Міністерство освіти і науки України

Державний вищий навчальний заклад

“Національний гірничий університет”

Авторизований навчальний центр “Schneider Electric”

м. Дніпропетровськ

2013

Что такое **CoDeSys?**

CoDeSys – пакет для создания программного обеспечения
для ПЛК в соответствии со стандартом МЭК 61131-3

- Инструмент программирования
- Инструмент отладки
- Инструмент тестирования
- Инструмент создания визуализаций
- Инструмент документирования проектов

Основные принципы стандарта МЭК **61131-3**

- Определяет принципы программирования ПЛК
- Включает хорошо известные и современные языки программирования
- Позволяет разработчику не зависеть от производителя системы программирования
- Повторное использование кода
- Стандарт является международным

Что определяет стандарт МЭК **61131-3**

- Структуру проекта
- Синтаксис и семантику 5 различных языков программирования: IL, FBD, LD, ST и SFC
- Типы строительных блоков проекта (POU): функции, программы и функциональные блоки
- Правила объявления и типы переменных

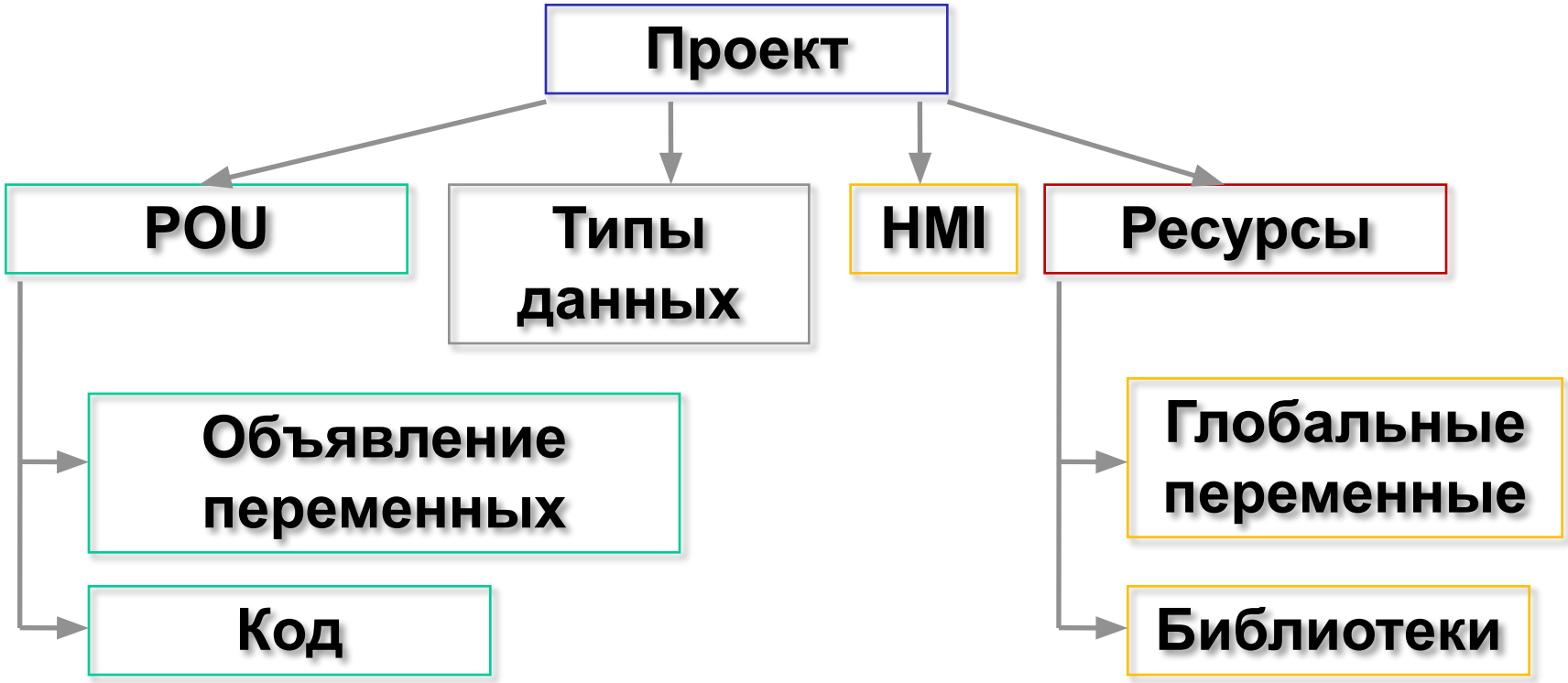
Введение в **CoDeSys**

- Состоит из двух частей : системы программирования и системы исполнения.

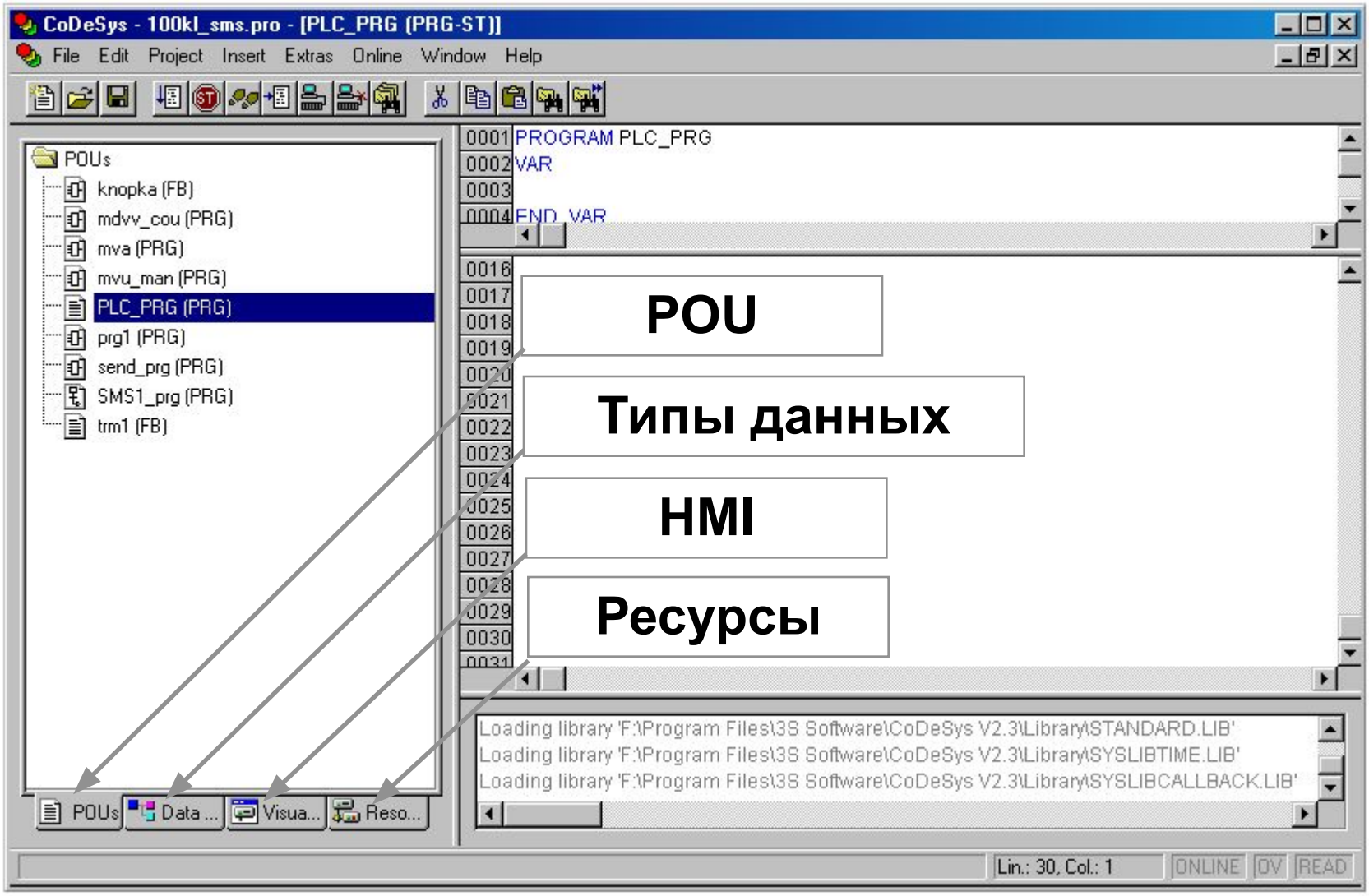
- Система программирования состоит из:
  - редактора, компилятора и отладчика МЭК проектов;
  - поддерживает все 5 языков программирования МЭК;
  - генерирует машинный код для довольно широкого набора процессоров.

- Система исполнения реализует:
  - управляющий цикл с обновлением входов/выходов;
  - связь с системой программирования;
  - загрузку приложения после включения питания контроллера.

