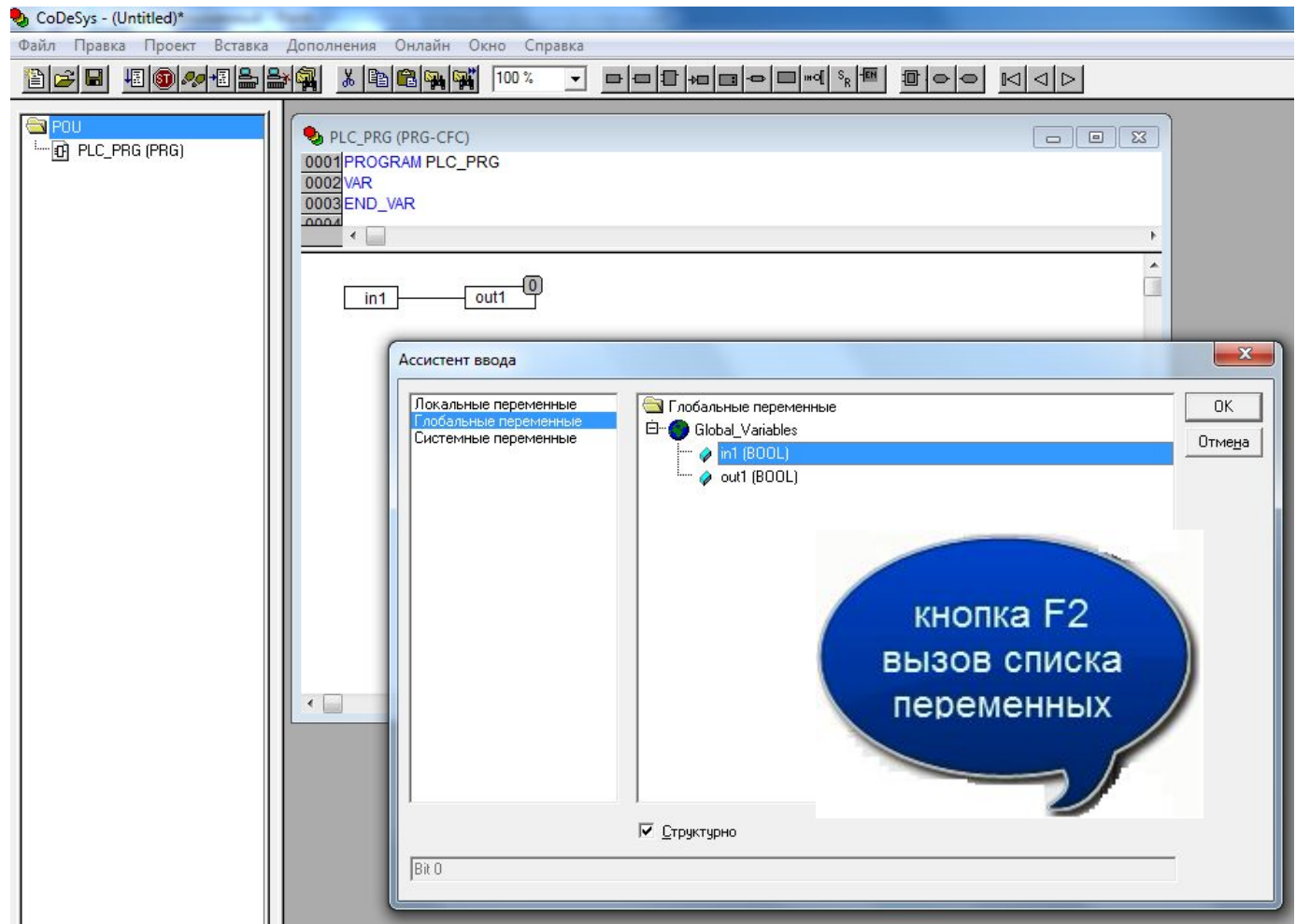
a1 — a2

Элемент	Ctrl+B
Вход	Ctrl+I
Выход	Ctrl+U
Переход	Ctrl+J
Метка	Ctrl+L
Возврат	Ctrl+R
Комментарий	Ctrl+K
Вход блока	Ctrl+A
Вход макро	
Выход макро	
Инверсия	Ctrl+N
Set/Reset	Ctrl+T
Соединяющий маркер	Ctrl+M
EN/ENO	Ctrl+E
<b>Порядок</b>	<b>Показать порядок</b>
Создать макрос	Упорядочить топологически
Показать содержимое макроса	<b>В соответствии с потоком данных</b>
Развернуть содержимое макроса	Порядок: выше
Вернуться на предыдущий уровень	Порядок: ниже
Перейти на верхний уровень	Порядок: в начало
Редактировать POU	Порядок: в конец

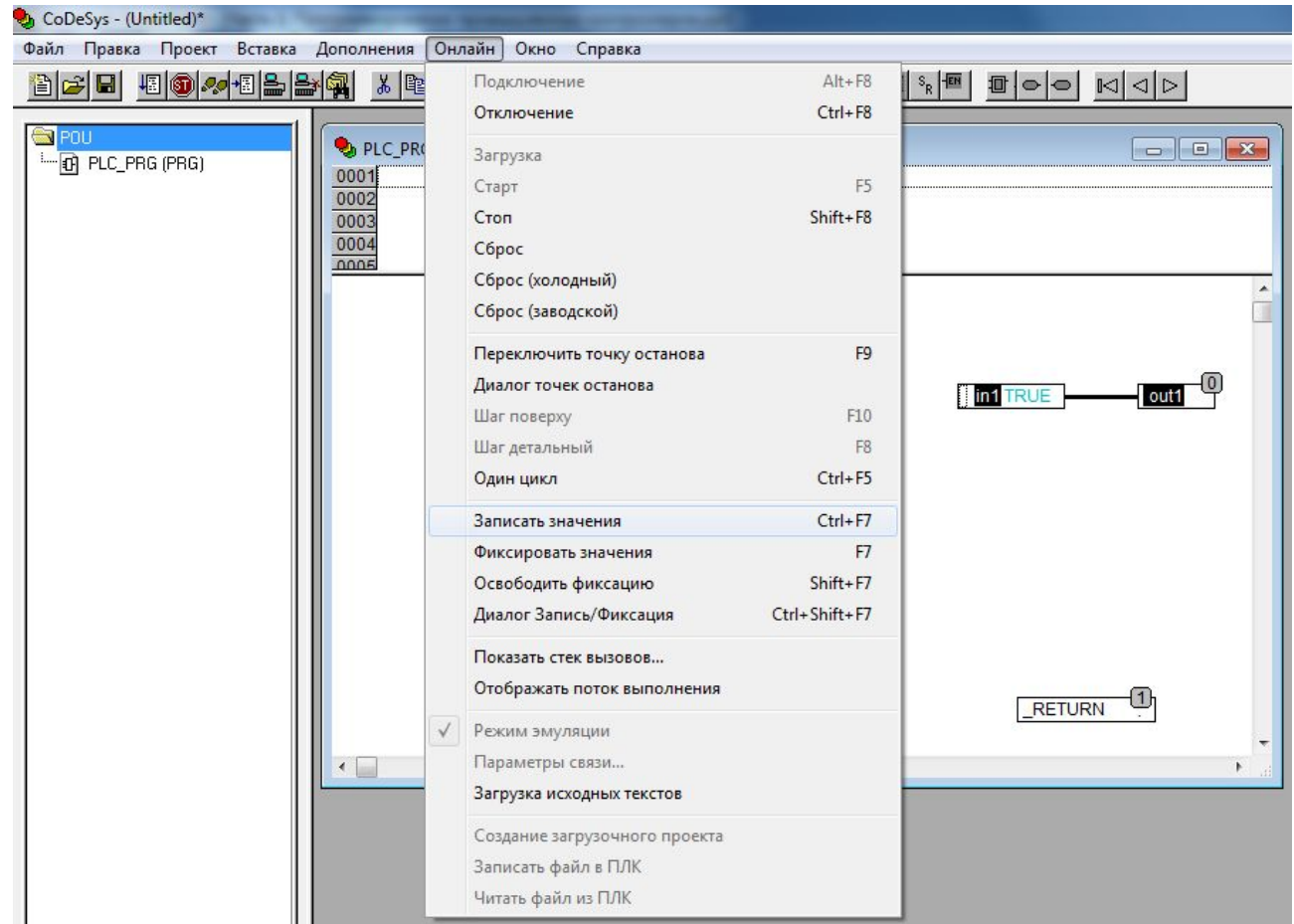
Alt+Enter

# Работа с Ассистентом ввода

Если переменные описаны Вами в конфигурации контроллера, то присвоить их имена программным компонентам поможет Ассистент ввода.

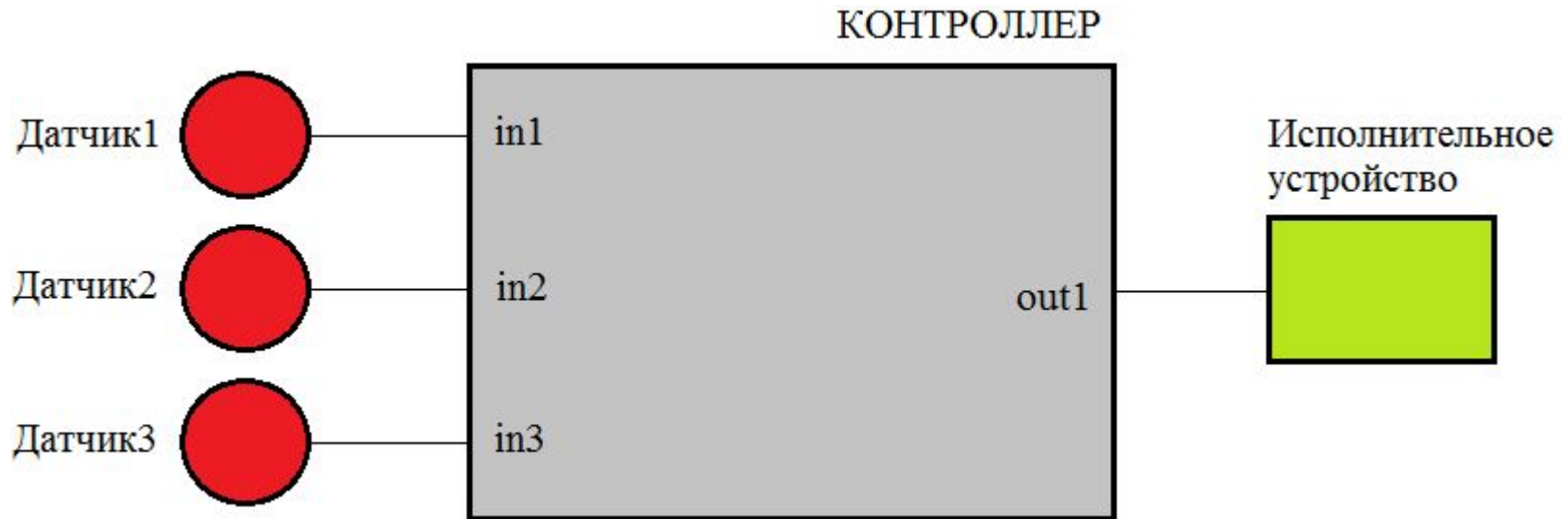


При отладке проекта Вы можете записывать текущие значения переменных или фиксировать их на все время выполнения программы.



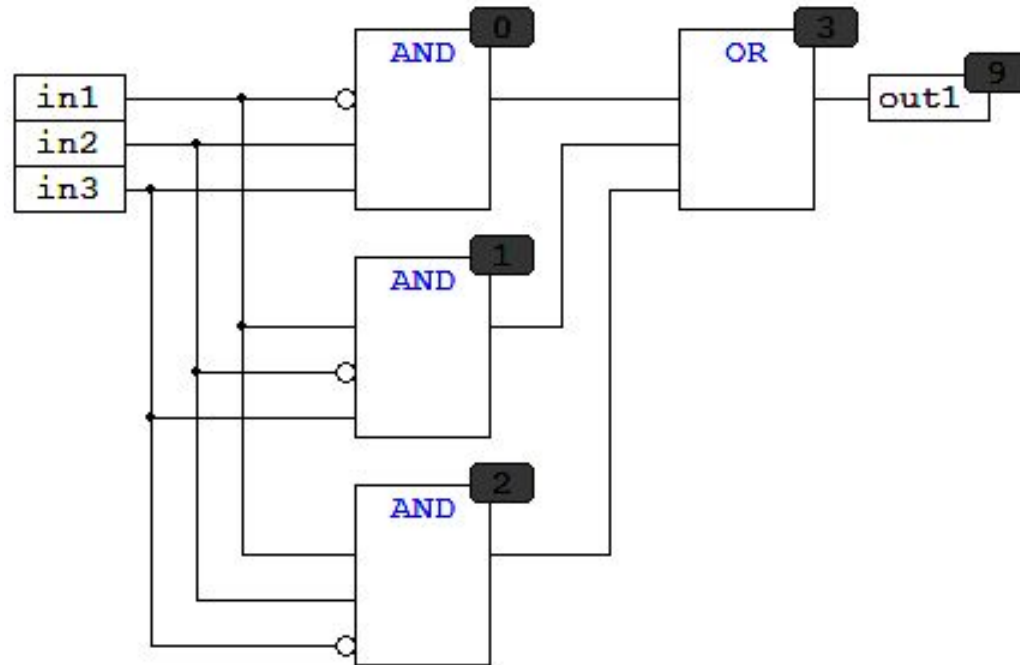
# ЗАДАНИЕ: Создание дешифратора

Система управления включает в себя три датчика. При срабатывании любых двух (и только двух) датчиков должен активизироваться один из дискретных выходов контроллера.





