AppMaker

➢*ISaGRAF 3.5*

Вступление *Introduction*

ICS Triplex ISaGRAF Inc.

Formerly Altersys Inc.

- **Зарегистрирована в 1984**
  - **Incorporated in 1984**
- **Разработчик программ ISaGRAF 3 и ISaGRAF 4**
  - **Designer of ISaGRAF 3 and ISaGRAF 4**
- **Разработчик HiBeam**
  - **Designer of HiBeam**
- **Приобретена компанией ICS Triplex в 2003**
  - **Acquired by ICS Triplex in 2003**

Цель – разработка Программ, позволяющих системным интеграторам и производителям оборудования создавать автоматические системы в реальном времени.

Aim to develop Software tools to allow integrators and Hardware Manufacturers to create Real Time Automation systems.

Объекты: Virtual PLC Objectives: Virtual PLC

- **Независимость от оборудования и операционной системы**
  - **Виртуальный легко переносимый PLC**
    - **Hardware & OS independence**
      - **Virtual highly portable PLC**
- **Совместимость и сертификация**
  - **Windows 98, NT, 2000, XP**
  - **IEC 61131, Блок схема**
    - **Compatibility & certification**
      - **Windows 98, NT, 2000, XP**
      - **IEC 61131, Flow Chart**
- **Член PLCopen**
  - **сертификация IL PLCopen**
    - **PLCopen member**
      - **IL PLCopen certification**
- **Международная дистрибуция**
  - **Английский, Французский, Немецкий, Японский, Испанский, Китайский...**
    - **International distribution**
      - **English, French, German, Japanese, Spanish, Chinese,...**

Стандарты по Управлению Процессами IEC 61131
Process-Control Standards IEC 61131

- **IEC 848 : Подготовка функциональных схем**
  - Графические элементы
  - Правила обработки
- **IEC 1131**
  - **IEC 61131-1**
    - Общее обозрение
  - **IEC 61131-2**
    - Оборудование
  - **IEC 61131-3**
    - Языки Программирования
  - **IEC 61131-4**
    - Руководства пользователя
  - **IEC 61131-5 <> IEC 1499**
    - Связь Распределитель

- **IEC 848 : Preparation of function charts**
  - Graphic elements
  - Execution rules
- **IEC 1131**
  - **IEC 61131-1**
    - General Overview
  - **IEC 61131-2**
    - Hardware
  - **IEC 61131-3**
    - Programming Languages
  - **IEC 61131-4**
    - User Guidelines
  - **IEC 61131-5 <> IEC 1499**
    - Communication Distribution model

Стандарты по Управлению Процессами
Process-Control Standards

- **IEC 61131-3 : Языки программирования для PLC**
  - Модель Программы
  - Модель Данных
  - Графические языки
  - Текстовые языки

- **IEC 61131-3 : Programming languages for PLCs**
  - The software model
  - The Data model
  - Graphic languages
  - Textual languages

Пользователи ISaGRAF
ISaGRAF Users

- **пользователи PLC с языками по IEC 61131-3 languages**
  - **PLC users request IEC 61131-3 languages**
- **системы логического управления на базе ПК**
  - **PC based Logic control systems**
- **разработчики и производители оборудования**
  - **Hardware designers & manufacturers**
- **Отдельное средство для системных интеграторов и производителей оборудования**
  - **Single tool for Integrators and OEMs**

Конфигурация ISaGRAF
ISaGRAF ***Configuration***

Загрузка / Отладка

Download / Debug

Рабочее место ISaGRAF
The ISaGRAF **Workbench**

Ядро ISaGRAF
Виртуальный PLC
Программируемый процессор

The ISaGRAF **kernel**
Virtual PLC
Software processor

Конфигурация системы
System Configuration

Modbus / Televis (*)

Modbus / Televis (*)

Любая операционная система
MS-Windows 3.1x, 95, 98, NT, 2000

Any MS-Windows system
98, NT, 2000

Микропроцессор
Fujitsu MB90F543G: 16 bit (16Mhz)
Microprocessor
Fujitsu MB90F543G: 16 bit (16Mhz)

Одно и то же Рабочее место для ряда задач
Same Workbench for Numerous Targets

Рабочее место

Любая система MS-Windows

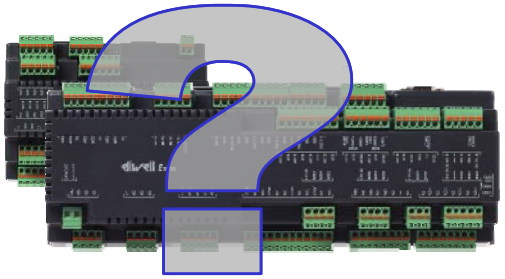
Workbench

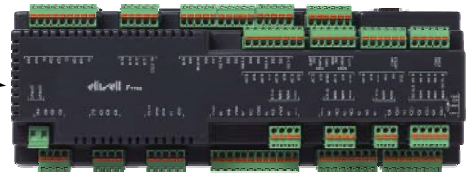
Any MS-Windows system

Код Приложения

Application Code

соединение com. link





Завтра Tomorrow

Сегодня Today

Рабочее место ISaGRAF

ISaGRAF ***Workbench***

- *Общее обозрение*
- *Overview*

Конфигурация разработки

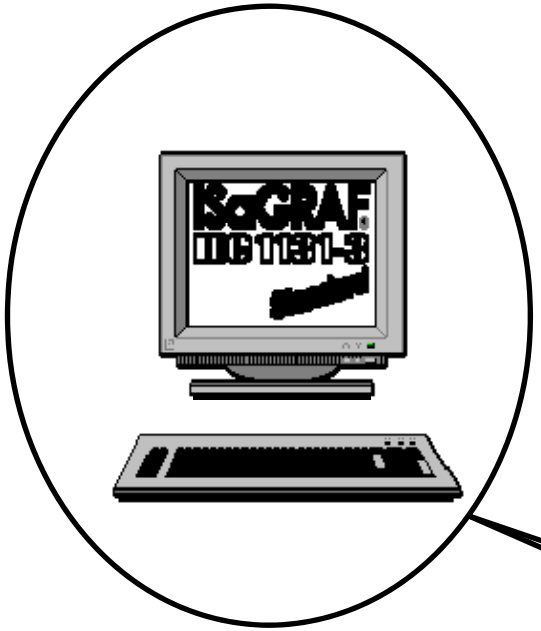
- **Рабочее место**

Development Configuration

- **The Workbench**

Средства автоматического проектирования управления процессами

A CASE tool for Process-Control



Рабочее место ISaGRAF

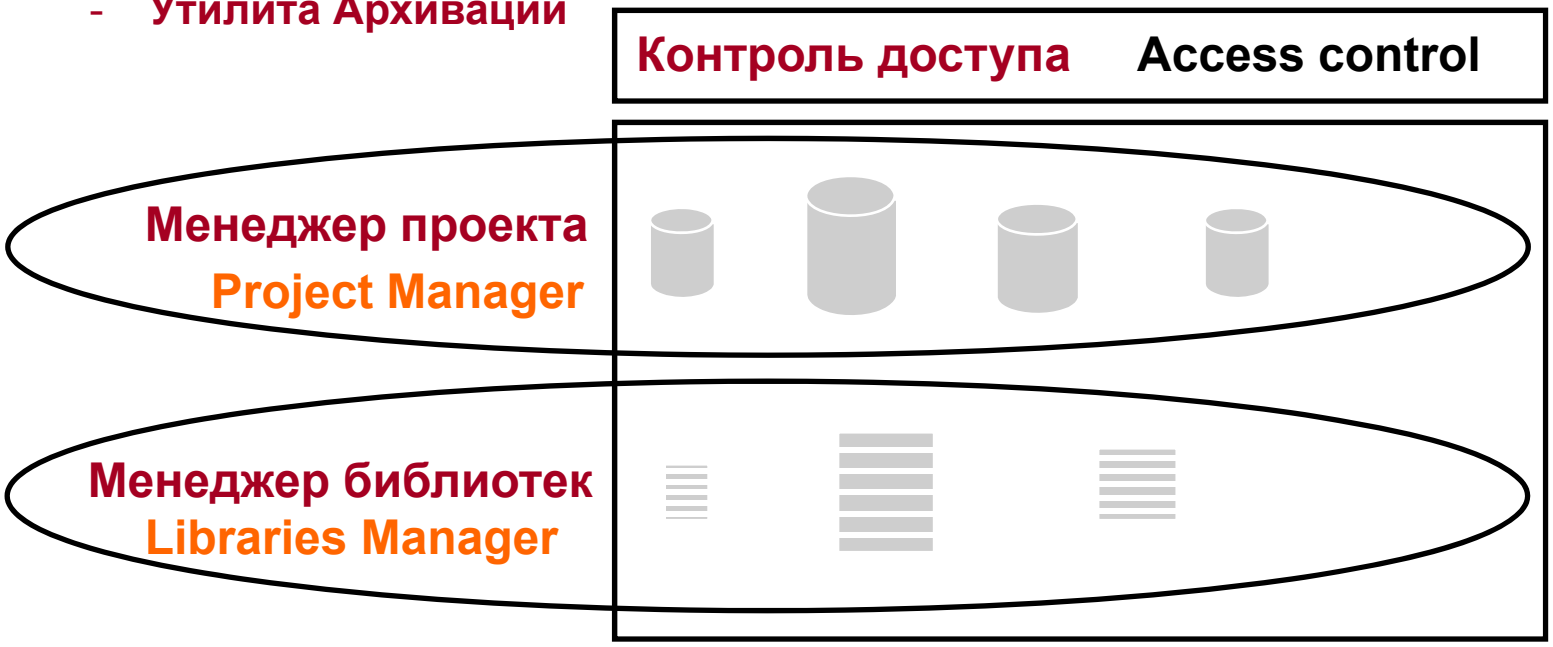
The ISaGRAF **Workbench**

- → **Редактор**
  - → **Editing tools**
- → **Генератор кода**
  - → **Code generation**
- → **Симулятор (имитатор)**
  - → **Simulation**
- → **Отладчик**
  - → **Debug**
- → **Управление проектом**
  - → **Project management**
- → **Перекрестные ссылки**
  - → **Cross reference**
- → **. . . .**
  - → **. . . .**

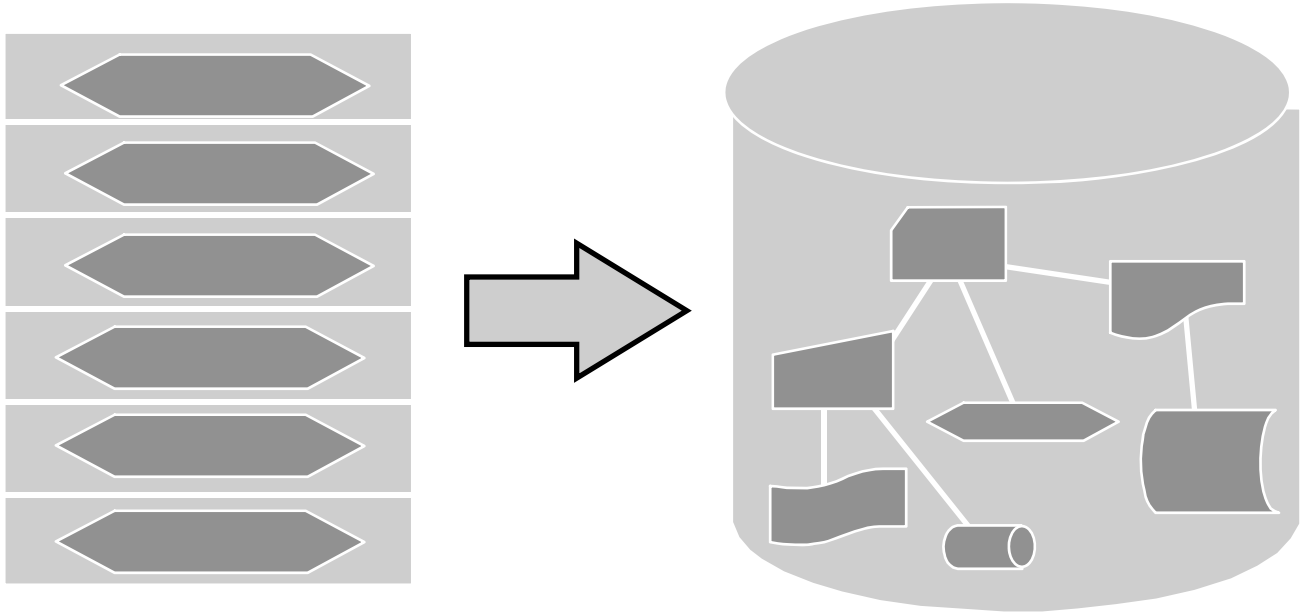
Структура рабочего места
Workbench Architecture

- **Рабочее место ISaGRAF включает**
  - **База данных проекта и библиотеки**
  - **Утилиты, работающие с базой данных**
  - **Система доступа с Паролями**
  - **Утилита Архивации**

- **The ISaGRAF workbench includes**
  - **A data base for projects and libraries**
  - **A set of utilities working on the data base**
  - **An access control system with password management**
  - **Archive Utility**



Проекты и Библиотеки
Projects and Libraries



- Проект = Объекты различных форматов объединенные решением задачи
  - Project = Objects with different format linked together
- Библиотека = Объекты одного формата не связанные с задачей
  - Library = Objects of the same format with no link

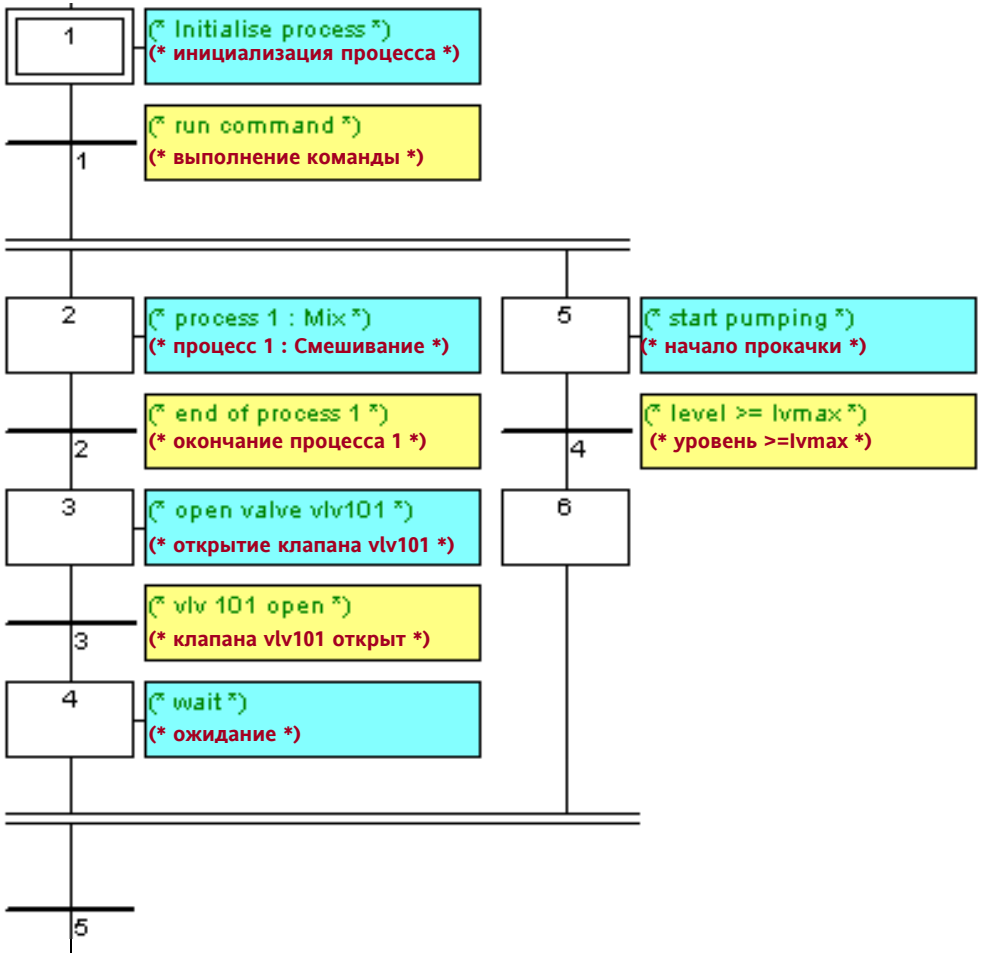
Концепция Библиотеки
The Library Concept

- **Библиотеки позволяют использовать готовые коды:**
- **Libraries allow the use of existing code:**

  - для облегчения использования сложных модулей
    - to ease the use of complex modules
  - сокрытия внутренних процессов модулей
    - hide the internal processing of modules
  - повторного использования кодов для новых задач
    - re-use code for further applications
  - объединять группы разработчиков и программистов
    - link process-control and computer development teams
  - переносить код задачи на PLC прибор
    - link application code with PLC hardware

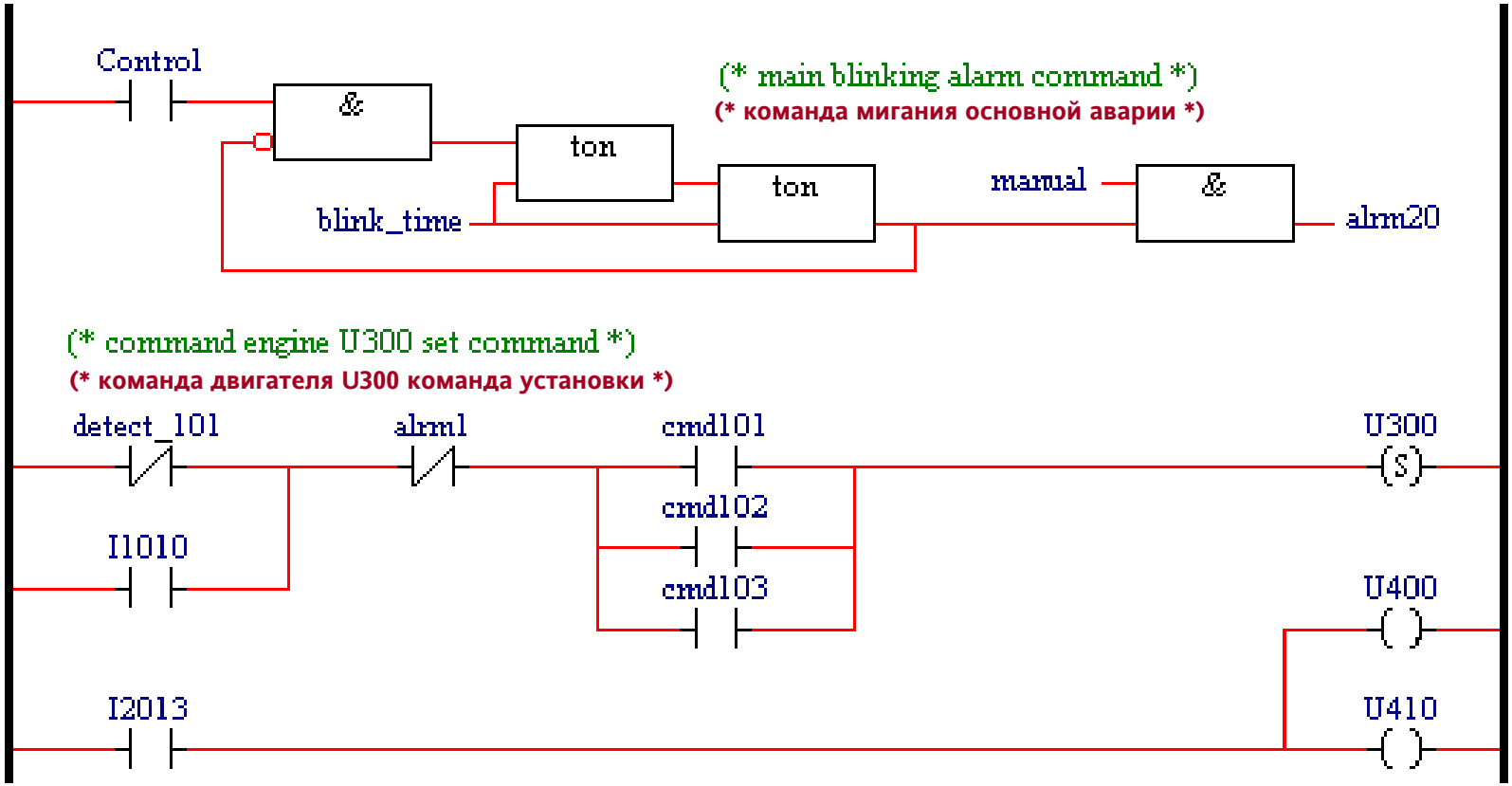
Языки программирования IEC 61131-3
➢ SFC

IEC 61131-3 Programming Languages
➢ SFC



Языки программирования IEC 61131-3
- FBD и LD

IEC 61131-3 Programming Languages
- FBD and LD



Языки программирования IEC 61131-3
- ST и IL

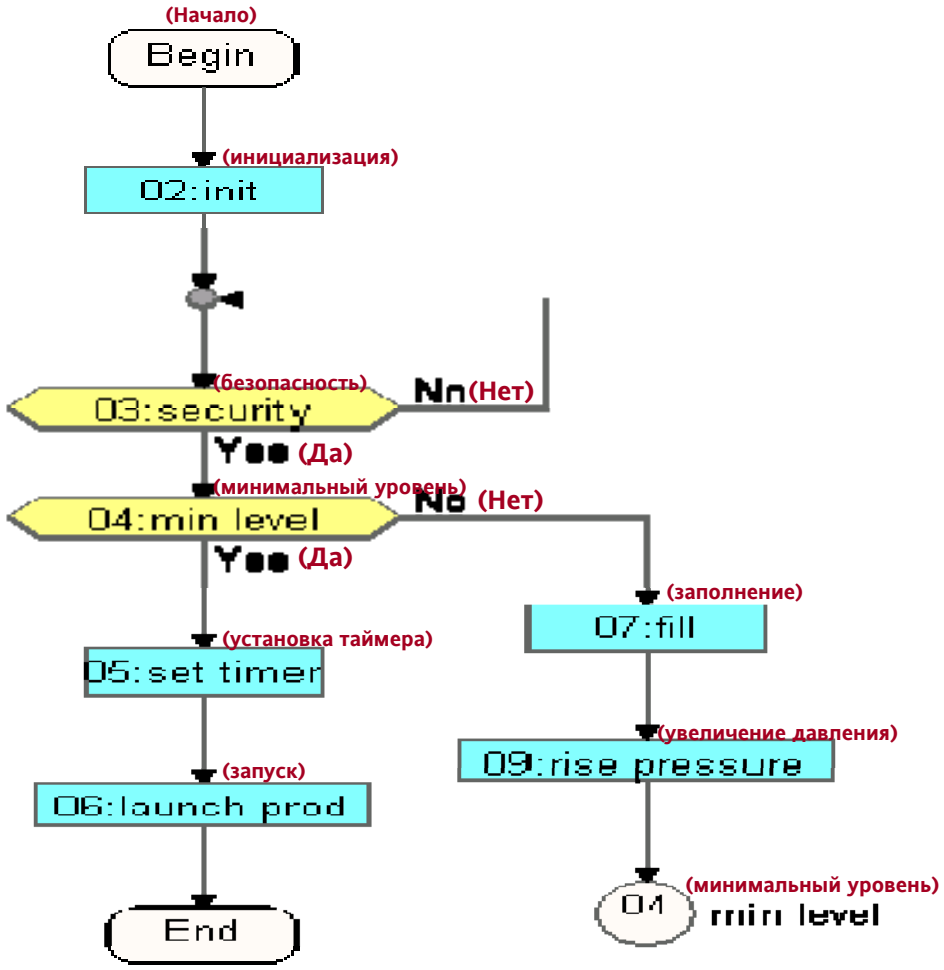
IEC 61131-3 Programming Languages
- ST and IL

```
if (level <= level_max)
then
  out_valve := false;
  memory_vlv := (vlv23+dbh18)/2;
else
  alarm_level := true;
  out_valve := false;
end_if;
```

```
start_cmd: LD   bi101
           ADD  10
mul_ope:   MUL( interm_bcmd
           SUB   bo100
           )
           ST       bcmd
           GT       top_level
           JMPNC mul_ope
```

Programming Languages
- FC

Programming Languages
- FC



Quick View Рабочей станции

Workbench Quick View

- **Шесть языков программирования**
  - Пять языков по IEC 61131-3
  - Язык FC для построения диаграмм решения
- **Менеджер проекта**
  - Для использования редактора
  - Построения проекта
  - Ввода паролей
  - ….
- **Менеджер библиотек**
  - Для общих объектов objects
  - Объектов повторного использования

- **Six programming languages**
  - Five programming languages IEC 61131-3
  - FC language for building decision diagrams
- **Project manager**
  - To access editing tools
  - Build projects
  - Set passwords
  - ….
- **Library manager**
  - For common objects
  - reusable objects

Объект ISaGRAF

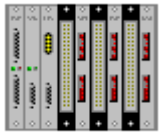
- Правила выполнения
- Воплощение цели
- Выгрузка кода
- Связь Объект- Рабочее место

ISaGRAF ***Target***

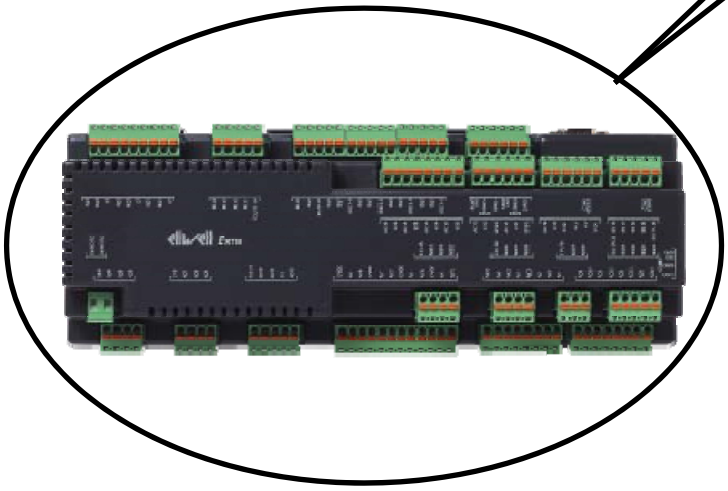
- Execution Rules
- Target Implementation
- Downloaded Code
- Target-workbench links

Конфигурация разработки
Development Configuration

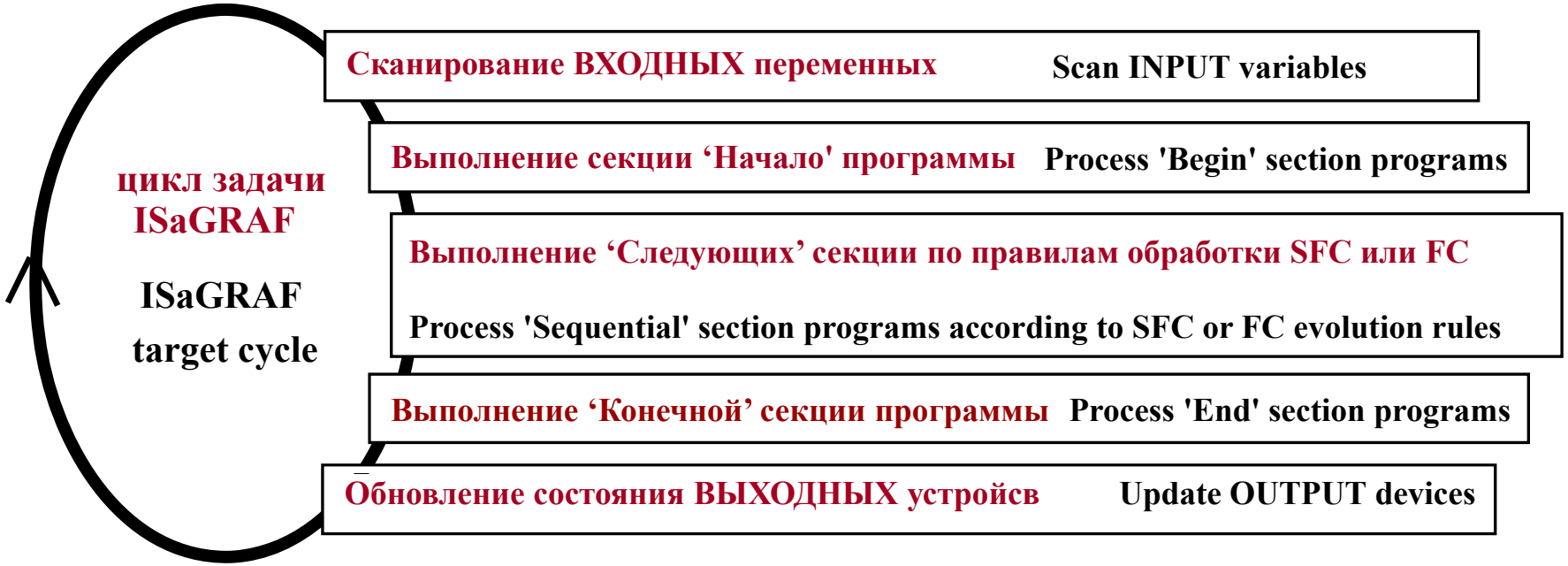
Цель
The Target



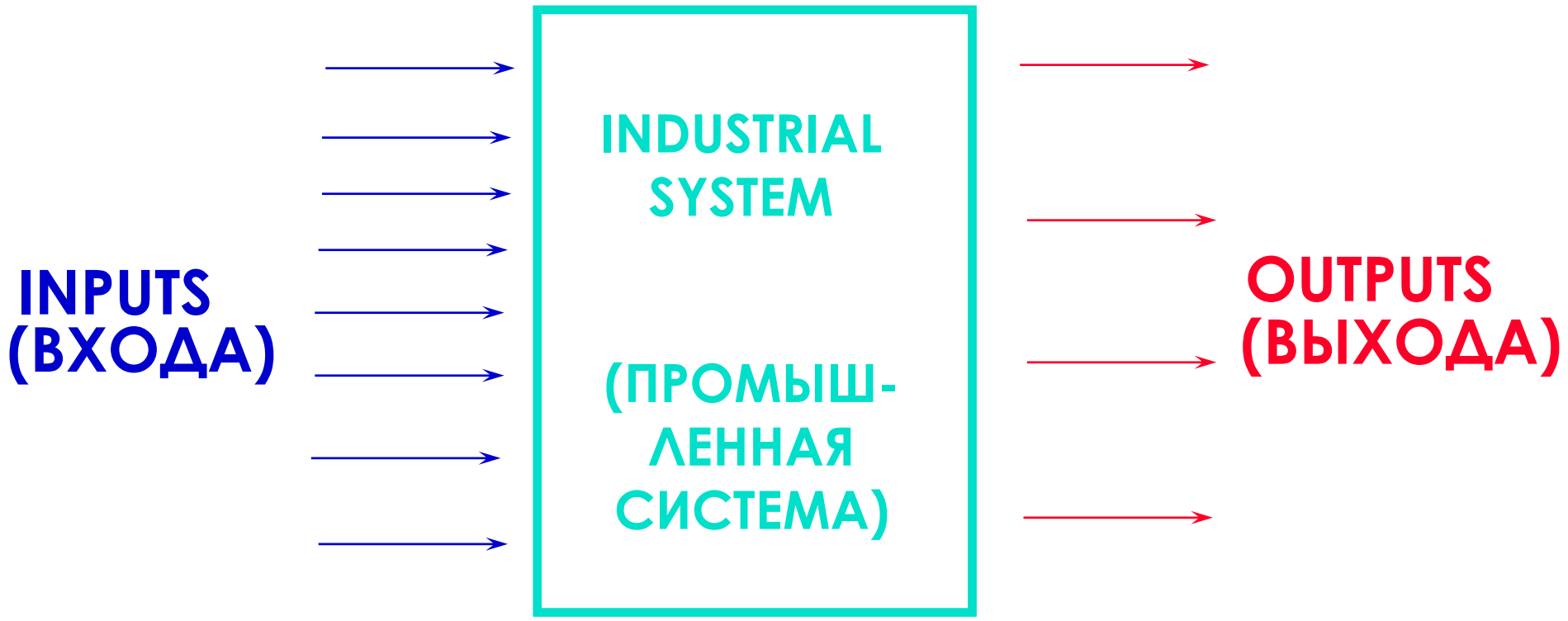
Ядро ISaGRAF
The ISaGRAF kernel
«Программируемый процессор”
Виртуальный PLC
The "software processor”
Virtual PLC



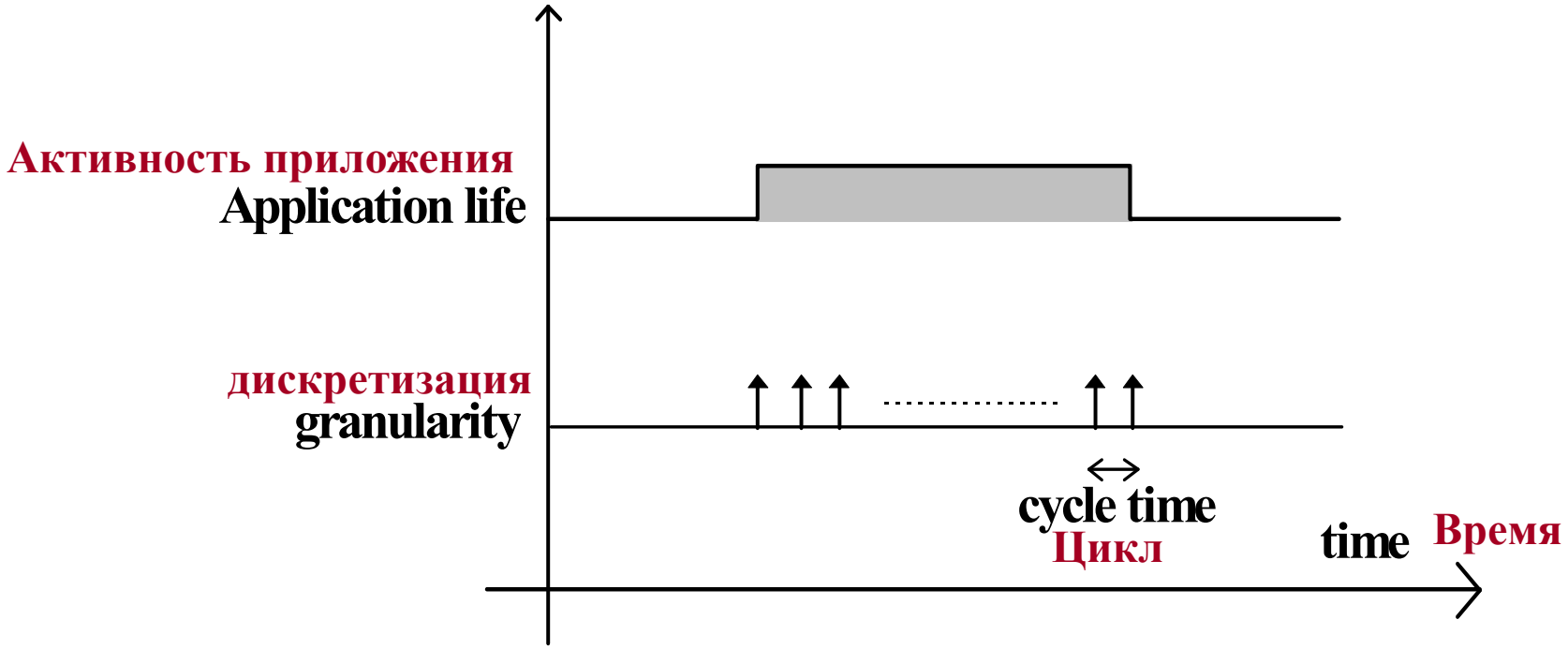
Цикл ISaGRAF
The ISaGRAF Cycle



Реагирующие системы
Reactive Systems



Реализация реакции
Reactive Execution



Выполнение ISaGRAF: Ядро
ISaGRAF Execution : The Kernel

- **Ядро работает в синхронном режиме по отношению к процессу («выполнятся все, что должно выполняться")**
  - **The kernel works in a synchronous mode towards the process ("all what has to be executed is executed")**

- **Длительность каждого цикла зависит от**
  - Сложности действий
  - Количества и размера программ
  - Количества используемых переменных

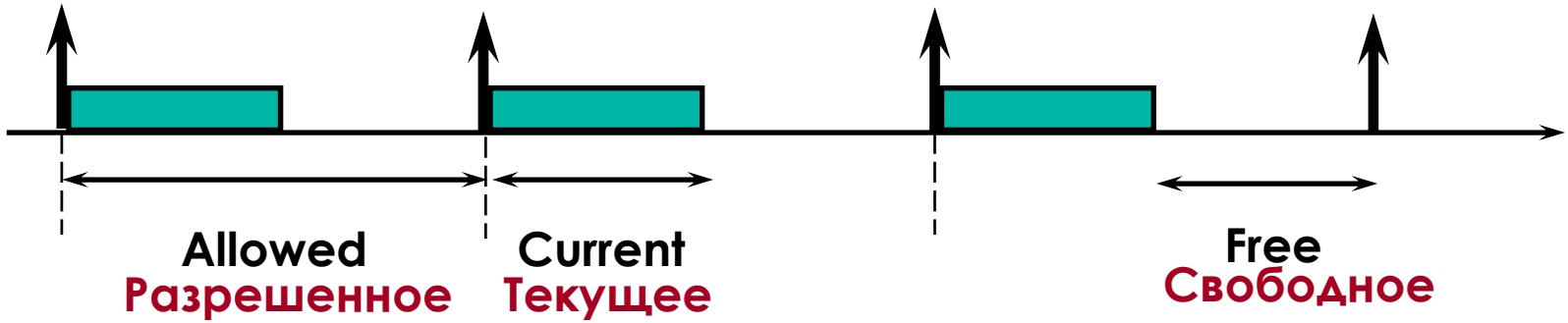
The duration of each cycle depends on

- The complexity of actions
- The number and size of programs
- The number of variables manipulated

Выполнение ISaGRAF : Переменные
*ISaGRAF **Execution : Variables***

- **Выходные переменные обновляются только после выполнения программы**
  - **При выполнении программы ядро ISaGRAF работает с внутренними значениями**
  - **Обновление выходных параметров (приводов) зависит от реализации интегратора:**
    - **Обновляются только заданные переменные ИЛИ**
    - **Обновляются все переменные, касающиеся устройства (которому соответствует выход)**
- **Output variables are only updated after programs execution**
  - **During programs execution, the ISaGRAF kernel works on internal values**
  - **Update of output variables (actuators) depends on integrator's implementation:**
    - **Only assigned variables are updated or**
    - **All variables belonging to a board (for which an output has been assigned) are updated**

Управлением временем Цикла
Cycle Timing Control



- Если Текущее >= Разрешенного => Перекрытие времени цикла
- Если Текущее < Разрешенного => Покой
  - If Current >= Allowed => Cycle time overflow
  - If Current < Allowed => Sleep

Режим выполнения Задачи

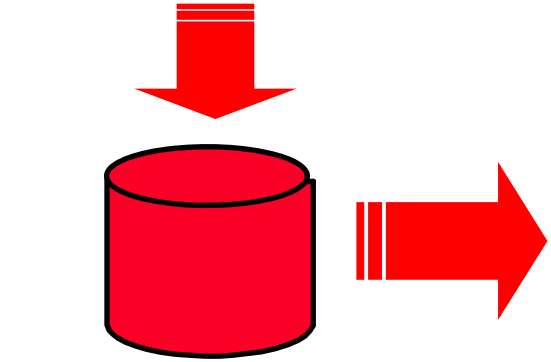
Target Execution Modes

- **Режим реального времени : Циклы выполнения запускаются временными циклами**
- **Режим цикл за циклом : пользователь управляет исполнением из меню отладчика**
  - **В этом режиме, автоматически обрабатываются Входа/Выхода**
  - **Цикл выполняется целиком после нажатия кнопкт "Execute cycle/Выполнить цикл”**
- **Режим можно выбрать перед генерацией кода (опции)**
- **Режим изменяться интерактивно (точки прерывания)**

