

Руководство пользователя по программированию ПЛК

**В
CoDeSys 2.3**



**S m a r t
Software
Solutions**

**Copyright © 1994, 1997, 1999, 2001, 2002, 2003, 2004, 2005, 2006 3S - Smart Software Solutions GmbH
All rights reserved.**

Copyright © 2003, 2004, 2005, 2006 ПК Пролог (Русская редакция)

Текст данного документа тщательно проверен. Однако практически невозможно гарантировать абсолютное отсутствие ошибок. Мы будем благодарны вам за замечания и предложения по улучшению текста и содержимого документа.

Trademark

Intel is a registered trademark and 80286, 80386, 80486, Pentium are trademarks of Intel Corporation.
Microsoft, MS and MS-DOS are registered trademarks, Windows is a trademark of Microsoft Corporation.

Документ подготовлен:

3S - Smart Software Solutions GmbH
Memminger Straße 151
D-87439 Kempten
Тел.: +49 831 5 40 31 - 0
Факс: +49 831 5 40 31 - 50

Русская редакция:

ПК Пролог
21400, Россия, г. Смоленск, ул. Октябрьской революции, 9
Тел.: +7 4812 38-29-31
Тел./Факс: +7 4812 65-81-71

Текст данного документа опирается на международную версию CoDeSys с английским интерфейсом. Наименования команд, тексты диалогов и сообщений соответствуют указанной версии.

Последнее обновление 02.02.2006

Редакция RU 2.4, для CoDeSys V2.3.6.x

Оглавление

1	Краткое представление CoDeSys	1-1
1.1	Что такое CoDeSys	1-1
1.2	Представление о работе в CoDeSys	1-1
1.3	Состав базовой пользовательской документации по CoDeSys	1-3
2	Что есть, что в CoDeSys	2-1
2.1	Компоненты проекта	2-1
2.2	Языки программирования	2-9
2.2.1	Список инструкций (IL)	2-10
2.2.2	Модификаторы и операторы IL	2-10
2.2.3	Структурированный текст (ST)	2-11
2.2.4	Язык последовательных функциональных схем (SFC)	2-18
2.2.5	Язык функциональных блочных диаграмм (FBD)	2-23
2.2.6	Непрерывные функциональные схемы (CFC)	2-23
2.2.7	Язык релейных диаграмм (LD)	2-24
2.3	Отладка и online функции	2-25
3	Пишем простой пример	3-1
3.1	Блок управления светофором	3-1
3.2	Визуализация примера	3-13
4	Работа в системе программирования CoDeSys	4-1
4.1	Главное окно	4-1
4.2	Опции проекта	4-3
4.3	Управление проектом	4-18
4.4	Управление объектами проекта	4-48
4.5	Основные функции редактирования	4-56
4.6	Основные функции Online	4-62
4.7	Работа с окнами	4-77
4.8	Помощь	4-78
5	Редакторы CoDeSys	5-1
5.1	Общие элементы редакторов	5-1
5.2	Редактор раздела объявлений	5-3
5.2.1	Работа в редакторе объявлений	5-3
5.2.2	Редактор раздела объявлений в режиме Online	5-11
5.2.3	Директивы компилятора	5-12
5.3	Текстовые редакторы	5-20
5.3.1	Работа в текстовых редакторах	5-20
5.3.2	Редактор языка IL	5-24
5.3.3	Редактор языка ST	5-25
5.4	Графические редакторы	5-25
5.4.1	Работа в графических редакторах	5-25
5.4.2	Редактор FBD	5-30
5.4.3	Редактор LD	5-36
5.4.4	Редактор SFC	5-43
5.4.5	Редактор CFC	5-52
6	Ресурсы	6-1
6.1	Обзор ресурсов	6-1
6.2	Глобальные и конфигурационные переменные, файл комментариев	6-2
6.2.1	Глобальные переменные	6-3
6.2.2	Конфигурационные переменные	6-7
6.2.3	Файл комментариев переменных	6-8
6.3	Конфигурация тревог (Alarm Configuration)	6-9

