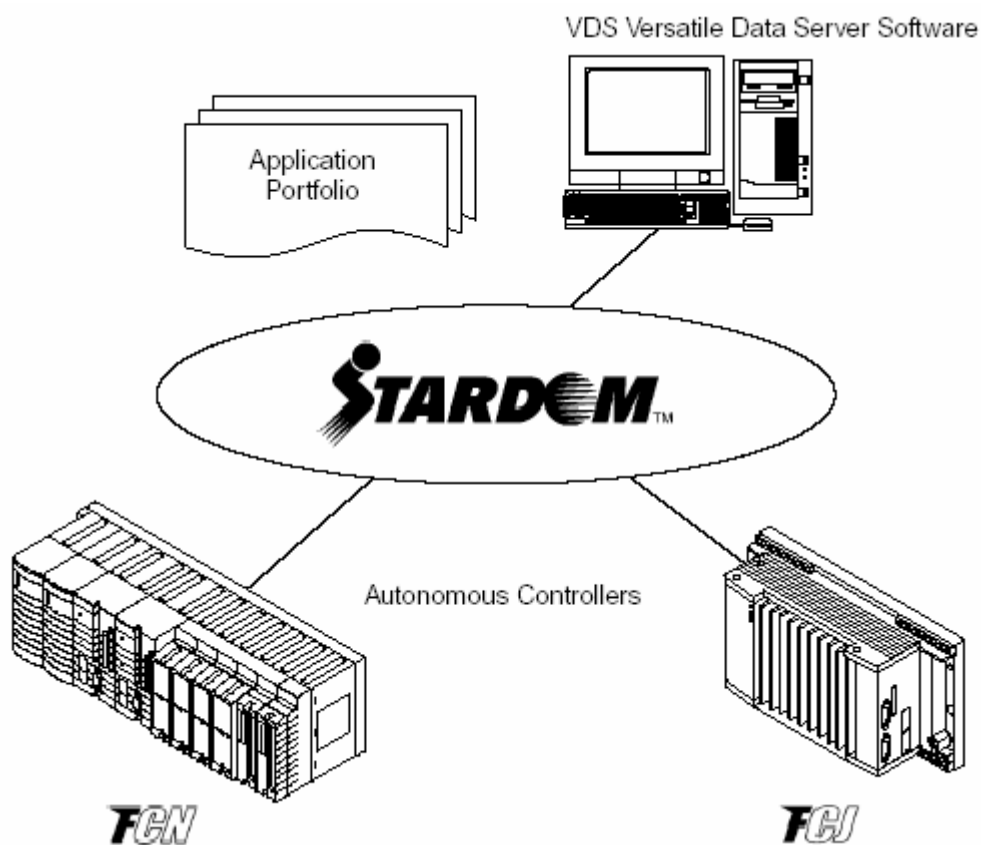


## ЧАСТЬ 2



## ФУНКЦИИ FCX





## СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	3
2.1 Введение.....	5
2.1.1 Конфигурация FCX.....	5
2.1.2 Программирование FCX.....	6
2.1.3 Последовательность создания проекта.....	7
2.1.4 Библиотеки приложений и многоуровневая концепция IES.....	8
2.2 Конфигурация аппаратуры FCX.....	9
2.2.1 Интерфейс ввода/вывода.....	9
2.2.2 Структура памяти FCX.....	11
2.2.3 Многозадачность (Multi-tasking).....	14
2.2.4 Присвоение IP адресов.....	16
2.2.5 Конфигуратор ресурсов (Resource Configurator).....	19
2.2.5.1 Процедуры – Подключение к FCX (Connecting).....	20
2.2.5.2 Общие установки CPU (General).....	21
2.2.5.3 Регистрация лицензий на программное обеспечение CPU.....	25
2.2.5.4 Конфигурирование модулей ввода/вывода.....	26
2.2.5.5 Импорт/экспорт конфигурации.....	27
2.2.5.6 Групповое копирование программ (APC - All Program Copy).....	27
2.2.6 Web страница обслуживания (The Maintenance Web Page).....	28
2.3 Функции Logic Designer.....	31
2.3.1 Обзор.....	31
2.3.2 Блоки организации программ (Program Organization Units) (POU).....	32
2.3.3 Задачи (Tasks).....	34
2.3.4 Оперирование.....	37
2.3.4.1 Меню “BUILD”.....	37
2.3.4.2 Меню “ONLINE”.....	37
2.3.4.3 Online загрузка с использованием Patch POU.....	41
2.3.4.4 Сохранение параметров настройки.....	42
2.3.4.5 Восстанавливаемые проекты (Backing-up projects).....	43
2.3.5 Процедуры.....	44
2.4 Данные.....	51
2.4.1 Типы данных.....	51
2.4.2 Структуры данных.....	53
2.4.3 Типы переменных.....	60
2.4.4 Декларирование переменных.....	62
2.4.5 Глобальные константы.....	65
2.5 Программирование в IES61131-3.....	66
2.5.1 Обзор.....	66
2.5.2 Порядок исполнения.....	68
2.5.3 Процедуры создания функционально блочных диаграмм (FBD).....	70
2.5.4 Защита POU.....	72
2.6 Функциональные блоки NPASPOU Yokogawa.....	74
2.6.1 Краткий обзор блоков NPASPOU.....	74
2.6.2 Использование блоков NPASPOU.....	79
2.6.3 Процесс коммутации ввод/вывода (Connecting Process I/O).....	80

---

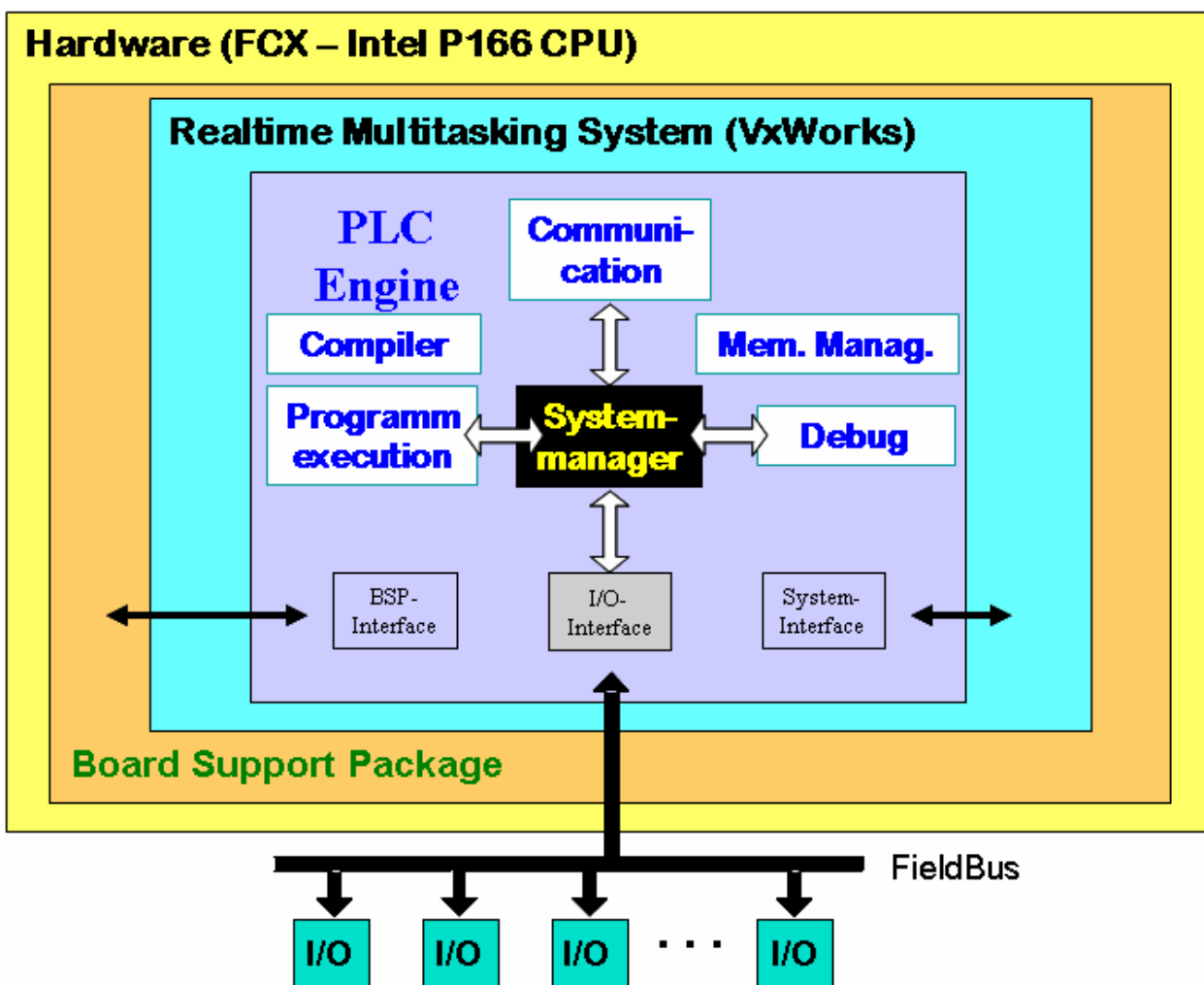
2.6.4 Коммутация внутренних параметров (Internal Parameters).....	82
2.6.5 Конвертирование типов данных.....	86
2.6.5.1 Конвертирование логических переключателей в переключатели PAS_POU...	86
2.6.5.2 Конвертирование логических переключателей в переключатели NPAS_POU.	86
2.6.5.3 Конвертирование других типов данных.....	88
2.6.6 Соединения обратной передачи данных каскадного подключения. ....	89
2.6.6.1 Контур каскадного управления. ....	89
2.6.6.2 Аналоговые выходы. ....	91
2.6.7 Трансляция сообщений об алармах в HMI.....	92
2.6.8 Межконтроллерные коммуникации (SD_FCXCOM_LIB). ....	93
2.6.8.1 Неподтверждаемые коммуникации (Unconfirmed communications).....	95
2.6.8.2 Подтверждаемые коммуникации (Confirmed Communications).....	97
2.6.8.3 Программирование функций чтения/записи в Logic Designer .....	100
2.7 Средства отладки (Debugging Tools). ....	102
2.7.1 Запуск отладочного режима. ....	103
2.7.2 Процедура связывания внешних входов/выходов.....	104
2.7.3 Процедуры логического анализатора (Logic Analyzer).....	106
2.7.4 Окно наблюдения (Watch Window).....	107
2.8 Приложения HTML (HTML Applications).....	110

## 2.1 Введение.

### 2.1.1 Конфигурация FCX.

FCN и FCJ, далее по тексту FCX базируются на процессорах типа Pentium P166 CPU.

Функции контроллера выполняются модулем CPU с использованием стандарта IEC61131 ориентированным на функции PLC. Всё программное обеспечение работает под управлением многозадачной операционной системы реального времени VxWorks. Более подробно о многозадачности см. раздел [2.2.3](#).



## 2.1.2 Программирование FCX.

FCX программируется с использованием стандарта IEC61131-3. Стандарт специфицирует следующие языки программирования PLC:

- Язык описания Функционально Блочных Схем (**Function Blocks (FB)**);
- Язык описания Логических Диаграмм (**Logic Diagrams (LD)**);
- Язык описания Структурированного Текста (**Structured Text (ST)**);
- Язык описания Последовательных Функциональных Карт (**Sequence Function Charts (SFC)**);
- Язык описания Листа Инструкций (**Instruction List (IL)**).

Стандарт специфицирует незначительное количество базовых функций, которое расширено фирмой Yokogawa включая ряд функций высокого уровня подобных Функциональным Блокам CS3000 (DCS). Описание этих блоков включено в этот раздел.

Наряду со специфицированием языков программирования стандарт также определяет структуры данных (**data structure**) для всех данных которые дефинированы в системах управления. Данные могут быть связаны с функциональными блоками и их элементами (**Tags**) или объявлены как глобальные (**global**) или внешними переменными (**external variables**), которые также свойственны системе как системе в целом. Они включают в себя также и переменные ввода/вывода (**I/O variables**), которые по сути своей являются глобальными переменными.

Программирование системы поддерживается логическим конструктором (**Logic Designer**). С его использованием создаётся проект (**project**), который содержит в себе конфигурационную информацию системы. Внутри проекта содержатся программы созданные и скомпонованные для всех FCX в виде задач (**tasks**), которые определяют, как работают программы внутри FCX.

Аппаратура настраивается при помощи конфигуратора ресурсов (**Resource Configurator**). При помощи него конфигурируется процессорный модуль (**CPU**) и модули ввода/вывода (**I/O moduls**) FCX, допускается также и копирование данных из FCX и сохранение их PC (**upload**).

### 2.1.3 Последовательность создания проекта.

Последовательность создания проекта состоит из трёх основных этапов:

- **FCX Builder** – редактируется в **Logic Designer**
- **Dataserver Builder** – редактируется в **Object builder** и **Graphic builder**
- **Webserver** – создаётся в **HMI Deployment tool**

*Для сокращения трудозатрат каждый из редакторов имеет средства импорта/экспорта:*

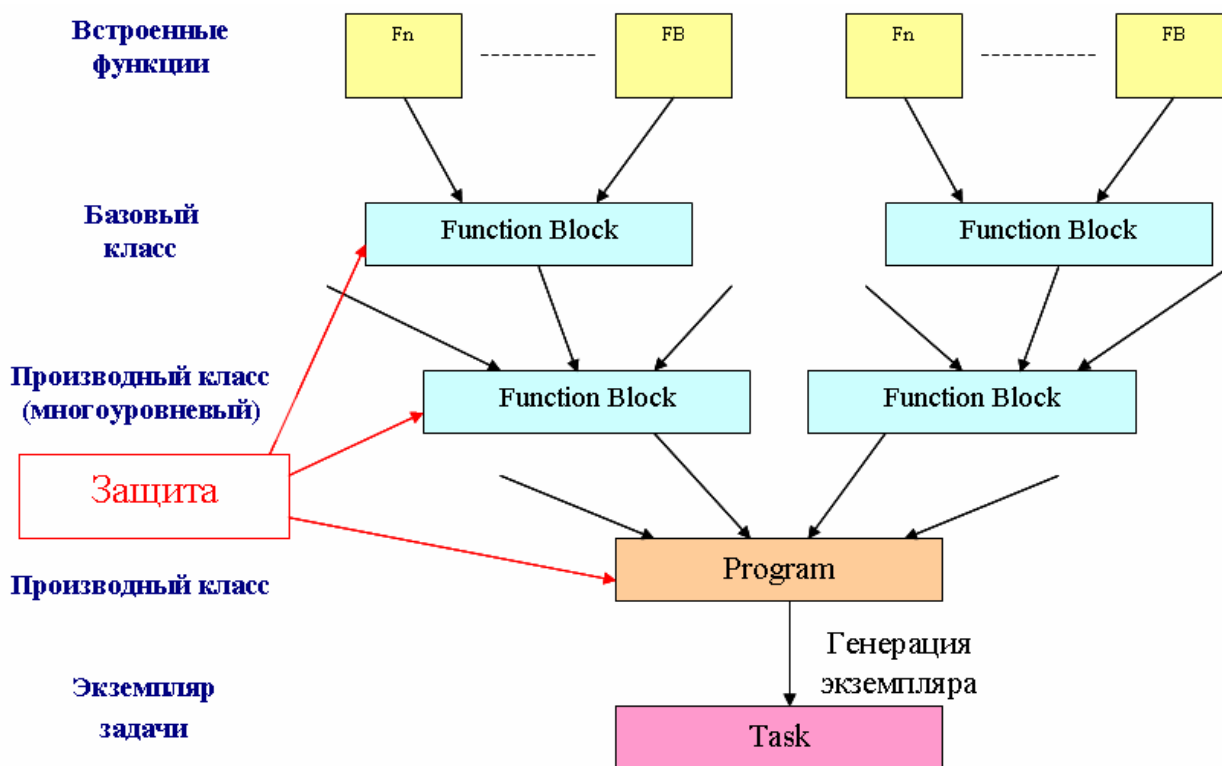
- **Resource Configurator**
  - “Plug & Play” функция (читает конфигурацию FCX)
  - Отсутствует возможность экспорта/импорта, но можно копировать и передавать в/из Excel
- **Logic Designer**
  - Не может импортировать конфигурацию из Resource Configuration
  - Импорт/экспорт **Device Label Definition** из других проектов
  - Импорт/экспорт CSV файлов в общие
- **Object Builder**
  - Импорт описаний из **Logic Designer** (ADLST)
  - только для NPAS POU модулей
  - функция генерации объектов для других модулей
- **Graphic Designer**
  - Функция импорта/экспорта
  - Поиск имён элементов в графическом редакторе

### 2.1.4 Библиотеки приложений и многоуровневая концепция ИЕС.

Благодаря реализации многоуровневой концепции и возможности защиты прикладных библиотек пользователь может создавать свои библиотеки и защищать свою интеллектуальную собственность.

Многоуровневая концепция означает, что функции и функциональные блоки могут быть подмножеством функций и функциональных блоков более высокого уровня с множеством уровней вложения. Более подробно см. раздел [2.3.2](#).

Функциональный блок, который может содержать много других функций и функциональных блоков, может быть запаролен, так что никто не сможет видеть или редактировать программы находящиеся в нём. Более подробно см. раздел [2.5.3](#).





## 2.2 Конфигурация аппаратуры FCX.

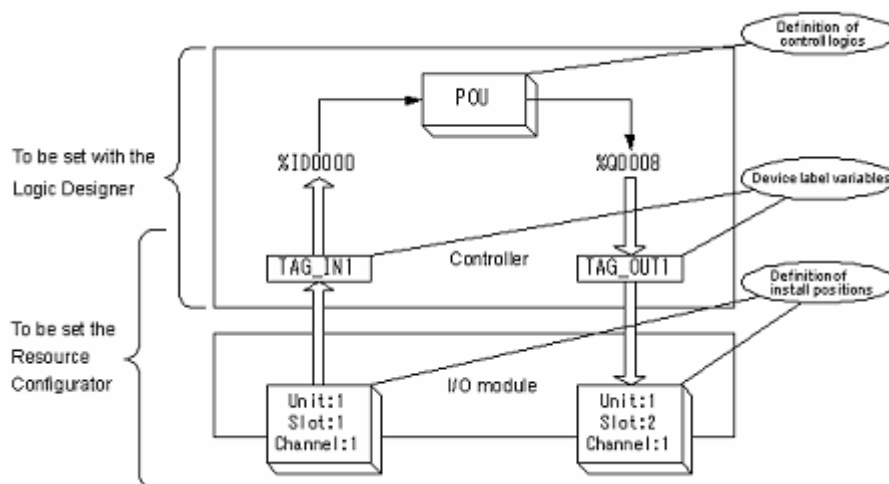
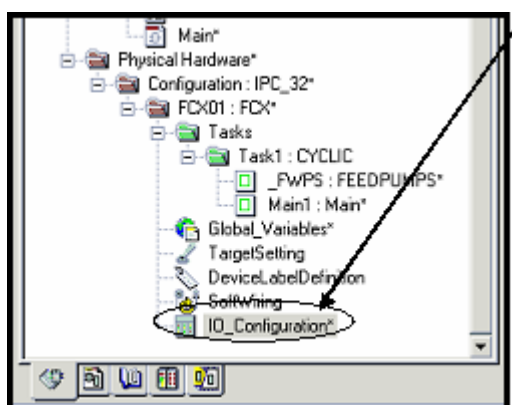
### 2.2.1 Интерфейс ввода/вывода.

Интерфейс ввода/вывода обеспечивает связь между управляющим программным обеспечением и процессом ввода/вывода. Процесс ввода/вывода может осуществляться следующими способами:

- Ввод/вывод через модули ввода/вывода (**On-board hardware I/O**);
- Ввод/вывод через полевые сети (**Fieldbus I/O**);
- Ввод/вывод через управляющую сеть (**PLC communications**).

Ввод/вывод конфигурируется непосредственно с использованием **Resource Configurator** (см. раздел [2.2.5](#)). При этом конфигурируются имена меток устройств (**device label**) и другие параметры ввода/вывода.

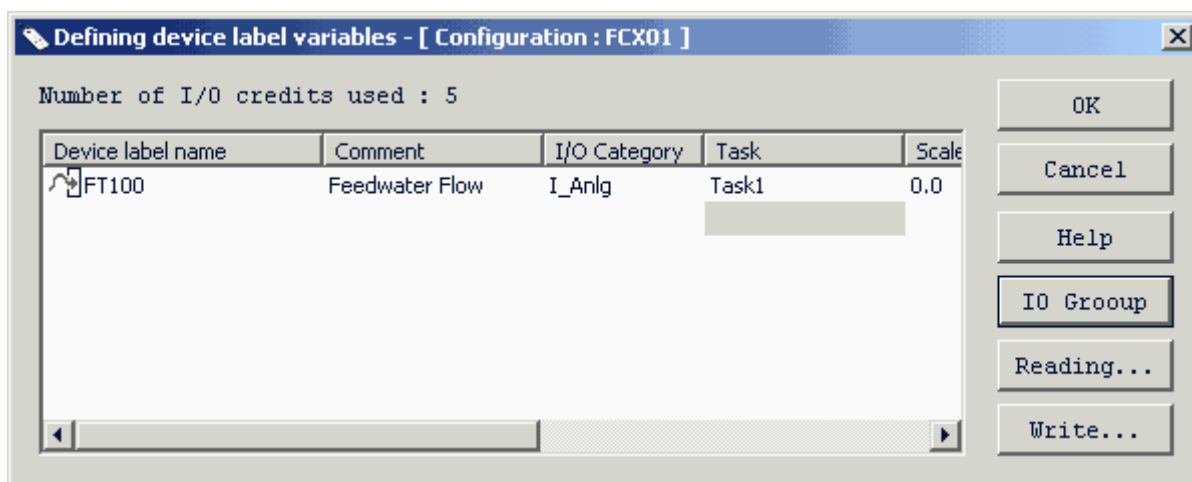
Эта конфигурация подключается к управляющему программному обеспечению, так что программы могут читать/писать данные из/в устройства ввода/вывода, как показано на рисунке ниже. Эта конфигурация находится в папке “**Physical Hardware**” проекта (см. в **Logic Designer**) в файле “**Device Label Definition**”.



### Определение меток устройств (Device Label):

Элементы ввода/вывода или метки устройств, определённые в **Resource Configurator** должны быть введены в файл “**DeviceLabelDefinition**”. Это обеспечивает непосредственную связь между точками процесса ввода/вывода и программным обеспечением. Когда система собрана и загружена имена меток согласовываются с именами меток ввода/вывода аппаратуры что позволяет преобразовывать форматы данных между программным обеспечением и аппаратурой.

Для того чтобы открыть **Device Label Definition** щёлкните дважды по имени файла “**DeviceLabelDefinition**” в папке “**FCX**” в **Logic Designer**:



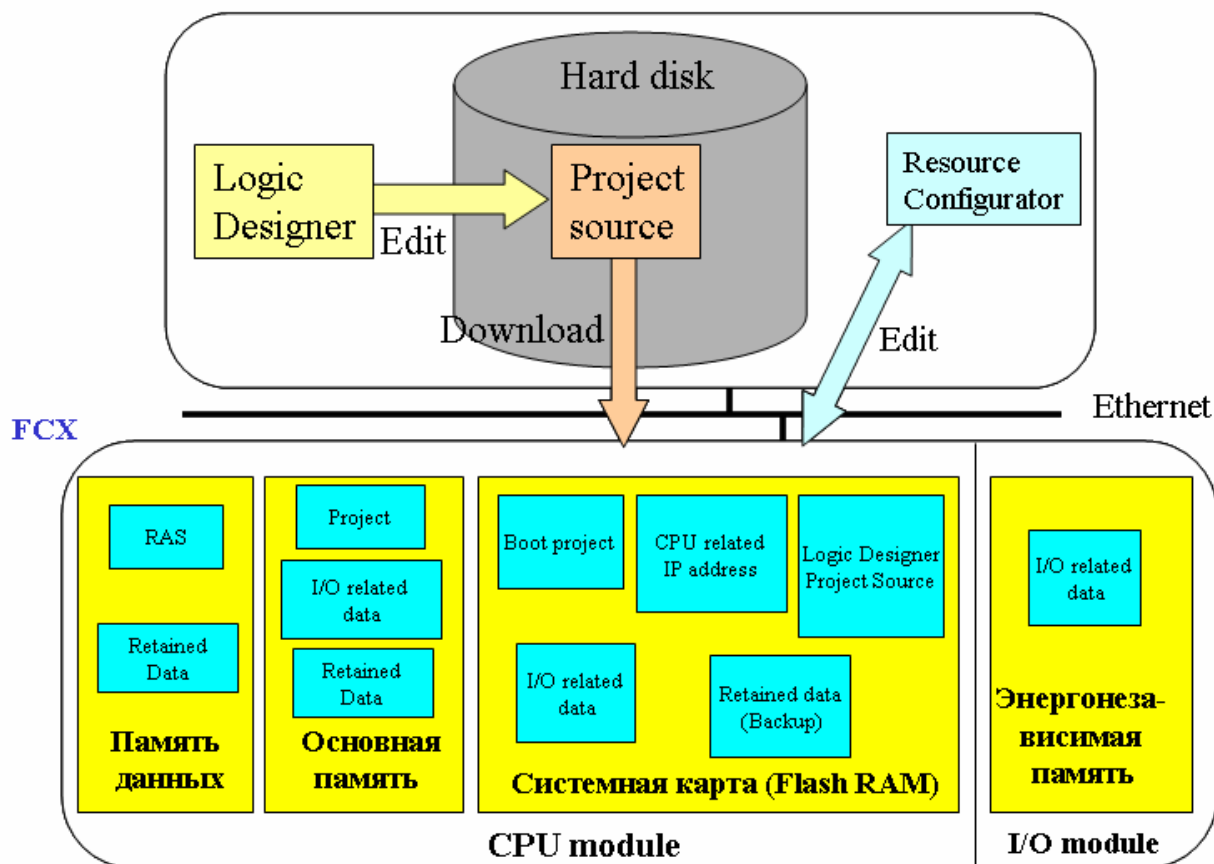
- **Device label name** представляет собой вход с таким же именем в описании аппаратуры. Шкалы и другие параметры могут быть заданы.
- Кнопки “**Reading...**” и “**Write...**” осуществляют импорт и экспорт ранее описанных конфигураций и поддерживают возможность копирования общих установок ввода/вывода для других проектов или ресурсов.
- Кнопка “**IO\_Group**” осуществляет изменение конфигурации групп ввода/вывода (**IO Groups**), но если информация в этом окне не модифицируется, то она не должна изменяться.
- Совместное использование входов/выходов различными задачами с разными циклами сканирования не рекомендуется, особенно в случаях выходов и в конфигурации резервирования CPU.

### 2.2.2 Структура памяти FCX.

Существует три вида памяти в FCX:

- **Основная память (Main Memory)** – DRAM (32 МВ), динамическая (*volatile*), хранит исполняемые программы, которые загружаются при загрузке из загружаемого проекта хранящегося в Flash RAM. Также хранит текущие данные (настроечные параметры (*tuning parameters*)). Заметим, что все эти данные теряются при отключении питания.
- **Память данных (Data Memory)** – SRAM (512КВ), Энергонезависимая статическая память с питанием от батареи питания, размещённой на материнской плате. Сохраняет настроечные параметры, которые могут быть возвращены во Flash RAM.
- **Системная карта (Flash RAM)** (сменная Compact Flash карта) – содержит системное программное обеспечение, загрузчик проекта и исходный файл проекта, настроечные параметры (поддерживаемые переменные (*retained variables*)).

#### PC - конфигурирование FCX



### Сохранение данных (Tuning Parameters):

Как глобальные (**Global**), так и локальные (**Local**) переменные могут быть сохранены в случае пропадания питания FCX. Такие переменные обычно называются поддерживаемыми данными “**retained data**”. Они содержат доступные параметры (настроечные параметры (tuning parameters)) NPAS POU блоков. Данные могут сохраняться в нескольких вышеупомянутых областях памяти, и это определяется следующим образом:

- Поставьте “**Retain**” галочкой (в Logic Designer – см. раздел [2.4.4](#)) для требуемых переменных.
  1. Если “**Enable Hard-backup for retained data**” помечен в **Resource Configurator**, то данные сохраняются в **Data Memory** (энергонезависимая память)
  2. Если “**Enable Hard-backup for retained data**” не помечен, то данные сохраняются в **Main Memory** (энергозависимая память). Это обеспечивает более быстрый доступ к данным, но их меньшую защищённость.
- Данные могут быть скопированы во **Flash RAM**:
  1. Инициацией глобальной переменной (**GS\_RETAIN\_SAVE\_SW**). Кнопка на графическом окне может быть связана с этой переменной для быстрого доступа к этой функции.
  2. Эти данные перезагружаются в программу из **Data memory** или **Main memory**, если оказались утерянными.

### Начальные значения величин (Initial Values):

Как глобальные (**Global**), так и локальные (**Local**) переменные могут иметь начальные значения величин (**Initial Values**). Они специфицируются в уставках переменных в **Logic Designer** в поле “**Init**”, сохраняются во **Flash RAM** и загружаются в **Main RAM** при включении питания или при холодном старте (**Cold Start**) FCX.

Name	Type	Usage	Description	Address	Init	Retain	P...	OPC
[-] Default								
P1INT	BOOL	VAR_INPUT			0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
P1MODE	DW...	VAR_INPUT				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
AUTSTAT	BOOL	VAR			1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
P1DTY	BOOL	VAR_OUT...				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### Перезагрузка поддерживаемых данных при возникновении неисправности источника питания:

Если в **Data Memory** поддерживаемые данные отсутствуют:

1. Если во Flash-RAM отсутствуют сохраняемые данные, то по умолчанию используются данные загрузчика проекта **Boot Project** из **Flash-RAM**.
2. Если во Flash-RAM есть сохраняемые данные, то они перезагружаются как в Data Memory, так и в Main Memory.

Если в **Data Memory** поддерживаемые данные присутствуют:

- 
1. Если поддерживаемые данные согласованы с управляющим приложением (т.е. не испорчены), то производится перегрузка данных из **Data Memory** в **Main Memory**.
  2. Если поддерживаемые данные не согласованы с управляющим приложением (т.е. испорчены), то производится перегрузка данных из **Flash-RAM** как в **Data Memory**, так и в **Main Memory**.

### 2.2.3 Многозадачность (Multi-tasking).

Управляющее программное обеспечение является многозадачным, так что каждая задача со своим временем сканирования и приоритетом будет выполняться в запланированный момент времени.

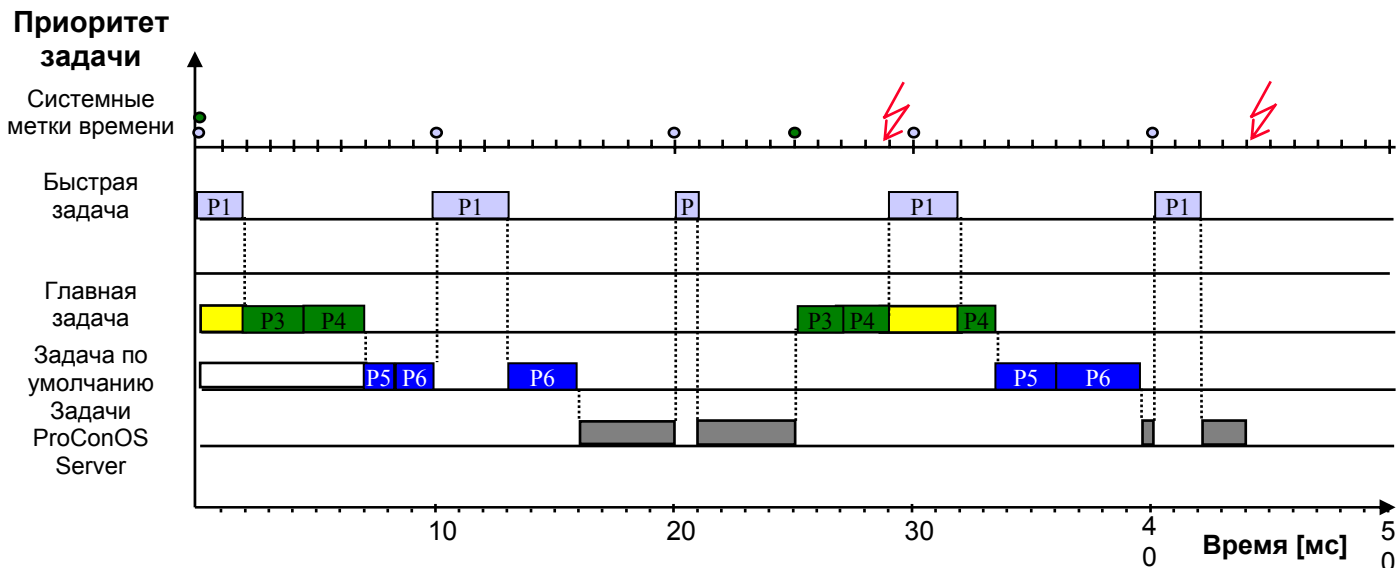
Задачи обсуждаются в разделе [2.3.3](#), и являются внутренней частью многозадачной системы. Задачи могут быть:







- **Циклическими (Cyclic)** – выполняются периодически с фиксированным временем сканирования (не менее 10 мс.);
- **По умолчанию (Default)** – выполняются не периодически, а в свободное от других задач время (в фоновом режиме);
- **Системные (System)** – задачи операционной системы, выполняемые по наступлению системных событий, таких как холодный/теплый/горячий старт, возникновение системных ошибок и т.д.

**Приоритет (Priority)** – Если система перегружена, не все циклические события могут быть обработаны своевременно. Чем выше задан приоритет задачи, тем больше времени выделяется на ее выполнение. Задача по умолчанию может вообще не получить возможности выполниться так как она имеет меньший приоритет и система перегружена.

Приоритетность задач варьируется в пределах от 0 до 15. Более высоким приоритетом обладает задача с меньшим номером приоритетности. По умолчанию присваивается 0, что соответствует наивысшему приоритету.