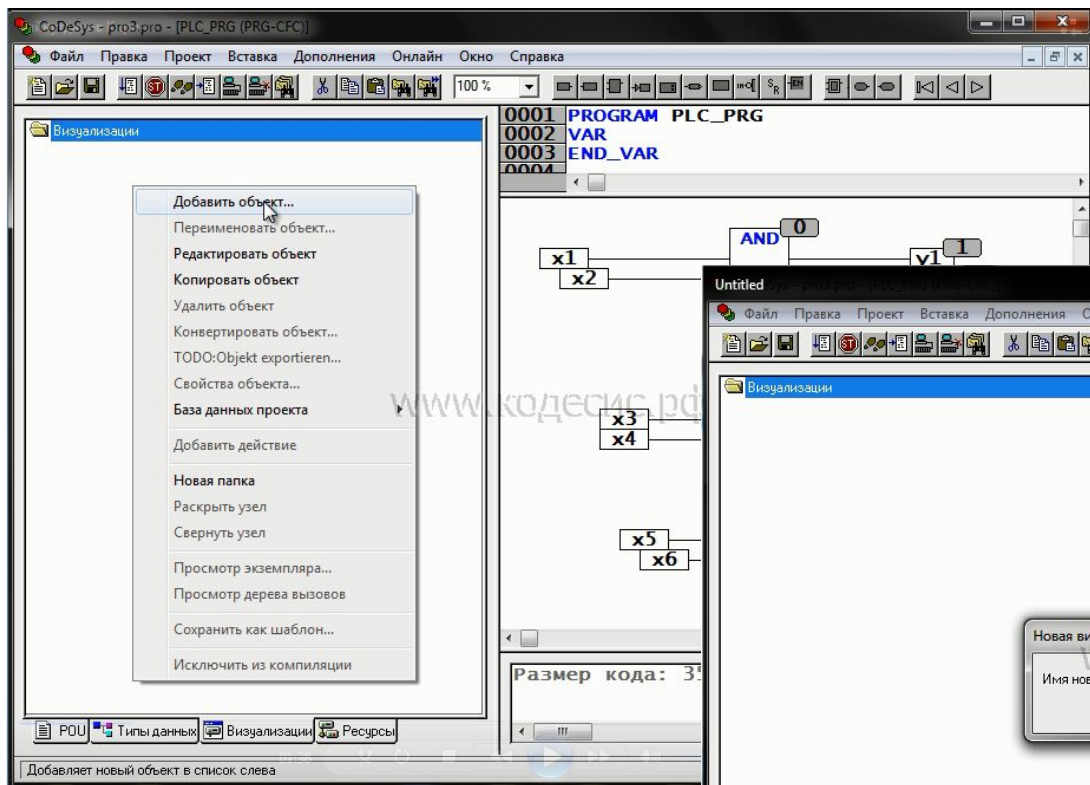
# Реализация дешифратора



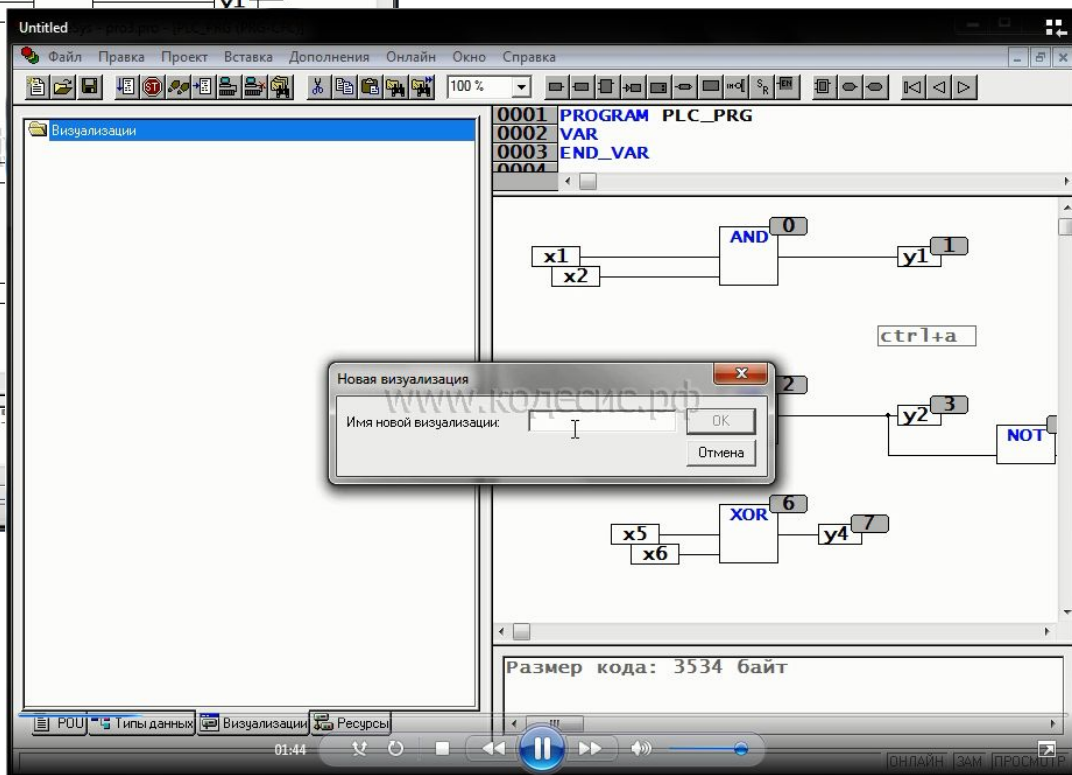
# Визуализация

В CoDeSys V2.3 используется в процессе отладки или при представлении проекта

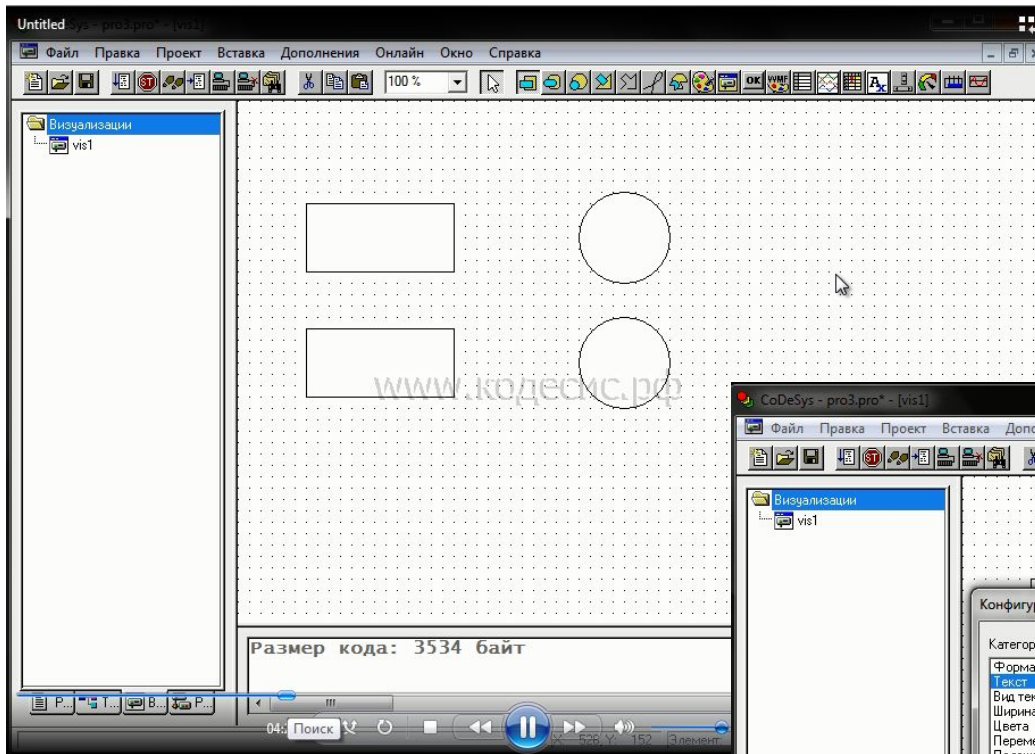
Добавление объекта визуализации



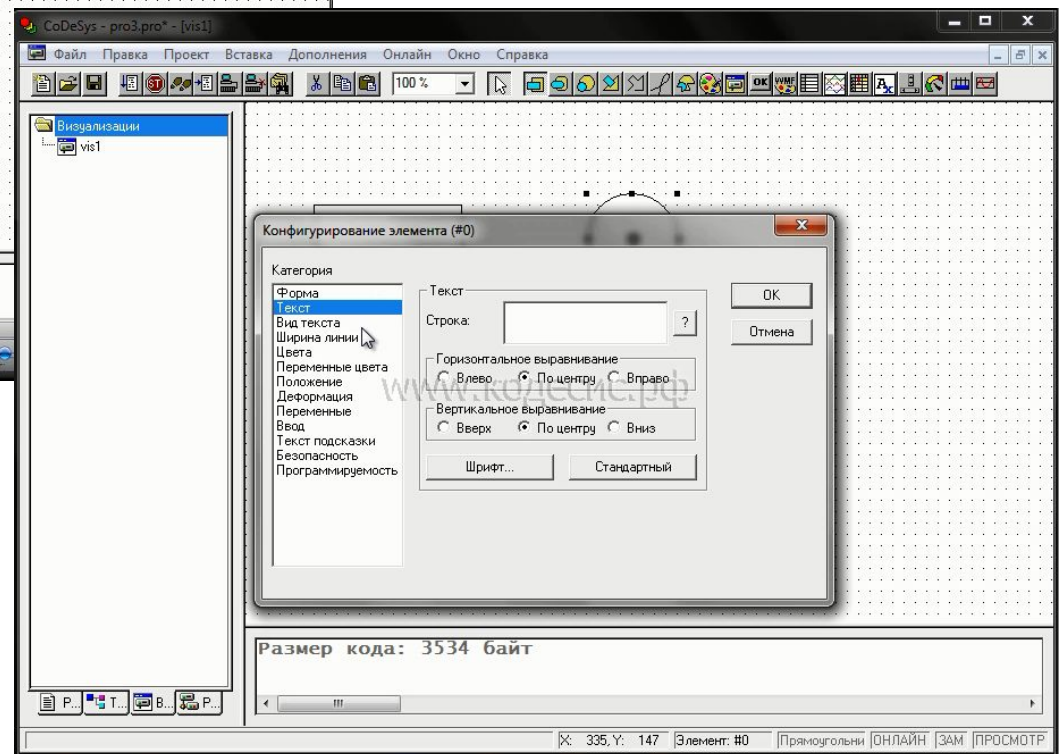
Задание имени объекту визуализации



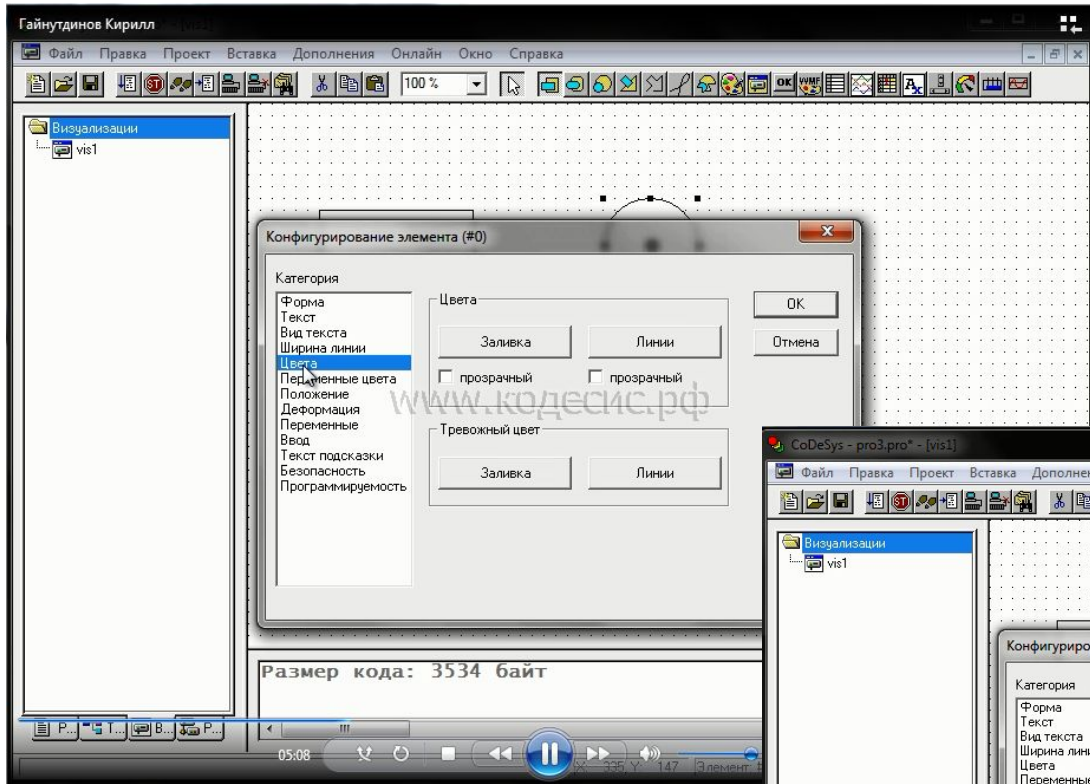
## Создание графических примитивов



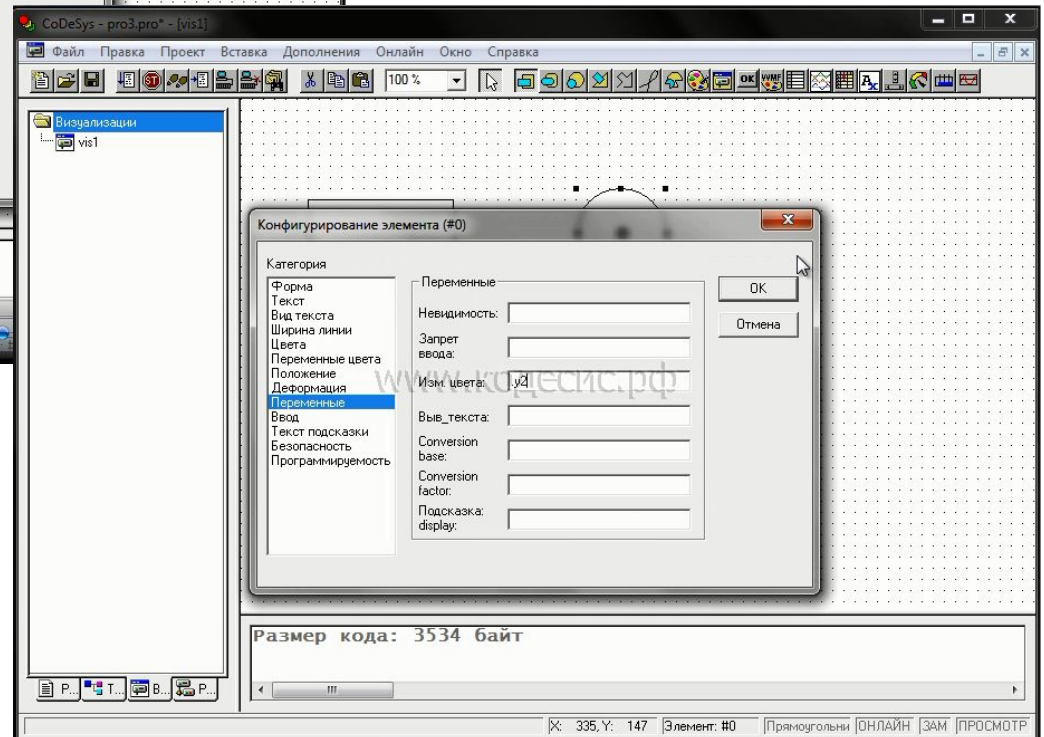
Добавление в примитив текста

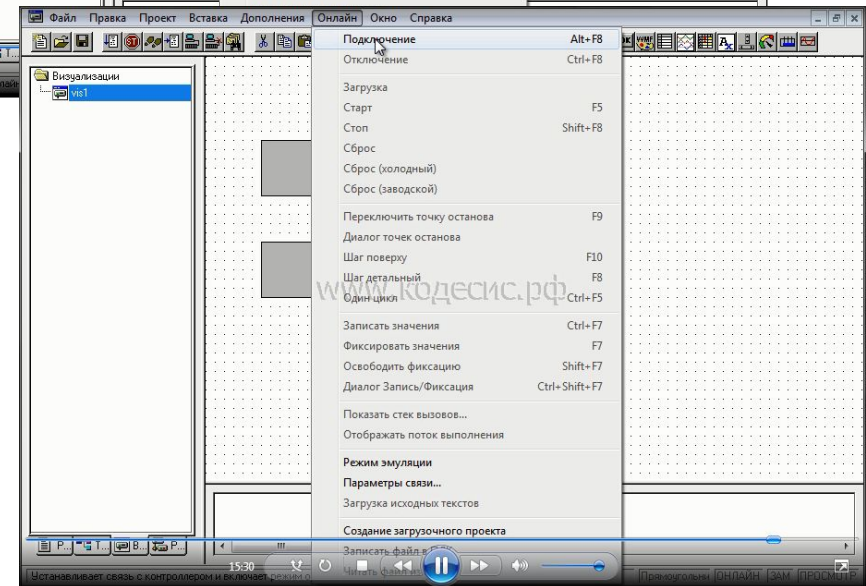
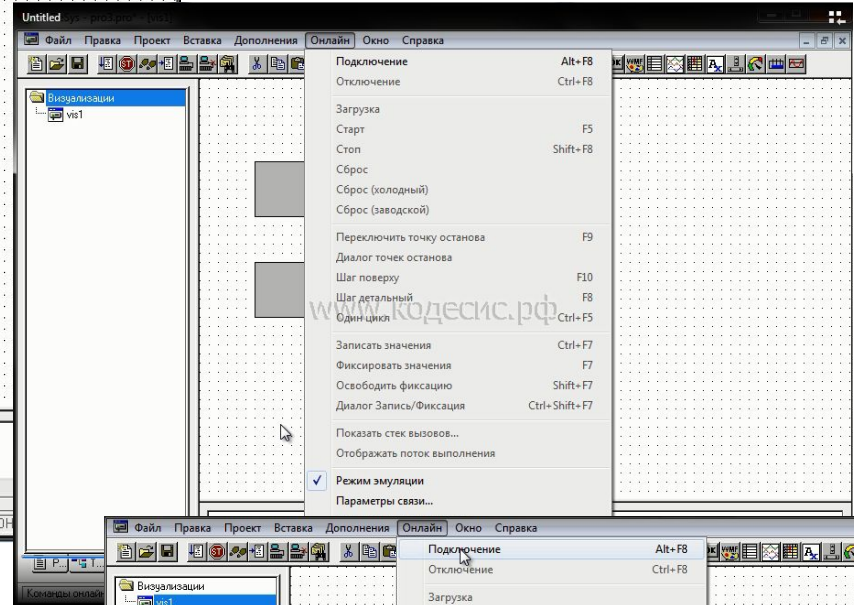
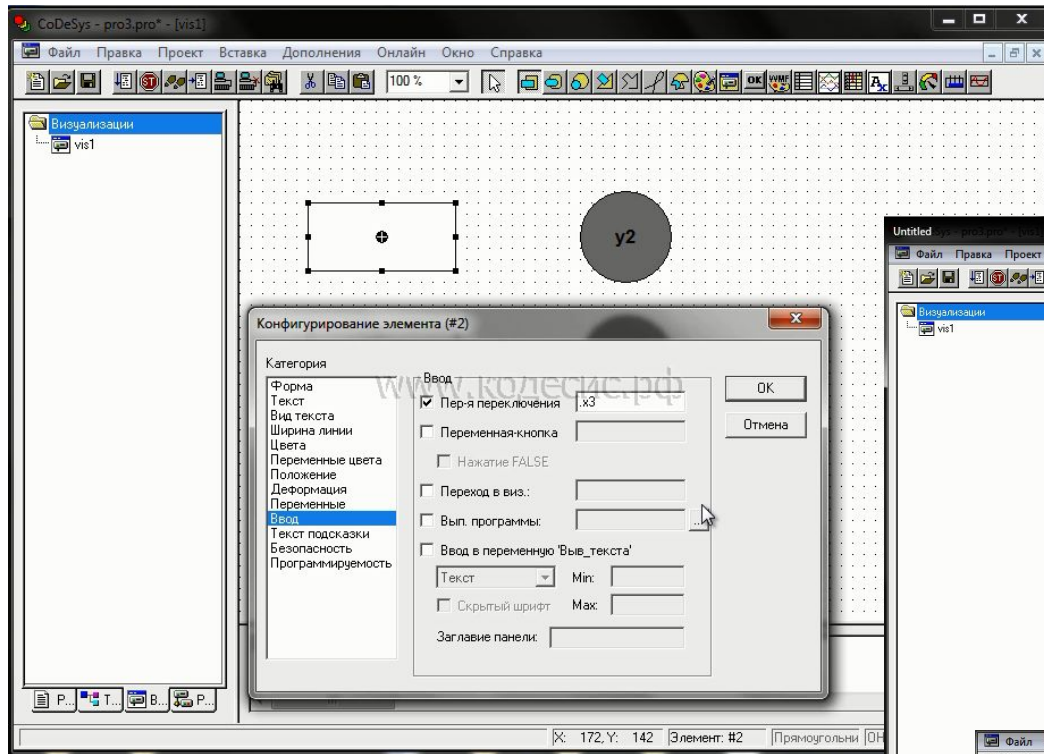


## Задание основного и «тревожного» цвета примитива



Связывание примитива с переменной программы





Связывание примитива (значения, соответствующей входной переменной) с действием

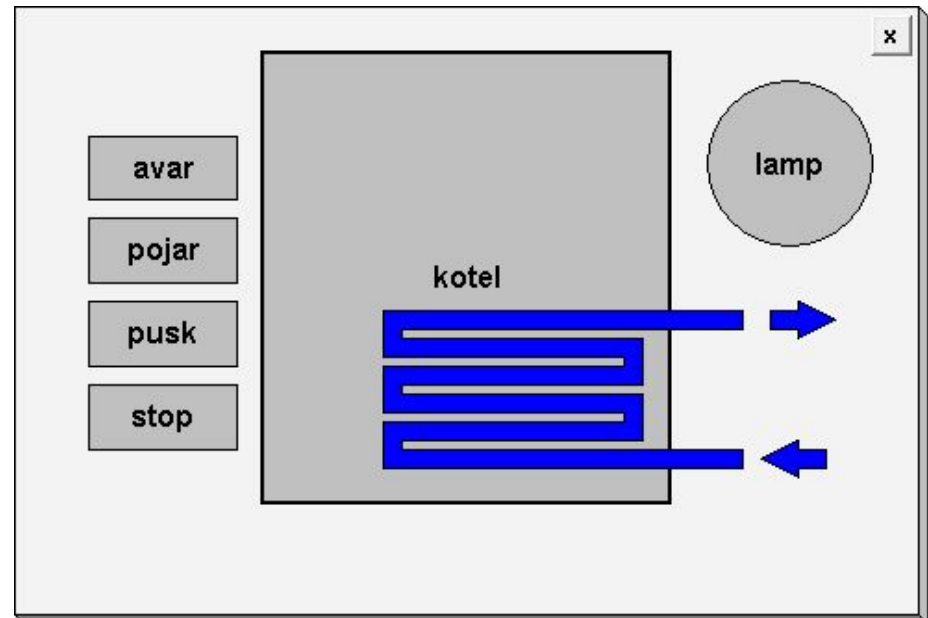
Визуализацию Вы можете запускать как в режиме «Эмуляции», так и в режиме «Онлайн».



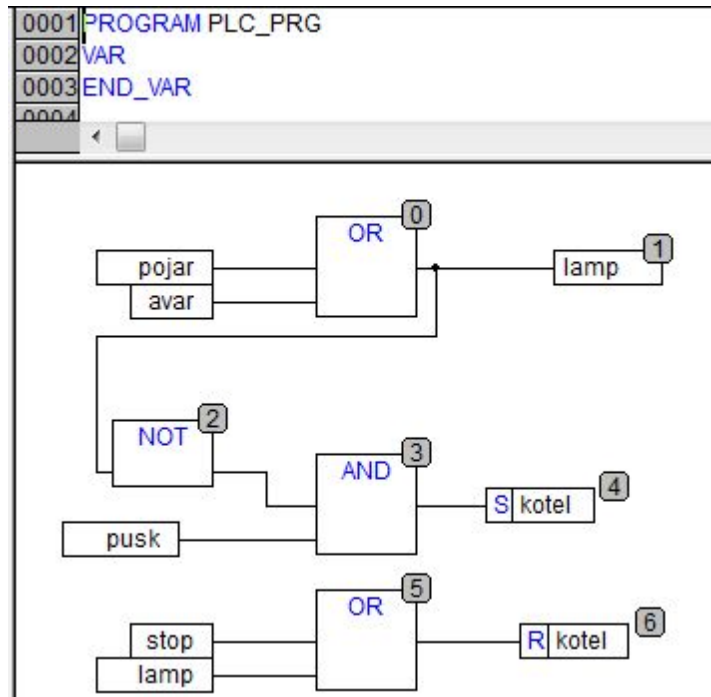
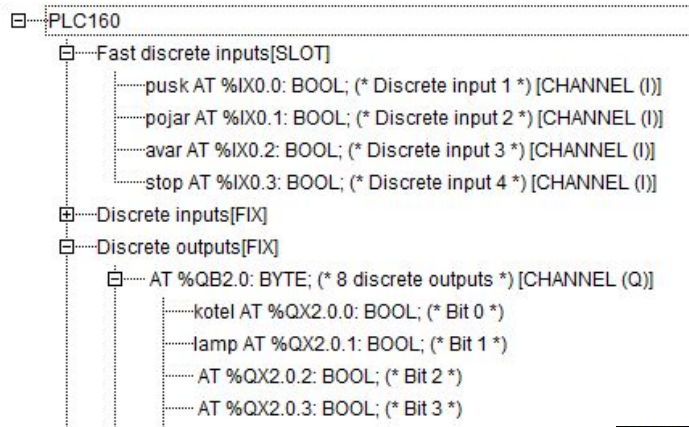
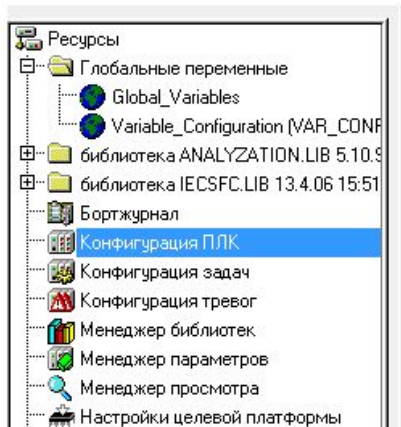
# Пример: управление котлом

Необходимо реализовать:

- Включение сигнализации при возникновении любой из аварий.
- Отключение котла при возникновении любой из аварий.
- Включение котла с кнопки, при условии отсутствия аварий.
- Отключение котла с кнопки.

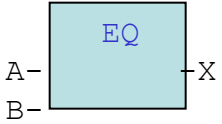
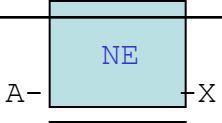
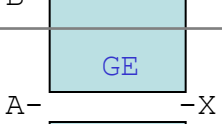
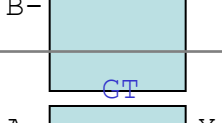
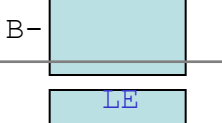
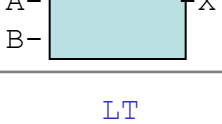


# Решение примера с котлом



# Операторы сравнения

Используются для работы со всеми типами данных

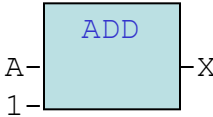
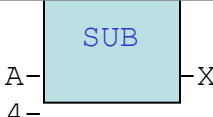


Оператор	CFC	ST
<b>EQ</b>		<b>X := (A = B) ;</b>
<b>NE</b>		<b>X := (A &lt;&gt; B) ;</b>
<b>GE</b>		<b>X := (A &gt;= B) ;</b>
<b>GT</b>		<b>X := (A &gt; B) ;</b>
<b>LE</b>		<b>X := (A &lt;= B) ;</b>
<b>LT</b>		<b>X := (A &lt; B) ;</b>

При определении типа блока (алгебраической операции) вместо аббревиатуры можно использовать простые символы: =, <>, >=, >, <=, <.



# Арифметические операторы

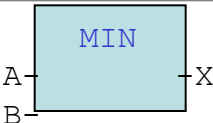
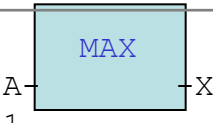
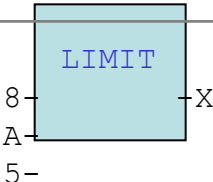


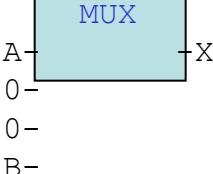
Выполняют алгебраические операции над целыми числами (INT, WORD) и числами с плавающей запятой (REAL)

Оператор	CFC	ST
<b>ADD</b>		$X := A + 1;$
<b>SUB</b>		$X := A - 4;$
<b>MUL</b>		$X := A * B;$
<b>DIV</b>		$X := A / 8;$

При определении типа блока (алгебраической операции) вместо аббревиатуры можно использовать простые символы: +, -, \*, /.

# Операторы выбора

- Предназначены для ограничения и выбора значений
- Используются с любыми типами данных

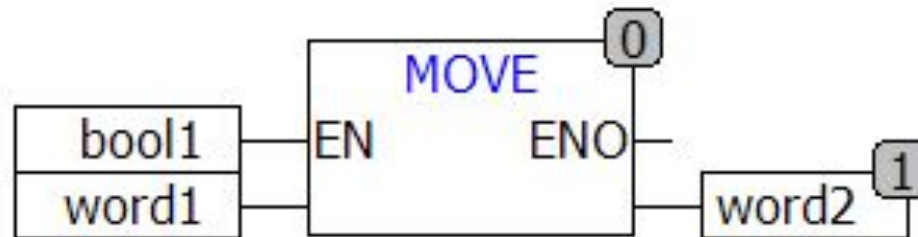
Оператор	CFC	ST
<b>MIN</b>		$X := \text{MIN}(A, B);$
<b>MAX</b>		$X := \text{MAX}(A, 1);$
<b>LIMIT</b>		$X := \text{LIMIT}(-8, A, 5);$  <i><math>X = -8</math> if <math>A &lt; -8</math></i> <i><math>X = 5</math> if <math>A &gt; 5</math></i>
<b>SEL</b>	 	$X := \text{SEL}(A, 10, B);$  <i><math>X = 10</math> if <math>A</math> is <b>FALSE</b></i> <i><math>X = B</math> if <math>A</math> is <b>TRUE</b></i>
<b>MUX</b>		$X := \text{MUX}(A, 0, 10, B);$  <i><math>X = 0</math> if <math>A</math> is <b>0</b></i> <i><math>X = 10</math> if <math>A</math> is <b>1</b></i> <i><math>X = B</math> if <math>A</math> is <b>2</b></i>

# Оператор move и разрешающий вход EN

**Move** присваивает значение слева переменной справа.

Используются с любыми типами данных.

При появлении значения *TRUE* на входе **En** операция выполняется, иначе операция игнорируется.



Вход EN используется с любыми операторами и ROU.