6.3.1	Обзор	6-9
6.3.2	Общая информация и терминология	6-10
6.3.3	Классы тревог	6-11
6.3.4	Группы тревог	6-14
6.3.5	Запись тревог	6-16
6.3.6	'Extras' меню: Settings (настройки)	6-17
6.4	Менеджер библиотек (Library Manager)	6-18
6.5	Бортжурнал (Log)	6-20
6.6	Конфигуратор ПЛК (PLC Configuration)	6-23
6.6.1	Обзор	6-23
6.6.2	Работа в редакторе конфигуратора ПЛК	6-25
6.6.3	Общие параметры конфигурации ПЛК	6-26
6.6.4	Диалог специфической настройки параметров	6-27
6.6.5	Конфигурация модулей ввода/вывода	6-28
6.6.6	Конфигурация канала	6-31
6.6.7	Конфигурирование модулей Profibus	6-31
6.6.8	Конфигурирование CANOpen-модулей	6-39
6.6.9	Конфигурация ведомого CANOpen-устройства (CANOpen Slave)	6-44
6.6.10	Конфигурирование модулей DeviceNet	6-48
6.6.11	Конфигурация ПЛК в режиме Online	6-53
6.6.12	Сканирование аппаратуры/ Состояние/ Диагностика ПЛК	6-53
6.7	Конфигуратор задач (Task Configuration)	6-54
6.7.1	Работа в конфигураторе задач	6-54
6.7.2	Системные события	6-58
6.7.3	Конфигуратор задач в режиме online	6-59
6.8	Менеджер рецептов (Watch and Receipt Manager)	6-61
6.8.1	Обзор	6-61
6.8.2	Менеджер рецептов в режиме Offline	6-61
6.8.3	Менеджер рецептов в режиме Online	6-63
6.9	Трассировка (Sampling Trace)	6-64
6.9.1	Обзор	6-64
6.9.2	Конфигурация трассировки	6-65
6.9.3	Управление процессом трассировки	6-66
6.9.4	Отображение данных	6-67
6.9.5	Сохранение данных трассировки ('Extras' 'Save trace values')	6-69
6.9.6	Внешние конфигурации трассировки ('Extras' 'External Trace Configurations')	6-70
6.10	Рабочая область (Workspace)	6-70
6.11	Менеджер параметров (Parameter Manager)	6-71
6.11.1	Обзор и подключение	6-71
6.11.2	Редактор менеджера параметров. Обзор	6-72
6.11.3	Типы списков параметров и их атрибуты	6-73
6.11.4	Управление списками параметров	6-75
6.11.5	Редактирование списка параметров	6-77
6.11.6	Менеджер параметров в режиме online	6-78
6.11.7	Экспорт/импорт списков параметров	6-79
6.12	Настройка целевой платформы (Target Settings)	6-80
6.13	ПЛК-Браузер (PLC-Browser)	6-81
6.13.1	Общие сведения	6-81
6.13.2	Набор команд ПЛК-Браузера	6-82
6.13.3	Макро расширения команд ПЛК-Браузера	6-83
6.13.4	Вспомогательные команды ПЛК-Браузера	6-84
6.14	Инструменты (Tools)	6-84
6.14.1	Свойства доступных инструментов (Object Properties)	6-85
6.14.2	Настройка команд инструментов	6-88
6.14.3	Часто задаваемые вопросы по инструментам	6-90
7	ENI7-1	

7.2 Условия работы с ENI базой данных в проекте	7-1
7.3 Работа с ENI базой данных в проекте CoDeSys	7-2
7.4 Категории объектов в базе данных проекта	7-2
8 DDE интерфейс	8-1
8.1 DDE интерфейс CoDeSys	8-1
8.2 DDE обмен посредством GatewayDDE Server	8-2
9 Менеджер лицензирования CoDeSys	9-1
9.1 Обзор	9-1
9.2 Создание лицензированных библиотек в CoDeSys	9-1
10 Приложения	10-1
<i>Приложение А: Операторы и функции МЭК</i>	<i>10-1</i>
10.1 Арифметические операторы	10-1
10.2 Битовые операторы	10-4
10.3 Операторы сдвига	10-6
10.4 Операторы выборки	10-9
10.5 Операторы сравнения	10-11
10.6 Адресные операторы и вспомогательные функции	10-13
10.7 Оператор вызова	10-16
10.8 Явное преобразование типов	10-16
10.9 Математические функции	10-23
<i>Приложение В: Операнды в CoDeSys</i>	<i>10-30</i>
10.10 Константы	10-30
10.11 Переменные	10-32
10.12 Адреса	10-34
10.13 Функции в роли операндов	10-35
<i>Приложение С: Типы данных CoDeSys</i>	<i>10-36</i>
10.14 Элементарные типы данных	10-36
10.15 Пользовательские типы данных	10-37
<i>Приложение D: Библиотеки CoDeSys</i>	<i>10-44</i>
10.16 Стандартная библиотека Standard.lib	10-44
10.16.1 Строковые функции	10-44
10.16.2 Переключатели	10-48
10.16.3 Детекторы импульсов	10-49
10.16.4 Счетчики	10-51
10.16.5 Таймеры	10-53
10.17 Библиотека UTIL.LIB	10-56
10.17.1 ВCD преобразования	10-56
10.17.2 Бит/байт функции	10-57
10.17.3 Дополнительные математические функции	10-58
10.17.4 Регуляторы	10-60
10.17.5 Генераторы сигналов	10-62
10.17.6 Преобразования аналоговых сигналов	10-64
10.17.7 Аналоговые компараторы	10-66
10.18 Библиотека AnalyzationNew.lib	10-67
10.19 Системные библиотеки CoDeSys	10-67
<i>Приложение E: Краткий справочник по операторам и компонентам библиотек</i>	<i>10-68</i>
<i>Приложение F: Командная строка / командный файл</i>	<i>10-73</i>
10.20 Командная строка	10-73
10.21 Командный файл (cmdfile)	10-73