- **Real-time mode : Execution cycles are triggered by the cycle timing**
- **Cycle to cycle mode : user driven execution from the debugger menu**
  - **In this mode, only I/Os are automatically processed**
  - **The whole cycle is executed when the user enters the "Execute cycle" key**
- **The mode can be prepared before generation (options)**
- **The mode may be changed interactively (breakpoints)**

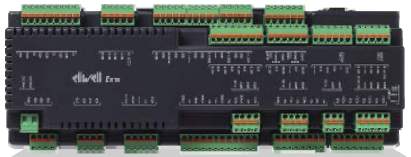
Implementation
Implementation

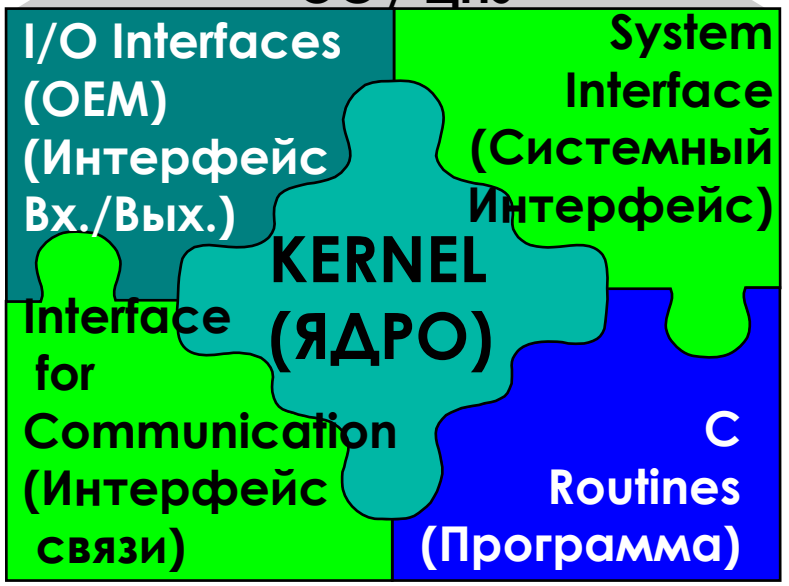
ISaGRAF Workbench



Код Применения
Application Code

Energy XT PRO

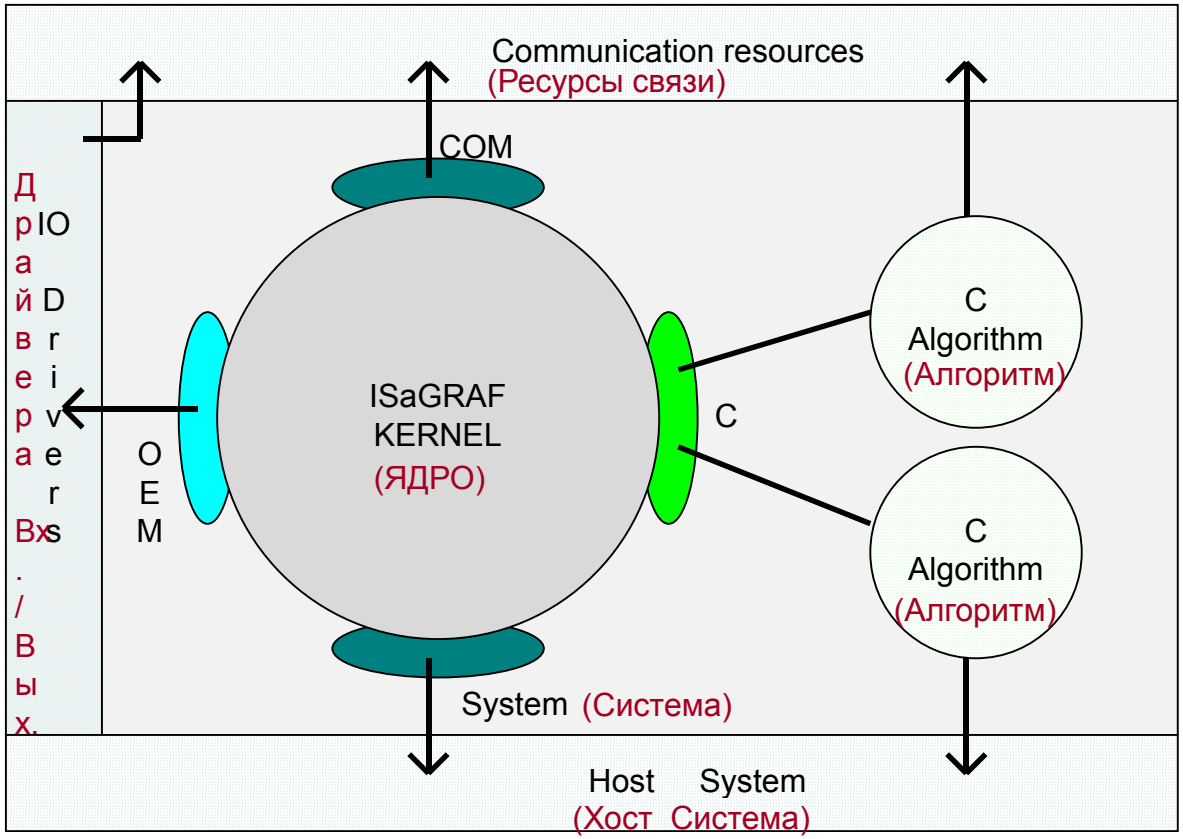




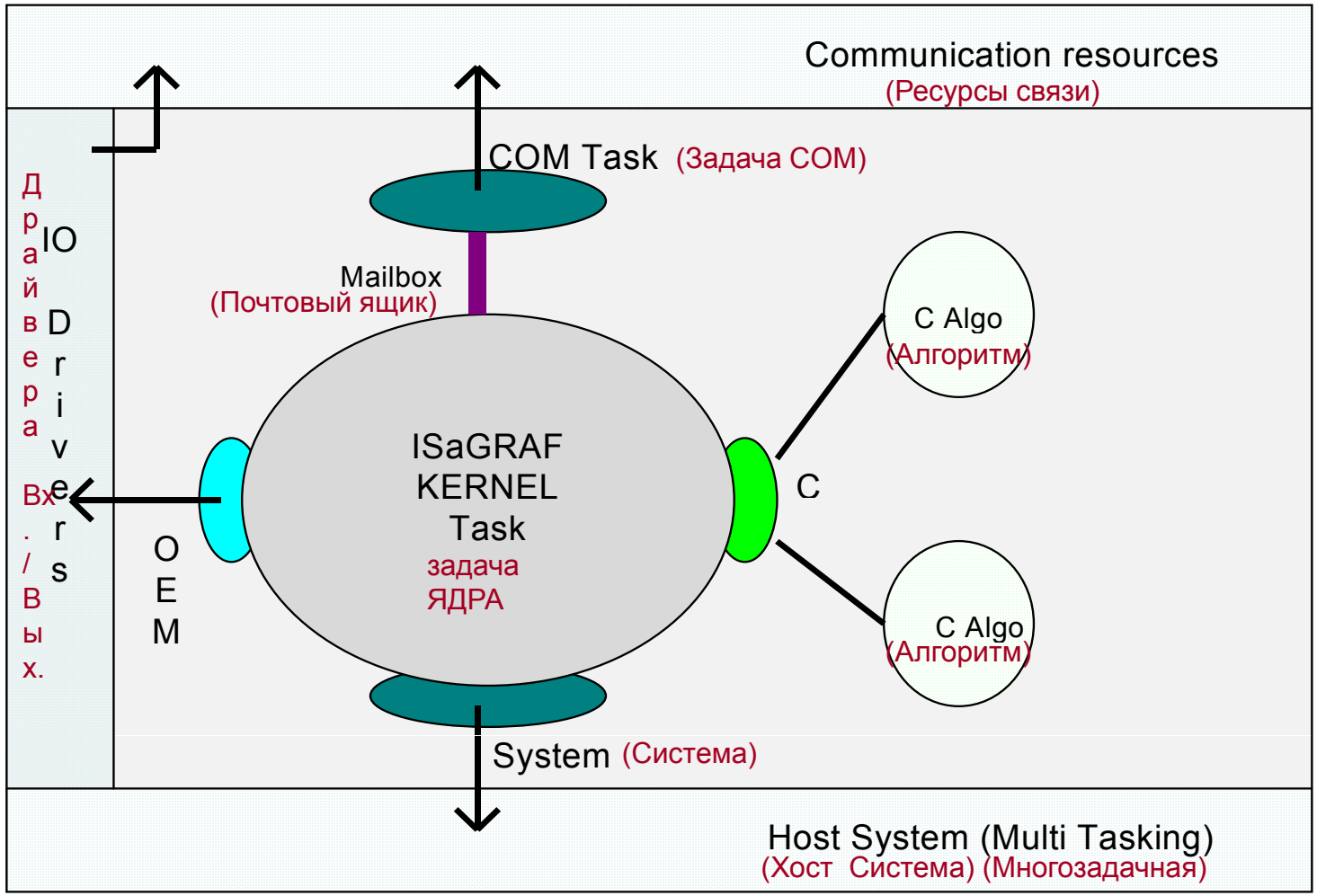
Устройство независимо от загружаемого кода / Стандартный, общий двоичный код
Легко переносимая виртуальная установка / Стандартные ворота открытия расширений

Hardware independent loaded code / Standard, public binary code
Highly portable virtual machine / Standard gates to open extensions

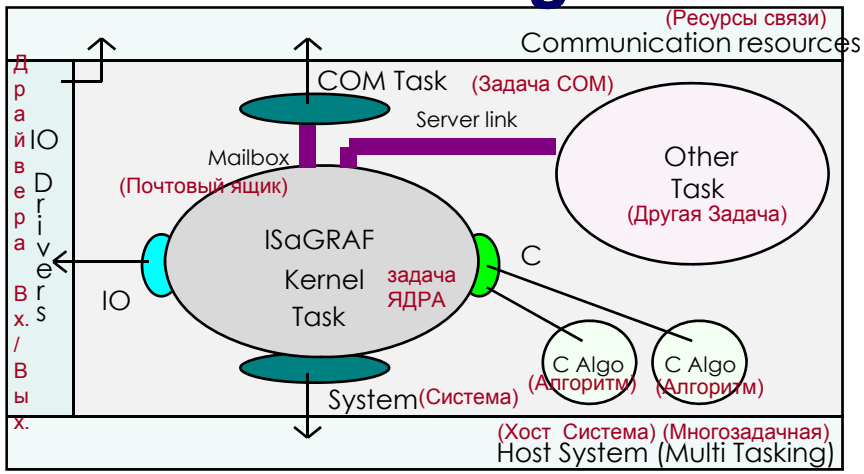
Монозадача ISaGRAF
ISaGRAF Monotask

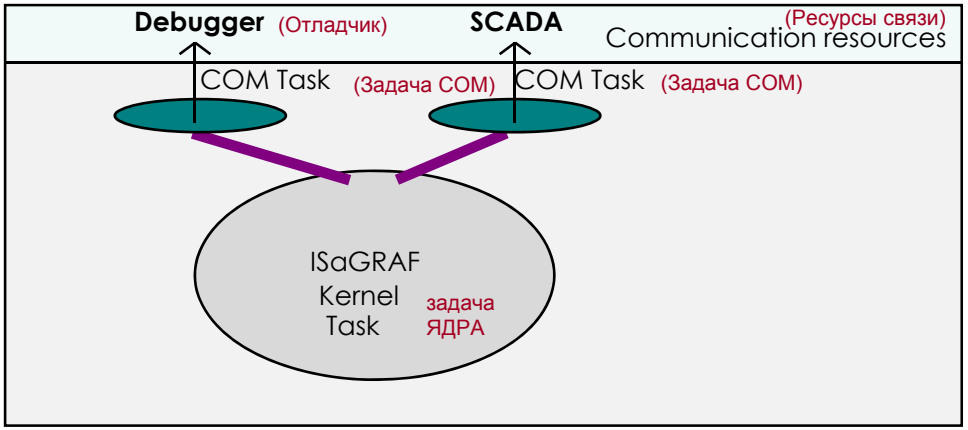


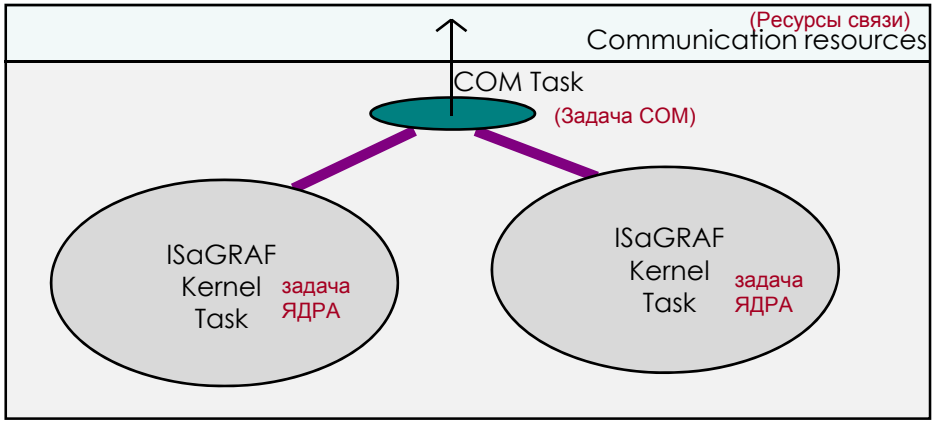
Многозадачность ISaGRAF
ISaGRAF Multitasking



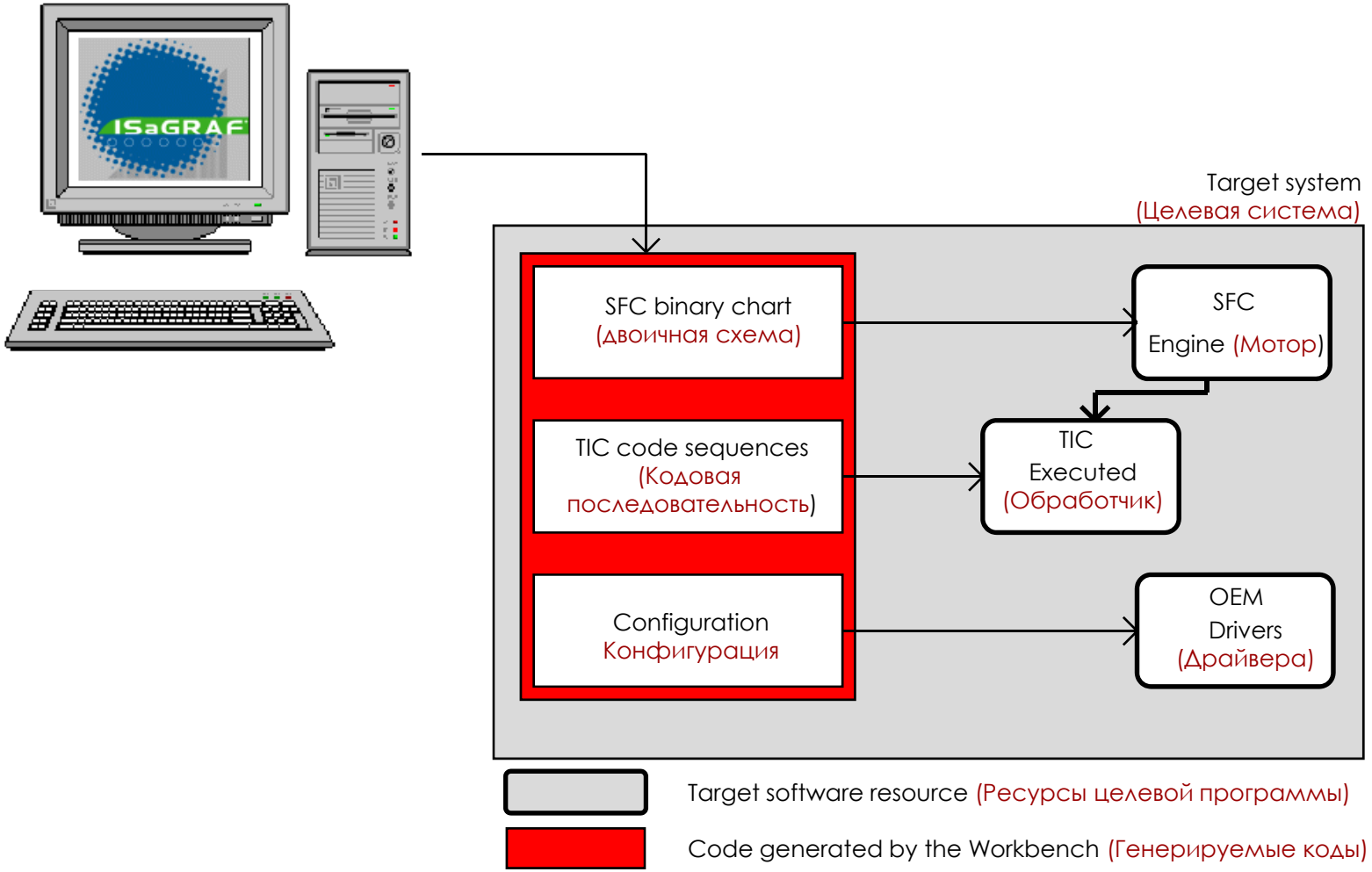
Многозадачность ISaGRAF
ISaGRAF Multitasking







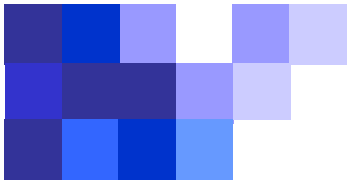
TIC: Независимый код задачи
TIC: Target Independent Code



Опции Генерации Кода
Code Generation Options

- **TIC код**
  - **Для Intel**
- **«C» опция**
  - **Версия 3.04**
  - **Структурированный ‘C’ код**
    - **Применение генерируется в C коде вместо TIC кода**
    - **Решение повышенной безопасности & Улучшенные характеристики Цели**
- **SFC трансляция**
  - **Для целей иных чем ISaGRAF (постпроцессор)**

- **TIC code**
  - **For Intel**
- **«C» option**
  - **Version 3.04**
  - **Structured ‘C’ code**
    - **Application generated in C code instead of TIC**
    - **High security solution & Improves target performances**
- **SFC translation**
  - **For targets other than ISaGRAF (postprocessor)**



Опции Генерации Кода
Code Generator Options

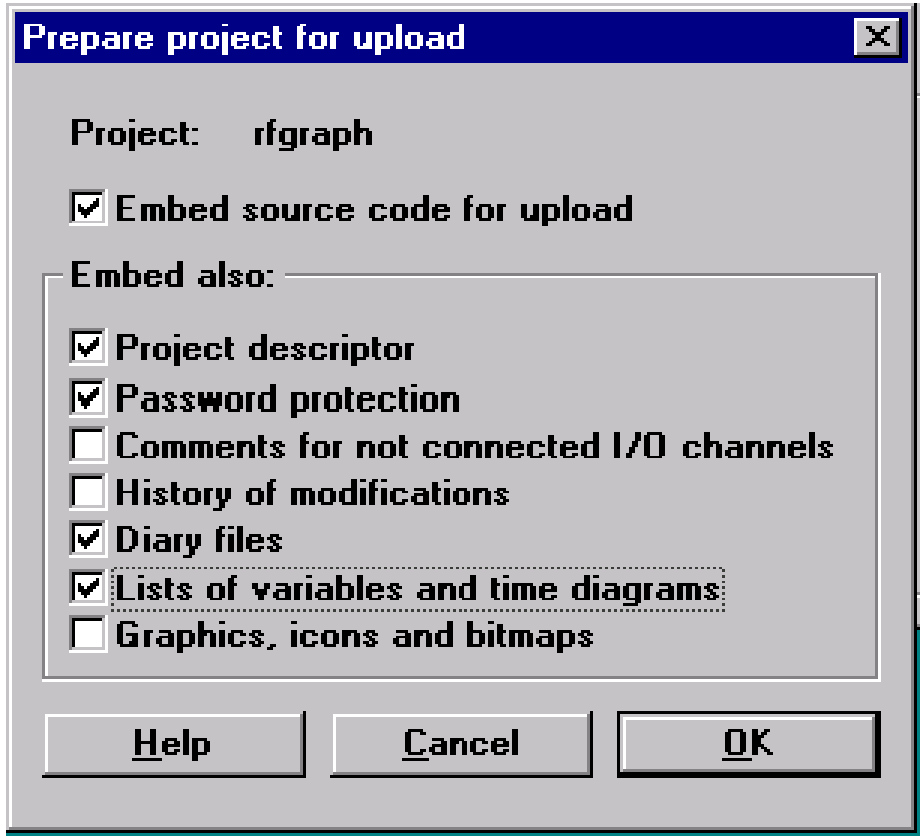
Опции Оптимизатора
Optimizer Options

- Обработка постоянных выражений
- Подавление неиспользуемых меток
- Оптимизация копирования переменных
- Оптимизация выражений
- Подавление неиспользуемых кодов
- Оптимизация арифметических выражений
- Оптимизация логических операций
- Построение Двоичной Диаграммы Решения

- Evaluate constant expressions
- Suppress unused labels
- Optimize variables copying
- Optimize expressions
- Suppress unused code
- Optimize arithmetic expressions
- Optimize boolean operations
- Build Binary Decision Diagrams

Опции Генерации Кода
Code Generator Options

➢ *Функциональность загрузки*
➢ *Upload Functionality*



- **Для выгрузки**
  - Начиная с версии 3.23
  - Только исходный код
  - Без элементов библиотеки
  - Доп. размер +1.5 до 2
- **При загрузке**
  - С менеджера проекта
  - Возрожденный код
  - Используются ВАШИ библиотеки

- **For downloading**
  - From 3.23 version on
  - Only source code
  - No library element
  - Additional size +1.5 to 2
- **When uploading**
  - From project manager
  - Regenerate code
  - Use YOUR libraries

Resources
Resources

- Указанные пользователем данные: список значений, список переменных, двоичные или текстовые файлы
- Загружаются вместе с кодом Применения
- Не выполняются применением ISaGRAF, а другими задачами
  - Конфигурацией сети
  - Настройками приборов
- Редактируются ST как синтаксис

- User defined data: lists of values, list of variables, binary or text files
- Downloaded with the application code
- Not processed by the ISaGRAF application, but by other tasks
  - Network configuration
  - Hardware setting
- Edited with an ST like syntax

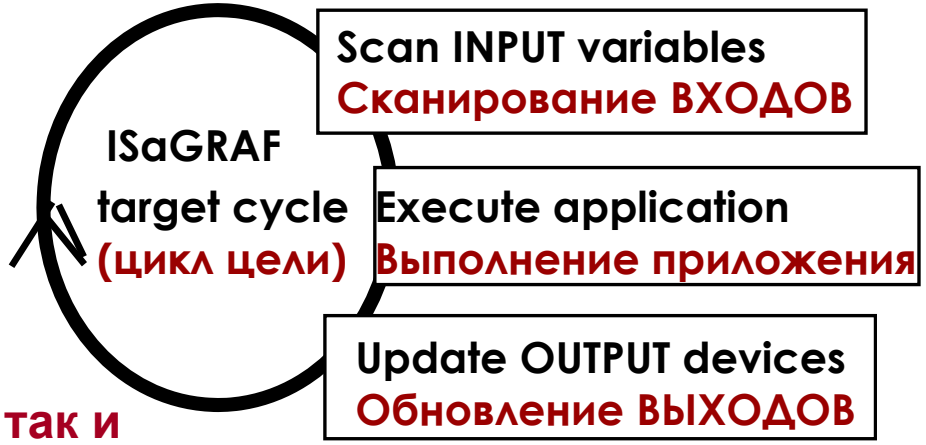
Независимость Оборудования от Рабочей станции
Workbench Hardware Independence

- **Подключение входов/выходов является единственной связью между применением и оборудованием**
- **Опции прогона применения определяют время выполнения цикла**
- **Опции компиляции определяют тип задачи и критерии оптимизации**

- **The I/O connection is the only link between the application and the hardware**
- **Application run-time options define the cycle execution**
- **Compiler options define the type of target and the optimizer criteria**

Цель Quick View
Target Quick View

- Целевая задача – это циклическая петля
- Информация о времени зависит от ОС
- Цикл может быть триггерным
- Генерируемые кода :
  - TIC, C, SFC преобразованные схемы
  - TIC может выгружаться или встраиваться
  - Функциональность загрузки
- Реализация задачи может быть как моно- так и многозадачной
- Несколько целевых задач могут выполняться одним ЦПУ



- Target is a cycle loop
- Time information is OS dependant
- Cycle can be triggered
- Generated code :
  - TIC, C, SFC charts translated
  - TIC can be downloaded or embedded
  - Upload functionality
- Target implementation can be mono tasking or multi-tasking
- Several targets can run on same CPU

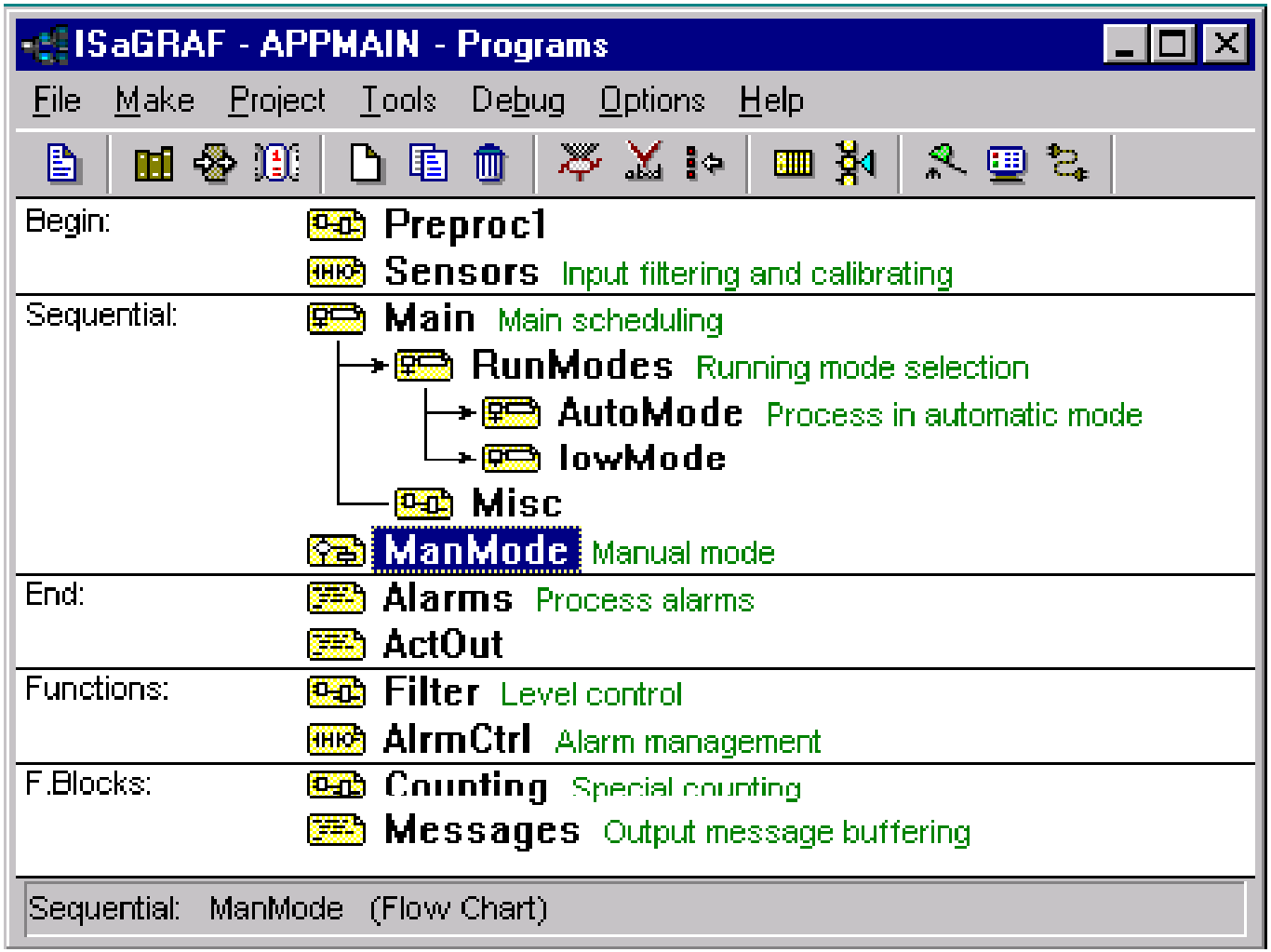
Обслуживание Программ

- *Секция ISaGRAF*
- *Подпрограммы*
- *Функции & Функциональные Блоки*
- *Функциональные Блоки "В цепи"*

Programs Management

- *ISaGRAF Sections*
- *Sub Programs*
- *Functions & Function Blocks*
- *"In Line" Function Blocks*

Секция ISaGRAF
ISaGRAF Sections



Программы
Programs

- Проект ISaGRAF выполнен как набор программ, организованных в секции
- Программы могут соединяться специальными связями
- Циклические программы
  - Обрабатывается цикл каждой цели
- Последовательные программы
  - Программы SFC выполняются в соответствии с правилами и реализацией SFC
  - Секции программы FC обрабатываются на каждом цикле цели

- An ISaGRAF project is made of a set of programs organized in sections
- Programs can be linked together with special relations
- Cyclic programs
  - Are executed at each target cycle
- Sequential programs
  - SFC programs are executed conforming to SFC rules and implementation
  - FC programs sections are executed at each target cycle

Циклические Программы
Cyclic Programs

- **Лист состояний, записанный в FBD, LD, ST или IL**
- **Выполнение в начале цикла (после сканирования входов): «Выражения по умолчанию"**
- **Выполнение в конце цикла (перед обновление выходов): «Выходная комбинация"**
- **Может использовать глобальные переменные совместно с другими программами (циклическими или последовательными)**

- **List of statements written in FBD, LD, ST or IL**
- **Executed in the beginning of the cycle (after input scanning): "Default equations"**
- **Executed at the end of the cycle (before output update): "Output combinations"**
- **Can share global variables with other programs (cyclic or sequential)**

SFC Последовательные программы
SFC Sequential Programs

- Программирование высокого уровня в соответствии со спецификацией
- Диаграмма Состояния
- Может выполняться параллельное программирование
- Может использовать глобальные переменные совместно с другими программами (циклическими или последовательными)
- Поддерживает отношения SFC «от отца к сыну" для параллельных действий
  - High level programming assumable as specifications description
  - State Diagram
  - Can process parallel programming
  - Can share global variables with other programs (cyclic or sequential)
  - Support the SFC "father to child" relation with parallel execution

FC Последовательные программы
FC Sequential Programs

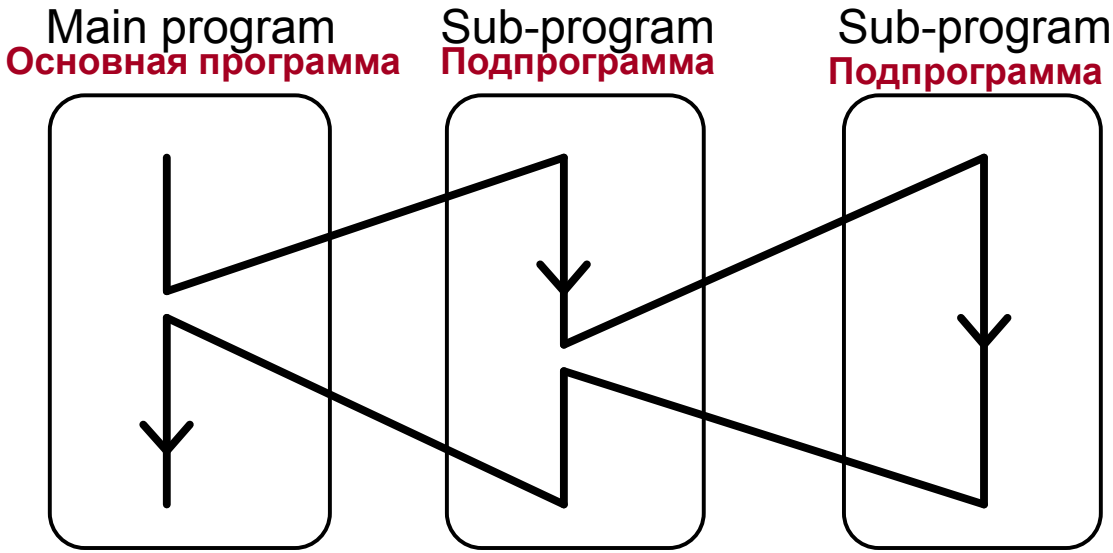
- Язык программирования высокого уровня
- Решение диаграммы протекания
- Позволяет увидеть различные пути процесса
- Может использовать глобальные переменные совместно с другими программами (циклическими или последовательными)
- Поддерживает связь с подпрограммами FC
  - без одновременных действий, вызываемых FC и подпрограммами FC

- High level programming language
- Decision flow diagram
- Allow to view the different process paths
- Can share global variables with other programs (cyclic or sequential)
- Support FC sub programs link
  - No simultaneous action between calling FC and FC sub programs

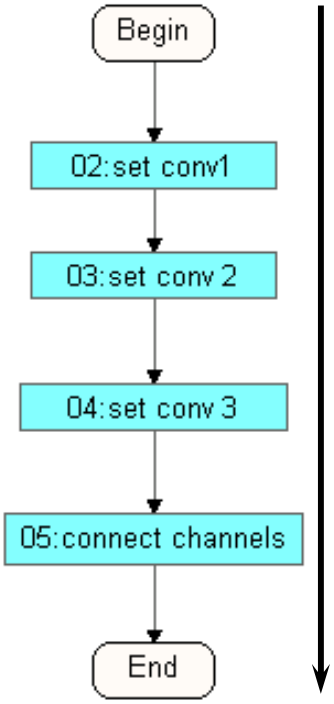
IEC 1131-3 Подпрограммы
IEC 1131-3 Sub-Programs

- Могут вызываться из любой циклической программы
- Могут вызываться любым SFC действием или условием
- Могут вызываться любым FC действием или условием
- Синхронизация выполнения
- 1 возвращаемый параметр, до 31 вызываемого параметра

- Can be called from any cyclic program
- Can be called from any SFC action or condition
- Can be called from any FC action or condition
- Synchronous execution
- 1 return parameter, up to 31 calling parameters



FC Подпрограммы
FC Sub Programs



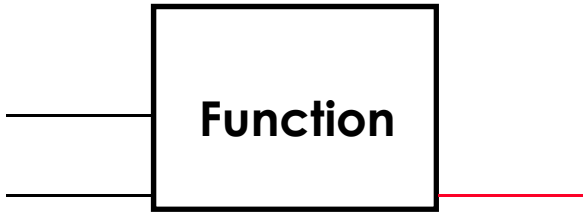
- **При выполнении FC подпрограмм**
  - Отцовская FC программа приостанавливается
  - Один цикл затрачивается на начальный шаг
- **Конечный Блок**
  - Выполнение подпрограммы всегда заканчивается блоком ‘End/Конец’
  - Объект ‘End/Конец’ использует для выполнения один цикл
  - После обработки блока ‘End/Конец’ возобновляется выполнение вызывавшей программы

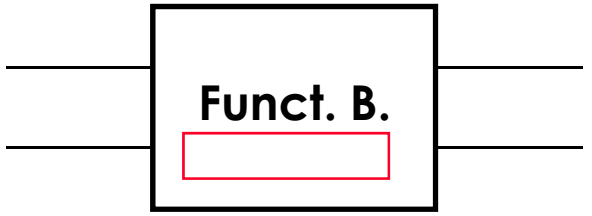
- **During FC sub program execution**
  - Father FC program is suspended
  - One cycle is used to execute the begin step
- **End Block**
  - Sub program flow always finishes connected to ‘End’ object
  - ‘End’ object always takes one cycle to be executed
  - After ‘End’ is executed the calling program resumes execution

Функции & Функциональные блоки
Functions & Function Blocks

- Аналогичны подпрограмме, но относятся к специальной вызывающей программе
- Пишутся на языках IEC 61131-3 I (кроме SFC &FC)
- Не проявляются в структуре выполняемой программы
- Отображаются в последней секции описания проекта
- Операции импорта и экспорта могут выполняться в отношении «Библиотеки Функций и Функциональных блоков”

- Equivalent to a sub-program, but not dedicated to a special caller
- Written in IEC 61131-3 languages (except SFC &FC)
- Do not appear in the executed programs hierarchy
- Appear in the last section of the project description
- Import and export operations can be done towards the "Functions & Function blocks library”





Функциональные блоки “In Line/В строке”

“In Line” Function Blocks

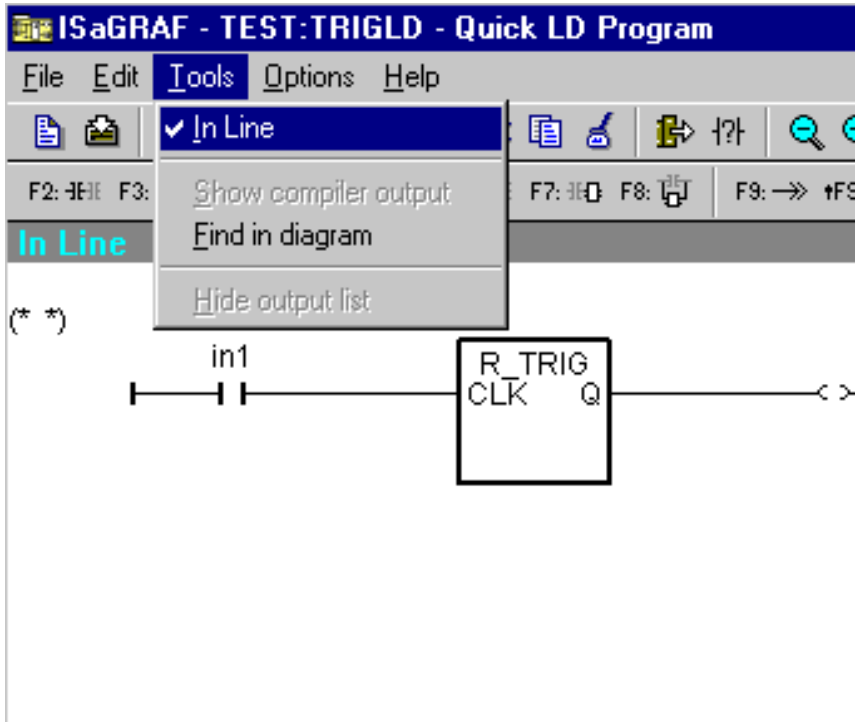
- *Определение*
  - *Definition*
- *Использование Функциональных блоков “In line”*
  - *Use of “In line” Function Blocks*

Определение
Definition

- Коды Функциональных блоков “In Line” повторяются в вызывающей программе при каждом вызове (случае)
- Функциональные блоки “In Line” должны писаться и вызываться на языке Quick LD.
- Функциональные блоки“In Line” могут быть вложенными (FB в FB)
  - The code of an “In Line” function block is repeated in the calling program on each call (each instance)
  - “In Line” function blocks must be written and called in Quick LD language.
  - “In Line” function blocks can be nested. (FB in FB)

Использование
Use

- **В редакторе языка Quick LD командой “Tools / In Line” устанавливается флаг атрибута “In Line”.**
- **In Quick LD editor, use “Tools / In Line” command to toggle “In Line” attribute of the edited block.**



В функциональном блоке “In Line” Вы можете

In an “In Line” FB you can

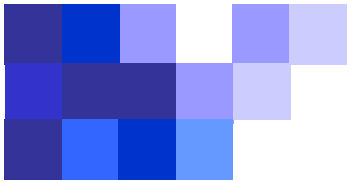
- **Использовать переходы, “возвраты” и метки**
- **Использовать стандартные и “C” функциональные блоки**
- **Использовать “P” и “N” контакты и катушки**
- **Вызывать другие функциональные блоки “In Line”**
  - **Use jumps, “return” and labels**
  - **Use standard and “C” function blocks**
  - **Use “P” and “N” contacts and coils**
  - **Call another In Line function block**

Компиляция Системы

Compiling System

- **Функциональные блоки “In Line” компилируются первыми**
- **Функциональные блоки “In Line” и все программы, их использующие, рекомпилируются каждый раз, когда генерируется код приложения.**

- **“In Line” FBs are compiled first**
- **“In Line” FBs and all programs using them are systematically re-compiled each time you generate the application code.**



Как это работает
How it Works

- Компилятор дублирует коды тела FB при каждом вызове.
- Новые переменные назначаются при каждом вызове параметров или локальных переменных FB.
- Размер применения увеличивается!
- Время выполнения оптимизируется (нет передачи параметров, нет вызова FB/Функц. блоков)

- The compiler duplicates the code of the FB body on each call.
- New variables are allocated on each call for parameters and FB local variables.
- The size of the application is bigger !
- The execution time is optimized (no parameter passing, no FB calling)

Ограничения
Limitations

- **Только для языка Quick LD!**
- **Только в проекте (не в библиотеке)**

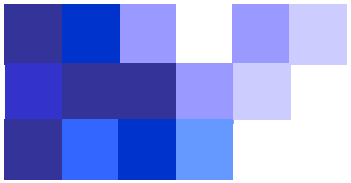
- **For Quick LD language only !**
- **Only in projects (not in library)**

Управление программой Quick View
Program Management Quick View

- **Циклические программы (В секциях ‘begin’ и ‘end’ ’)**
  - **Cyclic programs (in ‘begin’ and ‘end’ sections)**
- **Последовательные программы (в секции ‘seq’, SFC и FC)**
  - **Sequential programs (in ‘seq’ section, SFC and FC)**
- **Локальные функции и функциональные блоки**
  - **Local Functions and Function Blocks**
- **Функциональные блоки Quick-LD “In Line”**
  - **Допускает вложенность Функциональных блоков**
  - **Quick-LD “In Line” Function Block**
    - **Allow Function Block nesting**

Общие Элементы
Common Elements

- *Элементы Словаря*
  - *Dictionary Elements*
- *Элементы Библиотеки*
  - *Library Elements*



Переменные
Variables

- Нет задекларированного языка
- Декларирование выполняется посредством словаря
- Переменные не имеют физических адресов
- Переменные могут иметь сетевые адреса
- Переменные могут размещаться в энергонезависимой памяти (EEPROM)

- There is no declaration language
- Declarations are made through the dictionary
- Variables have no physical address
- Variables may have network addresses
- Variables may be located in memory (EEPROM)

Группы переменных
Variable Groups

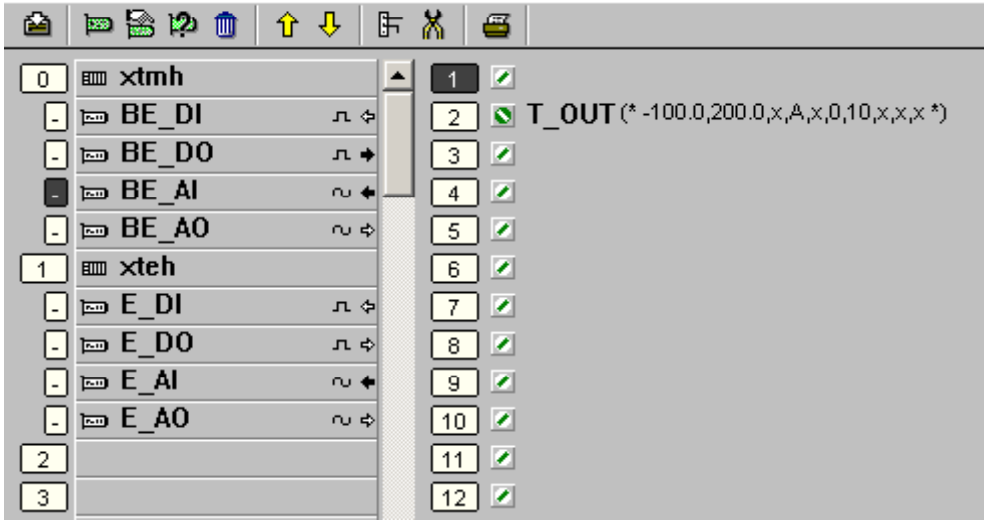
- **Окно редактора для каждой группы переменных**
- **Переменные сортируются**
  - **по типу: Логические, Аналоговые, Таймер, Сообщение, Функц.Блок**
  - **по сферам: Глобальные, Локальные**
- **Переменные могут иметь атрибуты**
  - **Внутренние**
  - **Входные или выходные**
  - **Целые (для Аналоговых, 32 битных)**
  - **Константы**
- **Вх./Вых. всегда имеют Глобальный уровень**

- **There is an editing window for each group of variable**
- **Variables are sorted**
  - **by type: Boolean, Analog, Timer, Message, F.Block**
  - **by scope: Global, Local**
- **Variables may have attributes**
  - **Internal**
  - **Input or Output**
  - **Integer (for Analogs, 32 bit)**
  - **Constant**
- **I/Os always have the global range**

Непосредственно представляемые переменные
Directly Represented Variables

- **Для свободных каналов**
  - %IXs.c Логический вход
  - %IDs.c Аналоговый вход
  - %ISs.c вход Сообщения
  - %QXs.c Логический выход
  - ...