Структура проекта



Структура проекта



Что такое проект в **CoDeSys ?**

- …хранится в одном файле (name.pro)
- …содержит программные компоненты (POU), визуализации, ресурсы и т.д.
- ... выполнение приложения начинается с POU PLC_PRG(аналог функции main)
- … выполняется циклически

Что такое **POU ?**

POU (Program organisation unit) –это программный модуль

POU PLC_PRG вызывается неявно системой исполнения

Стандарт МЭК 61131-3 определяет 3 типа POU

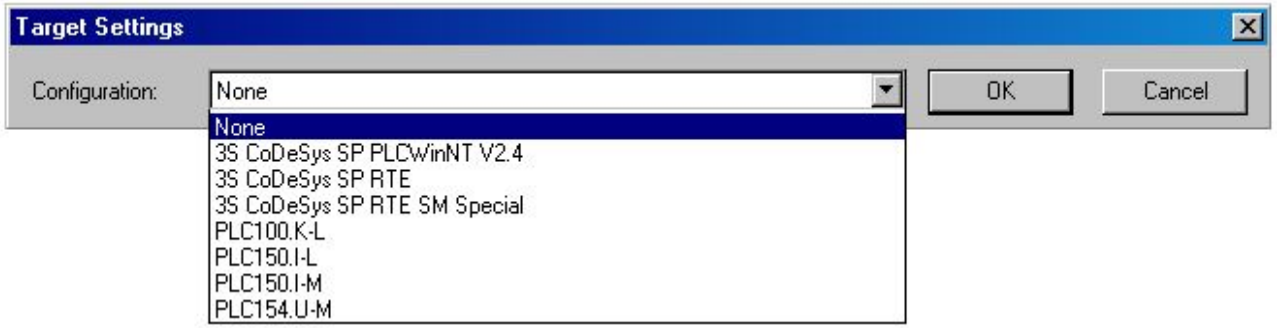
- Программы <PROGRAM>
- Функциональные блоки <FUNCTION_BLOCK>
- Функции <FUNCTION>

Главная программа PLC_PRG:

Для однозадачных систем программа **PLC_PRG** соответствует **OB1** в системах **S5/7**.

Эта программа вызывается циклически системой исполнения

Первый проект (Инкремент переменной)



- <File / New>
- Target Settings
- Создание главной программы PLC_PRG
- Автоматическое объявление
- <Online / Simulation>
- <Online / Login>
- <Online / Start>

Стандартные типы данных

- В МЭК 61131-3 определенны следующие типы данных:

Ключевое слово	Диапазон	Пример
BOOL	0 , 1	FALSE, TRUE, 0, 1
SINT, INT, DINT	-128 .. 127, -32768 .. 32767, -2147483648 .. 2147483647	0, 24453 -38099887
USINT, UINT, UDINT	0 .. 255, 0 .. 65535, 0 .. 4294967295	200, 47453 138099887
BYTE, WORD, DWORD	0 .. 255, 0 .. 65535, 0 .. 4294967295	8450 16#2102
REAL, LREAL	$-1.2x\ 10^{-38}$ .. $3.4x\ 10^{38}$ $-2.3x\ 10^{-308}$ .. $1.7x\ 10^{308}$	1.34996 2.8377E-15
TIME, TOD, DATE, DT	0 ms .. 1193h2m47s295ms 00:00:00 .. 23:59:59 01.01.1970 до. 06.02.2106	T#1d8h12m8s125ms TOD#12:34:17 D#2001-03-15 DT#2001-03-15-12:17:03
STRING	1 .. 255 символов	`Emergency Stop`

Представление данных в **CoDeSys**

3 метода объявления переменных

текстовый, табличный и автоматический

Локальные *(для 1 ФБ)* или Глобальные *(для всех ФБ)*

Сохраняемые и постоянные переменные

Синтаксис идентификаторов

- Буквы и цифры
- Должен начинаться с буквы
- Только одинарные подчеркивания
- Без пробелов
- Нельзя использовать зарезервированные слова МЭК и операторы
- Регистр не различается

- Примеры
  - Otto, otto, OTTO
  - Valve1
  - a_long_name

Основные команды режима **Online**

- <Online / Simulation Mode >

 <Online / Login [Alt+F8] / Logout [Ctrl+F8]>

 <Online / Start [F5]>

 <Online / Stop [Shift+F8]>

- <Online / Single Cycle>

 <Online / Breakpoint [F9]>

- <Online / Write Values [Ctrl+F7]>
- <Online / Force Values [F7]>
- <Online / Release Force [Shift+F7]>

Запуск приложения в целевой платформе

- Запустить систему исполнения
- Выключить режим эмуляции <Online / Simulation Mode>
- Настроить параметры связи <Online / Communication Parameter…>
- Осуществить вход <Online / Login>

Языки МЭК **61131-3**

- Список инструкций (IL)
- Структурированный текст (ST)
- Язык функциональных блоковых диаграмм (FBD)
- Язык релейных диаграмм (LD)
- Язык последовательных функциональных схем (SFC)

Список инструкций **(IL)**

- Текстовый язык
- Схож с ассемблером
- Все операции производятся через аккумулятор
- Легко читается в случае небольших программ
- Не поддерживает структурного программирования

Структурный текст **(ST)**

- Текстовый язык
- Язык высокого уровня
- Схож с Паскалем
- Лучший язык для программирования циклов и условий (IF, WHILE, FOR, CASE)

Язык релейных диаграмм**(LD)**

- Графический язык
- Программа состоит из схем
- Использовался для программирования практически всех классических ПЛК
- Удобен для программирования логических выражений
- Сложно использовать для работы с аналоговыми типами данных
- Переключение между FBD и LD

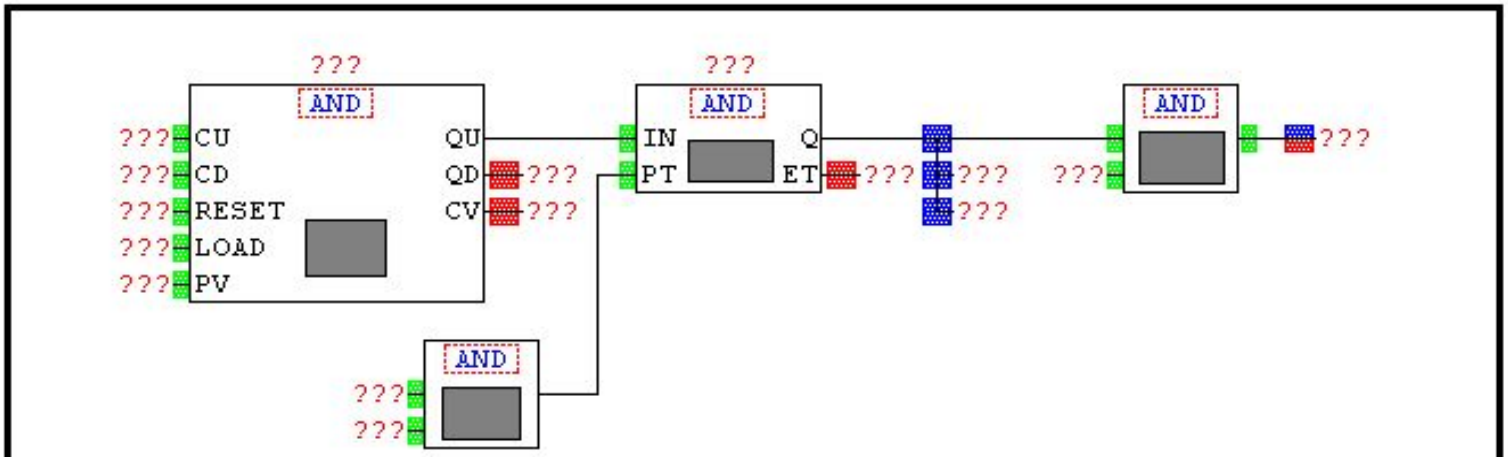
Язык функциональных блоковых диаграмм **(FBD)**

- Графический язык
- Программа состоит из нескольких схем
- Легко читается
- Каждая схема состоит из блоков и операндов

Непрерывные функциональные схемы **(CFC)**

- Схож с FBD, но…
- Блоки и соединители располагаются свободно
- Разрешаются циклы и свободные соединения

Язык функциональных блоковых диаграмм**(FBD)**



		[Выход] [Блок] [Присваивание] [Переход] [Возврат] [Инверсия]
		[Добавление входа]
		[Установка/Сброс]
		[Выход]
		[Выход] [Установка/Сброс]
???		**< Имя переменной / Имя экземпляра >**
AND		**<Имя оператора/функции/функционального блока/программы>**

ЛЕКЦИЯ 4

Язык последовательных функциональных схем **(SFC)**

- Графический язык
- Используется для структурирования приложений
- Состоит из шагов и переходов
- Действия выполняются внутри шагов
- Не конвертируется в другие языки
- CoDeSys поддерживает два типа SFC
- Подробнее будет рассмотрен завтра !

Упражнение **2**. Управление освещением в длинном коридоре

- Есть длинный коридор. Для управления освещением в коридоре используется три переключателя:
- Msw- главный переключатель
- Bsw – переключатель в начале коридора.
- Esw – переключатель в конце коридора.

Упражнение **2**. Управление освещением в длинном коридоре

Подача питания в коридор осуществляется с помощью переключателя Msw.

Необходимо решить задачу включения/выключения света с помощью любого из двух переключателей Bsw и Esw, установленных в разных концах коридора.