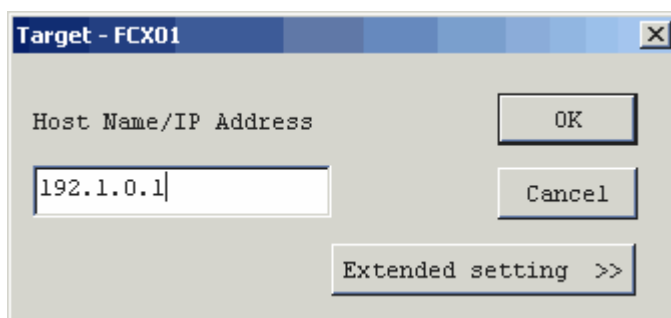
Схема внизу показывает на примере, как различные задачи выполняются в течение временного цикла FCX:



-  - Быстрая задача (**Fast Task**): Периодическая задача с временным интервалом от 10мс, обозначена как P1
-  - Главная задача (**Main Task**): Периодические задачи с временным интервалом от 25 мс, обозначены как P3, P4
-  - Задача по умолчанию (**Default Task**): Периодические задачи без фиксированного временного интервала, обозначены как P5, P6
-  - Задачи ProConOS Server: Системные задачи, например коммуникации, отладчик и т.д.
-  - Пустая задача (Task ready)
-  - Запросы на обслуживание (Task signals) (Задача переходит в состояние "ready")

## 2.2.4 Присвоение IP адресов.

Установки FCX (**FCX Settings**) выполняются при помощи **Logic Designer** при нажатии двойным щелчком на **Target Setting** файл в конфигурационной папке (**Configuration folder**):



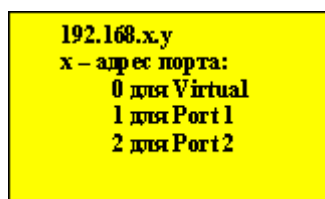
Введите необходимый IP адрес (как показано на рисунке сверху в качестве примера), или имя управляющей станции (**hostname**) FCX, если оно существует в файле LM Hosts. Если IP адрес FCX неизвестен или требуется переустановить, то следуйте инструкции, описанной в п. [2.2.5.1](#).

Так как FCX может иметь дублированное подключение к Ethernet, IP адресация состоит из более чем одного IP адреса. IP адреса подразделяются на:

- **VIP** – виртуальный IP адрес (**Virtual IP address**);
- **PIP** – физический IP адрес (**Physical IP address**).

При инсталляции FCX необходимо устанавливать только VIP, PIP адрес устанавливается автоматически, для PC это делается иначе (см. описание далее).

Если в FCX установлен один CPU, то ему присваиваются два PIP адреса, по одному на каждый порт Ethernet. Они вычисляются путем инкрементирования третьего кода IP адреса как показано на примере ниже:



### 192.168.x.1

где:

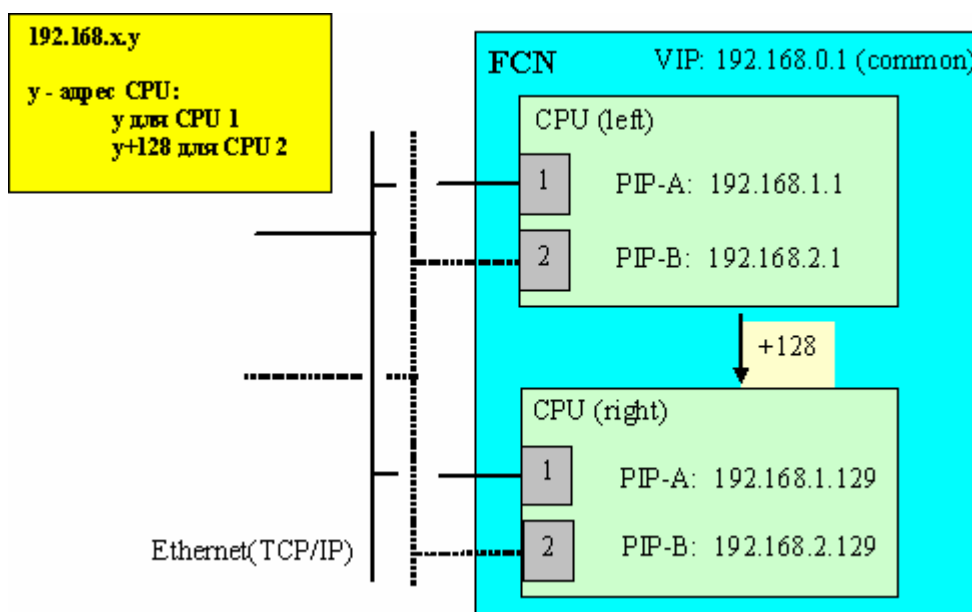
- x = 0 (для VIP адреса);
- x = 1 (для PIP адреса, порт 1);
- x = 2 (для PIP адреса, порт 2).

Если в FCX установлены два CPU, то второму CPU также присваиваются два PIP адреса, по одному на каждый порт Ethernet. Они вычисляются путем добавления 128 к четвёртому коду IP адреса первого CPU как показано на примере ниже:



**192.168.x.129** – Устанавливается функцией **APC Command Transfer** в меню инструментов (**Tools**) конфигулятора ресурсов (**Resource Configurator**) (см. п. [2.2.5.1](#)).

**Примечание:** Для первого CPU не устанавливайте четвёртый код IP номера больше 126, так как они используются для адресации второго CPU, а номер 127 вызывает появление номера 255 для второго CPU ( $127+128=255$ ) который используется системой для адресации широковещательных соединений (**Broadcast Address**).



Так как FCX имеет два вида адресов (**Virtual IP address** и **Physical IP address**), то соответственно и адресация PC, который имеет **Logic Designer, Resource Configurator** and **VDS** должна также иметь множество адресов, как показано ниже:

### **Нерезервированный Ethernet, нерезервированный или резервированный CPU FCX:**

Устанавливаются два IP адреса для порта Ethernet, виртуальный и физический адреса:

Пример: 192.168.0.100 (для VIP адреса);  
192.168.1.100 (для PIP адреса).

### **Резервированный Ethernet, нерезервированный или резервированный CPU FCX:**

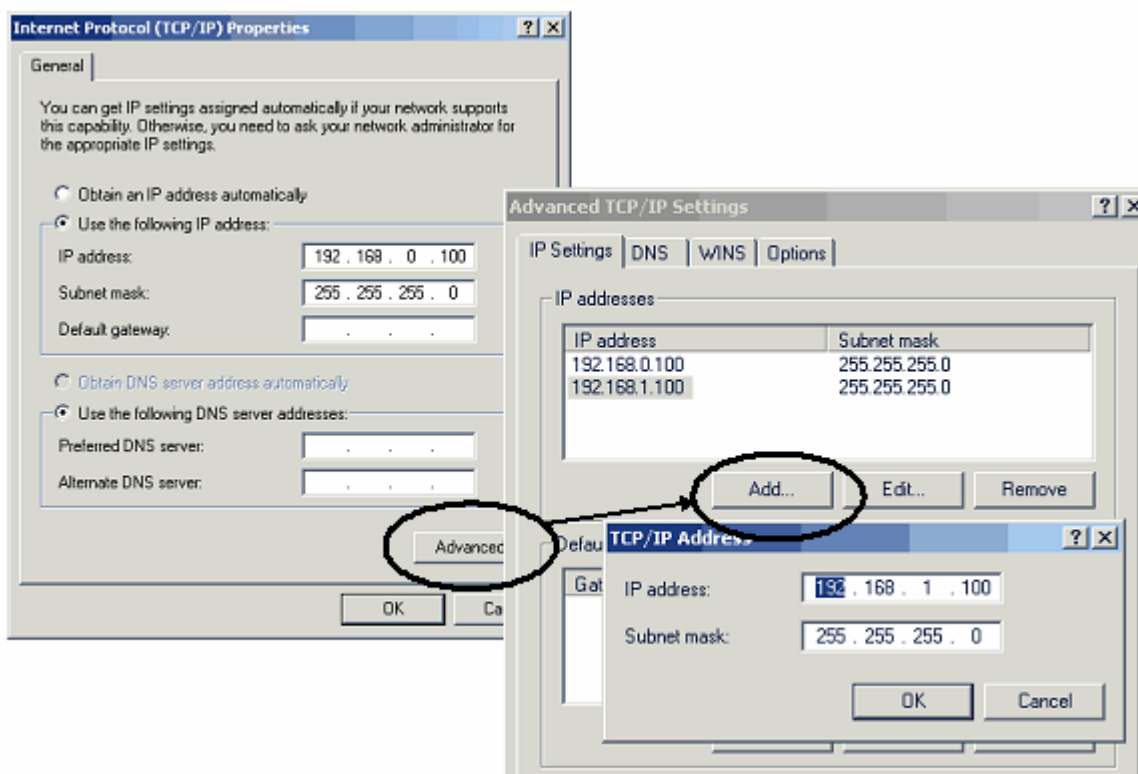
В этом случае в PC имеется второй порт Ethernet. Присвоение адресов первого порта проводится так же, как описано выше, для второго порта не устанавливается виртуальный IP адрес, а только физический:

Пример: 192.168.0.100 (для VIP адреса);  
192.168.1.100 (для PIP адреса, порт 1);  
192.168.2.100 (для PIP адреса, порт 2).

**192.168.x.y**

**x – адрес порта:**  
 0 для Virtual  
 1 для Port 1  
 2 для Port 2

**y – фиксированный для PC:**  
 например 100



## 2.2.5 Конфигуратор ресурсов (Resource Configurator).

Конфигуратор ресурсов используется для инсталляции FCX в систему.

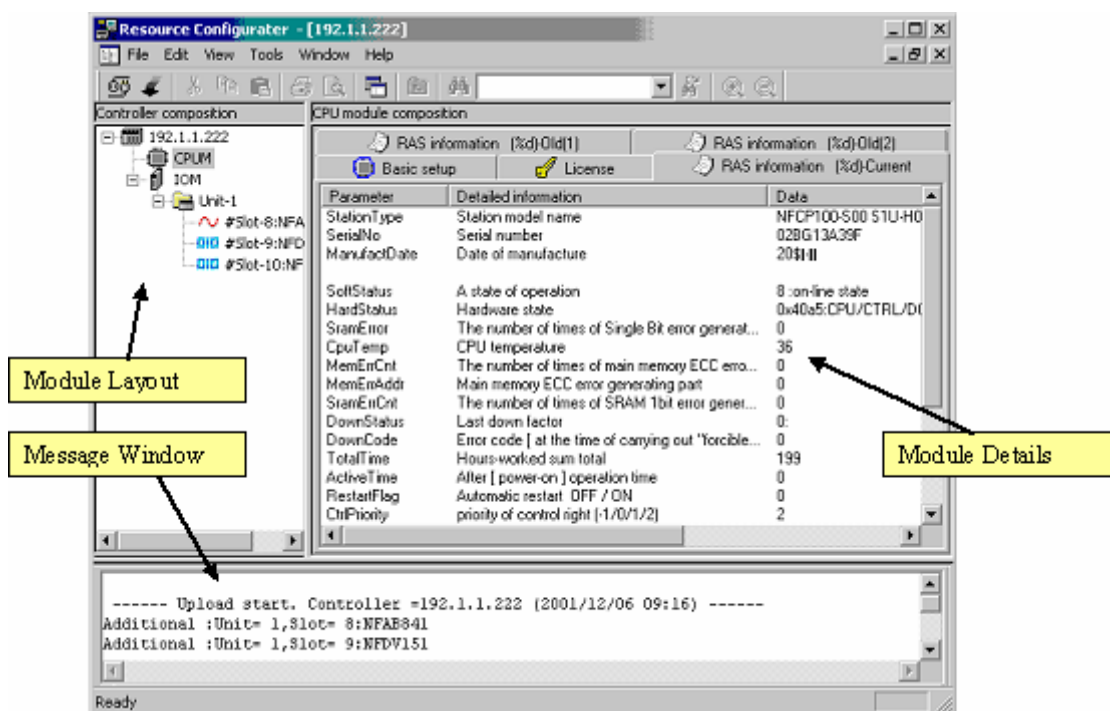
### *Конфигурируются следующие основные пункты:*

- CPU – общие установки (**General Setup**);
- CPU – регистрация лицензий на программное обеспечение (**License Setting**);
- Модули ввода/вывода (**IO Module**) – конфигурирование меток устройств (**Device Label**).

### *Конфигуратор ресурсов имеет также и следующие функции:*

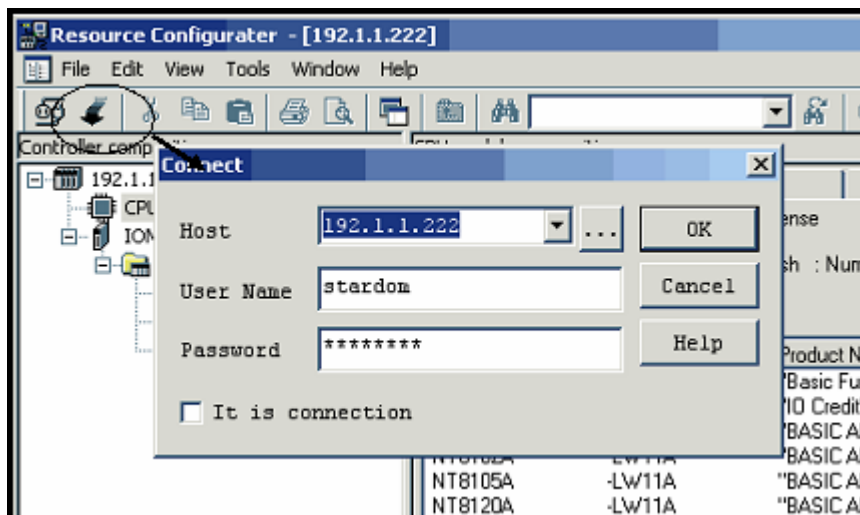
- Импорт/экспорт существующих конфигураций;
- Установка IP адресов;
- Чтение конфигурации из FCX с её сохранением в файле (**Upload**) и загрузка FCX (**Download**);
- APC – групповое копирование программ (**All Program Copy**).

### *Основные компоненты конфигуратора ресурсов:*



### 2.2.5.1 Процедуры – Подключение к FCX (Connecting).

Если IP адрес FCX уже установлен внутри FCX и известен пользователю, то конфигуратор подключит FCX при выборе главного PC (HOST) в диалоге подключения (CONNECT dialog).



- Имя пользователя (User Name): stardom;
- Пароль (Password): YOKOGAWA.

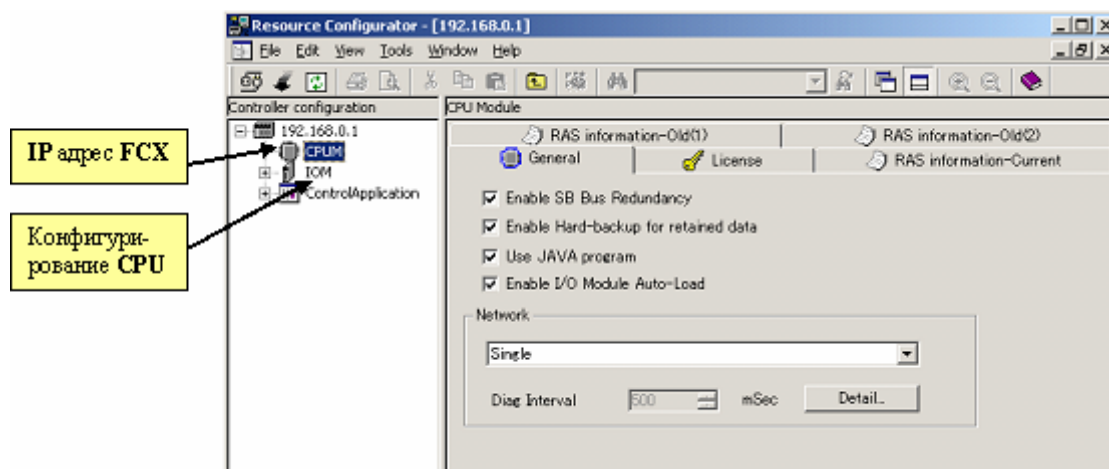
Если IP адрес FCX ещё не установлен внутри FCX или не известен пользователю, необходимо установить этот адрес следующим образом:

1. Подключите питание к FCX. Индикаторы HRDY и RDY на CPU должны засветиться;
2. Пока они мигают, с помощью тонкого штыря нажмите кнопку SHUTDOWN на CPU;
3. Индикатор HRDY продолжит мигать с такой же частотой, но индикатор RDY начнёт мигать реже. Это означает, что FCX перешёл в режим установки IP адреса;
4. Запустите **Resource Configurator**;
5. Убедитесь в том, что область сообщений конфигулятора ресурсов “**Message Window**” видна (если не видна, вызовите выпадающее меню “**VIEW**” и отметьте “**Message Area**”);
6. В течение 30 секунд должно появиться сообщение о том, что FCX найден;
7. Вызовите выпадающее меню “**FILE**” и отметьте “**IP Address Setup**” после чего введите требуемый IP адрес FCX.

**Примечание:** FCX может быть идентифицирован MAC адресом, нанесённым на боковой стороне модуля CPU, в случае FCN, или на задней стороне блока, в случае FCJ.

Если FCX имеет конфигурацию с резервированным CPU, то второй IP адрес устанавливается с помощью функции “APC” меню “Tools”. APC осуществляет групповое копирование программ и копирует содержимое первого CPU во второй. При этом для второго CPU автоматически устанавливается IP адрес.

### 2.2.5.2 Общие установки CPU (General).



#### *Закладка **Basic Setup** поддерживает установку следующих опций:*

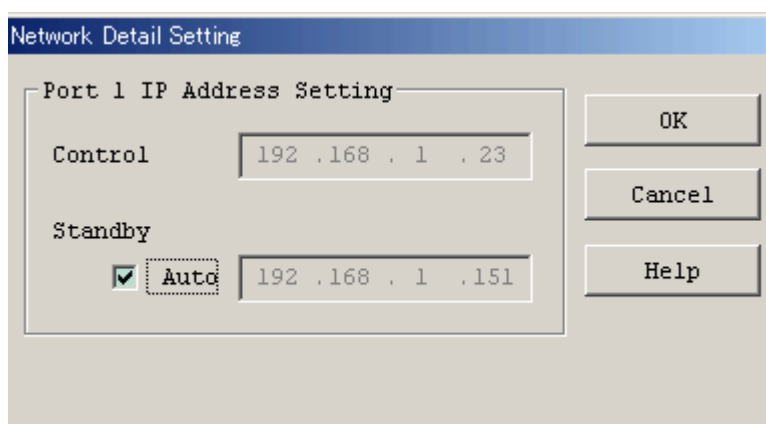
- **Резервирование шины SB (Enable SB Bus Redundancy)** – шина SB является шиной подключающей узлы (**nodes**) FCN (максимум до 3х узлов). Отметьте эту позицию для специфицирования резервирования шинного соединения. Для FCJ не применимо;
- **Аппаратное сохранение поддерживаемых данных (Enable Hard-backup for retained data)** – параметры функциональных блоков (Function Block) могут быть поддержаны при пропадании питания FCX. Эти данные называются поддерживаемыми данными (**Retentive Data**) и назначаются при создании переменных (см. раздел [2.4.4](#)). Отметьте эту позицию для обеспечения сохранения этих данных в SRAM;
- **Использование Java программ (Use JAVA program)** – Отметьте эту позицию для разрешения запуска Java приложений FCX;
- **Использование автоматической перезагрузки настроек модулей ввода/вывода (Enable I/O Module Auto-Load)** – если модули ввода/вывода заменяются, их настроечная информация, которая ранее загружалась, будет автоматически перезагружена. Если вы не хотите, чтобы перезагрузка происходила сразу после замены модуля, снимите марку с этой позиции. В этом случае, конечно, необходима ручная перезагрузка настроечной информации из **Resource Configurator**.
- **Локальная сеть (Network)** – существует три типа локальных сетей, “Single”, “Duplex”, “Separated”. Описание смотри далее.
  - **Single** – одинарная нерезервированная управляющая сеть;
  - **Duplex** – резервированная управляющая сеть;
  - **Separated** – отдельные сети.

Если специфицированы “Single” или “Separated” необходимо установить детальные настройки в диалоговом окне “**Network Detail Setting**”, которое вызывается нажатием кнопки “**Details**”.

Если специфицирован “**Duplex**”, автоматически устанавливаются IP адреса. Вы можете проверить их в диалоговом окне “**Network Detail Setting**”, которое вызывается нажатием кнопки “**Details**”.

- “**Diag Interval**” – используется для установки периода выполнения диагностики сети. Если сеть имеет резервированную конфигурацию, убедитесь в том, что эта функция специфицирована. Период выполнения диагностики сети должен специфицироваться как величина большая или равная 500 мс. с шагом 100 мс.

*Когда к FCN подключается одинарная нерезервированная управляющая сеть:*



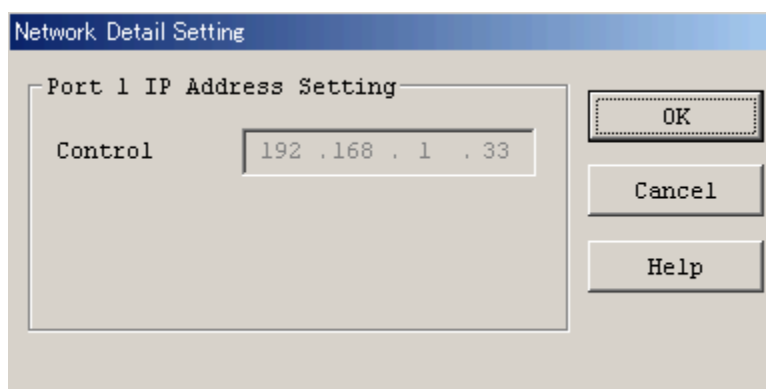
- IP адрес управляющей сети

Индицируемый IP адрес установлен через диалоговое окно “**Setting IP Address**” и не может быть изменён.

- IP адрес резервной сети

При помеченном режиме “**Auto**” IP адрес резервной сети вычисляется и индицируется автоматически. Для ручного ввода IP адреса, снимите марку с режима “**Auto**”.

*Когда к FCJ подключается одинарная нерезервированная управляющая сеть:*



- IP адрес управляющей сети

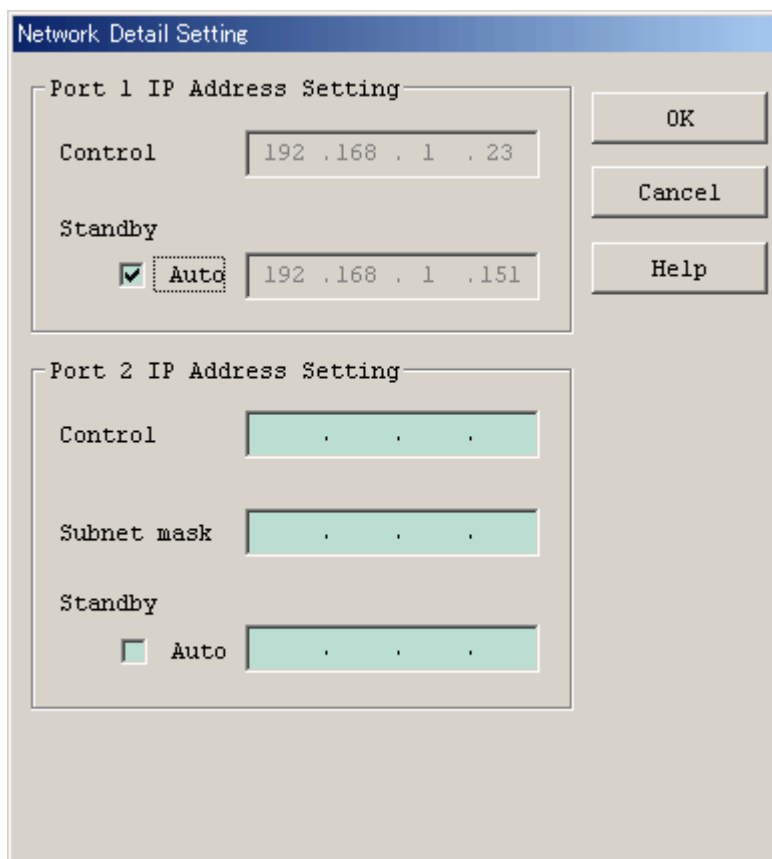
Индицируемый IP адрес установлен через диалоговое окно “Setting IP Address” и не может быть изменён.

*Когда к FCJ или FCN подключается резервированная управляющая сеть:*

- Кнопка “Details” деактивирована.

”Diag Interval” – период выполнения диагностики сети должен быть специфицирован. Период должен специфицироваться как величина большая или равная 500 мс. с шагом 100 мс. (по умолчанию 500 мс.).

*Когда к FCN подключаются отдельные сети:*



- IP адрес управляющей сети порта 1

Индицируемый IP адрес установлен через диалоговое окно “**Setting IP Address**” и не может быть изменён.

- IP адрес резервной сети порта 1

При помеченном режиме “**Auto**” IP адрес резервной сети вычисляется и индицируется автоматически. Для ручного ввода IP адреса, снимите марку с режима “**Auto**”.

- IP адрес управляющей сети порта 2

Введите IP адрес управляющей сети порта 2.

- Сетевая маска порта 2

Введите сетевую маску порта 2.

- IP адрес резервной сети порта 2

Введите IP адрес резервной сети порта 2.

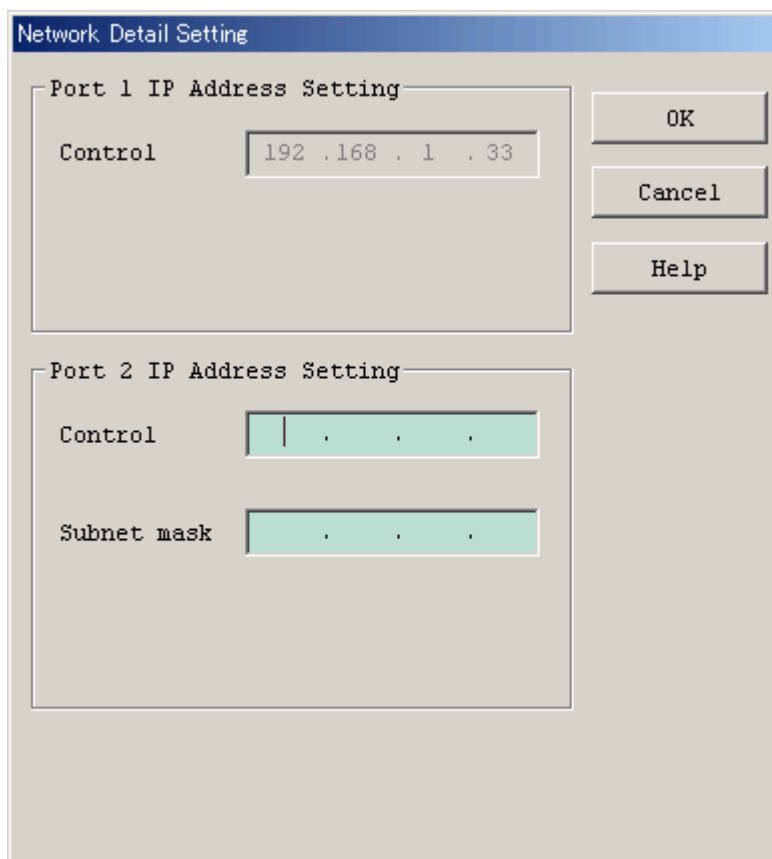
При помеченном режиме “**Auto**” IP адрес резервной сети вычисляется и индицируется автоматически.

Для автоматического вычисления IP адреса резервной сети IP адрес и сетевая маска управляющей сети порта 2 должны быть введены заблаговременно.



Для ручного ввода IP адреса резервной сети, снимите марку с режима “Auto”.

*Когда к FCJ подключаются отдельные сети:*



- IP адрес порта 1

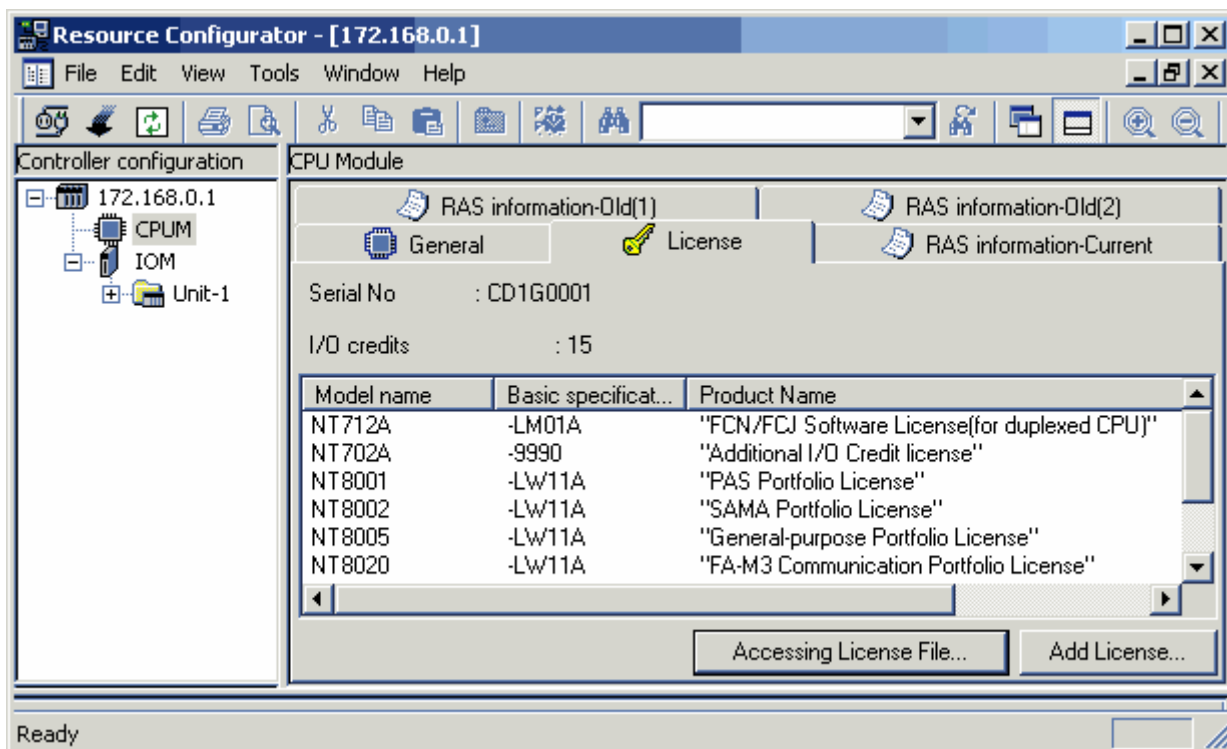
Индицируемый IP адрес установлен через диалоговое окно “Setting IP Address” и не может быть изменён.

- IP адрес порта 2

Введите IP адрес и сетевую маску сети порта 2.

### 2.2.5.3 Регистрация лицензий на программное обеспечение CPU.

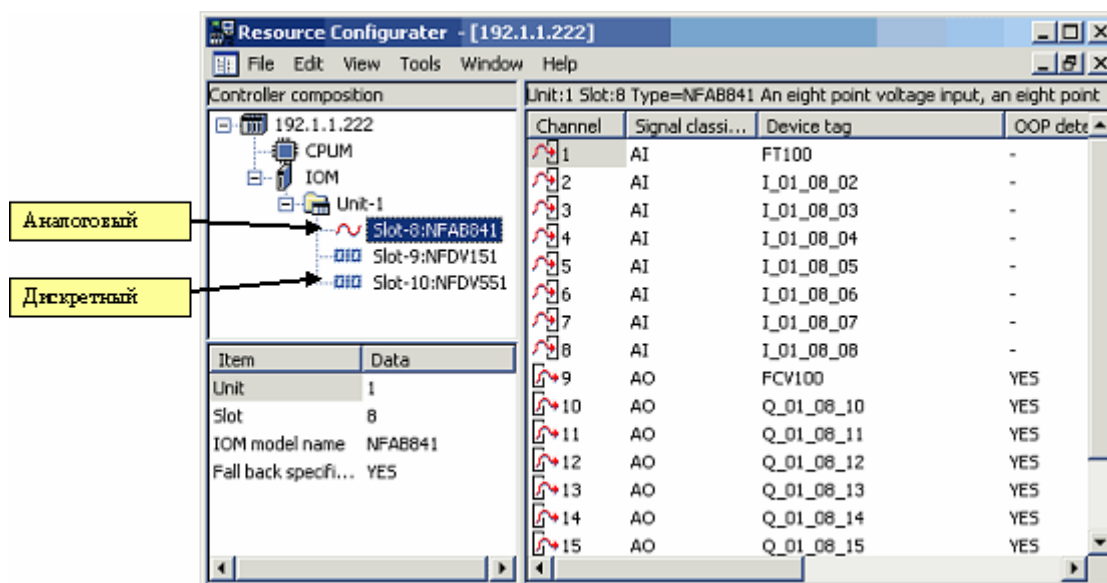
Когда FCX впервые подключается к сети, его лицензия должна быть загружена.



1. Откройте закладку “License”;
2. Нажмите кнопку “Accessing License File...”;
3. Выберите файлы с гибкого диска;
4. Для их загрузки в FCX выберите “DOWNLOAD” в выпадающем меню “FILE”;
5. Произведите рестарт FCX (выключите, а затем включите питание FCX).

#### 2.2.5.4 Конфигурирование модулей ввода/вывода.

Конфигуратор ресурсов автоматически идентифицирует установленные модули FCX. Они распечатываются в трассировочном окне Resource Configurator. Детальная конфигурация может быть просмотрена выделением одного из модулей.



В FCN может быть до 3х каркасов для установки модулей ввода/вывода и в трассировочном окне они отображаются как **UNITS**. Каждый **UNIT** содержит до 8 модулей ввода/вывода.

**Device Label** – Каждому входу или выходу присваивается идентификатор, называемый меткой устройства (**Device Label**). Он должен соответствовать “**Device Label Names**”, конфигурируемым в **Logic Designer** (см. раздел [2.2.1](#)).