Приложение G: Сименс импорт. 10-82

Приложение H: Опции целевых систем 10-89

10.22 Системные опции целевых платформ (Target Platform)	10-89
10.23 Опции распределения памяти (Memory Layout).	10-95
10.24 Опции общей категории (General)	10-96
10.25 Опции категории Сетевая функциональность	10-98
10.26 Опции категории Визуализация	10-99
<i>Приложение I: Использование клавиатуры</i>	<i>10-102</i>
<i>Приложение J: Ошибки и предупреждения компилятора</i>	<i>10-105</i>

10.27 Предупреждения	10-105
10.28 Ошибки	10-111

1 Краткое представление CoDeSys

1.1 Что такое CoDeSys

CoDeSys - это современный инструмент для программирования контроллеров (**CoDeSys** образуется от слов **C**ontrollers **D**evelopment **S**ystem).

CoDeSys предоставляет программисту удобную среду для программирования контроллеров на языках стандарта МЭК 61131-3. Используемые редакторы и отладочные средства базируются на широко известных и хорошо себя зарекомендовавших принципах, знакомых по другим популярным средам профессионального программирования (такие, как Visual C++).

1.2 Представление о работе в CoDeSys

С чего начинается программный проект?

Прежде всего нужно дать проекту новое имя, оно же послужит и названием файла проекта.

Первый программный компонент (**POU – Program Organization Unit**) помещается в новый проект автоматически и получает название PLC_PRG. Именно с него и начинается выполнение процесса (по аналогии с функцией main в языке C), из него будут вызываться другие программные блоки (программы, функции и функциональные блоки).

Нет необходимости писать вручную текст для PLC_PRG, поскольку конфигурация задачи определяется на вкладке проекта Task Configuration. Подробнее это будет описано в главе, посвященной Task Configuration.

Проект содержит ряд разнородных объектов POU, данных разных типов, элементов визуализации и ресурсов.

Организатор объектов (Object Organizer) управляет списком всех объектов Вашего проекта.

Как создать собственный проект?

Для начала вы определяете конфигурацию ПЛК в соответствии с аппаратными средствами своего контроллера.

Затем вы создаете программные компоненты, необходимые для решения проблемы.

Далее вы пишете программный код для созданных компонентов на выбранных языках.

Сразу после завершения программирования, вы компилируете проект и исправляете ошибки, если они есть.

Как проверить проект?

Когда все ошибки устранены, можно приступить к отладке.

Включите флажок эмуляция (**simulation**) и «подключитесь» к контроллеру. Теперь вы в режиме Online.

Откройте окно с конфигурацией ПЛК (**PLC Configuration**) и проверьте правильность выполнения проекта. Для этого измените вручную входные данные и убедитесь, что выходы контроллера отреагировали нужным образом. Если необходимо, вы можете наблюдать значения переменных в программных компонентах. Используя менеджер просмотра и заказа значений переменных (короче, менеджер рецептов) **Watch and Receipt Manager**, вы сможете задать список переменных, значения которых необходимо наблюдать.

Отладка

В случае ошибок в работе кода вы можете задать точки останова. Когда процесс остановлен в определенной точке, вы можете просмотреть значения переменных проекта в данный момент времени. Выполняя проект в пошаговом режиме (single step), вы можете проверить логическую корректность своих программ.

Дополнительные возможности режима Online

В процессе отладки вы можете устанавливать значения переменных программ, задавать фиксированные значения на входы и выходы контроллера, контролировать последовательность исполнения процесса и определить место в программе, которое сейчас выполняется. Используя трассировку (Sampling Trace), отслеживать в графическом представлении изменения значений переменных за определенный промежуток времени.

Когда проект закончен и отлажен, переходите к окончательной доводке в рабочих условиях на реальном "железе". Естественно, при этом полностью доступны все отладочные функции.

Дополнительные возможности CoDeSys

Весь проект может быть экспортирован в текстовый файл и сохранен в печатном виде.

Средства коммуникации **CoDeSys** включают символьный и DDE интерфейсы. **Коммуникационный сервер, OPC и DDE серверы** входят в стандартный пакет поставки.

Путем выбора целевой платформы CoDeSys позволяет использовать один проект в различных системах.

Сетевые переменные общего доступа и **Менеджер параметров** обеспечивают средства сетевого взаимодействия контроллеров.

ENI: инжиниринговый интерфейс применяется совместно с любыми системами управления версиями через автономный ENI сервер. Программные компоненты CoDeSys сохраняются в единой базе данных, доступной другим пользователям. ENI сервер служит хранилищем конструкторских данных не только для CoDeSys, но и для сторонних программных инструментов.

CoDeSys позволяет задействовать «фирменные» программные инструменты. Файлы, включающие исполняемый код, могут быть скомпонованы с кодом проекта и загружены в контроллер.