Т.е. при входе в коридор с одной стороны необходимо переключить Bsw, чтобы зажечь свет. На выходе с другой стороны коридора необходимо переключить Esw, чтобы свет погас. И наоборот.

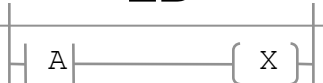
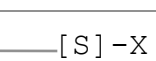
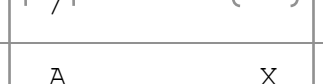
Операторы в **CoDeSys**

CoDeSys поддерживает все операторы МЭК 61131-3

- Оператор присваивания
- Битовые операторы
- Сдвиговые операторы
- Операторы сравнения
- Числовые операторы
- Работа с действительными числами
- Логарифмические операторы
- Тригонометрические операторы
- Операторы выбора

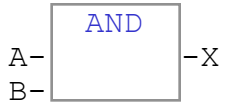
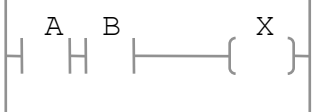
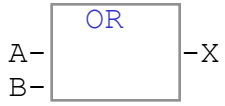
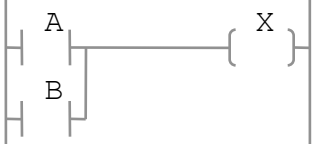
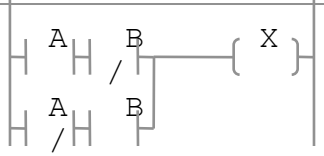
Операторы присваивания

- Используются для работы со всеми типами данных

Оператор	IL	FBD	LD	ST
LD / ST	LD A ST X			X := A;
LDN / ST	LDN A ST X			X := NOT(A);
LD / S	LD A S X			IF A THEN X := TRUE; END_IF
LD / R	LD A R X			IF A THEN X := FALSE; END_IF

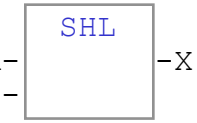
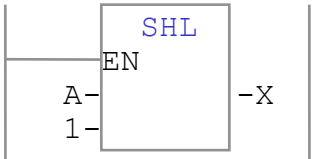
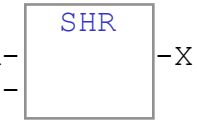
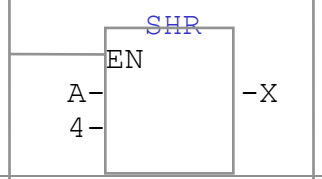
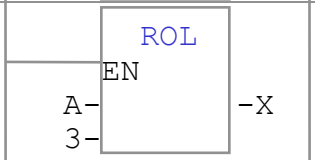
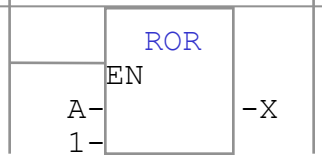
Битовые операторы

- Используются для работы с двоичными типами данных (BOOL, BYTE, WORD, DWORD)

Оператор	IL	FBD	LD	ST
NOT	LD A STN X			X := NOT(A);
AND	LD A AND B ST X			X := A AND B;
OR	LD A OR B ST X			X := A OR B;
XOR	LD A XOR B ST X			X := A XOR B;

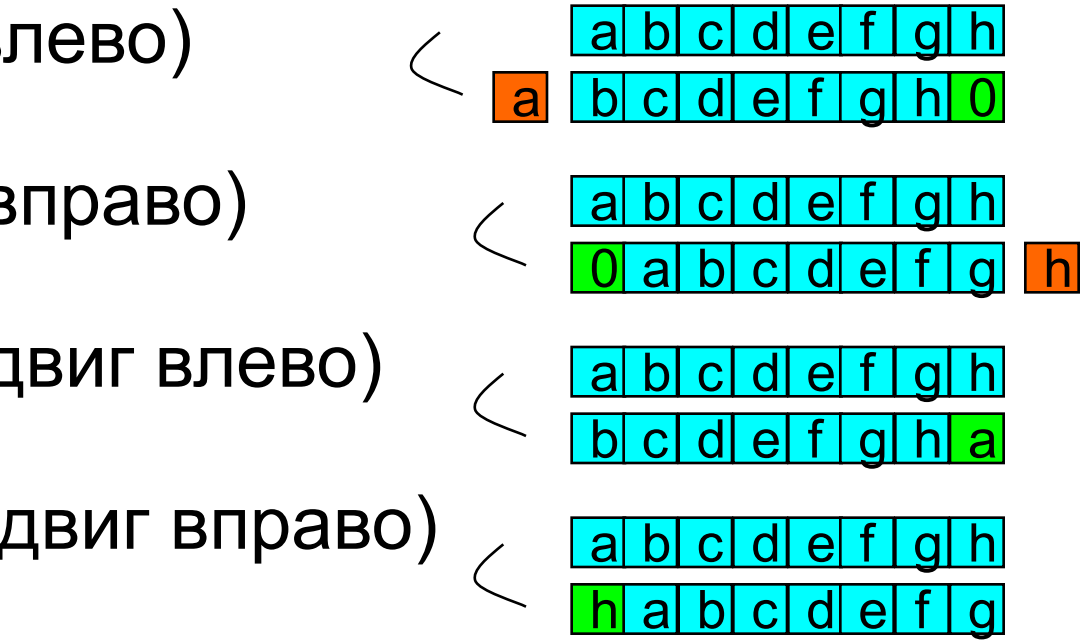
Сдвиговые операторы **(1)**

- Используются для работы с двоичными типами данных (BOOL, BYTE, WORD, DWORD)

Оператор	IL	FBD	LD	ST
SHL	LD A SHL 1 ST X			`X := SHL(A, 1);`
SHR	LD A SHR 4 ST X			`X := SHR(A, 4);`
ROL	LD A ROL 4 ST X			`X := ROL(A, 3);`
ROR	LD A ROR 1 ST X			`X := ROR(A, 1);`

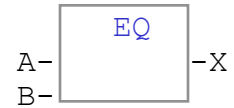
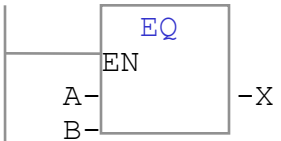
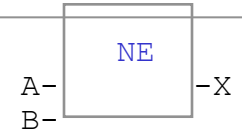
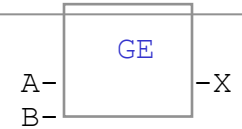
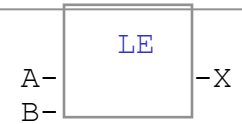
Сдвиговые операторы **(2)**

- SHL (сдвиг влево)
- SHR (сдвиг вправо)
- ROL (цикл. сдвиг влево)
- ROR (цикл. сдвиг вправо)



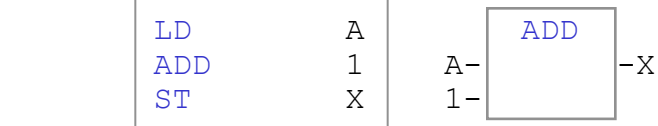
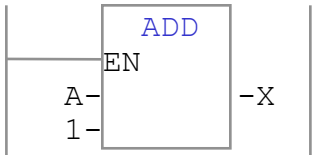
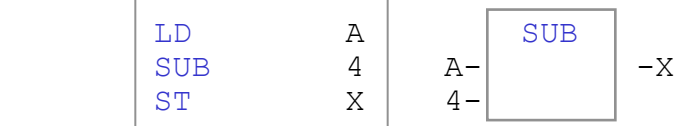
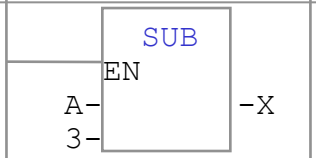
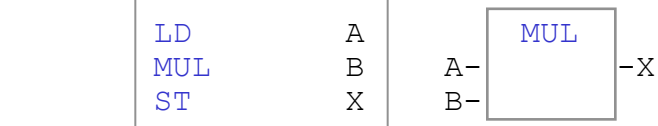
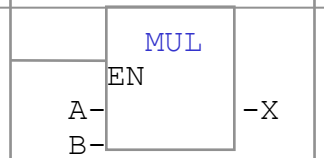
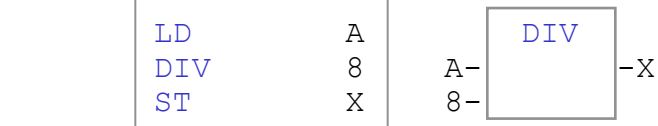
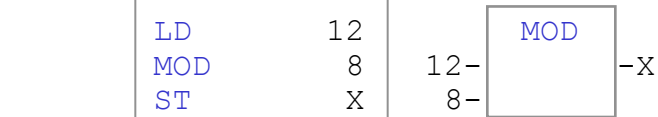
Операторы сравнения