- **To represent free channels**
  - %IXs.c Boolean input
  - %IDs.c Analog input
  - %ISs.c Message input
  - %QXs.c Boolean output
  - ...



Задание переменных
Variables Definition

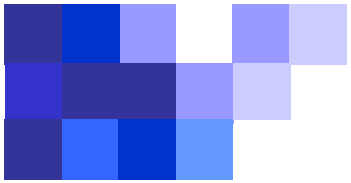
- **Имя**
  - Максимальная длина 132символов
  - Буква, затем буквы, цифры или символы подчеркивания
  - Переменные не чувствительны к регистру
  - Внимание на Зарезервированные ключевые слова ! (глава B.2.3.1)
- **Комментарии**
  - Вольный текст
- **Сетевые адреса**
  - Опциональный диапазон от 193 до 19C7 hex
  - Не символьные (цифровые) идентификаторы используются в основном для сети (Modbus), SCADA
- **Исходное значение**
- **Наименование**
  - Символы, отображаемые перед “:” или первые 16 символов Коментария.

- **Name**
  - Maximum length is 32 characters
  - A letter, then letters, digits or underscore characters
  - Variables are case non-sensitive
  - Beware Reserved keywords ! (chapter B.2.3.1)
- **Comment**
  - Free text
- **Network address**
  - Optional field ranging from 193 to 19C7 hex
  - Non symbolic (numerical) identification mostly used for networking (Modbus), for SCADA communication
- **Initial value**
- **Alias**
  - Characters displayed before “:” or first 16 characters of comment.

Логические переменные
Boolean Variables

- **Постоянные значения**
  - **TRUE или FALSE**
- **значения эквивалентные TRUE/FALSE**
  - **Строки, используемые для замены FALSE и TRUE при отладке**
  - **Должны быть заданы как "defined words/определенные слова" для использования в программе**
  - **Могут импортироваться в Глобальное или Локальное определение**

- **Constant value**
  - **TRUE or FALSE**
- **TRUE/FALSE equivalent values**
  - **Strings used to replace FALSE and TRUE at debug time**
  - **Need to be entered as "defined words" in order to be used in programs**
  - **Can be imported into global or local definitions**



Аналоговые переменные
Analog Variables

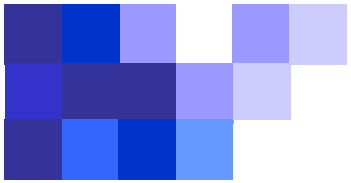
- **Constant value**
  - 32 битные целые значения со знаком: 123, -12, 16#4FCE, 8#1756, 2#010
- **Строка единицы измерения**
  - Вольный текст, отображаемый при отладке
- **Формат**
  - Целое или с десятичной точкой
  - Отображение формата при отладке

- **Constant value**
  - 32 bit signed integer value: 123, -12, 16#4FCE, 8#1756, 2#010
- **Unit string**
  - Free text displayed at debug time
- **Format**
  - Integer format
  - Display format at debug time

Переменные Таймера
Timer Variables

- **Постоянное значение**
  - Базовое значение - 1 миллисекунда
  - t#1h450ms, time#1h3m, t#0s
  - Таймер ограничен значением t#23h59m59s999ms
  - Значение таймера твсегда положительно
  - Не может относиться к устройству Вх/Вых.
- **Механизм**
  - После запуска значение автоматически обновляется
  - Значение увеличивается по системным часам прибора
  - Таймер может запускаться и останавливаться
  - Таймер не может быть обратным счетчиком (но см. библиотеку FB)

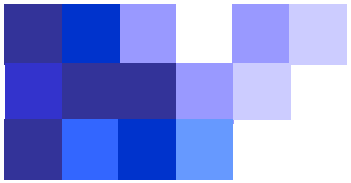
- **Constant value**
  - Basic unit is 1 millisecond
  - t#1h450ms, time#1h3m, t#0s
  - A timer cannot exceed t#23h59m59s999ms
  - A timer value is always positive
  - Cannot be connected to IO board
- **Mechanism**
  - When started, the variable is automatically updated
  - The timer is increased according to the target system clock
  - A timer can be started or stopped
  - A timer cannot be a decreasing counter (but see FB library)



Переменные сообщения
Message Variables

- **Постоянное значение**
  - **'', 'Hello', '$41'**
  - **Любой символ можно сохранить в переменной сообщения**
- **Максимальная длина**
  - **Максимальное число сохраняемых символов не может быть более 255**
- **Текущая длина**
  - **Может быть получена с помощью функции MLEN**
  - **Не может быть больше Максимальной длины**

- **Constant value**
  - **'', 'Hello', '$41'**
  - **Any character can be stored in a message variable**
- **Maximum length**
  - **Maximum number of characters to be stored in it**
  - **Cannot exceed 255 characters**
- **Current length**
  - **Can be retrieved by the MLEN function**
  - **Cannot exceed the maximum length**



Определения
Defined Words

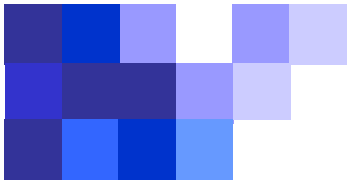
- Идентификаторы для замены любой константы ST
- Нельзя использовать Определения внутри Определения
- Примеры
  - OPEN эквивалентно TRUE
  - PI_100 эквивалентно 3141
  - IS_OK эквивалентно ALARM1 & (TEMP >= 646)
- Сортируется по сфере: ОБЩИЕ, ГЛОБАЛЬНЫЕ или ЛОКАЛЬНЫЕ

- Identifiers used to replace any ST constant expression
- No possibility to use Defined words in Define words
- Examples
  - OPEN is equivalent to TRUE
  - PI_100 is equivalent to 3141
  - IS_OK is equivalent to ALARM1 & (TEMP >= 646)
- Are sorted by range: COMMON, GLOBAL or LOCAL

Элементы Библиотеки
Library Elements

- Они включают закрытые блоки двух типов
- ФУНКЦИИ
  - Имеют только один выходной параметр
  - Могут иметь до 31 параметров вызова
  - Всегда возвращают одно значение, название которого - название функции
- ФУНКЦИОНАЛЬНЫЕ БЛОКИ
  - Могут иметь много выходных параметров
  - Всего до 32 параметровl
  - Каждая копия (экземпляр) блока библиотеки (ссылки) должен иметь уникальное имя

- They consist in black boxes of two kinds
- FUNCTIONS
  - Only have one output parameter
  - May have up to 31 calling parameters
  - Always return one value which name is the one of the function
- FUNCTION BLOCKS
  - May have many output parameters
  - Have 32 parameters maximum in all
  - Each copy (instance) of a library block (reference) must have a unique name



IEC Функции & Функциональные блоки
IEC Functions & Function Blocks

- Пишутся в FBD, LD, ST или IL
- Импортируемы/экспортируемы из/в библиотеку
  - При экспорте регенерируется код библиотеки !
- Могут обрабатывать Глобальные переменные (не рекомендуется)
- Позволяет формировать пакеты данных, только в QLD

- Written in FBD, LD, ST or IL
- Importable/exportable from/to the library
  - When exporting, regenerate code in library !
- Can process global variables (non recommended)
- Enable encapsulation mechanisms in QLD only

"C" Функции и Функциональные блоки
"C" Functions & Function Blocks

- Необходим дополнительный C компилятор
- Стандартные "C" объекты (могут имитироваться)
  - Математика, тригонометрия, управление регистром, строки, массивы
  - Триггера, счетчики, сигналы ...
- Системно-зависимые "C" объекты
  - не могут имитироваться
  - задачи, почтовые ящики, алгоритм ОС, специальный интерфейс / связь
- Использующие ‘C’ объекты продукты могут имитироваться, если подключены к имитатору

- Need an extra C compiler
- Standard "C" objects (can be simulated)
  - Maths, trigonometrics, register control, strings, arrays
  - Triggers, counting, signal ...
- System dependent "C" objects
  - Cannot be simulated
  - tasks, mailboxes, OS routines, special interfacing / communication
- User ‘C’ objects can be simulated if linked to the simulator

Общие элементы Quick View

Common Elements Quick View

- **Словарь**
  - Различные области переменных
  - Различные типы переменных
  - Возможность импорта/экспорта
  - Инструменты графического описания
  - Быстрое описание

- **Библиотека**
  - Независима от Менеджера проекта
  - Включает IEC631-3 или ‘C’ объекты
  - Возможно архивирование

- **Dictionary**
  - Different scopes for variables
  - Different types of variables
  - Export-Import possibilities
  - Graphical declaration tool
  - Quick declaration

- **Library**
  - Independent form project management
  - Includes IEC61131-3 or ‘C’ objects
  - Archive possibility

Языки

- *Структурированный текст (ST)*
- *Перечень инструкций (IL)*
- *Многоступенчатая схема (LD)*
- *Схема функционального блока (FBD)*
- *Схема последовательной функции (SFC)*
- *Схема выполнения (FC)*

Languages

- *Structured text (ST)*
- *Instruction List (IL)*
- *Ladder Diagram (LD)*
- *Function Block Diagram (FBD)*
- *Sequential Function Chart (SFC)*
- *Flow Chart (FC)*

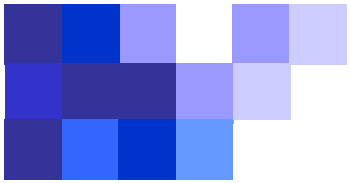
Язык структурированного текста
Structured Text Language

- *ST Синтаксис*
- *ST Редакторr*
- *ST Syntax*
- *ST Editor*

Синтаксис языка ST
ST Language Syntax

- ST предложение включает идентификаторы и разделители
- Идентификатор может быть названием объекта или ключевым словом ST
- Бездействующий разделитель отделяет идентификаторы и другие символы (пробелы, табуляторы, EOL, точка с запятой)
- Предложения разделяются точкой с запятой
- Скобки позволяют исключить введенное выражение
- Комментарии выделяются (* и *) и не могут перемежаться

- ST statements combine identifiers and separators
- An identifier can be an object name or an ST keyword
- An inactive separator separates identifiers and other symbols (space, tabs, eol, semi-colons)
- Statements are separated by semi-colons
- Braces can isolate a typed expression
- Comments are (* and *) and cannot be interleaved



Читаемость ST
ST Readability

- **Выбирайте значимое название переменной**
- **Разделяйте действия различными атрибутами**
- **Не пишите более одного предложения в строке**
- **Вставляйте пробелы**
- **Вставляйте поясняющие комментарии**

- **Choose judicious names for variables**
- **Separate actions with different attributes**
- **Do not write more than one statement on the same line**
- **Insert blank characters**
- **Enter useful comments**

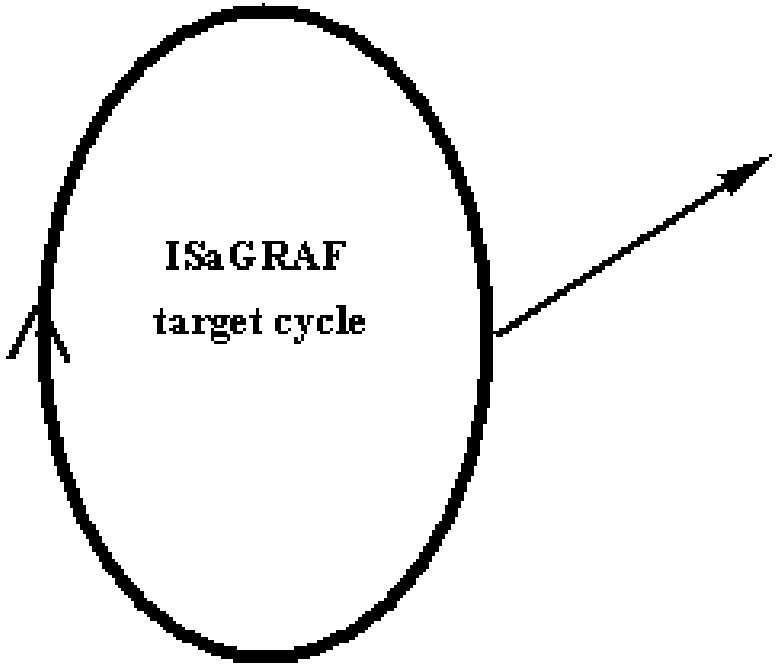
Предложения ST
ST Statements

- := *assignation Присвоение*
- IF / THEN / ELSE / ELSIF / END_IF; *Binary selection Логические операции*
- CASE / OF / ELSE / END_CASE; *Selection Операции выбора*
- WHILE / END_WHILE / REPEAT / END_REPEAT; *Iterations Повтор*
- FOR / TO / BY / DO / END_FOR; *Indexed iterations Индексированный повтор*
- RETURN; *Program termination Прерывание программы*
- EXIT; *Iteration statement termination Прерывание повторения предложения*

+ WHILE и REPEAT для использования в специальных случаях

+ WHILE and REPEAT to be used with special care

Пример : While (Пока) или Repeat (Повторить)
Example : While or Repeat




```
While (input) do
   bla_bla;
end_while;

while (timer) do
   bla_bla;
end_while;
```

Логические операторы ST
ST Boolean Operators

- Логические операторы (понижение приоритета)
  - Boolean operators (decreasing priority)
    - NOT — Логическое отрицание / Boolean negation
    - AND or/или & — Логическое И / Boolean AND
    - OR — Логическое ИЛИ / Boolean OR
    - XOR — Логическое исключающее ИЛИ / Boolean exclusive OR
- Обработка фронта сигнала
  - Edges management
    - redge (,) — Нарастание фронта Rising edge test
    - fedge (,) — Падение фронта Falling edge test

  Обратите внимание на то, что Redge & Fedge нет в IEC61131

  Please note that Redge & Fedge are not in the IEC61131

  => See/Смотри R_TRIG & F_TRIG
- Сравнение : = и <>
  - Comparisons : = and <>

Аналоговые операторы ST

ST Analog Operators

- Арифметические операторы (для целых (ANA))
  - Arithmetic operators (in integer (ANA))
    - \+ Сложение / Addition
    - \- Вычитание / Subtraction
    - * Умножение / Multiplication
    - / Деление / Division
- Порязрядные функции (для целых (ANA))
  - Bitwise functions (in integer (ANA))
    - not_mask ( ) побитное дополнение / bit to bit complement
    - and_mask(,) побитная маска И / AND bit to bit mask
    - or_mask (,) побитная маска ИЛИ / OR bit to bit mask
    - xor_mask (,) побитная маска искл.ИЛИ / XOR bit to bit mask
- Сравнения : =, <>,>=, <=, <, >
  - Comparisons : =, <>,>=, <=, <, >

ST Операторы таймера
ST Timer Operators

- **Арифметические операторы**
  - **Arithmetic operators**

  - **+** — **Сложение / Addition**
  - **-** — **Вычитание / Subtraction**
- **Контроль активности**
  - **Activity control**
  - **TSTART ()** — **Запуск таймера (от его значения) / Start a timer variable (from its value)**
  - **TSTOP ()** — **Остановка таймера (с его значением) / Stop a timer variable (with its value)**
- **Сравнение: >, <**
  - **Comparisons : >, <**

Операторы сообщений ST

ST Message Operators

- **Присвоение**
  - **Assignment**
    - **:=** **Непосредственное копирование / Direct copy**
- **Соединение**
  - **Concatenation**
    - **+** **Соединение двух сообщений / Concatenates two messages**
- **Сравнение : =, <>, >=, <=, >, <** **(лексикографический порядок)**
  - **Comparisons : =, <>, >=, <=, >, <** **(dictionary order)**

Функции преобразования типа ST
ST Type Conversion Functions

- **BOO** Conversion to Boolean

  Преобразование в Логическое
- **ANA** Conversion to integer (ANA)

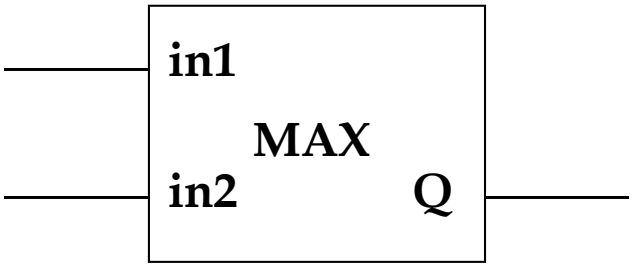
  Преобразование в Целое
- **TMR** Conversion to timer

  Преобразование в Таймер
- **MSG** Conversion to message

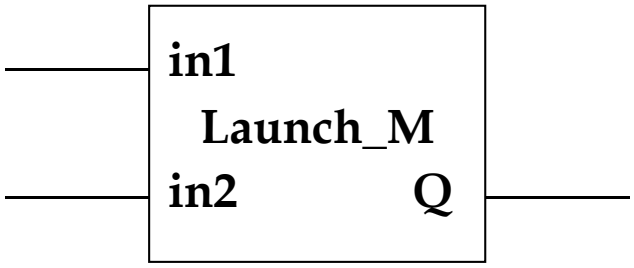
  Преобразование в Сообщение

ST Вызов функций
ST Calling a Function

- MaxVar := Max(ana1, ana2);
- OKStart := launch_M(speed,time_activation);
- Dummy := launch_M(speed,time_activation);



ISaGRAF **function**
Функция ISaGRAF

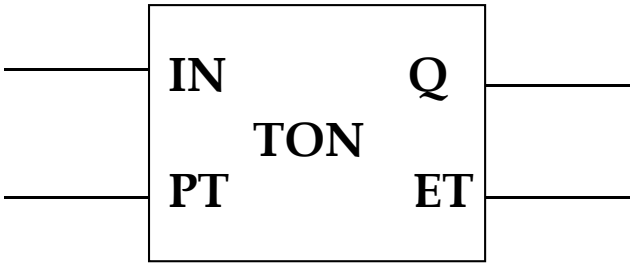


User Function
Пользовательская функция

ST Вызов функций
ST Calling a Function Block

- Заявленная сноска в словаре
  - TON1 сноска блока TON
- Активизировать сноску на функциональный блок
  - TON1 (True, t#3s);
- Выдать возвращаемое значение функционального блока
  - RESULT := TON1.Q;

- Declare instance into dictionary
  - TON1 instance of block TON
- Activate instance of function blocks
  - TON1 (True, t#3s);
- Get back return value of the function block
  - RESULT := TON1.Q;

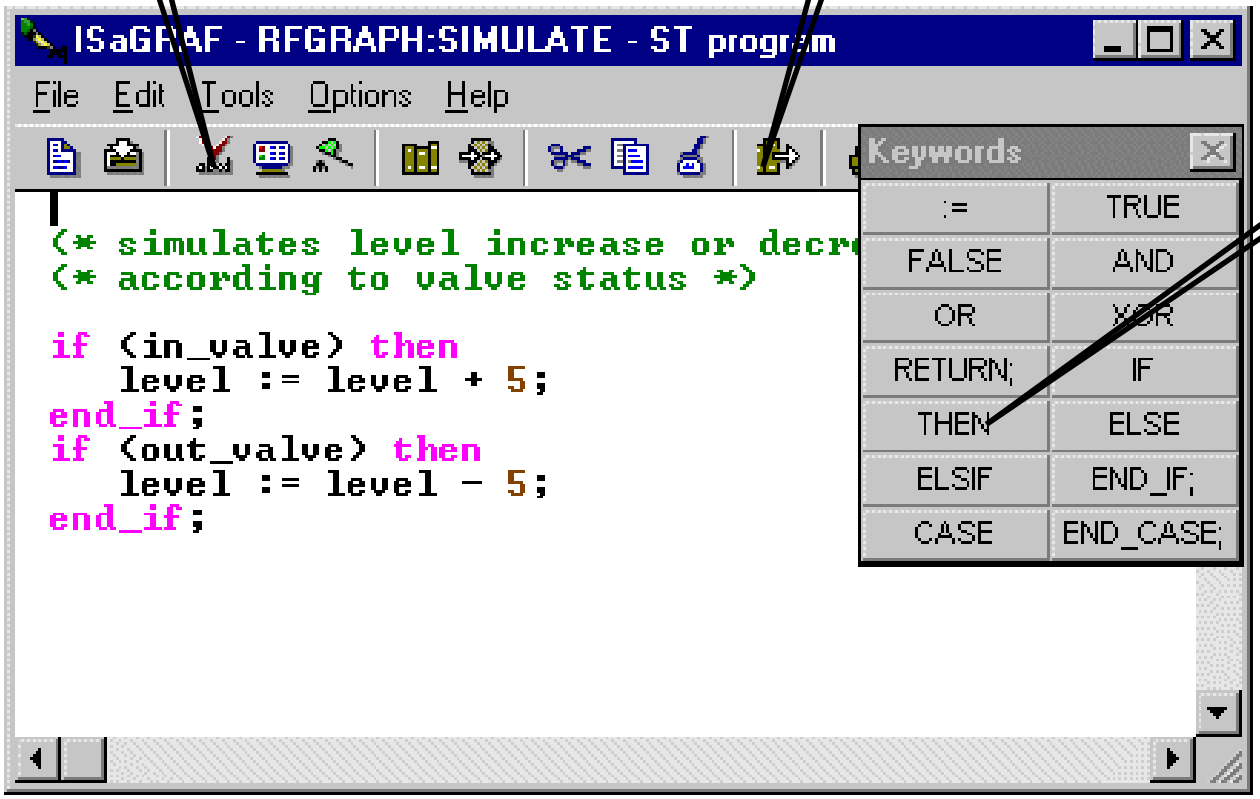


ISaGRAF **F. Bloc**

ISaGRAF ST Редактор
ISaGRAF ST Editor

Syntax Verification
Проверка Синтаксиса

Icon to insert Dictionary variables
Иконка для вставки переменных словаря



Memo of most used key words
Запоминание наиболее используемых ключевых слов

ST Quick View (Быстрый просмотр)
ST Quick View

- Текстовый язык высокого уровня
- Для использования в основной программе
- Для использования в SFC действиях или переходах или в FC действиях и выборах

- Удобен в использовании если
  - Имена переменных смысловые
  - Расставлены комментарии
  - коды правильно размещены

- Textual High level language
- To be used in main programs
- To be used in SFC actions or transitions or FC actions and decisions

- Easy to maintain if
  - Variable names are meaningful
  - Comments are introduced
  - code is well-spaced

Язык Перечня инструкций

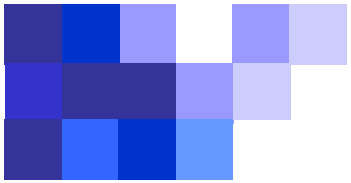
Instruction List Language

- *IL Синтаксис*
- *IL Редактор*
- *IL Syntax*
- *IL Editor*

Формат строки инструкции
Instruction Line Format

- Метка опциональна (без пробела между меткой и двоеточием :)
- Комментарий опционален (последний компонент строки)
- Модификатор операции опционален
- Без пробелов между названием операции и модификатором

- Label is optional (no space between label and :)
- Comment is optional (last line component)
- Operation modifier is optional
- No space between operation name and modifier



ST Эквивалентность
ST Equivalence

- Каждая инструкция выполняется с аккумулируемым значением
- Аккумулируемое значение изменяется каждой инструкцией
- Операнд может быть переменной, константой или меткой имени (какой-либо программы)
- Инструкция IL

- Each instruction works on the accumulator value
- The accumulator is modified by each instruction
- The operand can be a variable, a constant expression or a label name (same program)
- The IL instruction

Операция	Операнд	эквивалентно
Operation	Operand	is equivalent to

Accu := Accu <operation> operand;

IL Инструкции (Указания)
IL Instructions

LD / ST	Присвоение / Assignment
AND / & / OR / XOR S / R	Логические операторы / Boolean operators
ADD / SUB / MUL / DIV	Операторы аналоговые и таймера / Analog or timer operators
GT / GE / EQ / LE / LT	Сравнение / Comparisons
JMP / RET	Операторы Выполнения / Execution operators
CAL <FB_instance_name>	Вызов Функциональных блоков / Function block calls
<Function_name>	Вызов Функций / Function calls

IL Операторы модификатора
IL Operator Modifiers

- **N** — **Логическое отрицание Операнда / Boolean negation of the operand**

ANDN IX12 **в ST означает / means in ST**

Accu := Accu AND NOT (IX12);

- **(** — **Инструкция задержки / Delayed instruction**
- **C** — **Инструкция условия / Conditional instruction**

JMPC LABEL1 означает / means

If Accu then JMP LABEL1

Перечень модификаторов
Modifiers List

- **N** — LD, ST, AND, &, OR, XOR, JMP, RET, CAL
- **(** — AND, &, OR, XOR, ADD, SUB, MUL, DIV, GT, GE, EQ, LE, LT,NE
- **C** — JMP, RET, CAL

Инструкции Задержки
Delayed Instructions

- Используются для отсрочки выполнения инструкции
- (Модификатор указывает, что инструкцию необходимо отложить
-) Оператор обработки отложенной инструкции

```
AND(  IX12
OR    IX35
)
```

выполняется как:

```
Accu := Accu AND (IX12 OR IX35);
```

- Used to postpone the execution of an instruction
- (Modifier Indicates that the instruction has to be postponed
-) Operator Executes the delayed instruction

```
AND(  IX12
OR    IX35
)
```

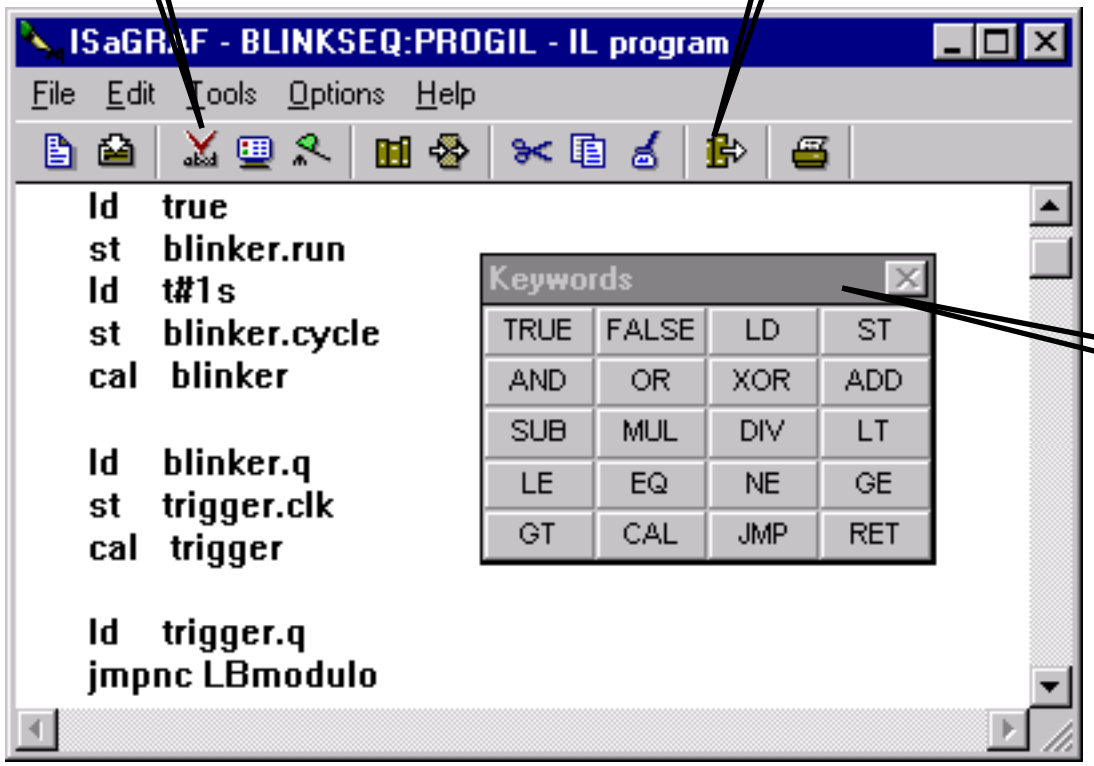
is executed as:

```
Accu := Accu AND (IX12 OR IX35);
```

ISaGRAF IL Редактор
ISaGRAF IL Editor

Syntax Verification
Проверка Синтаксиса

Icon to insert Dictionary variables
Иконка для вставки переменных словаря



Memo of most used key words
Запоминание наиболее используемых ключевых слов

IL Quick View (Быстрый просмотр)
IL Quick View

- **Язык низкого уровня**
- **Точка входа в** ISaGRAF

- **Ассоциируется со скрытым аккумулятором**
- **Заданный перечень инструкций**

- **Обычный Текстовый редактор**

- **Low level language**
- **Entry point in** ISaGRAF

- **Associated to a hidden accumulator**
- **Given list of instructions**

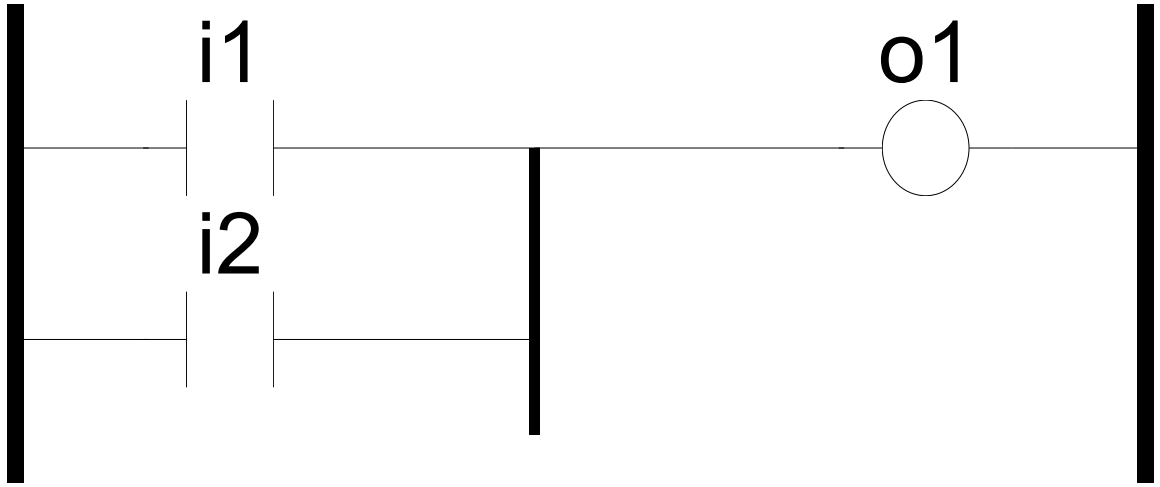
- **Normal text Editor**

Язык Загрузчика Диаграмм
Ladder Diagram Language

- *Правила Языка LD*
- *Редактор Quick LD*
- *Редактор FBD/LD*

- *LD Language Rules*
- *Quick LD Editor*
- *FBD/LD Editor*

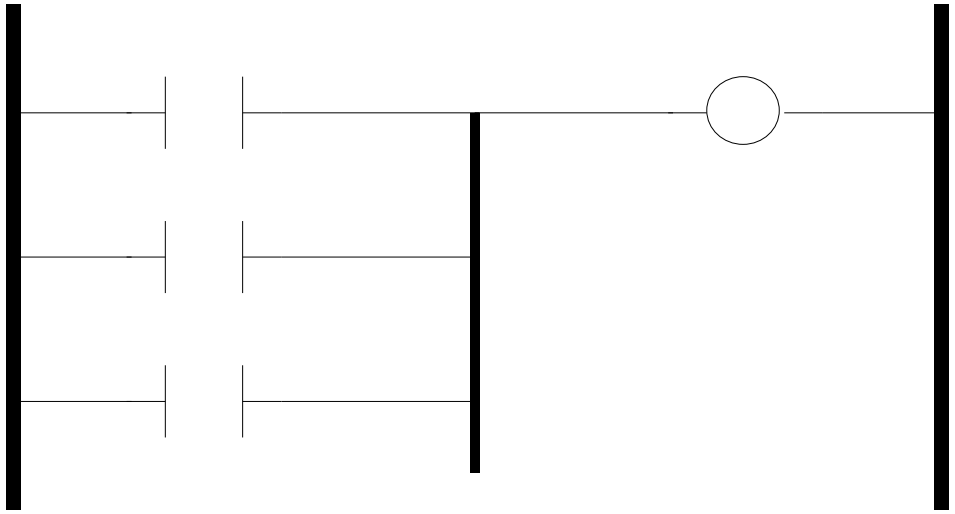
LD Ступени и Каналы
LD Rungs and Rails



- LD представляет Логический сигнал следующим слева направо
- Состояние связи может изменяться контактом
- Переменные могут присваиваться через реле (или катушку)

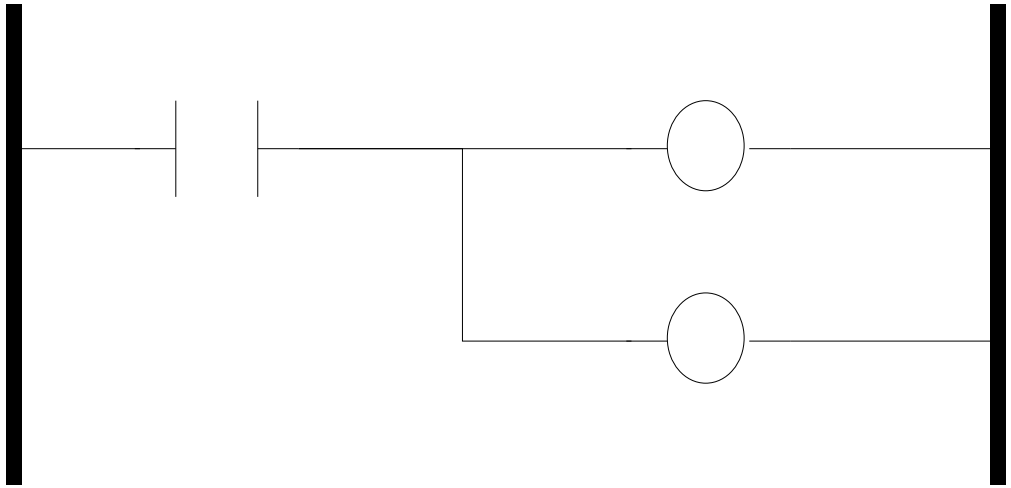
- LD represents a Boolean signal flowing from the left to the right
- The state of a link can be modified by a contact
- A variable can be assigned through a relay (or coil)