Созданная в **CoDeSys** визуализация может выполняться не только в среде программирования, но и в целевой платформе или в **Web**. Это позволяет контролировать процесс и управлять им через Интернет.

1.3 Состав базовой пользовательской документации по CoDeSys

Модуль	Документы	Файлы
Среда программирования CoDeSys	Печатное руководство и встроенная система помощи Первые шаги с CoDeSys (пример)	Manual_V23_RU.pdf First Steps with CoDeSys RU.pdf
Gateway Server	Концепции, установка, встроенная система помощи, интерфейс и настройка (открывается двойным щелчком мыши на иконке в панели задач)	Gateway Manual.pdf
OPC Server	OPC-Server V2.0, установка и применение	OPC_20_How_to_use.pdf
CoDeSys Visualization	Описание CoDeSys визуализации, включая CoDeSys HMI, Target- и Web-Visualization	CoDeSys_Visu_RU.pdf
SoftMotion	Описание применения и библиотек SoftMotion	SoftMotion_Manual_V23.pdf
Библиотеки	Standard.lib и Util.lib описаны в печатном руководстве по программированию. Для каждой системной библиотеки CoDeSys существует отдельный документ <library name>.pdf SoftMotion библиотеки: см. SoftMotion-документацию.	<SysLib-Name>.pdf UserManual_V23_RU.pdf
ENI Server	Установка и настройка ENI Сервера, управление версиями, работа с внешней базой данных. Настройка ENI в CoDeSys: описана в печатном руководстве по программированию. ENI Admin, ENI Control и ENI Explorer: см. встроенные системы помощи.	EniServerQuickstart.pdf UserManual_V23_RU.pdf

2 Что есть что в CoDeSys

2.1 Компоненты проекта

Проект

Проект включает следующие объекты: POU, типы данных, визуализации, ресурсы, библиотеки. Каждый проект сохраняется в отдельном файле.

POU (Program Organization Unit)

К программным компонентам (POU) относятся функциональные блоки, функции и программы. Отдельные POU могут включать действия (подпрограммы).

Каждый программный компонент состоит из раздела объявлений и кода. Для написания всего кода POU используется только один из МЭК языков программирования (IL, ST, FBD, SFC, LD или CFC).

CoDeSys поддерживает все описанные стандартом МЭК компоненты. Для их использования достаточно включить в свой проект библиотеку `standard.lib`.

POU могут вызывать другие POU, но рекурсии недопустимы.

Функция

Функция – это POU, который возвращает только единственное значение (которое может состоять из нескольких элементов, если это битовое поле или структура). В текстовых языках функция вызывается как оператор и может входить в выражения.

При объявлении функции необходимо указать тип возвращаемого значения. Для этого после имени функции нужно написать двоеточие и тип.

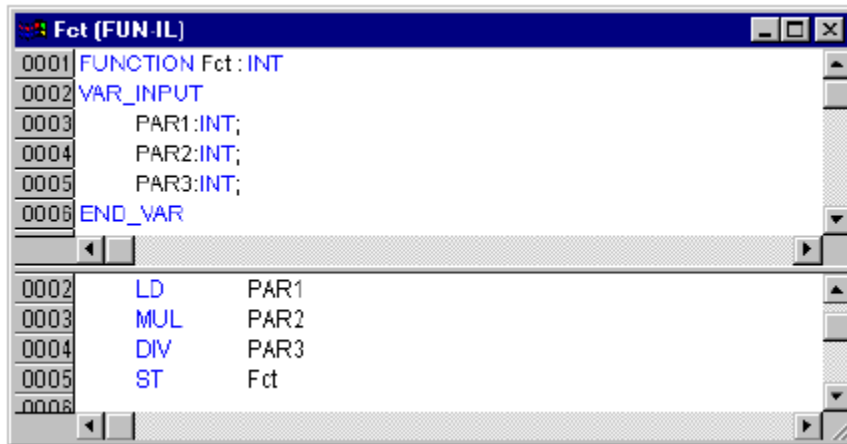
Правильно объявленная функция выглядит следующим образом:

```
FUNCTION Fct: INT;
```

Имя функции используется как выходная переменная, которой присваивается результат вычислений.

Объявление функции должно начинаться с ключевого слова `FUNCTION` и заканчиваться ключевым словом `END_FUNCTION`. Вот пример функции, написанной на IL, которая использует три входных переменных и возвращает результат деления произведения первых двух на третью.

Пример функции, написанной на языке IL:



В языке ST вызов функции может присутствовать в выражениях как операнд.

В SFC функция вызывается только из шага или перехода.

Функция не имеет внутренней памяти. Это означает, что функция с одними и теми же значениями входных переменных всегда возвращает одно и то же значение.

Например:

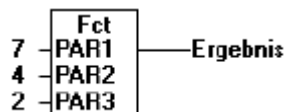
На IL:

```
LD 7
Fct 2,4
ST Result
```

На ST:

```
Result := Fct (7, 2, 4);
```

На FBD:



Внимание: объявление в функции RETAIN локальной переменной не приведет к желаемому результату. Не пытайтесь создать локальные энергонезависимые переменные в функциях!