- Используются для работы со всеми типами данных

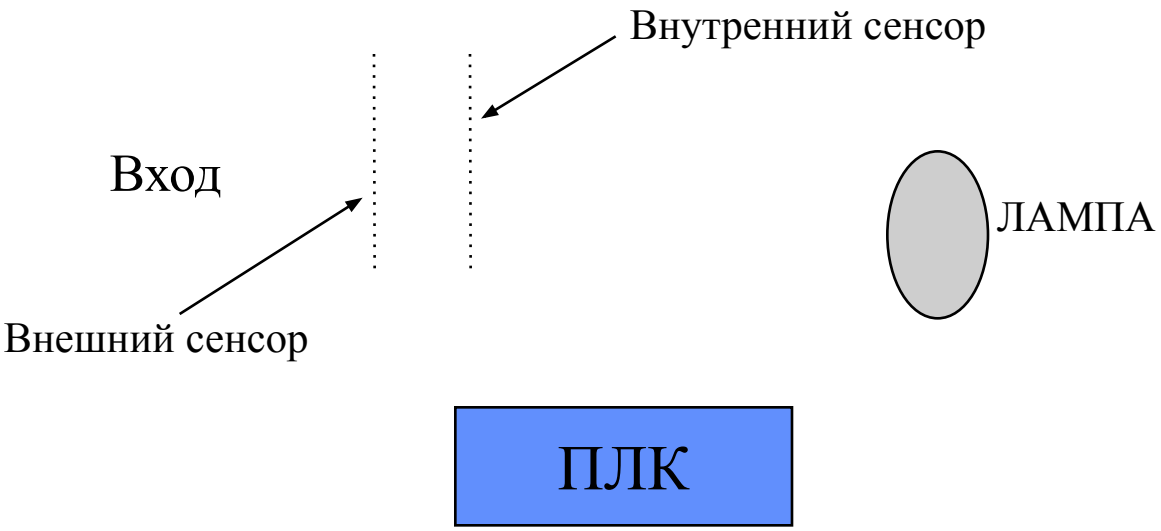
Оператор	IL	FBD	LD	ST
EQ	LD A EQ B ST X			`X := (A = B);`
NE	LD A NE B ST X		*(аналогично)*	`X := (A <> B);`
GE	LD A GE B ST X		*(аналогично)*	`X := (A >= B);`
GT	LD A GT B ST X		*(аналогично)*	`X := (A > B);`
LE	LD A LE B ST X		*(аналогично)*	`X := (A <= B);`
LT	LD A LT B ST X		*(аналогично)*	`X := (A < B);`

Арифметические операторы

- Выполняют алгебраические операции над целыми числами и числами с плавающей запятой

Оператор	IL	FBP	LD	ST
ADD	LD A ADD 1 ST X			X := A + 1;
SUB	LD A SUB 4 ST X			X := A - 4;
MUL	LD A MUL B ST X			X := A * B;
DIV	LD A DIV 8 ST X		*(аналогично)*	X := A / 8;
MOD	LD 12 MOD 8 ST X		*(аналогично)*	X := 12 MOD 8; *(Result = 4)* *(не исп. для REAL)*

Упражнение **3.** Управление освещением в комнате



Цель - свет должен быть выключен, когда в комнате никого нет!

Упражнение 3. Управление освещением в комнате

На входе установлены два дискретных датчика: один снаружи комнаты, другой внутри.

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.

Задача1: Если человек вошел – включить свет, Если человек вышел – выключить свет.

Задача2: Необходимо считать количество людей, заходящих и выходящих из комнаты.

Пока в комнате остается хотя бы один человек, свет должен быть включен.

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

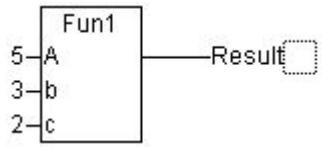
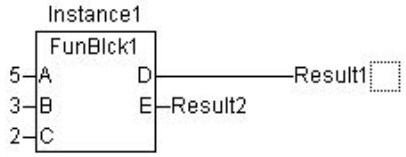
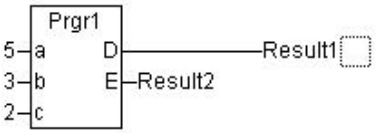
Функциональный блок

- 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

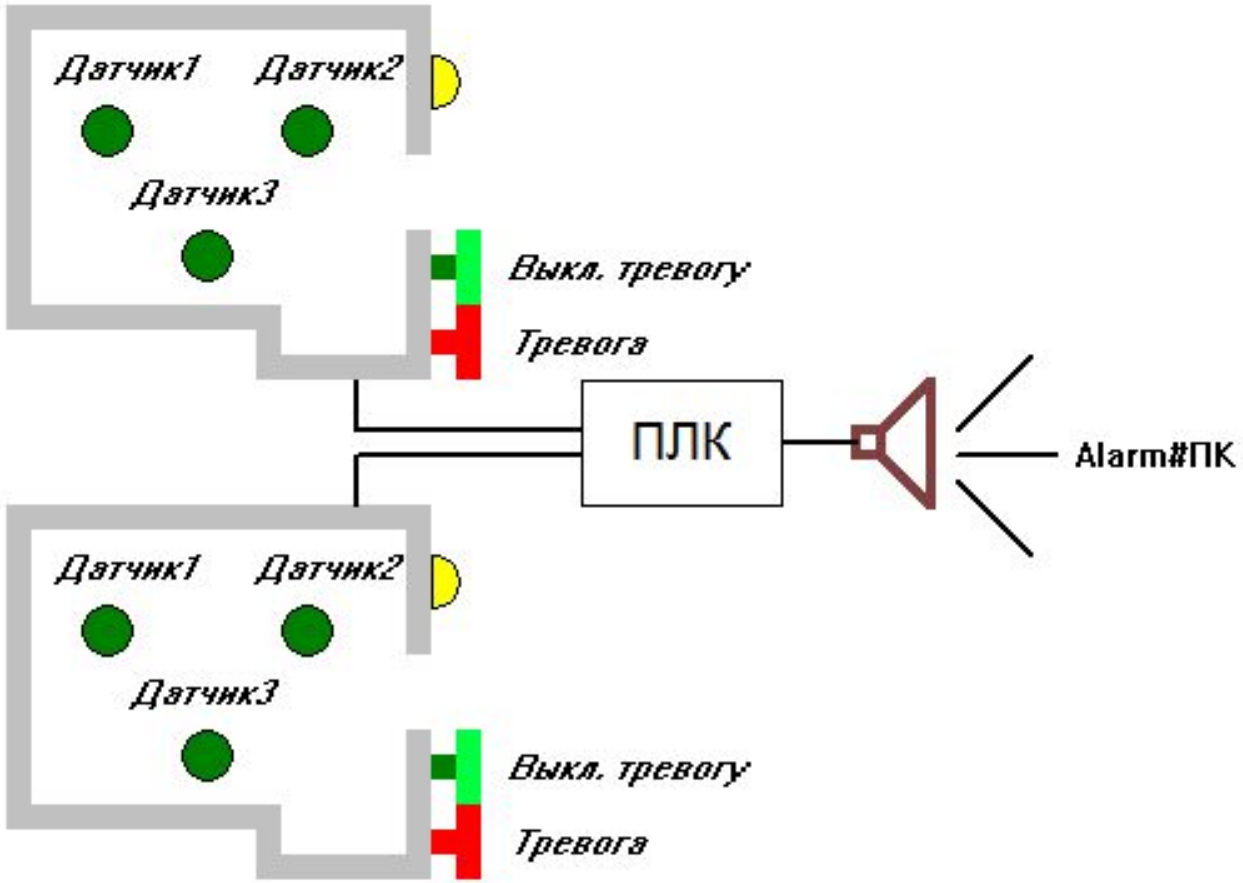
Вызов POU

	Функция	ФБ	Программа
Пример	**Function Fun1:INT** **3 входа (INT): A, B, C** **Переменная Result (int)**	**Function_Block FunBlck1** **3 входа (INT): A, B, C** **2 выхода (INT): D, E** **Экземпляр: Instance1**	**Program Prgr1** **3 входа (INT): A, B, C** **2 выхода (INT): D, E**
IL	**LD 5** **Fun1 3,2** **ST Result**	**CAL Instance1(A:=5, B:=3, C:=2)** **...** **LD Instance1.D** **ST Result1** **LD Instance1.E** **ST Result2**	**CAL Prgr1(a := 5, b := 3, c := 2)** **...** **LD Prgr1.D** **ST Result1** **LD Prgr1.E** **ST Result2**
ST	**Result:=Fun1(5,3,2);** **или** **Result:=Fun1(A:=5,B:=3,C:=2);**	**Instance1(A:=5, B:=3, C:=2,** **D => Result1, E => Result2);** **или** **Result1:=Instance1.D; …**	**Prgr1(A := 5, B := 3, C := 2,** **D => Result1, E => Result2);** **или** **Result1:= Prgr1.D; …**
LD **FBD** **CFC**			

Упражнение **4.** Работа с программными компонентами **CoDeSys (POU)**

- Функция расчета мощности постоянного тока по напряжению и сопротивлению
- Счетчик положительных фронтов дискретного сигнала
- Вызов функций и функциональных блоков из программы

Упражнение **5.** Система пожарной сигнализации здания



Упражнение **5.** Система пожарной сигнализации здания

В здании две одинаковые комнаты.

В каждой комнате установлено три пожарных датчика, кнопка ручного включения сигнализации и кнопка ручного отключения сигнализации. Для каждой комнаты предусмотрена сигнальная лампа. Сигнализация пожара 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 включения вне зависимости от состояния датчиков.