### 2.2.5.5 Импорт/экспорт конфигурации.

Однажды сконфигурированные элементы устройств ввода/вывода (**I/O Device Tags**) могут быть экспортированы в виде XML файла на диск и использоваться для последующей загрузки данных в FCX. Эта информация может также быть загружена в другой FCX, имеющий такую же конфигурацию аппаратных средств.

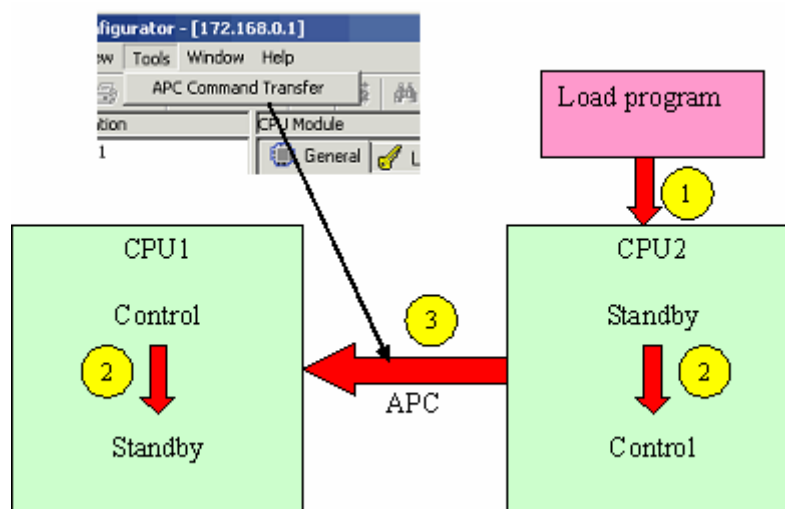
Опция “**Import/Export**” вызывается через выпадающее меню “**FILE**”.

Заметим что, данные не могут быть экспортированы в формат используемый **Logic Designer**. В тоже время они могут быть скопированы и занесены в электронные таблицы и далее форматированы для других целей.

### 2.2.5.6 Групповое копирование программ (APC - All Program Copy).

Команда группового копирования осуществляет копирование содержимого памяти одного CPU в другой внутри FCN. Специфика этого процесса заключается в том, что производится копирование из управляющего CPU (**Control CPU**) в резервный (**Standby CPU**). Эта команда не используется в FCJ.

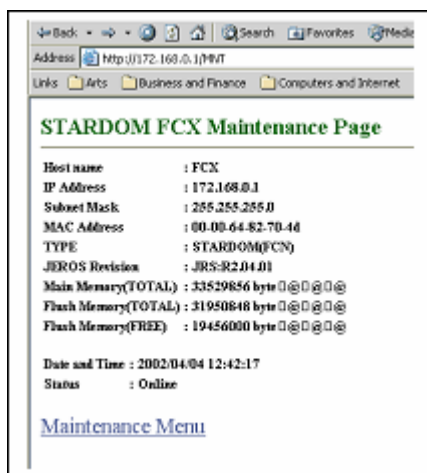
CPU может быть извлечён или отключен для проведения обслуживания, в то время как другой CPU продолжает процесс управления. После установки его на место управление процессом может быть передано ему после выполнения команды группового копирования.



## 2.2.6 Web страница обслуживания (The Maintenance Web Page).

Web страница обслуживания FCX содержит важную конфигурационную информацию о настройках программно аппаратных средств FCX. Она также содержит инструментарий доступа к программам и настроечным данным FCX.

### Доступ к странице обслуживания FCS:

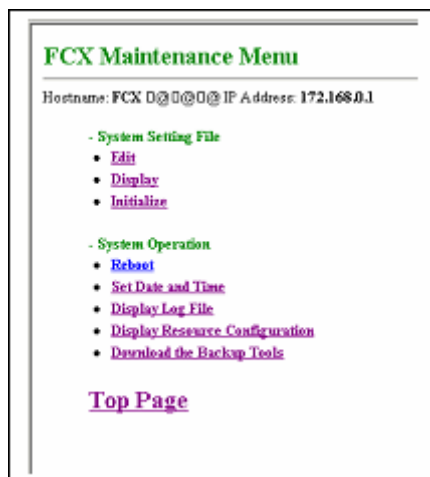


1. Откройте **Web Browser** и введите IP адрес FCX, завершив его последовательностью символов “/MNT”;

Пример: http://172.168.0.1/MNT

2. Введите имя пользователя – “**stardom**”;
3. Введите пароль – “**YOKOGAWA**”;
4. Должна появиться главная страница, которая содержит базовую информацию об FCX;
5. Вызовите “**Maintenance Menu**”

### Становятся доступны следующие функции:



- **Файл системных настроек (System Setting File)** – отображает и позволяет устанавливать системные параметры;
- **Оперирование с системой (System Operation)** – позволяет производить перезагрузку системы и открывает доступ к инструментарию.

Перезагрузка (Reboot).

После редактирования файлов системных настроек необходимо отредактированные файлы перезагрузить. Перезагрузка проводится в режиме обслуживания (Maintenance Mode). Необходимо помнить, что эта операция вызывает останов функций управления FCX. Поэтому необходимо возвращать FCS обратно в online режим (**Online Mode**).

### *Для проведения перезагрузки:*



1. Вызовите страничку **“Reboot”**;
2. Выберите тип перезагрузки (например **“Reboot (Maintenance Mode)”** вызывает переключение FCX в режим обслуживания) и нажмите кнопку **“OK”**;
3. Повторите эту процедуру для возврата в online режим **“Reboot (Online Mode)”** или для изменения IP адреса **“Reboot (IP Address Setting Mode)”** (см. раздел [2.2.4](#)).

### *Инструменты для сохранения (Back-up tools):*

Инструментарий является DOS базированным набором утилит, которые позволяют извлечь из FCX полный объектный файл (Complete Dump) в PC, а также провести обратную операцию его загрузки в FCX.

Для загрузки инструментария в PC вызовите **“Download the Back-up Tools”**. После этого происходит загрузка самоэкстрагируемого файла **“FCXTOOL.EXE”**. Загрузите его в рабочую папку, после чего запустите его на исполнение. В результате из него извлекаются утилиты:

**“FcxBackup.bat”** – извлечение объектного файла

**“FcxRestore.bat”** - загрузка объектного файла.

Для запуска Back-up утилиты запустите командное окно CMD через команду “Выполнить” (“RUN”) в пусковом меню Windows. В командном окне, указав путь к рабочей папке и имя исполняемого файла:

```
fcxbackup <ip_address>
Например: fcxbackup 172.168.0.1
```

запустите его на исполнение.

Аналогично запускается Restore утилита.

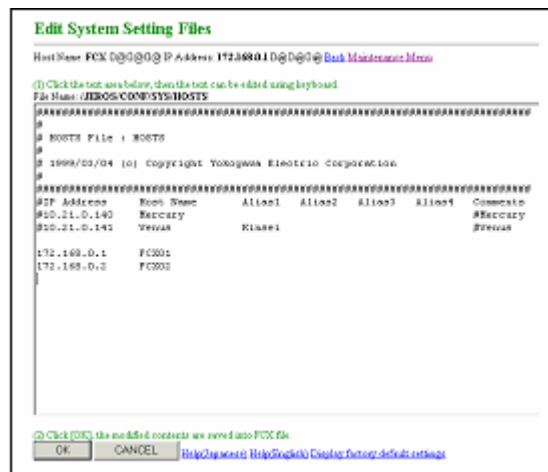
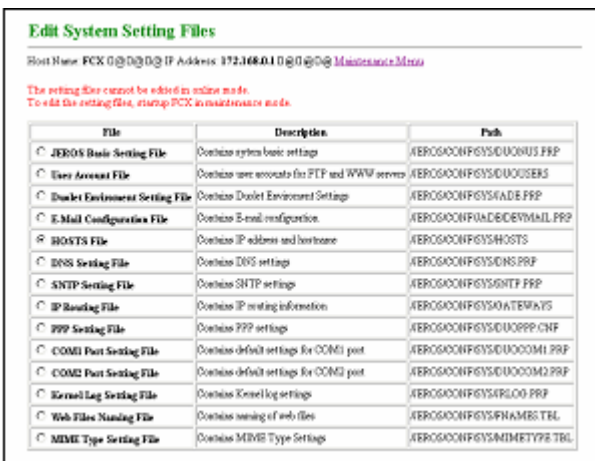
### Hostfile Setting

Более удобно и быстрее обращаться к FCX не по IP адресам, а по хост имени (“**host name**”). Это возможно когда установлены коммуникации между FCX (см. раздел [2.6.8](#)).

Хост имена специфицируются в “**Hosts file**” FCX. В этом файле специфицируются имена хостов и IP адреса всех FCX в сети. Этот процесс должен быть повторен для всех FCX.

Процедура создания имён хостов следующая:

1. Перезагрузите FCX в **Maintenance Mode** (редактирование **Hosts file** не может быть выполнено **Online Mode**);
2. Выберите “**EDIT**” под “**System Setting File**” в **FCX Maintenance Menu**;
3. Выберите “**HOSTS File**” и нажмите кнопку “**OK**”;
4. Отредактируйте страничку **Hosts file**, путём введения имени хоста и IP адреса для каждого FCX.
5. Нажмите кнопку “**OK**” и вернитесь в “**Maintenance Menu**”.
6. Перезагрузите в **Online Mode**.



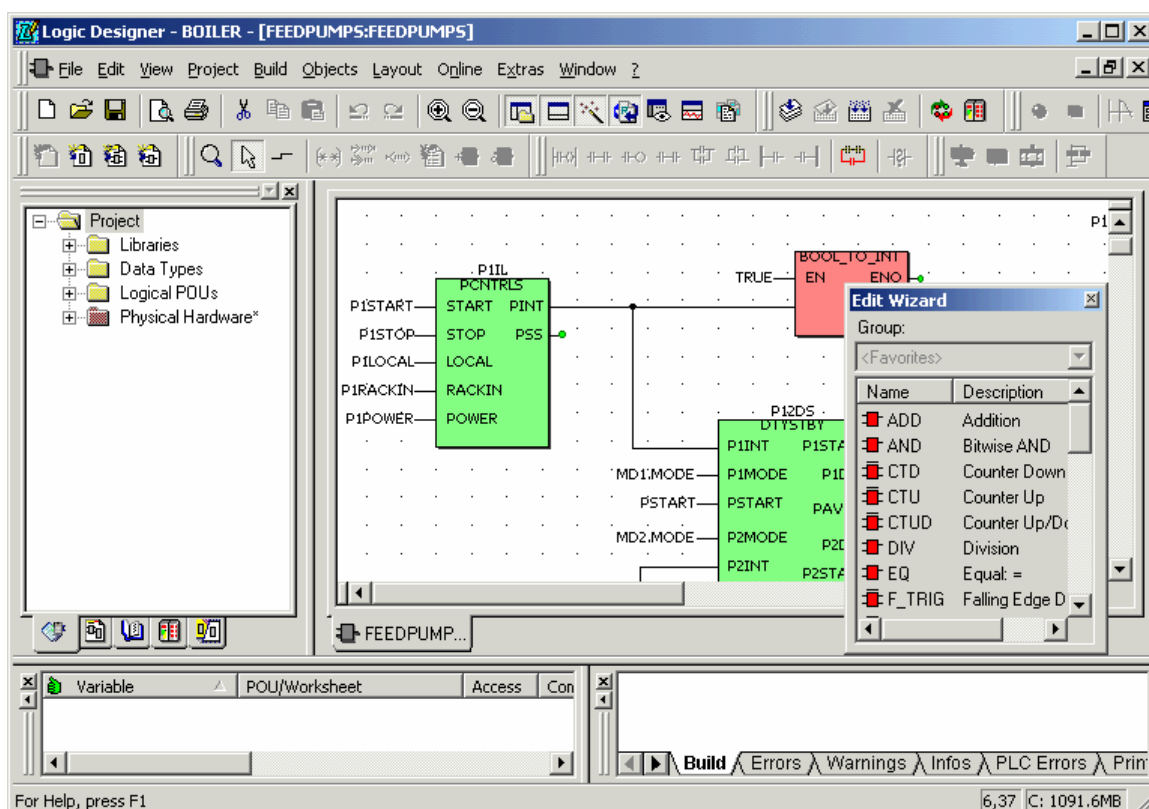
## 2.3 Функции Logic Designer.

### 2.3.1 Обзор.

**Logic Designer** это программирующий интерфейс для FCX. Он имеет аналогичный Explorer интерфейс с заглавной папкой “**Project**”. Под ней располагаются папки представляющие функциональные секции проекта.

Основными папками проекта являются:

- “**Libraries**” – папка, в которой регистрируются все функциональные библиотеки;
- “**Data Types**” – папка, в которой регистрируются все типы данных и структуры;
- “**Logical POU’s**” – папка, в которой размещены все программы и функциональные блоки;
- “**Physical Hardware**” – содержит конфигурации аппаратуры. Каждая конфигурация включает в себя FCX, с набором задач, модулями ввода/вывода и глобальными переменными.

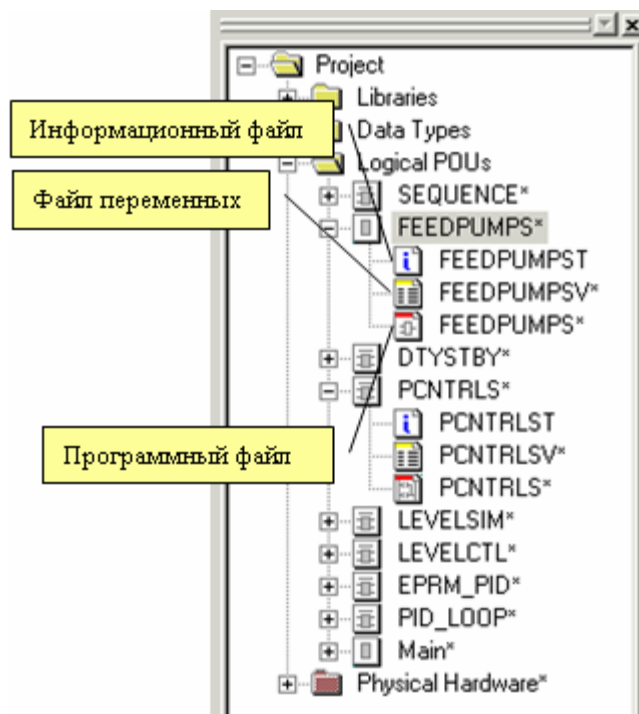


### 2.3.2 Блоки организации программ (Program Organization Units) (POU).

Программы инкапсулированы в блок организации программ **Program Organization Unit (POU)**. POU может быть собран из комбинации пяти языков программирования в соответствии с IEC-61131 и может быть обозначен как функция (**Function**), функциональный блок (**Function Block**), или программа (**Program**).

- **Function Block** - Если POU назначен как функциональный блок, он может быть использован другими POU, тем самым обеспечивается многоуровневая вложенность функций. Такой POU может быть использован многократно по мере необходимости в других POU. Они могут быть также запаролены с целью защиты от несанкционированного редактирования.
- **Function** – Функция подобна функциональному блоку за исключением того, что она не может иметь внутренних переменных (**data items**). Она имеет только входы и выход результата. Примером функции может быть функция сложения (**ADD**).
- **Program** – Для того чтобы POU выполнялась в FCX, она должна быть назначена программой (**Program**). Программа затем назначается задачей в FCX и размещается в папке “**Physical Hardware**”. Функциональный блок POU никогда не исполняется сам по себе. Для исполнения он должен быть размещён как функциональный блок в программу POU.

POU создаются и сохраняются в папке “Logical POU” как показано ниже:





---

*Каждый POU содержит три файла:*

- **Информационный файл (Information file)** (“имяГ”) – текстовый файл с комментариями пользователя о функции;
- **Файл переменных (Variables file)** (“имяV”) – содержит все переменные, используемые в программе;
- **Программный файл (Program file)** (“имя”) – файл с программой.

где: “имя” – имя POU

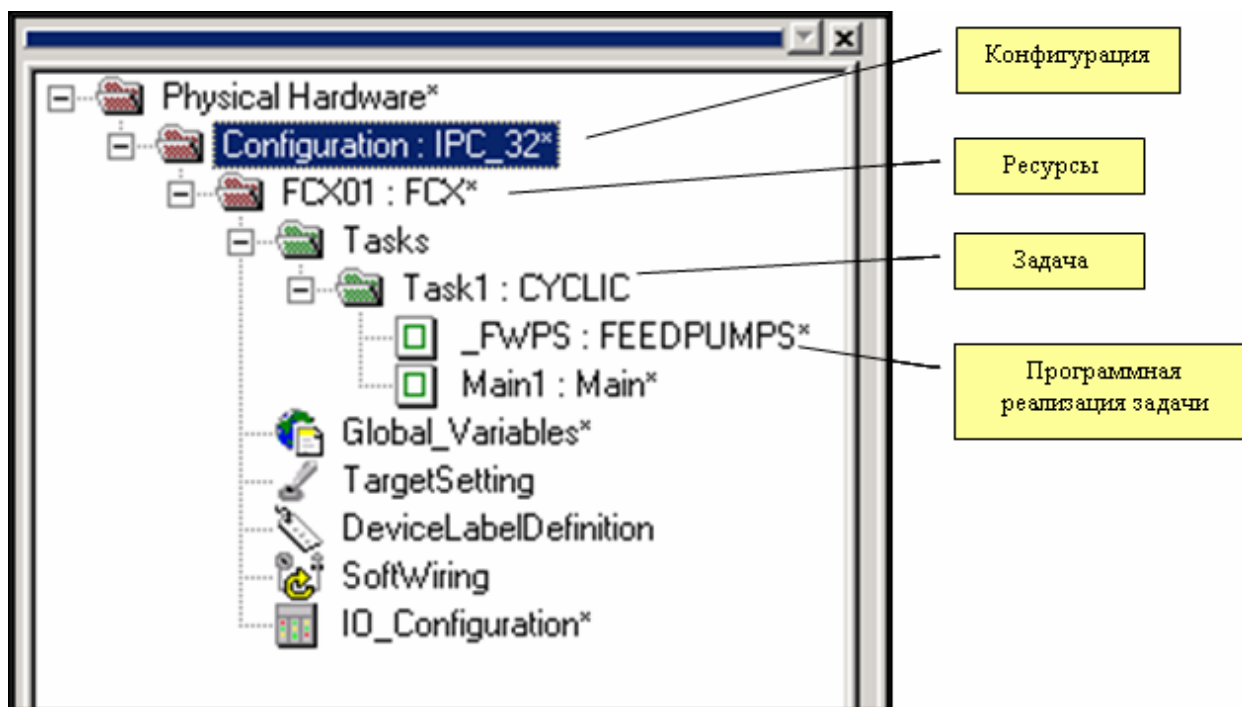
### 2.3.3 Задачи (Tasks).

Экземпляр ROU, назначенного как **Program** в FCX, обозначается как задача, при этом говорят, что ROU является *наследником* по отношению к задаче. Она присваивает следующие атрибуты программе:

- Какой FCX её исполняет. Программа может исполняться в более чем одном FCX;
- Как часто она должна исполняться;
- Каков её приоритет;
- Тип задачи;
- Как много стековой памяти требуется для её исполнения.

Перед тем как ROU может быть присвоена задаче в качестве наследника, задача должна быть создана. Задачи для каждого FCX создаются при его конфигурировании. Задачи, FCX и Конфигурации соотносятся между собой следующим образом:

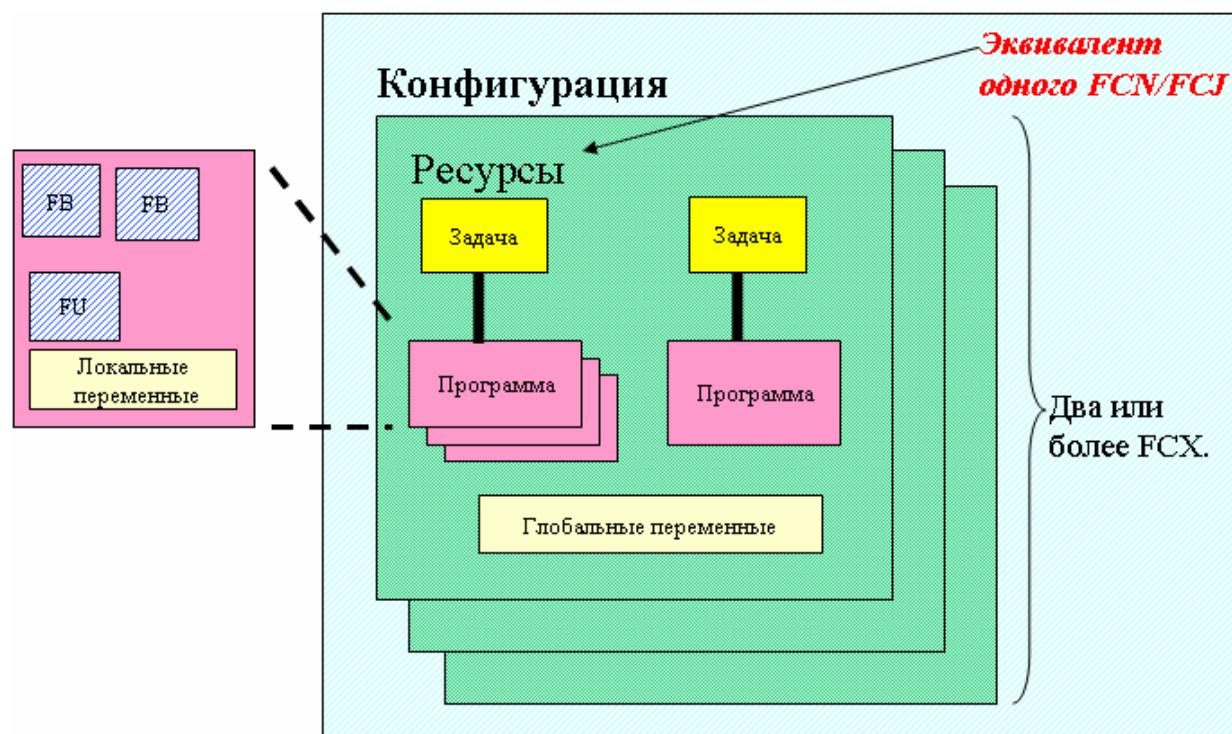
- “**Physical Hardware**” – головная папка для хранения ресурсов проекта;
- “**Configurations**” – следующий уровень папок включающие в свой состав FCX;
- “**Resource**” – внутри Configuration содержит несколько FCX, другие устройства, называемые как ресурсы (**Resources**);
- “**Tasks**” – внутри каждого FCX есть папка “Tasks” содержащая набор задач;
- “**Program Instances**” – внутри каждой задачи может находиться несколько реализаций этой задачи (**Program Instances**), т.е., программ ROU. Когда программа ROU связывается с задачей, она называется *наследником* (“**instantiated**”). Программа ROU может быть связана с более чем одной задачей в FCX, и несколько различных программ могут быть связаны с одной задачей.



Задача может быть одним из трёх типов:

- **Default** – по умолчанию. Работает так быстро, как это возможно, отсутствует заранее определённая частота исполнения;
- **Cyclic** – циклический. Работает с заранее определённой частотой исполнения. Однако она завершает своё исполнение во времени в зависимости от назначенного ей приоритета. Это детерминированные (**deterministic**) задачи.
- **System** – системный. Выполняются по мере наступления системных событий, таких как холодный/тёплый/горячий старт, ошибка обмена и т.д.

Эти типы поддерживаются возможностями многозадачной операционной системы FCX. Более подробно об этом см. раздел [2.2.3](#).



### 2.3.4 Оперирование.

В настоящем разделе описаны способы оперирования в **Logic Designer**, доступ к которым осуществляется через выпадающие меню:

- **“BUILD”** – компилирование программ и перестройка проектов.
- **“ONLINE”** – загрузка программ и их запуск в online режим управления FCX.

Этот раздел также описывает другие связанные с **Logic Designer** операции, такие как:

- **“Saving Tuning Parameters”** – сохранение настроечных параметров;
- **“Online downloading using Patch POU”** – online загрузка с использованием **Patch POU**;
- **“Project backups”** – сохранение проекта.

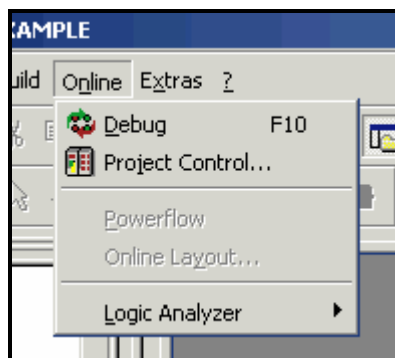
#### 2.3.4.1 Меню “BUILD”.

Меню **“BUILD”** включает в себя следующие функции:

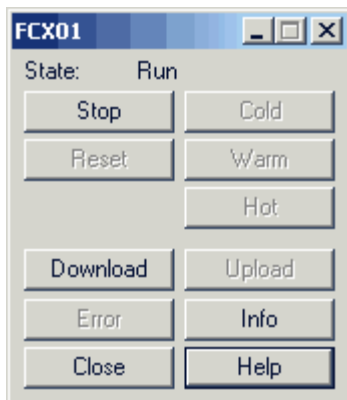
- **“Make”** – компилирует изменяемые в проекте программы и создаёт необходимые при загрузке в FCX исходные файлы;
- **“Patch POU”** – функция загрузки программ в Online режиме, которая компилирует и загружает модифицируемые программы без останова FCX. Существуют некоторые ограничения на использование этой функции, см. п. [2.3.4.3](#);
- **“Rebuild Project”** – перекомпиляция всех программ, включая декларированные переменные, лицензии и все дополнительные элементы. Это необходимо после проведения изменений в ресурсной части проекта. Например, если IP адрес FCX изменён или загружена дополнительная лицензия, то проект требуется перестроить;
- **“Build Cross References”** – создание полного перечня тагов со списком ссылок на них.

#### 2.3.4.2 Меню “ONLINE”

Меню **“ONLINE”** включает в себя следующие функции:

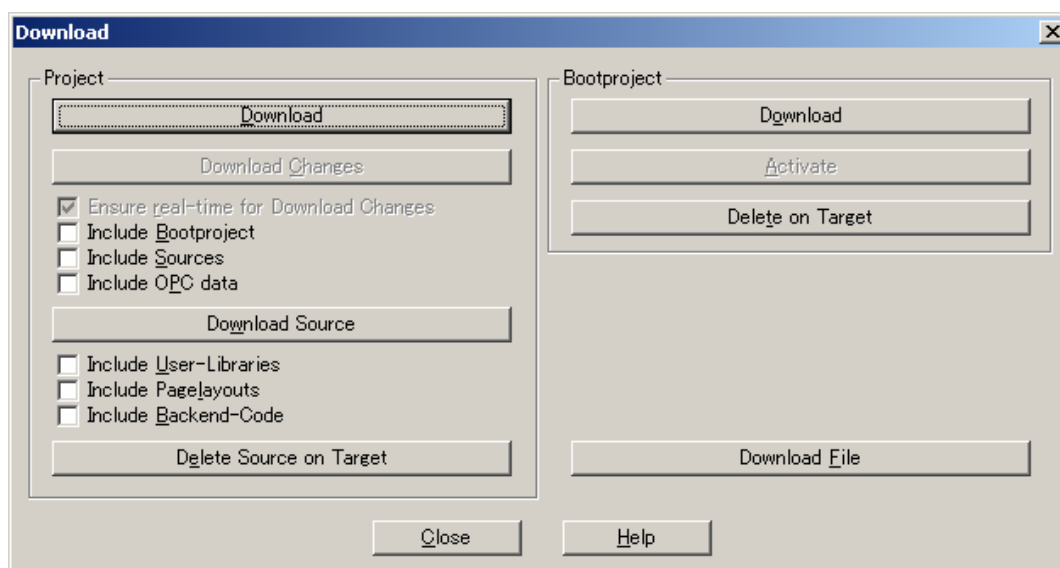


- **“Debug”** – переключает FCX в отладочный режим. Для работы в отладочном режиме FCX должен быть запущен и связан с PC. При запущенном FCX данные могут отображаться и изменяться из программных таблиц **Logic Designer**;
- **“Project Control”** – управление режимами работы FCX.



- **“Stop”** – останавливает исполнение программ в FCX;
- **“Cold”** – выполняет холодный старт FCX. Все данные инициализируются;
- **“Warm”** – выполняет тёплый старт FCX. Все неподдерживаемые данные инициализируются;
- **“Hot”** – выполняет горячий старт FCX. Данные не инициализируются;
- **“Reset”** – удаляет загруженный в FCX проект;
- **“Download”** – загружает программы в FCX;
- **“Upload”** – выгружает из FCX проект в виде сжатого исходного файла. Более подробно см. п. [2.3.4.5](#);
- **“Error”** – печать списка ошибок произошедших при оперировании с FCX;
- **“Info”** – выдаёт системную информацию о работе с PLC.

Функция **Download** имеет следующие опции:



Существуют две основные области, куда проект может быть загружен:

- **Project** – область текущего исполняемого проекта в основной памяти (**Main Memory**) (см. раздел [2.2.2](#)). Если выполняется загрузка в эту область, FCX приостанавливает выполнение текущего управления на время выполнения перезагрузки, а затем осуществляется рестарт с новой программой. Заметим, что рестарт должен быть выполнен вручную путём вызова процедуры холодного, тёплого или горячего старта.
- **Bootproject** - область внутри Flash RAM (см. раздел [2.2.2](#)). Загрузка в эту область не останавливает выполнение текущего управления, но новые программы не будут активизированы, пока не будет выключено и затем включено питание FCX, или не нажата кнопка “**ACTIVATE**”.

В таблице перечислены опции:

ОПЦИЯ	ФУНКЦИЯ
“Project”	
<input type="checkbox"/> “Download”	Загружает программы в выбранный FCX.
<input type="checkbox"/> “Download Changes”	В настоящее время не действует.
“Ensure real-time for Download Changes”	В настоящее время не действует.
“Include Bootproject”	Загружает в FCN/FCJ загрузочный проект одновременно с исполняемым.
“Include Sources”	Сжимает проект и загружает сжатые исходные файлы в FCN/FCJ вместе с исполняемым проектом. Сжатые исходные файлы могут быть использованы как резервная копия текущего исполняемого проекта.
“Include OPC data”	Включает загрузку данных OPC сервера в процесс загрузки. Данные загружаются в виде CSV файла вместе с проектом. В файле перечислены переменные доступные к использованию OPC Server.
<input type="checkbox"/> “Download Source”	Сжимает проект и загружает сжатые исходные файлы в FCN/FCJ. Сжатые исходные файлы могут быть использованы как резервная копия текущего исполняемого проекта.
“Include User-Libraries”	Включает все библиотеки пользователя в сжатые исходные файлы проекта.
“Include Pagelayouts”	Включает все pagelayouts в сжатые исходные файлы проекта.
“Include Backend-Code”	
<input type="checkbox"/> “Delete source on target”	Удаляет исходные файлы проекта из FCN/FCJ (активизируется, только если проект был загружен ранее).
“Bootproject”	
<input type="checkbox"/> “Download”	Загружает загрузочный проект в FCN/FCJ
<input type="checkbox"/> “Activate”	Активизирует загруженный загрузочный проект. Если загрузочный проект был загружен, он не выполняется до тех пор, пока не будет активизирован.
<input type="checkbox"/> “Delete on target”	Удаляет загрузочный проект из FCN/FCJ (только активизированный).
“Download File”	Вызывает поисковый диалог, в котором вы



	можете выбрать один или несколько файлов для загрузки их в контроллер.
--	--

### 2.3.4.3 Online загрузка с использованием Patch POU.

Корректировка (**Patch**) POU означает что изменения, которые вы выполнили в проекте, компилируются, соответствующие коды генерируются и загружаются автоматически в контроллер в один этап. В течение этого процесса контроллер не останавливается, т.е. выполняемый код не удаляется из контроллера до тех пор, пока не завершатся операции компиляции и загрузки изменений.

**Примечание:** Опция **Patch POU** действует, только если вы включите соответствующий режим в **offline** режиме путём вызова иконки **“Debug on/off”** на панели инструментов.

Команда **“Patch POU”** может использоваться в двух целях:

- **Корректировка обнаруженных ошибок** – если вы обнаружили программные ошибки в online режиме и переключили ключ отладки в offline для её устранения, вы можете использовать **“Patch POU”** для компиляции и загрузки исправлений.
- **Развитие базового проекта** – В некоторых случаях удобнее разрабатывать проект, не останавливая функционирования контроллера. После первоначального создания и загрузки проекта вы можете дополнять массив программ (т.е. программную оболочку) путём редактирования спецификации POU, исполняемой контроллером, с последующим исполнением Patch процедуры.

В обоих случаях изменения загружаются автоматически, не останавливая процесса, так что вы можете видеть результат немедленно после переключения в online режим. При этом должны соблюдаться некоторые правила и ограничения, а также должен быть предусмотрен соответствующий резерв памяти.

### **Как корректировать POU:**

1. Убедитесь, что память, зарезервированная POU, достаточна для корректировки данных (см. далее). Если недостаточна, команда **“Patch POU”** не может быть выполнена. В этом случае вам придётся использовать команду **“Make”** для перекомпиляции изменяемого проекта.
2. Убедитесь, что изменяемая рабочая спецификация программы находится в активном окне.
3. Убедитесь, что рабочая спецификация программы находится в offline режиме.
4. Отредактируйте рабочую спецификацию или исправьте ошибки. Изучите правила и ограничения, перед тем как приступить к редактированию.
5. Откройте подменю **“Build”** и выберите пункт меню **“Patch POU”** или щёлкните по соответствующей иконке в меню инструментов.
6. Процесс компиляции запускается и его ход отображается в окне сообщений.

### Требования к резервированию памяти:

При корректировке система выделяет фрагмент памяти пользователя для корректируемой POU в объёме заданном для этих программ. Если памяти для корректировки недостаточно, выдаётся сообщение об ошибке и процесс корректировки прекращается.