# Описание локальных переменных

```
0001 PROGRAM PLC_PRG
0002 VAR
0003
0004     b1: BOOL;
0005     k1: BOOL;
0006     k2: BOOL;
0007     r1: REAL := 100;    (*Уставка по температуре*)
0008 END_VAR
0009 VAR RETAIN
0010     w1: WORD;
0011 END_VAR
0012
```

Ассистент ввода – F2

Объявление переменной

Объявление переменной

Класс	Имя	Тип
VAR	in1	BOOL

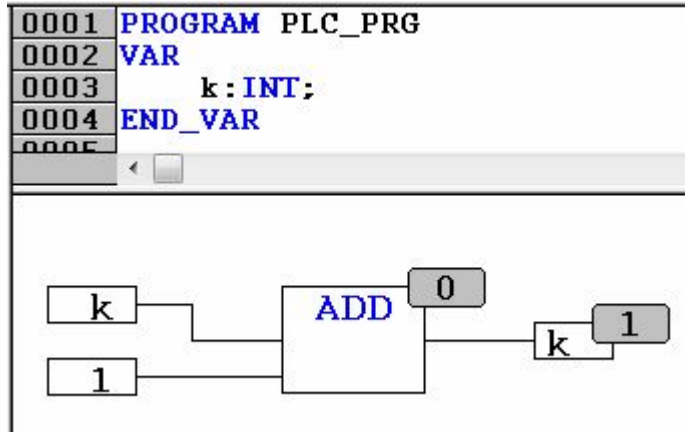
Нач. значение: [ ] Адрес: [ ]

Комментарий: Входная переменная

CONSTANT  
 RETAIN  
 PERSISTENT

OK Отмена

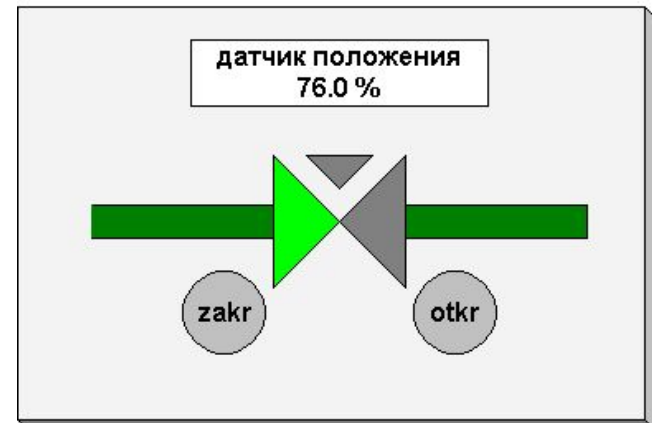
# Пример



# Пример: управление клапаном

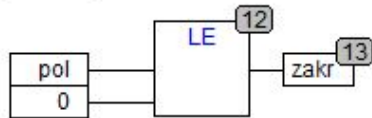
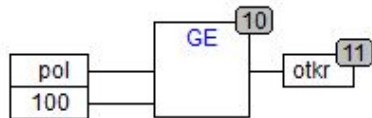
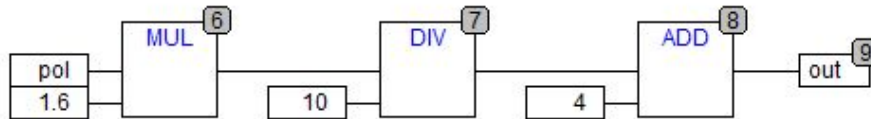
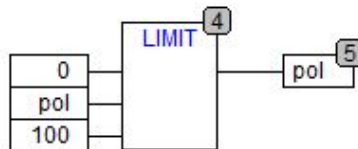
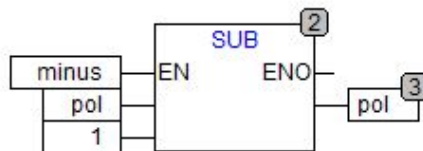
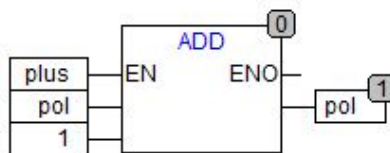
Необходимо реализовать:

1. Плавное увеличение и уменьшение степени открытия клапана (**pol**) с внешних кнопок (**plus** или **minus**);
1. Выдачу управляющего сигнала 4-20 мА (**out**) с выхода ПЛК на клапан;
2. Отображение степени открытия клапана (**pol**) в процентах;
3. Сигнализацию о достижении концевых положений (**zacr** и **otkr**);



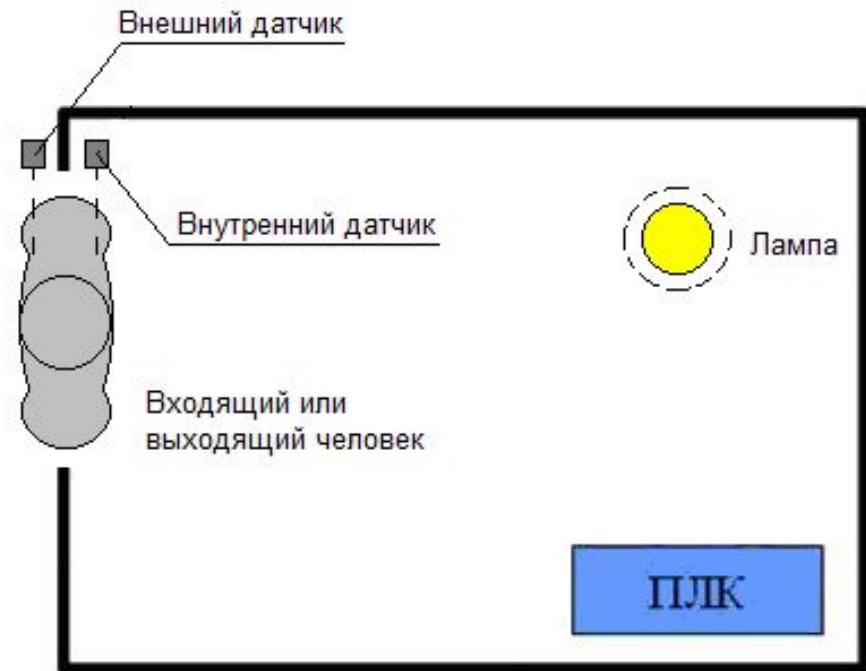
# Решение примера с клапаном

```
0001 PROGRAM PLC_PRG
0002 VAR
0003   out: REAL;
0004 END_VAR
0005 VAR RETAIN
0006   pol: REAL; (*положение клапана*)
0007 END_VAR
```



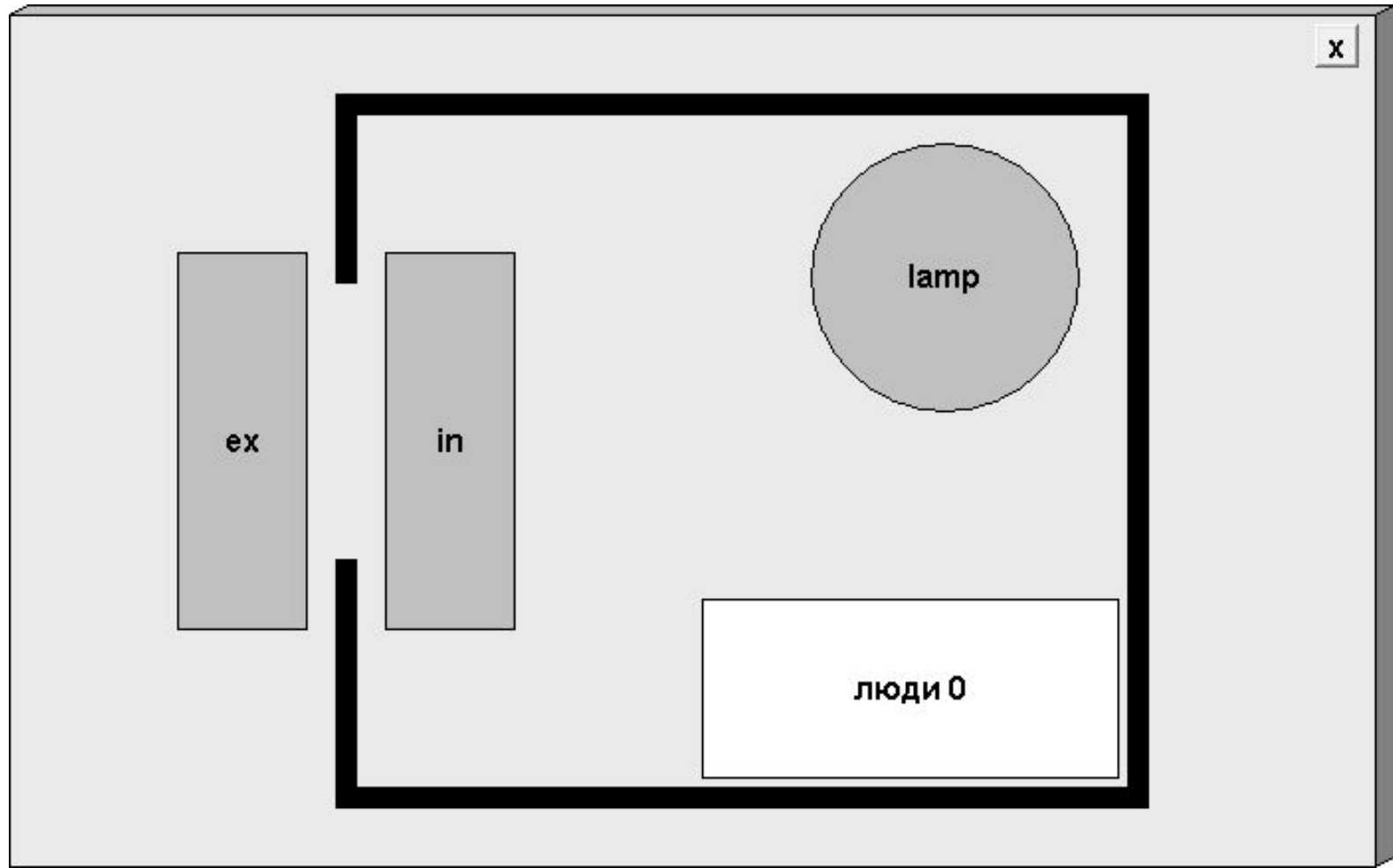
# Пример: управление светом в комнате

- На входе установлены два дискретных датчика: один снаружи комнаты, другой внутри. Человек входя или выходя, перекрывает собой по ширине оба датчика. Когда срабатывает сначала внешний датчик, затем внутренний, это означает, что человек зашел в комнату. Когда срабатывает сначала внутренний датчик, затем внешний, это означает, что человек вышел из комнаты.
- Необходимо определять количество людей, находящихся в комнате. Пока в комнате есть хотя бы один человек, свет должен быть включен. Если из комнаты вышел последний человек свет должен быть выключен.

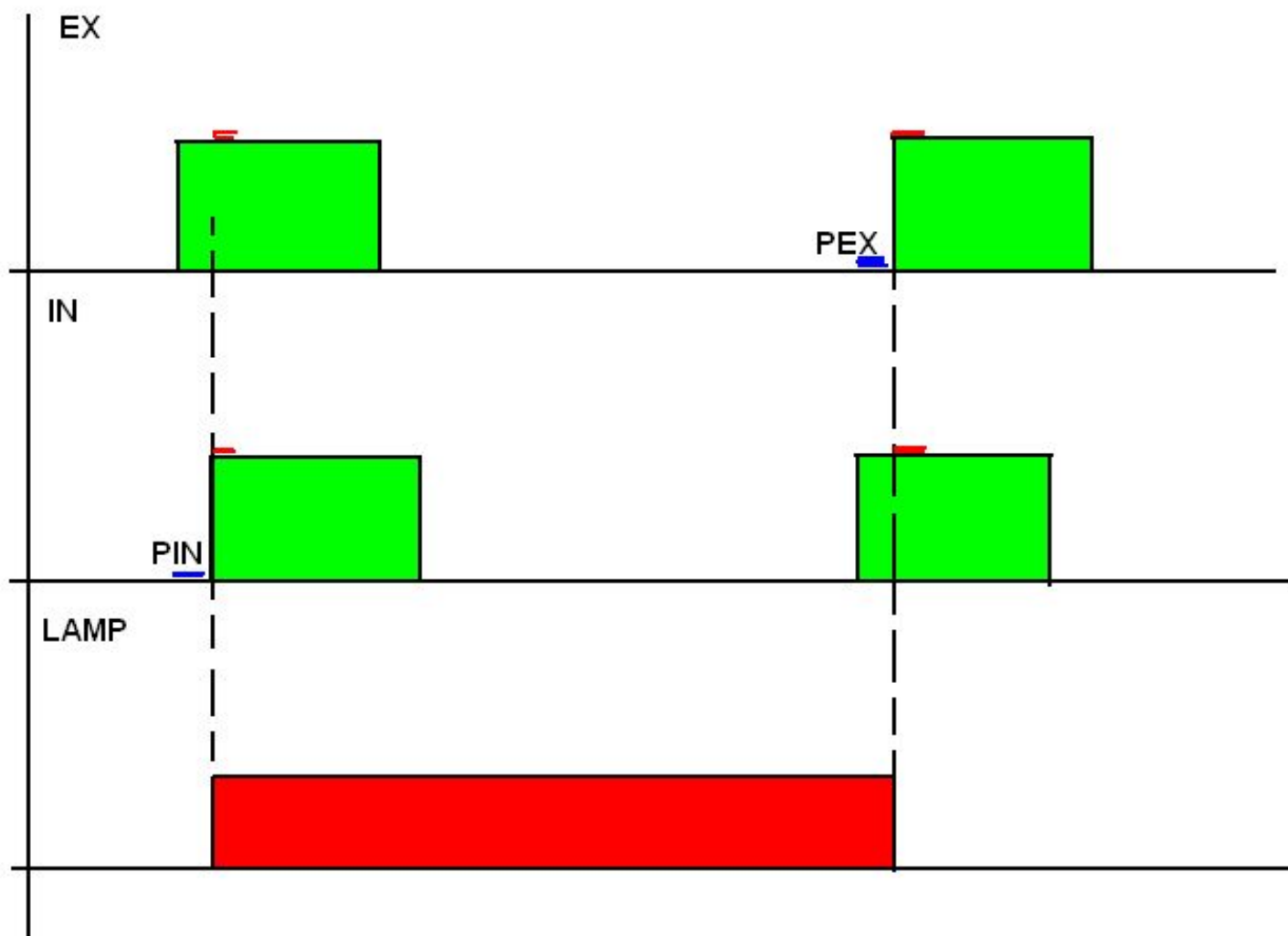




# Задача 1: Свет



# Временная диаграмма



# Решение примера:

The screenshot displays the CoDeSys software interface for a PLC program named "Свет.pro" (Light.pro). The main window shows the following components:

- Variable Declaration:**

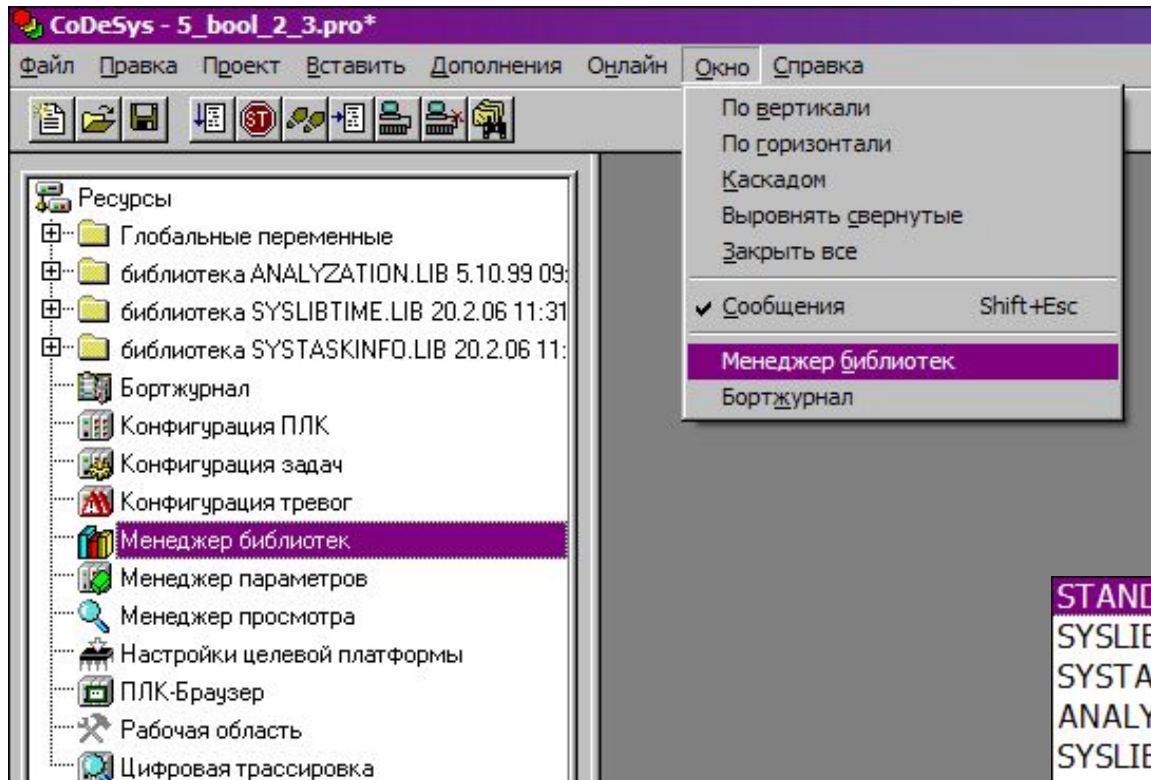
```
0001 PROGRAM PLC_PRG
0002 VAR
0003   ludi:WORD;
0004   pin,pex:BOOL;
0005 FND VAR
```
- Ladder Logic Network Diagram:**
  - Network 0:** An AND gate with inputs "ex", "in", and "pin". Its output is connected to the EN input of an ADD gate. The ADD gate has a constant input of "1" and its ENO output is connected to the "ludi" variable.
  - Network 3:** An AND gate with inputs "ex", "in", "pex", and "lamp". Its output is connected to the EN input of a SUB gate. The SUB gate has a constant input of "1" and its ENO output is connected to the "ludi" variable.
  - Network 6:** A GT (Greater Than) gate with input "ludi" set to "0" and output connected to the "lamp" variable.
  - Network 8:** A simple assignment where "in" is connected to "pin".
  - Network 9:** A simple assignment where "ex" is connected to "pex".

The status bar at the bottom indicates the library path: "Загрузка библиотеки 'C:\Program Files\3S Software\CoDeSys V2.3\Library\IEC".

# Предопределенные функциональные блоки (Библиотеки)

- Библиотека состоит из объектов, которые могут быть использованы в различных проектах.
- В библиотеку могут входить программные компоненты, списки переменных, визуализации и пр.
- Пользователь может создавать и использовать собственные библиотеки.
- Стандартные библиотеки **Standard.lib** и **Util.lib**

# Подключение библиотек

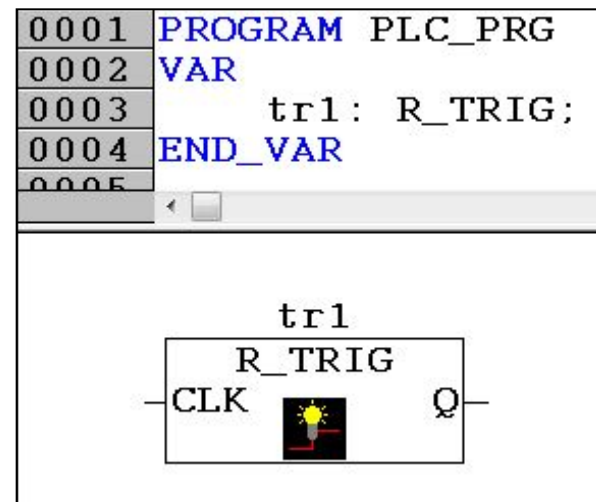
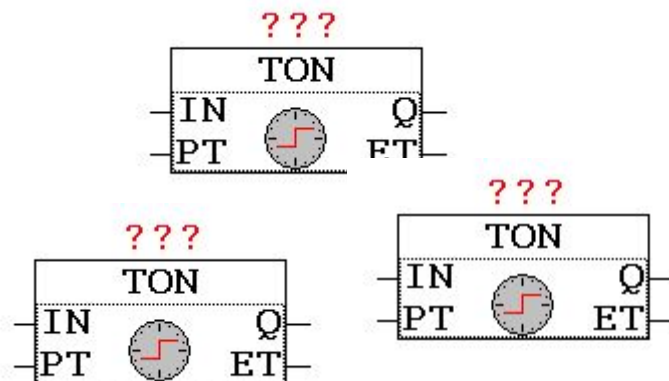


STANDARD.LIB 4.10.05 11:14:46  
SYSLIBTIME.LIB 20.2.06 11:31:16  
SYSTASKINFO.LIB 20.2.06 11:31:38  
ANALYZATION.LIB 5.10.99 09:05:06  
SYSLIBCALLBACK.LIB 20.2.06 11:31:08

Добавить библиотеку... Ins  
Удалить Del  
Свойства... Alt+Enter

# Работа с библиотечными функциональными блоками

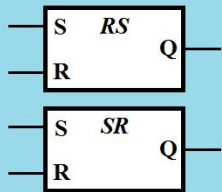
- Библиотека хранит исходники ФБ, в проекте используются экземпляры этих исходных ФБ.
- Один и тот же ФБ в проекте может иметь несколько экземпляров, работающих одинаково и независимо друг от друга.
- Каждый экземпляр должен иметь уникальное имя (как и любая переменная).
- Экземпляр ФБ объявляется аналогично тому, как это делается с локальными переменными.
- Допускается в рамках одного цикла ПЛК несколько раз обращаться одному и тому же экземпляру ФБ. При этом в конце цикла учитываются изменения, произведенные при последнем вызове экземпляра.



# Стандартная библиотека Standard.lib

## Триггеры

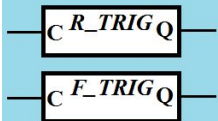
### RS и SR триггеры с доминантой выключения и включения



Входные переменные		Выходные переменные	
имя	тип	имя	тип
Set	Bool	Q	Bool
Reset	Bool		

RS (SR) триггер имеет два устойчивых состояния. Вход Set включает выход, вход Reset выключает выход. При одновременном воздействии на входы доминантным является вход Reset (Set).

### R\_TRIG и F\_TRIG детекторы переднего и заднего фронта

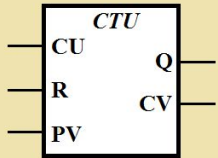


Входные переменные		Выходные переменные	
имя	тип	имя	тип
CLK	Bool	Q	Bool

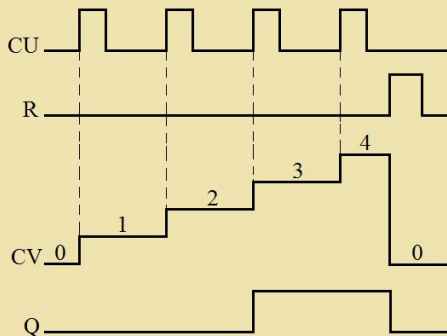
R\_TRIG (F\_TRIG) генерирует единичный импульс длительностью в один цикл контроллера) по переднему (заднему) фронту входного сигнала CLK



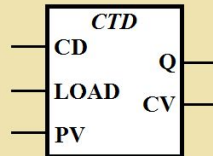
## CTU - инкрементный



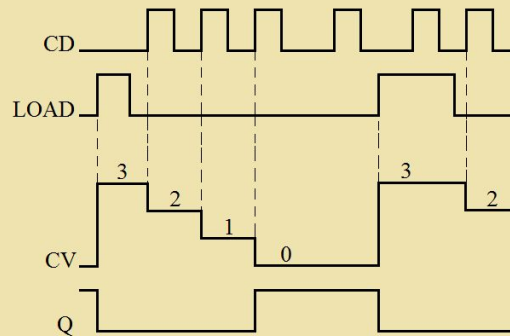
По каждому фронту на входе CU значение на выходе CV увеличивается на 1. Выход Q устанавливается в "1", когда счетчик достигнет или превысит порог PV. RESET устанавливает счет и обнуляет счетчик.



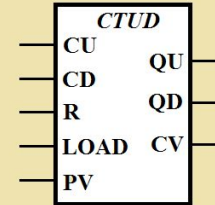
## CTD - декрементный



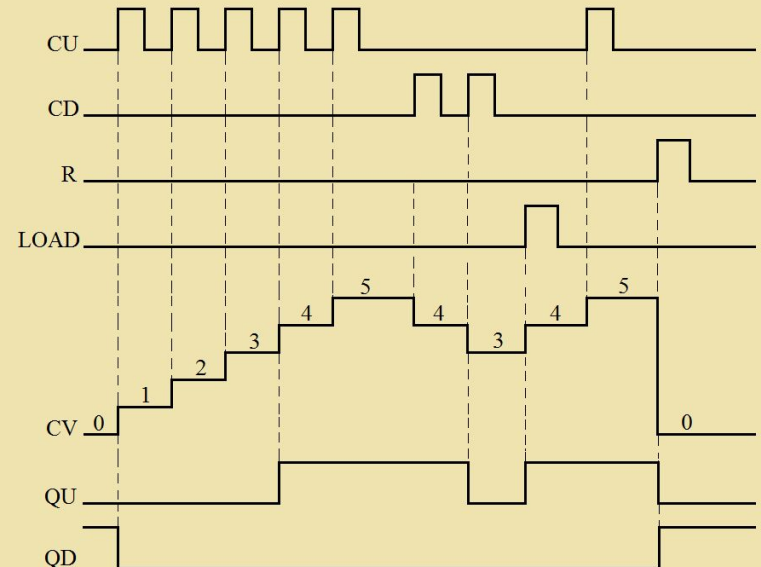
По каждому фронту на входе CD значение на выходе CV уменьшается на 1. Выход Q устанавливается в "1", когда счетчик достигнет значения 0. CV загружается новым значением PV по входу LOAD.



## CTUD - инкрементный/декрементный



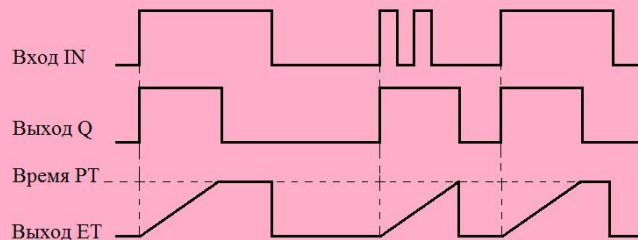
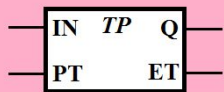
По фронту на входе CU счетчик CV увеличивается на 1. По фронту на входе CD счетчик CV уменьшается на 1. RESET сбрасывает CV в 0. LOAD загружает CV значением равным PV. QU равен "1", если  $CV \geq PV$ . QD равен "1", если CV равно 0.