Сложные типы данных

- Массив

```
abList : ARRAY[0..31] OF BOOL;
```

- Структура

```
TYPE SetType :
    STRUCT
        iCount   : INT;
        rValue   : ARRAY[0..9] OF REAL;
    END_STRUCT
END_TYPE
```

- Перечисление

```
TYPE ColorType :
    ( RED, YELLOW, GREEN, BLUE );
END_TYPE
```

- Псевдоним

```
TYPE Message : STRING(40); END_TYPE
```

Предопределенные блоки (Библиотеки)

- Библиотека состоит из объектов, которые могут быть использованы в различных проектах
- Пользователь может создавать и использовать собственные библиотеки.
- Можно создавать библиотеки с защитой.
- Библиотеки могут быть написаны не только на МЭК, но и на других языках программирования
- Библиотека standard.lib содержит POU описанные в стандарте МЭК

Стандартная библиотека

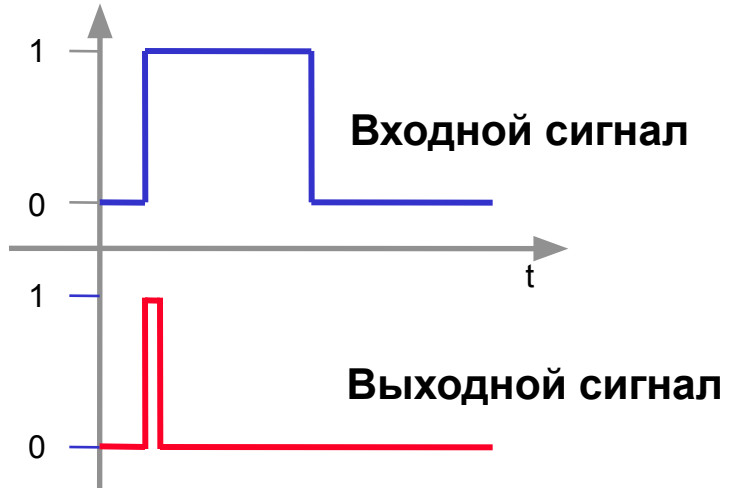
- **Функции работы со строками**
- **Детекторы фронтов**
- **Счетчики**
- **Таймеры**

Функции работы со строками

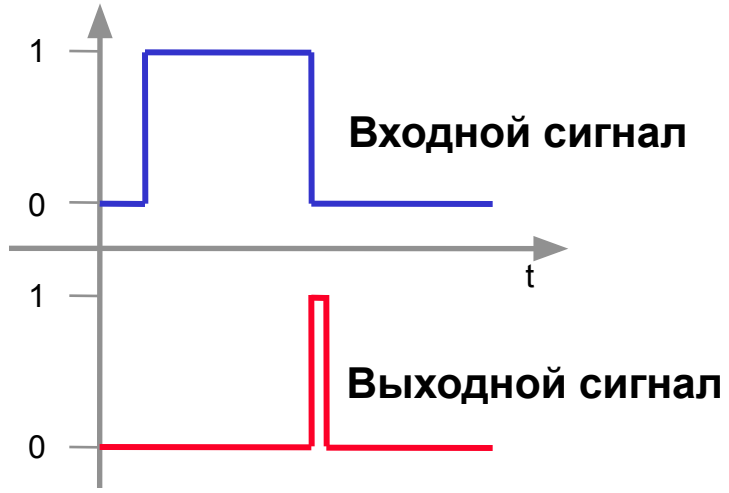
- **LEN**
- **LEFT**
- **RIGHT**
- **MID**
- **CONCAT**
- **INSERT**
- **DELETE**
- **REPLACE**
- **FIND**

Детекторы фронтов

- **R_TRIG**
  определяет передний фронт



- **F_TRIG**
  определяет задний фронт



Счетчики

- **CTU**
  Инкрементируется по переднему фронту

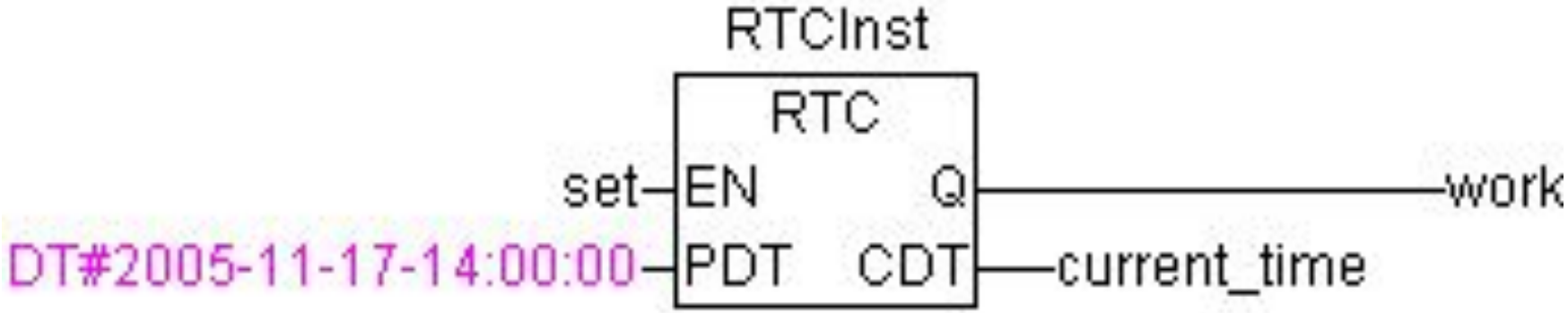
- **CTD**
  Декрементируется по переднему фронту

- **CTUD**
  Инкрементируется или декрементируется по разным входам

Временные типы данных МЭК **61131-3**

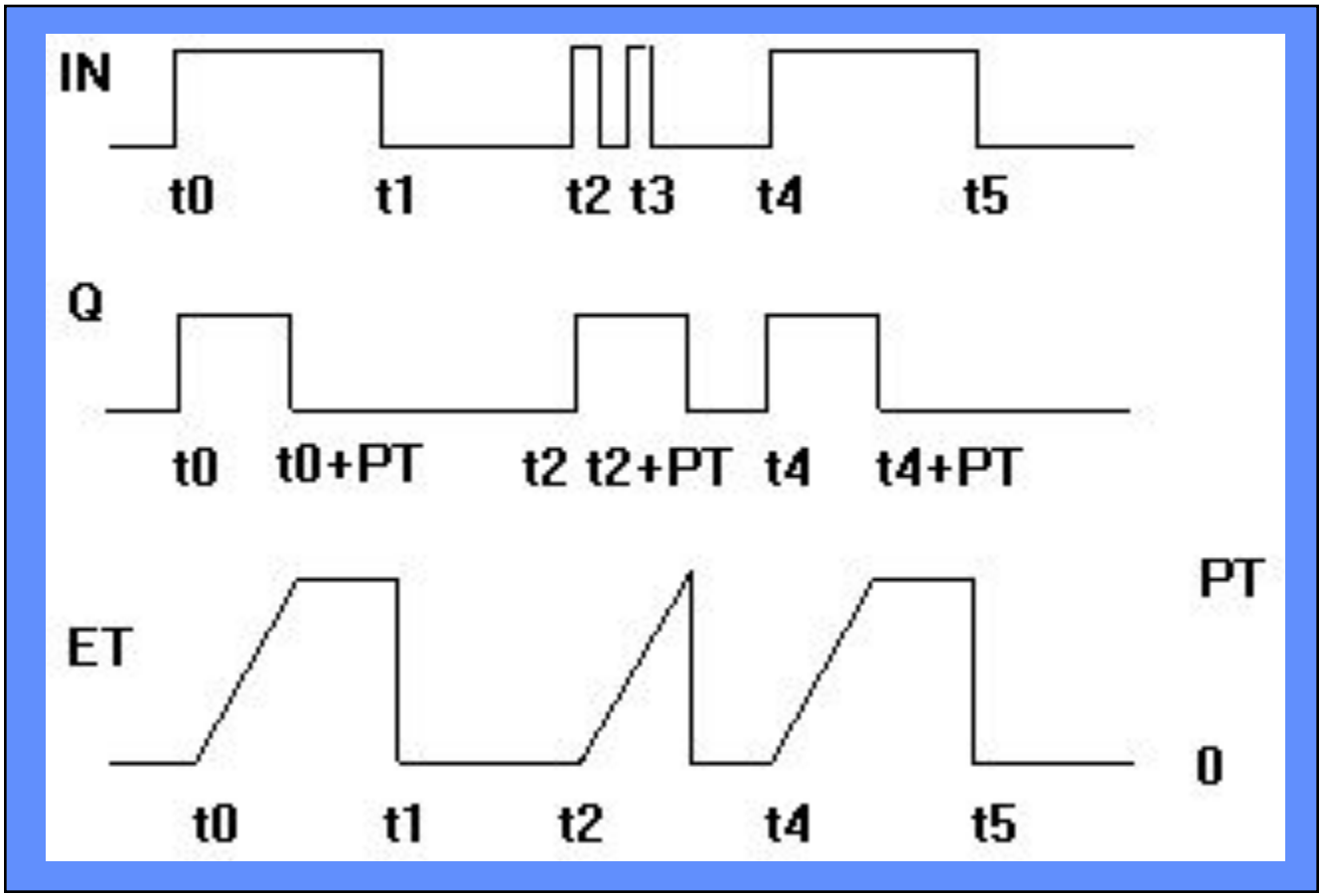
Тип	Описание	Пример
TIME	Используются для выражения интервалов времени	T1:=T#5h45m10s9ms T2:=T#100ms
DATE	Используются для выражения даты	DATE#1996-05-06 d#1972-03-29
TIME_OF_DAY	Используются для выражения времени дня	TIME_OF_DAY#15:36:30.123 tod#00:00:00
DATE_AND_TIME	Используются для выражения даты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