По умолчанию 20% всей наличной памяти в основной памяти контроллера (Main Memory) резервируется для последующих корректировок. Величина резервируемой памяти может задаваться глобально через установку опции при распределении ресурсов. Кроме того, размер резервируемой памяти может быть установлен для каждой POU индивидуально. Для этого необходимо выбрать POU в соответствующем дереве программ и щелчком правой кнопки мыши вызвать контекстное меню, в нём обратиться к свойствам (“**Properties**”). Более подробно можно посмотреть “**Help Manual**” (как описано ниже).

### Если “Patch POU” не работает:

Если команда “**Patch POU**” не выполняется по причине не выполнения одного из правил или ограничений необходимо перекомпилировать проект при помощи процедуры “**Make**” и инициировать новую загрузку. Когда вы проводите перекомпиляцию, распределение памяти производится заново, при этом резерв памяти, использованный для предыдущих корректировок, высвобождается.

Когда вы работаете с различными ресурсами, каждая реализация объекта (т.е. ресурс) корректируется отдельно путем переключения соответствующей рабочей спецификации из online режима в offline, изменения кода и выполнения “Patch POU” команды.

### Дополнительная информация:

Для получения более детальной информации о функции Patch POU, обратитесь к меню “**Help**” Logic Designer:

? → “**Contents**” (запускает окно Help) → “**Editing and developing a project**” → “**Compiling a project**” → “**How to compile a project**” → “**Patch POU**”

#### 2.3.4.4 Сохранение параметров настройки.

В терминах IEC61131 параметры настройки называются ‘**retained data**’, детальная информация о них дана в разделе [2.2.2](#). Этот раздел описывает процедуры сохранения и восстановления параметров настройки.

Параметр настройки это переменная, которая имеет своё значение атрибута **Retain**.

- Если он установлен, то данные хранятся в памяти;
- Они сохраняются в основной памяти (Main Memory) (энергозависимой памяти), однако если в Resource Configurator они помечены установленным признаком “**enable Hard-backup for retained data**”, они поддерживаются также и в Data Memory.

Использование установок глобальных переменных позволяет сохранять во Flash Card, и восстанавливать их оттуда по мере необходимости:

GS_RETAIN_SAVE_SW	Сохраняет данные во Flash RAM card (установить в состояние TRUE)
GS_RETAIN_RESTORE_SW	Восстанавливает данные из Flash RAM card (установить в состояние TRUE)
GS_RETAIN_SAVE_PROGRESS	Сохранить текущее состояние (Integer)
GS_RETAIN_RESTORE_PROGRESS	Восстановить текущее состояние (Integer)

Совсем не сложно сохранить параметры настройки и в PC.

#### 2.3.4.5 Восстанавливаемые проекты (Backing-up projects).

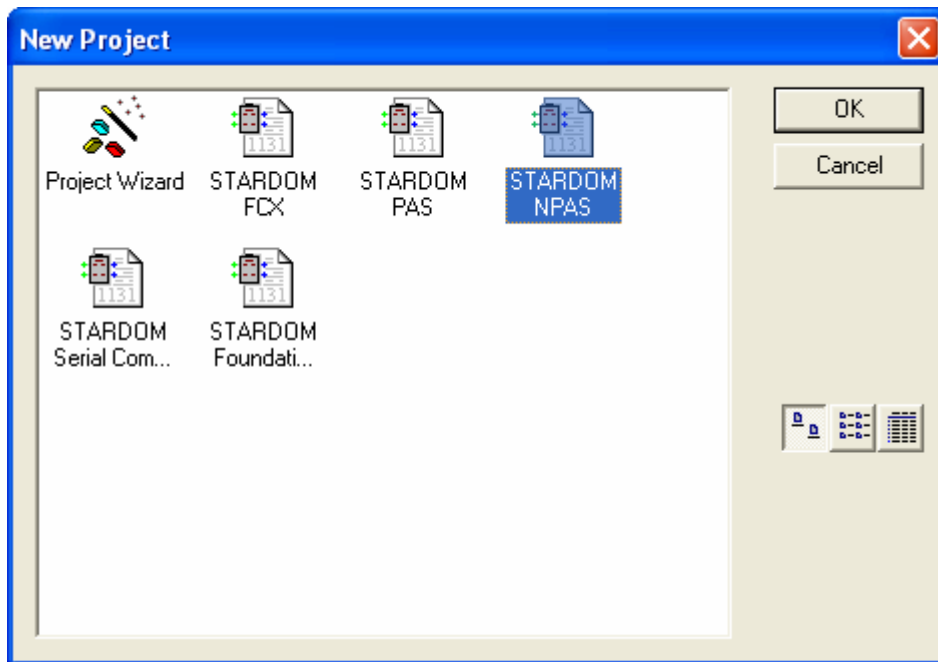
При восстановлении проекта полезно использовать функцию сжатия проекта и возможность загрузки и выгрузки сжатого файла в или из контроллера.

- Для того, что бы сжать проект выберите опцию “**Save Project As/Zip Project As**” в выпадающем меню “**FILE**”. Выберите “**Save As Type**” = “**Zipped Project File, \*.wzt**”, при этом создаётся файл сжатого проекта. Он может быть сохранён в любом месте как восстановленный (выгруженный из контроллера) проект.
- Для того чтобы получить несжатый проект выберите “**Open Project/Unzip Project**” в выпадающем меню “**FILE**”. Выберите “**Zipped Project File type**”. При этом проект из сжатой формы разворачивается в обычно используемую форму проекта и может быть записан в формате текущего открытого проекта или вновь создаваемого.
- **Источник загрузки (Download Source)** – проект может быть сжат, загружен в FCX и записан во Flash Card через диалог загрузки (“**Download Source**”). Это эффективно гарантирует портируемость восстановленного проекта, на тот случай, если кому-то понадобится его прочитать для дополнительного редактирования.
- **Выгрузка (Upload)** – сжатый проект, который был загружен в FCX, может быть скопирован в PC с использованием процедуры “**Upload**” в режиме Online. При этом образуется копия проекта в формате несжатого проекта.
- **Поддержка Back-up процедур (Maintenance Back-up)** – back-up (копирование объектного кода) и загрузка (обратная операция) выполняется с использованием “**Maintenance web Page**” как описано в разделе [2.2.6](#). Эти функции дают возможность получить в PC полный образ памяти контроллера в объектном виде (**dump**) и при необходимости его перезагрузить обратно в контроллер.

## 2.3.5 Процедуры.

### Создание нового проекта:

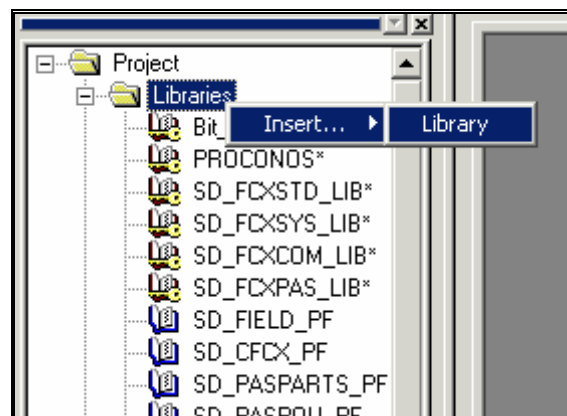
1. Выберите опцию “**NEW PROJECT**” в выпадающем меню “**FILE**”, при этом открывается окно генератора проекта;



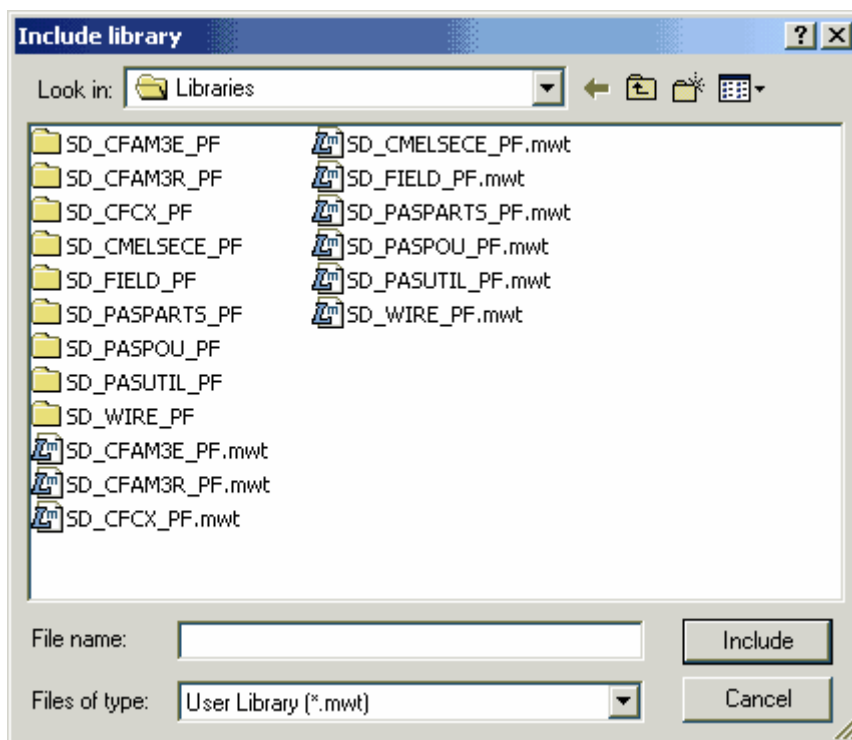
2. Выберите “**STARDOM NPAS**”. Выбор других из двух опций “**STARDOM PAS**” или “**STARDOM FCX**” создает проект без библиотеки функциональных блоков Yokogawa, которая может быть впоследствии добавлена вручную.

### Добавление библиотеки:

1. Щёлкните правой кнопкой мыши по папке “**Libraries**”, затем выберите “**Insert...**”→“**Library**” (возможно также через меню “**Project**”);



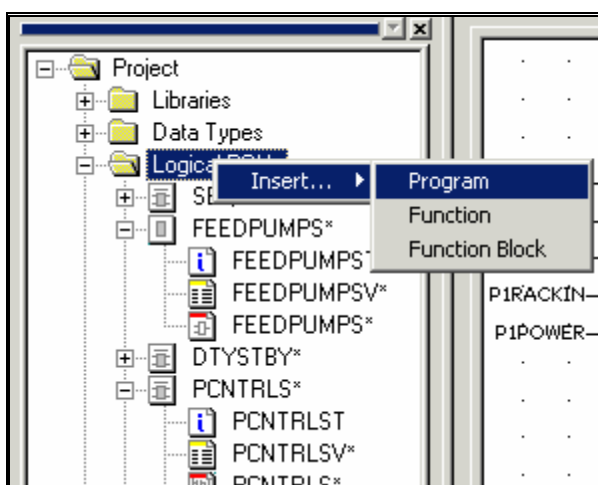
2. Возникает следующий диалог:



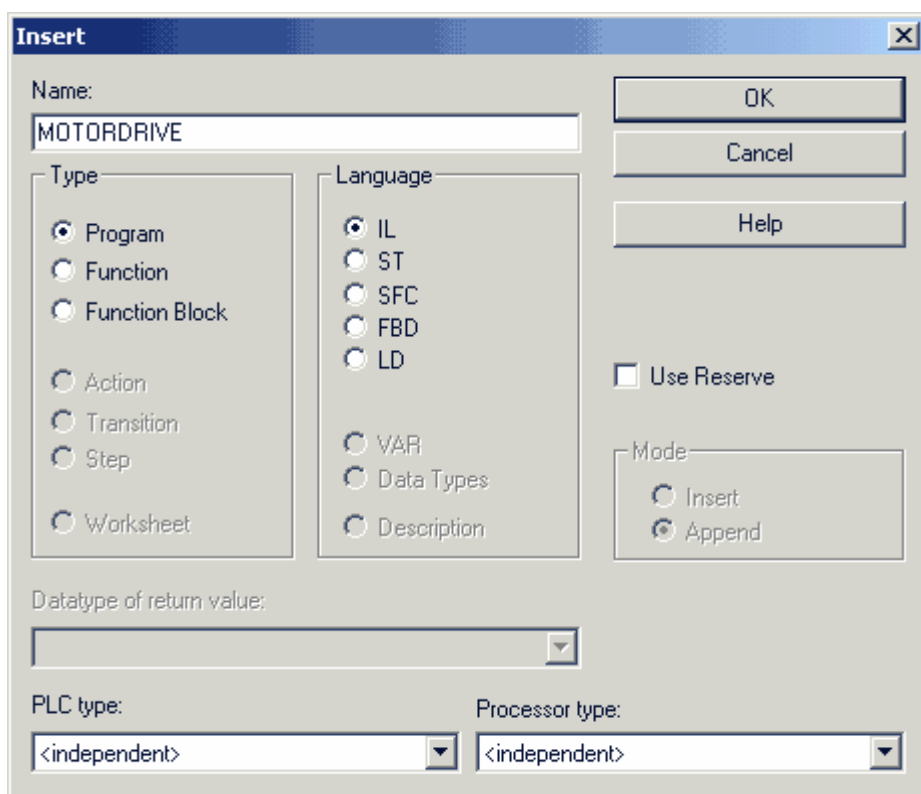
3. Выберите требуемую библиотеку (файл с расширением .mwt) и нажмите **“Include”**.

### Создание нового POU:

1. Щёлкните правой кнопкой мыши по папке **“Logical POU”**, затем выберите **“Insert...”** (возможно также через меню **“Project”**);
2. Выберите один из трёх типов POU;



3. Возникает следующий диалог:



**Insert**

Name:

OK  
Cancel  
Help

Type

- Program
- Function
- Function Block
- Action
- Transition
- Step
- Worksheet

Language

- IL
- ST
- SFC
- FBD
- LD
- VAR
- Data Types
- Description

Use Reserve

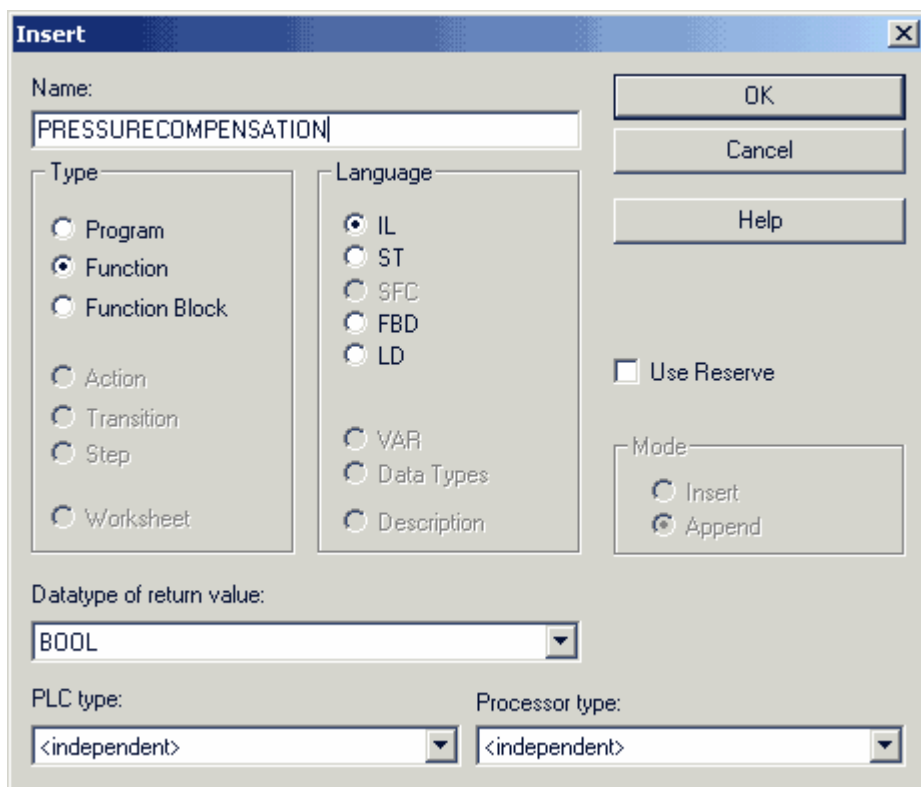
Mode

- Insert
- Append

Datatype of return value:

PLC type:  Processor type:

4. Введите имя для POU (без пробелов), и выберите язык описания.
5. Если выбрана **“Function”**, то язык типа **“SFC”** не может быть выбран. Кроме того, должен быть специфицирован тип данных результата **“Datatype of return value”**.



**Insert**

Name:

OK  
Cancel  
Help

Type

- Program
- Function
- Function Block
- Action
- Transition
- Step
- Worksheet

Language

- IL
- ST
- SFC
- FBD
- LD
- VAR
- Data Types
- Description

Use Reserve

Mode

- Insert
- Append

Datatype of return value:

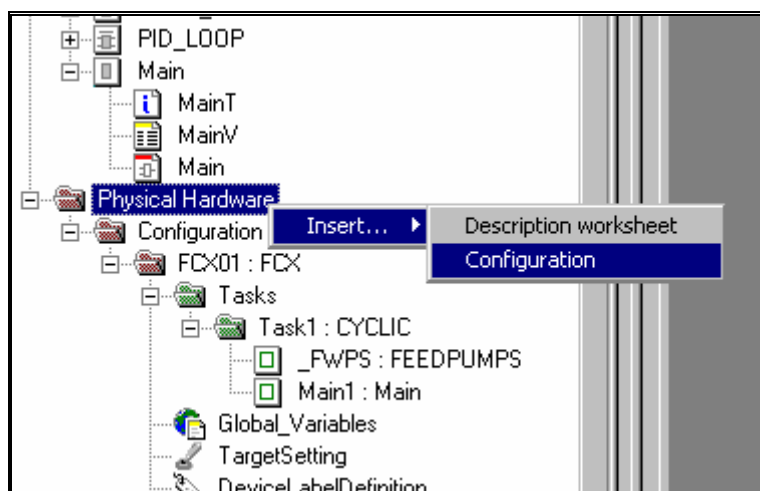
PLC type:  Processor type:

6. Рекомендуется чтобы “**PLC type**” и “**Processor type**” были специфицированы как “**<independent>**” для обеспечения портбельности конфигурации.

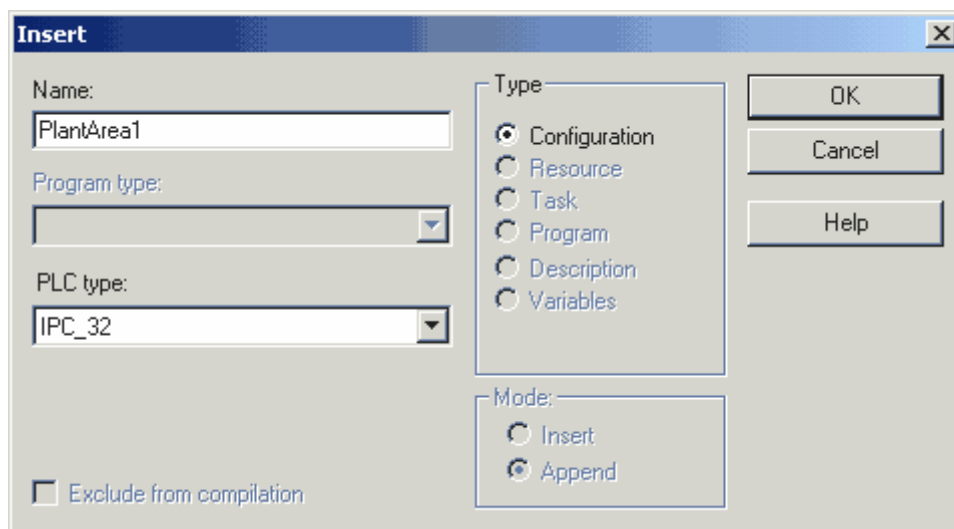
Все эти параметры могут быть изменены в любой момент. Для этого достаточно щёлкнуть правой кнопкой мыши по POU и выбрать его свойства “**Properties...**”.

### Создание конфигурации:

1. Щёлкните правой кнопкой мыши по папке “**Physical Hardware**”, затем выберите “**Insert...**”→“**Configuration**”;

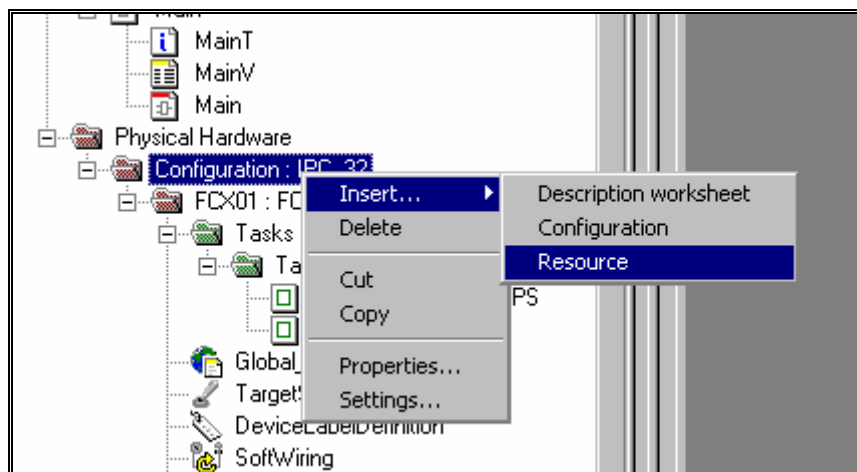


2. Введите имя для конфигурации “**Name:**”.

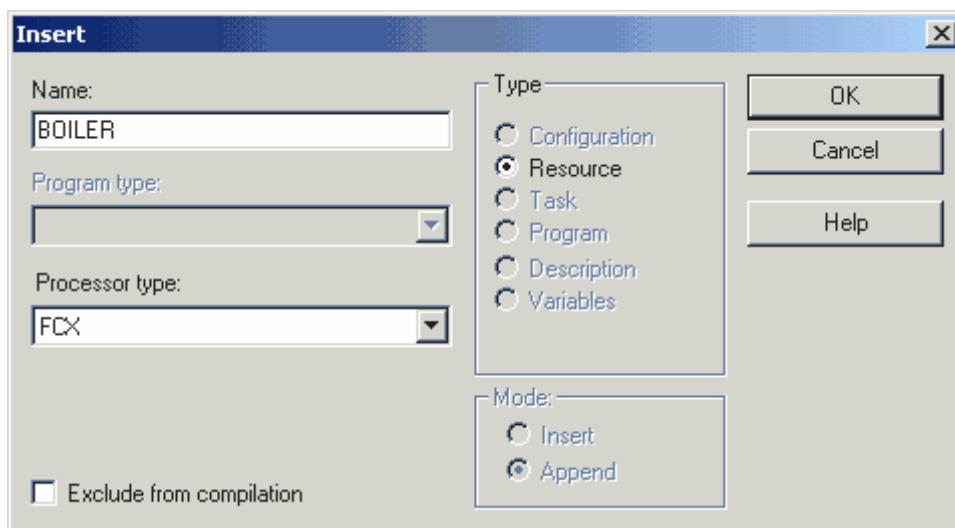


### Создание ресурса:

1. Щёлкните правой кнопкой мыши по папке “**Configuration**”, затем выберите “**Insert...**”→“**Resource**”;



2. Введите имя для ресурса “**Name:**” (Тип процессора всегда FCX).



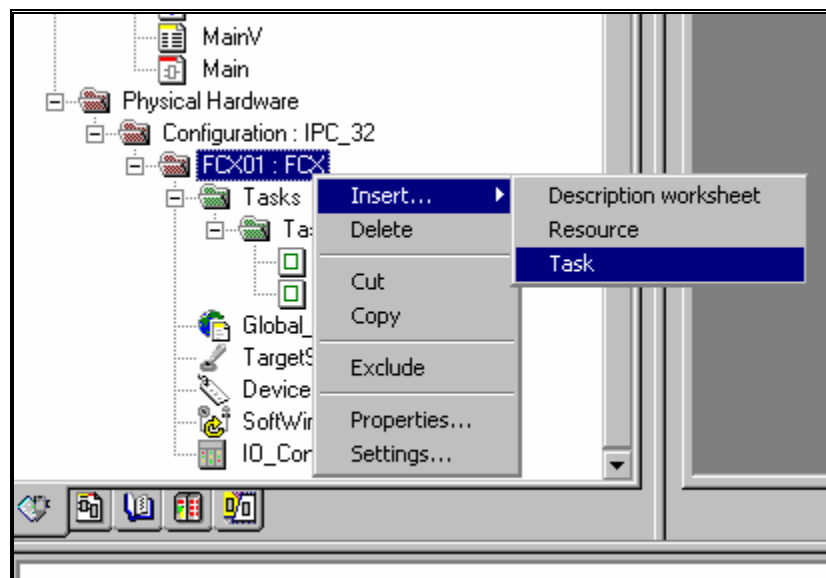
Эти параметры могут быть изменены после создания FCX. Для этого достаточно щёлкнуть правой кнопкой мыши по ресурсу и выбрать его свойства “**Properties...**”.

Остальные параметры для FCX могут быть доступны путём вызова опции “**Settings...**”.

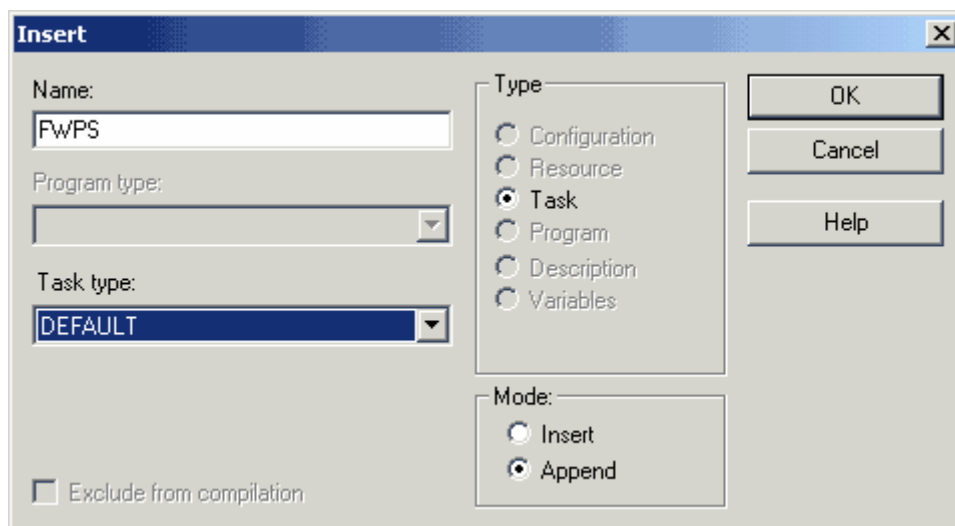
### **Создание задачи:**

1. Щёлкните правой кнопкой мыши по папке “**Resource (FCX)**”, затем выберите “**Insert...**” → “**Task**”;



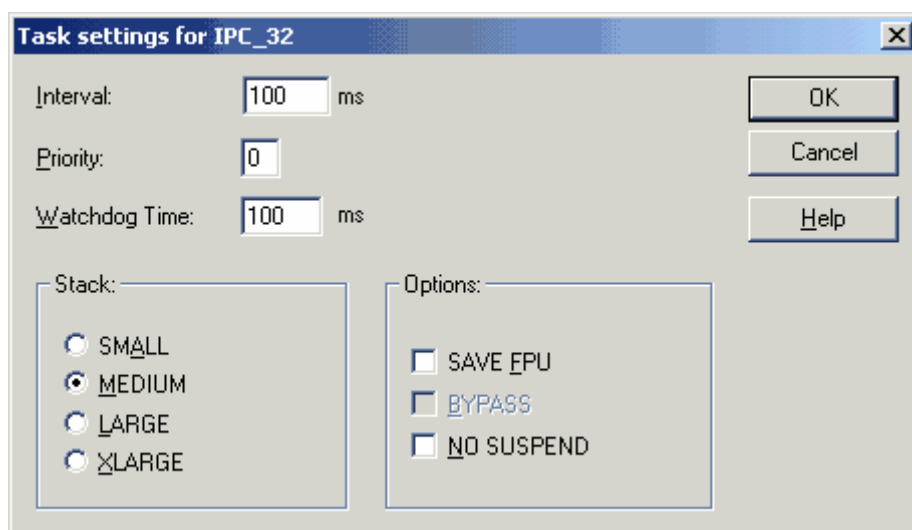


2. Введите имя для задачи “Name:” и её тип “Task type:”;



Эти параметры могут быть изменены после создания задачи. Для этого достаточно щёлкнуть правой кнопкой мыши по задаче и выбрать её свойства “**Properties...**”.

3. Введите приоритетность “**Priority:**” и период сканирования “**Interval:**”;



Эти параметры могут быть изменены после создания задачи. Для этого достаточно щёлкнуть правой кнопкой мыши по задаче и выбрать её параметры “Settings...”.

Приоритетность обсуждалась в разделе [2.2.3](#), её величина может быть в пределах от 0 до 15.

---

## 2.4 Данные.

### 2.4.1 Типы данных.

Типы данных описаны подробно в “IEC61131-3. Programming course”. Краткий перечень дан ниже:

КАТЕГОРИЯ	ТИП ДАННЫХ	ОПИСАНИЕ
Integer Numbers (Целые)	SINT	Short Integer (Короткое целое)
	INT	Integer (Целое)
	DINT	Double Integer (Целое двойной длины)
	USINT	Unsigned short Integer (Короткое целое без знака)
	UINT	Unsigned Integer (Целое без знака)
	UDINT	Unsigned double Integer (Целое двойной длины без знака)
Real Numbers (с плавающей запятой)	REAL	Real numbers (с плавающей запятой)
Time and Date (Время и Дата)	TIME	Time duration (Время и Дата)
Strings (Строковые)	STRING	Character strings (Строка символов)
Bit Strings (Битовые строки)	BOOL	Bit string of 1 bit (Битовая строка из 1 бита)
	BYTE	Bit string of 8 bit (Битовая строка из 8 бит)
	WORD	Bit string of 16 bit (Битовая строка из 16 бит)
	DWORD	Bit string of 32 bit (Битовая строка из 32 бит)
Boolean (Логические)	BOOL	Switch (Fals/True)

## 2.4.2 Структуры данных.

**Структуры данных (Data Structures)** означают набор инкапсулированных данных ранее описанных типов в одном объекте. Таким образом, функциональный блок содержит набор элементов данных, которые определяются как часть структуры данных блока. Например: NPAS\_POU, NPAS\_PVI блоки, имеют следующую структуру данных:

```
TYPE
    (*PVI*)
    SD_PPARAM_PVI      :   STRUCT
        MODE           :   DWORD;
        ALRM           :   DWORD;
        AF             :   DWORD;
        AOF            :   BOOL;
        PV             :   CData_REAL;
        PVCAL          :   BOOL;
        RAW            :   CData_REAL;
        SUM            :   SD_PSUM_DEF;
        HH             :   REAL;
        LL             :   REAL;
        PH             :   REAL;
        PL             :   REAL;
        VL             :   REAL;
        PVP            :   REAL;
        DPV            :   CData_REAL;
        ADFS           :   DWORD;
    END_STRUCT;
END_TYPE
```

Эта структура данных определяет все параметры внутри блока. Ниже описана структура данных, используемая внутри структур данных функциональных блоков NPAS\_POU.

Кроме того, все процессы ввода/вывода ассоциированы со структурами данных. Более подробно смотри ниже.

Некоторые из элементов структуры данных в свою очередь содержат также структуру данных. Например: PV представляет собой структуру данных именованную как **CData\_REAL** и содержит набор величин показанный ниже:

```
TYPE
    CData_REAL        :   STRUCT
        Value         :   REAL;
        Status        :   DWORD;
        CInfo         :   DWORD;
        Dummy         :   DWORD;
        SH            :   REAL;
        SL            :   REAL;
        Unit          :   IndUnit;
    END_STRUCT;
END_TYPE
```

Поэтому ссылка на PV в блоке PVI имеет следующий формат:

“Tag.PV.Value”

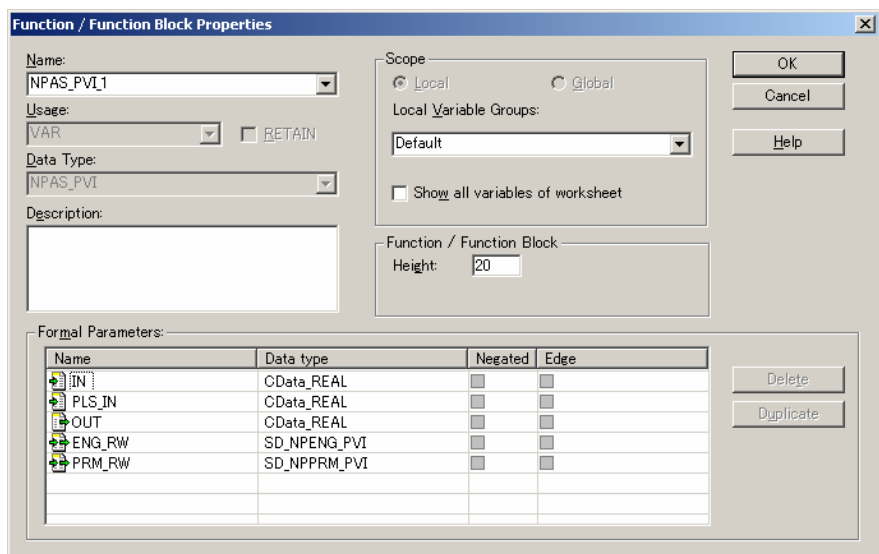
Заметим что “Unit” также декларирован типом данных, не смотря на это, в этом случае, это массив (“ARRAY”), а не структура данных:

```

TYPE
    IndUnit      : ARRAY[0..7] OF BYTE;
END_TYPE
    
```

NPAS POU блоки имеют также входы и выходы, которые являются частным случаем типов данных. Очень важно знать эти типы данных, когда подключаешь переменные к блокам или когда соединяешь блоки между собой.

При конфигурировании это можно определить, посмотрев свойства функционального блока.