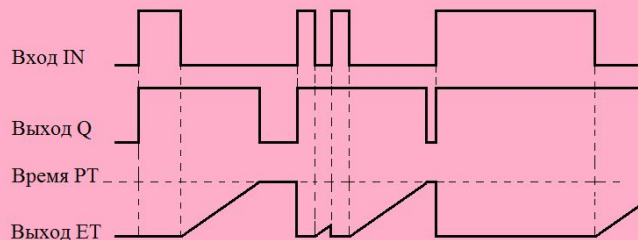
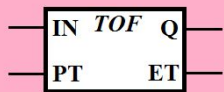
# Standard.lib Таймеры

## TP - таймер-генератор импульса



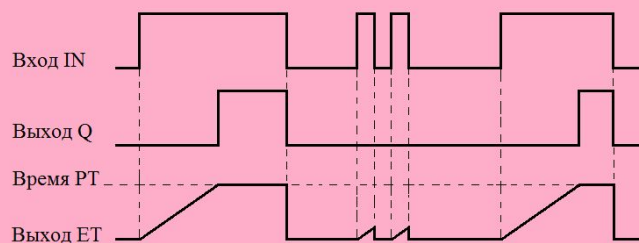
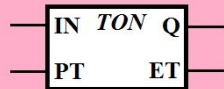
Запуск таймера осуществляется по фронту на входе IN. При этом выход Q устанавливается в "1". После запуска таймер не реагирует на изменение IN. PT задает длительность формируемого импульса. ET отсчитывает прошедшее время. При достижении ET значения PT отсчет времени останавливается и выход Q сбрасывается в "0".

## TOF - таймер с задержкой выключения



По фронту на входе IN выход Q устанавливается в "1". Сброс отсчета ET и начало отсчета времени происходит по каждому спаду IN. Выход Q будет сброшен через заданное время PT после спада на входе. Если во время отсчета сигнал IN будет установлен в "1", то отсчет времени приостанавливается.

## TON - таймер с задержкой включения

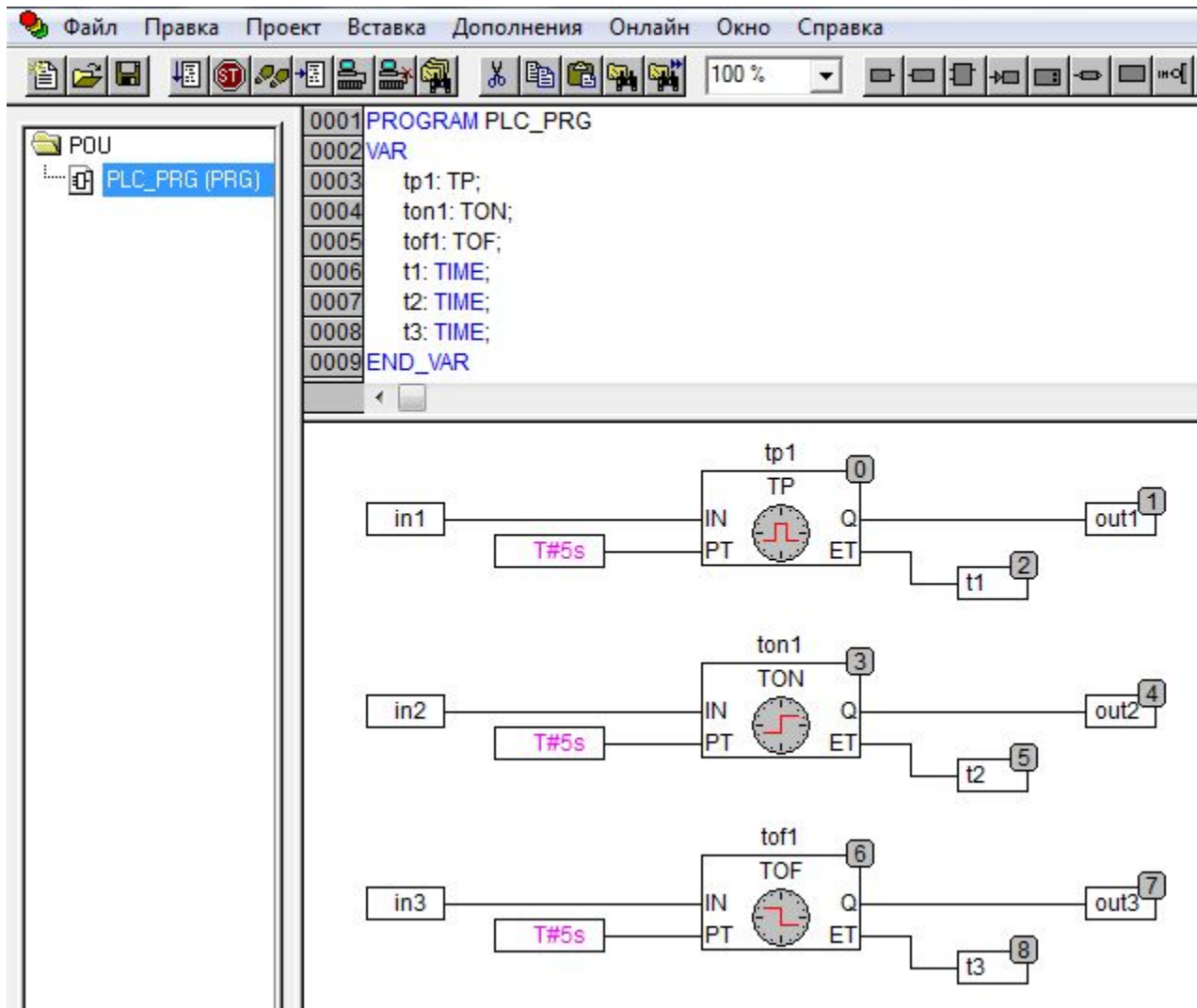


По фронту на входе IN выполняется обнуление счетчика времени и задается новый отсчет времени. Выход Q будет установлен в "1" через заданное время PT, если сигнал IN будет продолжаться оставаться в "1". Спад IN останавливает отсчет времени и сбрасывает выход Q в "0".

# Временные типы данных

<b>Тип</b>	<b>Описание</b>	<b>Пример</b>
<b>TIME</b>	Используются для выражения интервалов времени	<b>T1:=T#5h45m10s9ms</b> <b>T2:=T#100ms</b>
<b>DATE</b>	Используются для выражения даты	<b>DATE#1996-05-06</b> <b>d#1972-03-29</b>
<b>TIME_OF_DAY</b>	Используются для выражения времени суток	<b>TIME_OF_DAY#15:36:30.123</b> <b>tod#00:00:00</b>
<b>DATE_AND_TIME</b>	Используются для выражения даты и времени суток	<b>DATE_AND_TIME#1996-05-06-15:36:30</b> <b>dt#1972-03-29-00:00:00</b>

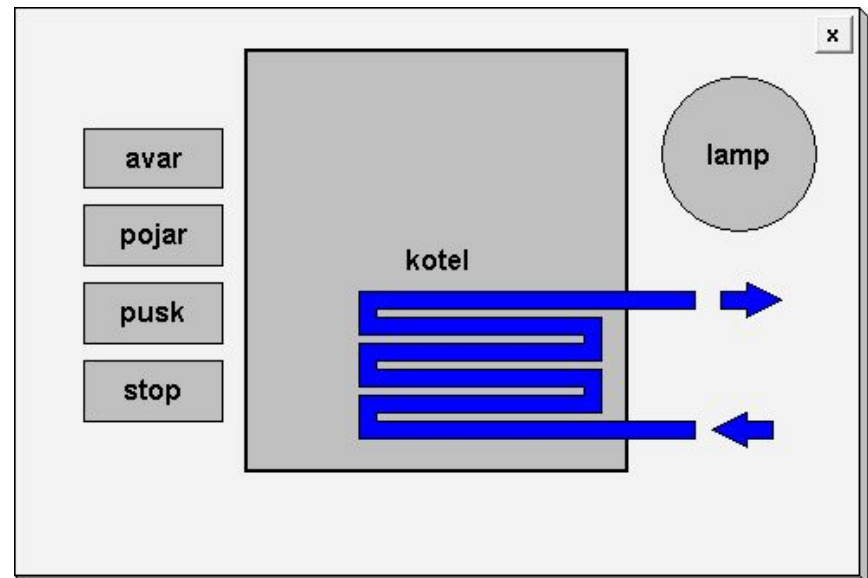
# Работа с таймерами



# Пример: управление котлом

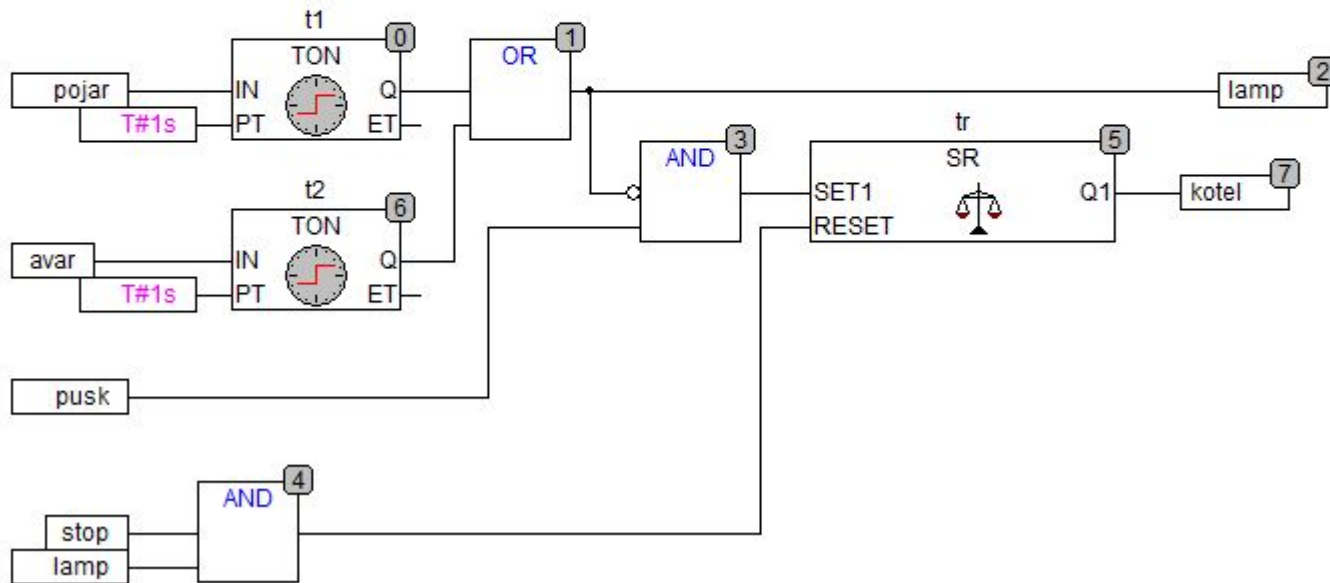
Необходимо реализовать:

- Включение сигнализации при возникновении любой из аварий
- Отключение котла при возникновении любой из аварий (реализовать проверку устойчивости срабатывания датчиков – отсутствия дребезга)
- Включение котла с кнопки, при условии отсутствия аварий.
- Отключение котла с кнопки.



# Реализация примера

```
0002 VAR
0003   tr: SR;
0004   t1s: TIME;
0005   t1: TON;
0006   t2: TON;
0007 END_VAR
0008
```

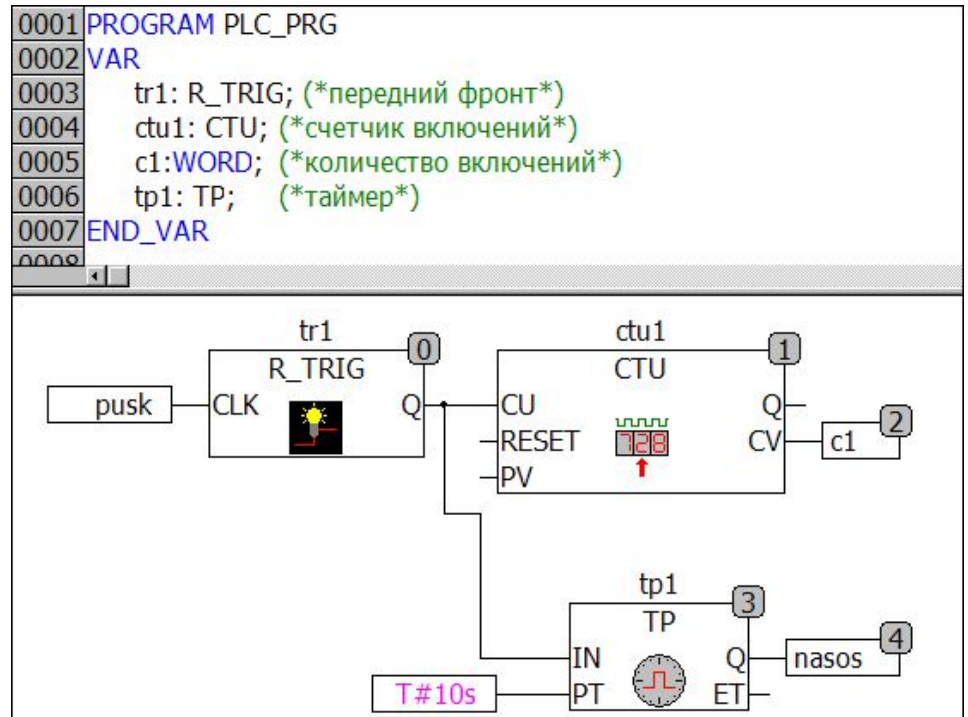
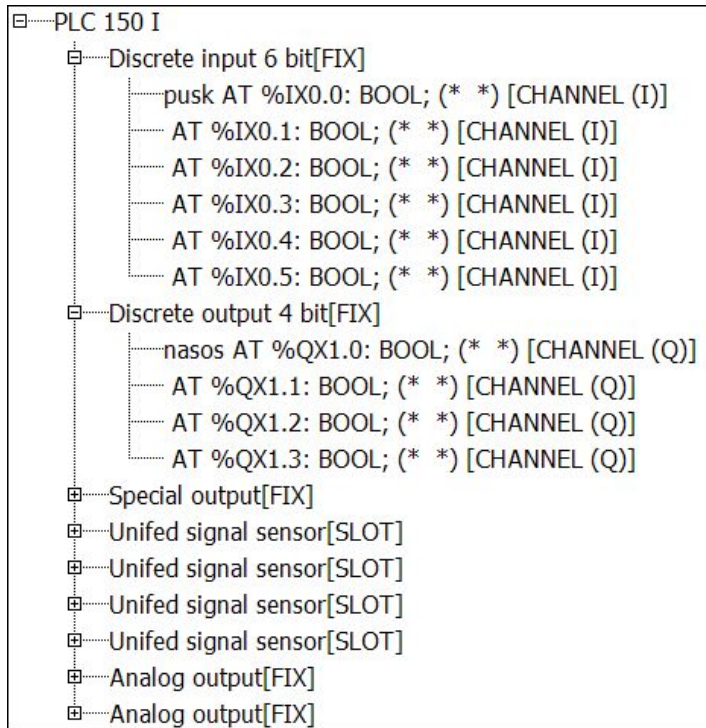


# Пример

- Включение насоса производится с кнопки.
- Включение насоса производится на 10 с, после чего он отключается. Отключение происходит даже при удержании кнопки или ее повторном нажатии в течении этих 10 с.
- Осуществляется подсчет числа включений.



# Решение примера



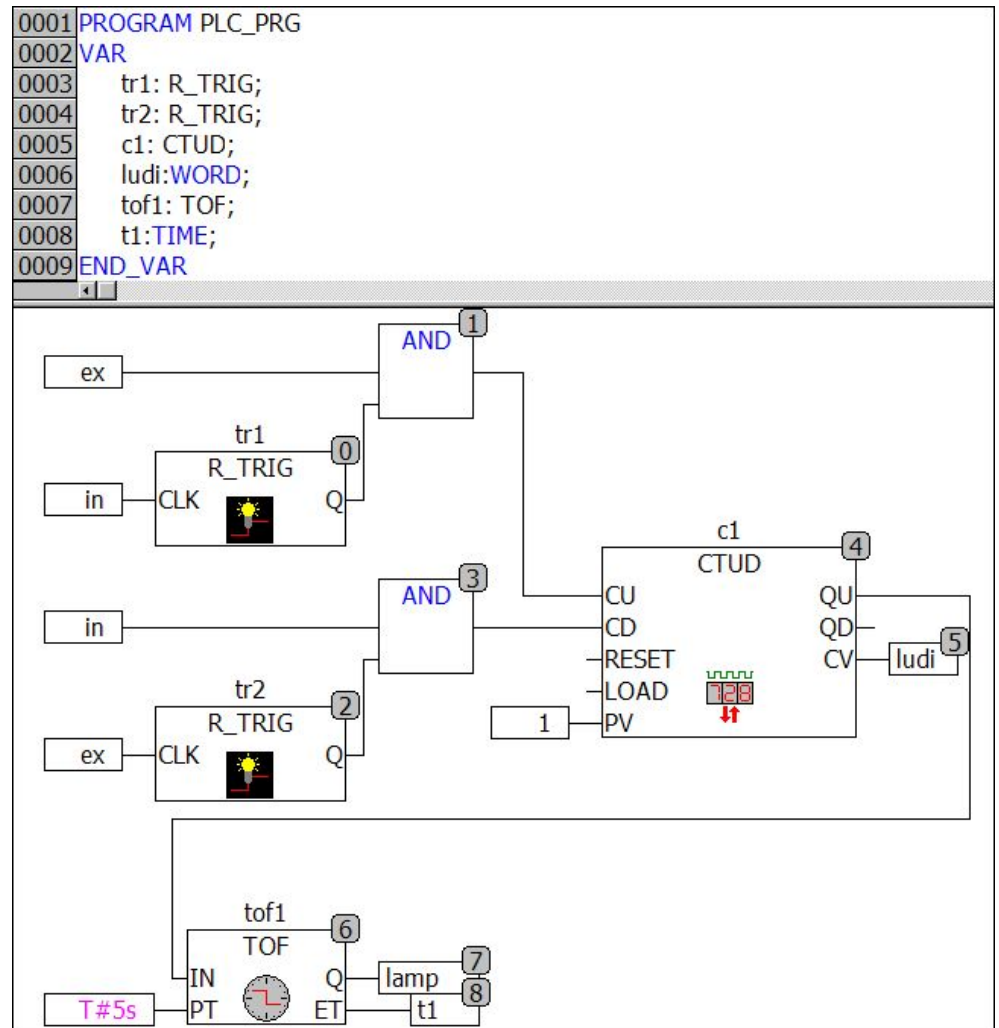
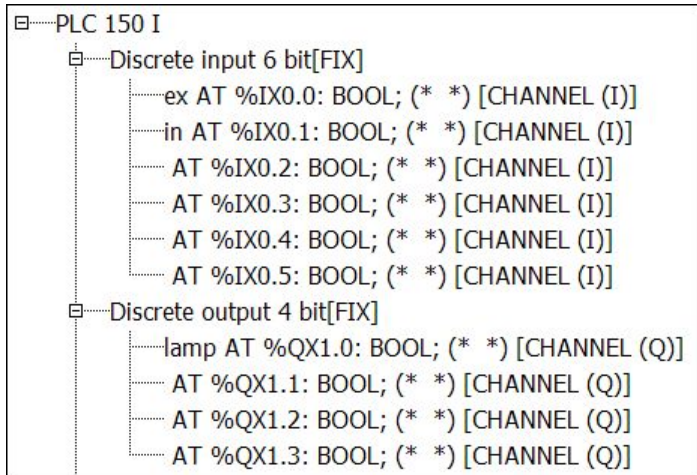
# Пример: управление светом в комнате

- На входе установлены два дискретных датчика: один снаружи комнаты, другой внутри. Человек входя или выходя, перекрывает собой по ширине оба датчика. Когда срабатывает сначала внешний датчик, затем внутренний, это означает, что человек зашел в комнату. Когда срабатывает сначала внутренний датчик, затем внешний, это означает, что человек вышел из комнаты (**используйте детекторы фронта**).
- Необходимо определять количество людей, находящихся в комнате (**подсчет произвести с помощью реверсивного счетчика**)
- Пока в комнате есть хотя бы один человек, свет должен быть включен. Если из комнаты вышел последний человек свет должен быть выключен **через 5 с** (**используйте таймер**).



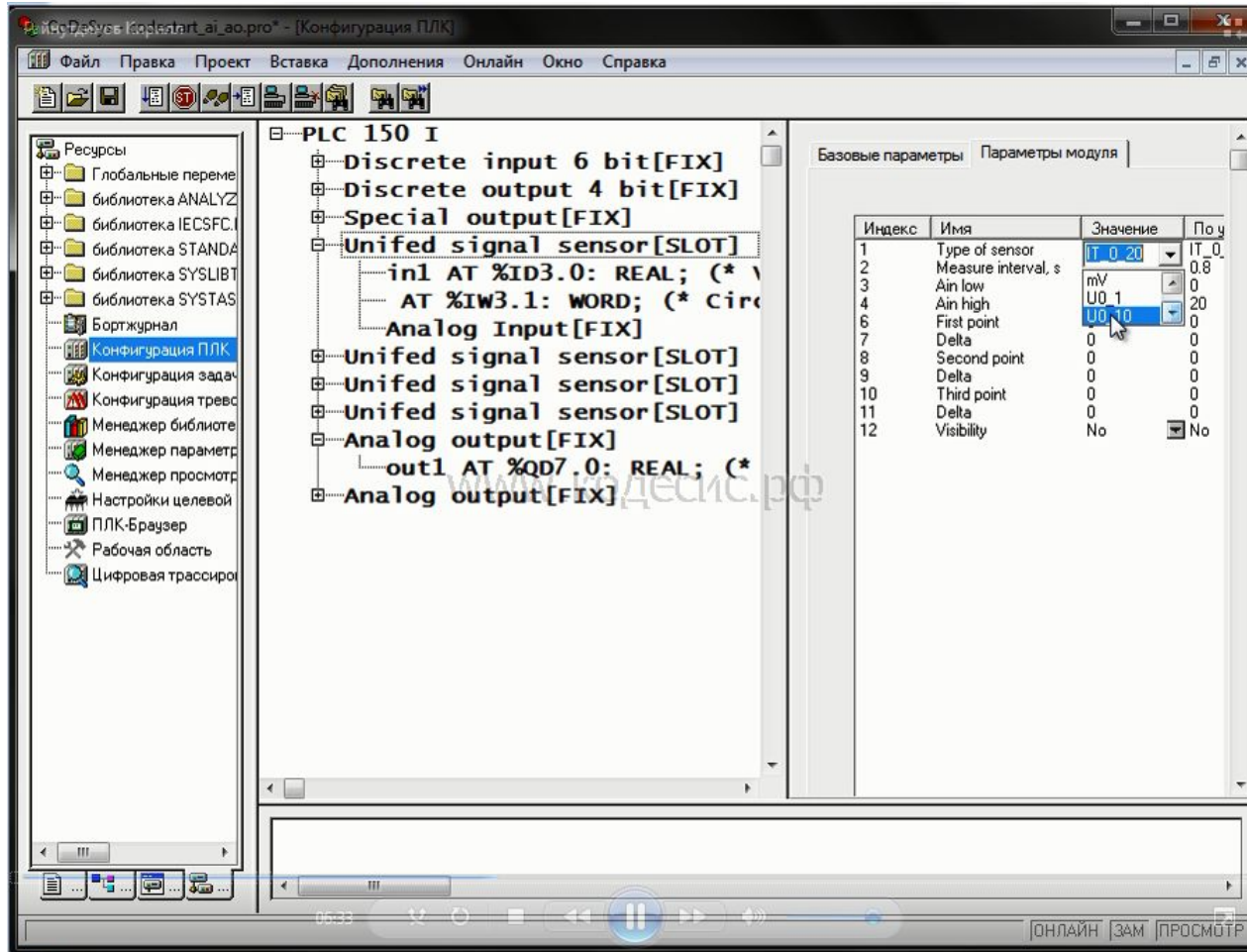


# Решение примера



# Работа с аналоговыми входами и выходами

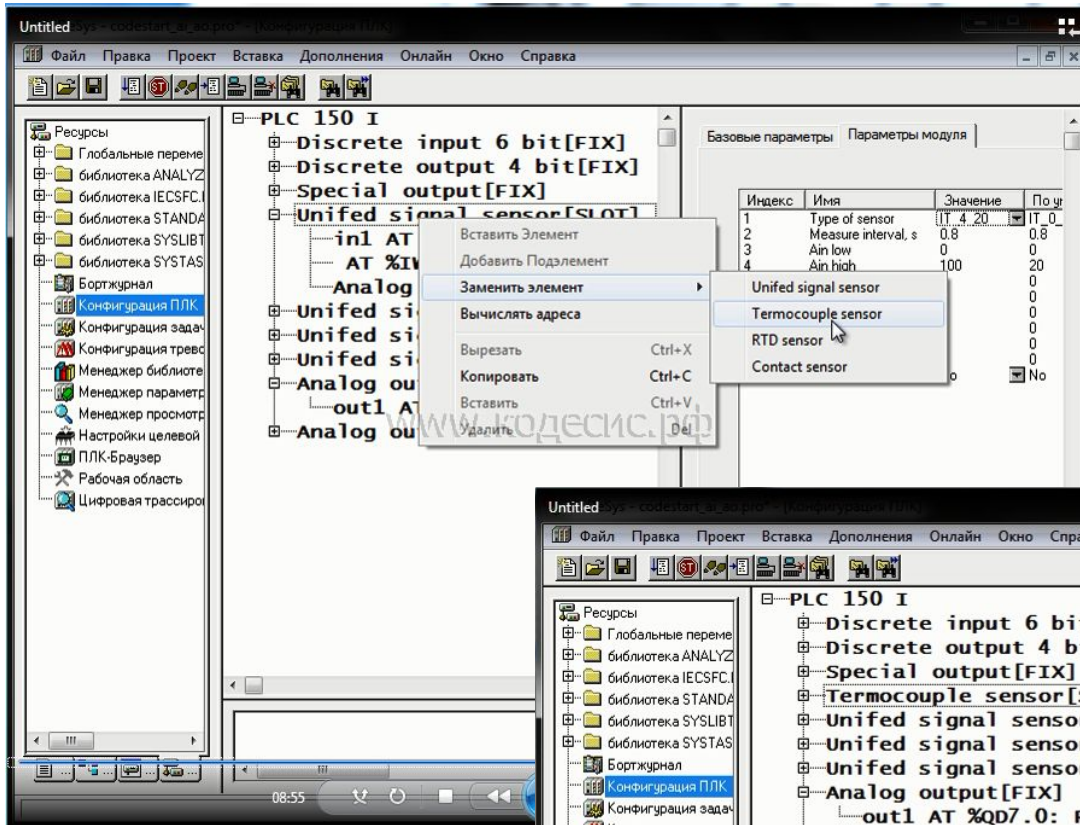
## Работа с унифицированным входным сигналом