Часы реального времени RTC



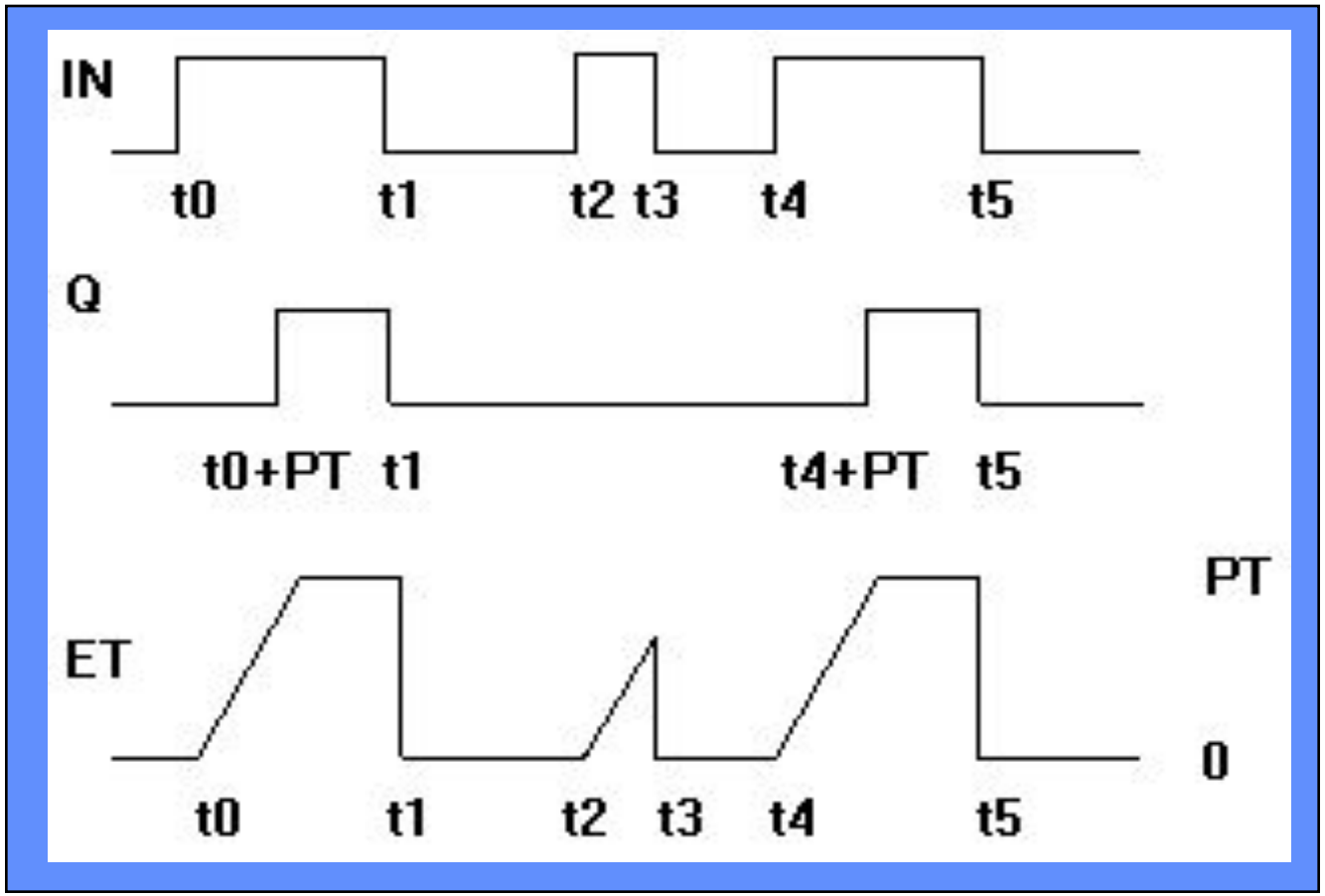
Таймер **TP**

Генерирует импульс заданной длительности



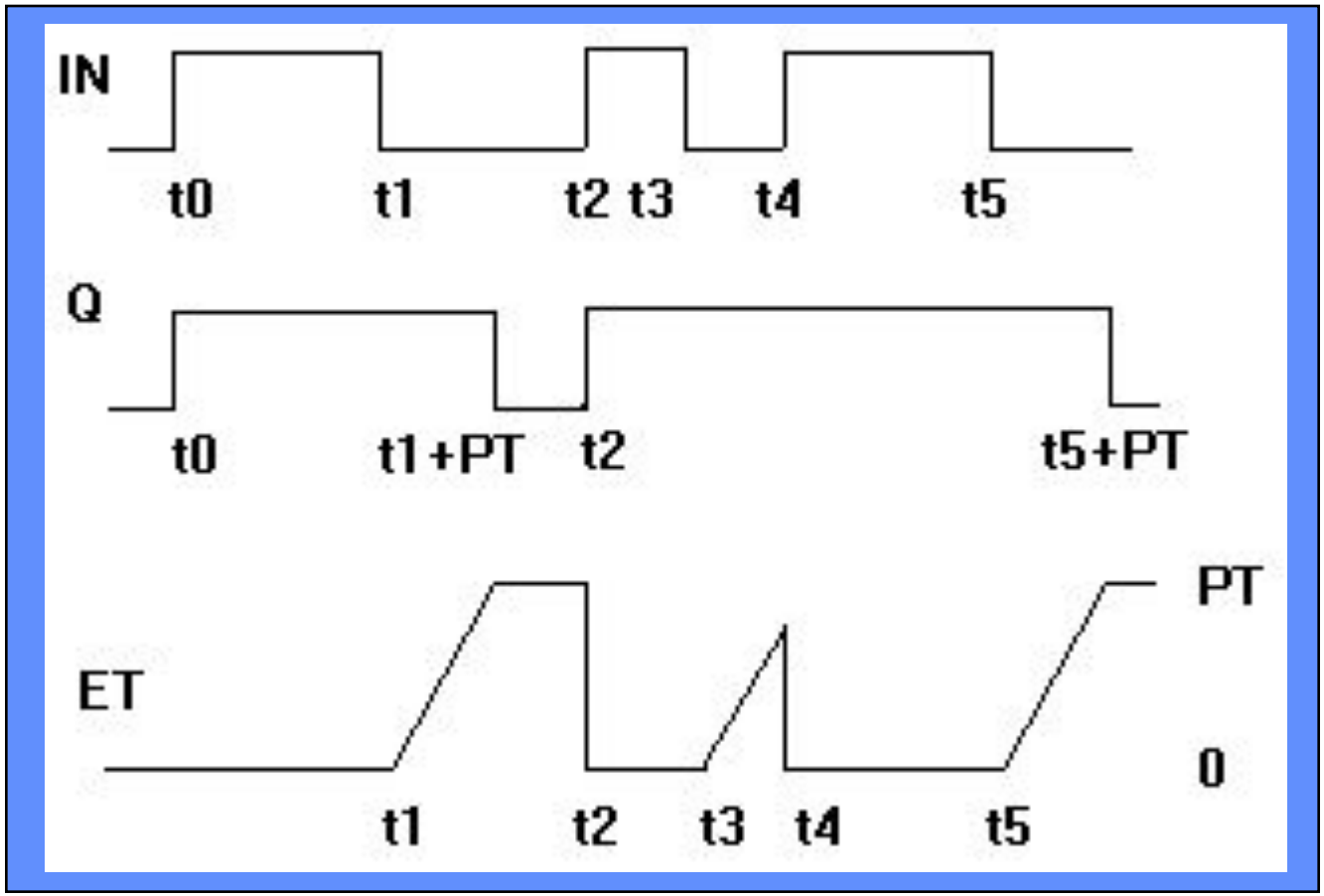
Таймер **TON**

Включает выход с задержкой по переднему фронту



Таймер **TOF**

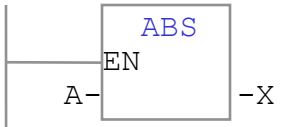
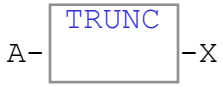
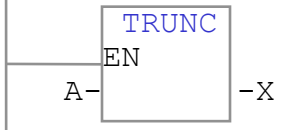
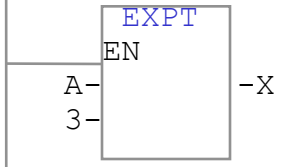
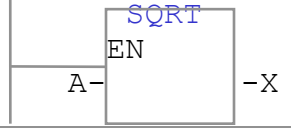
Выключает выход с задержкой по заднему фронту



Упражнение 6. Работа с элементами стандартной библиотеки

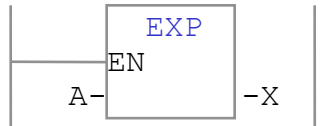
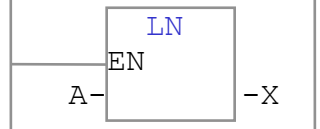
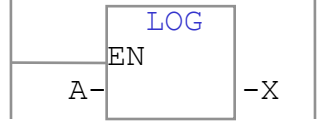
- Реализовать задачу управления светом комнате (упражнение 3) с помощью компонентов стандартной библиотеки. Свет должен выключаться через 5 секунд, после того как последний человек покинет комнату.