*Перечень функциональных блоков NPAS\_POU и ассоциированных с ними структур данных:*

БЛОК NPAS_POU	СТРУКТУРА ДАННЫХ
NPAS_ASTM1	SD_PPARAM_ASTM
NPAS_ASTM2	SD_PPARAM_ASTM
NPAS_AS_H/L/M	SD_PPARAM_AS
NPAS_AVE_C	SD_PPARAM_AVE_C
NPAS_AVE_M	SD_PPARAM_AVE_M
NPAS_BDBUF_R	SD_PPARAM_BDBUF_R
NPAS_BDBUF_T	SD_PPARAM_BDBUF_T
NPAS_CT	SD_PPARAM_CT
NPAS_DLAY	SD_PPARAM_DLAY
NPAS_FFSUM	SD_PPARAM_FFSUM
NPAS_FFSUM_BL	SD_PPARAM_FFSUMBL
NPAS_FOUT	SD_PPARAM_FOUT
NPAS_FUNC_VAR	SD_PPARAM_FUNC_VAR
NPAS_LDLAG	SD_PPARAM_LDLAG
NPAS_MLD	SD_PPARAM_MLD
NPAS_MLD_BT	SD_PPARAM_MLD_BT
NPAS_MLD_PB	SD_PPARAM_MLD_PB
NPAS_ONOFF	SD_PPARAM_ONOFF
NPAS_ONOFF_G	SD_PPARAM_ONOFF_G
NPAS_PG_L30	SD_PPARAM_PG_L30
NPAS_PG_L30BP	SD_PPARAM_PG_L30_BP
NPAS_PID	SD_PPARAM_PID
NPAS_PI_HLD	SD_PPARAM_PI_HLD
NPAS_PVI	SD_PPARAM_PVI
NPAS_P_CLF	SD_PPARAM_PCFL
NPAS_RATIO	SD_PPARAM_RATIO
NPAS_RATIO_RT	SD_PPARAM_RATIO
NPAS_SIO_11/SIO_12/SIO_21/SIO_22/SI_1/SI_2/SO_1/SO_2	SD_PPARAM_SIO
NPAS_SW13/31	SD_PPARAM_SW

NPAS_SW19/91	SD_PPARAM_SW
NPAS_TM	SD_PPARAM_TM
NPAS_TP_CFL	SD_PPARAM_TPCFL
NPAS_T_CFL	SD_PPARAM_TCFL
NPAS_VELLIM	SD_PPARAM_VELLIM
NPAS_VELLIM_PB	SD_PPARAM_VELLIM

Данные параметров, содержащиеся в этих структурах данных описаны в библиотеке “SD\_NPASPOU\_PF”, в папке “Data Types”, [как описано ниже](#).

### Структуры данных ввода/вывода:

Кроме функциональных блоков NPAS POU, процесс ввода/вывода также имеет свои структуры данных.

Структуры данных следующие:

СТРУКТУРА ДАННЫХ	ОПИСАНИЕ
DTag_I_Anlg	Стандартный аналоговый вход с целой величиной на выходе (integer)
DTag_I_Pcnt	Аналоговый вход с величиной на выходе в процентах
DTag_I_PulsL	Импульсный вход (используется как аналоговый вход модуля с импульсными входами)
DTag_I_PushB	Дискретный вход (используется как счетный вход модуля с релейными входами (Pushbutton Input))
DTag_I_Sts	Стандартный дискретный вход
DTag_I_Temp	Температурный вход (с преобразованием сигналов)
DTag_O_Anlg	Стандартный аналоговый выход
DTag_O_Sts	Стандартный дискретный выход

Более подробно структуры данных приведены в:

- SD\_FIELD\_PF → DEVICE\_TAG\_TYPE

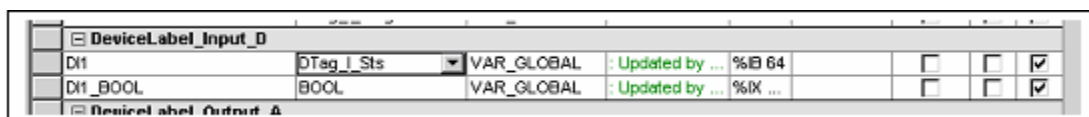
Структуры данных четырёх стандартных процессов ввода/вывода описаны далее:



СТРУКТУРА ДАННЫХ	ЭЛЕМЕНТ	ТИП	ОПИСАНИЕ
DTag_I_Anlg DTag_O_Anlg	Attr	WORD	Атрибут данных
	Value	UINT	Двоичные данные в диапазоне (0x1000 - 0x5000)
	Status	WORD	Статус данных (Good, Bad, etc)
	SH	REAL	Верхний предел шкалы
	SL	REAL	Нижний предел шкалы
	Unit	IndUnit	Инженерные единицы
DTag_I_Sts DTag_O_Sts	Attr	WORD	Атрибут данных
	Value	WORD	Данные состояния (On/Off)
	Status	WORD	Статус данных (Good, Bad, etc)

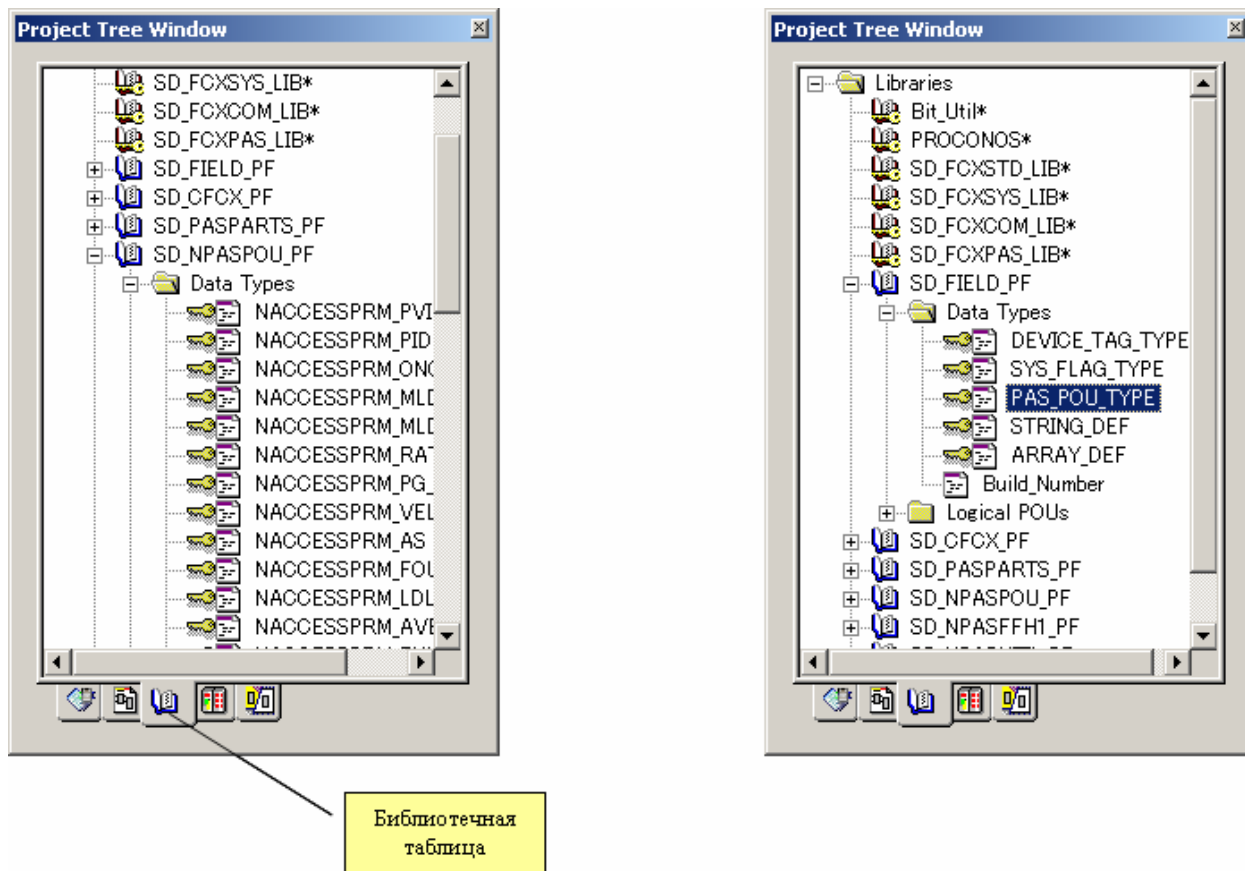
Заметим, что типом данных состояния дискретного ввода/вывода является WORD, и, следовательно, должен быть конвертирован в BOOL перед тем как использоваться в логических функциях. Тем не менее, всякий раз, когда дискретный вход/выход создаётся, создаётся и соответствующая ему логическая переменная.

Например, если дискретный вход создан под именем **DI1**, то автоматически создаётся другая логическая переменная под именем **DI1\_BOOL**. Она может быть использована при логическом управлении.



DeviceLabel_Input_0			
DI1	DTag_I_Sts	VAR_GLOBAL	: Updated by ... %B 64
DI1_BOOL	BOOL	VAR_GLOBAL	: Updated by ... %X ...
DeviceLabel_Output_0			

**Как определить тип данных переменной:**



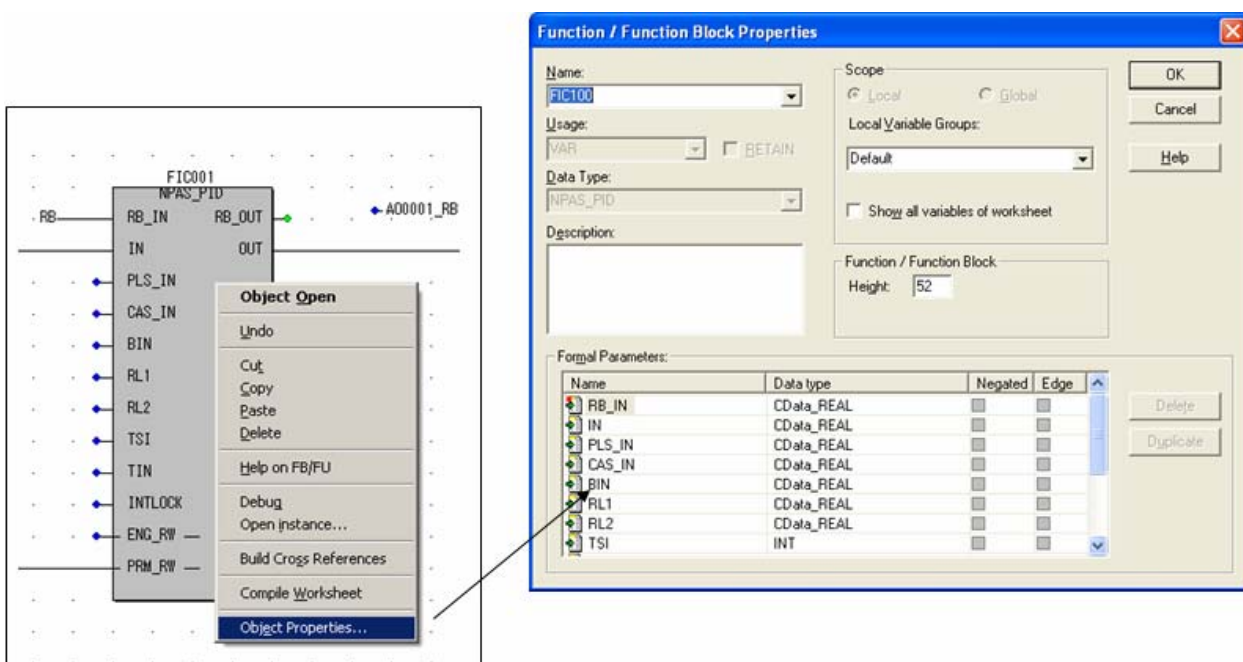
Для того чтобы увидеть структуру данных любого блока NPASPOU (функциональный блок Yokogawa) необходимо:

1. Открыть папку “**Libraries**”;
2. Открыть папку “**SD\_NPASPOU\_PF**”;
3. Открыть папку “**Data Types**”;
4. Двойным щелчком открыть необходимый функциональный блок.

Другие связанные с функциональными блоками Yokogawa структуры данных могут наблюдаться в папке “SD\_FIELD\_PF”. Файл “PAS\_POU\_TYPE” содержит важные структуры данных, такие как “CData\_Real”.

**Как определить типы данных входов/выходов функциональных блоков:**

1. Щёлкнуть правой кнопкой по функциональному блоку (или функции) и выбрать свойства объекта (“**Object Properties**”);
2. Щёлкнуть по папке “**FB/FU**”. Типы данных всех входов/выходов выводятся на дисплей.



### 2.4.3 Типы переменных.

Существуют данные в формах глобальных и локальных переменных:

- **Локальные переменные (Local Variables)** – внутренние данные, которые используются внутри POU (функционального блока или программы) и не требуется доступ к ним снаружи. Они могут быть в форме переменной или переменной ввода/вывода.
- **Глобальные переменные (Global Variables)** – данные доступные всем POU в FCX. Они декларируются как внешние переменные (**External Variable**) внутри POU, и как глобальные переменные (**Global Variable**) в папке ресурсов (“FCX...”).

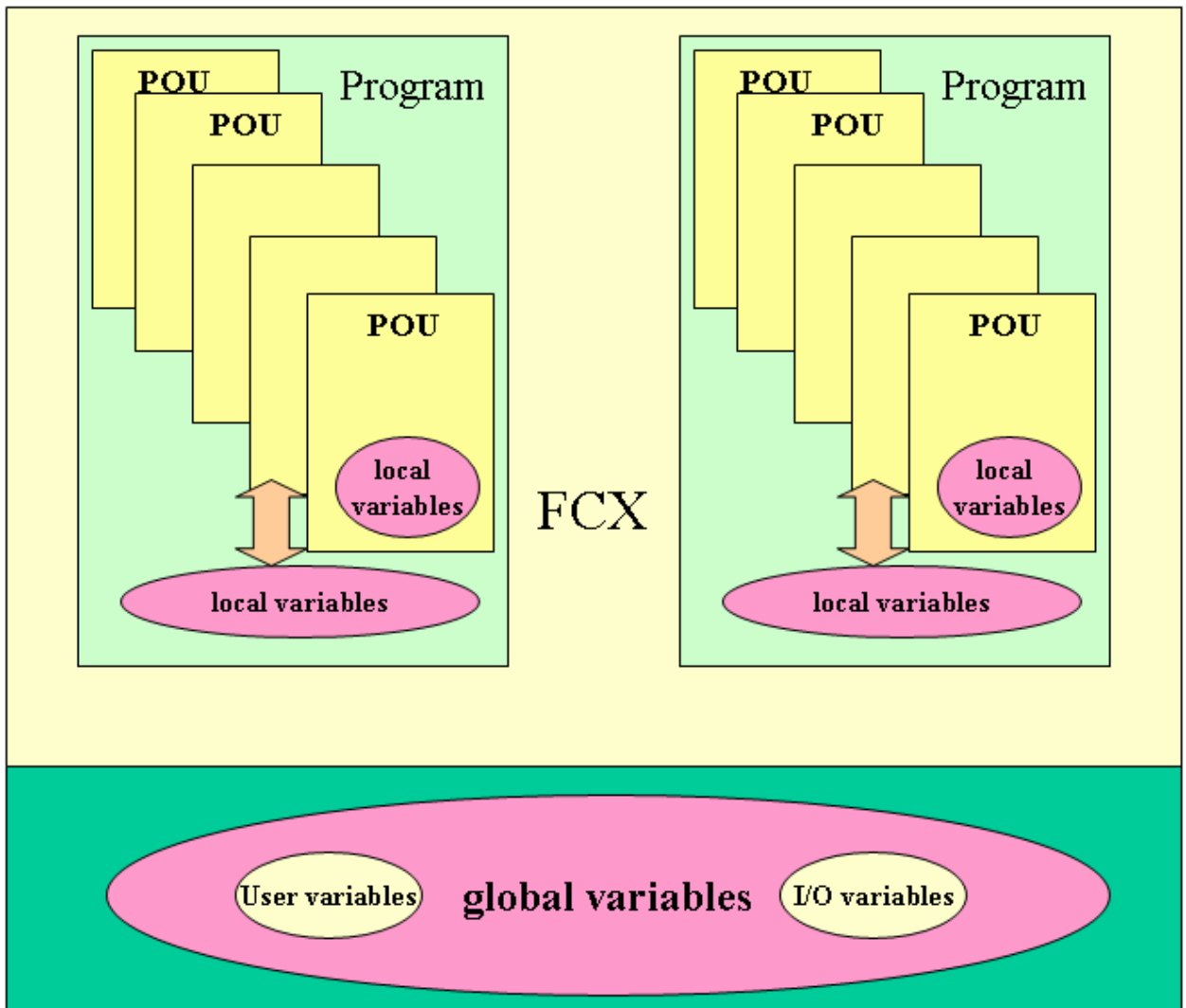
Переменные внутри функционального блока декларируются следующим образом:

КЛЮЧЕВОЕ СЛОВО	ОПИСАНИЕ	ИСПОЛЬЗОВАНИЕ
VAR	Локальная переменная (Local Variable)	Внутренние данные
VAR_EXTERNAL	Глобальная переменная (Global Variable)	Данные, используемые другими блоками
VAR_IN_OUT	Переменные ввода/вывода (Input/Output Variable)	Данные, которые являются входными и выходными функционального блока.
VAR_INPUT	Входная переменная (Input Variable)	Данные, которые являются входными в функциональный блок.
VAR_OUTPUT	Выходная переменная (Output Variable)	Данные, которые являются выходными из функционального блока.

Если переменная используется только внутри POU, она называется **Local Variable**. В этом случае применимы ключевые слова “VAR”, “VAR\_INPUT” and “VAR\_OUTPUT”.

Если переменная используется внутри всего проекта, она называется **Global Variable**. Она должна быть декларирована как “VAR\_GLOBAL” в глобальной декларации и как “VAR\_EXTERNAL” в каждой POU, который её использует.

Связь между локальными и глобальными переменными и конфигурацией FCX следующая:



## 2.4.4 Декларирование переменных.

Для того чтобы система могла использовать данные они должны быть задекларированы. При конфигурировании Logic Designer основную массу данных декларирует автоматически. Более подробно это описано в “IEC61131-3. Programming course”.

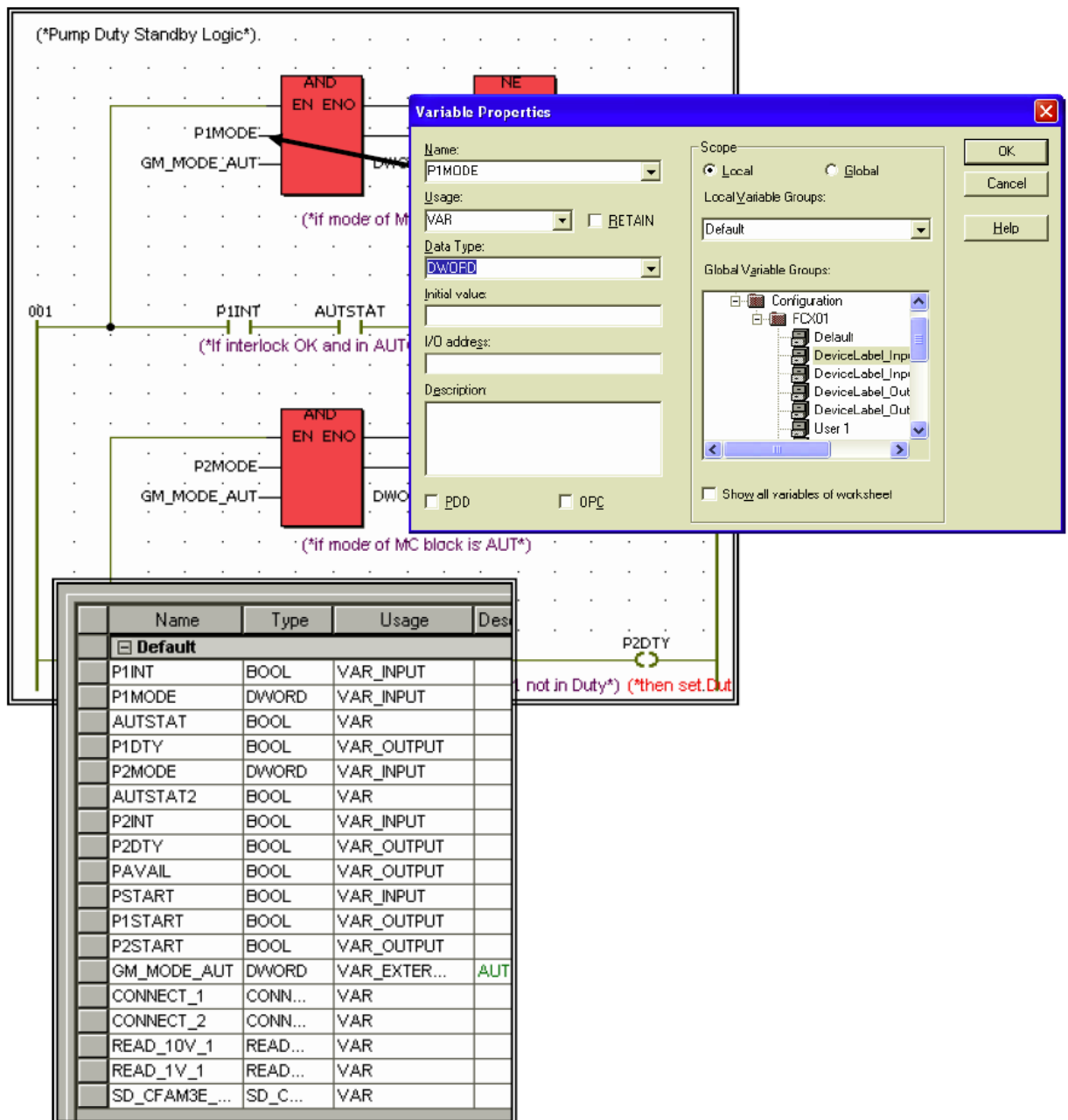
Кроме стандартных позиций декларации могут быть установлены и другие:

Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC
☐ Default								
LIC100	PAS_PID	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FIC100	PAS_PID	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DUMMY1	STRING_PRM	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
LEVELIN	CData_REAL	VAR_INPUT				<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
FLOWIN	CData_REAL	VAR_INPUT				<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
FWV	CData_REAL	VAR_OUTPUT				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

- **Retain** – этот маркер указывает должна ли эта переменная поддерживаться в SRAM (см. раздел [2.2.5.2](#));
- **OPC** – этот маркер указывает должна ли эта переменная быть доступной VDS для web графики. Если переменная не помечена маркером, она не будет обнаружена при импорте через **Object Builder** (см. раздел 3.2.2);
- **PDD** – этот маркер должен быть установлен, если переменная участвует в коммуникации с другим FCX (см. раздел [2.6.8](#)).

### Как регистрировать данные:

1. Дважды щёлкните по точке соединения функционального блока в функционально блочной схеме или логического элемента в релейной схеме;
2. Введите имя переменной и выберите опцию “**Local**” или “**Global**”. Более подробно о выборе “**Global Variable**” смотри далее;
3. Выберите тип переменной в области “Usage” (например “VAR”) и тип данных в области “Data Type” (например “DWORD”). Если данные должны читаться другой системой, такой как VDS, то поставьте маркер OPC.



(\*Pump Duty Standby Logic\*).

Variable Properties dialog for P1MODE:

- Name: P1MODE
- Usage: VAR
- Data Type: DWORD
- Initial value:
- I/O address:
- Description:
- Options:  PDD,  OPC

Global Variable Groups:

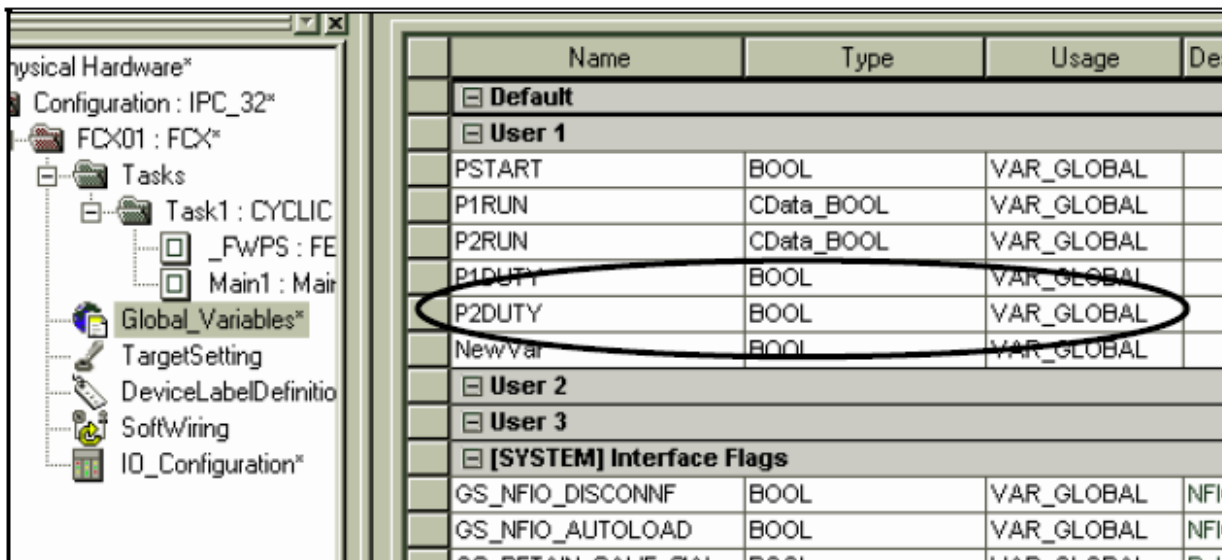
- Configuration
  - FCX01
    - Default
    - DeviceLabel\_Inp
    - DeviceLabel\_Inp
    - DeviceLabel\_Out
    - DeviceLabel\_Out
    - User 1

Variable Declaration Table:

Name	Type	Usage	Des
<b>Default</b>			
P1INT	BOOL	VAR_INPUT	
P1MODE	DWORD	VAR_INPUT	
AUTSTAT	BOOL	VAR	
P1DTY	BOOL	VAR_OUTPUT	
P2MODE	DWORD	VAR_INPUT	
AUTSTAT2	BOOL	VAR	
P2INT	BOOL	VAR_INPUT	
P2DTY	BOOL	VAR_OUTPUT	
PAVAIL	BOOL	VAR_OUTPUT	
PSTART	BOOL	VAR_INPUT	
P1START	BOOL	VAR_OUTPUT	
P2START	BOOL	VAR_OUTPUT	
GM_MODE_AUT	DWORD	VAR_EXTER...	AUT
CONNECT_1	CONN...	VAR	
CONNECT_2	CONN...	VAR	
READ_10V_1	READ...	VAR	
READ_1V_1	READ...	VAR	
SD_CFAM3E_...	SD_C...	VAR	

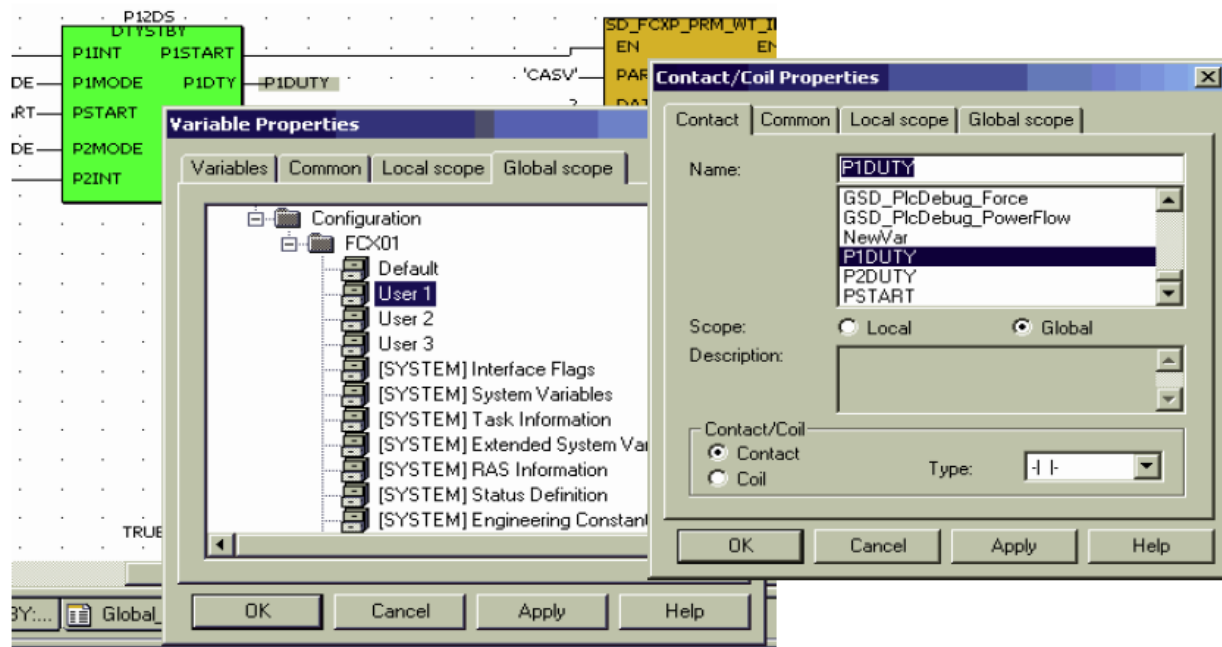
**Как регистрировать данные глобальной переменной:**

1. Глобальная переменная должна быть задекларирована в файле “Global Variable Definition” в первую очередь как показано ниже:



Name	Type	Usage	Description
<b>Default</b>			
<b>User 1</b>			
PSTART	BOOL	VAR_GLOBAL	
P1RUN	CData_BOOL	VAR_GLOBAL	
P2RUN	CData_BOOL	VAR_GLOBAL	
P1DUTY	BOOL	VAR_GLOBAL	
P2DUTY	BOOL	VAR_GLOBAL	
NewVar	BOOL	VAR_GLOBAL	
<b>User 2</b>			
<b>User 3</b>			
<b>[SYSTEM] Interface Flags</b>			
GS_NFIO_DISCONN	BOOL	VAR_GLOBAL	NFIO
GS_NFIO_AUTOLOAD	BOOL	VAR_GLOBAL	NFIO
GS_RETAIN_GATE_VAL	BOOL	VAR_GLOBAL	

2. После этого глобальная переменная может быть использована в ROU;
3. Дважды щёлкните по точке соединения функционального блока в функционально блочной схеме или логического элемента в релейной схеме.
4. Откройте закладку “Global Scope”, выберите ресурс (FCX) и категорию глобальной переменной, в которую переменная должна быть размещена (например, User1);
5. Откройте закладку “Contact”, введите имя переменной в графу “Name:” и выберите “Global”;
6. OPC устанавливается в “Global Variables” странице в папке FCX.





## 2.4.5 Глобальные константы.

Различные константы, которые используются в блоках NPAS POU и других описываются в файле глобальных переменных “**Global Variables**”. Они могут использоваться по мере необходимости. Например, записать “**HH**” Аларм в VDS (см. раздел [2.6.7](#)) или установить режим контроллера (см. раздел [2.6.4](#)).

Константы подразделяются на 3 категории:

- **Status Definition** – Определения, которые используются часто, такие как режим и статус аларма;
- **Engineering Constant** – Константы, используемые для установки инженерных параметров.
- **Application Flags** – Внутренние константы и соединения в блоках NPAS POU. Обычно пользователями не используются.

Константы с именами, начинающимися с “**GM\_**”(зарезервировано системой):

ИМЯ КОНСТАНТЫ	ВЕЛИЧИНА
GM_MODE_MAN	MAN
GM_MODE_AUT	AUT
GM_MODE_CAS	CAS
GM_ALRM_HH	HH
GM_ALRM_HI	HI
GM_ALRM_LO	LO
GM_ALRM_LL	LL

## 2.5 Программирование в IEC61131-3.

### 2.5.1 Обзор.

IEC61131-3 специфицирует пять языков программирования используемых в системе:

- Язык описания Релейных Схем (**Ladder Diagrams “LD”**);
- Язык описания Функционально Блочных Схем (**Functional Blocks “FBD”**);
- Язык описания Последовательностных Функциональных Карт (**Sequence Functions Charts “SFC”**);
- Язык описания Структурированного Текста (**Structured Text “ST”**);
- Язык описания Листа Инструкций (**Instruction List “IL”**).

Изучение языков вне пределов настоящего курса, в то же время, функции системы, реализующие их, несомненно, увеличивают её ценность.

Yokogawa предлагает ряд функциональных блоков, которые могут быть использованы в FBD, ST и LD. Эти блоки обычно применяются в DCS CS3000 и поддерживают функции комплексного управления без использования базовых функциональных блоков IEC61131.

Предлагаются следующие библиотеки функций и функциональных блоков:

БИБЛИОТЕКИ	ОПИСАНИЕ
Functions	Функции IEC61131
Function Blocks	Функциональные блоки IEC61131
String FUs	Функции обработки символьных строк
Type Conv. FUs	Преобразование типов данных
Bit UTIL	Побитовая обработка
PROCONOS	Утилиты для доступа к памяти O/S
SD_CFCX_PF	Межконтроллерные коммуникации
SD_FCXCOM_LIB	Межконтроллерные коммуникации
SD_FCXNPAS_LIB	Чтение/Запись данных из/в одного функционального Блока в другой
SD_FCXSTD_LIB	Утилиты Yokogawa
SD_FCXSYS_LIB	Утилиты Yokogawa
SD_FIELD_PF	Утилиты Yokogawa
SD_NPASPARTS_PF	Утилиты для PASPOU функциональных блоков
SD_NPASPOU_PF	Функциональные Блоки Yokogawa
SD_NPASUTIL_PF	Функции преобразования Данных
SD_WIRE_PF	Утилиты WDA
SD_CFAM3E_PF	Коммуникационные Утилиты для FA-M3 PLC
SD_CMELSECE_PF	Коммуникационные Утилиты для Melsec PLC

**Примечание:** Все 'SD' библиотеки являются библиотеками функций Yokogawa.

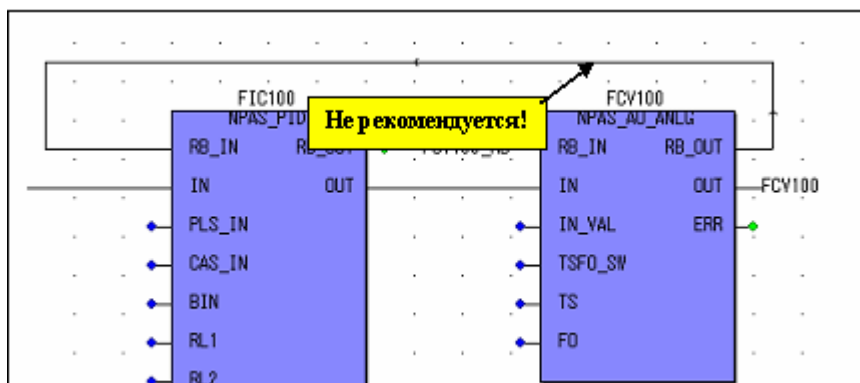
## 2.5.2 Порядок исполнения.