Примечание: имена перечисленных ниже функций зарезервированы для описанных целей:

В проекте можно определить функцию с именем **CheckBounds**, которая используется для проверки выхода за границы массива (Подробнее см. описание арифметических функций в приложении).

С помощью функций: **CheckDivByte**, **CheckDivWord**, **CheckDivDWord** и **CheckDivReal** осуществляется контроль деления на 0 (См. DIV).

Специализированные функции **CheckRangeSigned** и **CheckRangeUnsigned** контролируют границы диапазонов переменных (См. Типы данных).

Функциональный блок

Функциональный блок - это POU, который принимает и возвращает произвольное число значений. В отличие от функции функциональный блок не формирует возвращаемое значение.

Объявление функционального блока начинается с ключевого слова `FUNCTION_BLOCK`

и заканчивается ключевым словом `END_FUNCTION_BLOCK`.

Ниже приведен пример функционального блока, написанного на IL, который имеет две входных и две выходных переменных. Значение выходной переменной `MULERG` равно произведению значений двух входных переменных, а значение `VERGL` определяется в результате сравнения значений входных переменных.

Пример функционального блока:

```

0001 FUNCTION_BLOCK FUB
0002 VAR_INPUT
0003     PAR1:INT;
0004     PAR2:INT;
0005 END_VAR
0006 VAR_OUTPUT
0007     MULERG:INT;
0008     VERGL:BOOL;
0009 END_VAR
0010
0001 LD PAR1
0002 MUL PAR2
0003 ST MULERG
0004
0005 LD PAR1
0006 EQ PAR2
0007 ST VERGL
0008
0009
    
```

Экземпляры функционального блока

Определение функционального блока подобно определению типа данных. Для работы с функциональным блоком необходимо объявить (создать) его экземпляр. Один функциональный блок может иметь произвольное число экземпляров, каждый из которых имеет собственные независимые данные (память).

Каждый экземпляр функционального блока получает свой собственный идентификатор (имя экземпляра) и свои данные, содержащие входные, выходные и внутренние переменные. Экземпляры функционального блока объявляются глобально или локально как переменные, имеющие тип соответствующего функционального блока.

Пример объявления экземпляра с идентификатором `INSTANCE` функционального блока `FUB`:

```
INSTANCE: FUB;
```

Вызов экземпляра функционального блока происходит с помощью его имени. Входные и выходные переменные доступны вне функционального блока, а внутренние переменные доступны только в самом блоке.

Пример использования входных переменных:

функциональный блок fb имеет входную переменную in1 типа INT:

```
PROGRAM prog
VAR
  inst1 : fb;
END_VAR

LD 17
ST inst1.in1
CAL inst1
END_PROGRAM
```

Экземпляры функционального блока могут быть объявлены в другом функциональном блоке или в программе. Объявлять экземпляр функционального блока в теле функции нельзя. Экземпляры функционального блока доступны в том POU, в котором они объявлены, если они не объявлены глобально.

Экземпляры функциональных блоков могут быть использованы в качестве входных переменных других функциональных блоков или функций.

Замечания. После выполнения функционального блока все его переменные сохраняются до следующего выполнения. Следовательно, функциональный блок, вызываемый с одними и теми же входными параметрами, может производить различные выходные значения.

Если хотя бы одна переменная функционального блока объявлена как RETAIN, то все данные экземпляров целиком помещаются в энергонезависимый сегмент.

Вызов функционального блока

Для обращения к входным и выходным переменным функционального блока извне необходимо указать имя экземпляра функционального блока, следующей за ней точкой и именем переменной:

<Имя экземпляра>.<Имя переменной >

Присваивание параметров при вызове:

В текстовых языках (IL, ST) задать актуальные параметры и считать значения выходов можно непосредственно при вызове экземпляра функционального блока. Для входных переменных применяется присваивание ":=", выходы считываются при помощи "=>". Этот процесс упрощается, если использовать ассистент ввода (<F2>) с включенной опцией вставки с аргументами (With arguments).

Пример:

Допустим, FBINST - это локальная переменная типа функциональный блок, имеющий входную переменную xx и выходную переменную yy. При вставке FBINST в ST с помощью ассистента ввода получается следующая заготовка:

```
FBINST1(xx:= , yy=> );
```

Переменные вход-выход:

Обратите внимание, что переменные вход-выход (VAR_IN_OUT) передаются в экземпляр функционального блока через указатели. Поэтому таким переменным нельзя присваивать константы при вызове.

Пример:

```
VAR
  inst: fubo;
  var: int;
END_VAR
var1 := 2;
```

```
inst(instout1 := var1^);
```

не допустимые попытки присваивания констант:

```
inst(instout1:=2); или inst.inout1:=2;
```

Примеры вызова экземпляра вышеописанного функционального блока FUB:

Результат умножения сохраняется в переменной ERG, а результат сравнения в переменной QUAD. Экземпляр функционального блока FUB называется INSTANZ.