Операторы для работы с числами с плавающей запятой

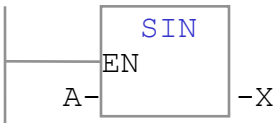
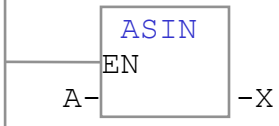
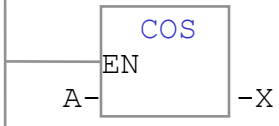
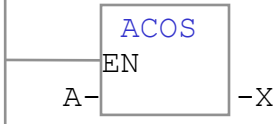
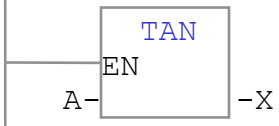
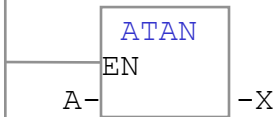
Оператор	IL	FBD	LD	ST
ABS	LD A ABS ST X			X := ABS(A); (Result = 12) (if A = -12.0)
TRUNC	LD A TRUNC ST X			X := TRUNC(A); (Result = 4) (if A = 4.32)
EXPT	LD A EXPT 3 ST X			X := EXPT(A,3); (Result = 8) (if A = 2)
SQRT	LD A SQRT ST X			X := SQRT(A); (Result = 5) (if A = 25)

Логарифмические операторы

- Вычисление логарифмов и экспоненты

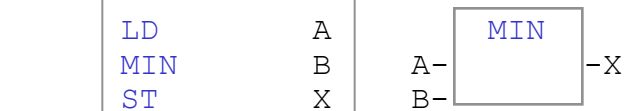
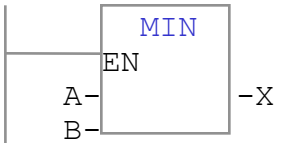
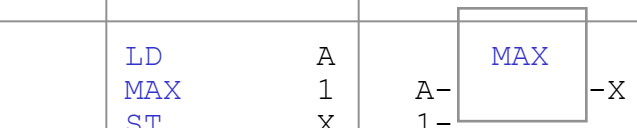
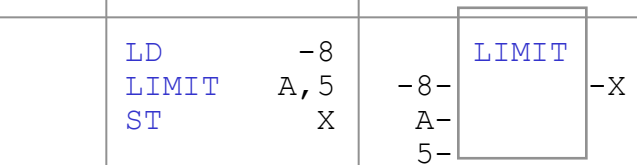
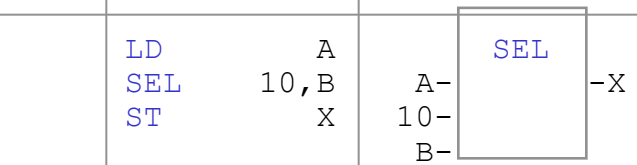
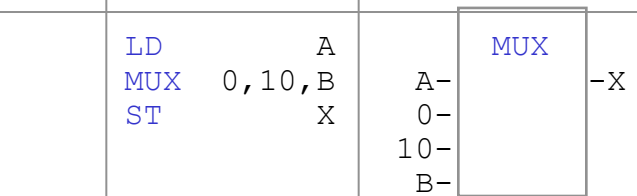
Оператор	IL	FBD	LD	ST
EXP	LD A EXP ST X			X := EXP(A); (Result = 7.389) (if A = 2)
LN	LD A LN ST X			X := LN(A); (Result = 2) (if A = 7.389)
LOG	LD A LOG ST X			X := LOG(A); (Result = 3) (if A = 1000)

Тригонометрические операторы

Оператор	IL	FBD	LD	ST
SIN	LD A SIN ST X			X := SIN(A);
ASIN	LD A ASIN ST X			X := ASIN(A);
COS	LD A COS ST X			X := COS(A);
ACOS	LD A ACOS ST X			X := ACOS(A);
TAN	LD A TAN ST X			X := TAN(A);
ATAN	LD A ATAN ST X			X := ATAN(A);

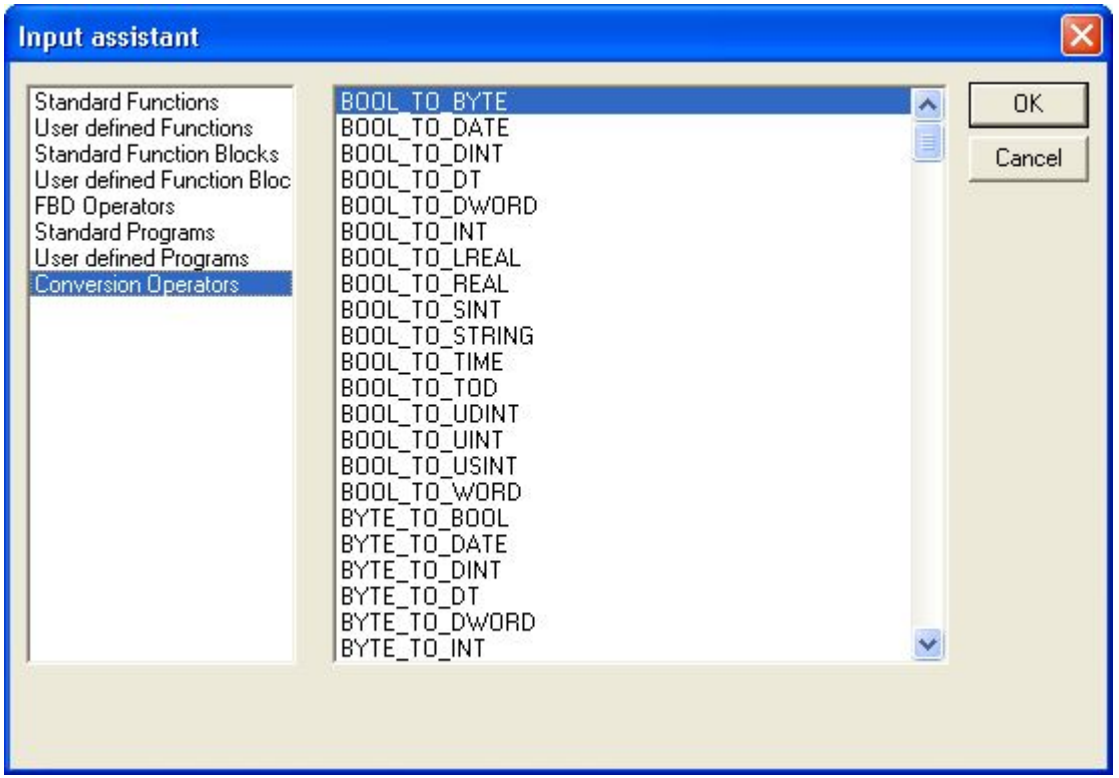
Операторы выбора

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

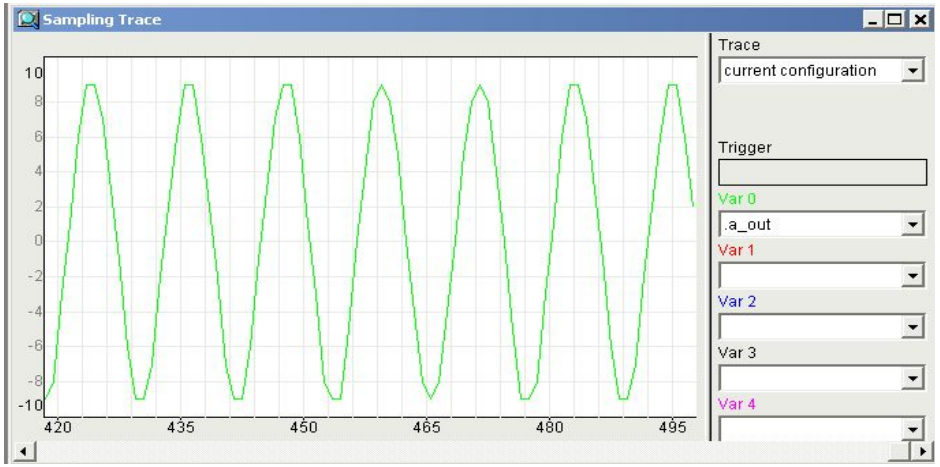
Оператор	IL	FBD	LD	ST
MIN	LD A MIN B ST X			X := MIN(A,B);
MAX	LD A MAX 1 ST X		*(как выше)*	X := MAX(A,1);
LIMIT	LD -8 LIMIT A,5 ST X		*(как выше)*	X := LIMIT(-8,A,5); ***X = -8 if A < -8*** ***X = 5 if A > 5***
SEL	LD A SEL 10,B ST X		*(как выше)*	X := SEL(A,10,B); ***X = 10 if A is FALSE*** ***X = B if A is TRUE***
MUX	LD A MUX 0,10,B ST X		*(как выше)*	X := MUX(A,0,10,B); ***X = 0 if A is 0*** ***X = 10 if A is 1*** ***X = B if A is 2***

Операторы преобразования типов данных

- Для каждой пары типов данных используется отдельная функция



Упражнение **7.** Генератор синусоиды



- Операции с вещественными числами
- Преобразование типов
- Первое знакомство с трассировкой

Язык Последовательных Функциональных диаграмм **(SFC)**

- Графический язык
- Управление последовательностью выполнения действия
- Состоит из шагов, действий и переходов
- Помогает структурировать приложение
- В CoDeSys есть упрощенная версия SFC

Упражнение 8. Управление сверлильным станком

Станок производит сверление отверстий в заготовках по заданной программе: запуск станка, опускание сверла, сверление по одному из выбранных режимов, подъем сверла.