Множественное соединение слева
Multiple Connection on the Left



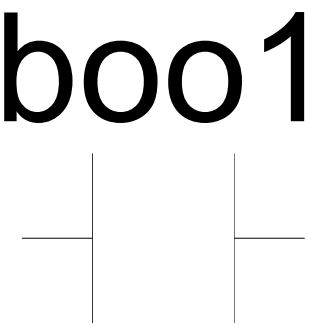
- **Значение соединения справа является результатом Логического ИЛИ трех левых краев.**
- **The value of the connection on the right is the result of the Boolean OR between the three edges on the left.**

Множественное соединение справа
Multiple Connection on the Right



- **Значение соединения слева распределяется на каждое из правых окончаний.**
  - **The value of the connection on the left is propagated into each of the edges on the right.**

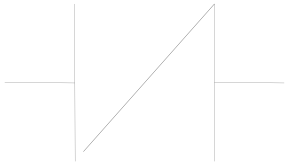
Прямой контакт
Direct Contacts



- **Значение справа является результатом Логического И левого значения и BOO1.**
  - **Right edge value is the result of the Boolean AND between the connection on the left and BOO1.**

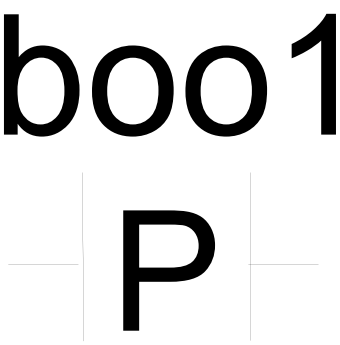
Инверсный Контакт
Inverted Contacts

boo1



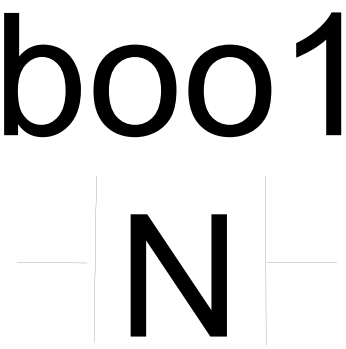
- **Значение справа является результатом Логического И левого значения и отрицания BOO1.**
  - **Right edge value is the result of the Boolean AND between the connection on the left and the negation of BOO1.**

Контакты «Положительного края»
Positive Edge Contacts



- **Значение справа является результатом Логического И левого значения и «теста возрастающего края» BOO1.**
  - **Right edge value is the result of the Boolean AND between the connection on the left and the rising edge test of BOO1.**

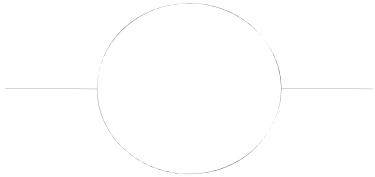
Контакты «Отрицательного края»
Negative Edge Contacts



- **Значение справа является результатом Логического И левого значения и «теста убывающего края» BOO1.**
  - **Right edge value is the result of the Boolean AND between the connection on the left and the falling edge test of BOO1.**

Прямая Катушка
Direct Coils

Out1

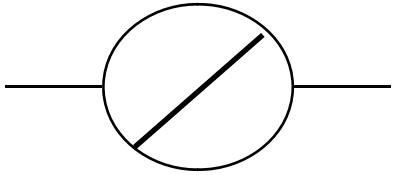


- **Выходу 1 (Out1) присваивается левое значение**
- **Левое значение передается на правый край**

  - **Out1 is assigned with the left connection value**
  - **The value of the connection on the left is propagated on the right edge**

Инверсная Катушка
Inverted Coils

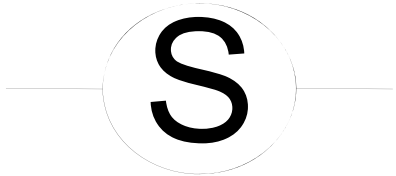
Out1



- **Выходу 1 (Out1) присваивается значение обратное левому значению**
- **Левое значение передается на правый край**
  - **Out1 is assigned with the negation of the left connection value**
  - **The value of the connection on the left is propagated on the right edge**

ЗАДАНИЕ Катушки

SET Coils

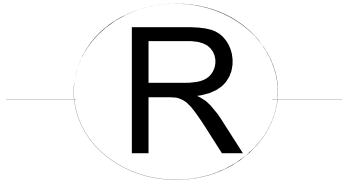


- Значение выхода 1 (Out1) устанавливается в TRUE если левое значение TRUE
- Левое значение передается на правый край

- Out1 is set to true if the left connection value is TRUE
- The value of the connection on the left is propagated on the right edge

СБРОС Катушки

RESET Coils

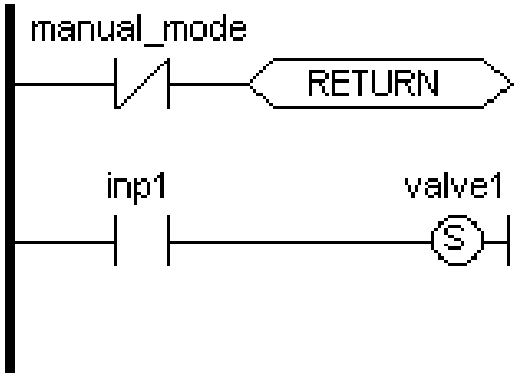


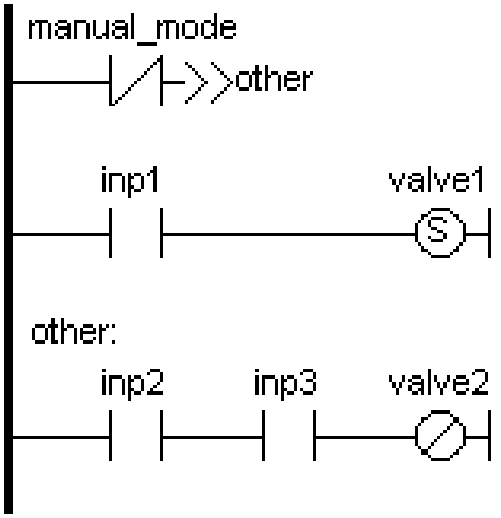
- Значение выхода 1 (Out1) устанавливается в FALSE если левое значение TRUE
- Левое значение передается на правый край

- Out1 is set to false if the left connection value is TRUE
- The value of the connection on the left is propagated on the right edge

Переходы и Метки
Jumps and Labels

- **Для контроля исполнения программы**
  - **To control the program execution**



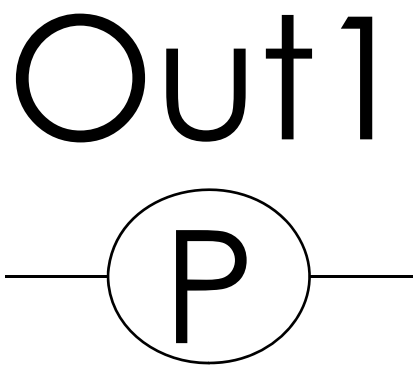


Катушка с определением «Нарастания фронта»

(только Редактор Quick Ld)

Coil with Rising Edge Detection

(Quick Ld Editor only)



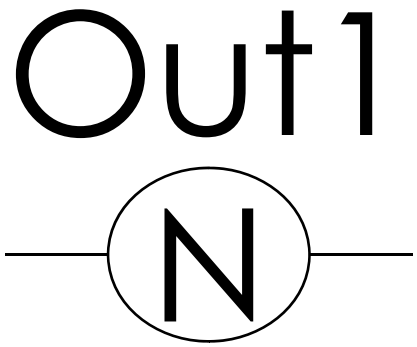
- **Out1 устанавливается в TRUE при увеличении левого значения с FALSE на TRUE (Положительный фронт).**
- **В других случаях Выход сбрасывается в FALSE**
- **Левое значение передается на правый край**
  - **Out1 is set to true when the Boolean state of the left connection rises from FALSE to TRUE.**
  - **The output variable resets to FALSE in all other cases**
  - **The value of the connection on the left is propagated on the right edge**

Катушка с определением «Нарастания фронта»

(только Редактор Quick Ld)

Coil with Falling Edge Detection

(Quick Ld Editor only)



- **Out1 устанавливается в TRUE при уменьшении левого значения с TRUE до FALSE (Отрицательный фронт).**
- **В других случаях Выход сбрасывается в FALSE**
- **Левое значение передается на правый край**

- **Out1 is set to true when the Boolean state of the left connection falls from TRUE to FALSE.**
- **The output variable resets to FALSE in all other cases**
- **The value of the connection on the left is propagated on the right edge**

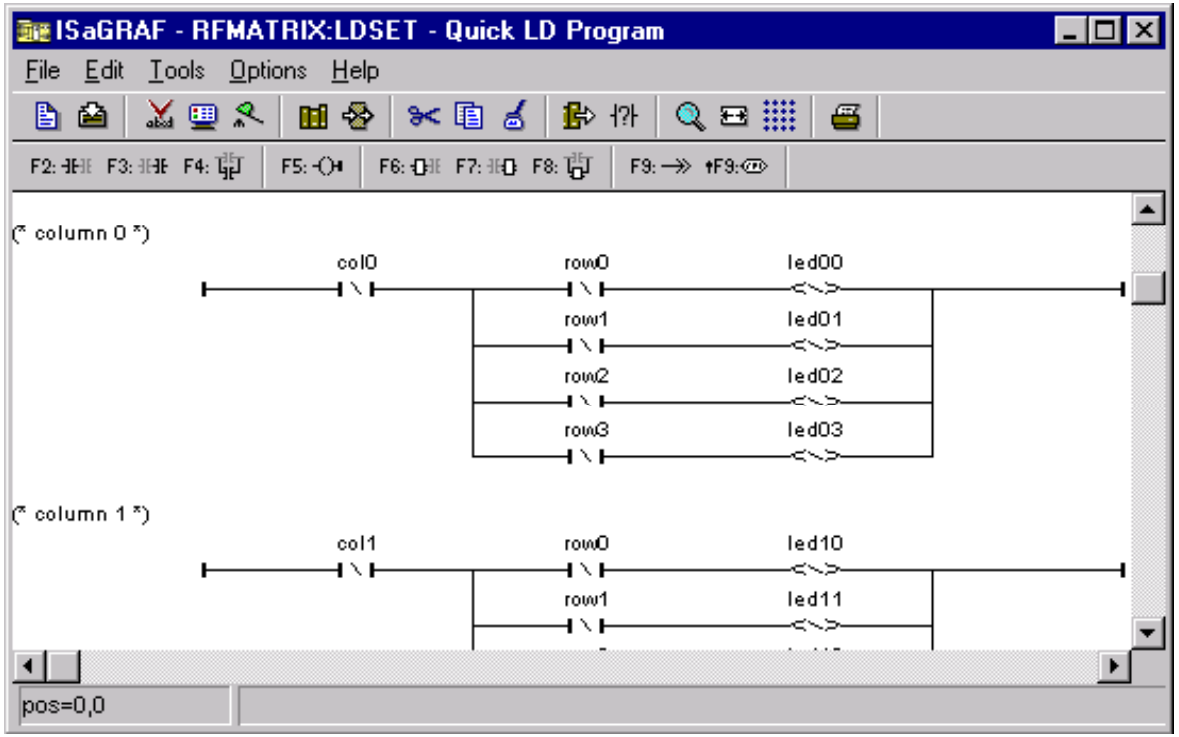
Редактор Quick LD

Quick LD Editor

Контакты / contacts

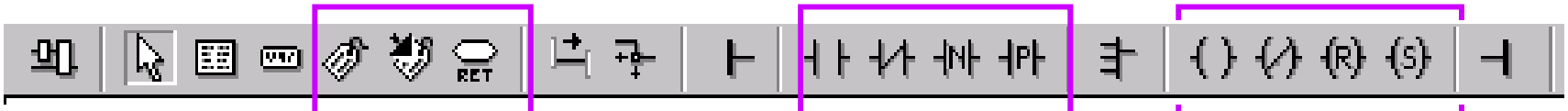
Катушка / Coil

Управление программой / Program control



- Редактирование функциональным и кнопками
- Пробел для изменения контакта или типа Катушки
- Key Board Edition
- Space Bar to change contact or coil type

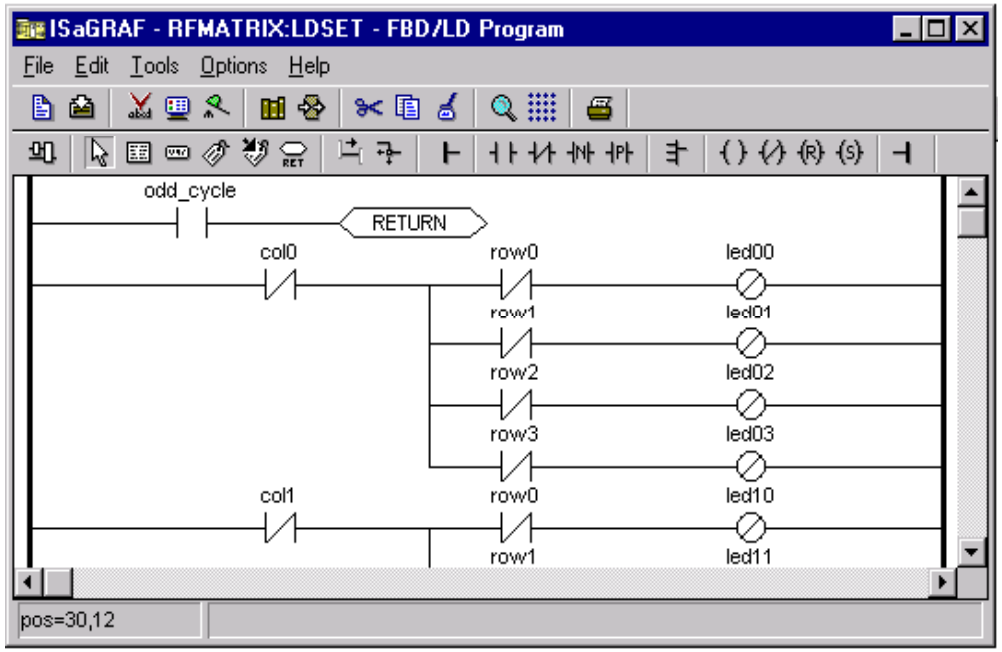
Редактор FBD/LD
FBD/LD Editor



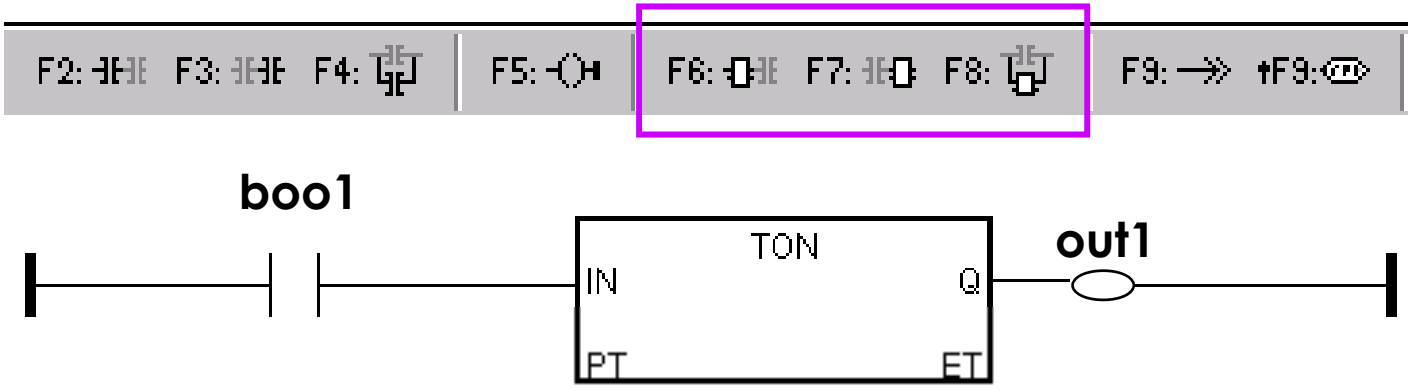
Управление программой / Program control

контакты / contacts

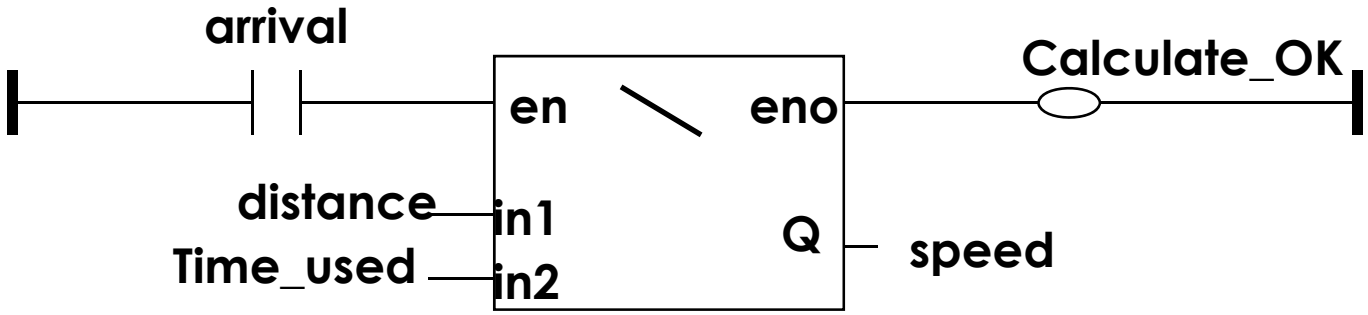
Катушки / Coils



Блоки в Редакторе Quick LD
Blocks in Quick LD Editor



Блоки с Логическими входами / Blocks with Boolean inputs



Блоки без Логических входов / Blocks without Boolean inputs

LD Quick View (Быстрый просмотр)
LD Quick View

- **В основном Логические функции**
- **Уравнения Графического And & OR**
- **Для тестирования входных и внутренних переменных**
- **Для задания комбинаторных уравнений**

- **Mainly for Boolean equations**
- **Graphical And & OR equations**
- **For tests on inputs or internal variables**
- **For defining combinatory equations**

- **Выполняются сверху вниз**
- **Контроль выполнения с переходами и повторами**

- **Is executed from top to bottom**
- **Execution control with jumps and returns**

- **Редактор QLD может использоваться в SFC преобразованиях, FC действиях и решениях**

- **QLD Editor can be used in SFC transitions, FC actions and decisions**

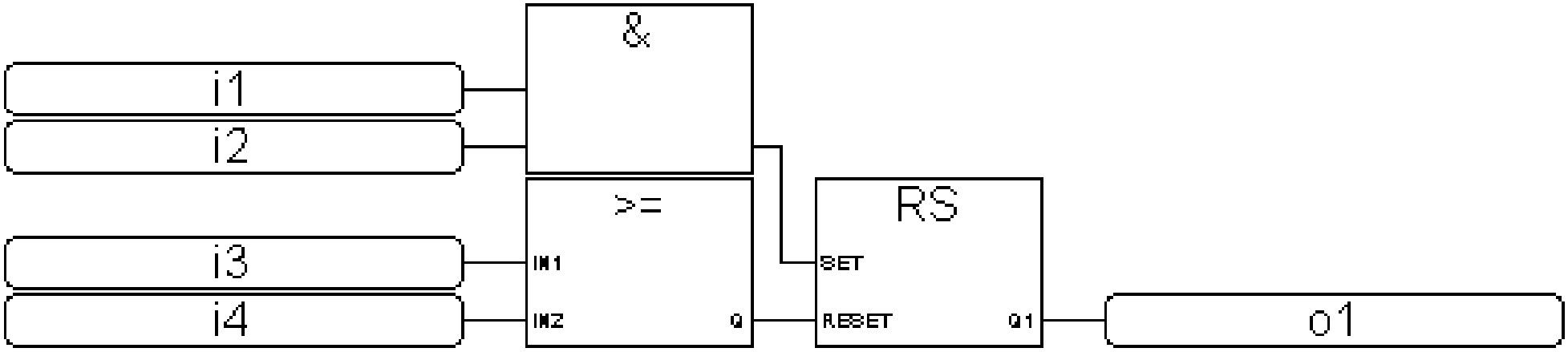
Язык Функциональных Блок-Схем

Function Block Diagram Language

- *Язык FBD*
- *FBD/LD Редактор & Специальные Характеристики*
- *Quick LD Редактор & Специальные Характеристики*

- *FBD Language*
- *FBD/LD Editor & Specific Features*
- *Quick LD Editor & Specific Features*

FBD Блоки и Переменные
FBD Blocks and Variables



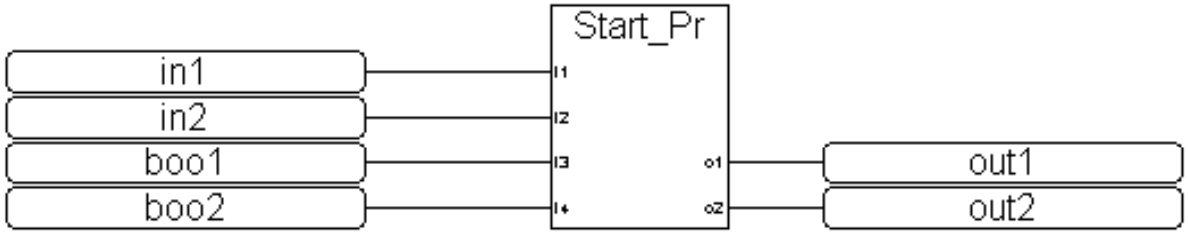
- **Блоки могут быть**
  - Стандартными операторами и Функциями
  - Функции из секции Функций или библиотеки Функций ISaGRAF
  - Функциональные блоки из секции Функц. блоков или библиотеки Функ. блоков ISaGRAF C
  - усовершенствованные операторы ISaGRAF (несколько входов)

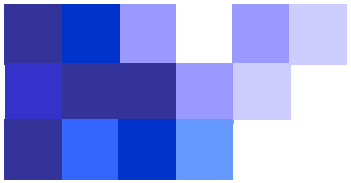
- **Blocks can be**
  - Standard Operators and Functions
  - Functions from the Function section or the ISaGRAF Functions library
  - Function Blocks from the F. Blocks section or the ISaGRAF F. Blocks library
  - C Functions or C Function blocks from the ISaGRAF library
  - ISaGRAF enhanced operators (several inputs)

Связь между блоками
Block Connection

- **Входом блока может быть**
  - **Входная переменная**
  - **Внутренняя переменная**
  - **Константа**
  - **Выходная переменная**
- **Выходом блока может быть**
  - **Внутренняя переменная**
  - **Выходная переменная**
  - **Название программы (для подпрограмм)**

- **Input of a block can be**
  - **Input variable**
  - **Internal variable**
  - **constant**
  - **Output variable**
- **Output of a block can be**
  - **Internal variable**
  - **Output variable**
  - **The name of the program (for sub-programs)**



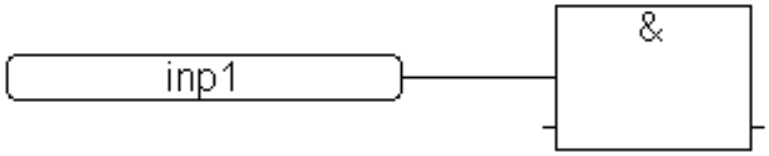


Одиночные Соединения
Single Connections

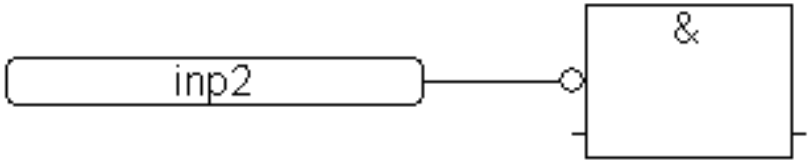
- **Соединение одиночными линиями**
  - **Входной переменной и входа блока**
  - **выход блока и вход другого блока**
  - **выход блока и выходная переменная**
- **Левая и правая сторона должны иметь один тип**

- **Single lines connect**
  - **An input variable and an input of a block**
  - **An output of a block and an input of another block**
  - **An output of a block and an output variable**
- **Left and right edges must have the same type**

Прямое или Реверсивное одиночное соединение
Direct or Reverse Single Connection



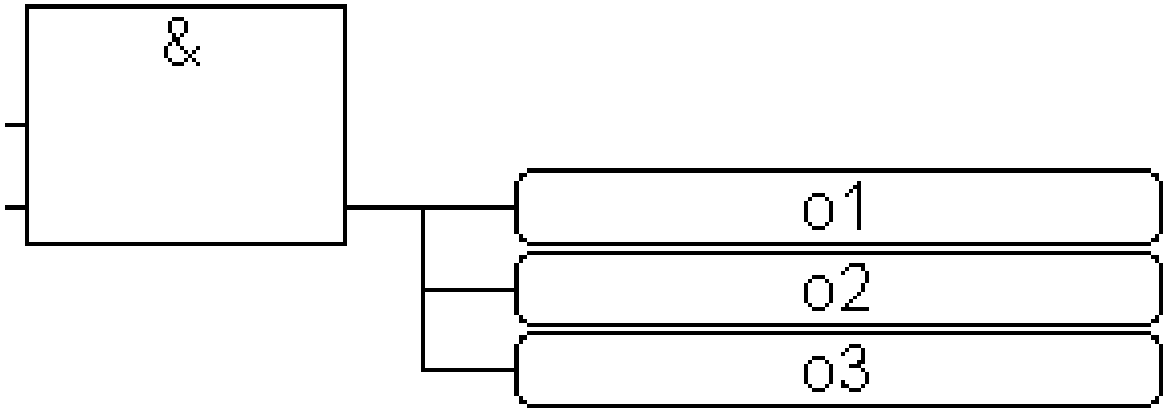
- **Прямое (положительное) соединение**
  - **Direct connection**



- **Реверсивное (отрицательное) соединение**
  - **Reverse connection**

Множественное соединение

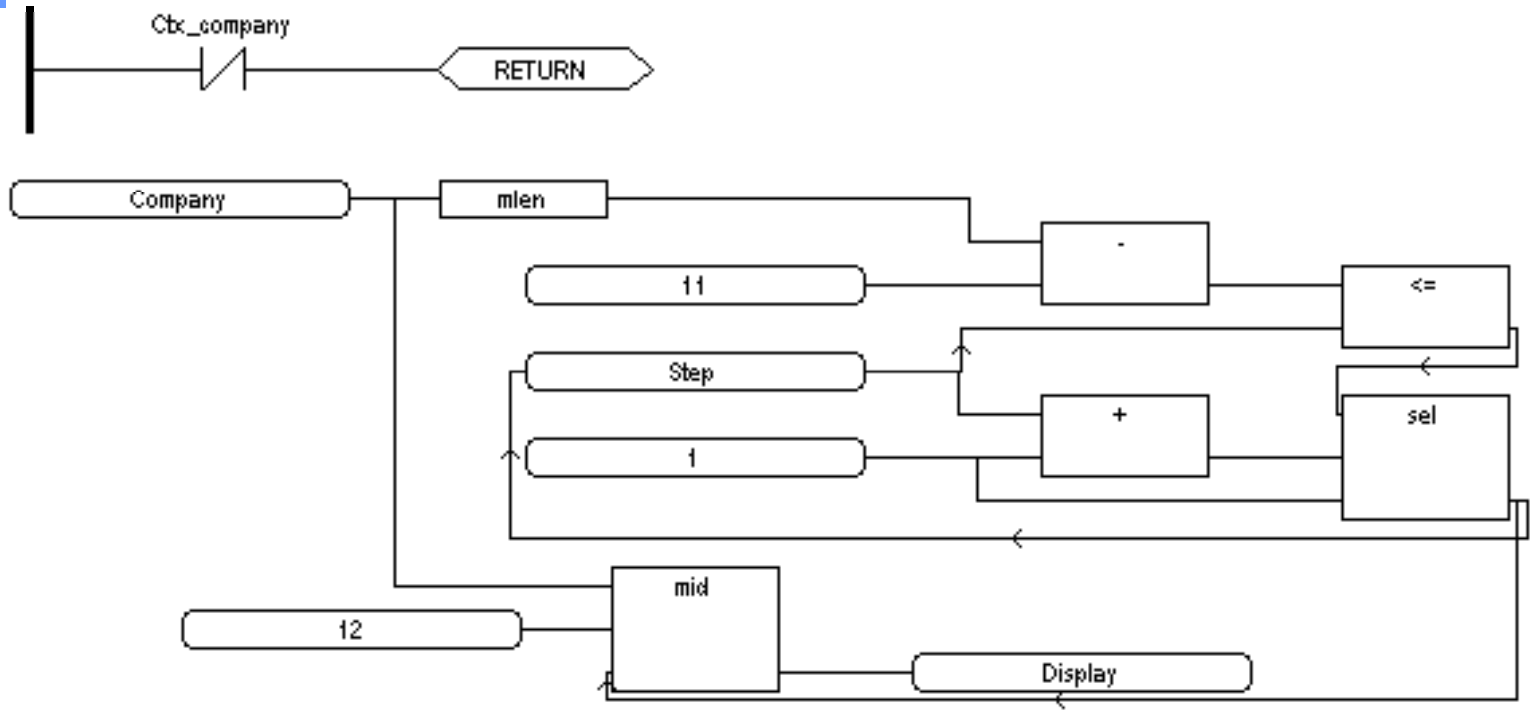
Multiple Connection



- **Используются для передачи информации с левой стороны на каждую из правых**
- **Все окончания соединений должны иметь один тип**

- **Are used to broadcast an information from left edge to each right edge**
- **All edges must have the same type**

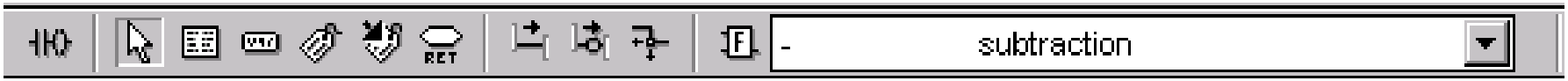
Выполнение FBD
FBD Execution

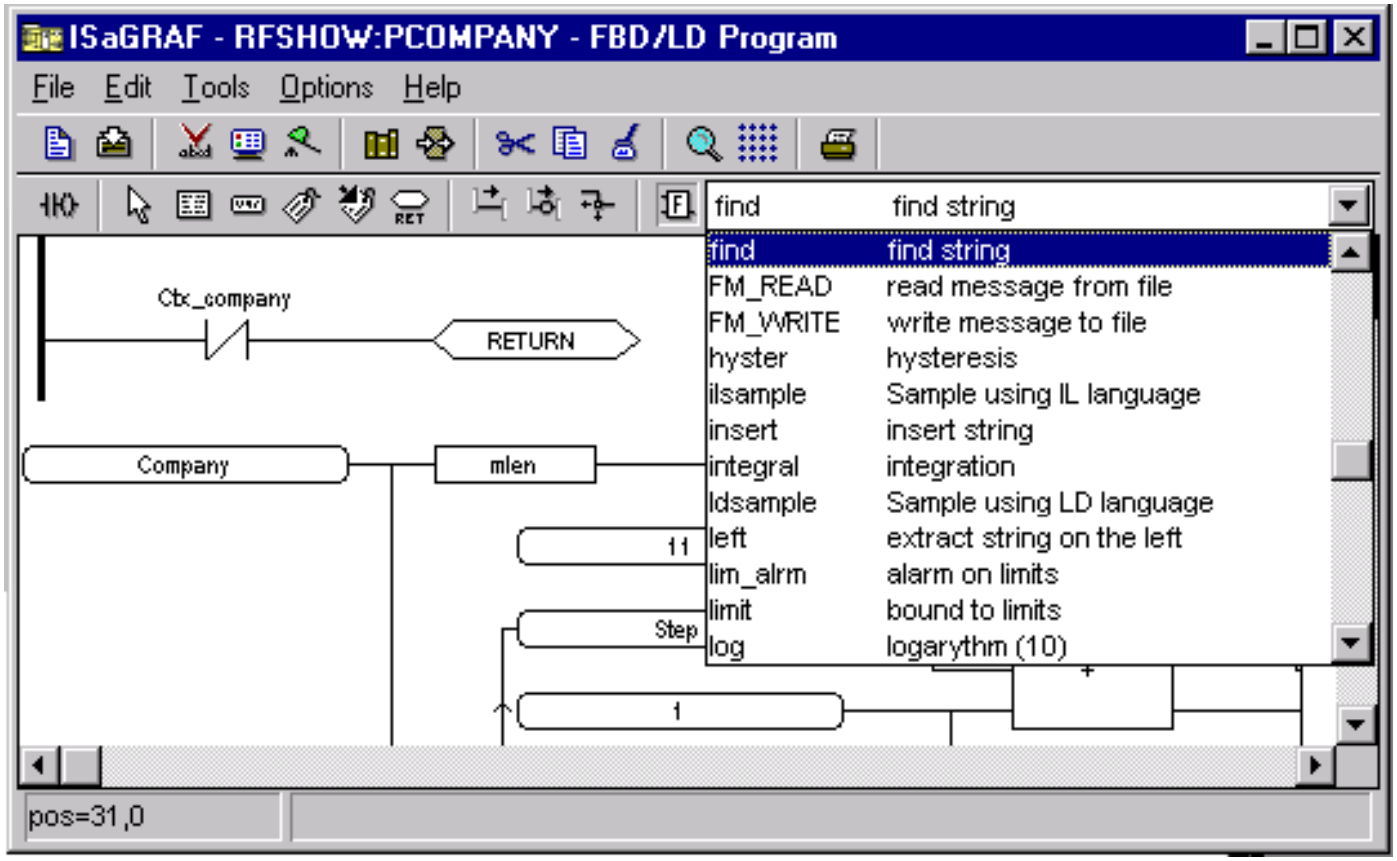


- Программа FBD выполняется сверху вниз
- Можно просматривать эквивалент ST всей программы

- An FBD program is executed from top to bottom
- It is possible to view the ST equivalent of the whole program

FBD/LD Редактор
FBD/LD Editor





FBD/LD Редактор специальных функций
FBD/LD Editor Specific Features

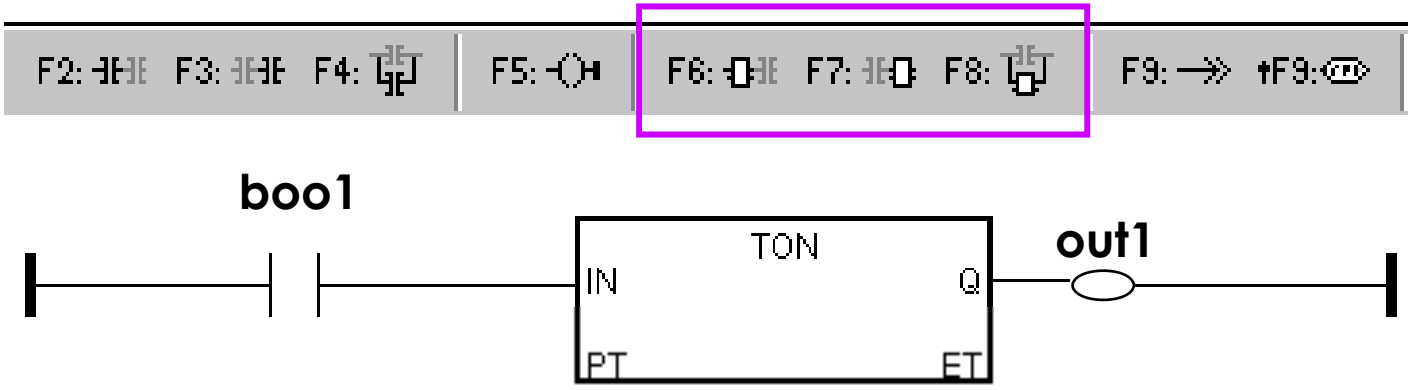
- Реверсивное соединение
- Редактирование в режиме изменения маркеров
- Функция показа очередности выполнения
- Просмотр прекомпиляционных кодов

- Плюс функции обычного редактора

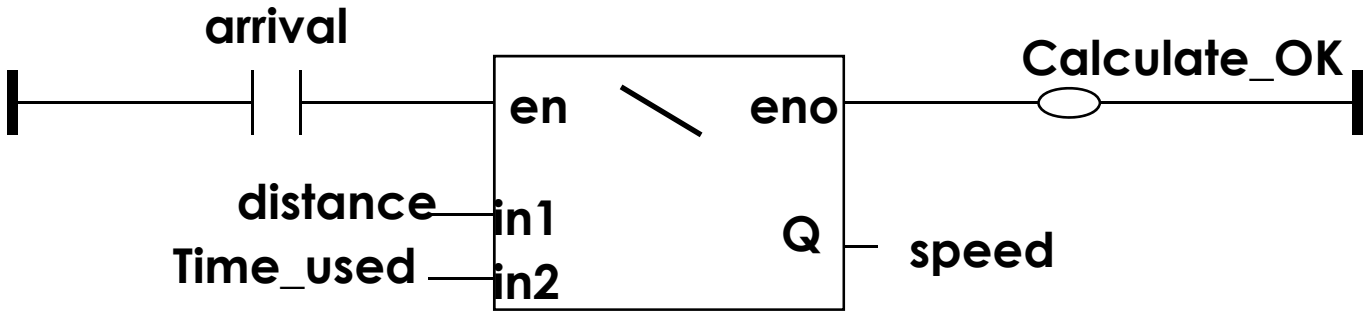
- Reverse connection
- Edit with marked modifications mode
- Show execution order feature
- View precompiled code

- Plus usual editor functions

Блоки в Редакторе Quick LD
Blocks in Quick LD Editor



Блоки с Логическими входами / Blocks with Boolean inputs



Блоки без Логических входов / Blocks without Boolean inputs

Quick View (Быстрый просмотр) языка FBD
FBD Language Quick View

- **Язык высокого уровня**
- **Позволяет реализовать сложные алгоритмы простым вызовом функций и функциональных блоков.**
- **Полностью графический язык**

- **Два редактора**

- **High level language**
- **Allow to process very powerful algorithms in easy call of function & function blocks**
- **Completely graphical language**