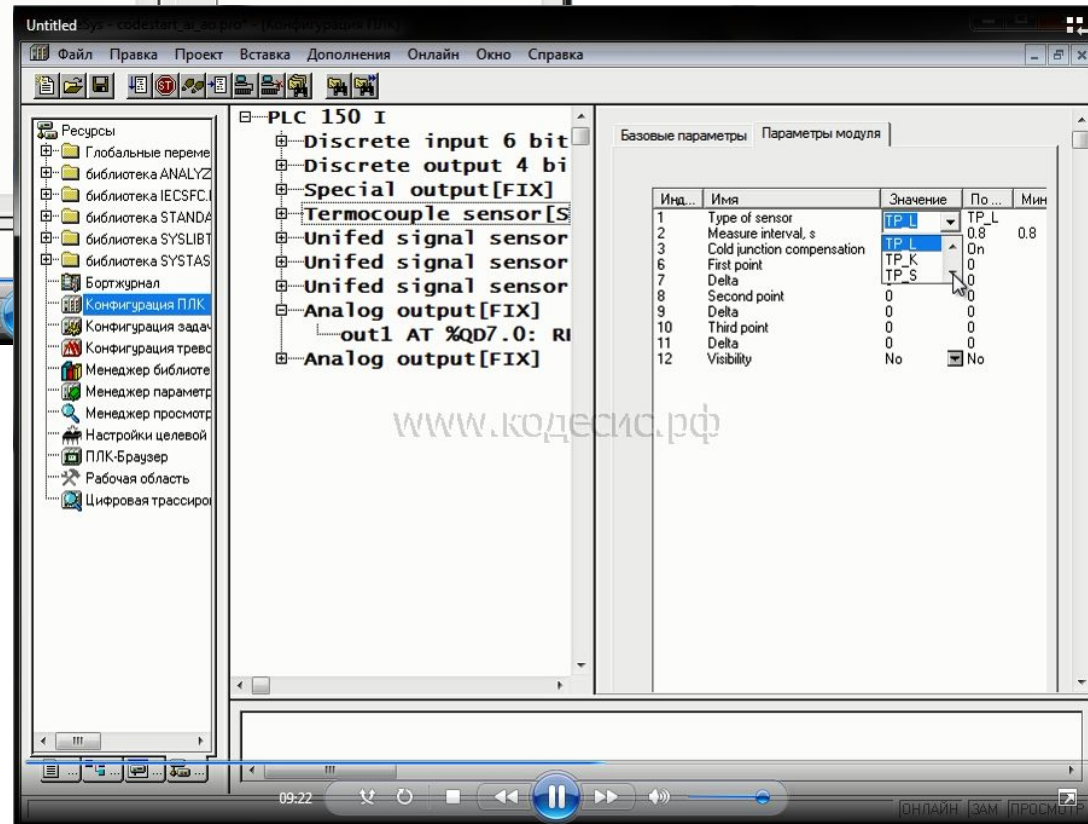
Выбор типа входного сигнала и его масштабирование

Выбор выходного сигнала при заказе

# Установка в качестве входного датчика термопары



Материал сплавов	Обозначение (тип)	
	Русское	МЭК
сильх-силини овые	ТСС	I
железо-константан овые	ТЖК	J
хромель-константан овые	ТХКн	E
хромель-алюмелевые	ТХА	K
хромель-копелевые	ТХК	L
медь-копелевые	ТМК	M
никросил-никсиль овые	ТНН	N
платинородий-платинородиевые	ТПР	B
медь-константан овые	ТМКн	T
вольфрамрениевые	ТВР	A



# Установка в качестве входного датчика терморезистора

The screenshot shows the SIMATIC Manager interface for configuring a PLC 150 I. The left pane displays a project tree with 'Конфигурация ПЛК' selected. The main workspace shows the PLC configuration tree with 'RTD sensor [SLOT]' highlighted. The right pane shows the 'Базовые параметры' (Basic parameters) tab for the selected sensor, displaying a table of parameters.

Индекс	Имя	Значение	По умолчанию
1	Type of sensor	r385_50	r385_50
2	Measure interval, s	0.8	0.8
6	First point	r385_100	0
7	Delta	r385_500	0
8	Second point	r385_1000	0
9	Delta	0	0
10	Third point	0	0
11	Delta	0	0
12	Visibility	No	No

www.кодесис.рф

Медный 100M  
0 оС – 100 Ом  
100 оС – 142,8 Ом  
W = 1,428

Pt1000  
0 оС – 1000 Ом  
1000 оС – 1385 Ом  
W = 1,385

# Пример по масштабированию входного и выходного аналогового сигнала и реализации простейшего релейного регулятора

- Измерение температуры осуществляется с помощью датчика напряжения на аналоговом входе. Напряжению 0 В соответствует температура 100 оС, напряжению 10 В соответствует температура 1000 оС.
- Сформировать переменную TEMPER, отображающую температуру
- Сформировать на аналоговом выходе напряжение, соответствующее температуре. Значению температуры 100 оС соответствует напряжение в 1 В, значению 1000 оС – 2 В;
- Включать исполнительный механизм после уменьшения температуры ниже уровня первой уставки UST1 (UST1 по умолчанию равна 100 градусов).  
Выключать исполнительный механизм при выходе температуры выше уровня второй уставки UST2 (UST2 по умолчанию равна 150 градусов).

Все масштабные преобразования выполнить программно.



PLC 150 U

- Discrete input 6 bit[FIX]
  - q1 AT %QX1.0: BOOL;
  - AT %QX1.1: BOOL; (\*)
  - AT %QX1.2: BOOL; (\*)
  - AT %QX1.3: BOOL; (\*)
- Special output[FIX]
- Unifed signal sensor[SLOT]
  - Uin AT %ID3.0: REAL;
  - AT %IW3.1: WORD; (\*)
  - Analog Input[FIX]
- Unifed signal sensor[SLOT]
- Unifed signal sensor[SLOT]
- Unifed signal sensor[SLOT]
- Analog output[FIX]
  - Uout AT %QD7.0: REA
- Analog output[FIX]

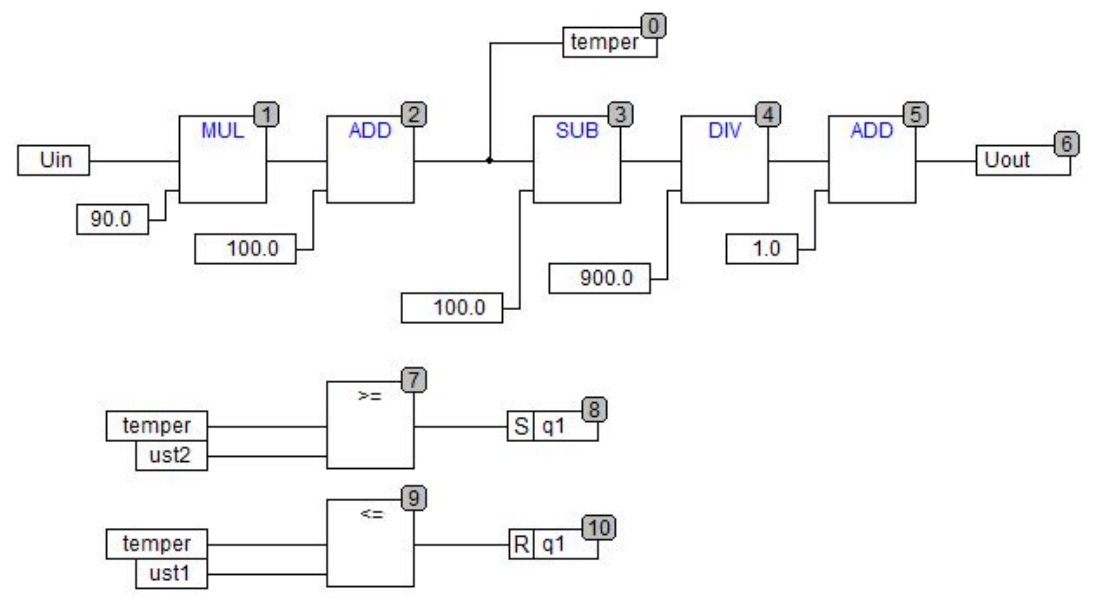
Базовые параметры | Параметры модуля

Индекс	Имя	Знач...	По ум...	Мин.
1	Type of sensor	U0_10	IT_0_20	
2	Measure interval, s	0.8	0.8	0.8
3	Ain low	0	0	
4	Ain high	10	20	
6	First point	0	0	
7	Delta	0	0	
8	Second point	0	0	
9	Delta	0	0	
10	Third point	0	0	
11	Delta	0	0	
12	Visibility	No	No	

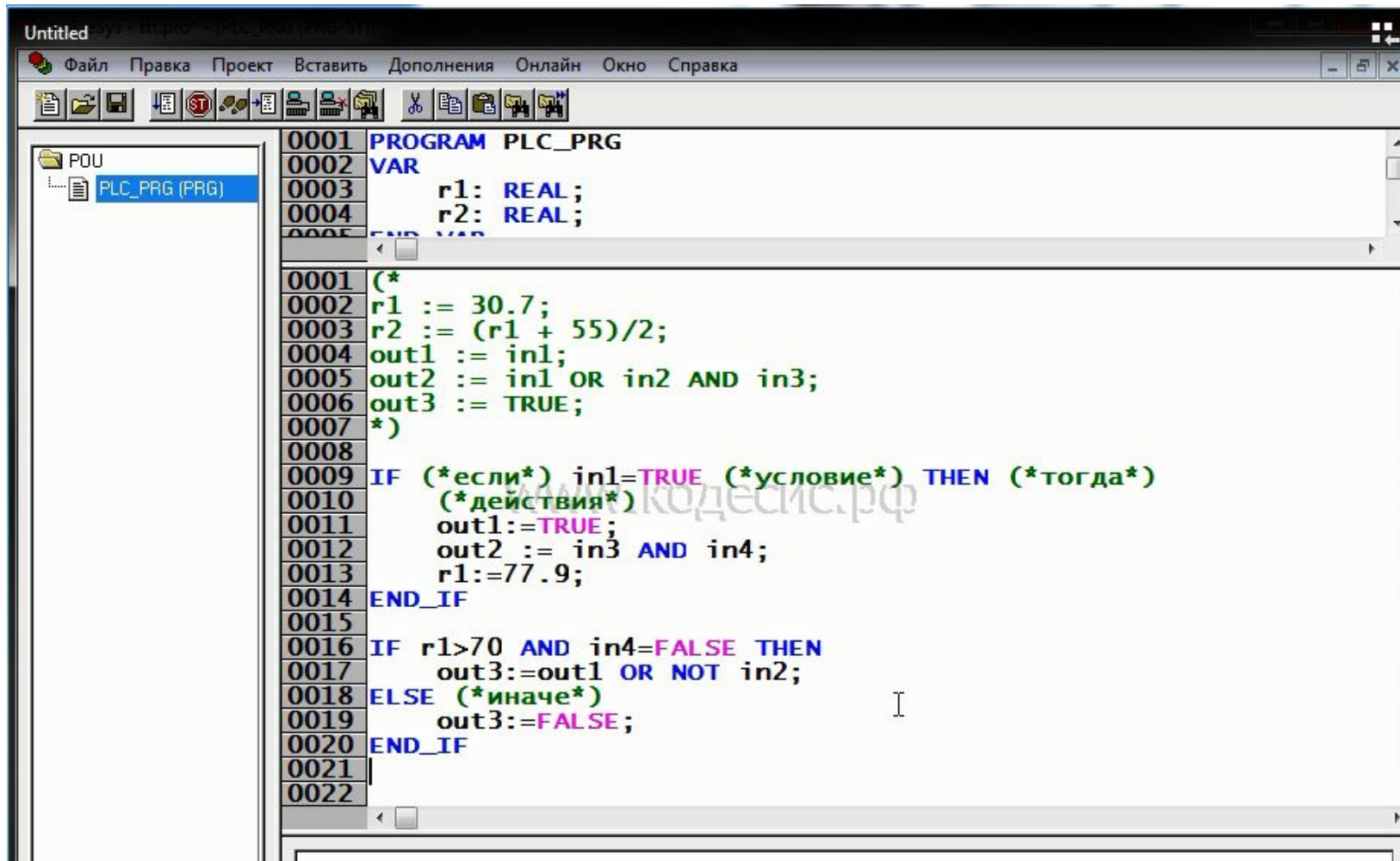
```

0002 VAR
0003   temper: REAL;
0004   ust2: REAL := 800;
0005   ust1: REAL := 100;
0006 END_VAR

```



# Язык ST



The image shows a screenshot of a software editor window titled "Untitled". The window has a menu bar with options: "Файл", "Правка", "Проект", "Вставить", "Дополнения", "Онлайн", "Окно", "Справка". Below the menu bar is a toolbar with various icons for file operations and editing. On the left side, there is a project tree showing a folder "POU" containing a file "PLC\_PRG (PRG)". The main editor area displays the following ST code:

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     r1: REAL;
0004     r2: REAL;
0005 END_VAR