На станке предусмотрена кнопка запуска, тумблер выбора режима сверления, кнопка останова сверления.

Контроллер подает три управляющие команды: опускание сверла, подъем сверла, сверление.

Предусмотрено два режима: либо сверление производится в течение 5 секунд (автоматический режим), либо сверление производится до нажатия оператором кнопки останова сверления. Режим выбирается с помощью тумблера выбора перед запуском станка.

Упражнение 8. Управление сверлильным станком

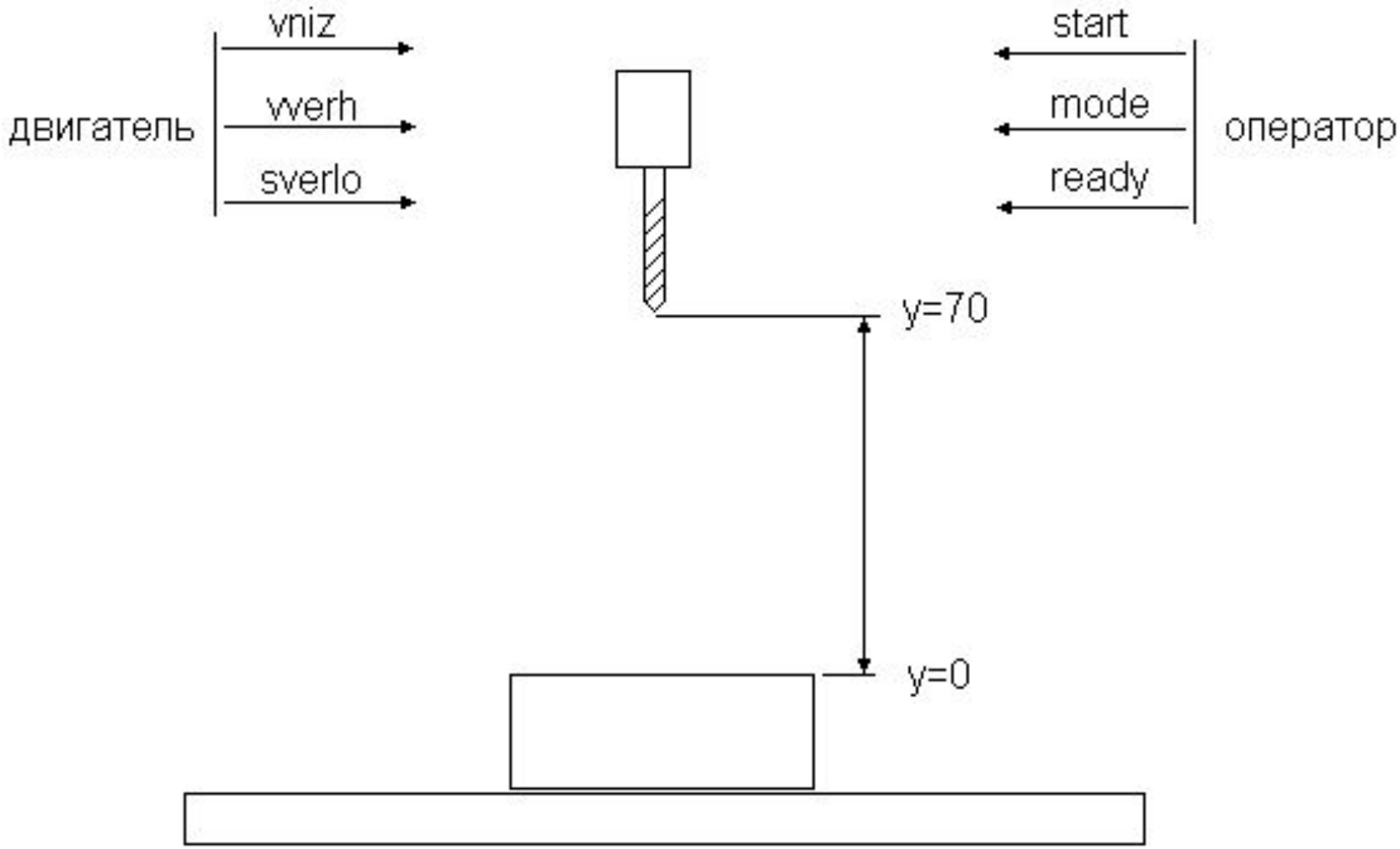
Перед началом работы оператор с помощью тумблера выбора определяет режим сверления.

После нажатия оператором кнопки запуска контроллер начинает управление станком. Подается команда опустить сверло и начинается обратный отсчет координаты. При достижении нижней точки (**y**=0) снимается команда на опускание и подается команда на сверление.

Если выбран первый режим, то команда сверления снимается через 5 секунд. Если выбран второй режим, то команда сверления снимается после нажатия оператором кнопки останова сверления.

Затем контролер подает команду на подъем сверла и начинает прямой отсчет координаты. После достижения верхнего положения (y=70) команда подъема снимается.

Упражнение 8. Управление сверлильным станком



Конфигурирование задач

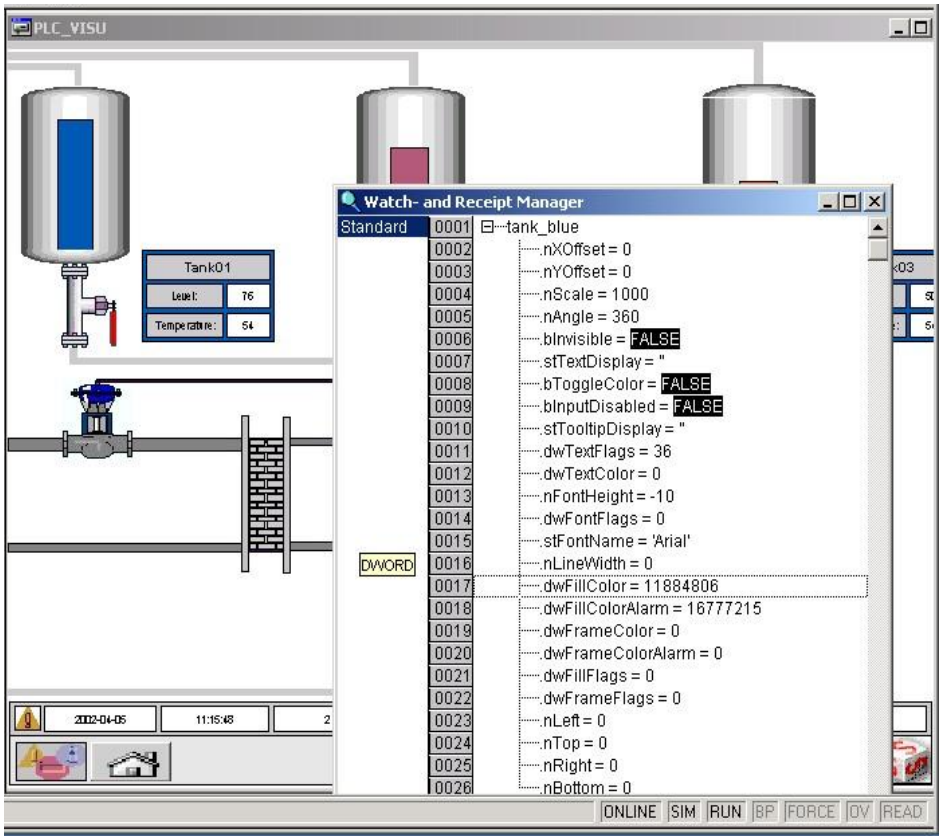
- Задачи выполняются по событию или циклически
- Имеют приоритет
- Вызывают программы
- Есть свободно-выполняемые задачи(аналог idle)

Упражнение 9. Работа с конфигуратором задач

- Создать циклическую задачу
- Создать задачу, выполняемую по событию
- Создать свободно-выполняемую задачу
- Создать программы – счетчики числа запусков задач
- Проследить за выполнением свободно-выполняемой задачи, изменяя параметры других задач

Визуализация

- Доступ ко всем данным проекта
- Графическое отображение логических и численных значений
- Ввод логических и численных значений
- Перемещение графических объектов



Инструменты визуализации