Вызов экземпляра функционального блока в IL:

```

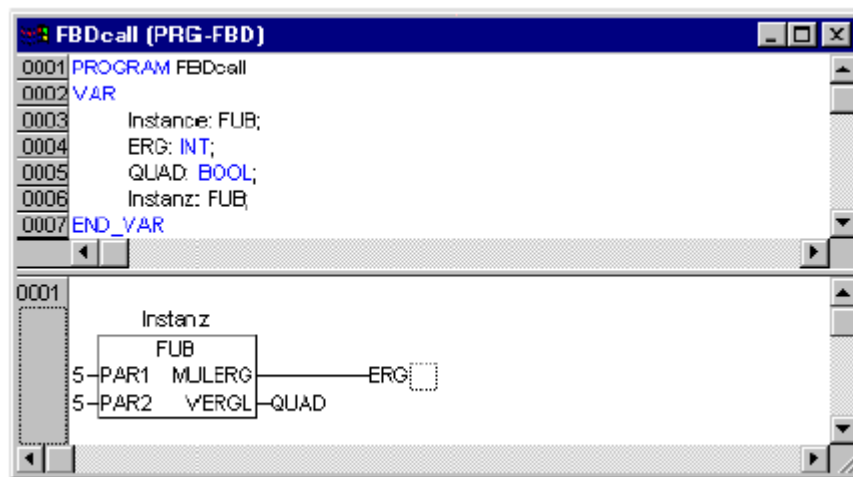
0001 PROGRAM AWLaufuf
0002 VAR
0003   QUAD:  BOOL;
0004   INSTANZ: FUB;
0005   ERG:   INT:=0;
0006 END_VAR
0007
0001  CAL  INSTANZ(PAR1:=5,PAR2:=5)
0002
0003  LD   INSTANZ.VERGL
0004  ST   QUAD
0005
0006  LD   INSTANZ.MULERG
0007  ST   ERG
0008
  
```

Тот же пример в ST (раздел объявлений такой же, как и в предыдущем примере):

```

0001 PROGRAM STcall
0002
0001 INSTANZ(PAR1=5,PAR2=5);
0002 QUAD:=INSTANZ.VERGL;
0003 ERG:=INSTANZ.MULERG;
  
```

Тот же пример в FBD:

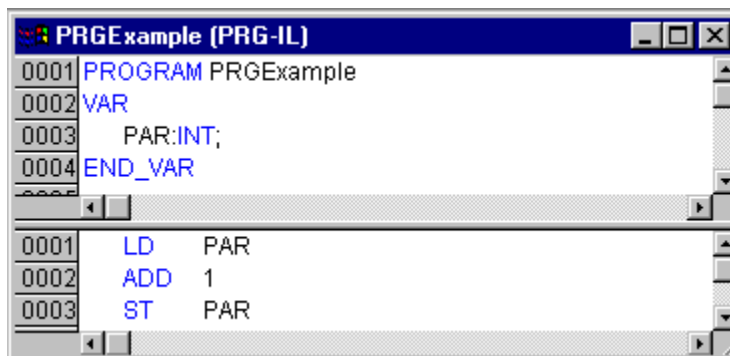


В SFC экземпляры функциональных блоков могут вызываться только из действий шага.

Программа

Программа – это POU, способный формировать произвольное значение во время вычислений. Значения всех переменных программы сохраняются между вызовами. В отличие от функционального блока экземпляров программы не существует. Программа является глобальной во всем проекте.

Пример программы:



Нельзя вызывать программу из функции.

Если вызвать программу, которая изменит значения своих переменных, то при следующем вызове ее переменные будут иметь те же значения, даже если она вызвана из другого POU.

В этом заключается главное различие между программой и функциональным блоком, в котором изменяются только значения переменных данного экземпляра функционального блока.

Объявление программы начинается ключевым словом PROGRAM и заканчивается ключевым словом END_PROGRAM.

Так же, как и для экземпляров функциональных блоков, в текстовых языках (IL, ST) задать актуальные параметры и считать значения выходов можно непосредственно при вызове программы. Для входных переменных применяется присваивание "=", выходы считываются при помощи "=>".

Пример вызова программы:

```
IL:
    CAL PRGexample2
```

```
LD   PRGexample2.out_var
ST   ERG
```

Или с присваиванием параметров:
 CAL PRGexample2(in_var:=33, out_var=>erg)

ST:

```
PRGexample2;
Erg := PRGexample2.out_var;
```

Или с присваиванием параметров:
 PRGexample2(in_var:=33, out_var=>erg);

FBD:



PLC_PRG

Программа PLC_PRG – это специальный POU, который должен быть в каждом проекте. Эта программа вызывается один раз за цикл управления.

При создании нового проекта автоматически открывается диалог **"Project" "Object Add"**, предлагающий создать новый POU - программу с именем PLC_PRG.

Не следует менять предложенные установки.