- **Two editors**

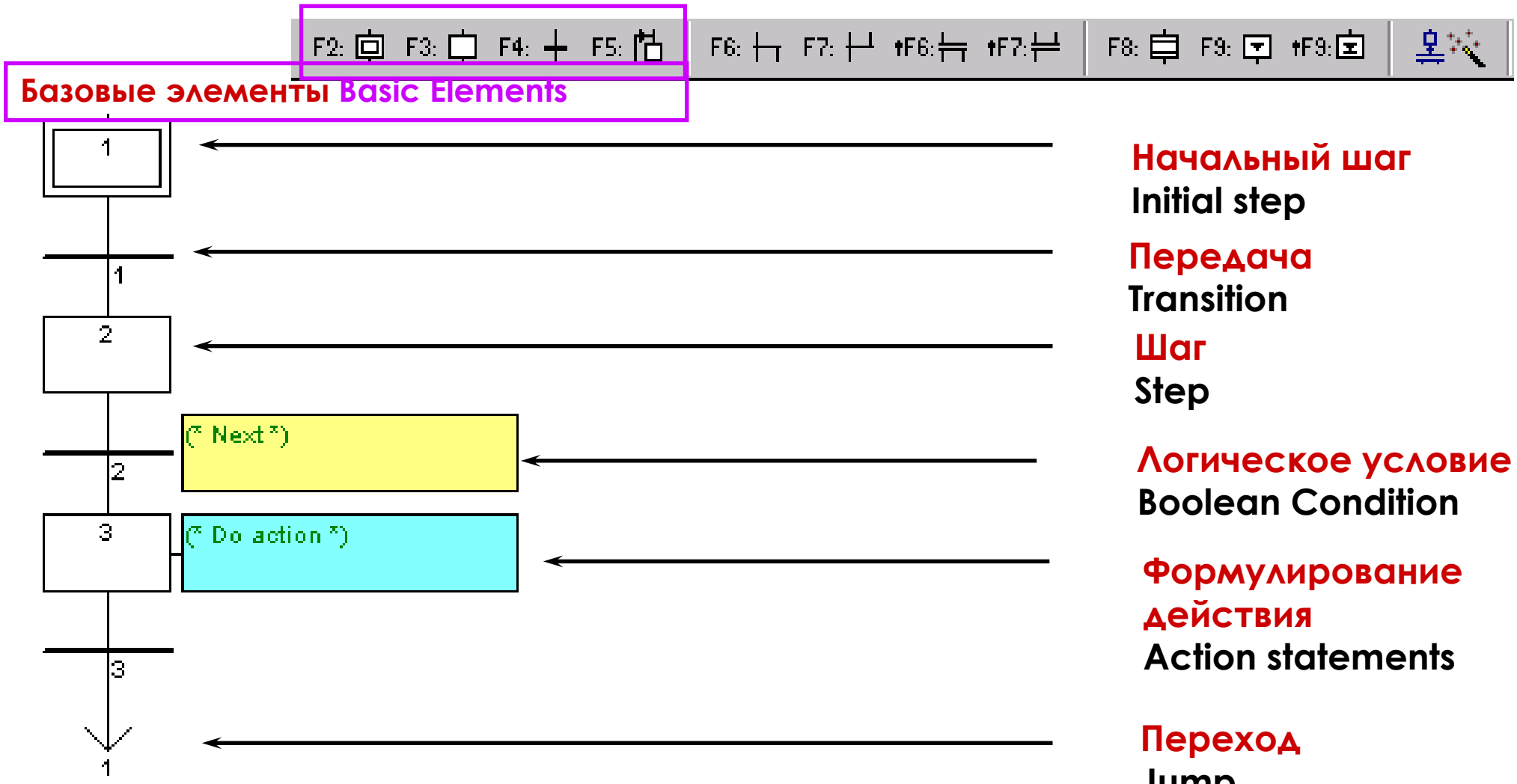
Язык Последовательных функциональных схем

Sequential Function Chart Language

- *Базисы*
- *Иерархические связи*
- *Язык*
- *Редактор*

- *Basics*
- *Hierarchy Links*
- *Language*
- *Editor*

Базовые компаненты SFC
SFC Basic Components



Начальный шаг
Initial step

Передача
Transition

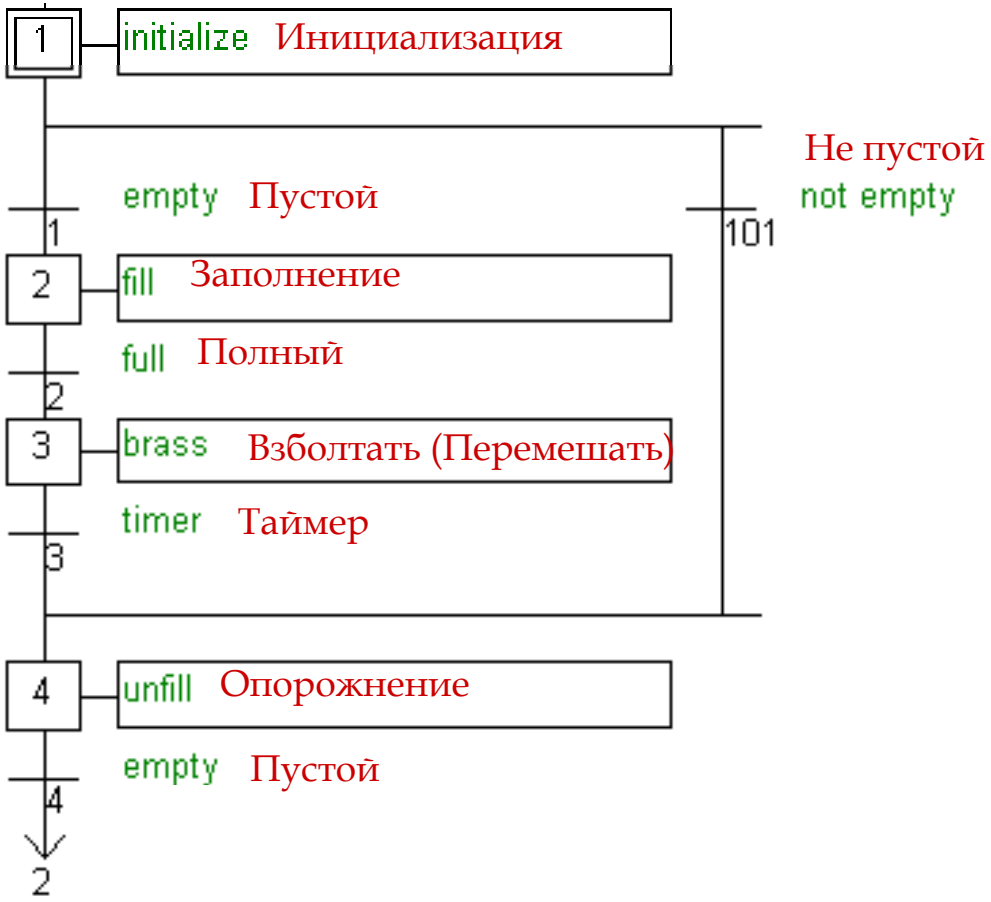
Шаг
Step

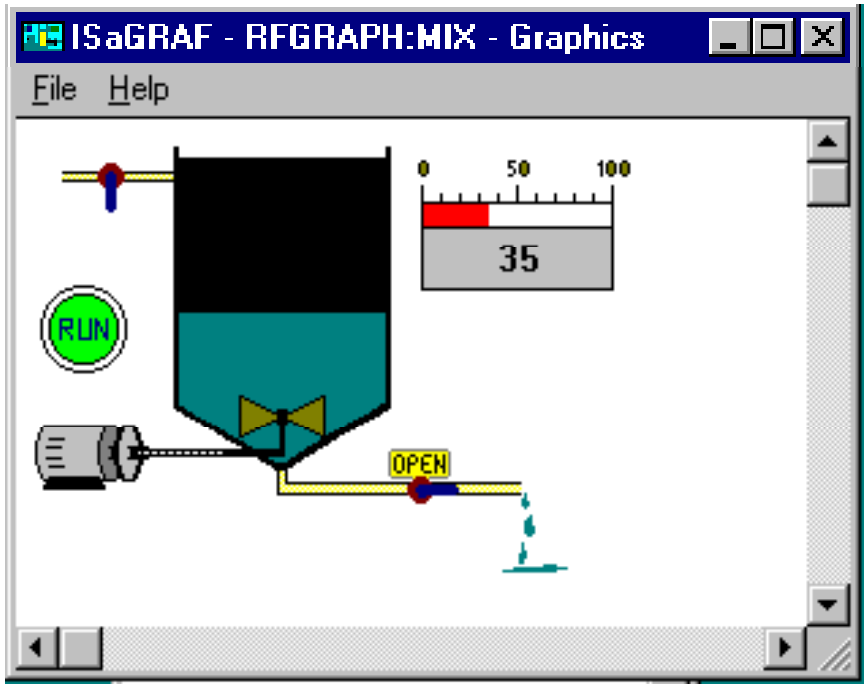
Логическое условие
Boolean Condition

Формулирование действия
Action statements

Переход
Jump

SFC : Общий вид
SFC : General View



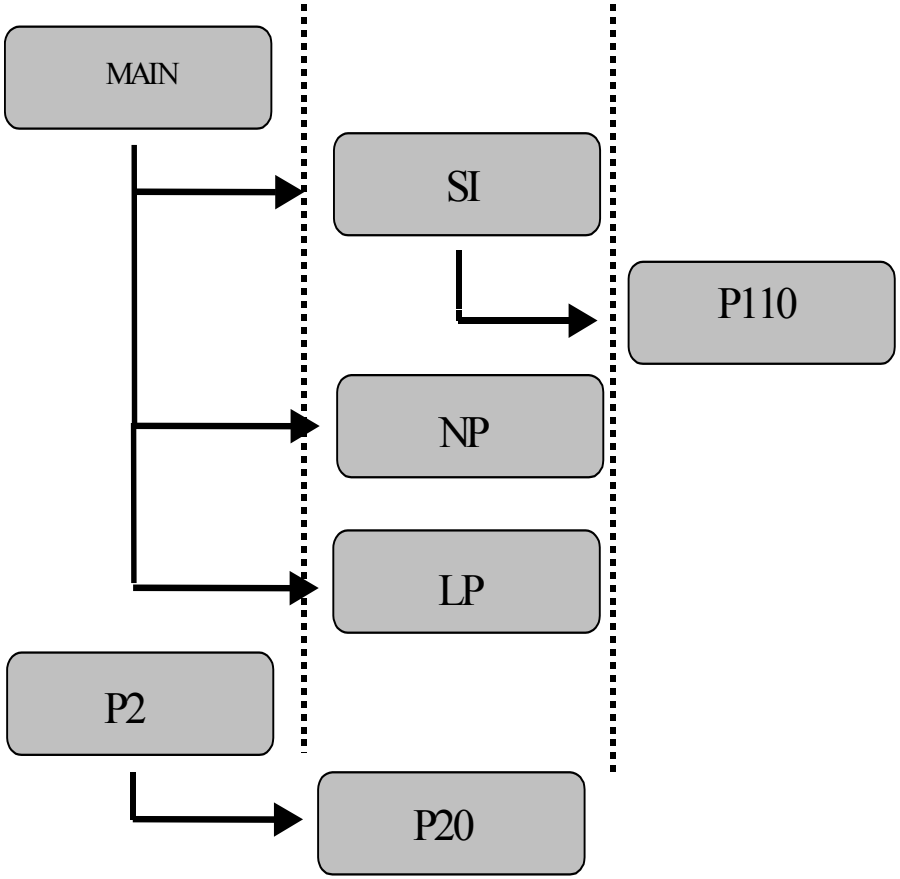


SFC Отношения между программами
SFC Relation Between Programs

Родитель с сыну / Father to Child

SFC Отображение иерархии
SFC Hierarchy Representation

Действие от Отца к Сыну / Father to Child Actions



SFC Принцип Иерархии
SFC Hierarchy Principle

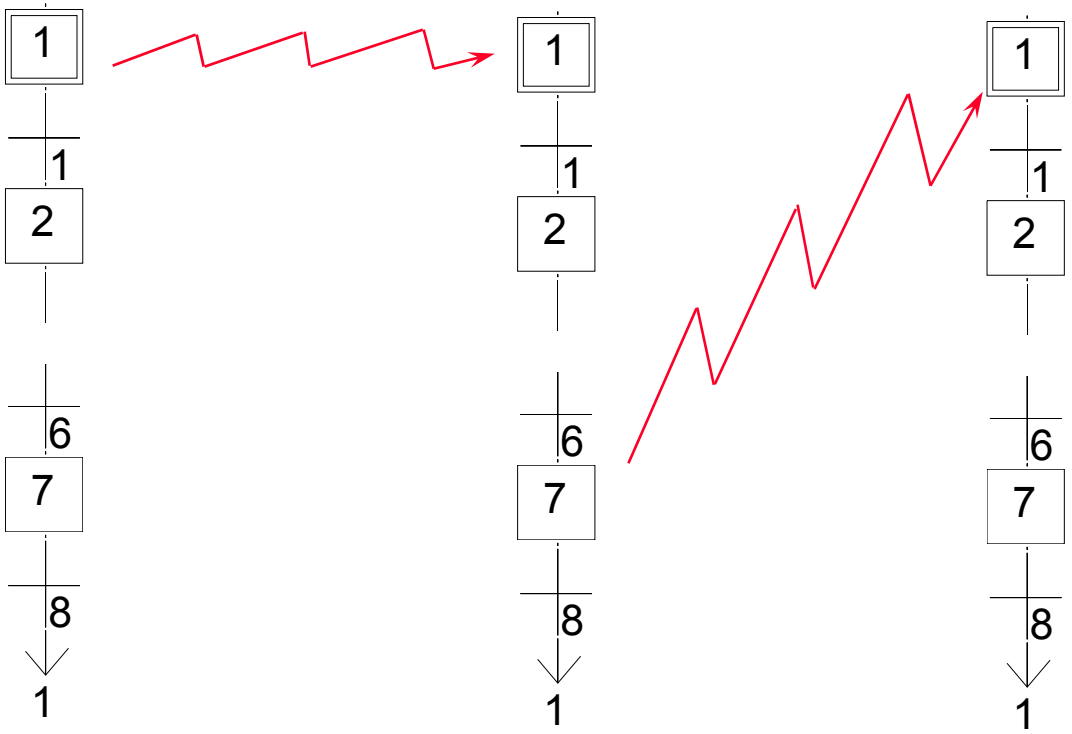
- Связь устанавливается между родительской и сыновней программой
- MAIN и P2 называются «основными программами» и автоматически запускаются ISaGRAF
- Другие программы запускаются Родителями
- Сыновние программы извесны только Родительским программам
- Сыновняя программа имеет только одну Родительскую
- P2 не может активизировать P110 иначе, как через глобальные переменные

- A link is set between a father and a child program
- MAIN and P2 are called "main programs" and are automatically started by ISaGRAF
- Other programs are started by their fathers
- A father program only knows its child programs
- A child program only has one father program
- P2 cannot act on P110 except through global variables

SFC : Сыновние SFC Программы
SFC : Child SFC Programs

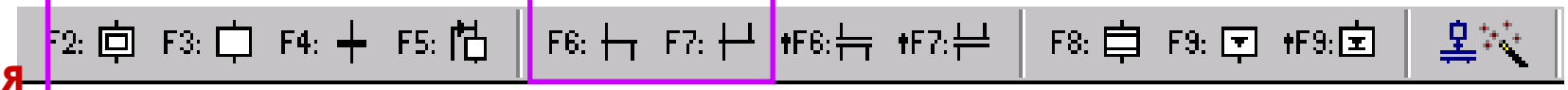
- Легкое программирование различных частей применения (по принципу иерархии SFC)
- Вертикальная параллельная структура

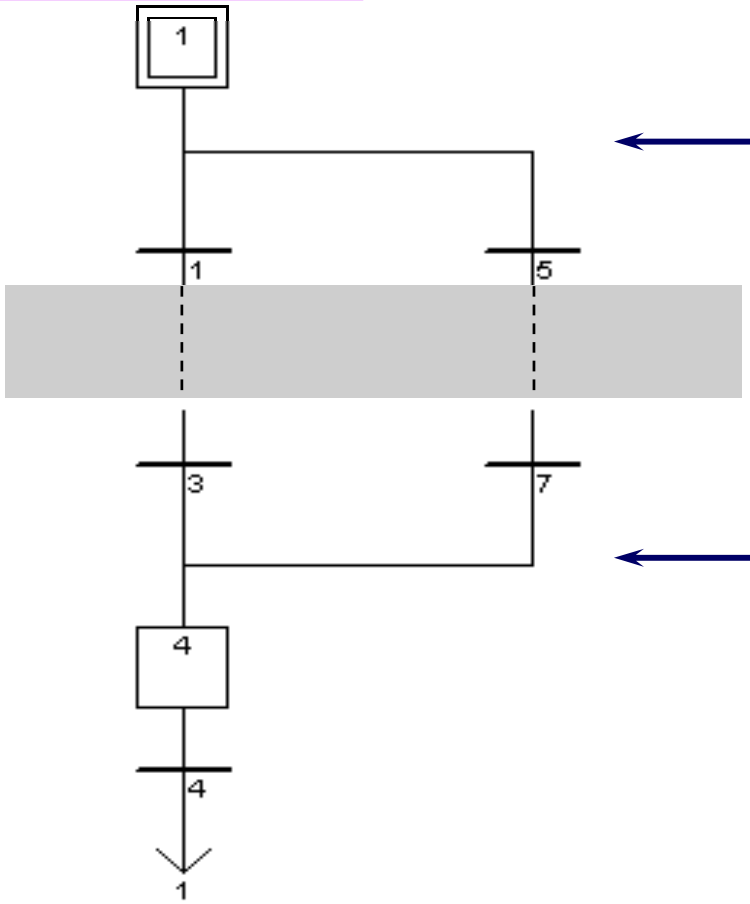
- Easy programming of the different application parts (with the SFC hierarchy principle)
- Vertical parallel structure



Параллельные ветви ИЛИ: Выбор
OR Parallel Branches: Selection

Divergences
Разветвления





Разветвление ИЛИ (одной линией)
(исключается однозначно !)

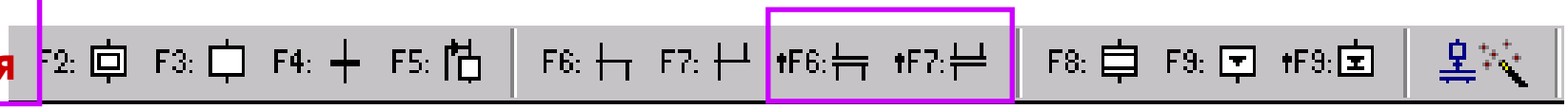
OR divergence (single line divergence)
(not implicitly exclusive !)

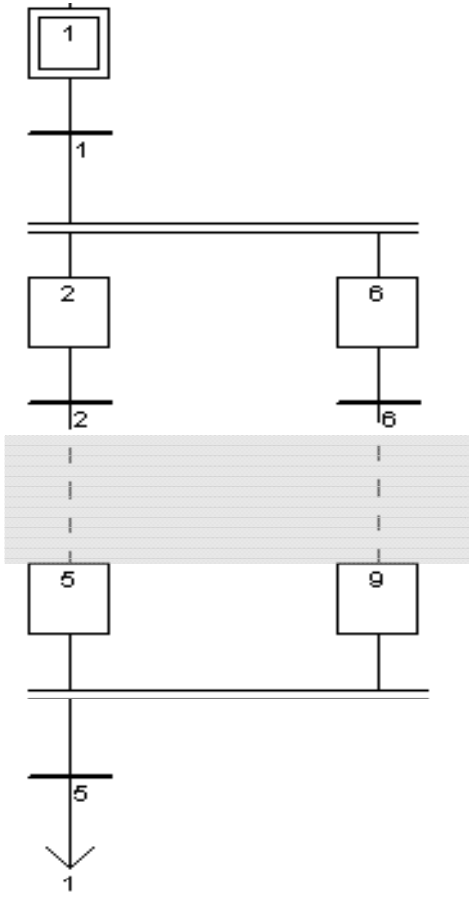
Объединение ИЛИ (одной линией)

OR convergence (single line convergence)

Параллельные ветви И: Синхронизация
AND Parallel Branches: Synchro

Divergences
Разветвления





Разветвление И (двойной линией)
(включительно!)

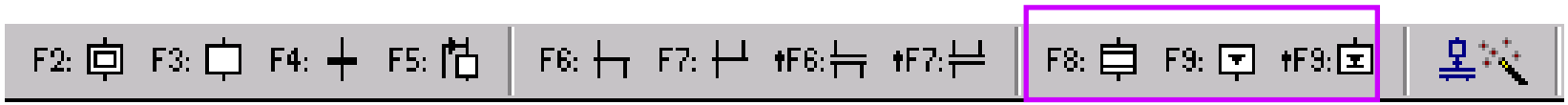
AND divergence (double line divergence)
(inclusive !)

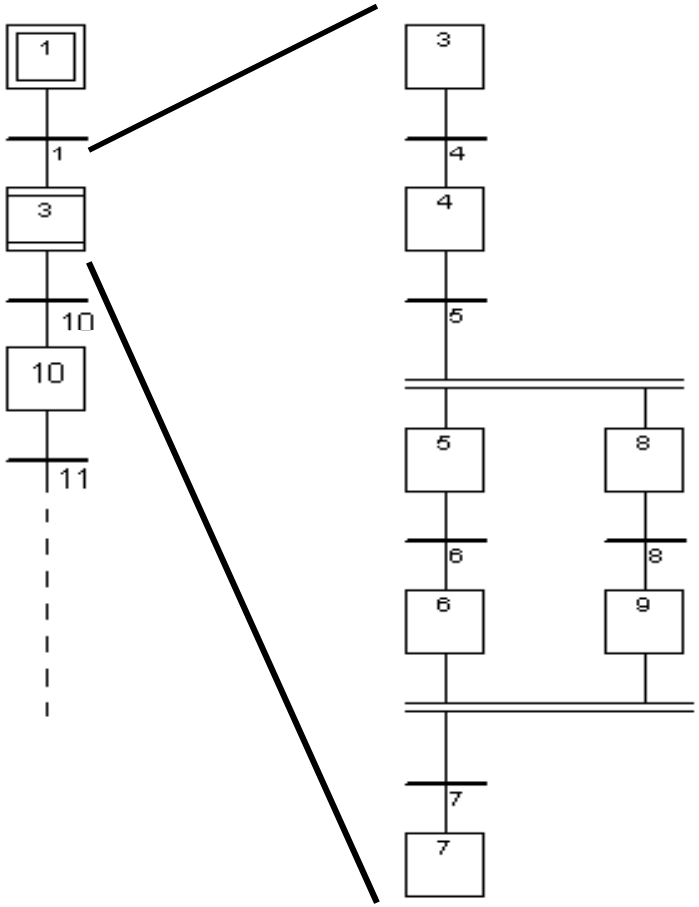
Объединение И (двойной линией)

AND Convergence (double line convergence)

Макро-Шаги
Macro Steps

Макрошаги
Macro steps

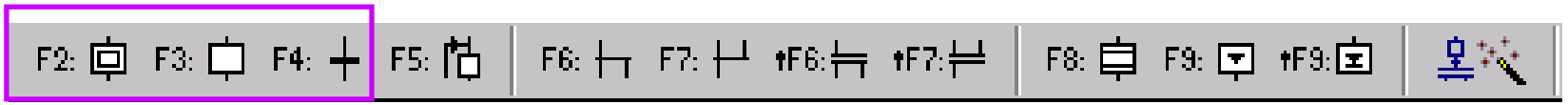


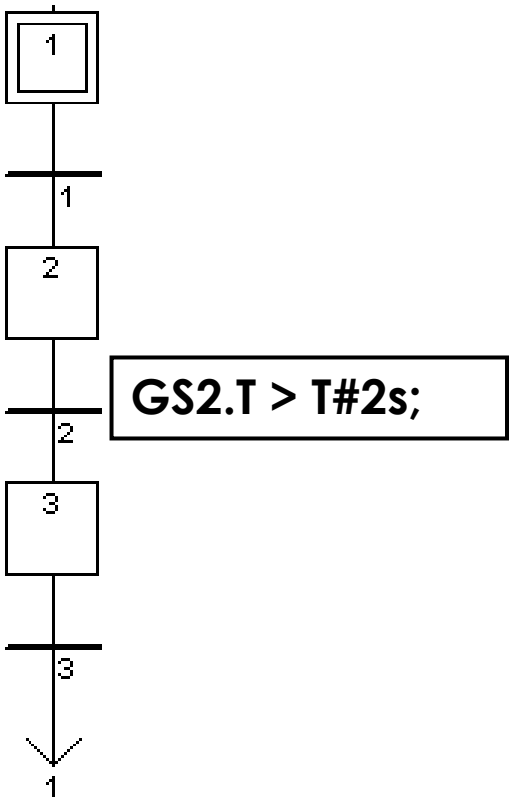


- Макрошаги и тело программы должны находиться в одной программе
- Macro step and body must be in the same program

- Макро шаг и первый шаг Макрошага должны иметь один номер.
- Macro step and macro first step must have the same number

Описание Объектов SFC
SFC Objects References



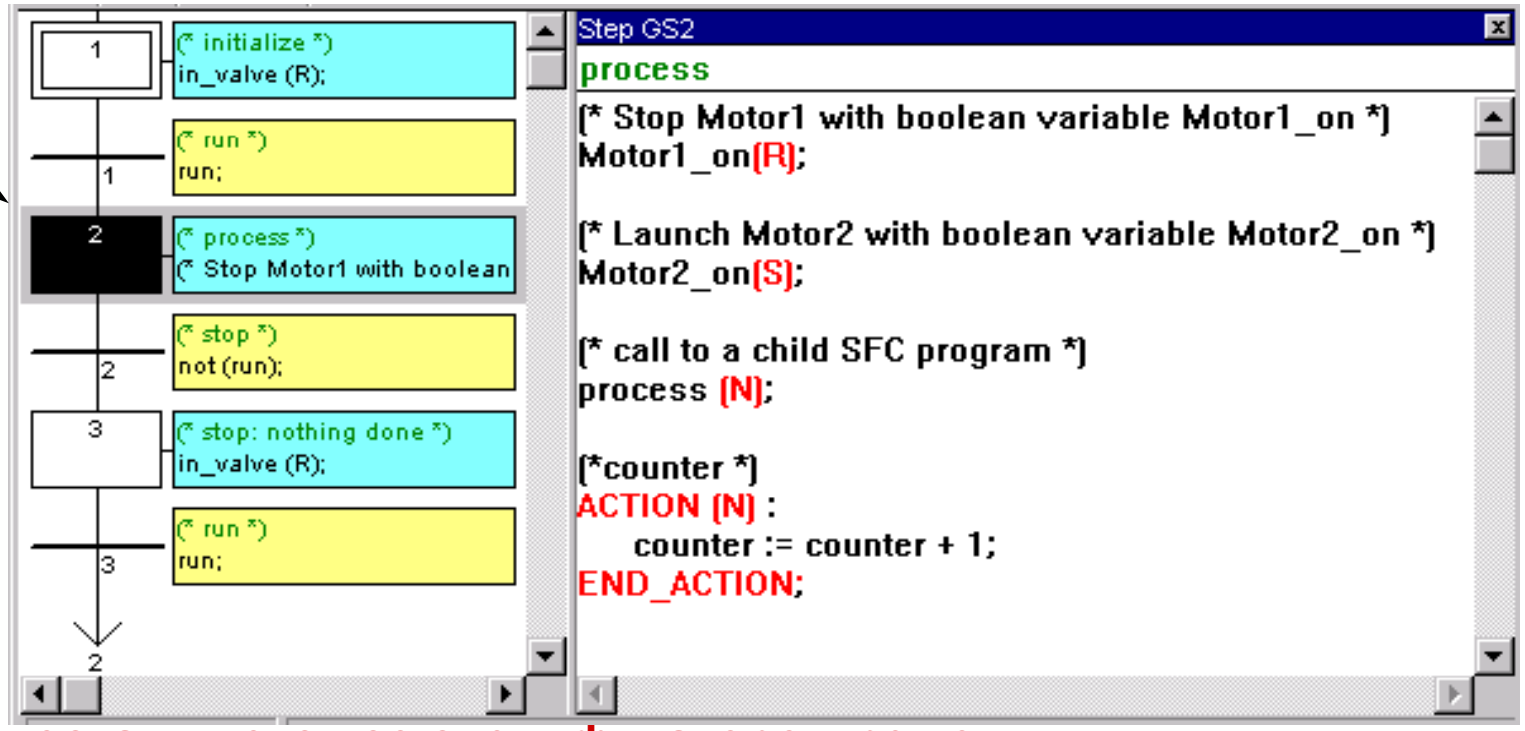


- Шаг номер 1 называется GS1
  - Step number 1 is called GS1
- Передача номер 1 называется GT1
  - Transition number 1 is called GT1

- GS101.X представляет активность GS101 (BOO vble)
  - GS101.X represents the activity of GS101 (BOO vble)
- GS2.T представляет время активности GS2 (TMR vble)
  - GS2.T represents the activity duration of GS2 (TMR vble)

Действия на шагах SFC
Actions Within SFC Steps

- **Шаг выполняется при активизации**
- **A step is executed when activated**



- **Все действия сопоставляются со спецификатором решения**
  - **Событием выполнения / или самого действия**
- **Имеются различные спецификаторы**
  - **Состояния действия / Работа с логическими / Работа с SFC детьми**

- **All actions are associated to a qualifier to determine**
  - **Execution occurrences / or the action itself**
- **There are different qualifiers**
  - **Action Statements / To deal with booleans / To deal with SFC children**

Действия на шагах SFC
Actions Within SFC Steps

Группы спецификаторов / Groups of Qualifiers

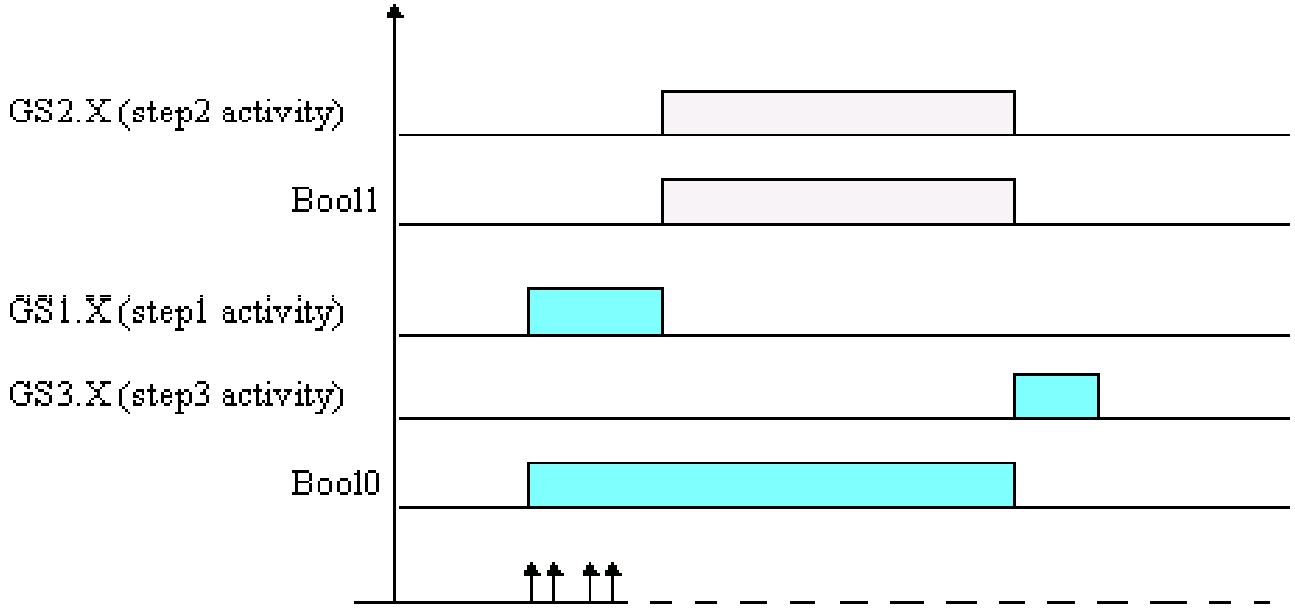
- **N : НЕ-СОХРАНЯЕМЫЕ : Действия активны пока остается активным шаг**
  - Для Логических действий, контроля сыновей SFC, Состояний
  - N является спецификатором по умолчанию
- **S : УСТАНОВКА : Действия устанавливаются при активации шага**
- **R : СБРОС : Действия сбрасываются при активации шага**
  - Спецификаторы для контроля Логических переменных, сыновних программ SFC
- **P, P1 : ИМПУЛЬС : Действие выполняется разово при активизации шага**
- **P0 : Действие выполняется разово при де активизации шага**
  - Спецификаторы для контроля состояний действия

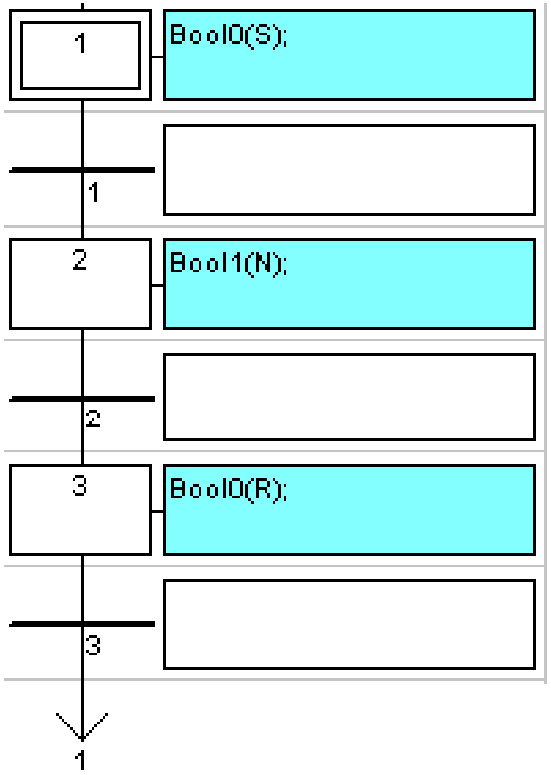
- **N : NON-STORED : Actions remain active as long as step is active**
  - For Boolean actions, Child SFC control, Statements
  - N is the default qualifier
- **S : SET : Actions are set when step is activated**
- **R : RESET : Actions are reset when step is activated**
  - Qualifiers to control Boolean variables, Child SFC programs
- **P, P1 : PULSE : Actions are executed once on step activation**
- **P0 : Actions are executed once on step de-activation**
  - Qualifiers to control action statements

Действия на шагах SFC
Actions Within SFC Steps

Логические Действия / Boolean Actions

- **Спецификатор ‘N’ поддерживает Логическую переменную в TRUE пока шаг активен**
- **Спецификатор ‘S’ устанавливает Логическую переменную в TRUE**
- **Спецификатор ‘R’ сбрасывает Логическую переменную в FALSE**



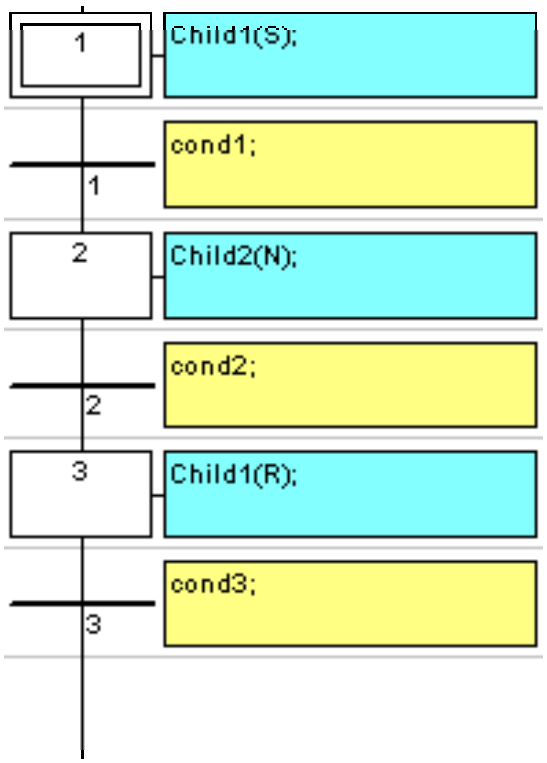


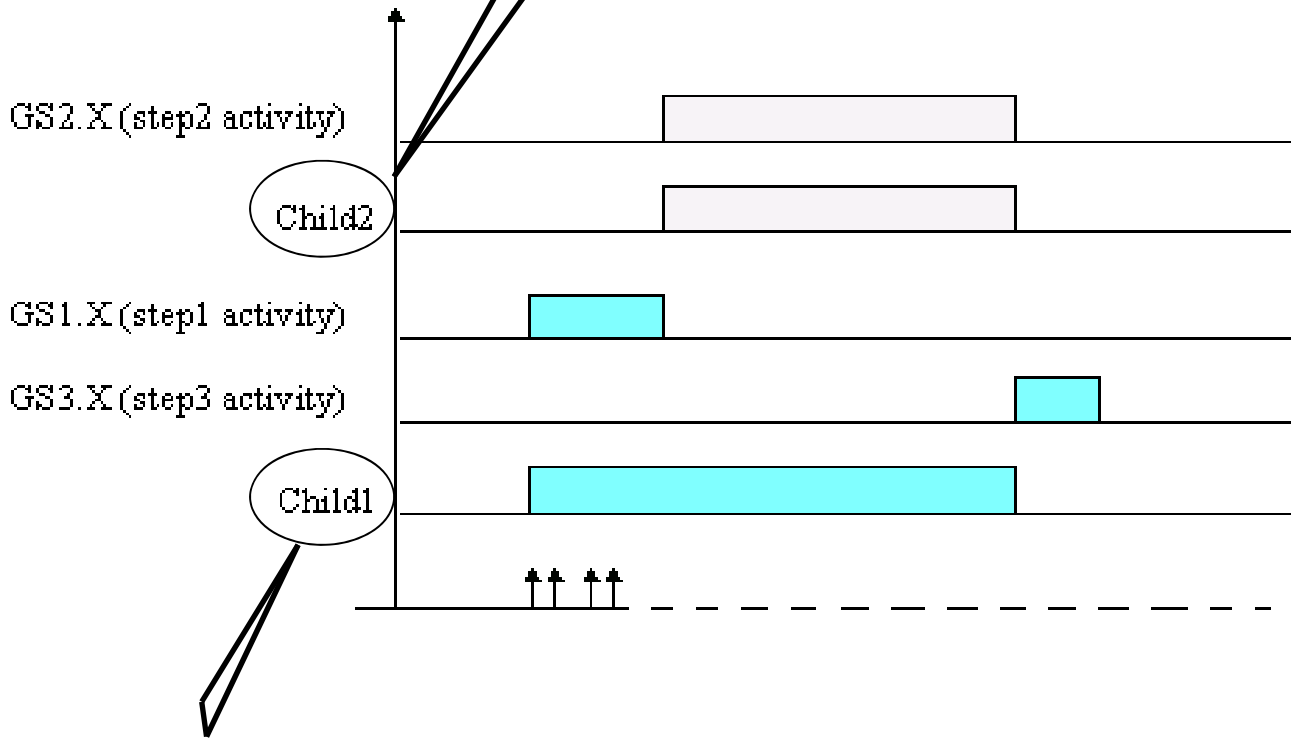
- **‘N’ Qualifier makes the Boolean variable be TRUE as long as the step is active**
- **‘S’ qualifier sets the Boolean variable to TRUE**
- **‘R’ qualifier resets the Boolean variable to FALSE**

Действия на шагах SFC
Actions Within SFC Steps

Действия детей SFC / Child SFC Actions

- Действие спецификатора N
- N qualifier action





Комбинация спецификаторов S и R
Combination of S and R qualifiers

Операторы Управления ST SFC
ST SFC Control Operators

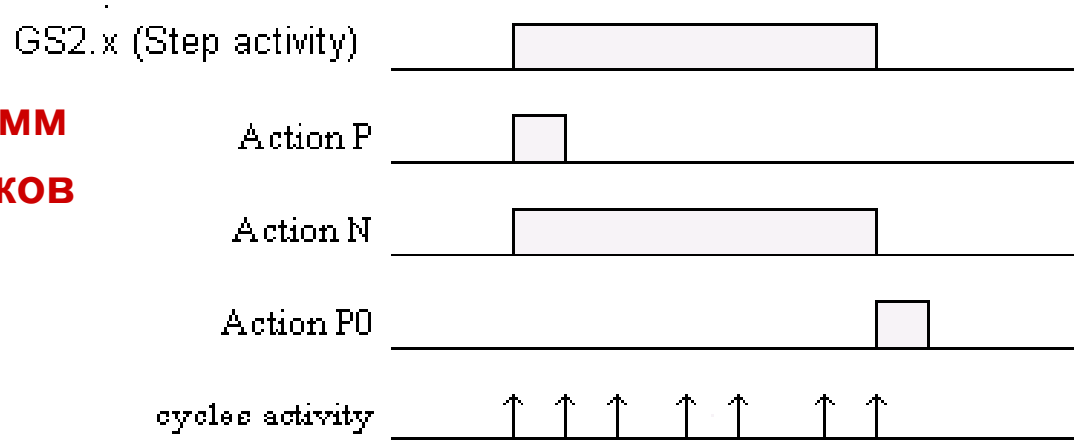
- управление сыном SFC
  - GSTART () Запуск сыновней программы
  - GKILL () Остановка сына (передается)
  - GFREEZE () Остановка сына (возвращаемый знак) (передается)
  - GRST () Сброс сыновней программы (после GFREEZE)
  - GSTATUS () Get status of a child
- Ознакомитесь с использованием N, S и R спецификаторов в сыновней программе
- Атрибуты шага (скрытые переменные)
  - GSn().X Активность шага n (Логическое значение)
  - GSn().T Длительность шага n (Значение времени)

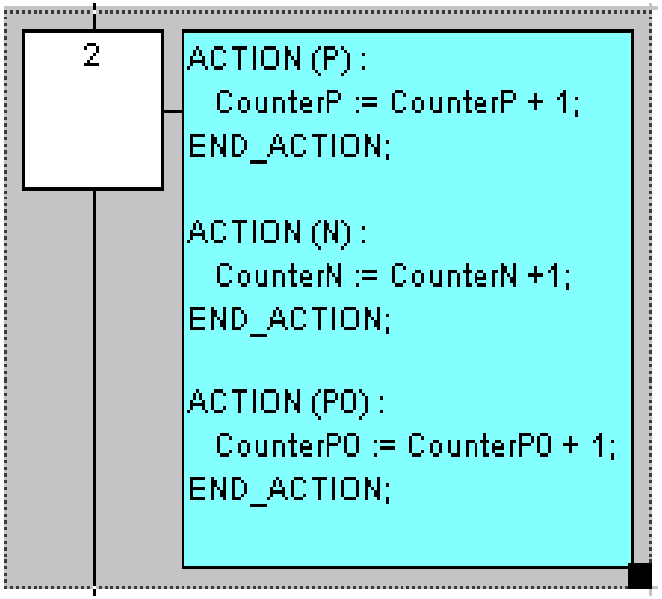
- SFC child control
  - GSTART () Start a child
  - GKILL () Stop a child (propagated)
  - GFREEZE () Stop a child (recoverable tokens) (propagated)
  - GRST () Restart a child (after GFREEZE)
  - GSTATUS () Get status of a child
- See N, S and R qualifiers used on a child program
- Attributes of a step (implicit variables)
  - GSn().X Activity of step n (Boolean value)
  - GSn().T Duration of step n (timer value)

Действия на шагах SFC
Actions Within SFC Steps

Предписания / Statements

- Для
  - Вызова функций / подпрограмм
  - Вызов функциональных блоков
  - Любое другое предписание





- For
  - Function call / sub program
  - Function Block call
  - Any other statement

SFC Динамические правила
SFC Dynamic Rules

- **Правило N° 1 : Ситуация инициализации**

Шаги инициализации SFC программ верхнего уровня отображают состояние инициализации всей системы

  - **Можно установить значения переменных инициализации**
  - **Должен быть хотя бы один шаг инициализации в SFC программе (их может быть и много)**
  - **Шаги инициализации сыновних SFC программ представляют собой «локальную» ситуацию инициализации**

- **RULE N° 1 : Initial situation**

Initial steps of the highest level SFC programs represent the whole system initial state

  - **Initial variable values can be set**
  - **There must be at least one initial step in an SFC program (there may be many)**
  - **Initial steps of child SFC programs represent "local" initial situations**

SFC Динамические правила
SFC Dynamic Rules

- **Правило N° 2 : Очистка перехода**

  Переход активизируется когда все все непосредственно предшествующие шаги активны.