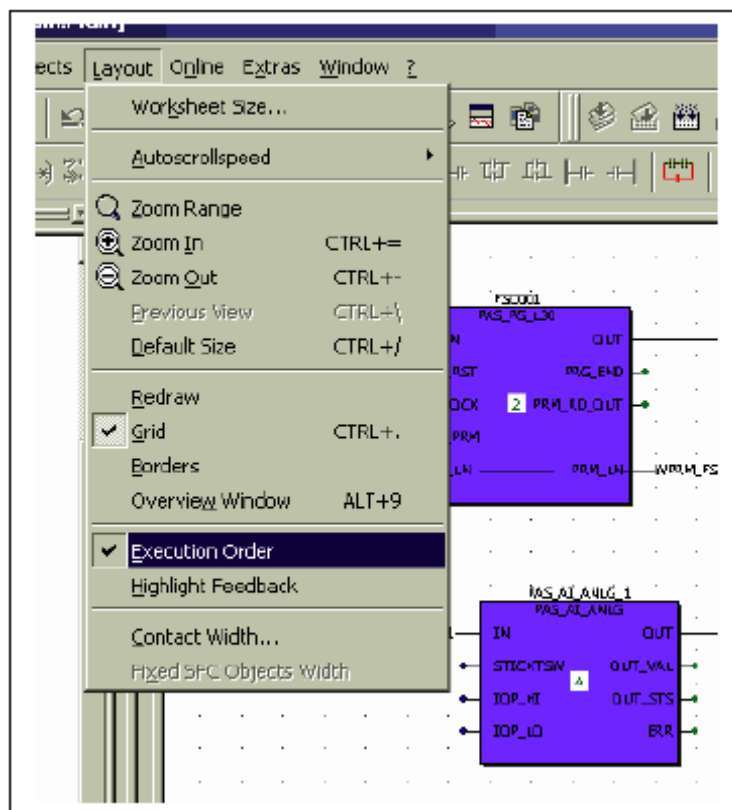
Порядок исполнения для программ: слева направо, сверху вниз. Для Структурированного Текста, Релейных Схем и Листа Инструкций, порядок исполнения естественный. Для SFC порядок исполнения определяется ветвящейся структурой.

Однако для FBD порядок исполнения далеко не очевиден и зависит не только от позиции функции/функционального блока на рабочем листе, но также от того, как они соединены между собой.

Например: если функциональный блок аналогового входа размещён правее блока PID, подключённого к нему, компилятор распознаёт это и выполняет его первым.

В случае если есть в наличии петли обратной связи от второго блока к первому, компилятор не знает какой блок исполнять первым и делает это наугад. В этой ситуации рекомендуется порядок объявлять явно. Для этого используйте локальные переменные (см. раздел [2.6.4](#)).



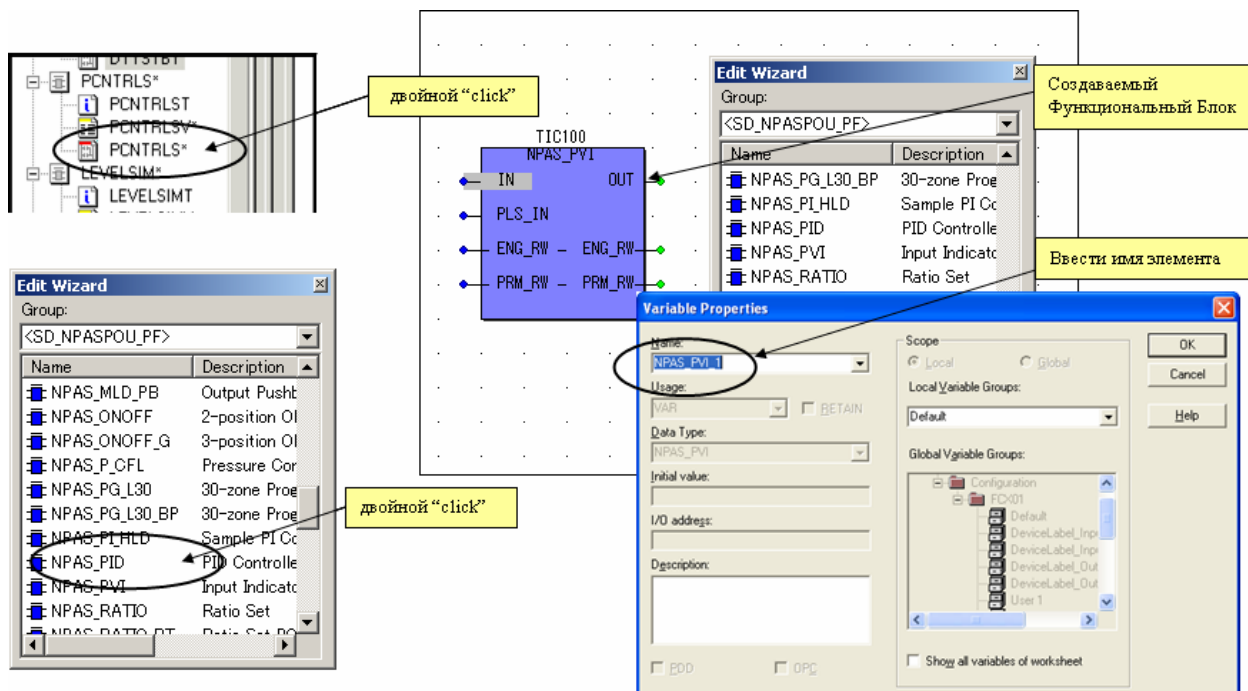


Порядок исполнения, определённый компилятором можно увидеть в опции “Execution Order” выпадающего меню “Layout”.

## 2.5.3 Процедуры создания функционально блочных диаграмм (FBD).

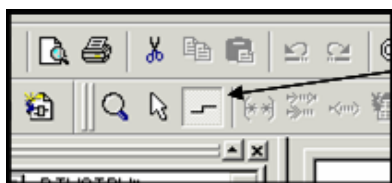
### Создание функционально блочных диаграмм:

1. Создайте логический POU (см. раздел [2.3.4](#)) обозначенный как функциональный блок;
2. Двойным щелчком по файлу программы POU вызовите рабочий лист программы, внутри него размещены блоки;
3. Откройте окно “**Edit Wizard**” в выпадающем меню “**View**”. Окно “**Edit Wizard**” содержит все библиотечные функции, включая блоки NPASPOU;
4. Щёлкните по рабочему листу и “**Edit Wizard**” станет доступным;
5. Двойным щелчком по требуемому функциональному блоку вызовите окно “**Variable Properties**”;
6. Введите в опцию “**Name:**” “**Variable Properties**” имя тага для блока.
7. Нажмите кнопку “**OK**”.



### Соединение блоков графическим способом:

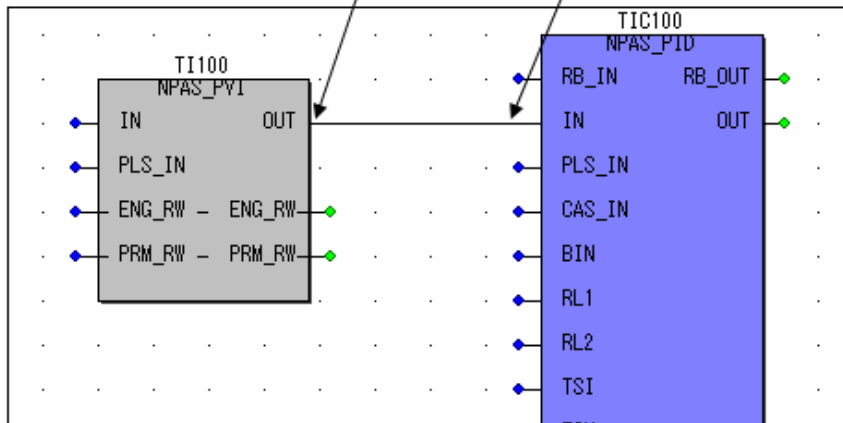
1. Для соединения функциональных блоков между собой выберите необходимый инструмент (“**Wiring Tool**”).
2. Щёлкните по выходной клемме соединяемого блока, а затем двойным щелчком укажите входную клемму другого соединяемого блока, на рабочем листе отображается связь в виде соединительной линии:



Wiring Tool

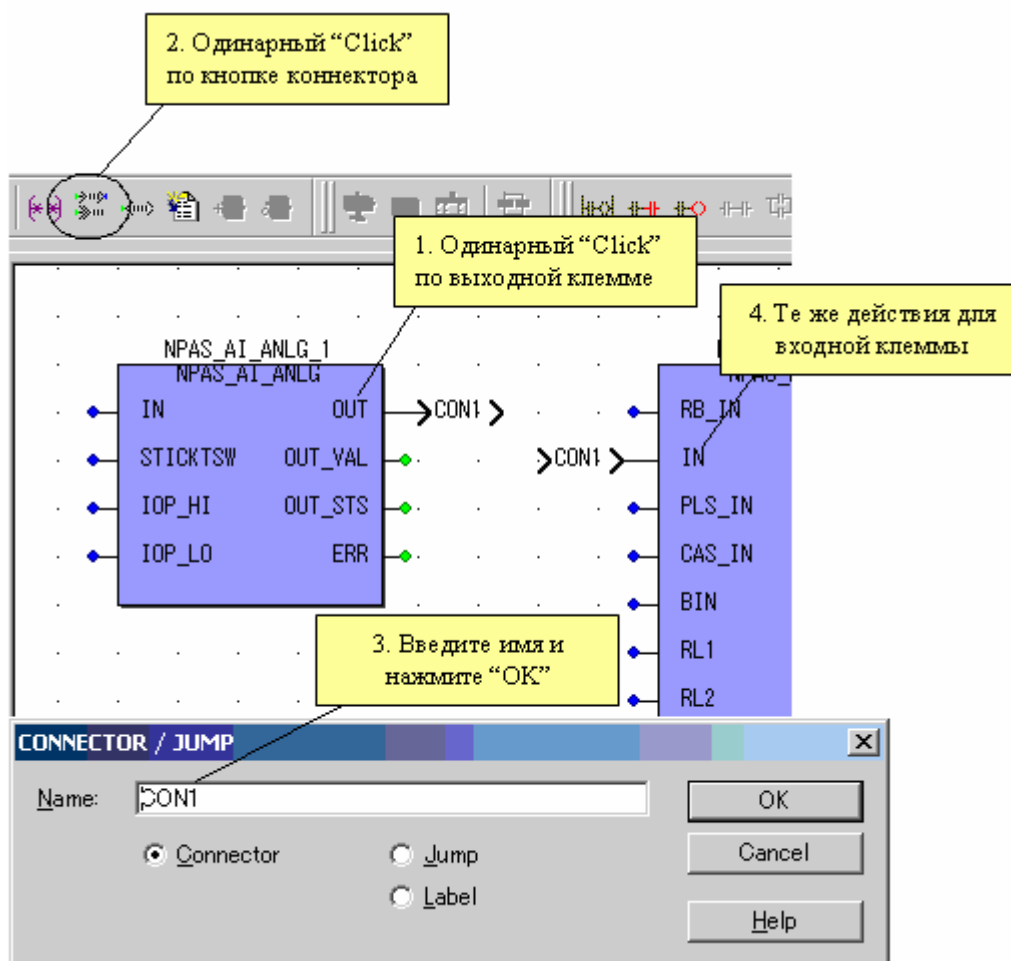
Одinarный "Click"

Двойной "Click"



**Соединение блоков через переменные:**

Когда неудобно соединять блоки между собой стандартным способом, описанным выше, функция соединения может быть реализована при помощи промежуточных переменных, как показано ниже:



Смотрите также раздел [2.6.4](#) для построения соединений между функциями и функциональными блоками.

## 2.5.4 Защита ROU.

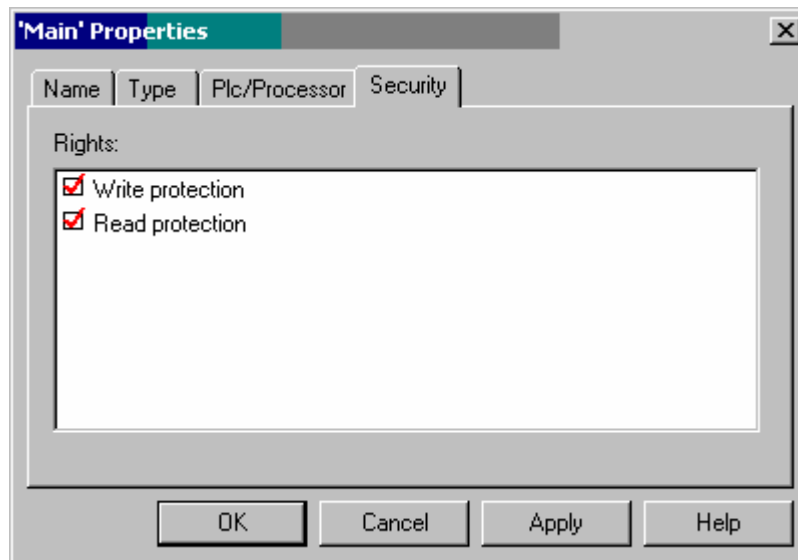
Если рабочий лист (или ROU) обозначен как функция или функциональный блок, он может быть защищён паролем так, что он может быть использован как блок, но не может быть открыт или отредактирован. ROU, обозначенный как программа, может также быть защищён паролем, так, что программа, запускаемая в FCX, целиком закрыта и не может быть доступной без пароля.

### Как установить защиту:

1. Щёлкните правой кнопкой по папке ROU, открывается контекстное меню;
2. Выберите Свойства "**Properties**", открывается окно "**Main' Properties**";
3. Выберите закладку "**Security**";
4. Пометьте маркером "**Write protection**" и/или "**Read protection**" и нажмите кнопку "**OK**";
5. Выберите опцию "**Enter Password**" в выпадающем меню "**File**" и введите пароль;



6. Закройте, а затем откройте заново Logic Designer. Защищённый POU будет иметь собственный пароль и доступ к его рабочему листу и данным будет ограничен.



### *Как модифицировать конфигурацию защиты:*

- Для редактирования защищённого POU выберите опцию **“Enter Password”** в выпадающем меню **“File”** и введите пароль.
- Защита может быть снята нажатием кнопки **“Remove Password”** в меню ввода пароля или путём снятия маркеров **“Write protection”** и **“Read protection”** в окне **“Main' Properties”**.

## 2.6 Функциональные блоки NPASPOU Yokogawa.

### 2.6.1 Краткий обзор блоков NPASPOU.

Библиотека NPASPOU состоит из функциональных блоков разработанных Yokogawa для использования при программировании управляющих приложений. Более подробно блоки описаны в Приложении 1. Все блоки подразделяются на группы:

- Блоки ввода/вывода (**Input Output blocks**);
- Регуляторы (**Controller blocks**);
- Блоки ручной загрузки (**Manual loader blocks**);
- Блоки задания сигнала (**Signal setter blocks**);
- Ограничители сигнала (**Signal limiter blocks**);
- Селекторы сигналов (**Signal selector Blocks**);
- Блоки распределения сигналов (**Signal distributor blocks**);
- Вычислительные блоки (**Calculation Blocks**);
- Блоки последовательного внешнего управления (**Sequence Auxiliary Blocks**).

**Блоки ввода/вывода:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Аналоговые входы и выходы	NPAS_AI_ANLG	Блок аналогового входа – Стандартный
	NPAS_AI_PCNT	Блок аналогового входа – Процентный
	NPAS_AI_PULS_CI	Блок аналогового входа – Импульсный
	NPAS_AI_PULS_QT	Блок аналогового входа – Счётный
	NPAS_AI_TEMP	Блок аналогового входа – Температурный
	NPAS_AO_ANLG	Блок аналогового выхода – Стандартный
	NPAS_PVI	Input Indicator – Индикатор входа
Дискретные входы и выходы	NPAS_DI_PUSHB	Блок дискретного входа – Кнопочный
	NPAS_DI_STS	Блок дискретного входа – Стандартный
	NPAS_DO_STS	Блок дискретного выхода – Стандартный
	NPAS_DO_STS_PW	Блок дискретного выхода – ШИМ выход

**Управляющие блоки:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Регуляторы	NPAS_PID	ПИД регулятор (PID)
	NPAS_PI-HLD	ПИ регулятор со стробированием (PI-HLD)
	NPAS_ONOFF	2-позиционный регулятор (ONOFF)
	NPAS_ONOFF_G	3-позиционный регулятор (ONOFF-G)

**Блоки ручной загрузки:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Блоки ручной загрузки	NPAS_MLD	Блок ручной загрузки (MLD)
	NPAS_MLD_BT	Блок ручной загрузки с индикатором входа (MLD-PVI)
	NPAS_MLD_PB	Блок ручной загрузки с переключателем Auto/Man (MLD-SW)

**Блоки задания сигнала:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Блоки задания сигнала	NPAS_RATIO	Блок установки соотношения
	NPAS_RATIO_RT	Блок установки соотношения RT
	NPAS_PG_L30	30-зонный программатор (PG-L30)
	NPAS_PG_L30_BP	30-зонный программатор (PG-L30- BP)

**Ограничители сигнала:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Ограничители сигнала	NPAS_VELLIM	Ограничитель скорости (VELLIM)
	NPAS_VELLIM_PB	Ограничитель скорости (VELLIM-PB)

**Селекторы сигналов:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Селекторы сигналов	NPAS_AS_H	Автоселектор (AS-H)
	NPAS_AS_M	Автоселектор (AS-M)
	NPAS_AS_L	Автоселектор (AS-L)

**Блоки распределения сигналов:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Блоки распределения сигналов	NPAS_FOUT	Распределитель сигналов при каскадном управлении (FOUT)
	NPAS_FFSUM	Блок компенсации сигнала при упреждающем управлении (FFSUM)
	NPAS_FFSUM_BL	Блок компенсации сигнала при упреждающем управлении (FFSUM-BL)

**Вычислительные блоки:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Вычислительные блоки	NPAS_LDLAG	Апериодическое звено (LDLAG)
	NPAS_DLAY	Блок времени задержки (DLAY)
	NPAS_AVE_M	Блок скользящего среднего (AVE_M)
	NPAS_AVE_C	Блок накопленного среднего (AVE_C)
	FUNC-VAR	Блок кусочно-линейной аппроксимации переменной (FUNC-VAR)
	NPAS_P_CFL	Блок коррекции по давлению (P-CFL)
	NPAS_T_CFL	Блок коррекции по температуре (T-CFL)
	NPAS_TP_CFL	Блок коррекции по температуре и давлению (TPCFL)
	NPAS_ASTM1	Блок коррекции по ASTM (Старый JIS) (ASTM1)
	NPAS_ASTM2	Блок коррекции по ASTM (Новый JIS) (ASTM2)
	NPAS_SW-13	Блок 3-полюсного 1-позиционного селекторного переключателя (SW-13)
	NPAS_SW-31	Блок 1-полюсного 3-позиционного селекторного переключателя (SW-31)
	NPAS_SW-19	Блок 9-полюсного 1-позиционного селекторного переключателя (SW-19)
	NPAS_SW-91	Блок 1-полюсного 9-позиционного селекторного переключателя (SW-91)
	NPAS_BD_BUF_R	Блок установки группы данных (BD_BUF_R)
NPAS_BD_BUF_T	Блок установки группы данных (BD_BUF_T)	

**Блоки последовательного внешнего управления:**

ТИПЫ БЛОКОВ	ОБОЗНАЧЕНИЕ	НАИМЕНОВАНИЕ
Таймеры	NPAS_TM	Блок таймера (TM)
	NPAS_CT	Блок счётчика серии импульсов (CT)
Блоки выключателей и управления двигателями	SI-1	Блок выключателей на 1 вход (SI-1)
	SI-2	Блок выключателей на 2 входа (SI-2)
	SO-1	Блок выключателей на 1 выход (SO-1)
	SO-2	Блок выключателей на 2 выхода (SO-2)
	SIO-11	Блок выключателей на 1 вход и 1 выход (SIO-11)
	SIO-21	Блок выключателей на 2 вход и 1 выход (SIO-21)
	SIO-12	Блок выключателей на 1 вход и 2 выхода (SIO-12)
	SIO-22	Блок выключателей на 2 вход и 2 выхода (SIO-22)

## 2.6.2 Использование блоков NPASPOU.

Более подробно функциональные блоки описаны в Приложении 1. Этот раздел описывает наиболее важные вопросы применения блоков.

Большинство блоков NPASPOU имеют следующие клеммы:

НАИМЕНОВАНИЕ КЛЕММЫ	НАЗНАЧЕНИЕ
RB_IN	Эти клеммы, принадлежащие разным функциональным блокам соединённым каскадно, соединяются между собой для реализации процедуры обратного слежения (pushback). В большинстве блоков RB_OUT содержит CASV (см. раздел <a href="#">2.6.6</a> ).
RB_OUT	
ENG_RW	Используются для установки инженерных параметров, таких как комментарии, включение реверсивного действия PID алгоритма, инженерные единицы параметров и диапазон их изменения. Более подробно см. раздел <a href="#">2.6.4</a> .
PRM_RW	Используется для чтения и записи внутренних динамических данных (параметров настройки) (см. раздел <a href="#">2.6.4</a> ).
IN/OUT	Эти клеммы коммутируют входные и выходные потоки данных между блоками (см. раздел <a href="#">2.6.3</a> ).
CAS_IN	В каскадных контурах клемма OUT ведущего функционального блока подключается к клемме CAS_IN подчинённого (см. раздел <a href="#">2.6.6</a> ).

Более подробно назначение клемм описано в последующих разделах.

Кроме того, существует множество библиотек функций, которые содержат в себе функции для блоков NPASPOU. Некоторые из них также описаны в последующих разделах и перечислены ниже:

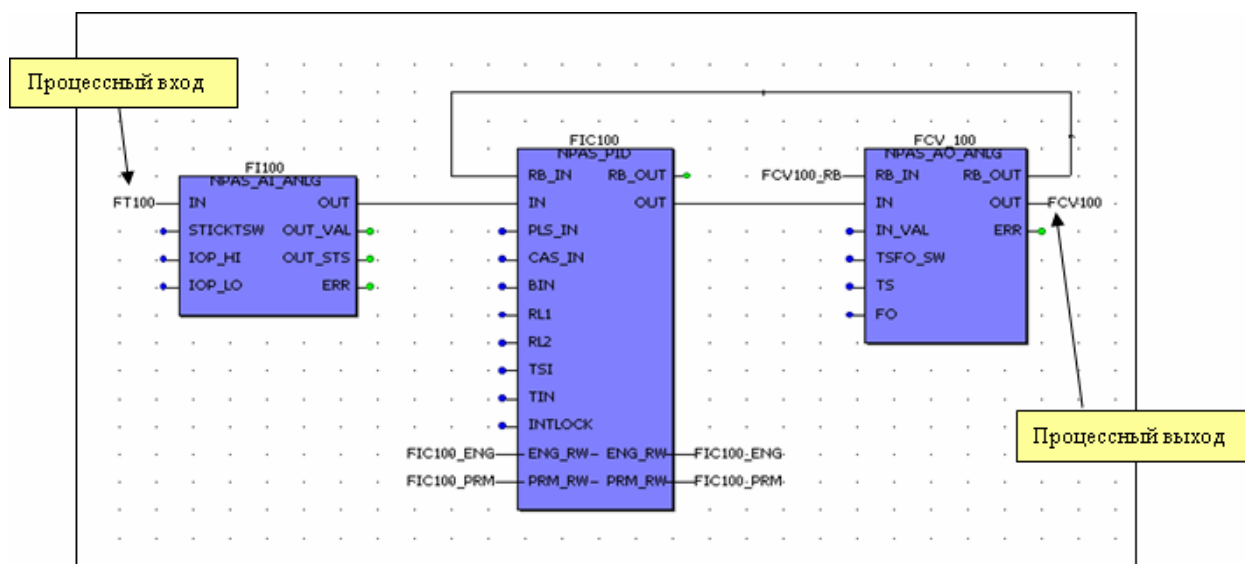
БИБЛИОТЕКА ФУНКЦИЙ	НАЗНАЧЕНИЕ	СМ. РАЗДЕЛ
SD_NPASUTIL_PF	Преобразование типов данных для блоков NPASPOU.	<a href="#">2.6.5</a>
SD_FIELD_PF	Генерация аларменных сообщений с использованием функции PAS_MSG_UPRCALM.	<a href="#">2.6.7</a>
SD_FCXCOM_LIB	Межконтроллерные коммуникации	<a href="#">2.6.8</a>

### 2.6.3 Процесс коммутации ввода/вывода (Connecting Process I/O).

Процесс коммутации ввода/вывода производится через функциональные блоки ввода/вывода, которые включены в библиотеку NPAS\_POU. Типы блоков перечислены в разделе [2.6.1](#). Более детальное описание смотри в Приложении 1.

- Блоки аналоговых входов (“NPAS\_AI\_”\*) преобразуют входные сигналы в величины представленные в инженерных единицах и в данные типа ‘CData’ соответственно;
- Блоки аналоговых выходов (NPAS\_AO\_ANLG) преобразуют ‘CData\_Real’, заданные в диапазоне 0-100%, в выходную величину типа REAL воспринимаемую модулем аналогового выхода.

Пример того, как модули ввода/вывода используют это преобразование, приведен ниже:



Аналоговые вход (FT100) и выход (FCV100) определены в файле “**DeviceLabelDefinition**” (конфигурирование определено описано в разделе [2.2.1](#)).

В процессе отслеживания состояния аналогового выхода (когда величина, используемая внешним устройством для выполнения своих функций, изменяется не под воздействием аналогового выхода, а под воздействием других факторов, например, вручную) блок аналогового выхода получает информацию о состоянии устройства (процесс обратного чтения (**Read Back**)) через клемму RB\_IN. Формат этой величины:

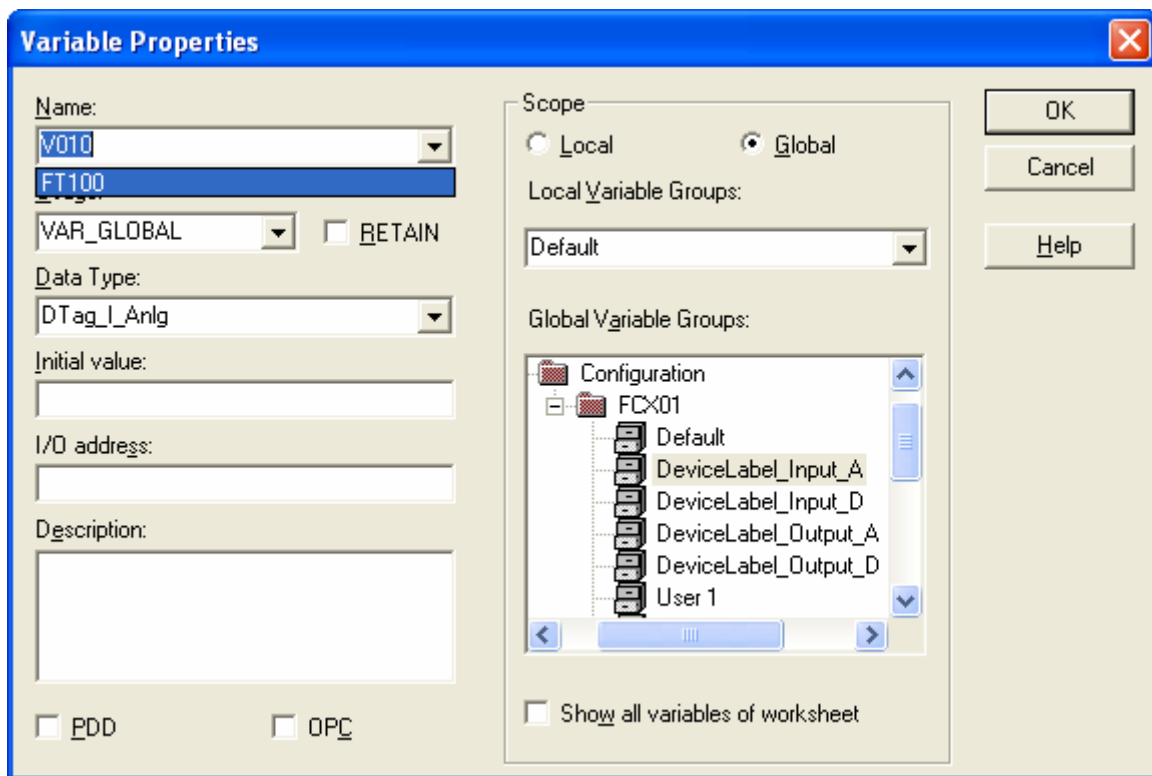
TAG\_RB

Эта переменная поддерживает процесс согласования функции аналогового выхода в модуле аналогового выхода. Более детально процесс описан в разделе [2.5.3](#).

#### **Как присвоить имя метки внешнего устройства (Device Label Name):**

Для присвоения имени метки внешнего устройства (аналоговый вход или выход) клемме функционального блока выполните следующую процедуру:





1. Двойным щелчком по клемме IN/OUT блока аналогового входа/выхода вызовите контекстное меню;
2. Выберите **“DeviceLabel\_Input\_A”** (для аналогового входа) или **“DeviceLabel\_Output\_A”** (для аналогового выхода), при этом вызывается окно **“Variable Properties”**;
3. На панели **“Scope”** выберите **“Global”**;
4. Выберите требуемое имя аналогового устройства из предлагаемых в **“Global Variable Groups”** панели **“Scope”** и нажмите кнопку **“OK”**.

## 2.6.4 Коммутация внутренних параметров (Internal Parameters).

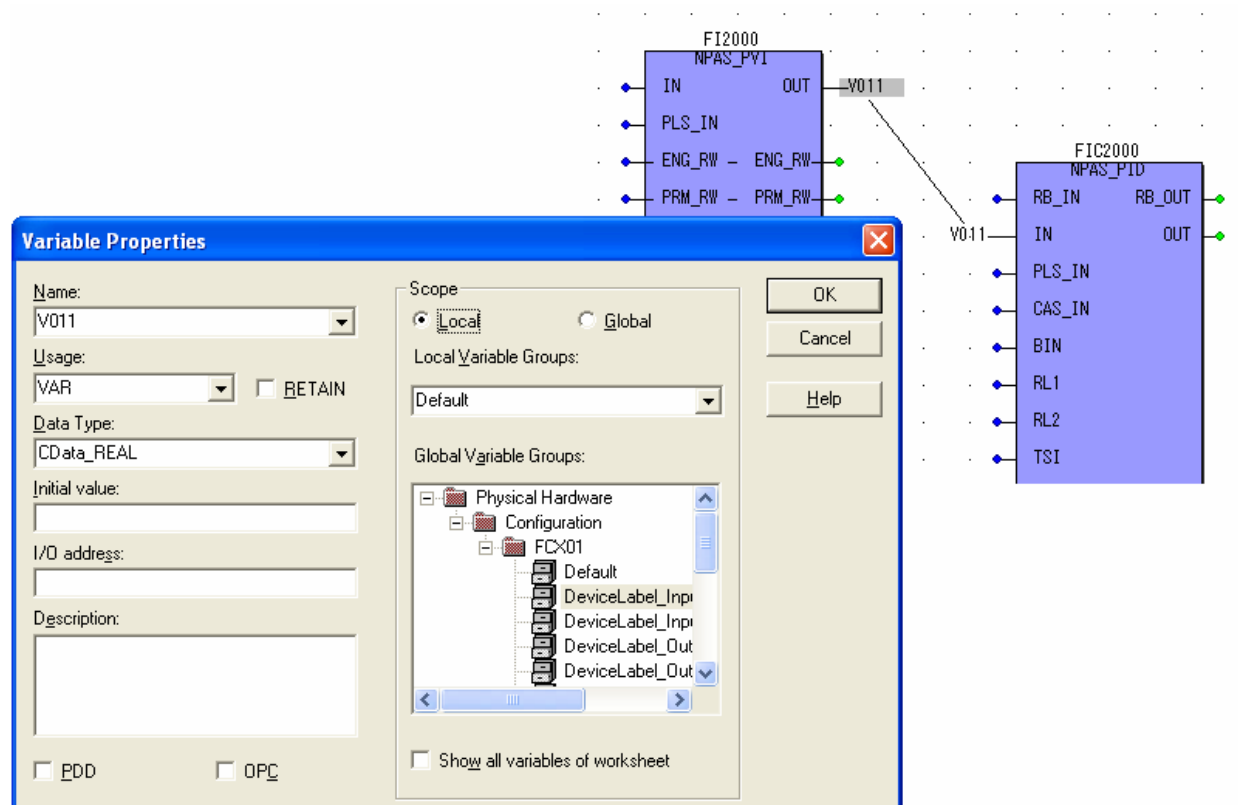
### Коммутации входов/выходов:

Обычные способы коммутации клемм IN и OUT любого функционального блока показаны на примере в разделе [2.5.3](#).