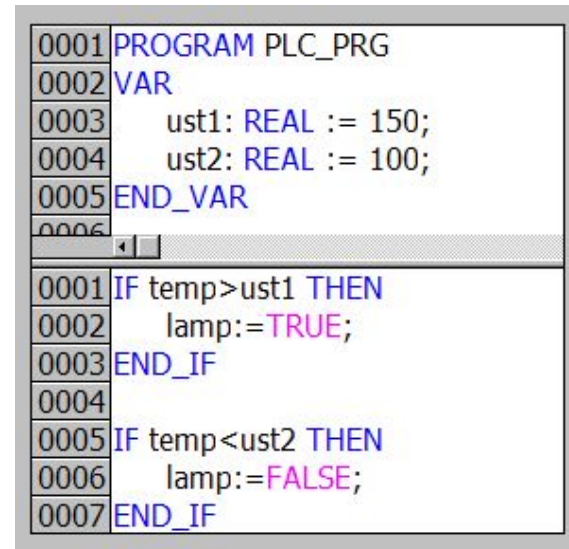
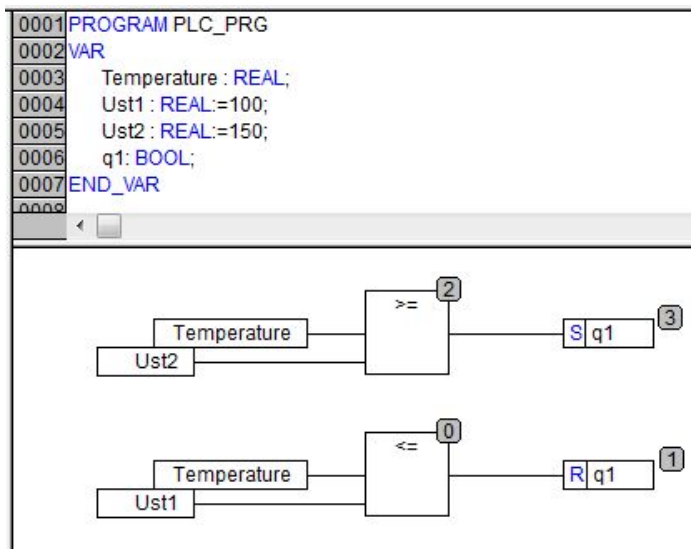
0001 (*
0002 r1 := 30.7;
0003 r2 := (r1 + 55)/2;
0004 out1 := in1;
0005 out2 := in1 OR in2 AND in3;
0006 out3 := TRUE;
0007 *)
0008
0009 IF (*если*) in1=TRUE (*условие*) THEN (*тогда*)
0010     (*действия*)
0011     out1:=TRUE;
0012     out2 := in3 AND in4;
0013     r1:=77.9;
0014 END_IF
0015
0016 IF r1>70 AND in4=FALSE THEN
0017     out3:=out1 OR NOT in2;
0018 ELSE (*иначе*)
0019     out3:=FALSE;
0020 END_IF
0021
0022
```

# Условный оператор If в языке ST

**IF** «условие» **THEN**      Логическая переменная или выражение  
«действие1»;      Операции, которые необходимо      «действие2»;  
производить при выполнении условия  
«действие3»;

...  
**ELSE**      **Иначе**  
«действие4»;      Операции, которые необходимо  
«действие5»;      производить при НЕвыполнении условия

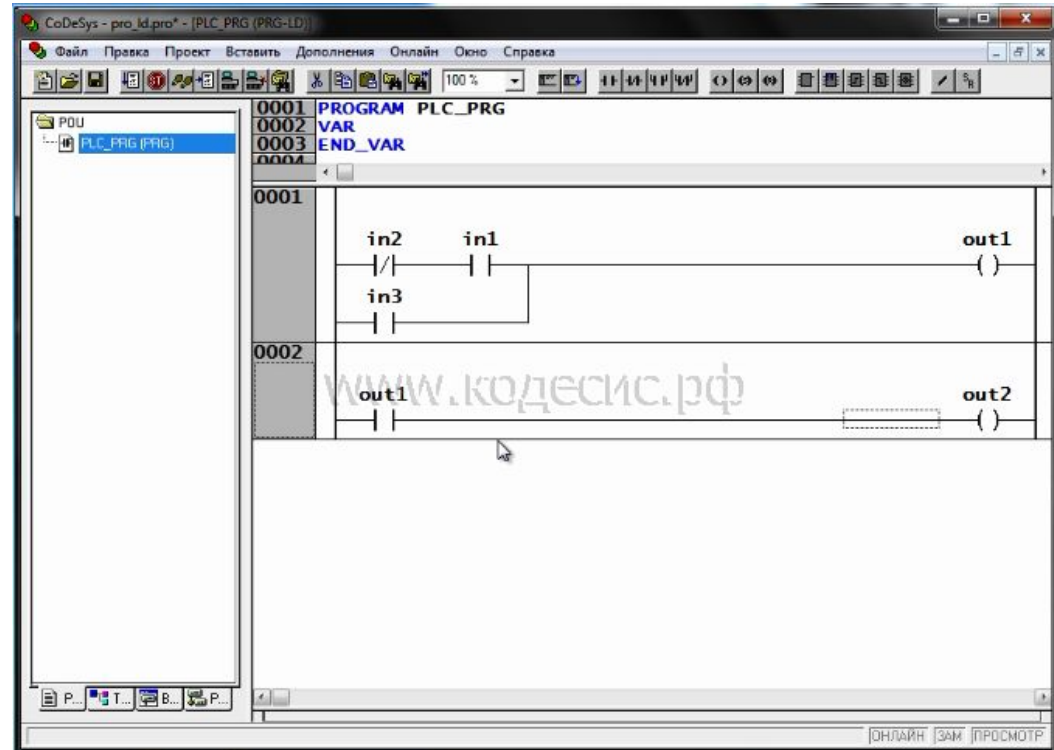
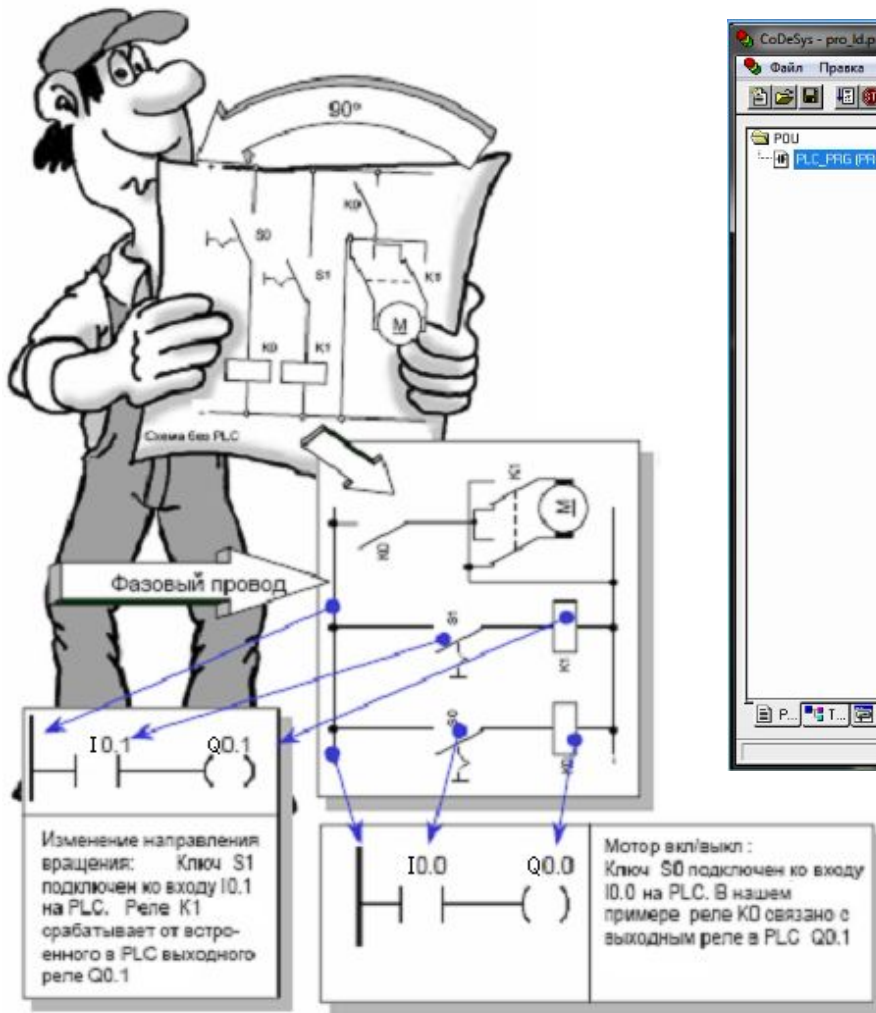
...  
**END\_IF**      Окончание описания условия



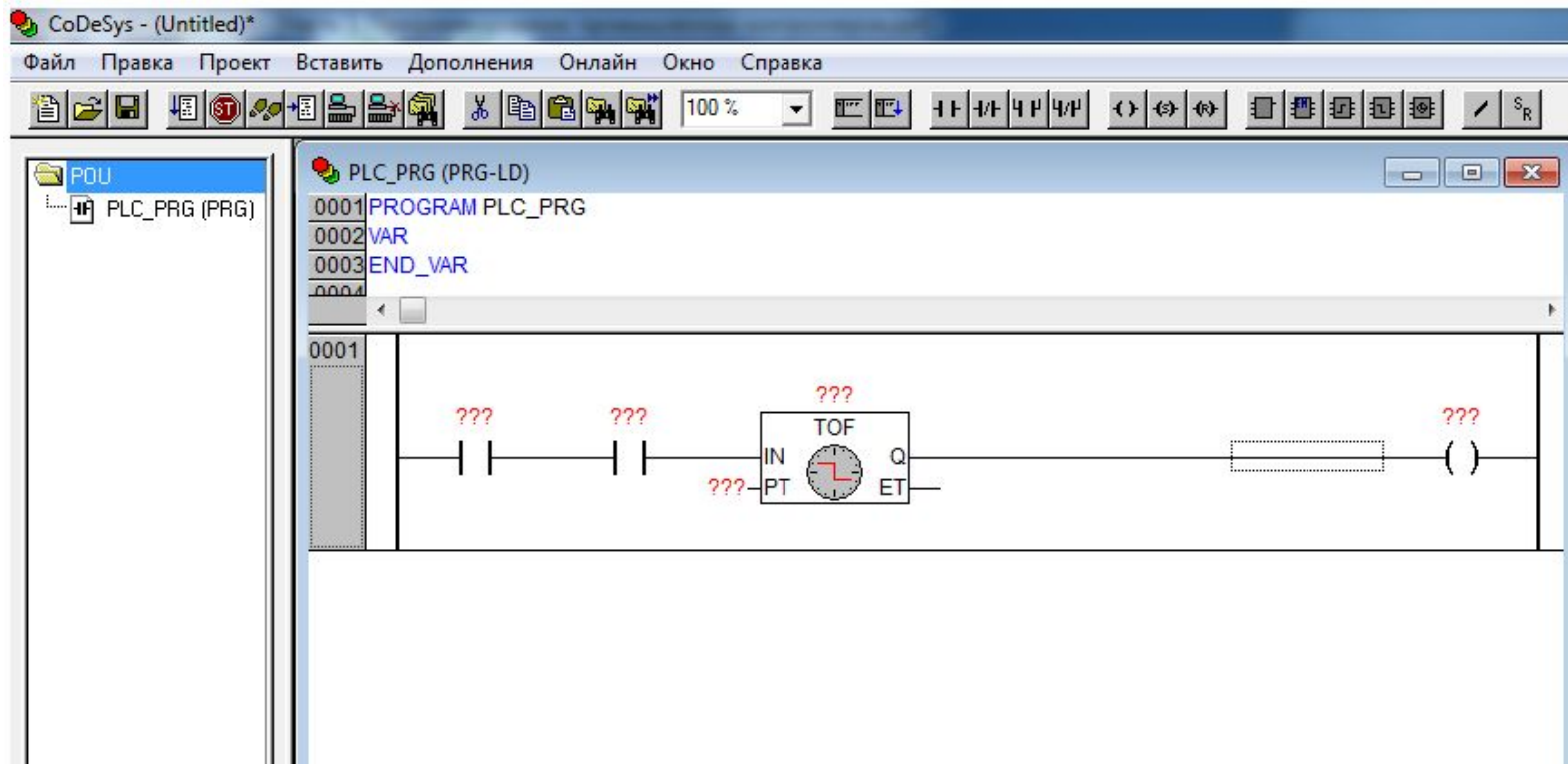


# Язык LD

Берет свое «начало» от релейных схем



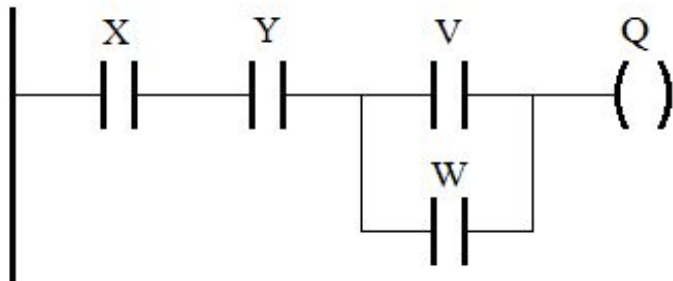
# Функциональные блоки в LD



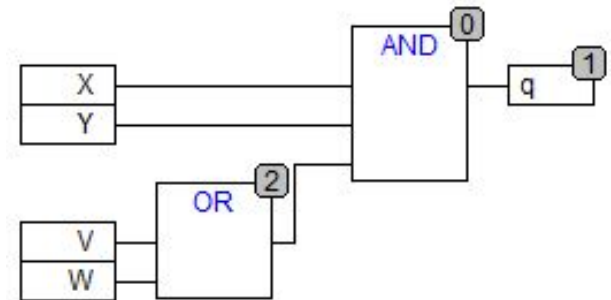
Работа с функциональными блоками в LD не отличается от работы с ними в других языках

# Язык IL

Привычен для специалистов, знакомых с программированием на ассемблере. Но в отличие от ассемблера, как и другие языки программирования контроллеров, является языком высокого уровня.



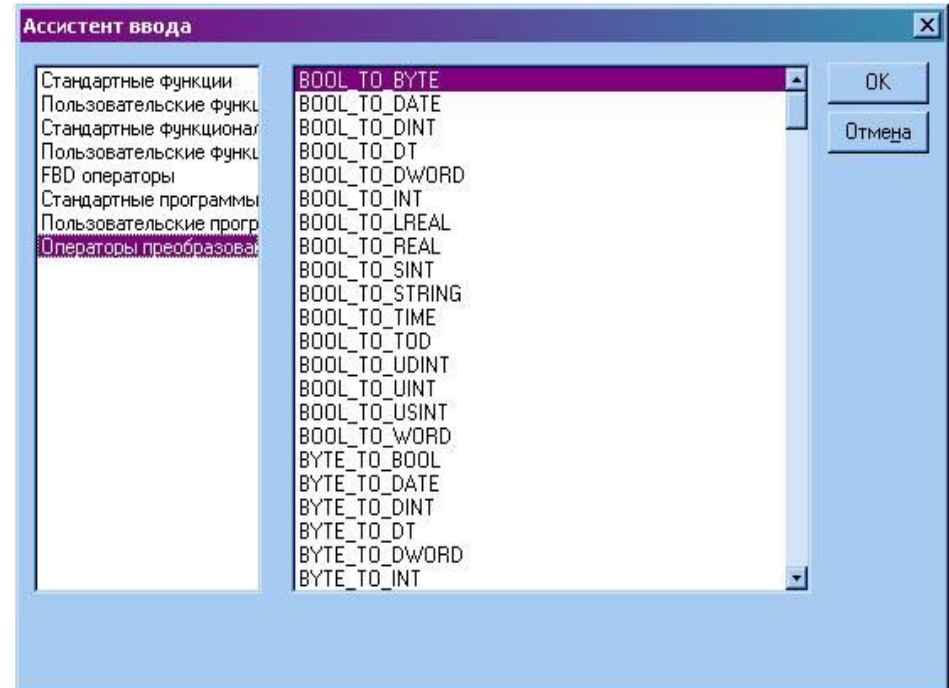
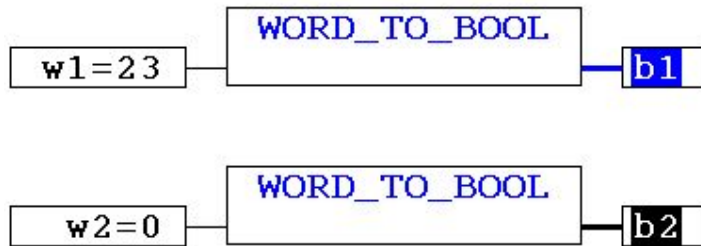
```
LD X
AND Y
AND (V
OR W
)
ST Q
```



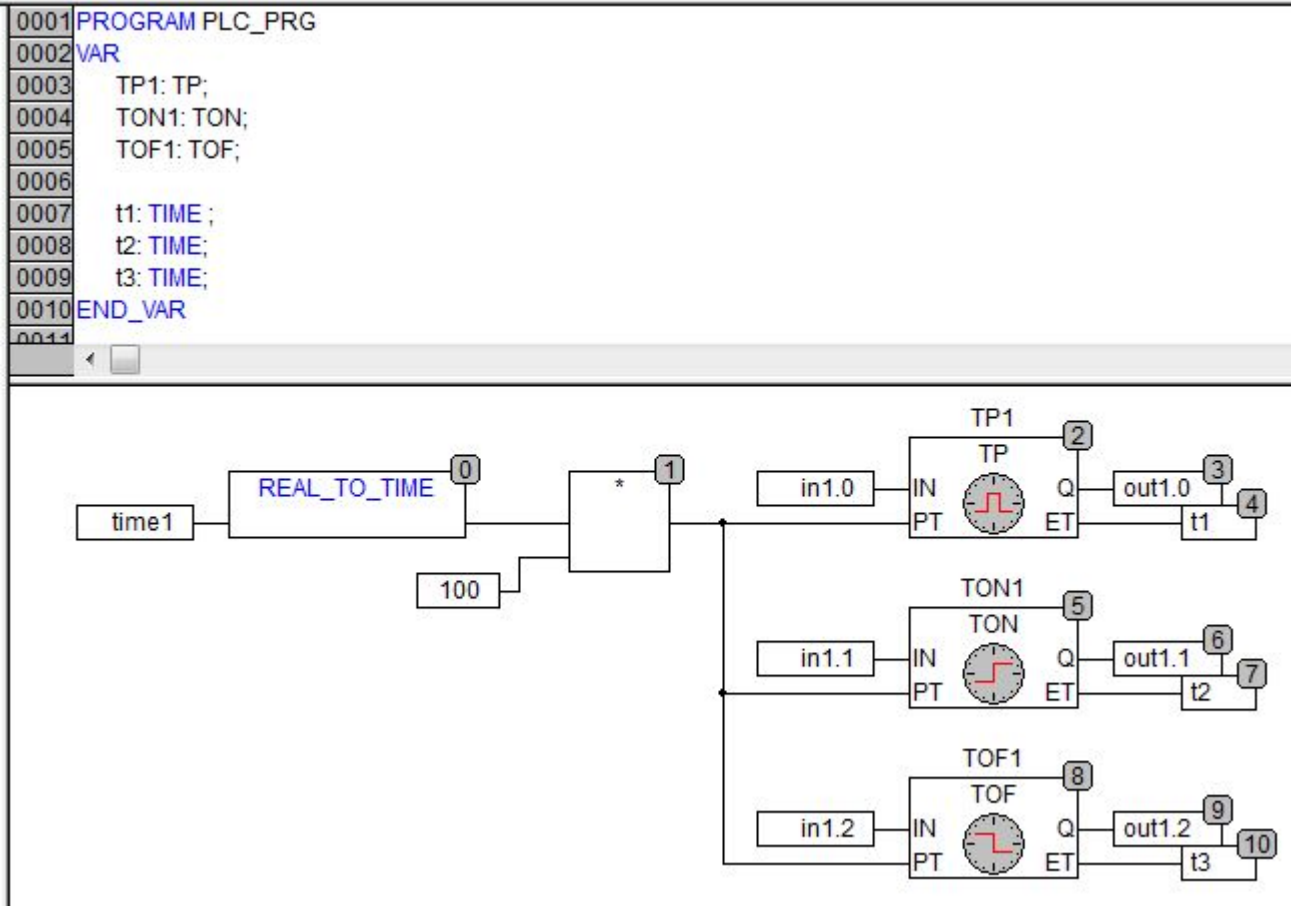
Сопоставление одних и тех же действий в программах на языках LD, IL и CFC

# Операторы преобразования типов данных

Для каждой пары типов данных используется отдельный оператор. В названии оператора сначала указывается исходный тип данных, а затем тип результата.

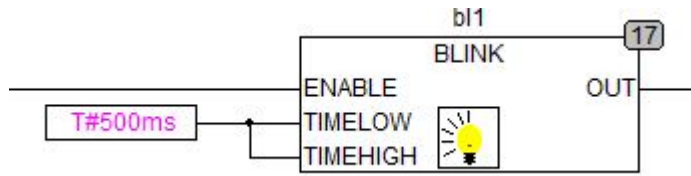


# Регулировка интервалов с помощью потенциометра



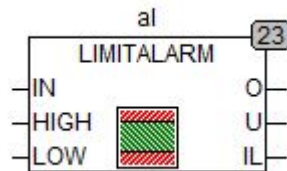
# Стандартная библиотека Util.lib

Примеры функциональных блоков библиотеки



Генератор прямоугольных импульсов запускается по входу `ENABLE = TRUE`.  
Длительность импульса задается `TIMEHIGH`,  
длительность паузы `TIMELOW`

Входы: `ENABLE` типа `BOOL`, `TIMELOW` и `TIMEHIGH` типа `TIME`. Выход `OUT` типа `BOOL`



Контролирует принадлежность значения входа `IN` заданному диапазону.

Если значение на входе `IN`:

превышает предел `HIGH`,

то выход `O = TRUE`;

меньше предела `LOW`,

то выход `U = TRUE`;

лежит в пределах между `LOW` и `HIGH`

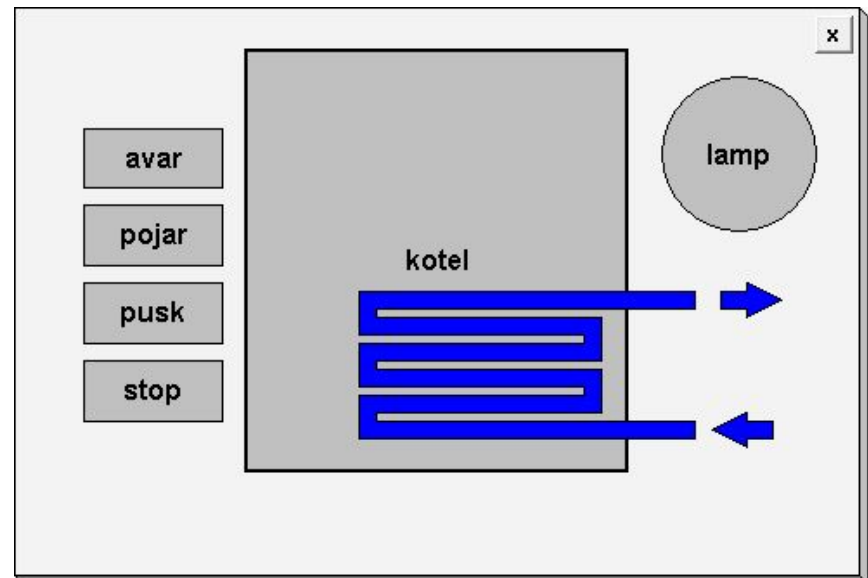
(включительно), то выход `IL = TRUE`

Входы `IN`, `HIGH` и `LOW` типа `INT`,  
выходы `O`, `U` и `IL` типа `BOOL`

# Пример: управление котлом

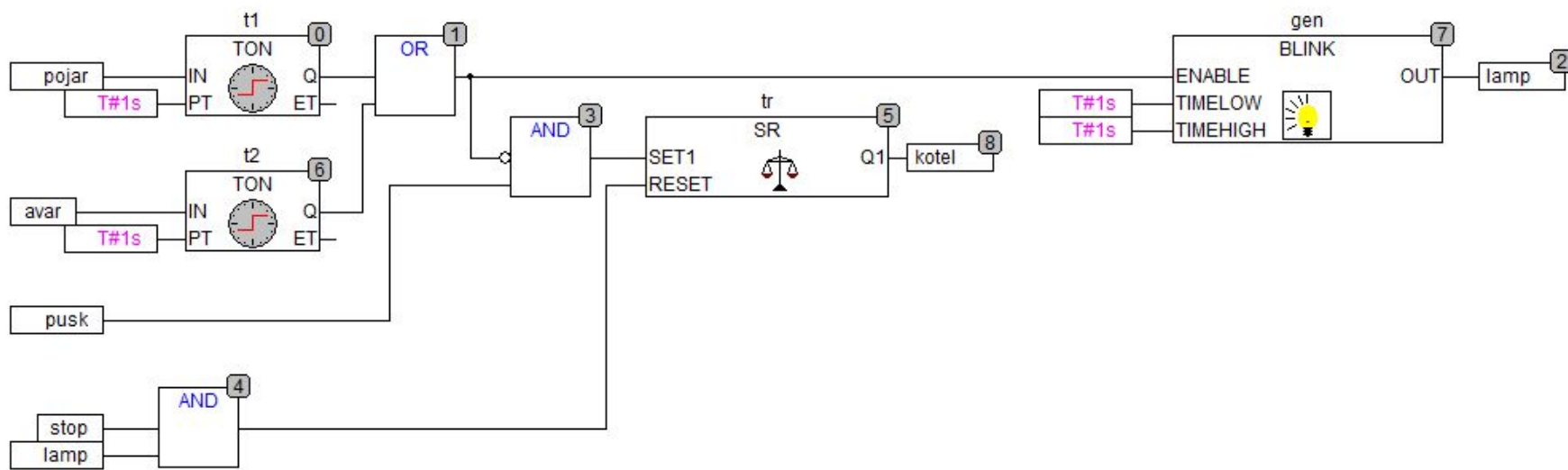
Необходимо реализовать:

- Включение сигнализации (должна мигать лампа) при возникновении любой из аварий
- Отключение котла при возникновении любой из аварий (реализовать проверку устойчивости срабатывания датчиков – отсутствия дребезга)
- Включение котла с кнопки, при условии отсутствия аварий.
- Отключение котла с кнопки.



# Реализация примера

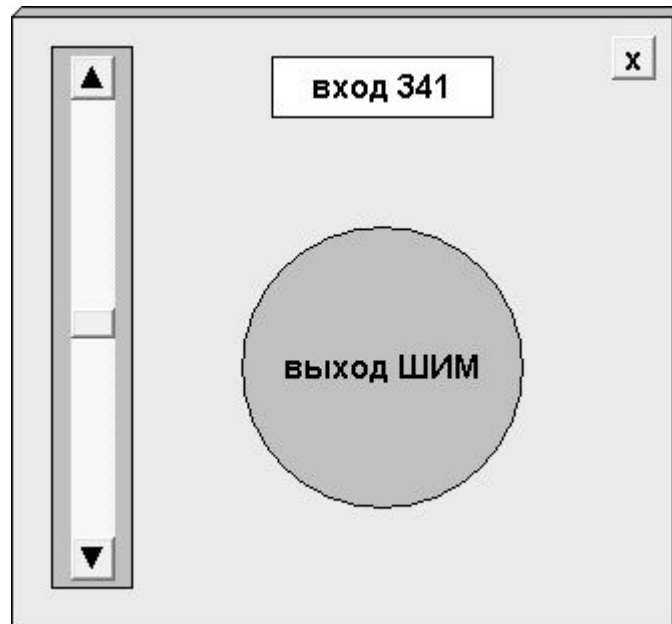
```
0002 VAR
0003   tr: SR;
0004   GEN: BLINK;
0005   t1s: TIME;
0006   t1: TON;
0007   t2: TON;
0008 END_VAR
```





# Пример. Формирование импульсов

Сигнал на аналоговом входе меняется в пределах от 0 до 10 В. При изменении сигнала на аналоговом входе от 4 до 10 В необходимо изменять скважность выходных импульсов в диапазоне от 20 до 50 %. Период импульсов – 1 с.



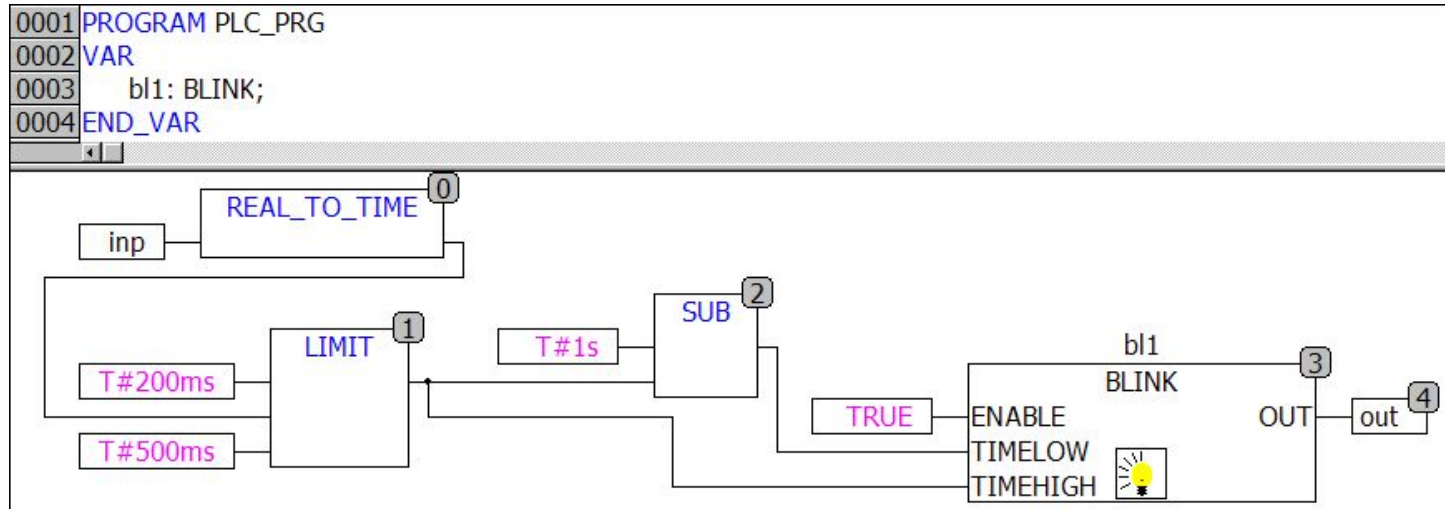
# Решение примера

PLC 150 U

- Discrete input 6 bit[FIX]
- Discrete output 4 bit[FIX]
  - out AT %QX1.0: BOOL; (\* \*) [CHANNEL (Q)]
  - AT %QX1.1: BOOL; (\* \*) [CHANNEL (Q)]
  - AT %QX1.2: BOOL; (\* \*) [CHANNEL (Q)]
  - AT %QX1.3: BOOL; (\* \*) [CHANNEL (Q)]
- Special output[FIX]
- Unified signal sensor[SLOT]
  - inp AT %ID3.0: REAL; (\* Value \*) [CHANNEL (I)]
  - AT %IW3.1: WORD; (\* Circular time \*) [CHANNEL (I)]
  - Analog Input[FIX]
- Unified signal sensor[SLOT]
- Unified signal sensor[SLOT]
- Unified signal sensor[SLOT]
- Analog output[FIX]
- Analog output[FIX]

базовые параметры | параметры модуля

Индекс	Имя	Значение
1	Type of sensor	RQ_5000
2	Measure interval, s	0.5
3	Ain low	0
4	Ain high	500
6	First point	0
7	Delta	0
8	Second point	0
9	Delta	0
10	Third point	0
11	Delta	0
12	Visibility	No



# Типы POU

- Функция: **< FUNCTION >**

Имеет один или более входов, один выход, рекурсии не допустимы

- Функциональный блок: **<FUNCTION\_BLOCK >**

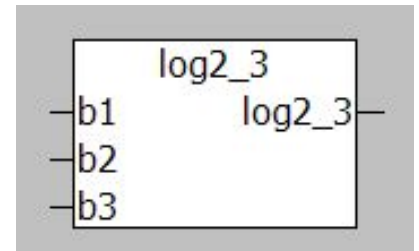
Имеет произвольное число входов и выходов. Имеет внутреннюю память. Для каждого функционального блока можно объявить несколько экземпляров

- Программа: **< PROGRAM >**

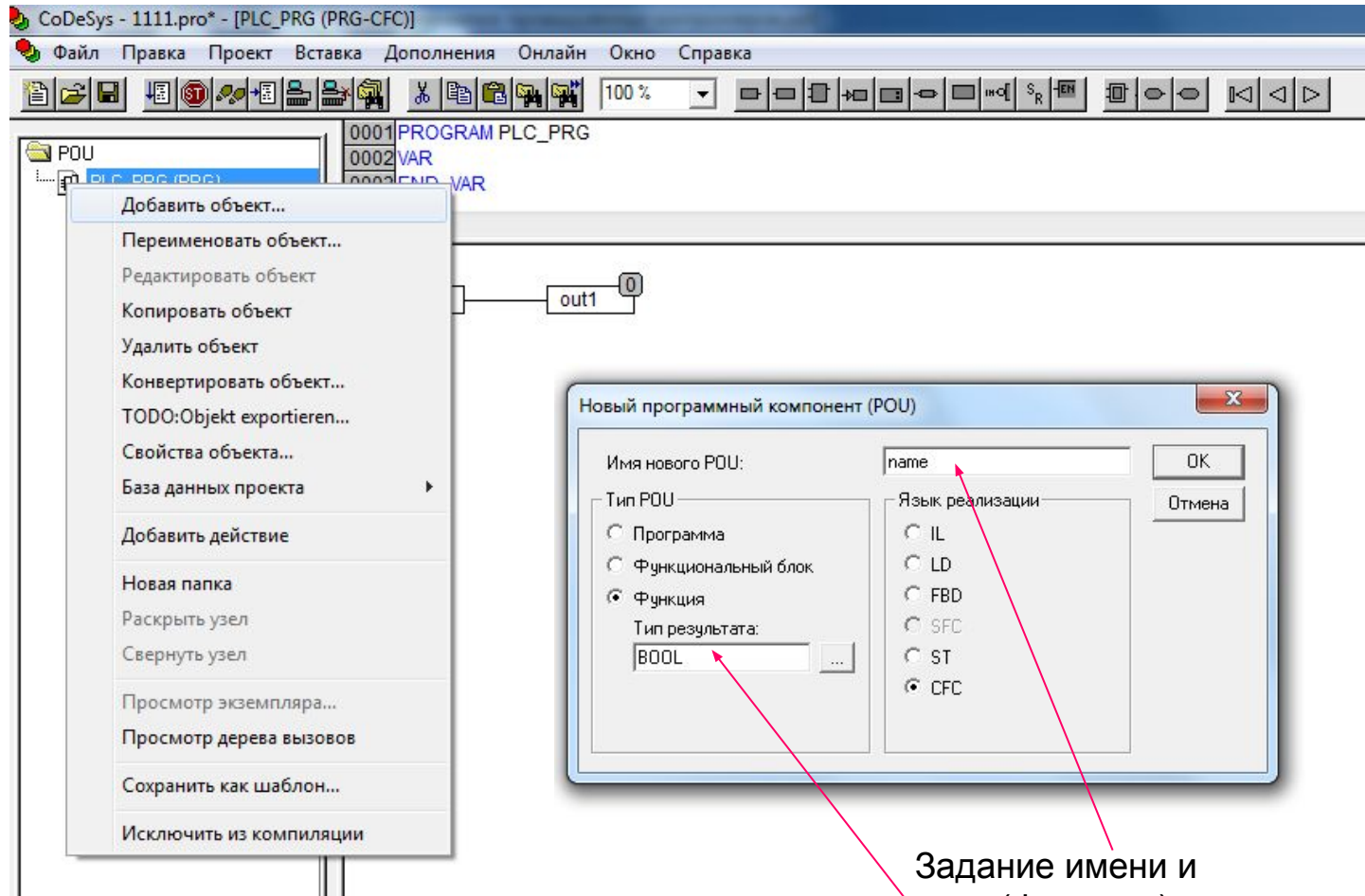
Подобна функциональному блоку, но имеет один глобальный экземпляр

# Функция

- Не имеет внутренней памяти.
- Локальные переменные инициализируются при каждом вызове.
- Функция возвращает значение, через свой идентификатор. Функция имеет тип!
- Удобна для реализации комплексных вычислений.
- Не рекомендуется использование глобальных переменных в функции.



# Создание функции



Задание имени и  
типа (формата) результата

Не забывайте указывать, какие переменные являются входными (VAR\_INPUT)  
Выходной переменной является переменная с именем и форматом функции

The screenshot shows the CoDeSys software interface. The main window displays a function declaration in ladder logic:

```
0001 FUNCTION name : BOOL
0002 VAR_INPUT
0003 END_VAR
0004 VAR
0005 END_VAR
0006
```

The left sidebar shows a project tree with the following structure:

- POU
  - name (FUN)
  - PLC\_PRG (PRG)

The main workspace shows a variable declaration dialog box titled "Объявление переменной" (Variable Declaration). The dialog box contains the following fields and options:

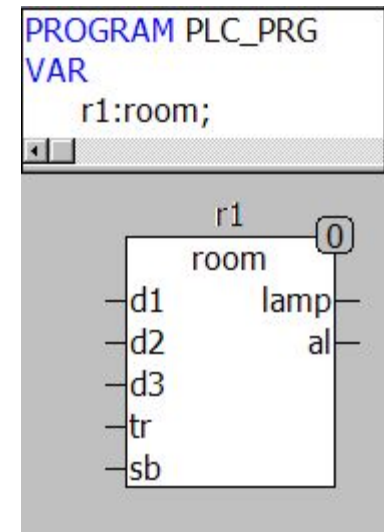
Класс	Имя	Тип
VAR	i1	BOOL
VAR_INPUT		
VAR_GLOBAL		

Additional fields and options in the dialog box:

- Нач. значение (Initial value):
- Адрес (Address):
- Комментарий (Comment):
- Buttons: OK, Отмена (Cancel)
- Checkboxes:  CONSTANT,  RETAIN,  PERSISTENT

# Функциональный блок

- Все переменные функционального блока сохраняют значения
- При создании экземпляра функционального блока создается новая копия переменных функционального блока. Копия кода функционального блока не создается.
- Рекомендуется для программирования повторно используемого кода, например, счетчиков, таймеров, триггеров и т.д.



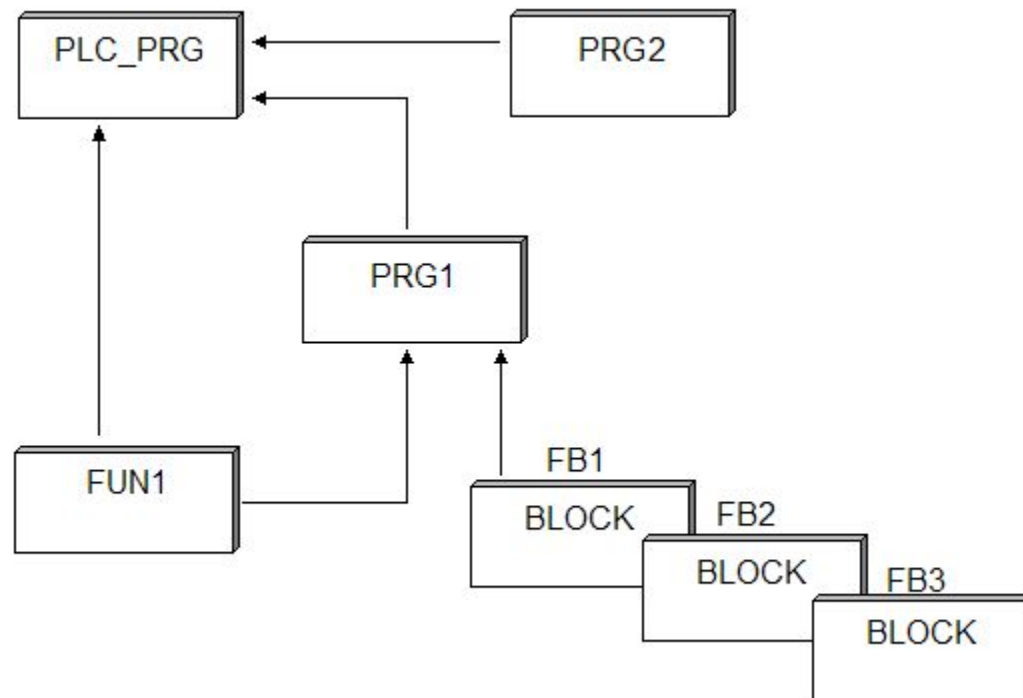
# Программа

- Все переменные сохраняют свои значения
- Используется для структурирования приложения



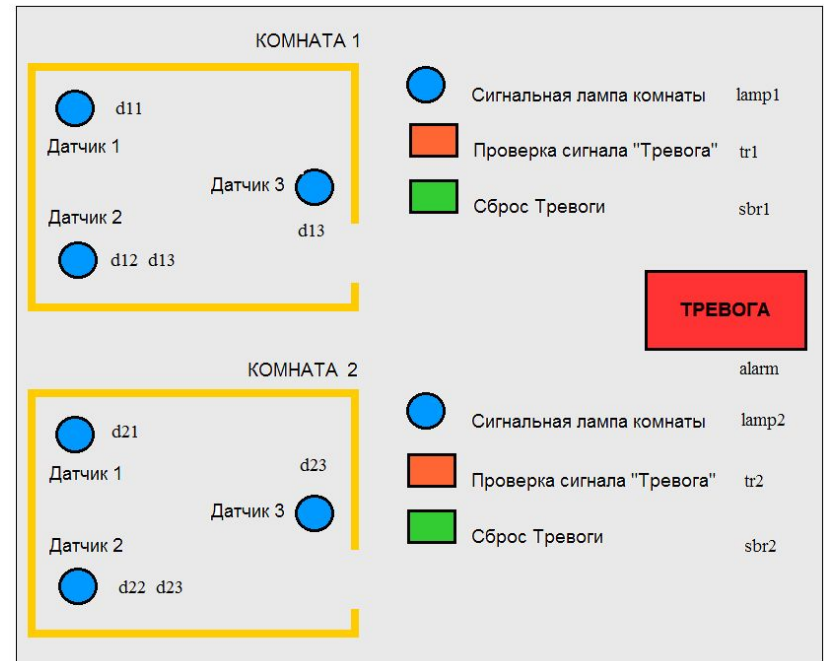
# Вложенность программных компонентов

Все программные компоненты должны вызываться прямо или косвенно **из главной программы PLC\_PRG.**

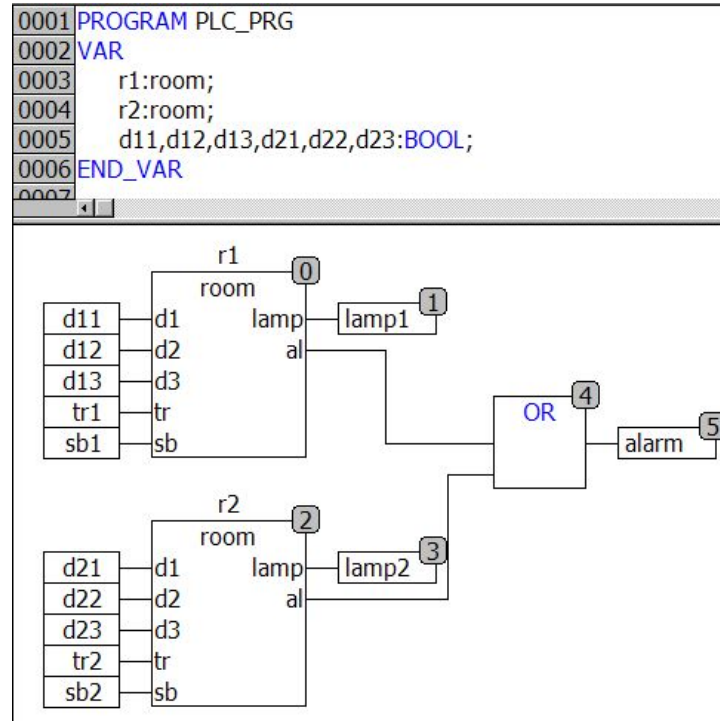
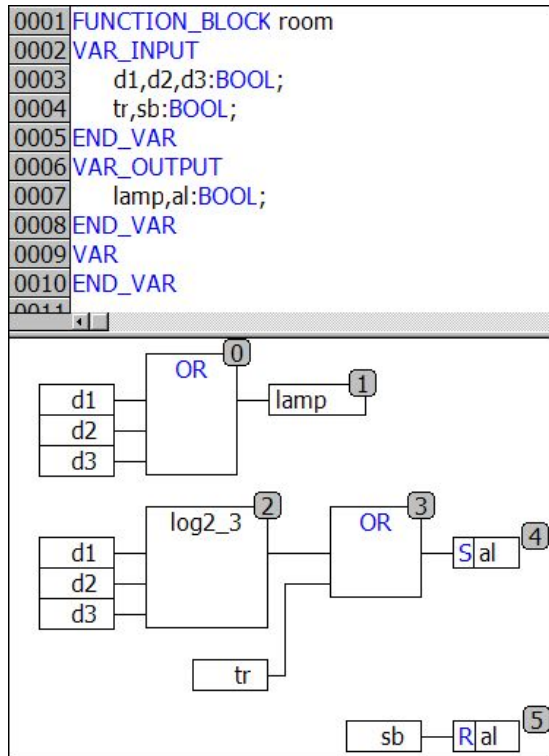
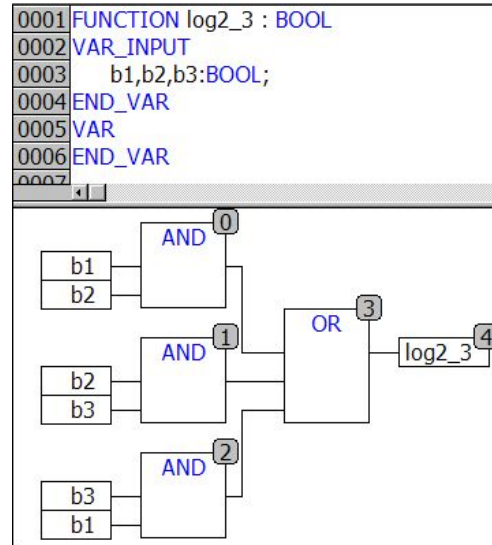
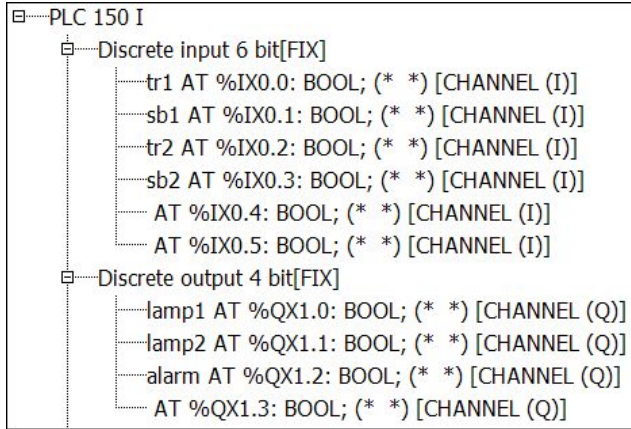


# Пример. Система пожарной сигнализации здания

- В здании две одинаковые комнаты.
- В каждой комнате установлено три пожарных датчика (**d11**, **d12**, **d13** и **d21**, **d22**, **d23**), кнопка ручного включения сигнализации (**tr1** и **tr2**) и кнопка ручного отключения сигнализации (**sb1** и **sb2**).
- Для каждой комнаты предусмотрена сигнальная лампа (**lamp1**, **lamp2**). Сигнализация пожара (**alarm**) является общей для обеих комнат.
- Если в комнате срабатывает хотя бы один из датчиков, то загорается сигнальная лампа для соответствующей комнаты. Лампа гаснет, если все датчики в комнате отключены.
- Если в комнате срабатывает любые два из трех датчиков, то включается пожарная сигнализация. Сигнализация работает до тех пор, пока ее не отключат соответствующей кнопкой.
- Сигнализация может быть включена кнопкой проверки вне зависимости от состояния датчиков.



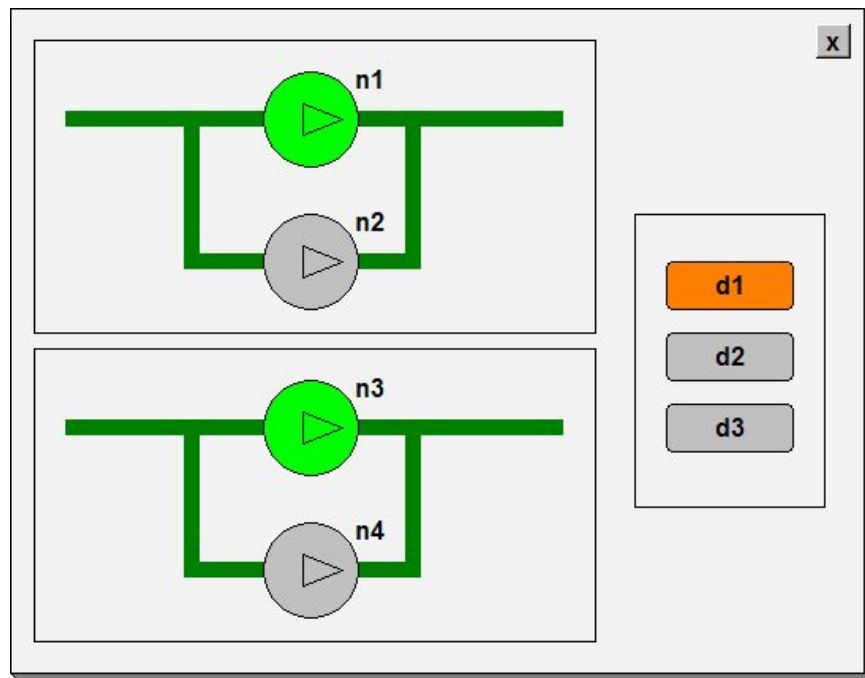
Анализ срабатывания двух датчиков из трех реализовать с помощью функции  
Обработку всех сигналов для каждой из комнат реализовать с помощью функционального блока



# Пример: автоматический ввод резерва

На объекте 2 группы насосов, по 2 насоса в каждой группе (n1, n2, n3 и n4).  
В каждой группе один насос рабочий, второй в резерве.  
Если срабатывают любые 2 из 3-х технологических датчиков (d1, d2, d3),  
то необходимо переключить насосы в каждой группе  
с работающего на резервный

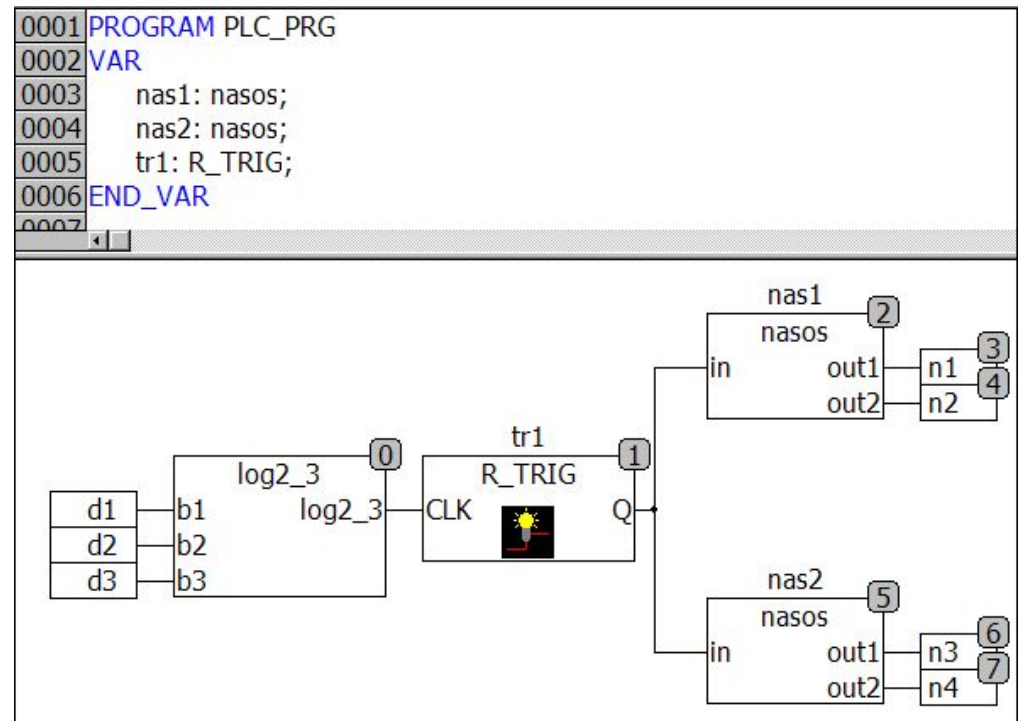
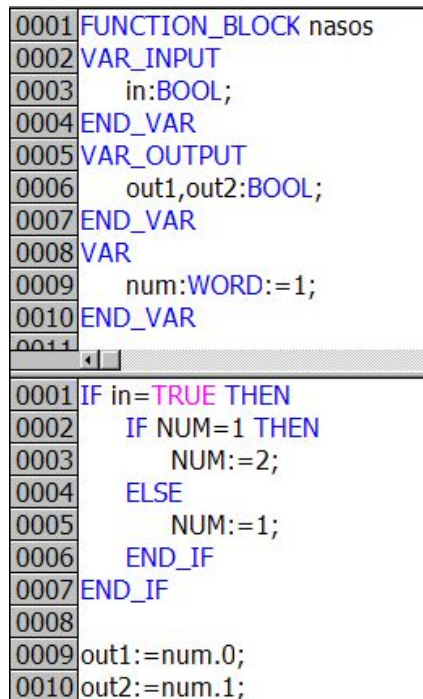
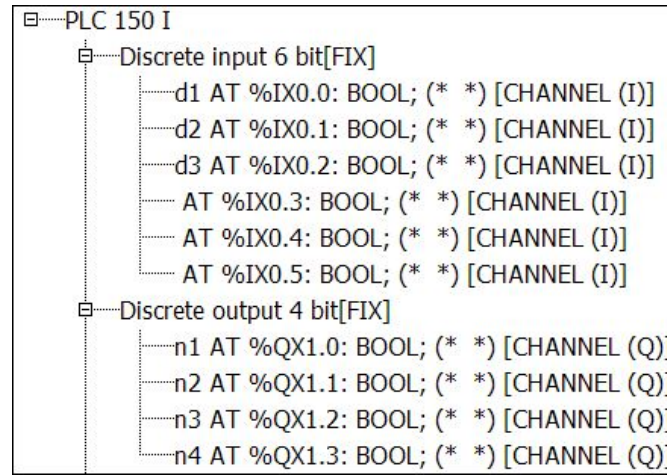
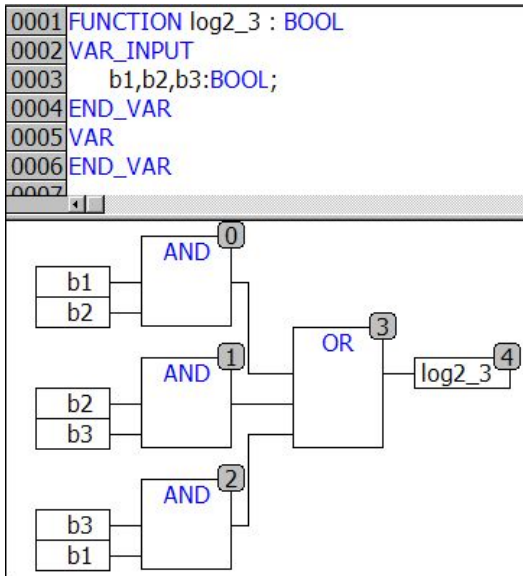
Используйте функциональные блоки и функции



Проверка срабатывания датчиков пусть выполняется с помощью функции LOG2\_3

Принятие решений в одинаковых группах двигателей пусть осуществляется с помощью функционального блока Nasos

# Решение примера





# Работа с реальным временем

Командная строка  
браузера

Время  
и дата  
для контроллера  
задаются в  
ПЛК Браузере

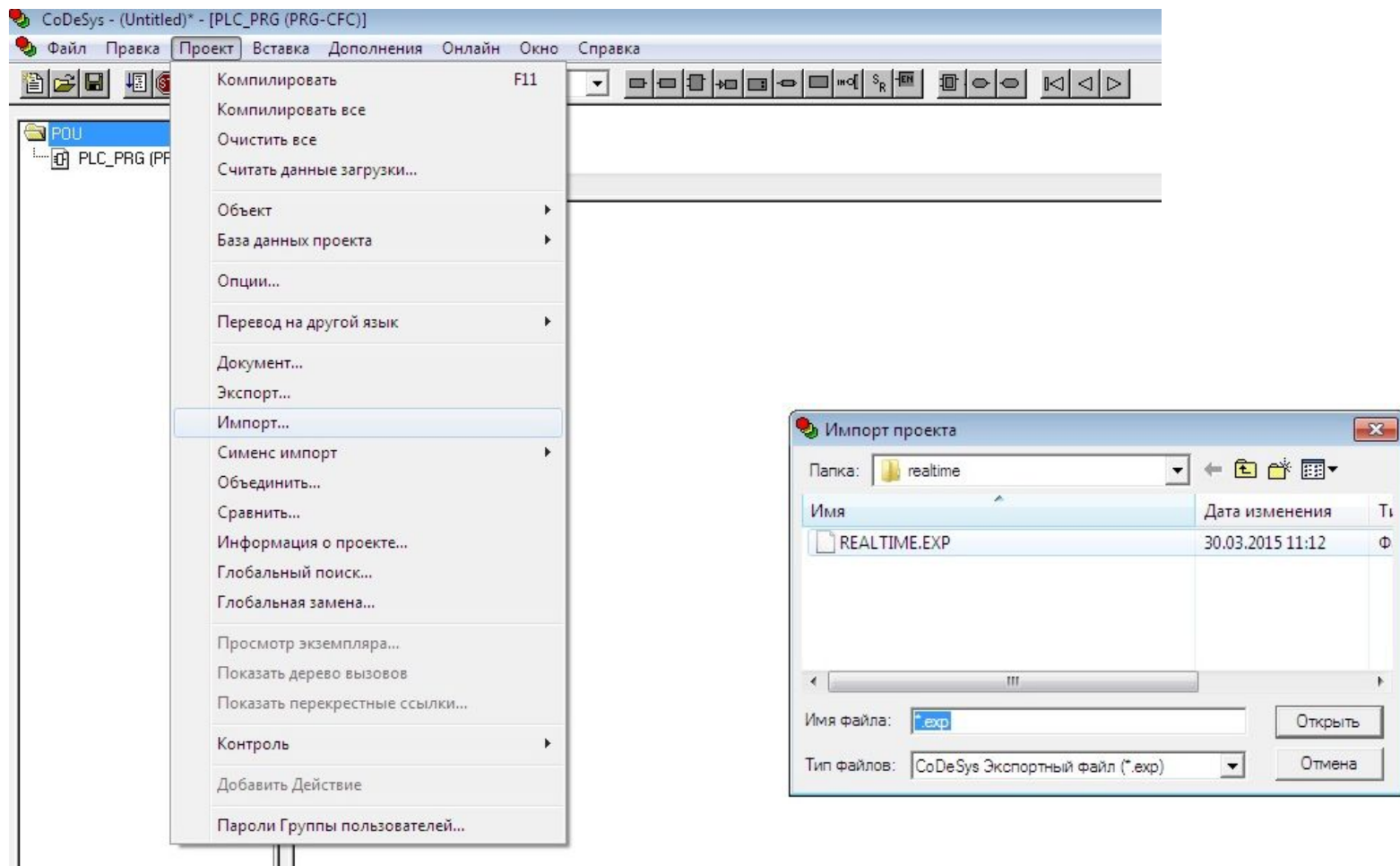
```
CoDeSys - (Untitled)* - [ПЛК-Браузер]
Файл  Правка  Проект  Вставить  Дополнения  Онлайн  Окно  Справка