  Переход очищается когда :

  - это возможно

  - и его условие TRUE

- **RULE N° 2 : Clearing of a transition**

  A transition is enabled when all immediately preceding steps are active.

  A transition is cleared when :

  - It is enabled

  - and its condition is TRUE

SFC Динамические правила
SFC Dynamic Rules

- **ПРАВИЛО N° 3 : Развитие активных шагов**

  За очисткой перехода следуют одновременно

  - активизация всех последующих шагов

  - деактивизация всех предшествующих шагов

- **RULE N° 3 : Evolution of active steps**

  The clearing of a transition leads to simultaneous

  - **activation of all its following steps**

  - **deactivation of all its preceding steps**

SFC Динамические правила
SFC Dynamic Rules

- **ПРАВИЛО N° 4 :**
  Одновременная очистка

  Все переходы одновременно очищаемые одновременно и очищаются

  Это распространяется на все переходы всех ветвей во всех SFC программах

- **RULE N° 4 : Simultaneous clearing**

  All transitions simultaneously clearable are simultaneously cleared

  This is true for all transitions of all branches in all SFC programs

SFC Динамические правила
SFC Dynamic Rules

- **ПРАВИЛО N° 5 : Одновременная активизация и деактивизация**

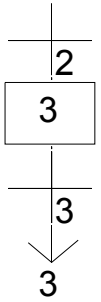
  Если шаг одновременно должен активизироваться и деактивизироваться то он сохраняет свое состояние

  – **Плохое управление ветвями ИЛИ**
  – **Переход назад на предыдущий шаг : N (BOO and CHILD), S, R и P, P0 действия не перевыполняются!**

- **RULE N° 5 : Simultaneous activation and deactivation**

  If a step has to be simultaneously activated and deactivated, it stays the same

  – **Bad management of OR branches**
  – **Jump back to the preceding step: N (BOO and CHILD), S, R and P, P0 actions are not re-executed !**



ISaGRAF Выполнение : SFC Алгоритм

ISaGRAF Execution : SFC Algorithm

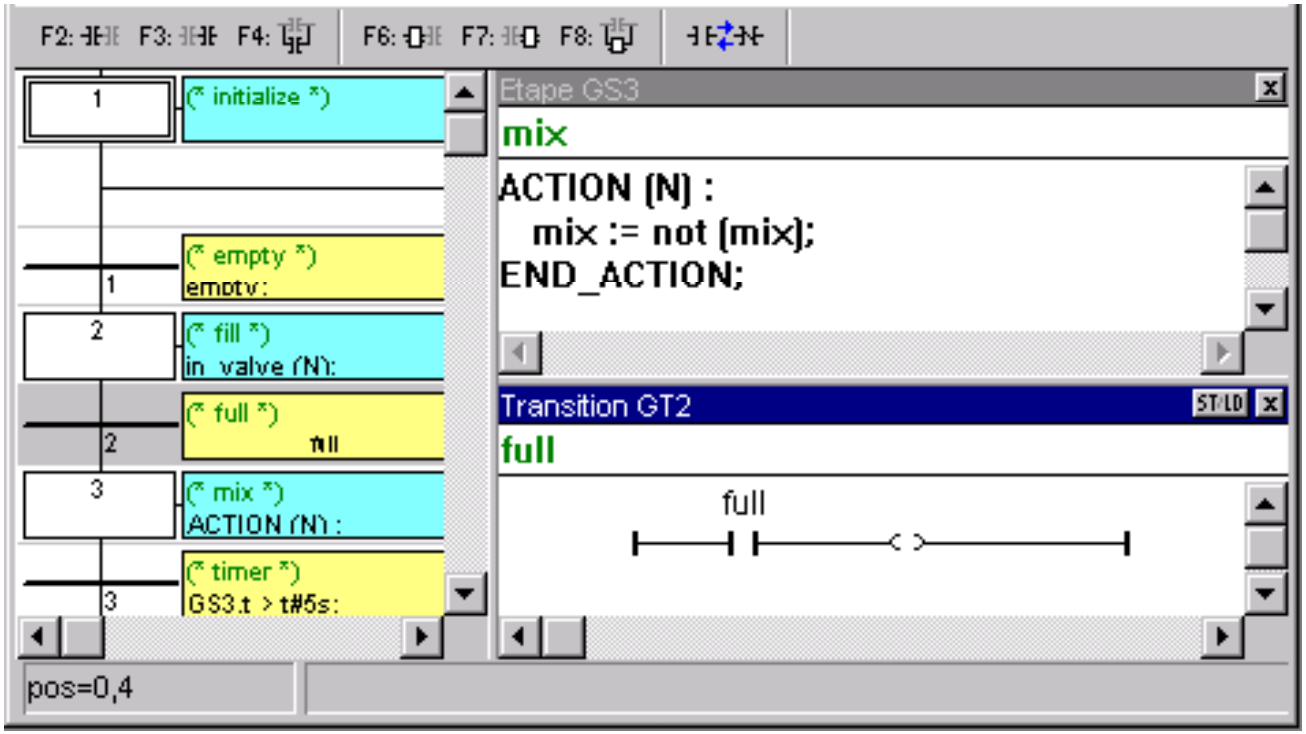
- **Последовательность выполнения программы можно разбить на следующие шаги:**
  - **Расчет действительных передач**
  - **Перемещение имеющих отношение SFC знаков**
  - **Выполнение действий, соответствующих новой ситуации:**
    - **Первое: N конечные действия (на BOO и CHILD), P0 действия на предписания**
    - **Второе: P, S, R и N начальные действия (на BOO и CHILD)**
    - **Третье: N действия на предписания**
- **Sequential programs execution can be divided into the following steps:**
  - **Calculate validated transitions**
  - **Move concerned SFC tokens**
  - **Execute actions corresponding to the new situation:**
    - **First: N ending actions (on BOO and CHILD),P0 actions on statements**
    - **Second: P, S, R and N beginning actions (on BOO and CHILD)**
    - **Third: N actions on statements**

ISaGRAF SFC Editor

ISaGRAF SFC Editor

- Действия : ST, IL
- Передача : ST, IL, LD

- Actions : ST, IL
- Transition : ST, IL, LD

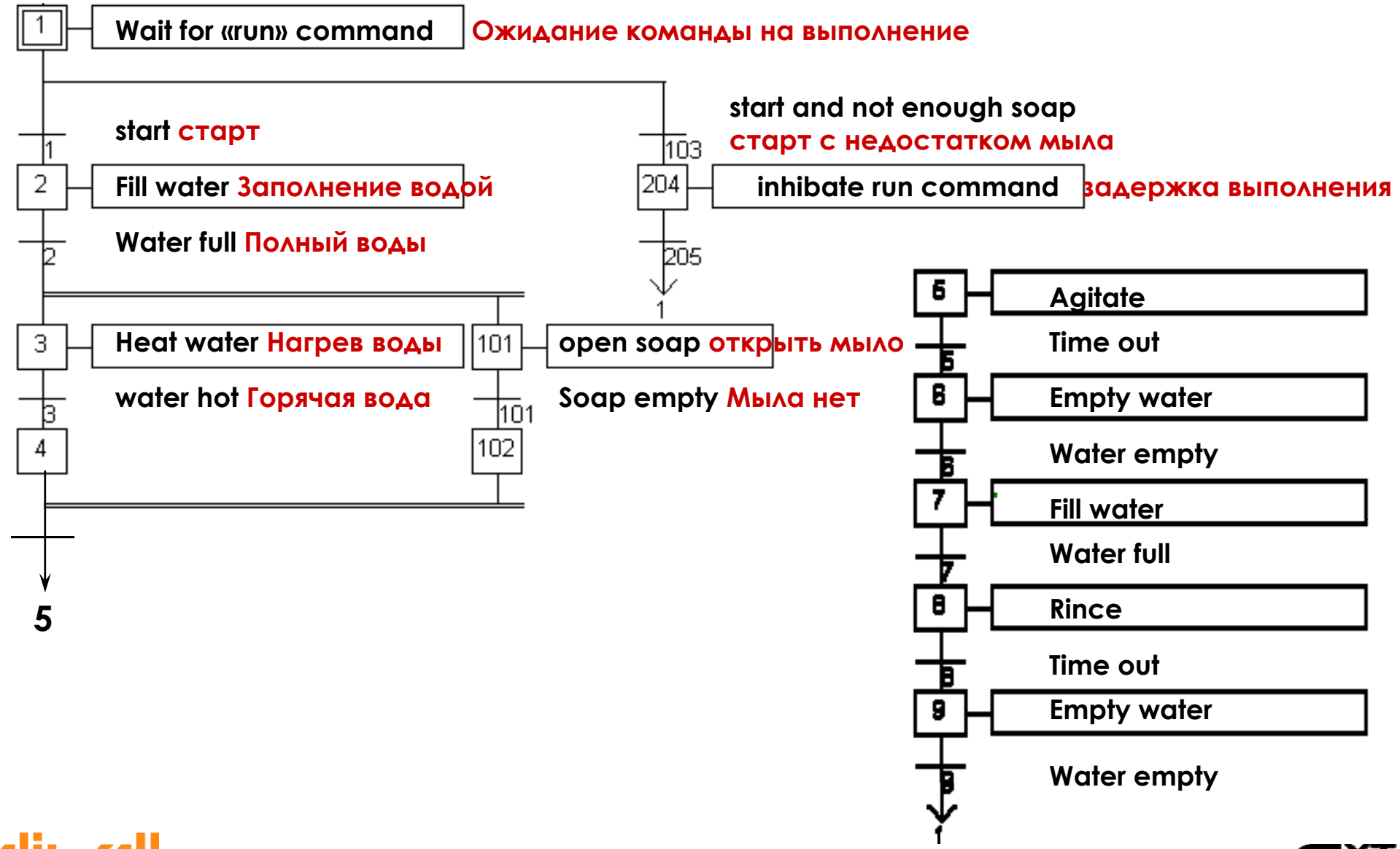


SFC Quick View (Быстрый просмотр)
SFC Quick View

- Диаграмма Состояния
- Язык определений
- Параллельные процессы (‘И ветви’ ИЛИ сыновние программы)
- Структура настоящих циклов
- Прямые спецификаторы для Логических переменных или Сыновних программ
- Прямые языки ST / IL /LD
- Редактор клавиатурный или мышкой

- State Diagram
- Definition language
- Parallel processing (‘And branches’ OR child programs)
- Natural cycling structure
- Direct qualifiers to control Boo var or Child programs
- Direct languages are ST / IL /LD
- Key board or mouse Editor

Пример : Последовательность процесса
Example : Process Sequence



Язык схемы протекания

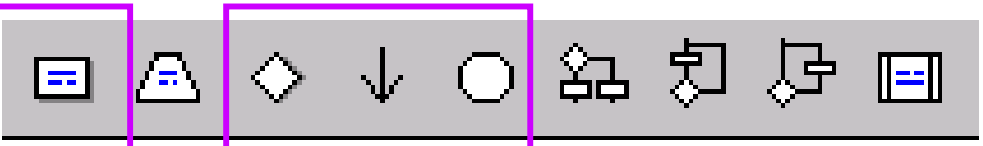
Flow Chart Language

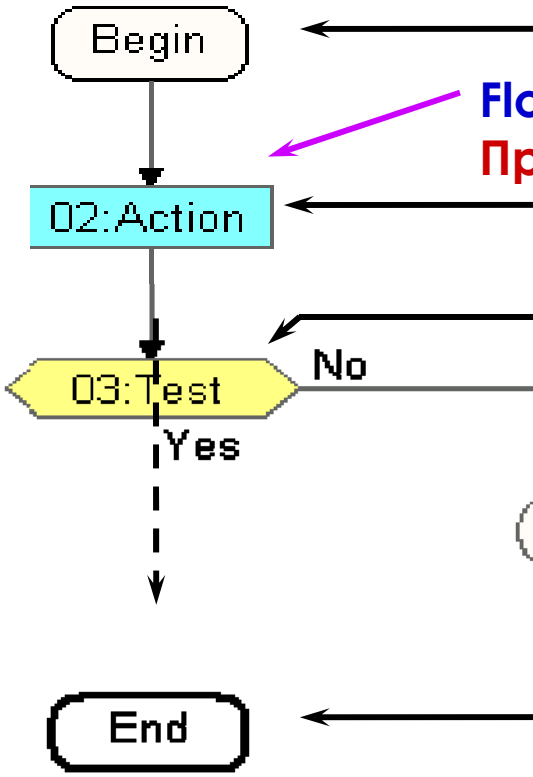
- *FC Язык*
- *FC Подпрограммы*
- *FC Редактор*

- *FC Language*
- *FC Sub Programs*
- *FC Editor*

FC Базовые компаненты
FC Basic Components

Elements
Элементы





One begin element Один начальный элемент

Action element Элемент Действия
-> Control statements Контроль состояния

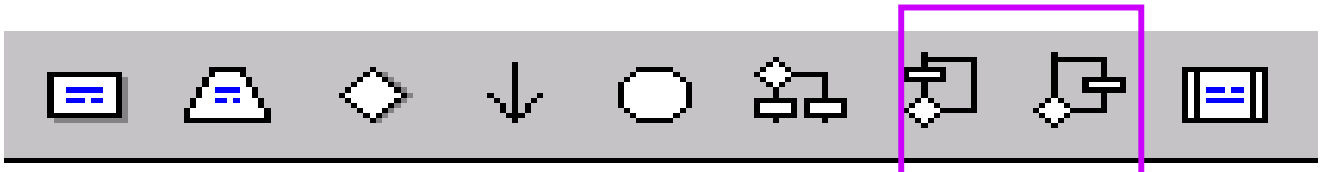
Decision element Элемент Решения
-> Evaluation Tests Тесты обработки

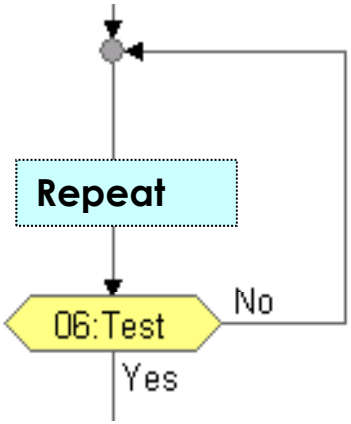
Connector element Элемент соединения
-> back jumps, forward jumps
-> Переход назад, переход вперед

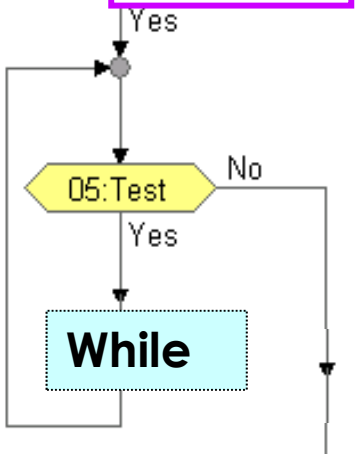
One end element Один конечный элемент

FC Компоненты сложной структуры
FC Complex Structure Components

Loops
Циклы







Действия протекают только вниз

Несколько блоков действий выполняется в одном цикле

-> Пока некий элемент не будет перевыполнен (действие или условие)

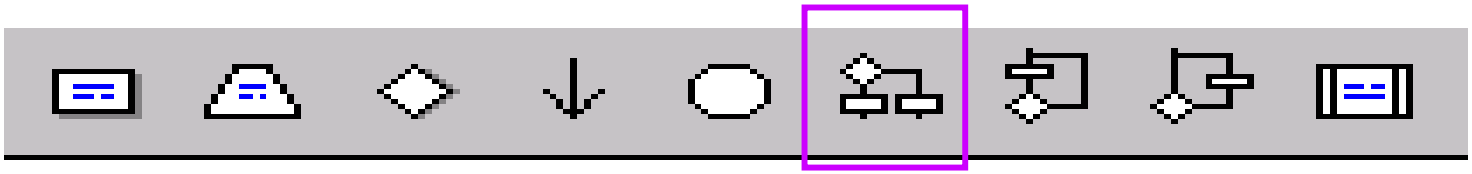
Action flow always goes down

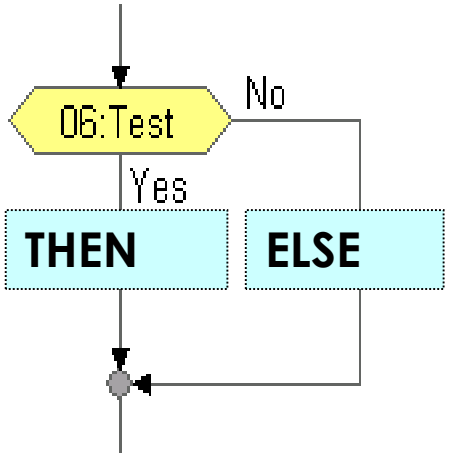
Several actions blocks are executed in the same cycle

-> Until same element is to be re-executed (action or condition)

FC Компоненты сложной структуры
FC Complex Structure Components

Condition
Условия



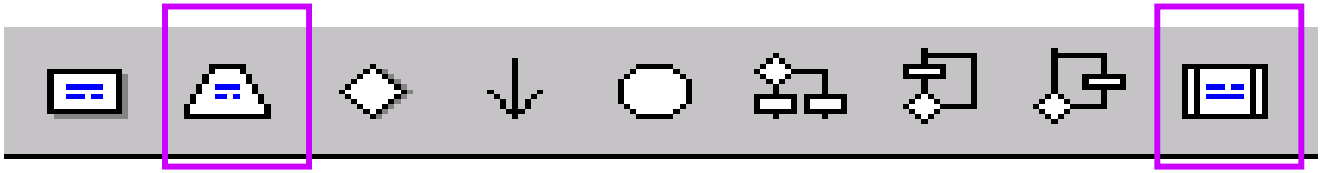


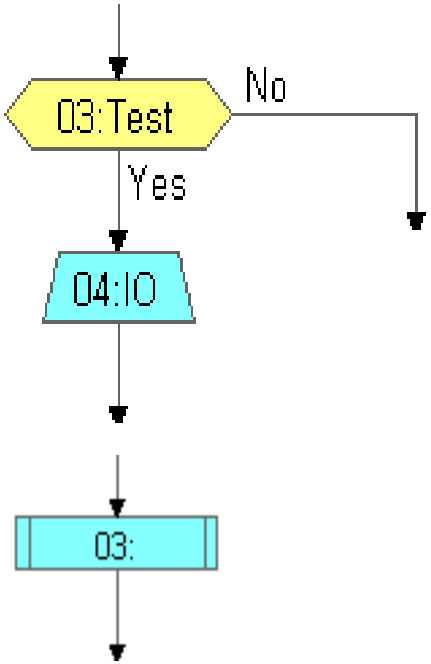
Выполняется ‘yes” ветвь если условие истинно
Actions on the ‘yes’ branch when condition is true

Выполняется ‘no” ветвь если условие не верно
Actions on the ‘no’ branch when condition is false

FC Компоненты сложной структуры
FC Complex Structure Components

Actions
Действия





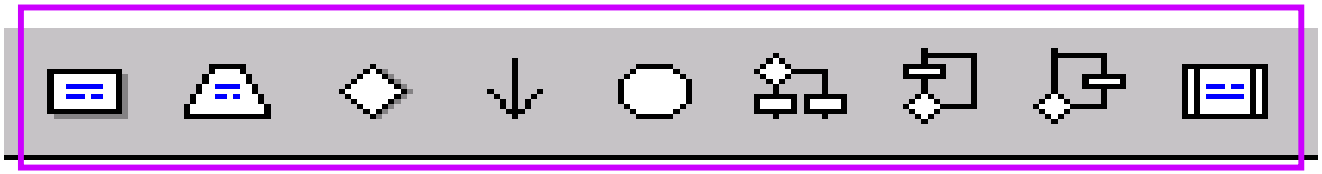
Специальные действия Вх./Вых: такие же как и обычные действия, но в блоке специальной формы для выделения среди помечаемых действий.

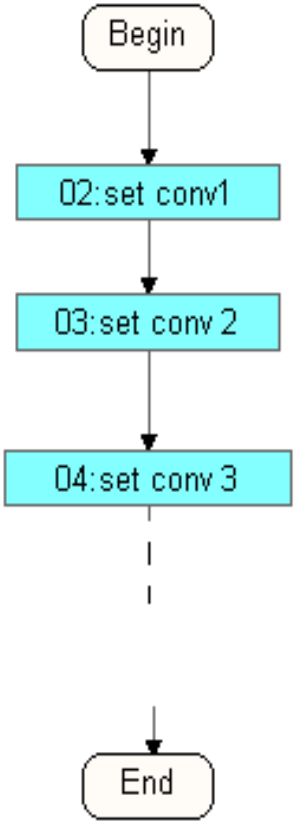
IO specific actions : same as normal action but with specific box to point out remarkable actions.

Подпрограммы: Лучшая архитектура проекта, повторно используемые коды.

Sub programs : Better architecture of project, reusable code.

Общие правила выполнения
General Execution Rules

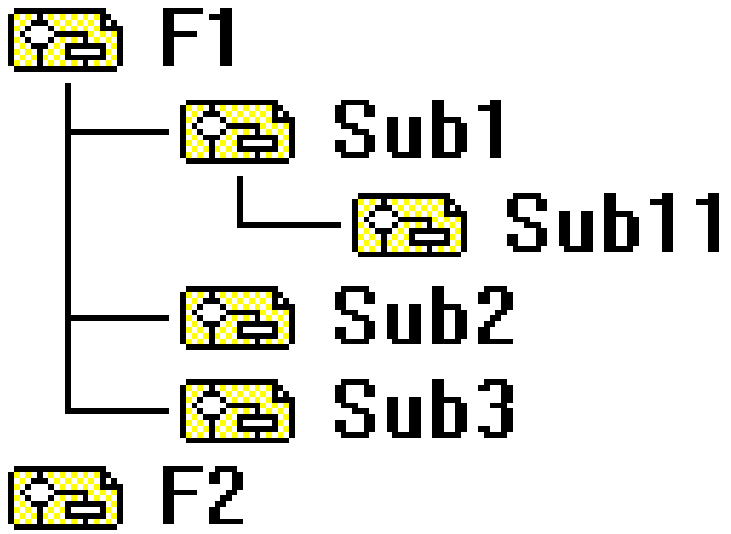




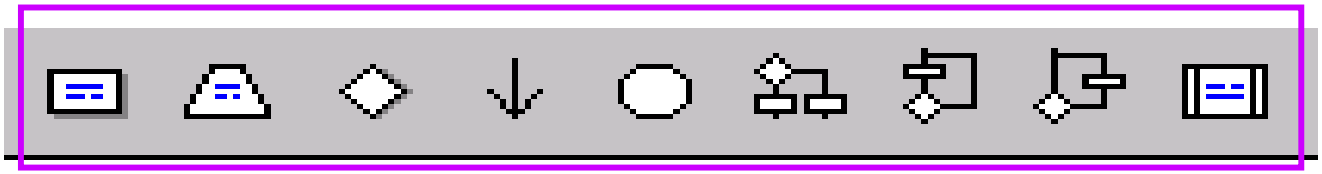
- **‘Begin/Начало’ занимает один цикл обработки**
  - на основной или под- программе FC
  - никаких действий состояния в начальном блоке
  - только в следующем цикле выполняются действия
- **Ряд последовательный действий выполняется в одном цикле пока НЕ**
  - достигнут уже выполненный элемент схемы
  - достигнута подпрограмма FC
  - достигнут ‘End/Конец’
- **‘End/Конец’ занимает один цикл обработки**
  - ничего не выполняется за чертой текущей программы

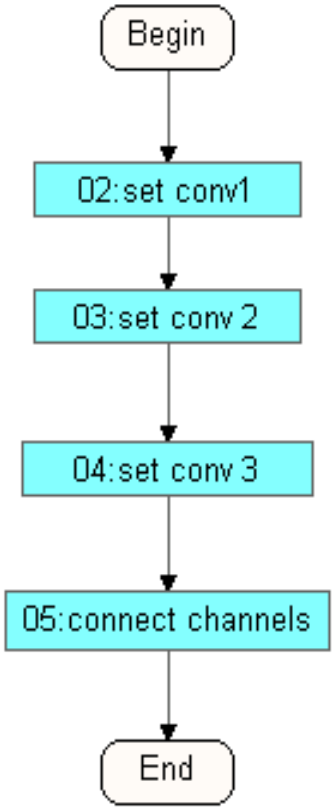
- **Begin always takes one cycle to be executed**
  - on main FC program OR FC sub program
  - no action statement in the begin Block
  - next cycle only will actions be executed
- **A set of consecutive actions is executed during one cycle according to the flow direction until**
  - already executed element of chart is reached
  - FC sub program item is reached
  - ‘End’ is reached
- **‘End’ always takes one cycle to be executed**
  - nothing is executed afterwards in the current program

FC Принципы Иерархии Программы
FC Hierarchic Program Principles



Правила выполнения подпрограммы
Sub Program Execution Rules





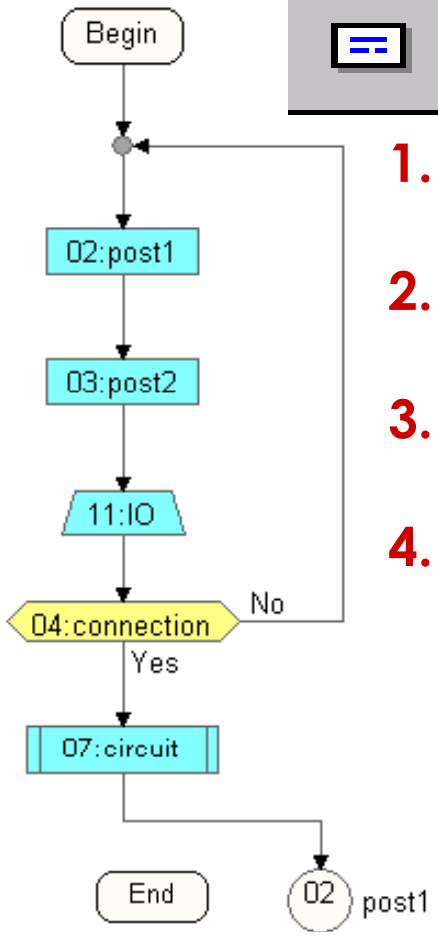
- **Во время выполнения подпрограммы**
  - Вызывающая программа останавливается
  - Подпрограмма не может иметь бесконечное число циклов
- **Конечный блок**
  - Протекание подпрограммы заканчивается на блоке ‘End’
  - Окончание занимает один цикл выполнения
  - После выполнения ‘End’ возобновляется вызывающая программа

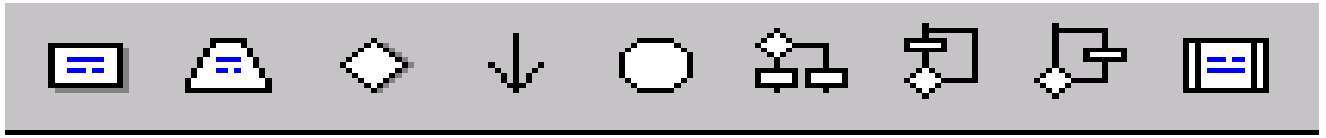
- **During sub program execution**
  - Calling program is suspended
  - The sub program cannot have an endless loop back
- **End Block**
  - Sub program flow always finish connected to ‘End’ object
  - End always takes one cycle to be executed
  - After ‘End’ is executed the calling program resumes execution

FC Редактор
FC Editor

Шаги построения Диаграммы выполнения

Flow Diagram building steps

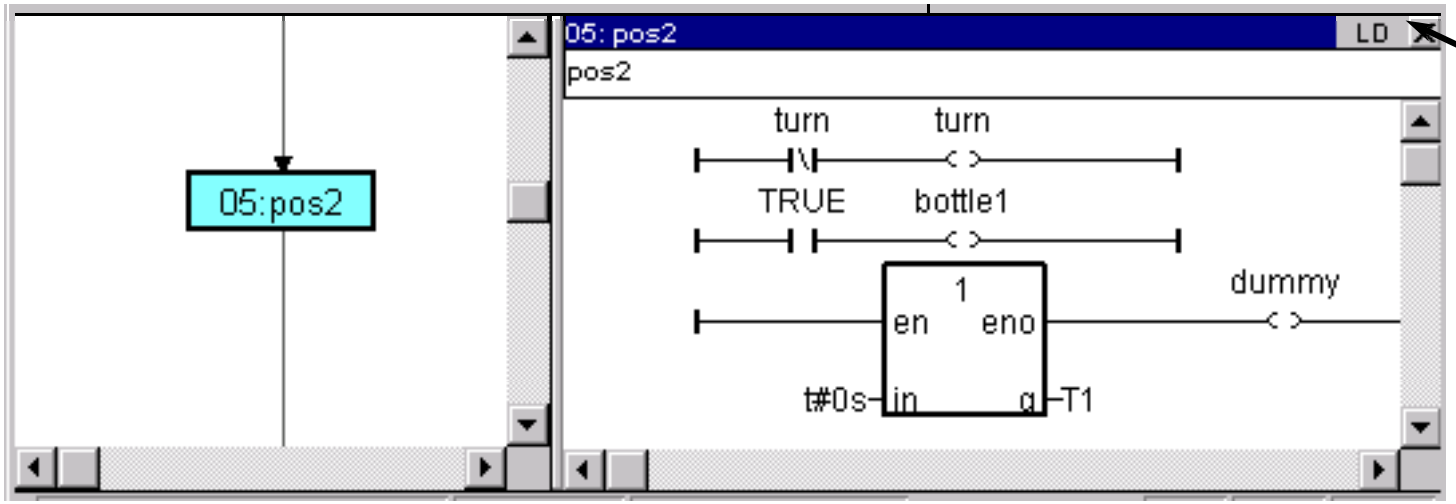


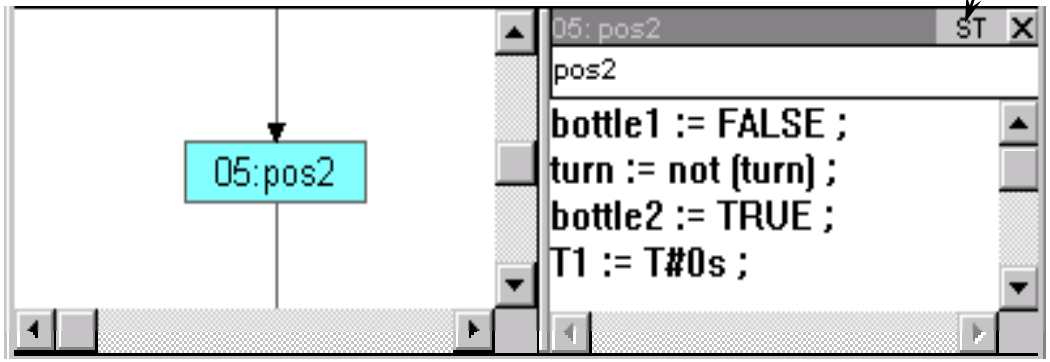


1. Расположите элементы выполнения на схеме
   1. Put elements of the flow chart in position
2. Установите соединения
   2. Build flow connections
3. Вставьте комментарии
   3. Enter comments
4. Введите коды
   4. Write code

FC Редактор
FC editor

Действия / Actions : ST, LD, IL





language Selection

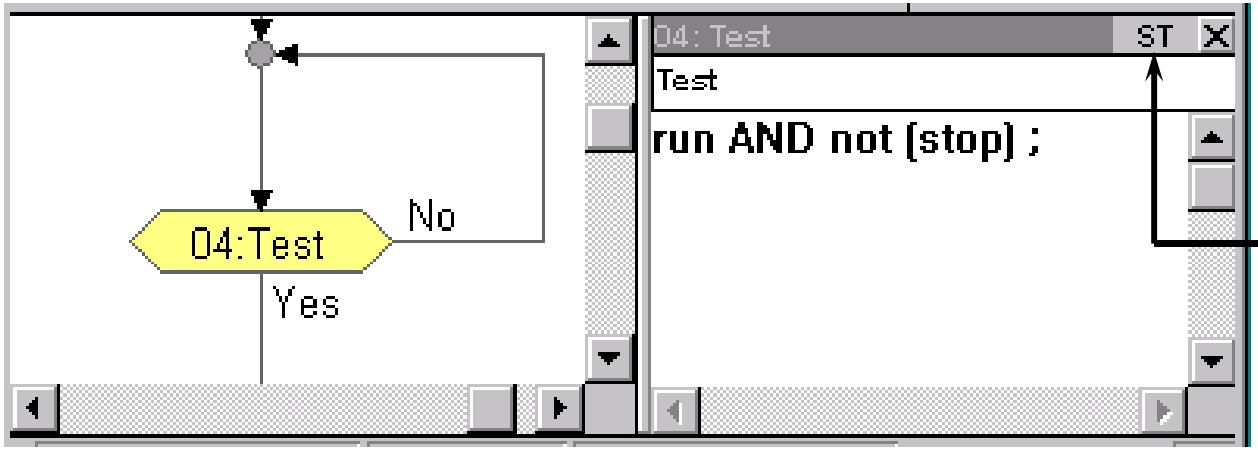
выбор языка

->before edition !

-> перед редактированием

FC Редактор
FC editor

Решения / Decisions: ST, LD, IL

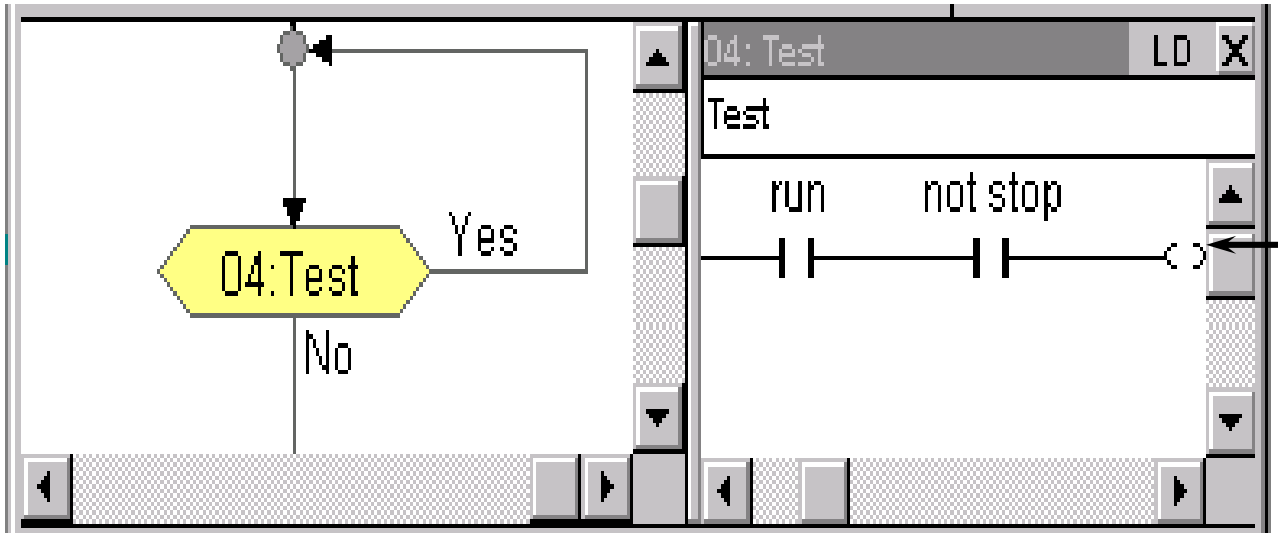


language Selection

выбор языка

->before edition !

-> перед редактирован.