Если определить последовательность выполнения задач в Task Configuration, то проект может не содержать PLC_PRG.

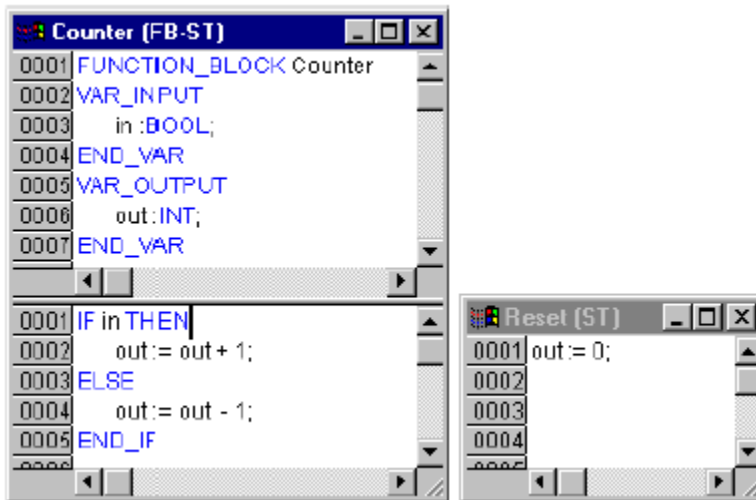
Внимание: Нельзя удалять или переименовывать POU PLC_PRG(если Task Configuration не используется). PLC_PRG является главной программой в однозадачном проекте.

Действие

Программы или функциональные блоки могут быть дополнены действиями. Фактически действия - это дополнительный набор встроенных в POU подпрограмм. Действия могут описываться на языке, отличном от того, на котором выполняется соответствующий функциональный блок или программа.

Действие оперирует с теми же данными, что и функциональный блок или программа, к которой оно принадлежит.

Пример действия функционального блока:



В данном примере функциональный блок Counter увеличивает или уменьшает выходную переменную "out" в зависимости от значения входа "in". При вызове действия Reset выходная переменная принимает значение 0. Одна и та же переменная "out" используется в обоих случаях.

Вызов действия:

Действие вызывается с помощью идентификатора:

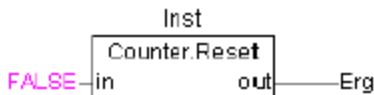
<Имя_программы>.<Имя_действия> или <Имя_экземпляра>.<Имя_действия >.

Если нужно вызвать действие из POU, к которому оно принадлежит, то в текстовых языках используется имя действия, а в графических – функциональный блок без указания имени экземпляра.

Пример вызова вышеописанного действия:

```

PROGRAM PLC_PRG
VAR
    Inst : Counter;
END_VAR
IL:
    CAL Inst.Reset(In := FALSE)
    LD Inst.out
    ST ERG
ST:
    Inst.Reset(In := FALSE);
    Erg := Inst.out;
FBD:
    
```



Замечание Действия играют ключевую роль в SFC (подробнее смотри раздел SFC).
 Стандарт МЭК определяет только действия SFC шагов. Применение действий в функциональных блоках и программах является расширением CoDeSys.

Ресурсы

Ресурсы отвечают за конфигурацию проекта, включая:

- § Глобальные переменные, используемые во всем проекте.
- § Менеджер библиотек (**Library manager**) для подключения необходимых библиотек к проекту
- § Журнал записи действий во время исполнения
- § Конфигуратор тревог (**Alarm Configuration**) для конфигурирования обработки тревог в проекте
- § Конфигуратор ПЛК (**PLC Configuration**) для конфигурирования аппаратуры контроллера.
- § Конфигуратор задач (**Task Configuration**) для управления задачами
- § Менеджер рецептов (**Watch and Receipt Manager**) для просмотра и заказа наборов значений переменных
- § Опции целевой системы (**Target Settings**)
- § Рабочая область для отображения опций проекта

В зависимости от системы исполнения и ее опций могут подключаться дополнительные объекты:

- § **Sampling Trace** - для задания графической трассировки значений переменных.
- § **Parameter Manager** - для взаимодействия с другими контроллерами в сети
- § **PLC-Browser** - монитор ПЛК
- § **Tools** – для вызова внешних, специфичных для каждой платформы инструментов
- § **SoftMotion** – компоненты системы управления движением (в соответствии с лицензией), редакторы CNC и CAM

Библиотеки

Проект может использовать несколько библиотек, в которые входят POU, необходимые им типы данных и глобальные переменные. Библиотечные POU можно использовать точно так же, как и определенные пользователем.

Библиотеки "standard.lib" и "util.lib" обязательно входят в стандартный комплект поставки.

(См. «Менеджер библиотек»)

Типы данных

Кроме стандартных типов данных, вы можете использовать определяемые пользователем типы данных. Ими могут быть структуры, перечисления и ссылки.

(См. «Приложение Типы данных»)

Визуализация

С помощью визуализации пользователь может создать графическое представление проекта. Форма и цвет графических элементов будут изменяться при работе программы в зависимости от значений переменных.