### Коммутация графическим способом:

Обычно для коммутации нормальных сигнальных потоков используется графический способ (см. раздел [2.5.2](#)), если эта связь осуществляется в пределах одного рабочего листа. Если это не так, графический способ не применим, и используется способ коммутации через переменную. Тип используемой переменной зависит от природы устанавливаемой коммутации:

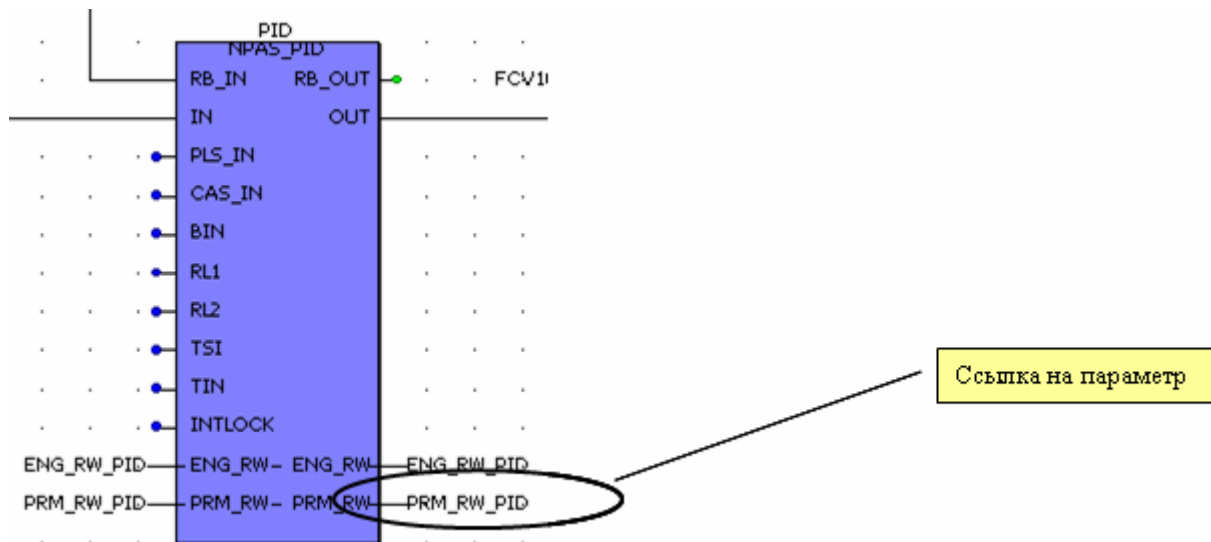
- Если коммутация осуществляется между разными ROU, переменная должна быть глобальной;
- Если коммутация осуществляется внутри одного ROU, переменная должна быть локальной.



**Примечание:** Если локальная переменная определена в пределах ROU назначенной как Program, то она может быть использована для коммутации между разными рабочими листами только внутри этой программы.

### Коммутации доступа к параметрам:

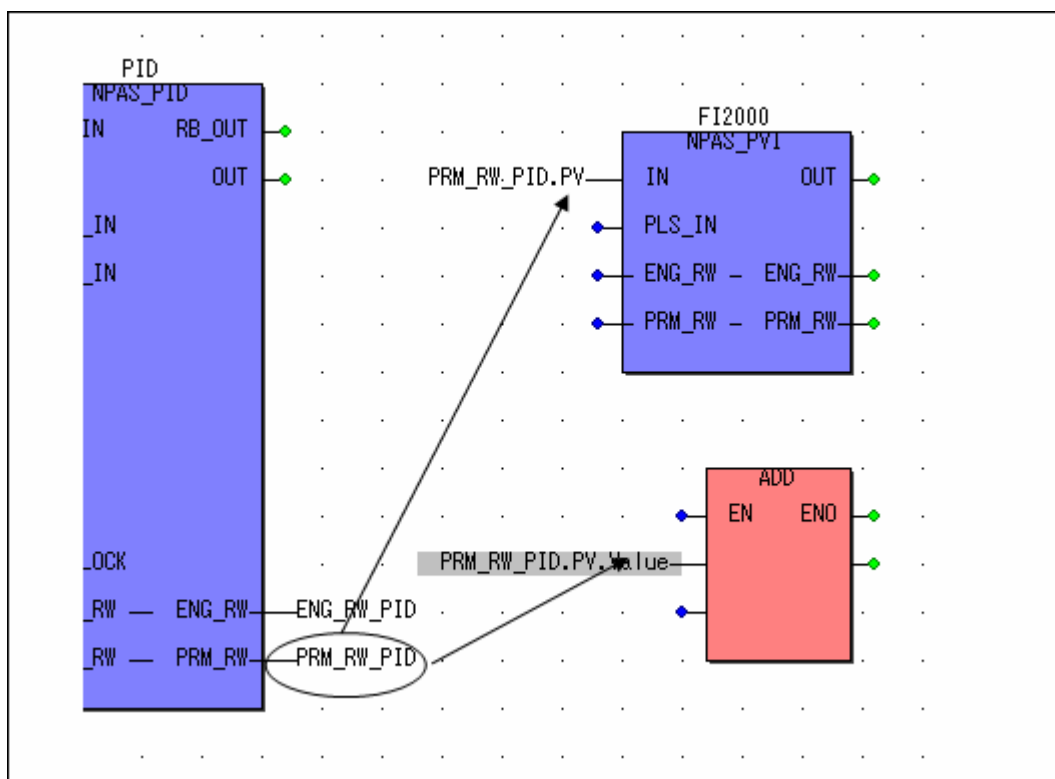
Часто необходимо что бы один функциональный блок мог писать или читать внутренние параметры другого функционального блока:



**PRM\_RW** – эта клемма применяется для чтения и записи в другом функциональном блоке. Переменная может иметь несколько имён, но это структурированная переменная (например, **SD\_NPPRM\_PID**) с типом, зависящим от того, как этот функциональный блок используется. Типы данных всех блоков NPASPOU приведены в разделе [2.4.2](#).

**Примечание:** Даже если это не используется, к каждой **PRM\_RW** клемме должна быть подключена переменная.

Так как это структурированная переменная, она обычно содержит в себе несколько более простых переменных, то важно корректное задание имени переменной в ссылке на неё, указанной в ссылающейся клемме читающего функционального блока.



Например, если имя переменной в блоке источнике PRM\_RW\_PID, то для того чтобы прочитать PV из этого блока коммутационная ссылка должна быть **PRM\_RW\_PID.PV**.

Помните, что PV является в свою очередь также структурой данных, поэтому, если читающий блок принимает величину типа REAL, то коммутационная ссылка должна быть **PRM\_RW\_PID.PV.Value**.

### Установка верхнего предела аварийной сигнализации (High Alarm limit) (PH) регулятора:

Установка величин в переменные проста. Например, для того чтобы установить верхний предел аварийной сигнализации (High Alarm limit) (PH) равным 20.0, задекларируйте следующую запись на рабочем листе:

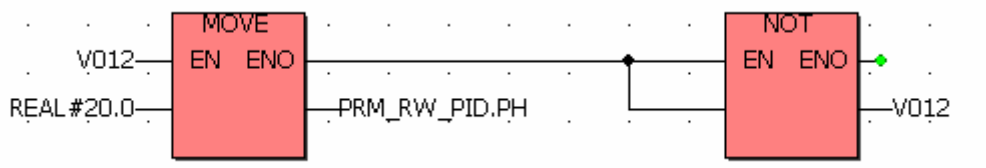
```

REAL #20.0—PRM_RW_PID.PH

```

Однако следует помнить, что эта декларация выполняется на каждом цикле. В большинстве приложений установка данных должна выполняться по условию или по флагу.

Для того чтобы установить данные по флагу:



Где: EN -Разрешающий ключ (логическая переменная – TRUE/FALSE)  
 ENO -Ретраслятор разрешающего ключа

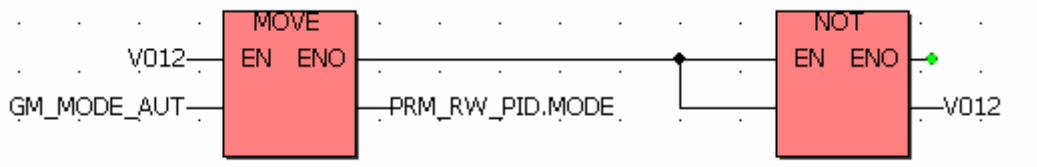
Функция MOVE устанавливает данные только тогда, когда V012 (логический флаг) становится истинным (TRUE). V012 возвратится в состояние ложный (FALSE), когда функция NOT станет истинной (TRUE).

### Установка режима (MODE) регулятора:

Способ такой же, как и в вышеприведённом примере. Однако в этом случае данными является слово двойной длины (DWORD).

Например, для того чтобы переключить регулятор в ручной режим (MANUAL), величина константы должна быть: 16#00800000.

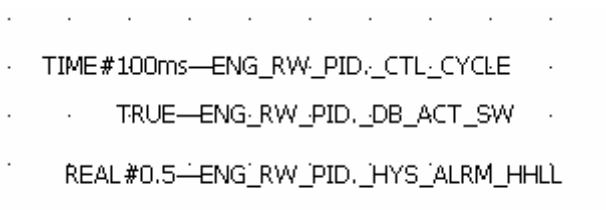
Для упрощения лучше использовать константы Status Definition содержащиеся в файле Global Variables (см. раздел [2.4.5](#)). В этом случае могут быть использованы естественные общепринятые имена.



В этом примере данные коммутации связаны с глобальной переменной “GM\_MODE\_AUT”, которая содержит величину “AUT” в слове двойной длины.

### Установка инженерных параметров в блоках NPAS POU:

Установка инженерных параметров выполняется аналогично. Инженерные параметры определяют начальный статус NPAS POU.



### 2.6.5 Конвертирование типов данных.

Каждой переменной в системе присвоен тип данных или их структура. Когда эти переменные коммутируются между блоками, они должны быть одинакового типа. Существуют несколько блоков преобразования типов данных, которые поддерживают эти преобразования.

Преобразования типов данных необходимы при следующих условиях:

- Коммутация между стандартными блоками функций IEC и функциональными блоками NPAS\_POU;
- Конвертирование логических переключателей в целые переменные;
- Коммутация блоков с разными типами данных.

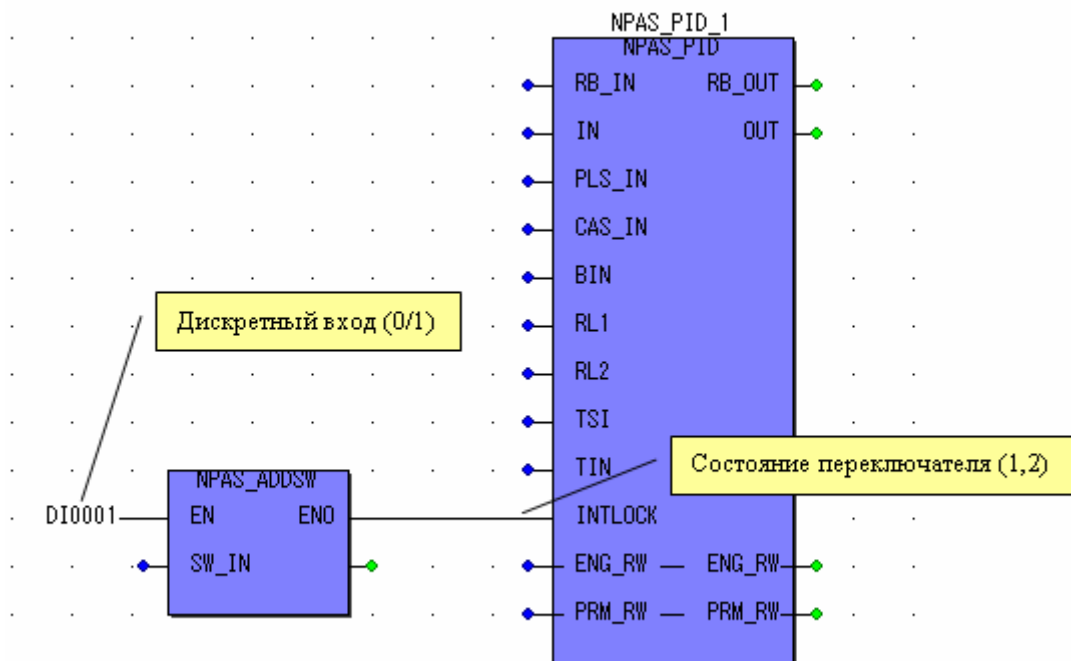
#### 2.6.5.1 Конвертирование логических переключателей в переключатели PAS\_POU.

NPAS POU не требует конвертирования данных между различными функциональными блоками.

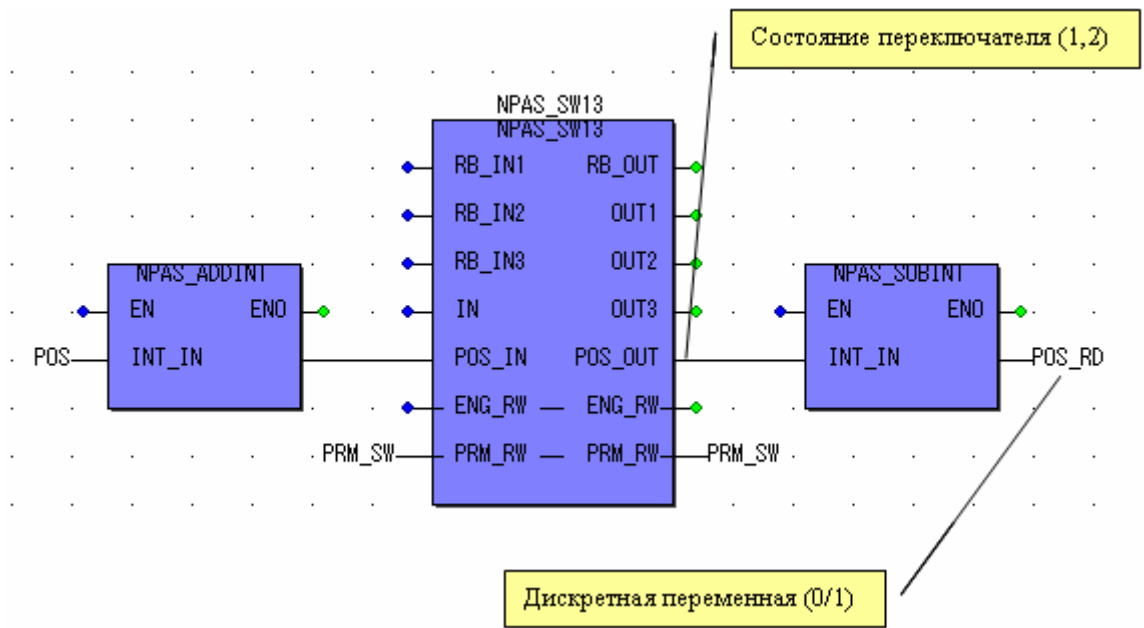
#### 2.6.5.2 Конвертирование логических переключателей в переключатели NPAS\_POU.

Многие переключатели используемые в функциональных блоках NPAS\_POU являются типа INT, а не BOOL.

- Следовательно, когда переменная типа BOOL (ON/OFF) используется для управления переключателями NPAS\_POU, она должна быть предварительно конвертирована с использованием функции NPAS\_ADD\*;



- При обратном конвертировании используются функции NPAS\_SUB\*.



**Функции конвертирования переключателей:**

ФУНКЦИЯ	ВХОД	ВЫХОД	ОПИСАНИЕ
NPAS_ADDINT	Integer	Integer	инкремент
NPAS_ADDSW	Bool	Integer	преобразует в Integer и добавляет 1
NPAS_SUBINT	Integer	Integer	декремент
NPAS_SUBSW	Integer	Bool	вычитает 1 и преобразует в Bool

**Типы входов переключателей блоков NPAS\_POU, которые требуют конвертирования:**

КЛЕММА	ОПИСАНИЕ	ВЕЛИЧИНА	ПАРАМЕТР
INTLOCK	Mode Change Interlock Switch	1 = OFF 2 = ON	ILOCKSW
TSI	Tracking Switch	1 = OFF 2 = ON	TSW
POS_IN POS_OUT	Selector Switch Position	1~3 or 4 or 9	POS
PRG_RST	Program Reset	1 = OFF 2 = ON	PRGRSTSW
PRG_END	Program End	1 = OFF 2 = ON	-
SWI	Answerback Bypass Switch	1 = OFF 2 = ON	BPSW

**2.6.5.3 Конвертирование других типов данных.**

Функции IEC включают в себя блоки конвертирования, содержащиеся в библиотеке “Type conv. FUs”.

Например, блок BOOL\_TO\_REAL конвертирует Boolean величину в Real. Более подробно см. “IEC61131 Programming training manual”.



## 2.6.6 Соединения обратной передачи данных каскадного подключения.

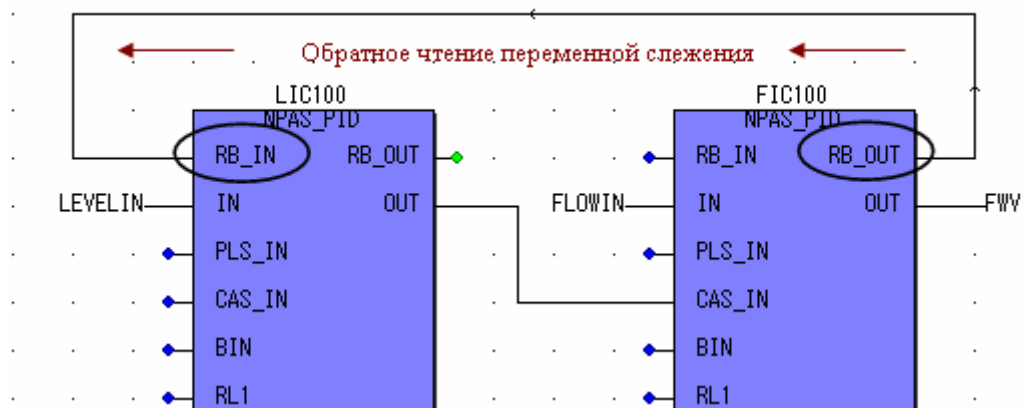
В функциональных блоках IEC61131 данные могут передаваться от одного блока к другому только в одном направлении. Однако существует много обстоятельств когда требуется обратная передача данных от блока приёмника к источнику. В частности это связано с контурами каскадного управления и поддержкой аналоговых выходов.

Функциональные блоки, которые участвуют в каскадных соединениях, имеют клеммы обратной передачи данных для реализации алгоритмов слежения величины:

- RB\_IN
- RB\_OUT

### 2.6.6.1 Контур каскадного управления.

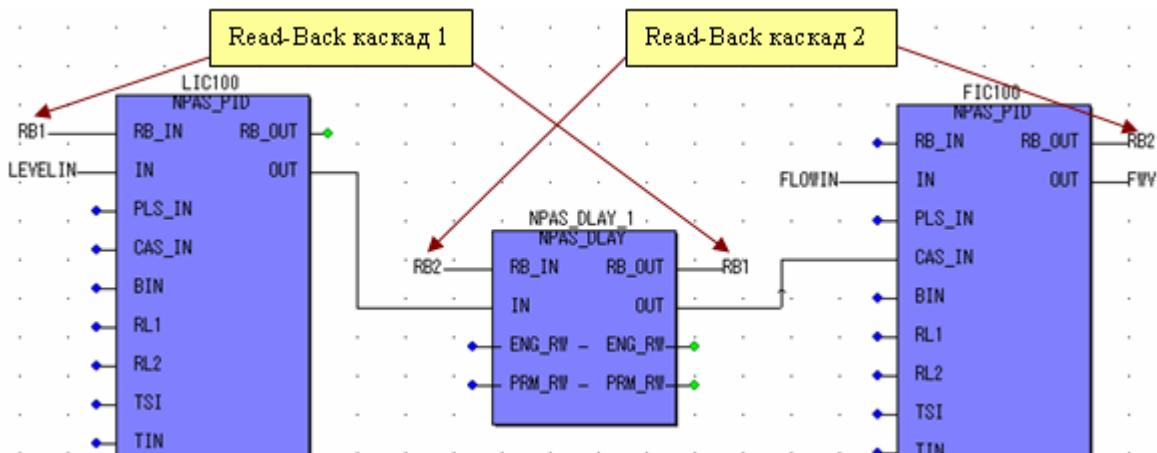
Когда контура PID регуляторов объединены в каскадный контур, выход RB\_OUT ведомого блока должен быть соединён со входом RB\_IN ведущего, как показано на рисунке:



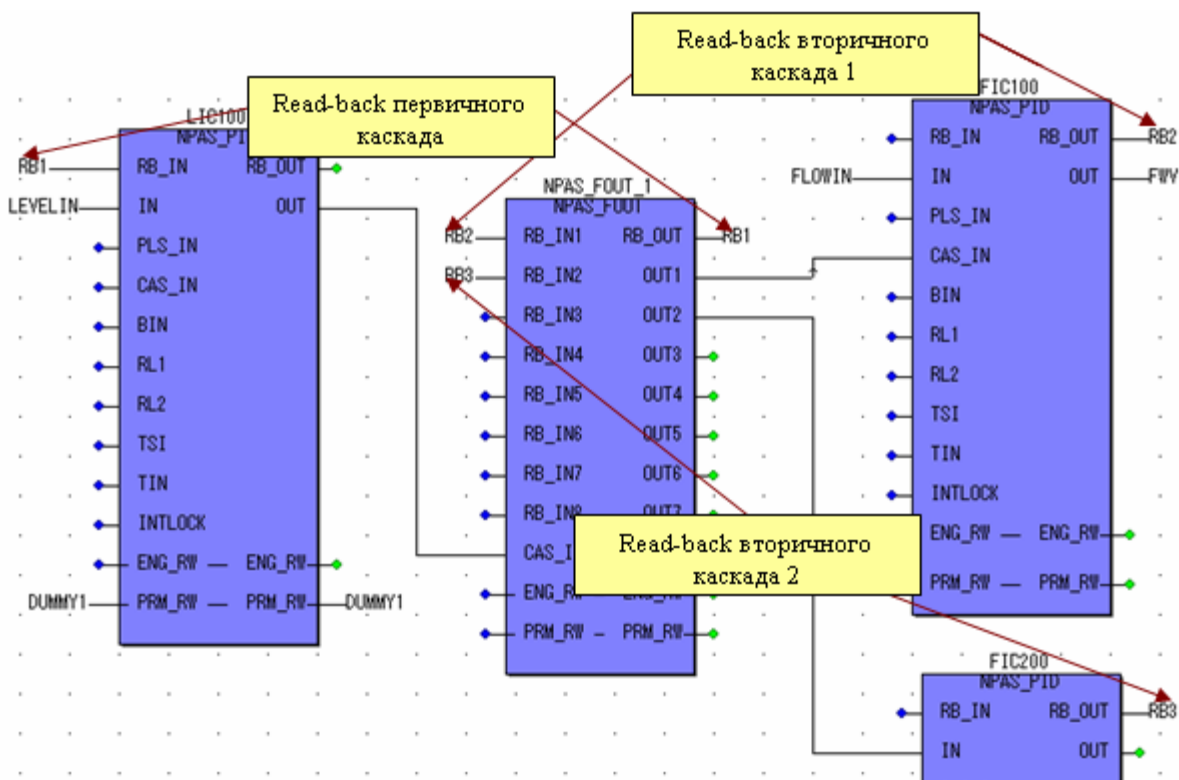
**Примечание:** Пример приведённый на рисунке показывает конфигурацию, в которой связь показана графическим способом, однако такой способ не рекомендуется стандартом IEC, поэтому при компиляции выводится предупреждающее сообщение. Чтобы этого не происходило лучше для этой цели использовать способ передачи через переменную, как показано ниже.

Такое подключение может осуществляться также и через селекторы сигналов, блоки распределения сигналов и другие промежуточные блоки, как показано на примерах ниже:

**Пример 1 – Каскадное соединение через вычислительный блок:**

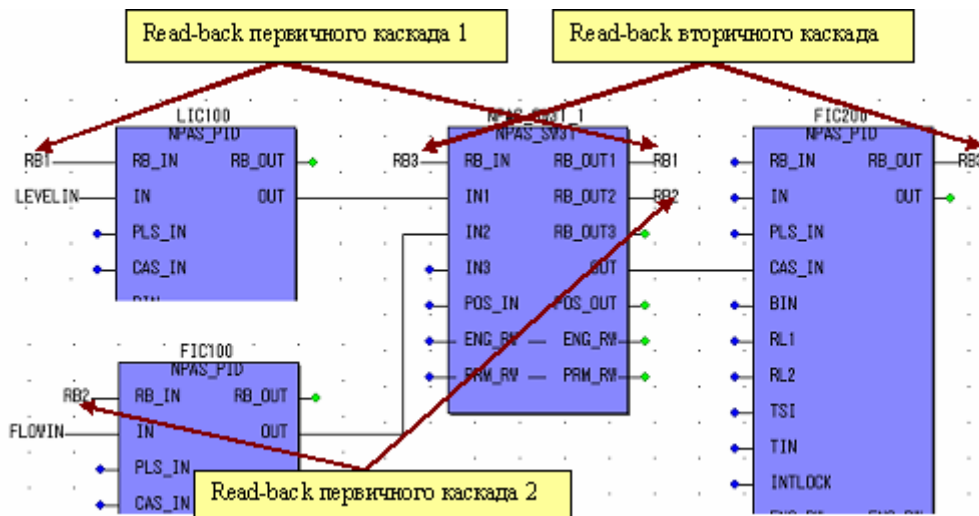


**Пример 2 – Каскадное соединение через блок распределения сигналов или блок переключателей:**



1. Заметим что, для каждого блока распределения сигналов расщеплению выходов OUT соответствует аналогичное расщепление входов RB\_IN;
2. Это справедливо также и для блоков переключателей, таких как NPAS\_SW13, у которого есть три выхода OUT и соответственно три входа RB\_IN.

**Пример 3 – Каскадное соединение через селектор сигналов:**

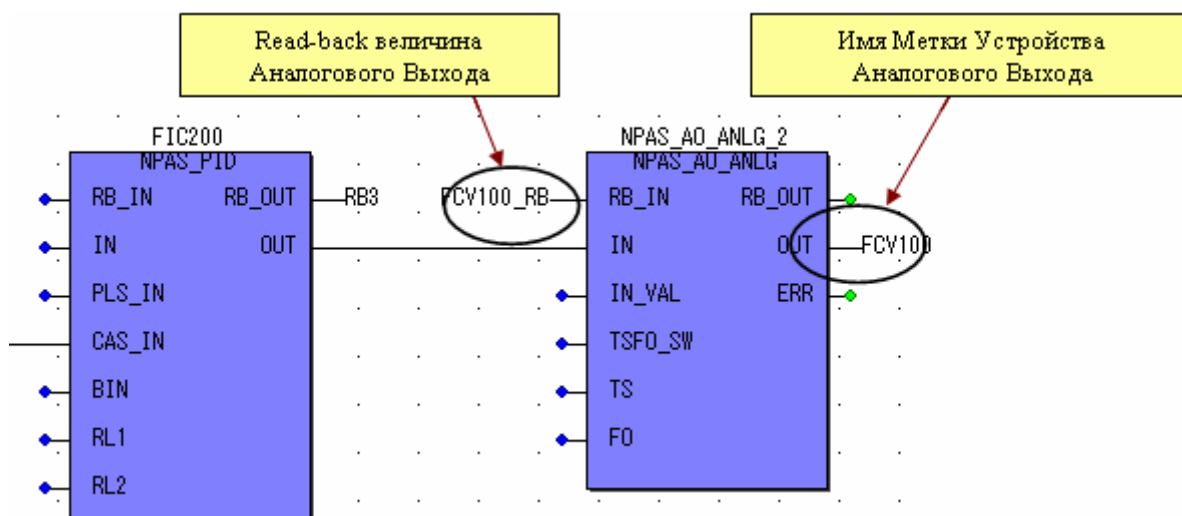


**Примечание:** Если используется блок FFSUM (NPAS\_FF\_SUM) клеммы RB\_OUT подключаются к ведущим блокам следующим образом:  
 RB\_OUT1 соединяется с RB\_IN блока, подключённого к клемме CAS\_IN;  
 RB\_OUT2 соединяется с RB\_IN блока, подключённого к клемме IN.I

**2.6.6.2 Аналоговые выходы.**

Когда подключаются аналоговые выходы (NPAS\_AO\_ANLG), то возможны два способа коммутации между функциональным блоком и меткой устройства (т.е. аппаратным ресурсом), в котором активный выход берёт информацию для загрузки в модуль выходов, с целью поддержания его активного состояния.

Если метка устройства присваивается выходу, система автоматически создаёт таг обратного чтения (**ReadBack Tag**). Например, если устройству присвоена метка CV100, то тагу обратного чтения присваивается имя CV100\_RB. Он подключается к выходу функционального блока следующим образом:



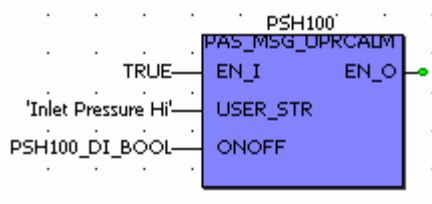
## 2.6.7 Трансляция сообщений об алармах в HMI.

Stardom имеет функцию поддержки сообщений об алармах и событиях, которая не требует программирования или настройки в FCX. Кроме того есть возможность создания аларменного сообщения в FCX, которое транслируется в VDS как стандартное аларменное сообщение. Это функциональный блок оповещательного типа (сигнализатор), который может использоваться для инициации тревожного сигнала вводимого через дискретный вход или вход PLC. Функция, используемая для создания этих алармов: **PAS\_MSG\_UPRCALM**. Имя библиотеки: **SD\_FIELD\_PF**.

Когда ON/OFF вход (типа Bool) переходит в состояние ON, сообщение подключённое ко входу строки пользователя (USER\_STR) транслируется как Аларм в VDS. Для активизации функции на вход разрешения EN должно поступать состояние TRUE.

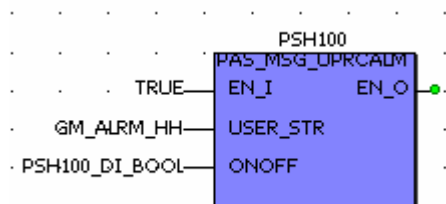
### Пример 1:

Когда на дискретный вход PSH100 поступает ON, в VDS активизируется Аларм с сообщением “Inlet Pressure Hi” с именем объекта PSH100:



### Пример 2:

Некоторые глобальные переменные имеют величины встроенные в них. Они размещаются в категории Status Definition в файле Global Variables и описаны более детально в разделе [2.4.5](#). В этом примере используется глобальная переменная GM\_ALARM\_HH, которая содержит в себе строку сообщения “HH”:



**Примечание:** Смотри раздел [2.4.2](#) для понимания назначения переменной \*\_BOOL.

## 2.6.8 Межконтроллерные коммуникации (SD\_FCXCOM\_LIB).

FCX может принять или послать данные другим контроллерам FCX (до 15 штук) через Ethernet используя функции библиотеки “SD\_FCXCOM\_LIB”.

Для этого используется два типа коммуникационных функций:

- **Неподтверждаемые коммуникации (Unconfirmed communications)** – когда отправитель не получает подтверждения в получении данных от получателя;
- **Подтверждаемые коммуникации (Confirmed Communications)** – когда отправитель получает подтверждение в получении корректных данных от получателя. Подтверждение требует большего расхода памяти чем в случае его отсутствия.

Функциональные блоки используемые для этого вида коммуникаций:

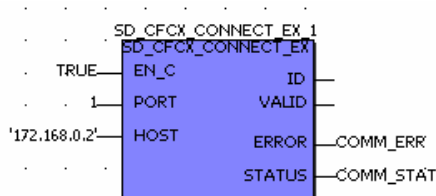
ФУНКЦИЯ	НАЗНАЧЕНИЕ
CONNECT	Блок подключения коммуникаций
READ_1V	Подтверждаемое чтение 1 величины
READ_5V	Подтверждаемое чтение 5 величин
READ_10V	Подтверждаемое чтение 10 величин
WRITE_1V	Подтверждаемая запись 1 величины
WRITE_5V	Подтверждаемая запись 5 величин
WRITE_10V	Подтверждаемая запись 10 величин
URECV_1V	Неподтверждаемое чтение 1 величины
URECV_5V	Неподтверждаемое чтение 5 величин
URECV_10V	Неподтверждаемое чтение 10 величин
USEND_1V	Неподтверждаемая запись 1 величины
USEND_5V	Неподтверждаемая запись 5 величин
USEND_10V	Неподтверждаемая запись 10 величин

Для упрощения программирования блока подключения коммуникаций CONNECT предусмотрен также ряд функциональных блоков Yokogawa:

ФУНКЦИЯ	НАЗНАЧЕНИЕ
SD_CFCX_CONNECT_EX	Используется вместо функции CONNECT
SD_CFCX_MKPARTNER	Устанавливает путь доступа к блоку CONNECT
SD_CFCX_PULSE	Используется в блоке TRIG
SD_CFCX_TRIG	Устанавливает временной интервал выполнения коммуникаций

Так как MKPARTNER и PULSE предназначены для использования внутри CONNECT\_EX и TRIG, то они, как правило, не используются самостоятельно.

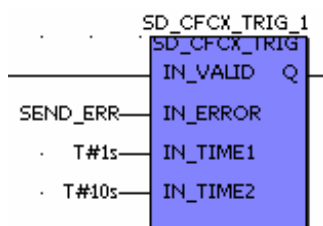
### SD\_CFCX\_CONNECT\_EX:



Содержит блок CONNECT, предназначенный для построения связи с соответствующим таким же блоком в другом FCX. Назначение контактов:

EN_C	Если вход находится в состоянии TRUE, соединение разрешено.
PORT	Устанавливается целое число в диапазоне от 1 до 16. Каждый номер определяет номер порта блока в другом FCX.
HOST	IP адрес или имя хоста FCX, к которому осуществляется подключение.
ID	Возвращает действующую величину идентификатора соединения ID для использования её в блоках Send и Receive.
VALID	Возвращает состояние TRUE, если соединение с блоком, размещенным в другом FCX, выполнено успешно.
ERROR	Возвращает состояние TRUE, если при коммуникации с блоком, размещенным в другом FCX, возникла ошибка.
STATUS	Возвращает код ошибки, возникшей при коммуникации.

### SD\_CFCX\_TRIG:



Устанавливает флаг для блока записи в соответствии с заданным временным интервалом нормальной коммуникации или временным интервалом для коммуникации в случае возникновения ошибки. Назначение контактов:

IN_VALID	Если вход находится в состоянии TRUE, то генерация флага разрешена.
IN_ERROR	Если вход находится в состоянии TRUE (т.е. при передаче возникло ошибочное состояние), то момент генерации флага базируется на IN_TIME2.
IN_TIME1	Временной интервал для нормальной коммуникации.
IN_TIME2	Временной интервал для коммуникации в случае возникновения ошибки.
Q	Выход импульса флага.