Decision output

Выход решения

Quick View (Быстрый просмотр) для схем выполнения
Flow Chart Quick View

- **Диаграмма решения не является последовательной !**
- **В последовательной секции**

- **По верхушкам**
  - При использовании таймера всегда сбрасывайте его перед запуском
  - используйте блоки Вх/Вых для специальных действий
  - Избегайте слишком больших программ
  - Используйте подпрограммы (помните, что 2 цикла проходят до первого выполнения)
  - рисуйте несколько последовательных блоков действий для лучшей читаемости
  - используйте соединения чтобы не запутаться
  - Возможность изменения размеров блоков

- **Decision diagram not a sequential diagram !**
- **In sequential section**

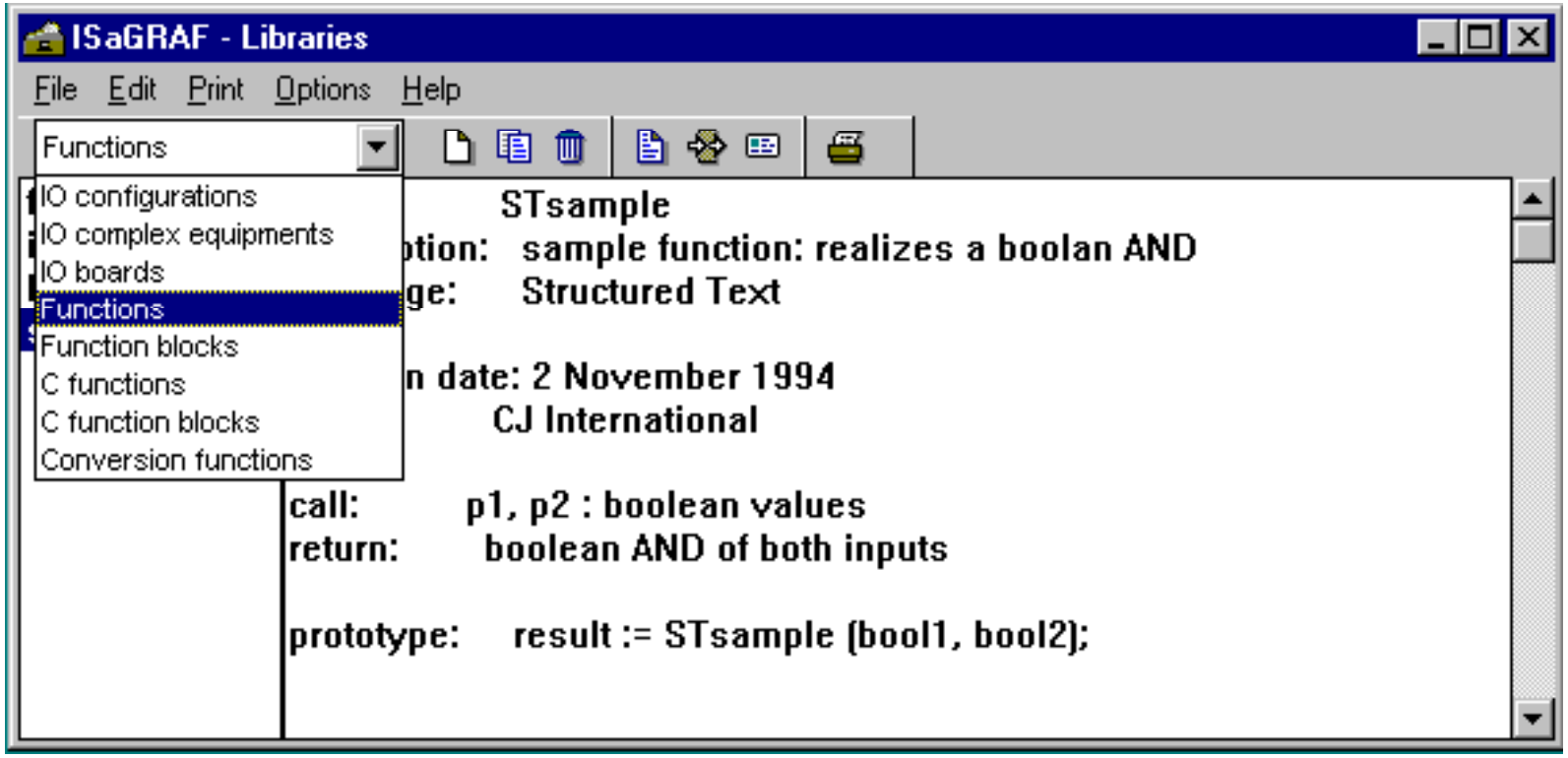
- **Tips**
  - When using a timer always reset it before launch
  - Use IO blocks for specific actions
  - Avoid too big diagrams
  - Use subprograms (beware 2 cycles will run before first execution)
  - draw several consecutive action blocks for better readability
  - use connectors to avoid cobwebs
  - Possibility to resize the blocks

ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ ЭЛЕМЕНТОВ

USE OF LIBRARY ELEMENTS

- *Программные элементы*
- *Элементы Устройств*
- *Software Elements*
- *Hardware Elements*

Различные объекты
Different Objects



- Программные объекты / объекты Оборудования
- Programming objects / Hardware objects

Программные объекты Библиотеки
Programming Objects of the Library

- **C функции**
  - Написаны на C : Требуют цели, зависящей от C компилятора
- **C функциональные блоки**
  - Написаны на C : Требуют цели, зависящей от C компилятора
  - Могут иметь много выходных параметров и обрабатываются соответствующим блоком библиотеки
- **C функции преобразования**
  - Написаны на C : Требуют цели, зависящей от C компилятора
- **Функции / Функциональные блоки**
  - Написаны на IEC языках
  - Импортные/Экспортные операции от/к выбору ФУНКЦИЙ

- **C functions**
  - Written in C : Need a target dependent C compiler
- **C function blocks**
  - Written in C : Need a target dependent C compiler
  - Can have many output parameters and are instantiated from a reference block in the library
- **C conversion functions**
  - Written in C : Need a target dependent C compiler
- **Functions / Function Blocks**
  - Written with IEC languages
  - Import/export operations from/to the FUNCTION section

IEC1131-3 Программные объекты
IEC1131-3 Programming Objects

- **Создание новой функции**
  - Задание входных параметров и тип возвращаемого значения
  - Задание значения возвращаемого параметра (название функции)

- **Creating a new function**
  - Define inputs parameters and return type
  - Assign value to the return parameter of the function (name of the function)

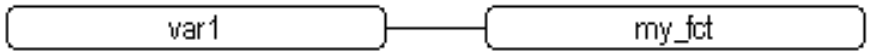
- **B/In ST**

```
-----------;
my_fct := var1 ;
```

- **B/In IL**

```
LD      boo1
ST      my_fct
```

- **B/In FB_**



- **Создание нового функционального блока**
  - Задание типа входных и выходных параметров
  - Задание внутренних переменных, которые будут использованы

- **Creating a new function block**
  - Define input & output parameters type
  - Define internal variables that will be instantiated

Вызов Функции
Calling a Function

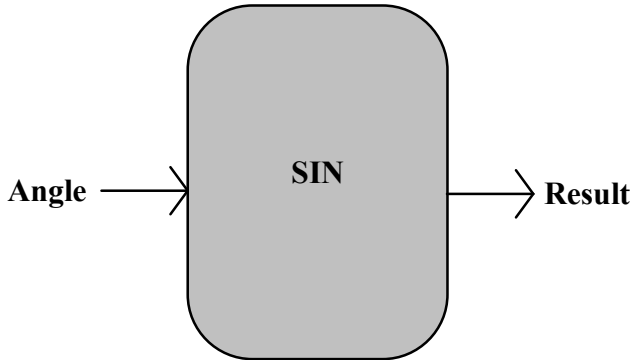
- **B/In ST**

```
result := sin (angle);
(* Same syntax for Spfct, Cfct *)
(*Синтаксис такой же как для Spfct, Cfct *)
```

- **B/In IL**

```
LD    v1
FCT   v2, ...
ST    result
```

- **B/In FBD**



Вызов функционального блока
Calling a Function Block

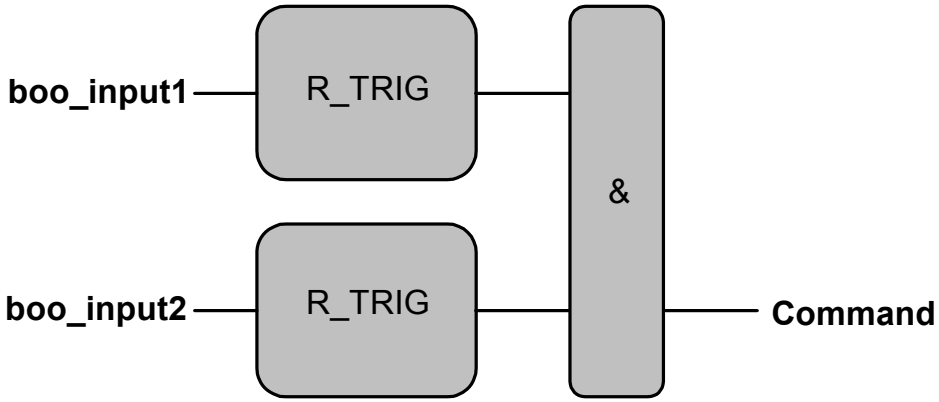
- B/In ST

```
trig_1 (boo1);
trig_2 (boo2);
command := trig_1.Q & trig_2.Q;
```

- B/In IL

```
LD      boo1
ST      trig_1.clk
CAL     trig_1
...
AND     trig_2.Q
ST      command
```

- B/In FBD



Объекты библиотеки оборудования
Hardware Objects of the Library

- **Платы Входов/Выходов**
  - Элементы для представления основных типов устройств ввода-вывода PLC
  - Соответствуют С функциям, интегрированным в целевую программу производителем оборудования
  - Поддерживают ТОЛЬКО один тип и одно направление
- **Сложное оборудование ввода/вывода**
  - Представляет собой основную плату эквивалентную набору отдельных плат ввода/вывода
  - Используется один слот разъема на стойку
- **Конфигурация Входов/Выходов**
  - Соответствует преустановленному перечню плат с параметрами и названиями каналов по умолчанию для включения в проект
  - Выбирается при создании нового проекта

- **I/O boards**
  - Elements made to represent certain types of I/O devices available in the PLC
  - Correspond to C functions, integrated into the target software by the hardware manufacturer
  - Only support one type and one direction
- **I/O complex equipment**
  - Represented as a main board equivalent to a set of single I/O boards
  - Only uses one slot in the I/O connection rack
- **I/O configurations**
  - Correspond to a predefined list of boards with default parameters values and channels names, to be included into a project
  - Chosen when a new project is created

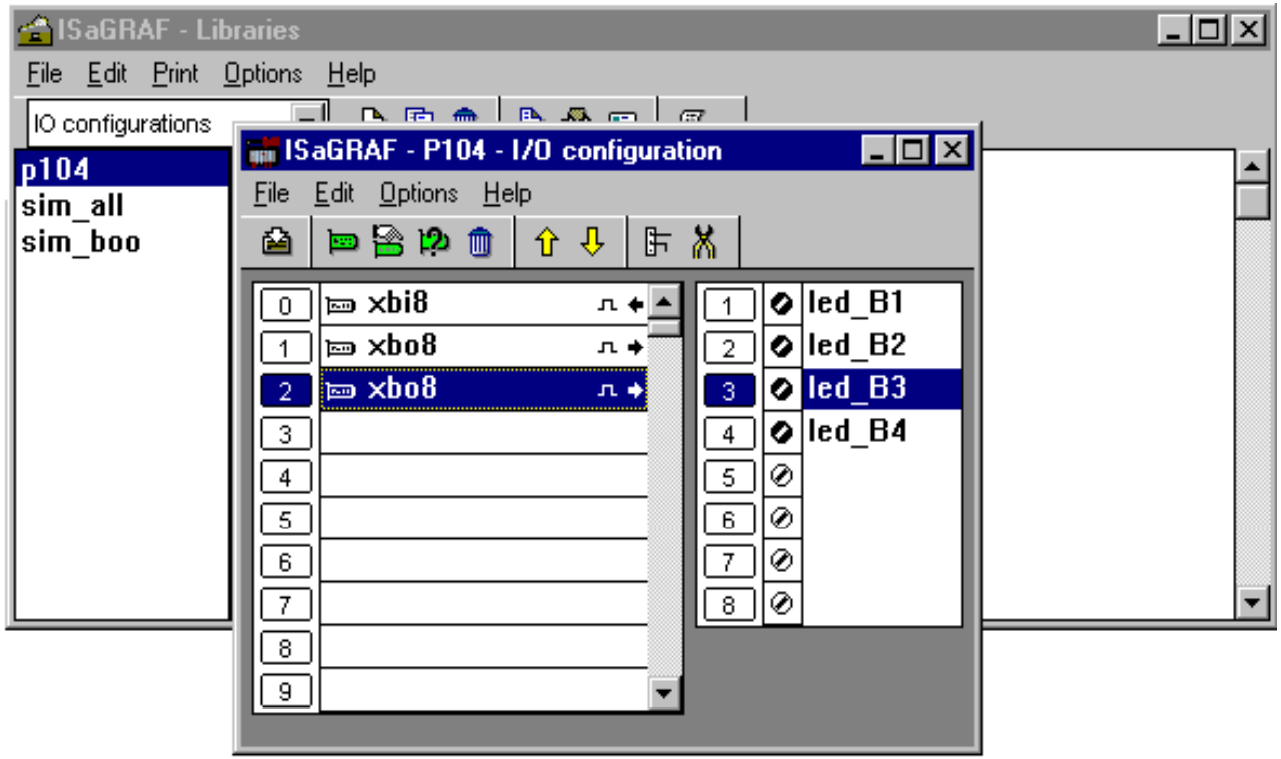
Использование устройств ввода/вывода

I/O Boards Use

- **В редакторе соединений**
- **Могут выбираться как платы так и сложные устройства**
- **Они задаются на Рабочем месте выделенном для реализации разработок**
- **Они задаются также и для целевой системы**

- **In the connection editor**
- **Both boards and complex equipment can be selected**
- **Their definition on the workbench is reserved to the OEM implementation**
- **Their definition on the target system also**

Конфигурирование Входов/Выходов
I/O Configuration Definition



- Редактор соединений вызывается из менеджера библиотеки
- Конфигурирование Входов/выходов выполняется при создании проекта

- The connection editor is called from the library manager
- The I/O configuration is selected at project creation

Поиск Функций/Функциональных блоков

Looking for a Function / Function Block

- **Из всех основных окон файл помощи, описаний**
  - Информация об ISaGRAF IEC61131-3 ‘C’ функциях
  - Информация об ISaGRAF IEC61131-3 ‘C’ функциональных блоках
- **Из меню помощи библиотеки**
  - Информация о новых пользовательских
    - » ‘C’ функциях, функциональных блоках и функциях преобразований
    - » IEC61131-3 функции и функциональные блоки
- **Из редактора FBD/LD**
  - Информация о всех функциях и функциональных блоках

- **From all main windows help file menu, reference**
  - Info on ISaGRAF IEC61131-3 ‘C’ functions
  - Info on ISaGRAF IEC61131-3 ‘C’ function block
- **From help menu, library**
  - Info on new customized
    - » ‘C’ functions, function blocks & conversion functions
    - » IEC61131-3 functions & function blocks
- **From FBD/LD editor**
  - Info on all functions & function blocks

Средства библиотеки ISaGRAF
*ISaGRAF **Library Tools***

- **‘C’ текстовый редактор**
- **Редактор Технических примечаний**
- **Средства генерирования кода**
- **Печать**
- **Управление Архивами**
- **Задание Паролей**

- **‘C’ text editor**
- **Technical notes Editor**
- **Code generation tools**
- **Printing**
- **Archive manager**
- **Password setting**

Библиотека Quick View (Быстрого просмотра)
Library Quick View

- **Программные элементы**
  - ‘C’ элементы, требуемые Вашим ‘C’ компилятором
  - IEC61131-3 элементы, импортируемые из проекта
- **Элементы оборудования**
  - Устройства ввода/вывода и сложные устройства, интегрируемые в целевой проект
  - Конфигурирование Входов/выходов используется для статического оборудования
- **Все библиотечные элементы могут использоваться в любом проекте**
- **Не изменяйте элементы библиотеки, которые уже используются**

- **Software elements**
  - ‘C’ elements require your own ‘C’ compiler
  - IEC61131-3 elements can be imported from project
- **Hardware elements**
  - IO boards and complex equipment need to be integrated on the target side
  - IO configurations are useful for static hardware
- **All library elements can be used from any project**
- **Do not modify a library element that has already been used**

Кто и Как выбирает ...
How and Why Choosing ...

- **SFC (Схема последовательных функций)**
  - Для последовательных действий
  - Может использоваться как язык спецификаций
  - Может представлять параллельные процессы
  - Включает механизмы синхронизации
  - Простые динамические правила
- **FC (Схема протекания)**
  - Диаграмма решения
- **LD (Диаграмма загрузки)**
  - Только для Логических уравнений
  - Простые правила
- **FBD (Диаграмма Функциональных блоков)**
  - Для уравнений смешанных типов
  - Большая библиотека блоков

- **SFC (Sequential Function Chart)**
  - For sequential operations
  - Can be used as a specification language
  - Can represent parallel processes
  - Includes synchronization mechanisms
  - Easy dynamic rules
- **FC (Flow Chart)**
  - Decision diagram
- **LD (Ladder Diagram)**
  - For boolean equations only
  - Easy rules
- **FBD (Function Block Diagram)**
  - For mixed type equations
  - Large library of blocks

Общий редактор
Common editor

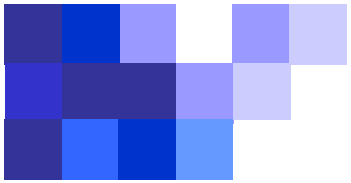
Кто и Как выбирает ...
How and Why Choosing ...

- **ST (Структурированный текст)**
  - Структурированный язык
  - Высокая читаемость исходных кодов
- **IL (Перечень инструкций)**
  - как SIEMENS (Шаг 5)
- **C**
  - Работает на переустановленном интерфейсе (Функции или Блоки)
  - Синхронный или Асинхронный (доступ к ОС)
  - Описание принтера
  - Требуемые не ISaGRAF средства

- **ST (Structured Text)**
  - Structured language
  - High readability of source code
- **IL (Instruction List)**
  - SIEMENS like (Step 5)
- **C**
  - Works on predefined interfaces (Functions or Blocks)
  - Synchronous or asynchronous (access to the OS)
  - Notion of pointer
  - Requires non-ISaGRAF tools

Больше об ISaGRAF

More on ISaGRAF



Установка

Installation

- **Три рабочих места**
  - Отличаются подключаемым к параллельному порту ключом
- **Three workbenches**
  - Hardware key to connect on the parallel port makes the difference
- **В Системе Windows System, добавить**
  - NT = 1 в секции [WS001] (для открытия соединений)
  - В системе Windows NT, предыдущая операция выполняется автоматически во время инсталляции.
- **On Windows System, add**
  - NT = 1 in [WS001] section (to allow communication)
  - On Windows NT system, the line above is automatically set by the install procedure.

Структура Рабочего места
Workbench Architecture

- **APL** → ISaGRAF **проекты**
- → ISaGRAF **projects**

- **SMP** → ISaGRAF **примеры проектов**
- → ISaGRAF **sample projects**

- **ARK** → ISaGRAF **архивы проектов**
- → ISaGRAF **archive projects**

- **COM** → **Общие объекты всех проектов**
- → **Objects Common to all projects**

- **EXE** → ISaGRAF **программа и файл помощи**
- → ISaGRAF **programs & help file**

- **LIB** → **Элементы библиотеки/ Элементы интерфейса /Технические замечания**
- → **Library elements/ Elements interface /technical notes**

- **TMP** → **Генератор кодов временных файлов**
- → **For temporary files code generator**

Архитектура целевой программы
Target Architecture

- **CMDS** → Выполняемые файлы ядра
  → Kernel executable files
- **DEFS** → Файлы, заданные в оглавлении
  → Header definition files
- **LIB** → Библиотеки
  → Libraries
- **RELS** → Перемещаемые файлы
  → Relocatable files
- **USER** → Исходные и заглавные файлы для использования ‘ C ’ функций, функциональных блоков и функций преобразования.
  → Source & header file for user ‘ C ’ functions, function blocks and conversion functions.

Архивирование
Archiving

- Вы можете архивировать
  - проекты / элементы библиотеки / определения
- Типы файлов
  - .pia проекты
  - .bia устройства ввода/вывода
  - .iia IEC61131-3 функции
  - .aia IEC61131-3 функциональные блоки
  - .uia ‘C’ функциональные блоки
  - .fia ‘C’ функции
  - .cia ‘C’ функции преобразования
  - .ria Конфигурация входов/выходов
  - .xia Сложные устройства

- You can archive
  - projects / library elements / definitions
- Filesare
  - .pia projects
  - .bia IO boards
  - .iia IEC61131-3 functions
  - .aia IEC61131-3 function blocks
  - .uia ‘C’ Functions blocks
  - .fia ‘C’ Functions
  - .cia ‘C’ Conversion functions
  - .ria IO Configurations
  - .xia Complex equipments

Защита через Пароль
Password Protection

- **Глобальная защита (определение доступа)**
- **Защита Только чтение (Изменения не допускаются)**
- **Уровни**
  - **Высший ‘0’**
  - **Низший ‘15’**
- **Для безопасности : Средства мониторинга**

- **Global protection (access denied)**
- **Read Only protection (no modification allowed)**
- **Levels**
  - **Highest is ‘0’**
  - **Lowest is ‘15’**
- **For security : monitoring tool**

Подключение к цели (продукту)
Target Connection

- Сообщение ‘Disconnected / Рассоединено’ перед загрузкой
  - Остановка или нет задачи в режиме ожидания
  - Кабель задачи ‘out / отключен’
  - Не корректные параметры связи
  - Задача ISaGRAF плохо проинсталлирована
- Сообщение ‘Disconnected / Рассоединено’ после загрузки
  - Системная ошибка
  - Конфигурирование входов/выходов
- Сообщение ‘No application / Нет применения’
  - Применение остановлено
  - Не загружено

- ‘Disconnected’ message before download
  - Target stopped or not in standby mode
  - Target cable ‘out’
  - Communication parameters not suitable
  - ISaGRAF target badly installed
- ‘Disconnected’ message after download
  - System error
  - IO configuration
- ‘No application’ message
  - Application stopped
  - Not downloaded

Таблица Символов
Symbol Table

- **Генератор кодов генерирует виртуальные адреса**
  - **таблицы символов**
  - **isawin\apl\project_name\appli.tst**

- **Может загружаться в цель (продукт)**
  - **Для обмена данными**

- **Code generator generates virtual addresses**
  - **Symbol table**
  - **isawin\apl\project_name\appli.tst**

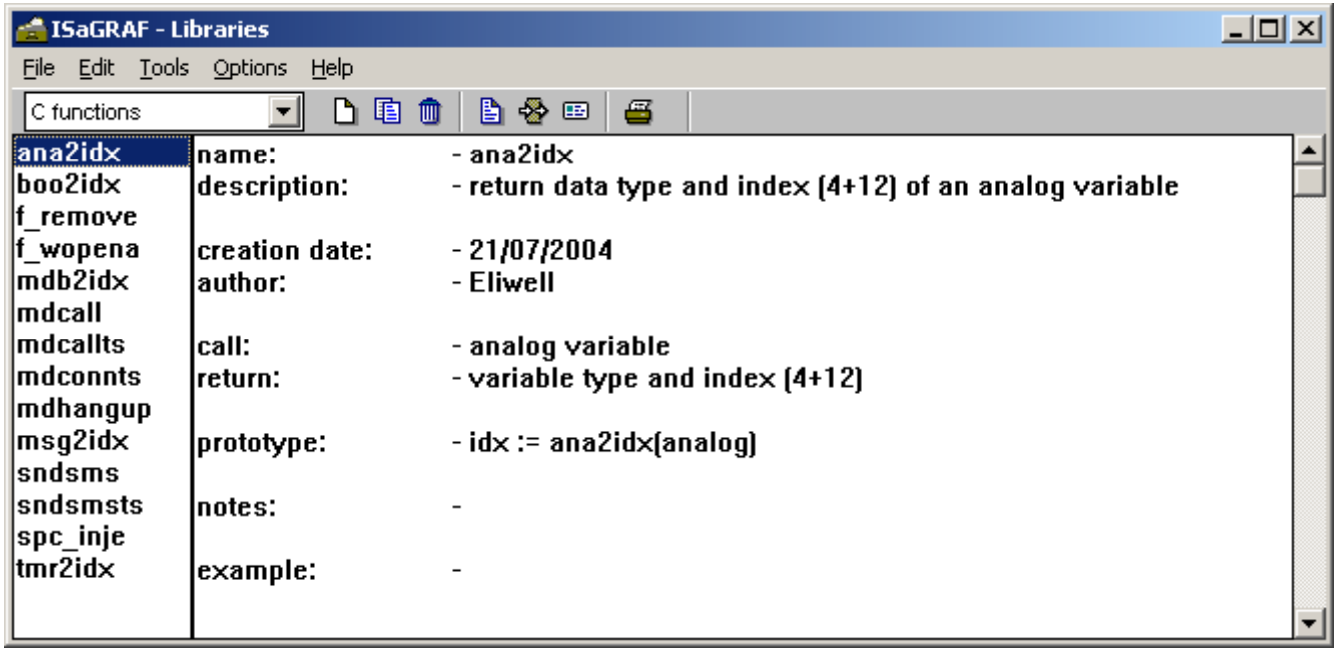
- **Can be downloaded to target**
  - **For data exchange**

ЭЛЕМЕНТЫ БИБЛИОТЕКИ XTPRO

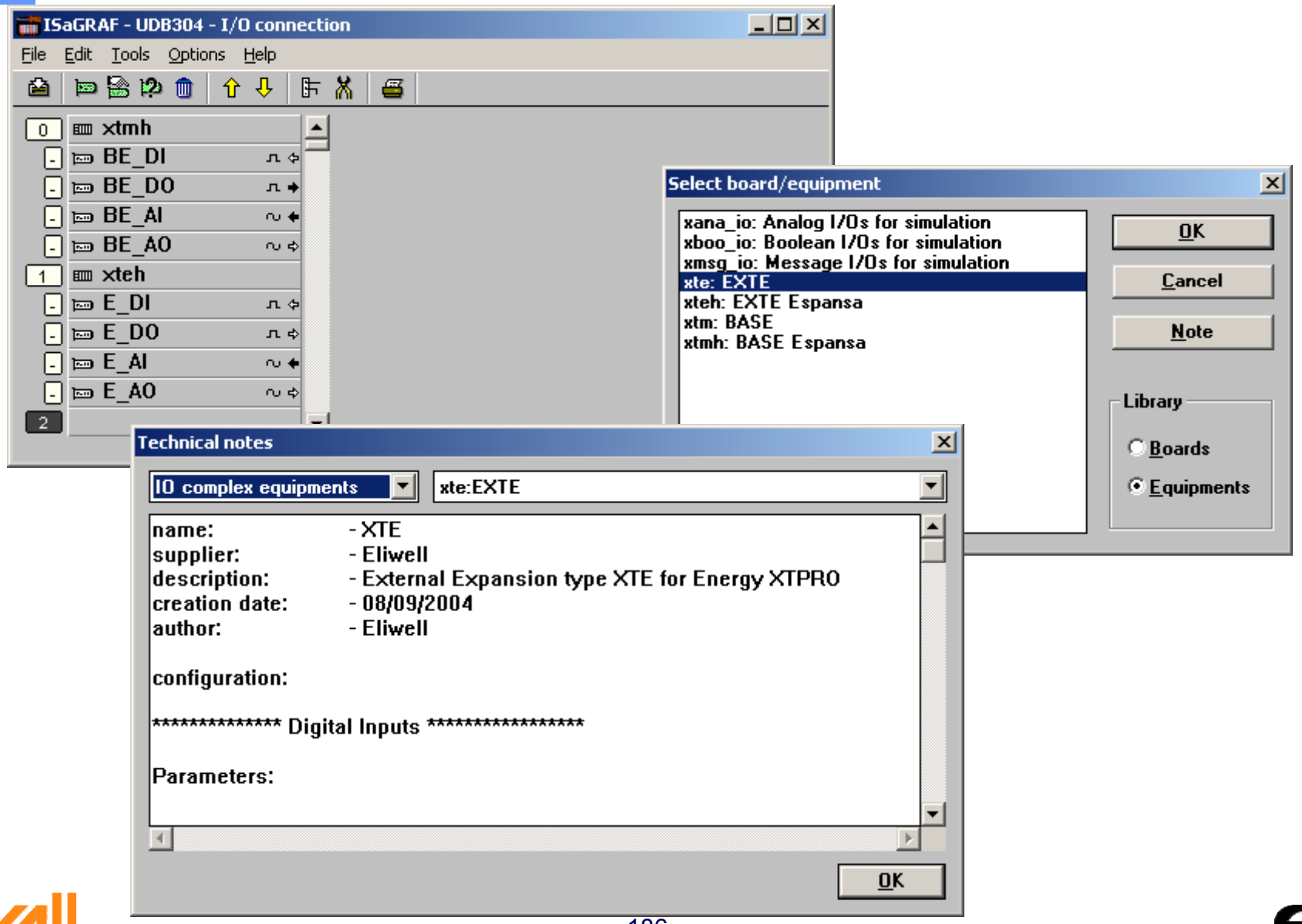
XTPRO LIBRARY ELEMENTS

- *Программные элементы*
- *Элементы оборудования*
- *Software Elements*
- *Hardware Elements*

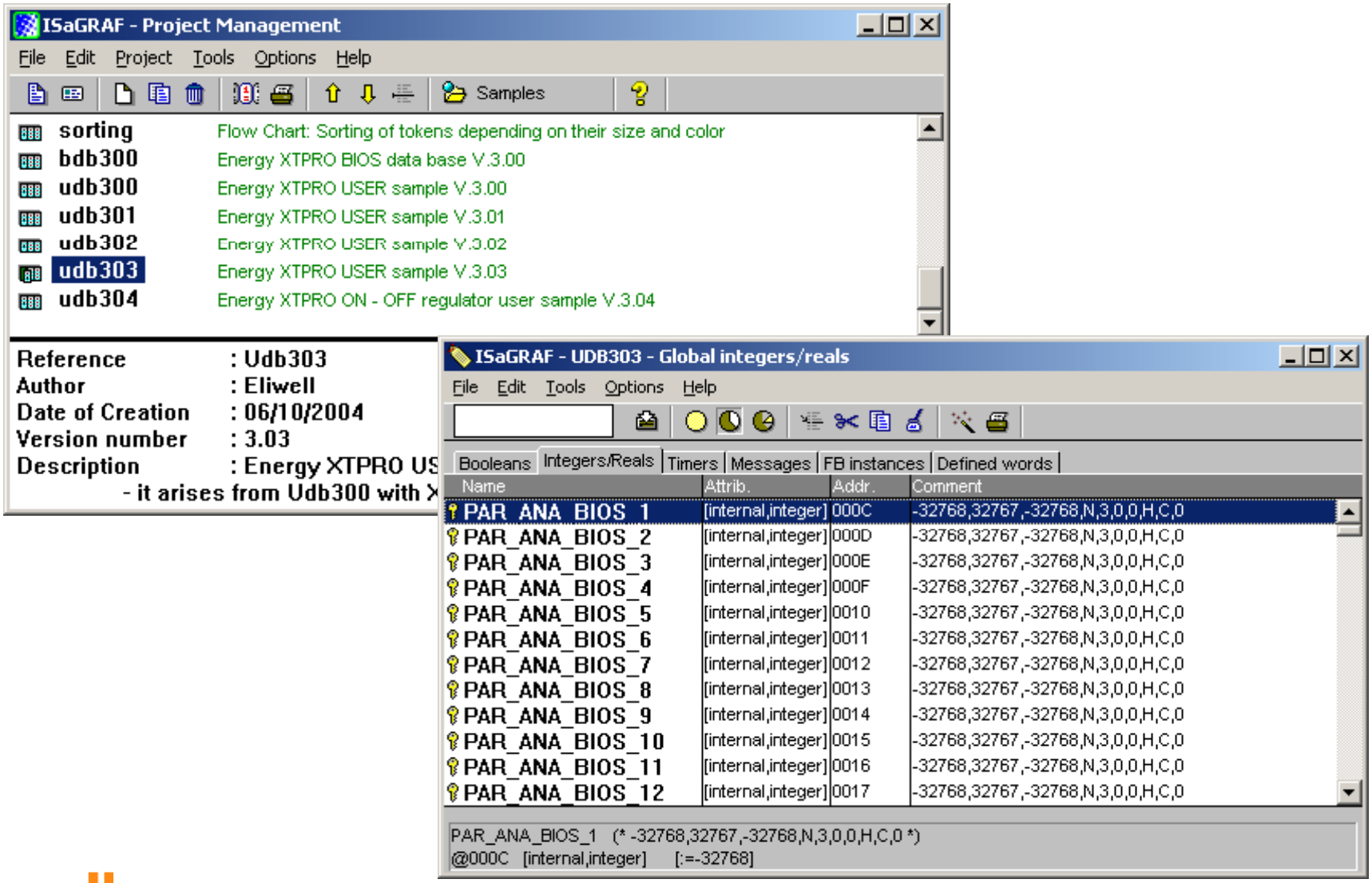
C Функции
C functions



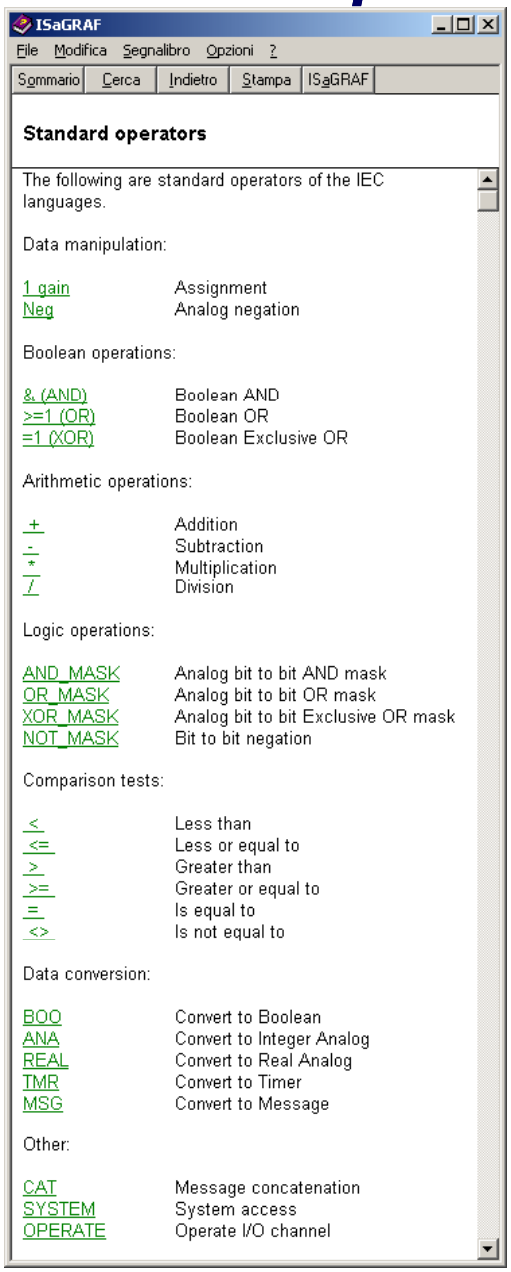
Комплексные устройства ввода/вывода
IO complex equipments

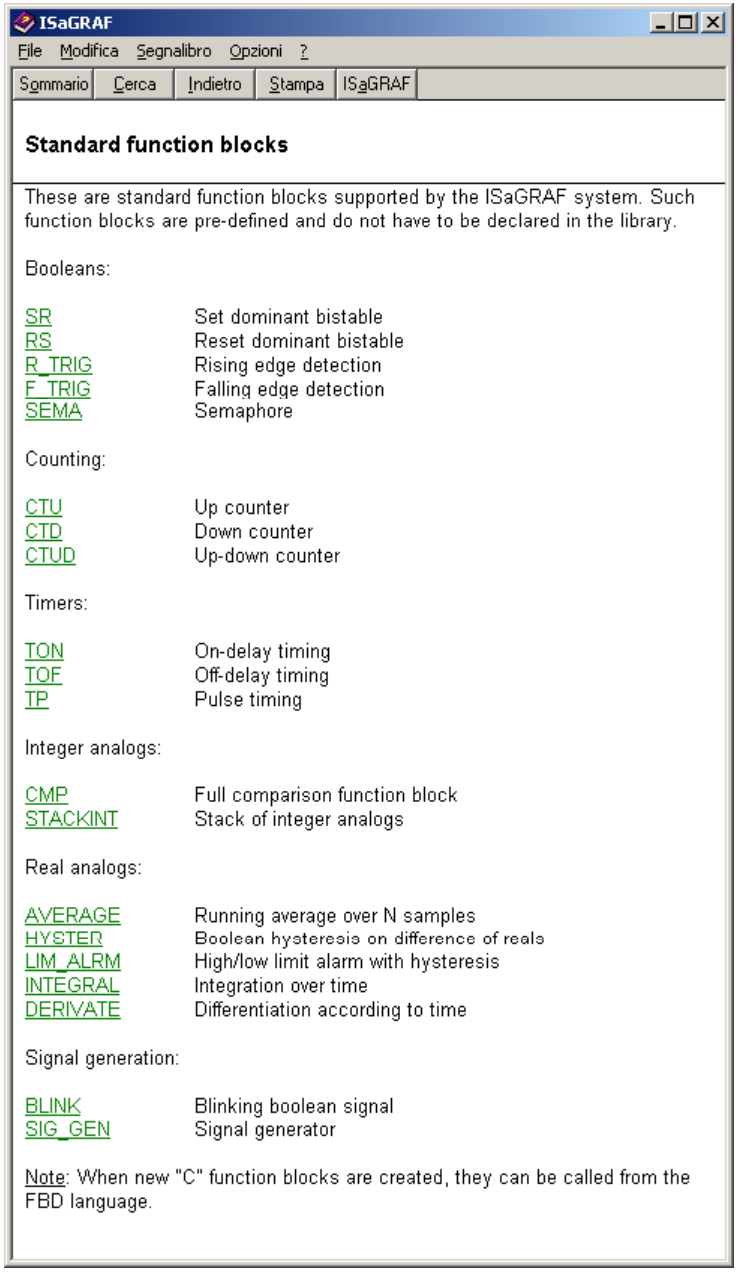


Пустые проекты и база данных BIOS
Empty projects and BIOS database

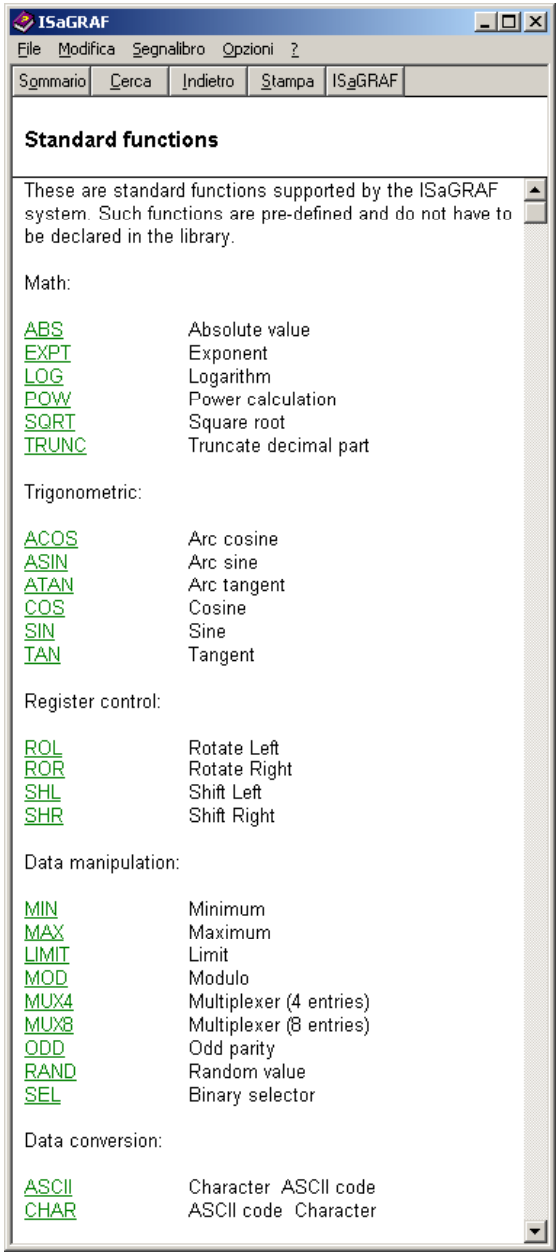


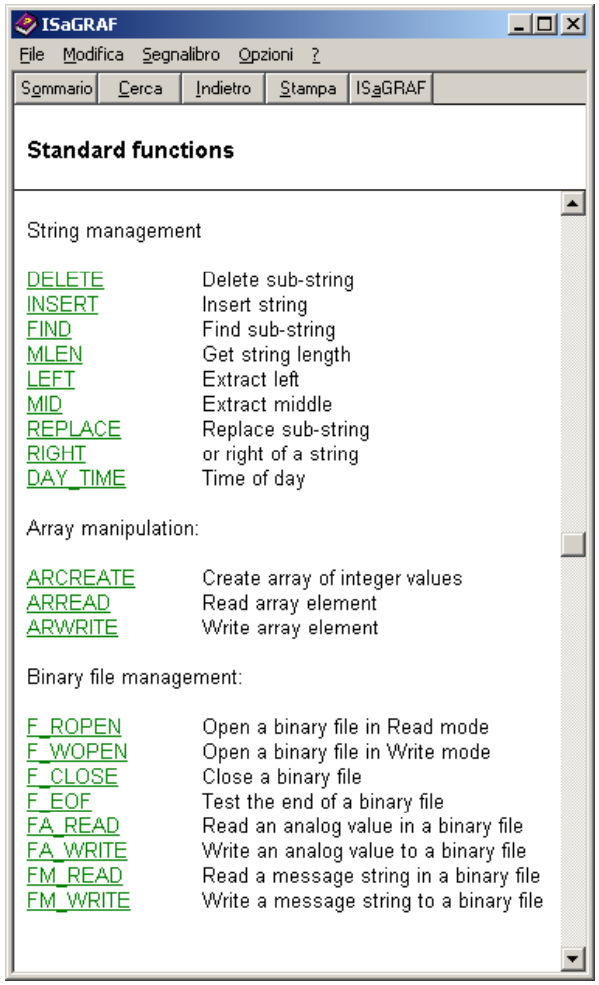
Стандартные операторы и функциональные блоки
Standard operator and function blocks





Стандартные функции
Standard functions





Стандартные функции
Standard functions

ISaGRAF

File Modifica Segnalibro ?