Ресурсы
├── Глобальные переменные
├── библиотека ANALYZATION.L
├── библиотека IECSFC.LIB 13.4
├── библиотека SYSLIBTIME.LIB
├── библиотека SYSTASKINFO.L
├── Бортжурнал
├── Конфигурация ПЛК
├── Конфигурация задач
├── Конфигурация тревог
├── Менеджер библиотек
├── Менеджер параметров
├── Менеджер просмотра
├── Настройки целевой платформы
├── ПЛК-Браузер
├── Рабочая область
└── Цифровая трассировка

?
?
- show implemented commands
mem
- Memorydump
memc
- Memorydump relative to code-startaddress
memd
- Memorydump relative to data-startaddress
reflect
- reflect current command (test!)
dpt
- get data-pointer-table
ppt
- get POU-table
pid
- get project ID
pinf
- get project info
startprg
- Start PLC program
stopprg
- Stop PLC program
resetprg
- Reset PLC program
resetprgcold
- Reset PLC program cold
resetprgorg
- Reset PLC program original
reload
- Reload boot project
getprgprop
- Program properties
getprgstat
- Program status
filecopy
- Copy files [from] [to]
filerename
- Rename files [old] [new]
filedelete
- Delete file [filename]
filedir
- display directory list
rtsinfo
- runtime system information (version)
setpwd
- set login password
delpwd
- delete login password

GetTime - return current time and date
SetTime Format[ SetTime HH:MM:SS]
SetDate Format[ SetDate DD.MM.YYYY]
SetIP Format[ SetIP XXX.XXX.XXX.XXX]
SetGate Format[ SetGate XXX.XXX.XXX.XXX]
SetMask Format[ SetMask XXX.XXX.XXX.XXX]
PLCInfo Information about PLC
UpdateCore Update PLC Core binary
SetModemPort[ SetModemPort X]
SetModemCfg[ SetModemCfg X]
SetModemPortSp Format[ SetModemPortSp X]
SetDDNS Format[ SetDDNS X]
SetDHCP Format[ SetDHCP X]
flush
- save all data to flash
```

# Импортирование функционального блока realtime



# Функциональный блок realtime

The screenshot shows the CoDeSys IDE interface. The title bar reads "CoDeSys - (Untitled)\* - [PLC\_PRG (PRG-CFC)]". The menu bar includes "Файл", "Правка", "Проект", "Вставка", "Дополнения", "Онлайн", "Окно", and "Справка". The toolbar contains various icons for file operations and editing. The left sidebar shows a project tree with "POU" containing "PLC\_PRG (PRG)" and "realtime (FB)". The main editor displays the following code:

```
0001 PROGRAM PLC_PRG
0002 VAR
0003 END_VAR
0004
```

The "realtime" functional block is shown with the following variables:

- hour
- minute
- second
- Dayofweek

Two red arrows point from the text "Формат всех переменных UINT" to the "hour" and "minute" variables, indicating their data type.

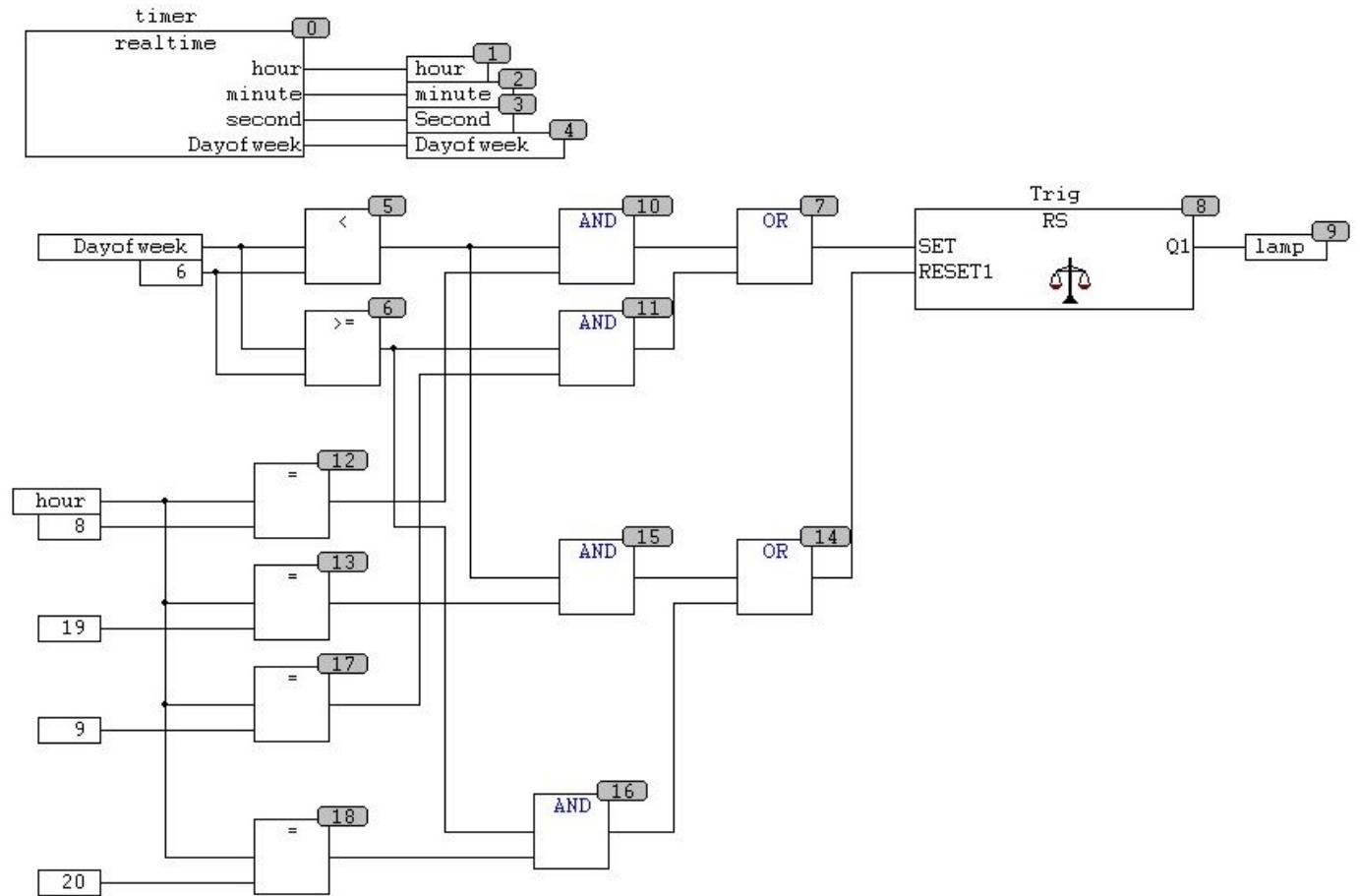


# Пример «Управление лампой по времени суток и дням недели»

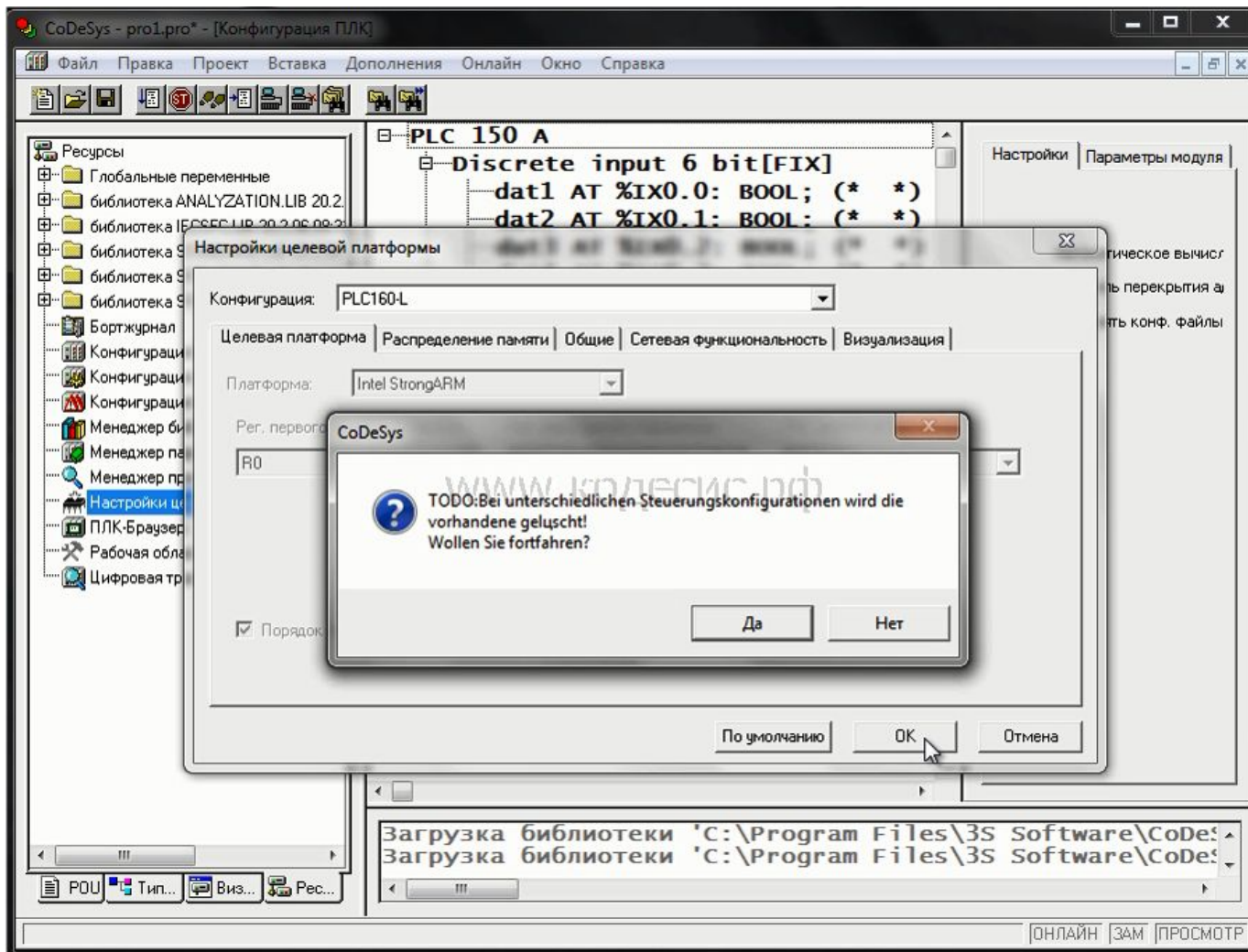
Лампа должна быть включена в рабочие дни недели  
(с понедельника до пятницы) с 8.00 до 19.00,  
а в субботу и воскресенье с 9.00 до 20.00

# Решение примера

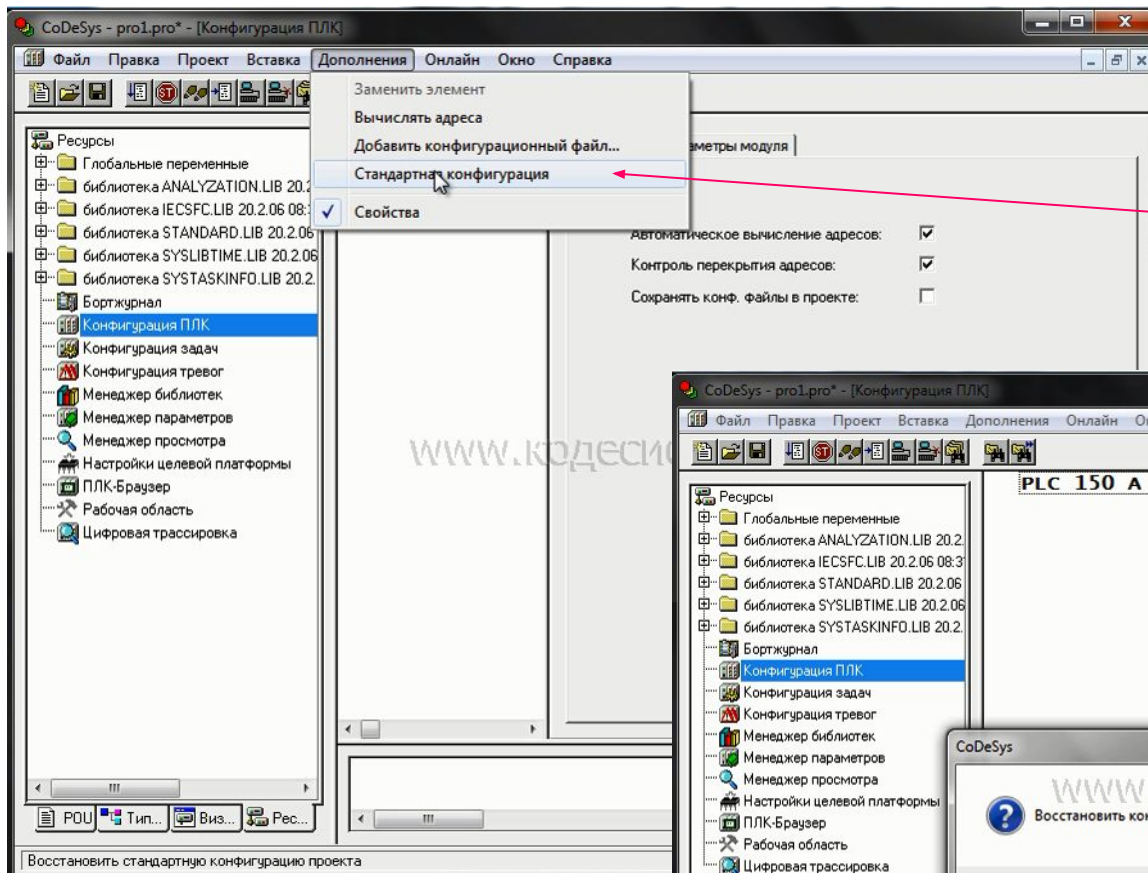
```
0001 PROGRAM PLC_PRG
0002 VAR
0003     hour: UINT;
0004     Dayofweek: UINT;
0005     Second: UINT;
0006     minute: UINT;
0007     Trig: RS;
0008     lamp: BOOL;
0009     timer: realtime;
0010 END_VAR
0011
```



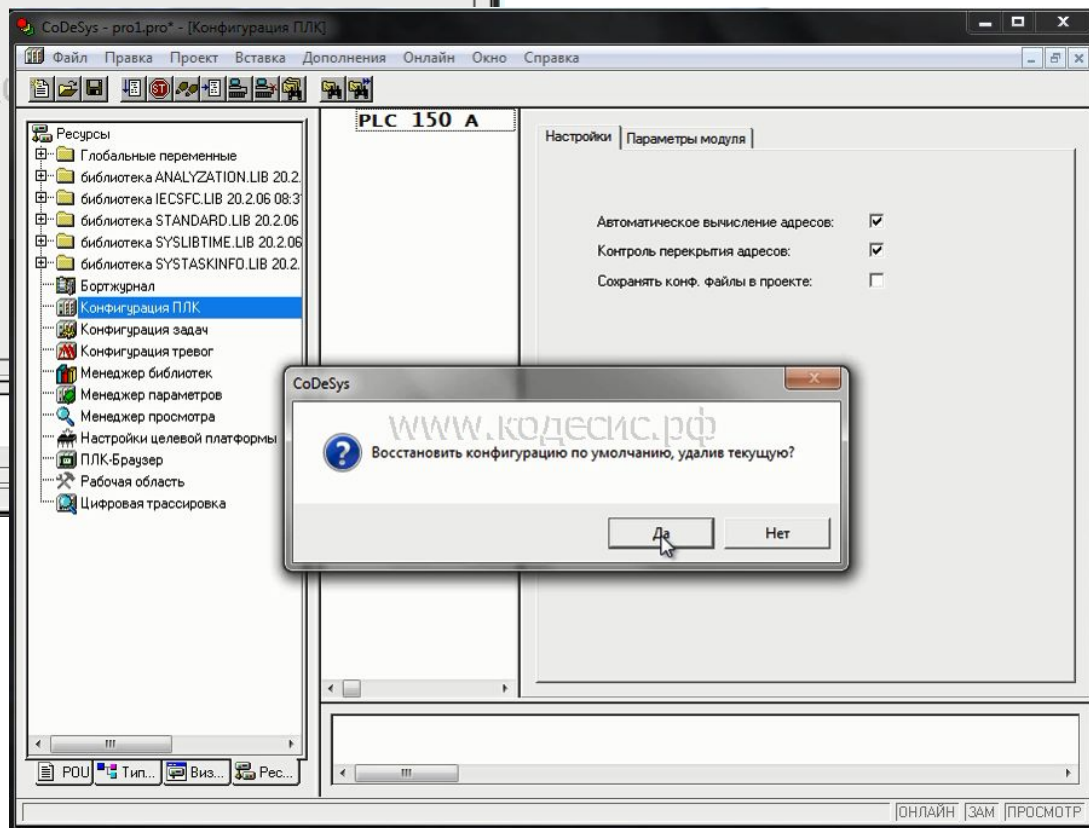
# Смена тагет-файла



В «Настройках целевой платформы» выберите нужную платформу (контроллер)

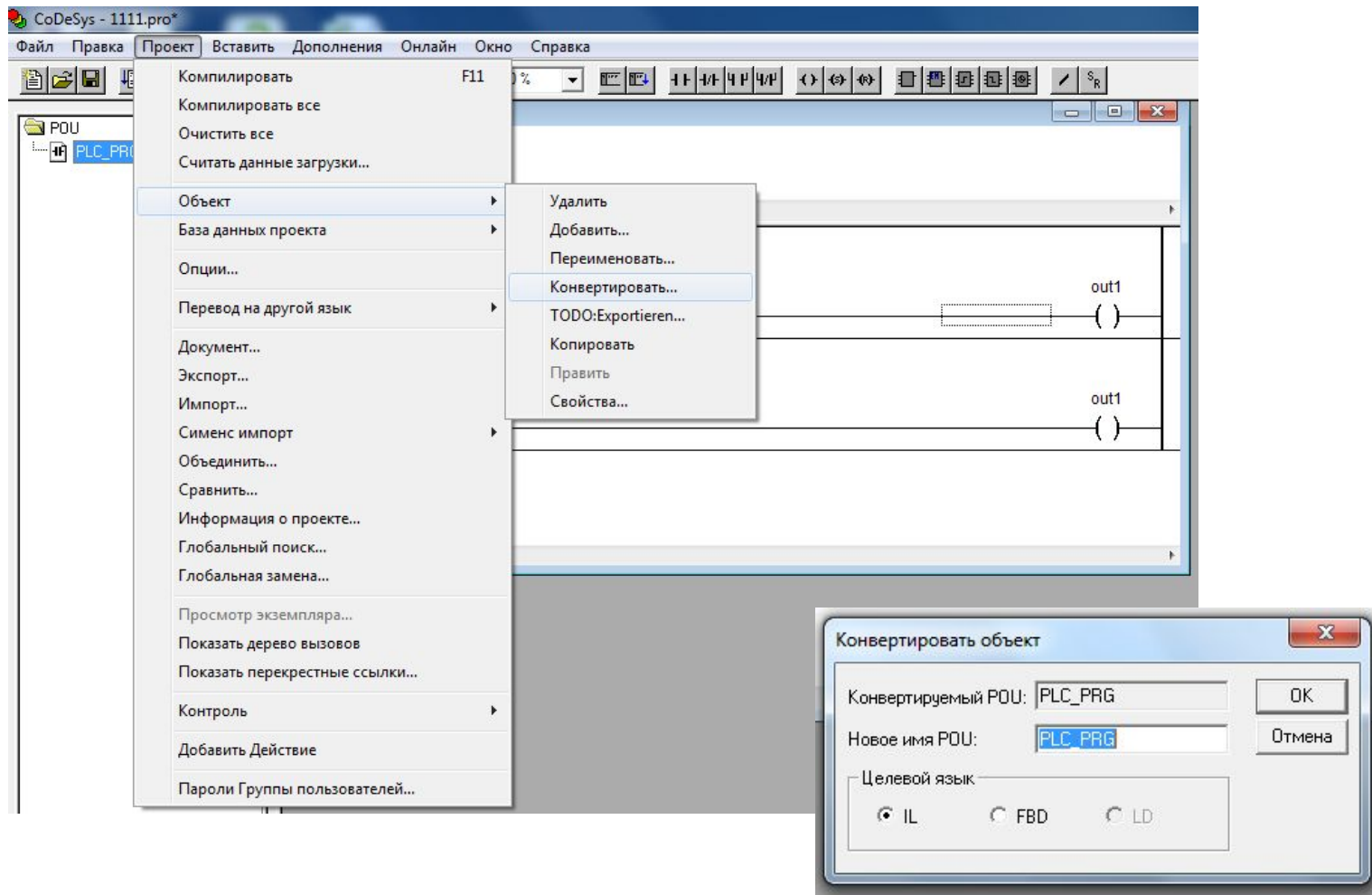


Подтвердите,  
что Вам нужна  
стандартная конфигурация  
выбранного контроллера



После этого необходимо конфигурацию вручную прописывать вновь!

# Переход с языка на язык



Имеются существенные ограничения!