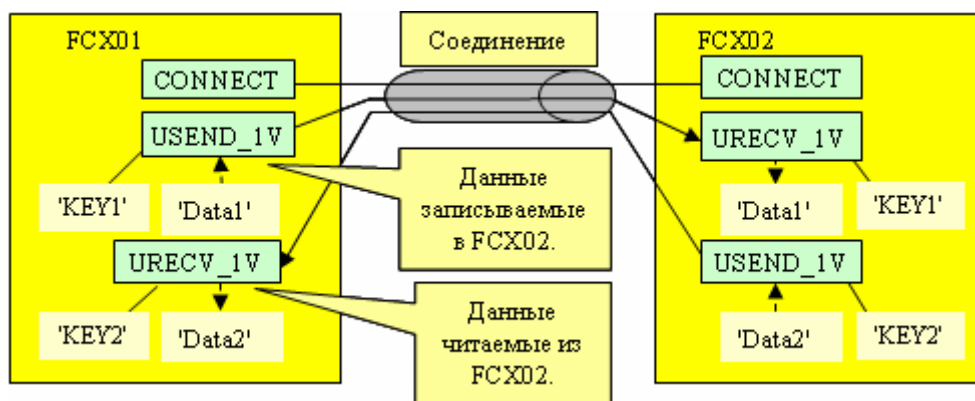
### 2.6.8.1 Неподтверждаемые коммуникации (Unconfirmed communications).

Неподтверждаемые коммуникации требуют как минимум четыре блока для передачи или чтения величины данных от одного FCX к другому:

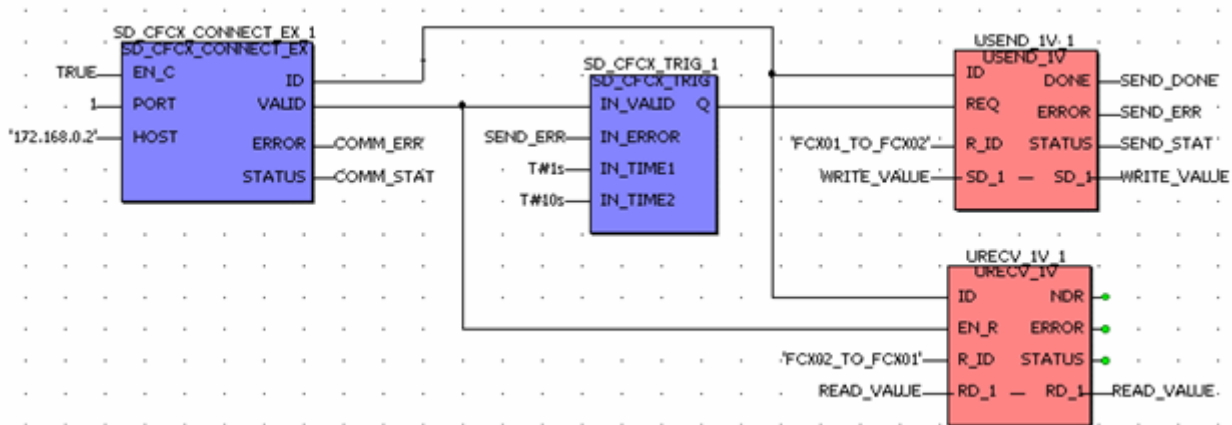
- **CONNECT** – устанавливает соединение между FCX, со спецификацией IP адреса (требуется 2 адреса). Для целей программирования блоки SD\_FCX\_CONNECT\_EX и SD\_FCX\_TRIG используются вместо блока CONNECT;
- **UREAD\_\*V** – читает данные из блока USEND в другом FCX;
- **USEND\_\*V** – записывает данные в блок UREAD в другом FCX.

Где: ‘\*’ – 1, 5 или 10, в зависимости от количества читаемых или записываемых данных.

**Примечание:** USEND и UREAD должны быть соединены парами между FCX.



**Пример создания коммуникации:**



1. Порт CONNECT\_EX устанавливает коммуникацию с FCX с IP адресом “172.168.0.2” первого порта;
2. Если связь установлена, выход VALID блока CONNECT\_EX устанавливается в состояние “TRUE” и запускается блок генерации флага TRIG;
3. Если отсутствует ошибка передающего блока USEND (флаг ошибки “SEND\_ERR” с выхода ERROR блока USEND подключен к входу IN\_ERROR блока TRIG), выход флага Q блока TRIG генерирует импульсы состояния “ON” каждую секунду (в соответствии с временным интервалом, установленным на входе IN\_TIME1 блока “T#1s”).
4. Когда блок USEND принимает активное состояние флага на свой вход REQ, он пересылает величину “WRITE\_VALUE” из SD\_1 в FCX задаваемый клеммой ID;
5. При успешном завершении коммуникации флаг успешного завершения (клемма DONE блока USEND) устанавливается в состояние “TRUE”. Если при коммуникации возникла ошибка, флаг ошибки “SEND\_ERR” с выхода ERROR блока USEND устанавливается в состояние “TRUE”.
6. Блок URECV не запускается, но принимает данные от FCX идентифицированного входом ID когда это разрешено входом EN\_R. Клемма EN\_R обычно связывается с клеммой VALID блока CONNECT\_EX.



Клеммные параметры SEND/RECEIVE блоков следующие:

ID	Целая величина, которая идентифицирует IP адрес и номер порта FCX, с которым устанавливается коммуникация. Генерируется блоком CONNECT.
R_ID	Строка символов. Одинаковые строки должны быть установлены в связанных блоках принадлежащих разным FCX.
EN_R/REQ	Разрешает выполнение функций чтения/записи.
RD_*/SD_*	Клеммы для подключения разнотипных данных соответственно для чтения/записи.
NDR/DONE	Возвращает состояние TRUE когда процесс чтения/записи завершён успешно.
ERROR	Возвращает состояние TRUE когда в процессе коммуникации возникла ошибка.
STATUS	Возвращает код ошибки (INT) если в процессе коммуникации возникла ошибка.

#### 2.6.8.2 Подтверждаемые коммуникации (Confirmed Communications).

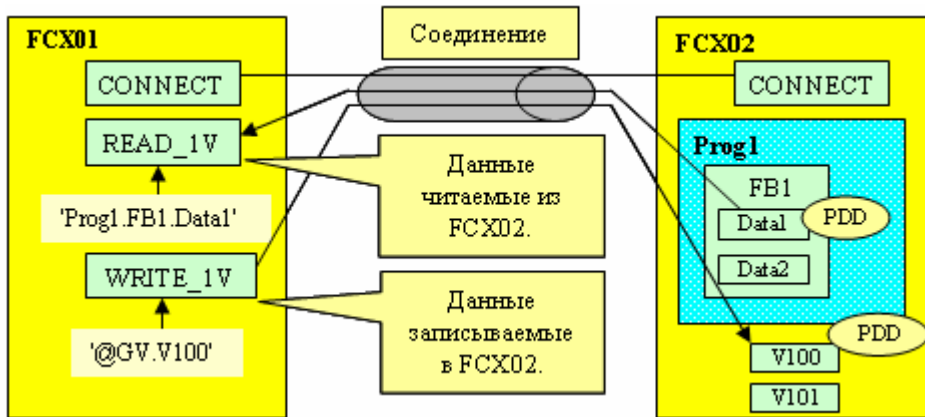
Подтверждаемые коммуникации требуют как минимум три блока для передачи/приема величин данных к/от другому FCX:

- **CONNECT** – устанавливает соединение между FCX, со спецификацией IP адреса (требуется 2 адреса). Для целей программирования блоки SD\_FCX\_CONNECT\_EX и SD\_FCX\_TRIG используются вместо блока CONNECT;
- **READ\_\*V** – читает данные из другого FCX;
- **SEND\_\*V** – записывает данные в другой FCX.

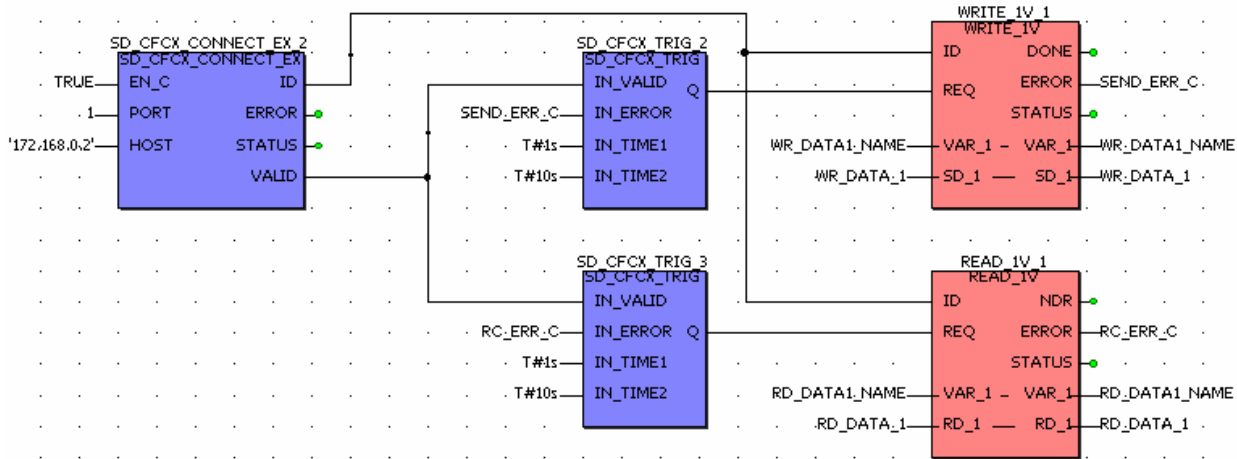
Где: '\*' – 1, 5 или 10, в зависимости от количества читаемых или записываемых данных.

В этом случае наличия пар блоков в разных FCX не является необходимым.

**PDD** – Данные, которые читаются или пишутся в локальную или глобальную переменные и должны быть специфицированы в диалоговом окне как **PDD** (Process Data Directory).



**Пример создания коммуникации:**

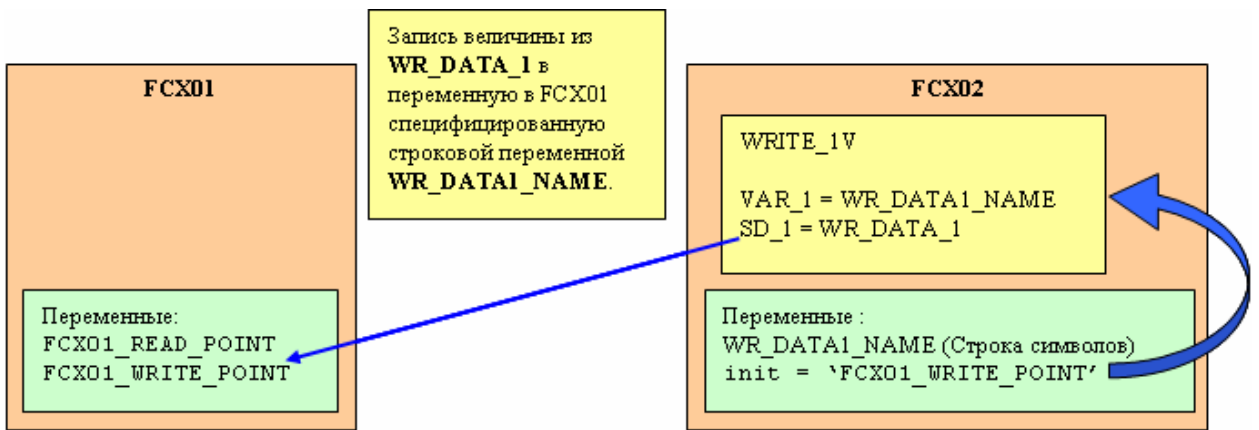


Описание функций такое же как и для неподтверждаемых коммуникаций, кроме следующих отличий:

- Отсутствует соответствие между блоками чтения и записи находящимися в разных FCX.
- Для каждого параметра, который подлежит чтению или записи, должны быть установлены два параметра:  
VAR\_\* and SD\_\*/RD\_\*.

**SD\_\* and RD\_\*** аналогичны как и для описанных выше неподтверждаемых коммуникаций, содержат величины для чтения или записи.

**VAR\_\*** - т.к. отсутствуют связанные между собой блоки чтения и записи в разных FCX, должно быть определено место куда/откуда данные должны быть записаны, прочитаны. Другими словами, VAR\_\* это вход подключаемый к строковой переменной, которая содержит в себе имя переменной в другом FCX данные из которой должны быть прочитаны или в которую они должны быть записаны.



### Определение входа VAR:

Предположим, что мы имеем два FCX называемых FCX01 и FCX02. В FCX01 определяем локальные или глобальные переменные для чтения данных из FCX02 и для записи данных в FCX02:

Name	Type	Usage	Description	Addr...	Init	Retain	PDD	OPC
<b>Default</b>								
FCX01_READ_POINT	CData_REAL	VAR_GLOBAL	Value to be r...			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
FCX01_WRITE_POINT	CData_REAL	VAR_GLOBAL	Value to be ...			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<b>User 1</b>								

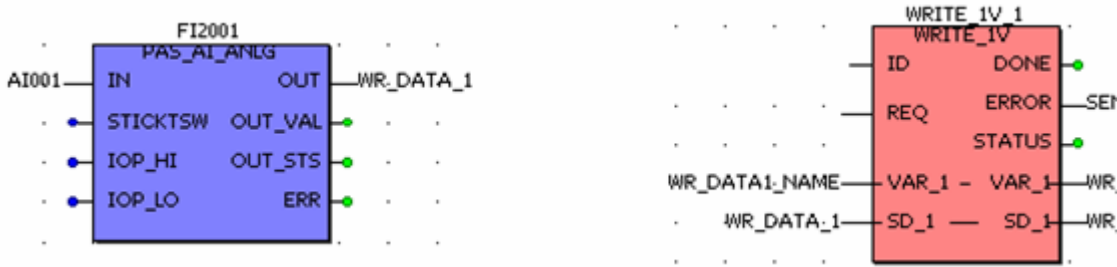
Тип данных для этого устанавливается в соответствии с типами данных функциональных блоков, которые должны подключаться и будут одинаковыми с типами данных SD и RD переменных. Маркеры **PDD** должны быть помечены.

В FCX02 в рабочем листе определяем строковую переменную (например WR\_DATA1\_NAME), и устанавливаем её начальное значение равное идентификатору имени связанной с ней глобальной переменной в FCX01 (например "@GV.FCX01\_WRITE\_POINT").

### 2.6.7.3 Программирование функций чтения/записи в Logic Designer

Как только программа чтения/записи инсталлирована, переменные могут быть использованы другими блоками.

*Пример 1 – запись величины Аналогового Входа в другой FCX с использованием подтверждаемой коммуникации (Confirmed Communication):*



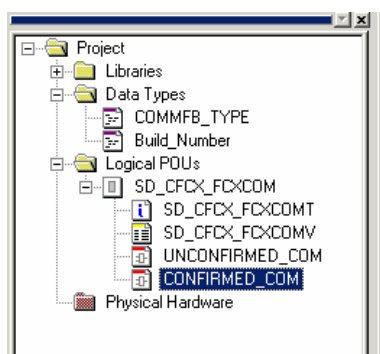
Аналоговый вход загружается в параметр “WR\_DATA\_1”. Блок WRITE\_1V пишет эти данные в другой FCX (определён в блоке CONNECT, на рисунке не показан).

*Пример 2 – чтение величины Аналогового Входа из другого FCX с использованием подтверждаемой коммуникации (Confirmed Communication):*



Входные данные читаются из другого FCX и загружаются в переменную RD\_DATA\_1 блоком READ\_1V, затем передаются блоку NPAS\_PVI.

Примеры чтения/записи как для подтверждаемой коммуникации, так и для неподтверждаемой коммуникации изложены в проектах “**Example\_E**” и может быть скопированы в пользовательские POU для изучения и использования.



## 2.7 Средства отладки (Debugging Tools).

Режим отладки разрешается в **Logic Designer** путём выбора отладочного режима (“**Debug mode**”) в меню “**Online**”. Он может работать только тогда, когда FCX подключен к PC, и в **Logic Designer** загружен такой же проект, как и в FCX. Когда контроллер запущен в рабочих листах можно наблюдать данные в реальном времени, а также использовать логический анализатор (**Logic Analyzer**).

Если **Logic Designer** запущен в отладочном режиме, процесс ввода/вывода может быть эмулирован благодаря лёгкости связывания внешних входов/выходов (**Softwiring**). При этом поддерживается эмуляция внешних связей от выходов к входам, простые задержки и запаздывания.

Логический анализатор (**Logic Analyzer**) является мощным инструментом для записи значений переменных в течение определённых интервалов времени, которые пользователь может задать путём ввода множества простых последовательностей. Используя это качество, вы можете проверить поведение вашей PLC программы путём записи переменных (в процессе работы FCX) с последующим отображением их графически в окне **Logic Analyzer**. Возможно отображение нескольких графиков изменения переменных одновременно в зависимости от их условий запуска, а также других условий.

Все записанные величины и установки **Logic Analyzer** загружаются автоматически вместе с проектом. Полученные файлы можно экспортировать в текстовые ASCII файлы с расширением \*.csv.

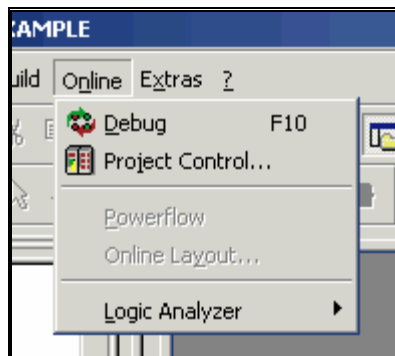
Система поддерживает использование **Logic Analyzer** с одним или несколькими различными ресурсами. Каждый ресурс имеет отдельный рабочий лист **Logic Analyzer**. Новый ресурс добавляется или удаляется в окне **Logic Analyzer** после компилирования проекта.

**Примечание:** Запись величин возможна только в режиме online.

### 2.7.1 Запуск отладочного режима.

Перед запуском отладочного режима убедитесь в том что:

1. Проект в **Logic Designer** тождественен проекту загруженному в FCX;
2. FCX находится в режиме Run (см. п. [2.3.4.2](#)).

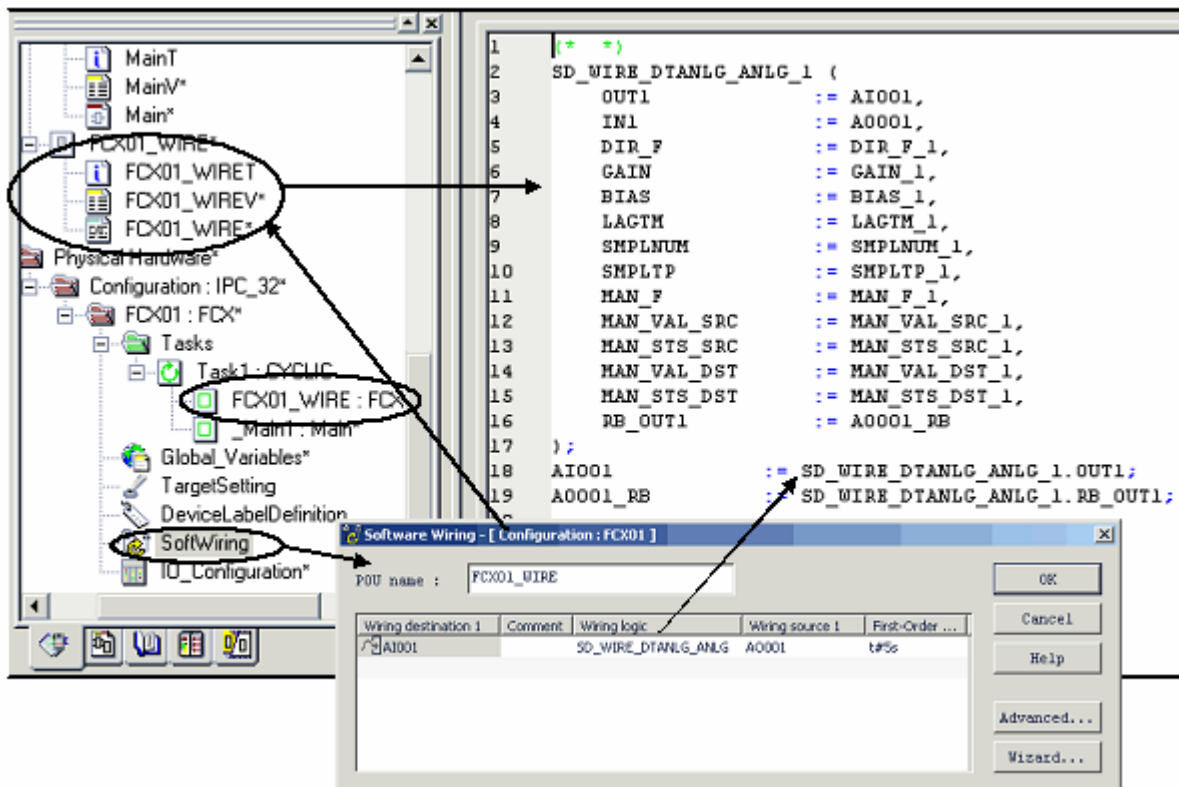


Признаком нормального запуска отладочного режима является наличие зелёной полосы отображаемой в нижней части окна **Logic Designer**.

## 2.7.2 Процедура связывания внешних входов/выходов.

Средства связывания обеспечивают возможность соединения входов и выходов между собой и эмуляции процесса ввода/вывода. Это средство на самом деле создаёт программу POU (типа ST), которая должна быть реализована как рабочая задача. В дополнение к этому имеется глобальная переменная `GS_NFIO_DISCONN`, которая должна быть установлена в состояние `TRUE` для отключения переменных ввода/вывода от входов/выходов.

Простейший способ связывания состоит в использовании **I/O Simulation Wizard**, который вызывается нажатием кнопки **“Wizard”** в диалоговом окне **“SoftWiring”**. Однако, процедура ручного связывания, показанная ниже, предполагает, что пользователь достаточно хорошо осведомлён в вопросах моделирования. Диалоговое окно **“Wizard”** показано ниже:



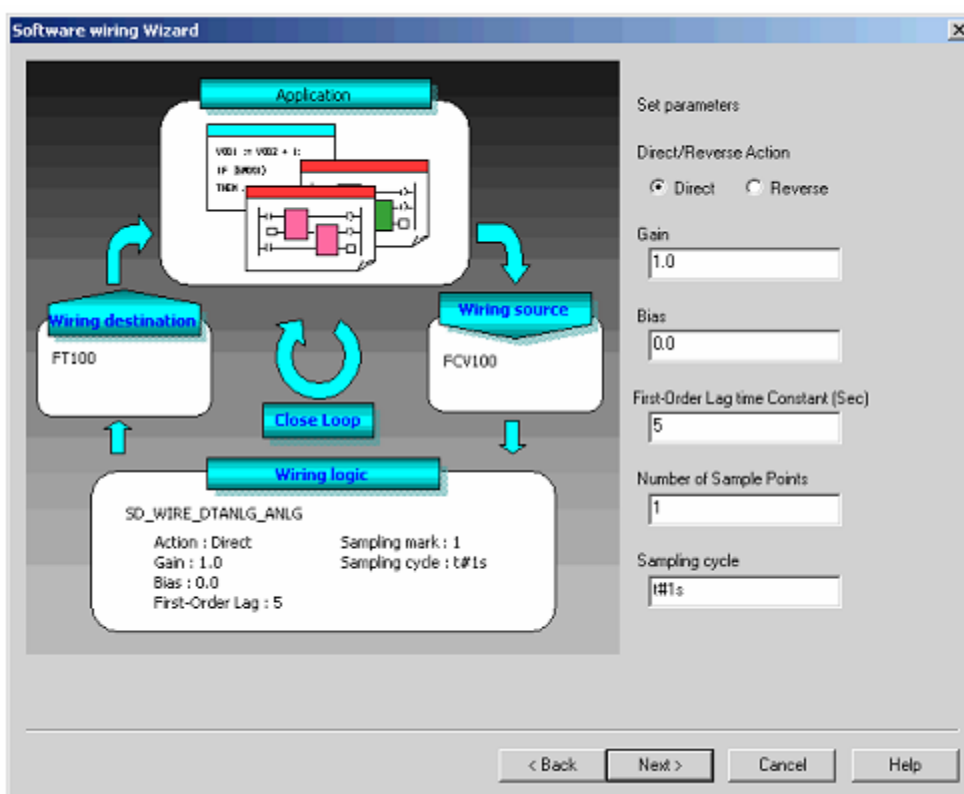
### Процедура:

1. Двойным щелчком по **“SoftWiring”** в папке **“Resources”** → **“PhysicalHardware”** → **“Configuration”** → **“FCX”** вызовите диалоговое окно **“Software Wiring”**;
2. Введите в **“POU name:”** имя программы POU, которую необходимо создать;
3. Введите имена входов и выходов, как они ранее были определены в **DeviceLabelDefinition**;
4. Выберите **“Wiring Logic”** **“SD\_WIRE\_DTANLG\_ANLG”** для аналоговых входов/выходов. Возможна установка и через главный диалог (Advanced);



5. Нажмите кнопку “OK”. Программа POU создана. Это программа структурированного текста (Structured Text program) использующая функции из библиотеки “SD\_WIRE\_PF” (Wiring Logic box);
6. Добавьте эту программу в задачу FCX;
7. Скомпонуйте (“Make”), загрузите (“Download”) программу и выберите “Debugging”.
8. Перейдите к файлу “Global Variables” → “Interface Flags section” и установите переменную: “GS\_NFIO\_DISCONN” в состояние “TRUE” для отключения обработки внешних входов/выходов и разрешения их связывания.

Другой вариант с помощью помощника, после шага 1 нажмите кнопку “Wizard...” и выводится диалоговое окно:



Здесь данные могут быть введены при помощи выпадающих меню. Когда закончите данные вводятся в диалоговое окно “SoftWiring”. Выполните действия начиная с шага 5.

### 2.7.3 Процедуры логического анализатора (Logic Analyzer).

Перед запуском Logic Analyzer убедитесь в том что:

- Скомпилированный проект загружен в FCX;
- FCX находится в запущенном состоянии;
- Система находится в режиме Online;
- Система находится в режиме Debug.

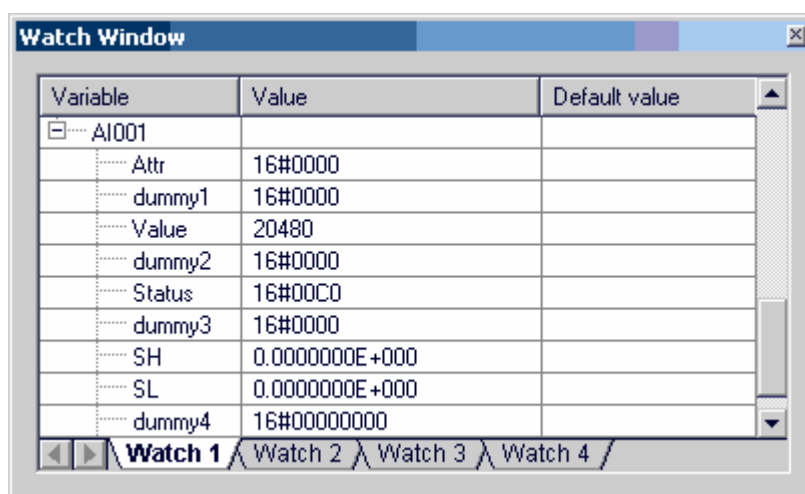
#### Процедура:

1. Откройте окно Logic Analyzer через выпадающее меню “VIEW”;
2. Добавьте переменные в Logic Analyzer. Это делается нажатием правой кнопкой мыши по переменной в программном листе и выбором опции “Add to Logic Analyzer” во всплывающем меню.  
**Примечание:** Для выполнения этих операций вы должны работать в отладочном режиме.
3. Если необходимо удалить/перенести переменные, подключённые к **Logic Analyzer**, используйте опции Delete/Remove в том же всплывающем меню;
4. Откройте контекстное меню Logic Analyzer (меню “Online” или нажатием правой кнопки мыши по вкладке) и выберите опцию меню “Window Width”. Установите начальную ширину окна в открывающемся диалоговом окне “Data Window”. Величина по умолчанию - 200 отсчётов;
5. Установите конфигурацию условия запуска. Более подробно смотри Logic Designer команда “HELP”;
6. Щёлкните по ярлыку “Start recording values” на инструментальной панели. Logic Analyzer начинает запись величин в соответствии с временным интервалом, определённым в диалоговом окне “Trigger configuration” ранее.
7. Если вы хотите отменить запись, щёлкните по ярлыку “Stop recording values” на инструментальной панели. При правильном процессе запись останавливается автоматически по окончании заданного числа записей;
8. Настройте окно Logic Analyzer если требуется;
9. Если вы хотите удалить все графики и вывести на экран переменные из **Logic Analyzer** в один приём, выберите либо в контекстном меню, либо в подменю “Online” опцию “Clear Curves”. Если отображается инструментальная панель, вы можете также использовать ярлык “Clear Curves”.
10. Если вы хотите сохранить отображаемые графики, щёлкните по ярлыку “Capture Curves” на инструментальной панели.
11. Если вы хотите экспортировать записанные данные в \*.csv файл, выберите опцию “Export Data” в окне “Online – Logic Analyzer”.

## 2.7.4 Окно наблюдения (Watch Window).

Окно наблюдения может использоваться для отображения состояния переменных из различных рабочих листов одновременно. Оно может быть также использовано для отладки элементов с типами данных определённых пользователем, таких как массивы или структуры данных.

В окно наблюдения вы можете добавлять или удалять переменные. Ограничений на количество переменных отображаемых в окне наблюдения нет. Кроме того, в окне можно задать значение переменной вручную или переписать его. Это выполняется путём маркирования переменной в рабочем листе окна и выбора опции в контекстном меню переменной “**Debug dialog...**”.



### *Как открыть/скрыть окно наблюдения:*

Выберите опцию “**Watch Window**” в выпадающем меню “**View**” или щёлкните по ярлычку “**Watch window**” на инструментальной панели. В зависимости от предыдущего состояния окно наблюдения становится видимым или скрытым.

**Примечание:** Можно открыть окно также через опцию контекстного меню переменной “**Open Watch Window...**” (возможно только в режиме online).

### *Окно наблюдения содержит в себе следующую информацию:*

- “**Variable**” – Отображает имя переменной. Определённые пользователем типы данных, таких как массивы или структуры данных маркированы символом '+'. Для отображения элементов этих типов данных необходимо щёлкнуть по этому символу.
- “**Value**” – Отображает текущее значение переменной.
- “**Default Value**” – Отображает значение по умолчанию (если определено).
- “**Type**” – Отображает тип данных.

- “Instance” – Отображает реализационный путь, где эта переменная используется. Путь всегда содержит конфигурацию (**Configuration**), ресурс (**Resource**), задачу (**Task**), связанное с ним имя программы (**Program Name**) и имя переменной (**Variable Name**).

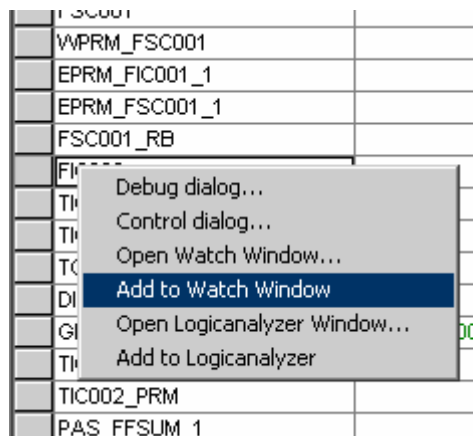
Окно наблюдения позволяет пользователю листать страницы. Каждая из страниц может использоваться независимо. Индивидуально любая страница может быть открыта выбором закладки в нижней части окна.

### Как отобразить перекрёстные ссылки на переменную, отображаемую в окне наблюдения:

В окне можно получить прямой доступ к информации о перекрёстных ссылках на переменную. Для этого необходимо в окне пометить выбранную переменную. Помеченная переменная автоматически маркируется и в окне перекрёстных ссылок.

### Как добавить переменные в окно наблюдения:

В окно можно добавить переменные из списка объявления переменных, из текстового или графического рабочего листа.



1. Пометьте переменную в рабочем листе или в списке объявления переменных в режиме online;
2. Нажмите правой кнопкой мыши по переменной, при этом откроется контекстное меню;
3. Выберите в меню опцию “Add to Watch Window”, переменная добавляется в окно наблюдения.

### Как удалить переменные из окна наблюдения:

Из окна можно удалить одну переменную или очистить всё окно:

- Для удаления одной переменной пометьте её и нажмите клавишу клавиатуры “Delete”. Альтернативно можно открыть контекстное меню переменной нажатием правой кнопки мыши и выбрать опцию “Delete”.

---

**Примечание:** Допускается процедура удаления элемента структуры данных или массива определённых пользователем.

- Для удаления всех переменных необходимо открыть контекстное меню любой переменной нажатием правой кнопки мыши и выбрать опцию “Clear”.

## 2.8 Приложения HTML (HTML Applications).

FCX имеет встроенный Web сервер, который может быть использован для отображения показаний операторного типа непосредственно, без использования операторской станции HMI.

Стандартная Web страница может быть сгенерирована с использованием Frontpage или другого редактора и загружена через FTP. Web страница требует Java Objects для подключения к данным FCX.

Для подключения к FCX через FTP, с использованием FTP пакета, войдите в FCX используя его IP адрес. Входные параметры по умолчанию:

- Имя пользователя (**Username**): “stardom”;
- Пароль (**Password**): “YOKOGAWA”.

Загрузите Web страницы в папку WWW. Главная Web страница должна иметь имя INDEX.HTM.

Для наблюдения Web страниц FCX, введите IP адрес FCX в Web браузер. Должна открыться главная страница.

Для использования **Java Objects** в Web страницах необходимо пройти дополнительный специальный курс обучения.