Sommario | Cerca | Precedente | Cronologia | ISaGRAF

LIMIT

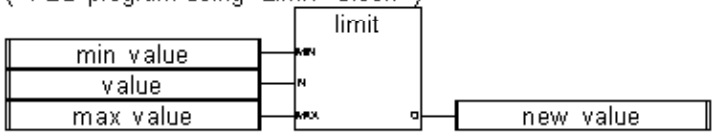
Arguments:

MIN	INT	minimum allowed value
IN	INT	any signed integer analog value
MAX	INT	maximum allowed value
Q	INT	input value bounded to allowed range

Description:

Limits an integer value into a given interval. Whether it keeps its value if it is between minimum and maximum, or it is changed to maximum if it is above, or it is changed to minimum if it is below.

(* FBD program using "LIMIT" block *)



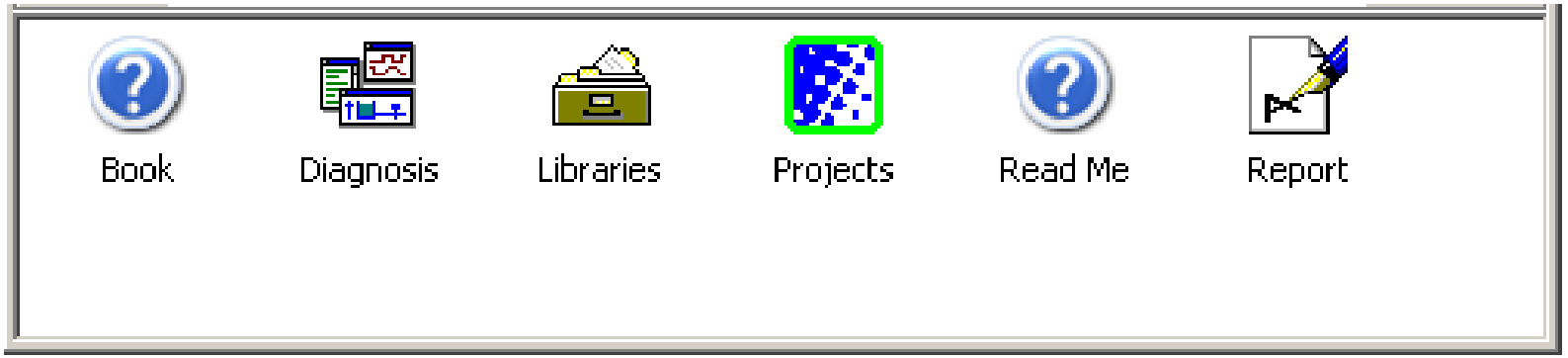

```
(* ST Equivalence: *)
new_value := LIMIT (min_value, value, max_value);
(* bounds the value to the [min_value..max_value] set *)

(* IL Equivalence: *)
LD          min_value
LIMIT       value, max_value
ST          new_value
```

ISaGRAF 3.5x

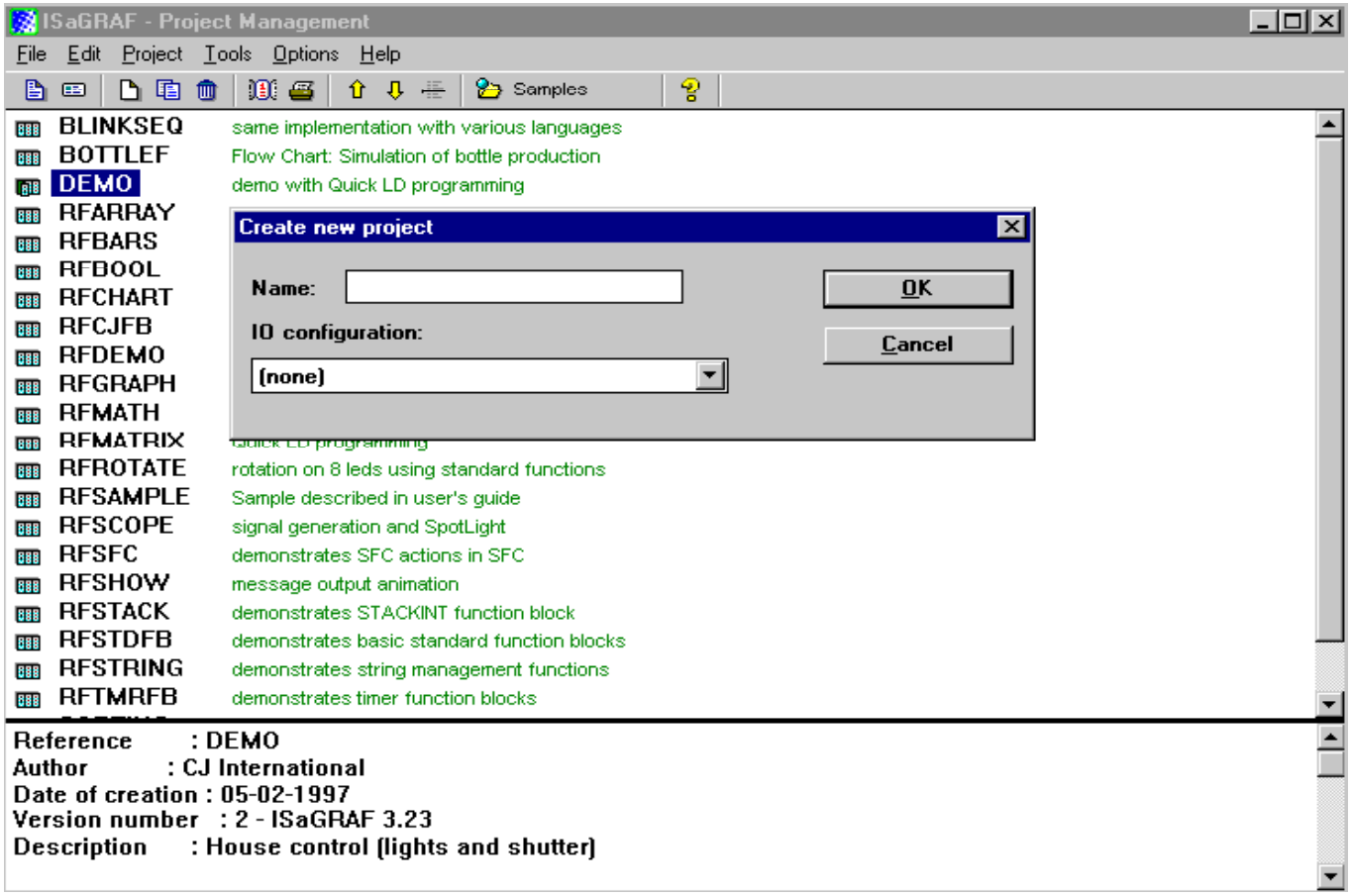
- *Упражнение*
- *Exercise*

Инструментарий Рабочего места ISaGRAF
ISaGRAF Workbench Tool

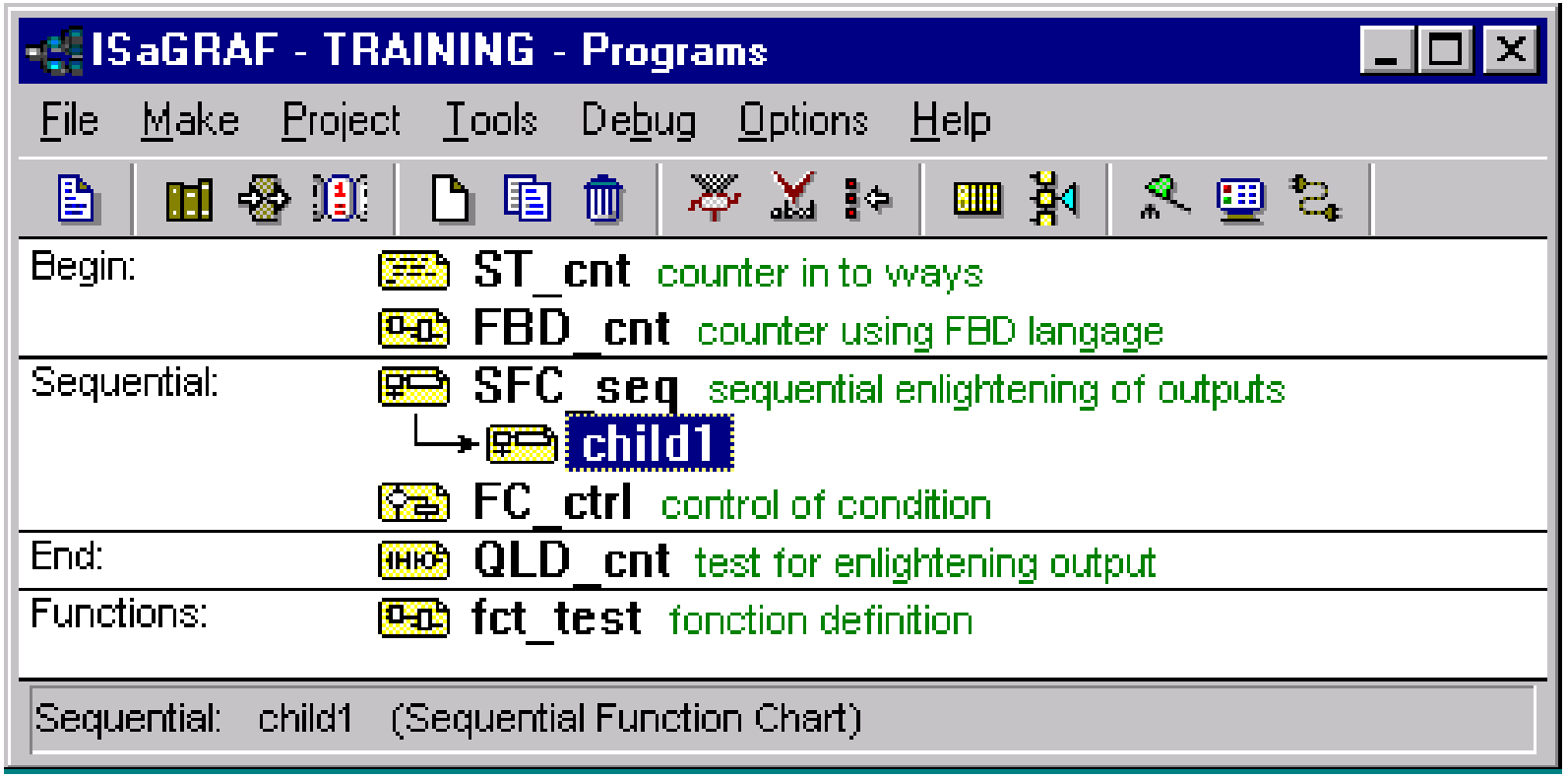


- Иконки / Icons are
  - **Projects:** **Управление проектом / Project management**
  - **Diagnosis:** **Средства пользователей сайта / Tool for on site users**
  - **Libraries:** **Управление библиотекой / Library management**
  - **Book:** **Контекстное руководство / On-line ISaGRAF manual**
  - **Readme:** **Информация о версии 3.51 и предыдущих / Information on v 3.51 and previous**
  - **Report:** **Пожелания по развитию / фиксация недостатков Wish of evolution / bug registration**

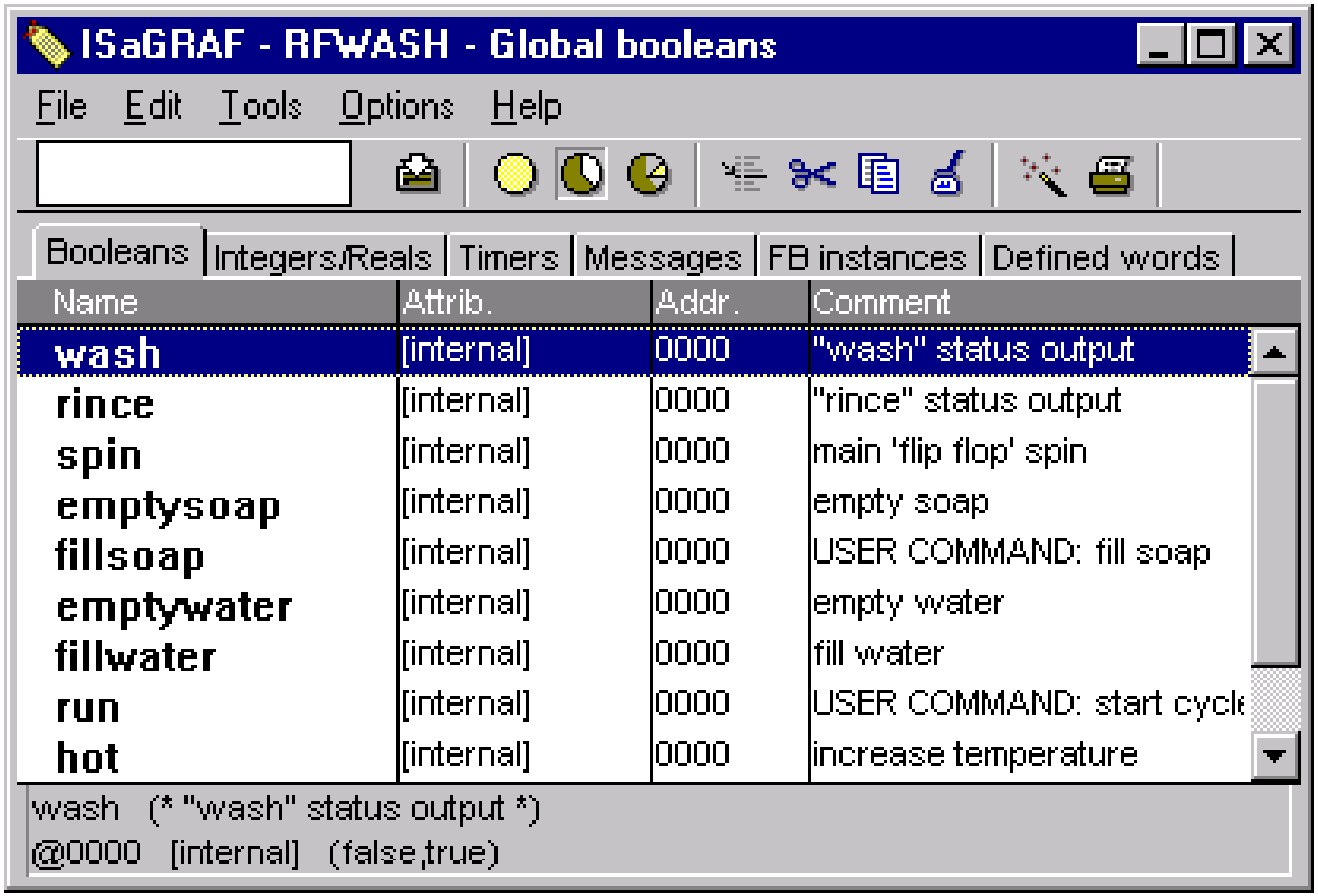
Окно Управления проектом
Project Management Window



Окно Управления проектом
Project Management Window



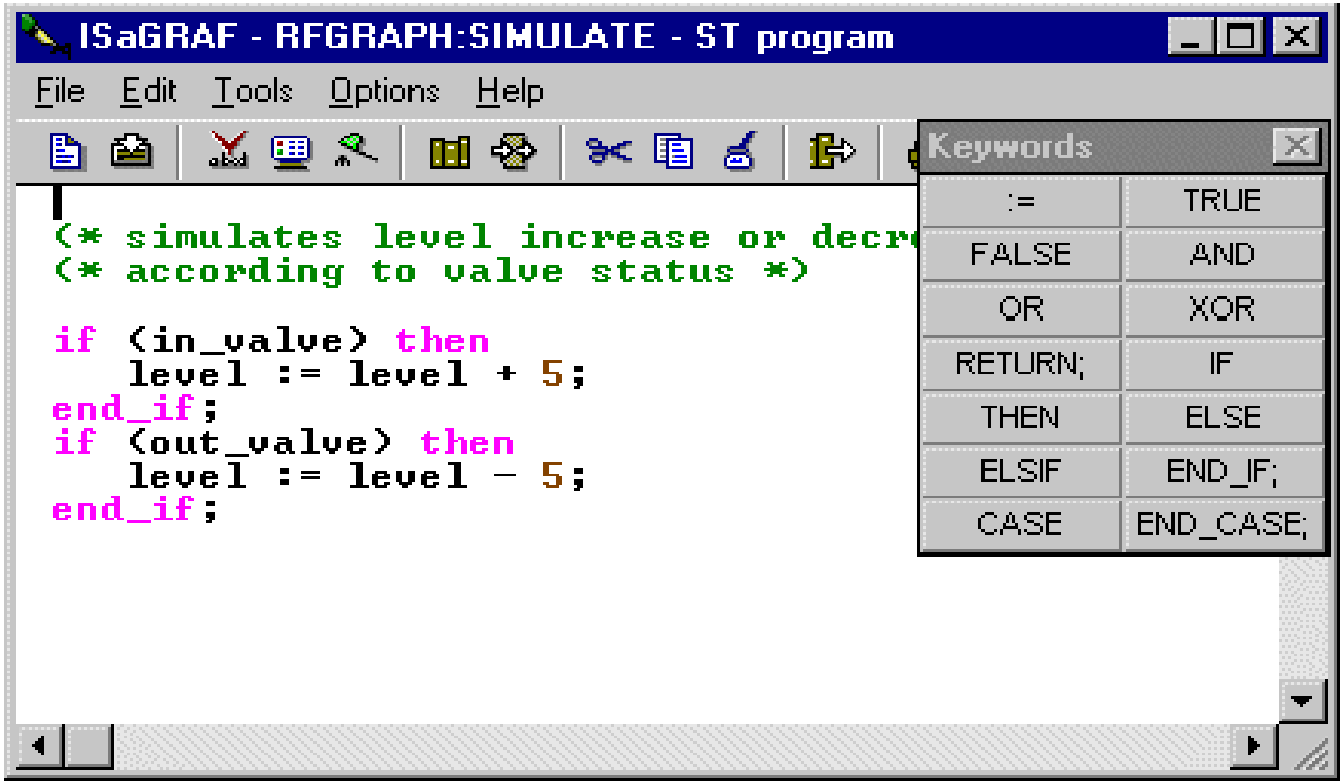
Словарь
Dictionary



- Двойной щелчок на переменной для перехода на второй уровень

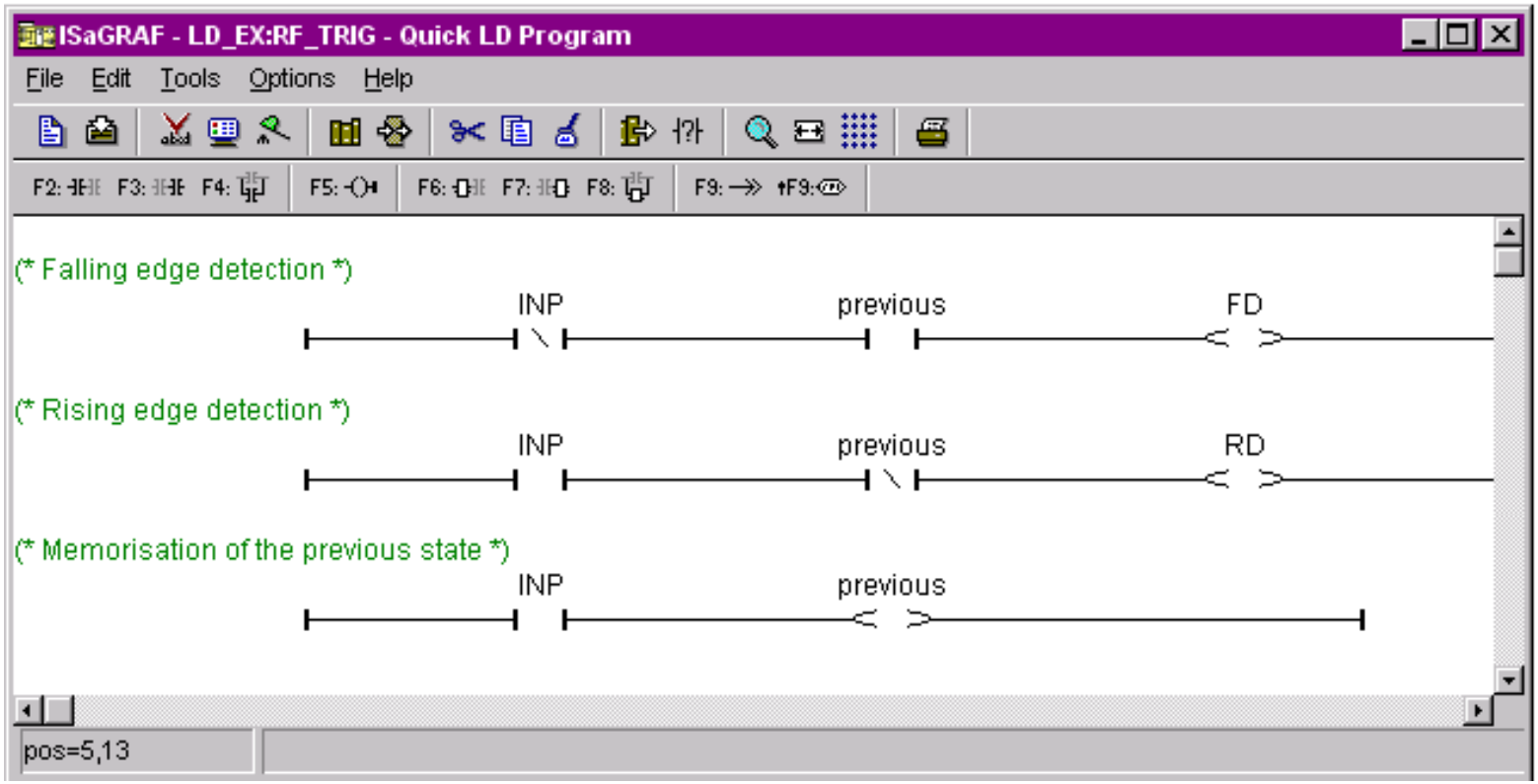
- Double click on variable to access second level edition

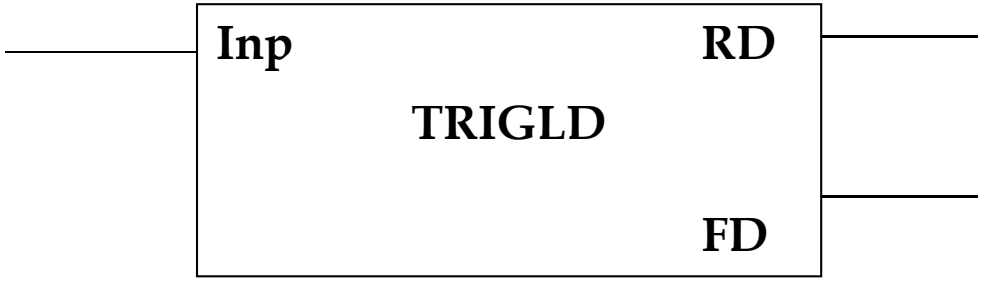
Редактор структурированного текста
Structured Text Editor

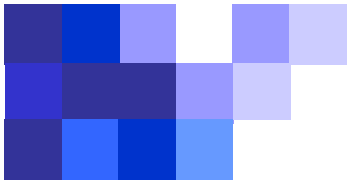


- Присвоение / Assignation :=
- Конец предложения / End of statement ;
- Комментарий / comments (* comment *)

Редактор Quick LD
Quick LD Editor





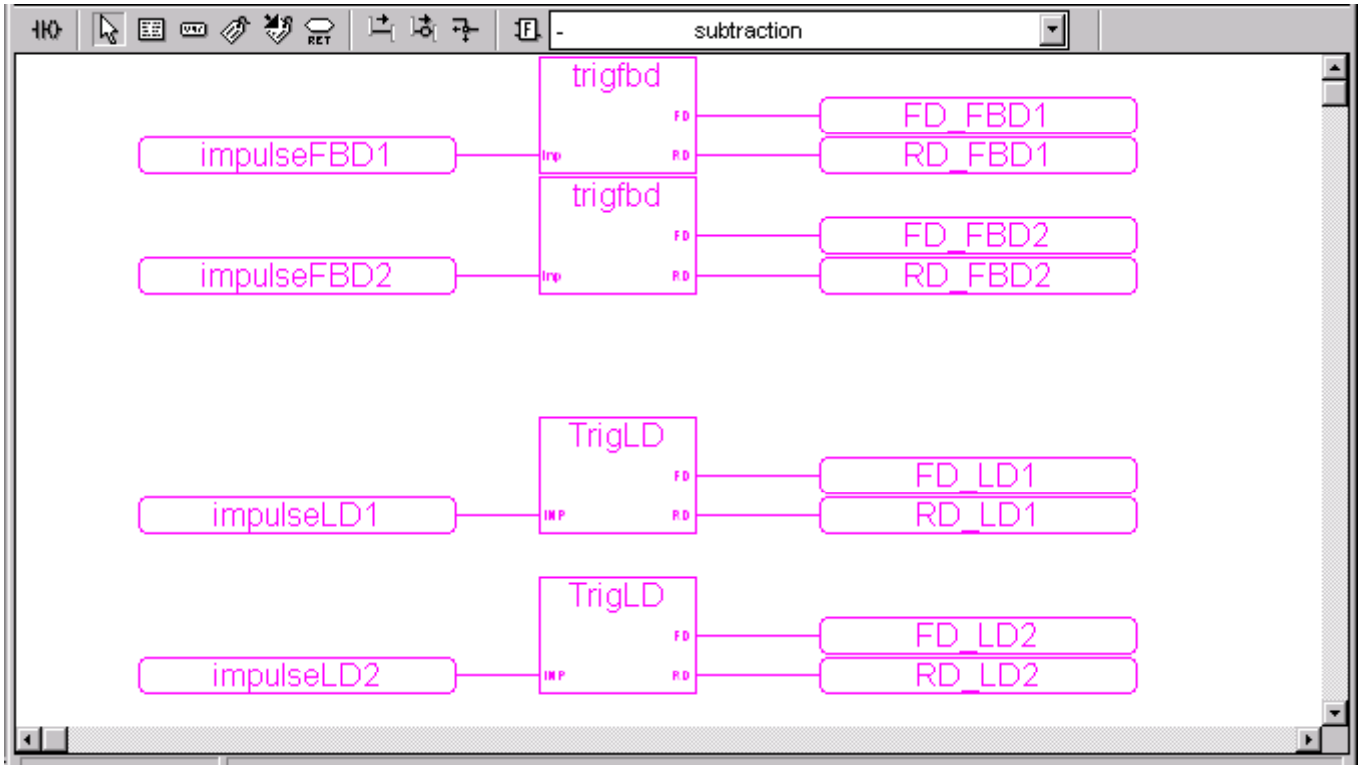


ST / LD / FBD

- **Напишите в FBD следующую ST структуру**
- **Write in FBD the following ST structure**

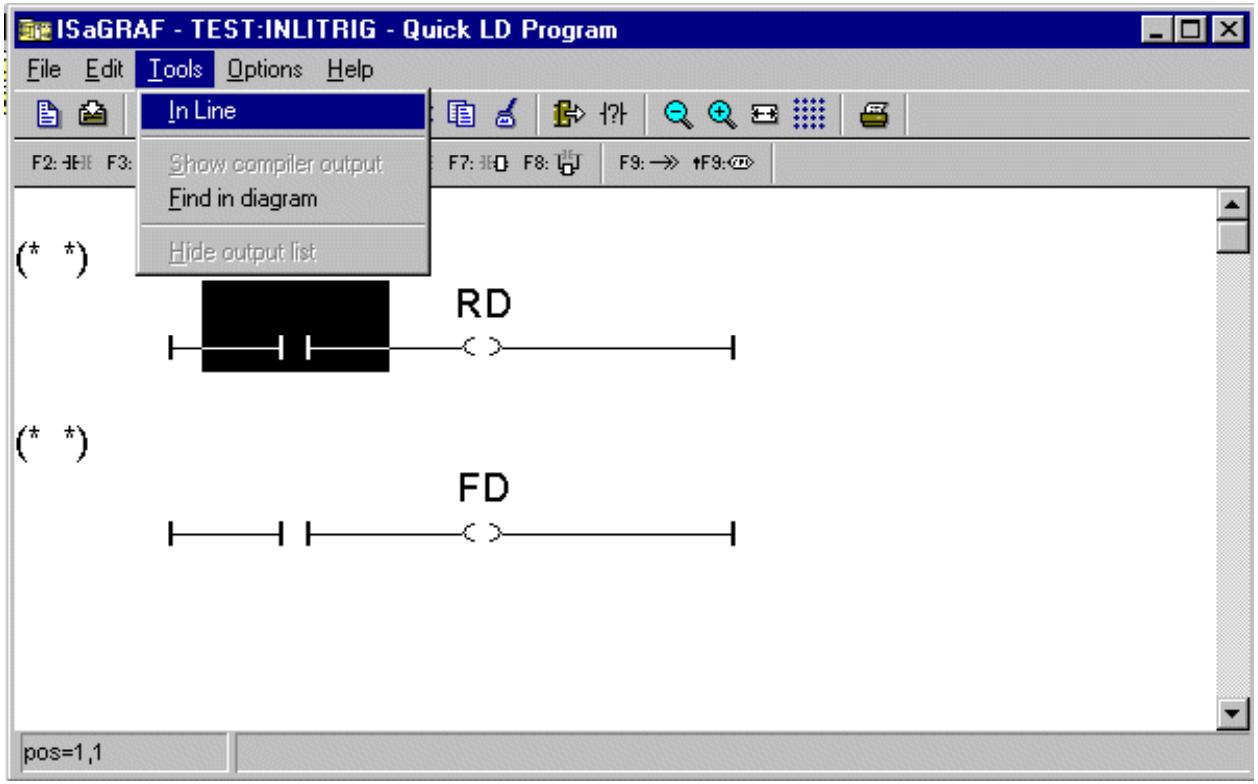
```
IF cmd1 THEN
   ana1 := ana1 + ana2 ;
   boo1 := boo1 AND boo2 ;
ELSE
   ana1 := ana1 + ana3 ;
   boo1 := boo1 AND boo3 ;
END_IF ;
```

Редактор FBD
FBD Editor



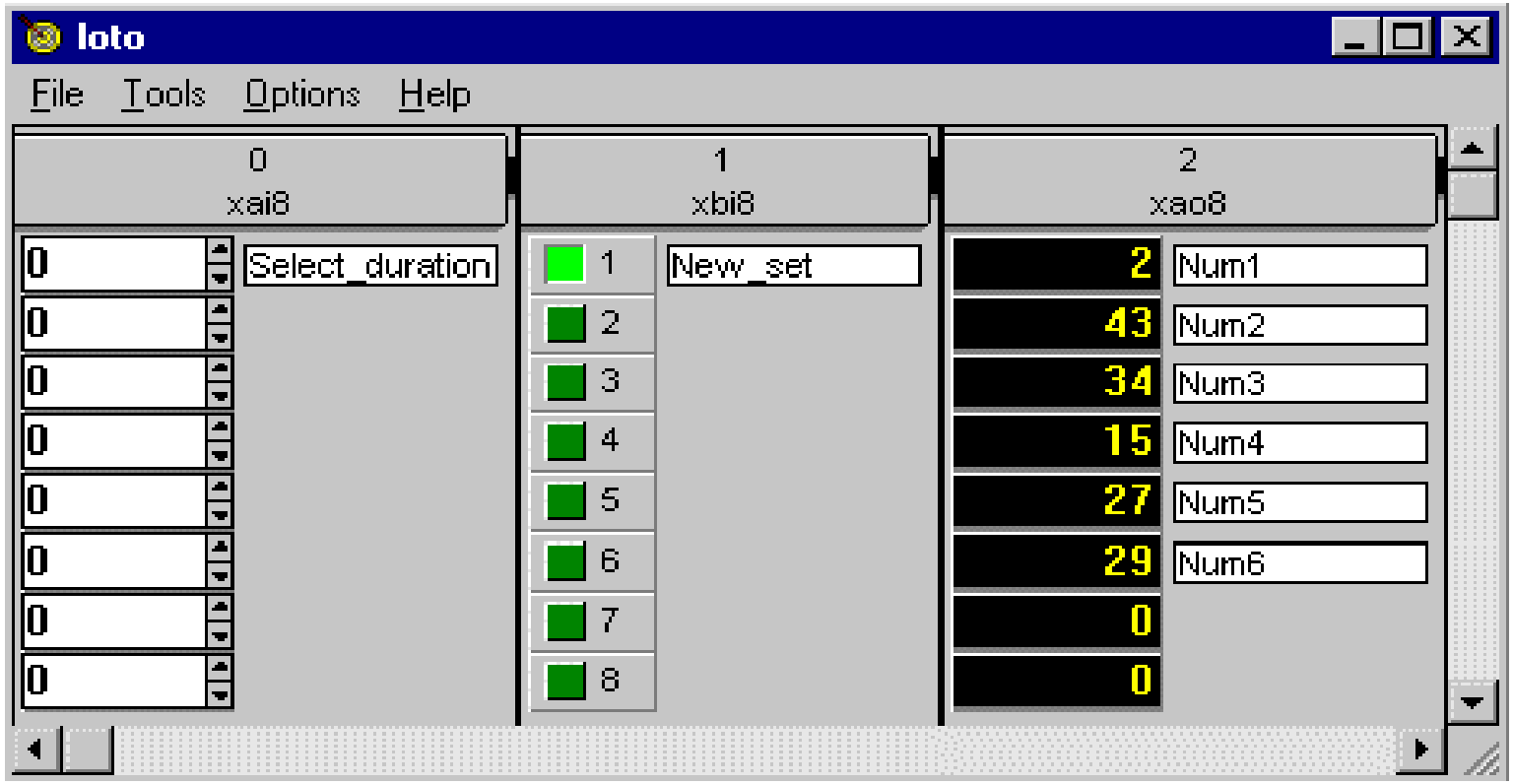
- **Можно использовать строку инструментов LD**
- **Can use LD tool bar**

Функциональные блоки “In-Line/в строке”
“In-Line” Function Block

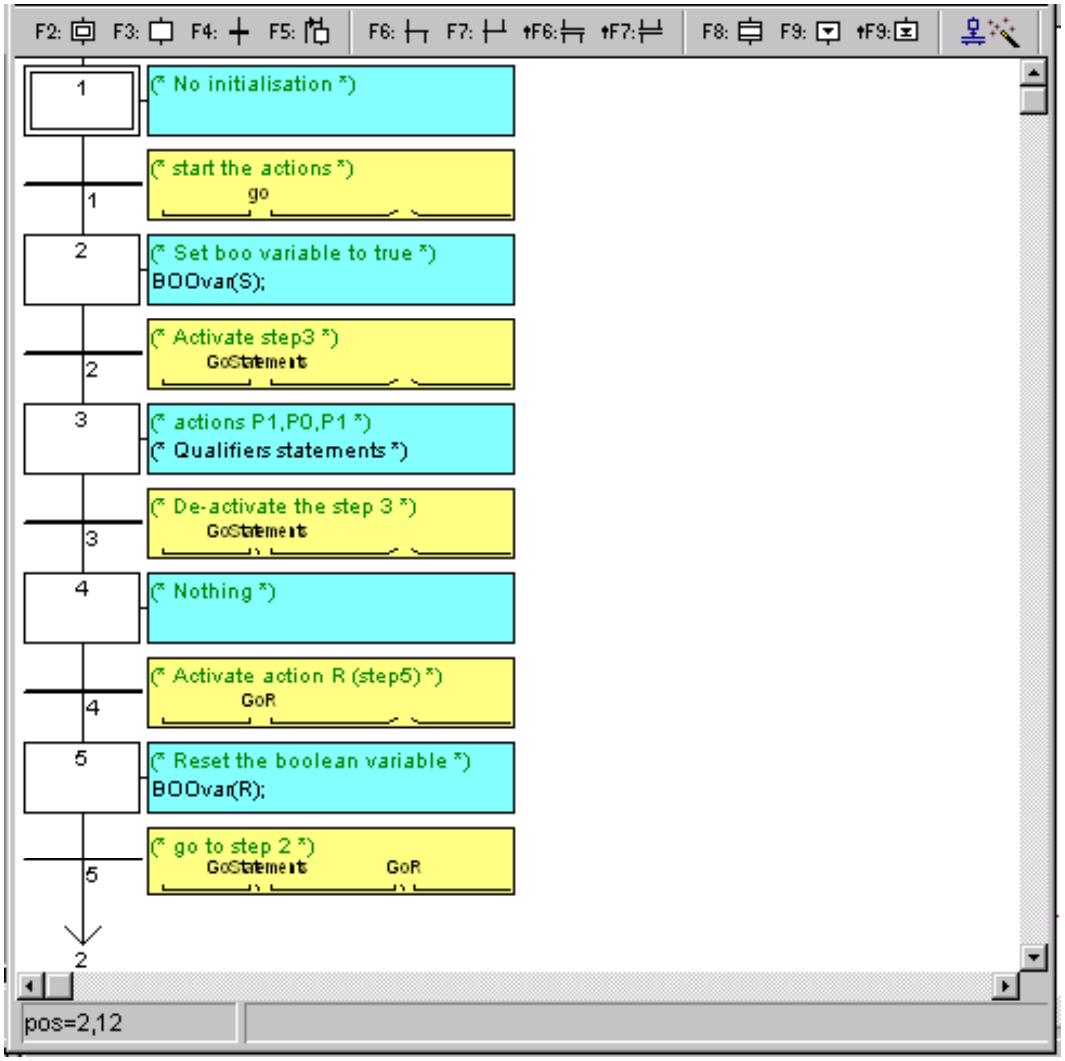


Редактор схемы протекания (выполнения) Flow Chart Editor

- Упражнение ЛОТО
- LOTO exercise



Редактор SFC
SFC Editor



Отладка
Debugging