Визуализация может исполняться в системе программирования, в отдельном приложении CoDeSys HMI или как Web или целевая (в ПЛК) визуализация. (См. раздел «CoDeSys Визуализация»)

2.2 Языки программирования

CoDeSys поддерживает следующие **текстовые**:

- Instruction List (IL)
- Structured Text (ST)

и **графические** МЭК языки:

- Sequential Function Chart (SFC)

- Function Block Diagram (FBD)
- Ladder Diagram (LD)

Кроме того, CoDeSys включает поддержку, основанного на Функциональных Блоковых Диаграммах, редактора Continuous Function Chart (CFC).

2.2.1 Список инструкций (IL)

Язык IL (Instruction list) дословно – список инструкций. Каждая инструкция начинается с новой строки и содержит оператор и, в зависимости от типа операции, один и более операндов, разделенных запятыми.

Перед операндом может находиться метка, заканчивающаяся двоеточием (:). Комментарий должен быть последним элементом в строке. Между инструкциями могут находиться пустые строки.

Пример:

```
LD      17
ST      lint      (* комментарий*)
GE      5
JMPC    next
LD      idword
EQ      istruct.sdword
STN     test
next:
```

2.2.2 Модификаторы и операторы IL

В IL можно использовать следующие операторы и модификаторы:

Модификаторы:

- | | | |
|----------|---------------------|---|
| C | с JMP, CAL, RET: | инструкция выполняется только тогда, когда результат аккумулятора ИСТИНА. |
| N | с JMPC, CALC, RETC: | инструкция выполняется тогда, когда результат аккумулятора ЛОЖЬ. |
| N | в других случаях: | отрицание операнда. |

Ниже приведена таблица всех операторов IL с пояснениями и допустимыми модификаторами:

Оператор	Модификатор	Значение
LD	N	Присвоение аккумулятору значения оператора
ST	N	Присвоение значения аккумулятора операнду
S		Присвоить логическому операнду значение ИСТИНА, если значение аккумулятора ИСТИНА
R		Присвоить логическому операнду значение ЛОЖЬ
AND	N, (Побитное И
OR	N, (Побитное ИЛИ
XOR	N, (Побитное исключающее ИЛИ
ADD	(Сложение
SUB	(Вычитание
MUL	(Умножение

DTV	(Деление
GT	(>
GE	(>=
QE	(=
NE	(<>
LE	(<=
LT	(<
JMP	CN	Переход к метке
CAL	CN	Вызов функционального блока
RET	CN	Выход из POU и возврат в вызывающую программу.
)		Вычисление задержанной операции

Список всех операторов МЭК приведен в приложении.

Пример IL программы с использованием некоторых модификаторов:

```
LD TRUE      (*загрузить значение ИСТИНА в аккумулятор*)
AND  BOOL1  (*выполнить И с инверсным значением переменной BOOL1*)
JMP  mark   (*если значение аккумулятора ИСТИНА, то перейти к метке
            "mark"*)
LDN  BOOL2  (*сохранить инверсное значение BOOL2 в аккумуляторе*)
ST  ERG    (*сохранить значение аккумулятора в ERG*)
```

После оператора можно поставить скобки, тогда значение выражения внутри скобок рассматривается как операнд.

Например:

```
LD  2
MUL 2
ADD 3
ST  ERG
```

Здесь значение ERG равно 7. Если поставить скобки, то порядок вычислений изменится:

```
LD  2
MUL ( 2
ADD 3
)
ST  ERG
```

Теперь значение переменной ERG равно 10.

Операция MUL выполняется только тогда, когда программа доходит до ")". В качестве операнда MUL использует значение 5.

2.2.3 Структурированный текст (ST)

ST представляет собой набор инструкций высокого уровня, которые могут использоваться в условных операторах ("IF...THEN...ELSE") и в циклах (WHILE...DO).

Пример:

```
IF value < 7 THEN
```

```

        WHILE value < 8 DO
            value:=value+1;
        END_WHILE;
    END_IF;

```

Выражения

Выражение – это конструкция, возвращающая определенное значение после его вычисления.

Выражение состоит из операторов и операндов. Операндом может быть константа, переменная, функциональный блок или другое выражение.

Вычисление выражений

Вычисление выражений выполняется согласно *правилам приоритета*. Оператор с самым высоким приоритетом выполняется первым, оператор с более низким приоритетом – вторым и т.д., пока не будут выполнены все операторы.

Операторы с одинаковым приоритетом выполняются слева направо.

В следующей таблице приведен список ST операторов, расположенных в порядке приоритета.

Операция	Обозначение	Приоритет
Выражение в скобках	(Выражение)	Самый высокий.
Вызов функции	Имя функции (список параметров)	
Возведение в степень	EXPT	
Замена знаков	-	
Числовое дополнение	NOT	
Умножение	*	
Деление	/	
Абсолютная величина	MOD	
Сложение	+	
Вычитание	-	
Сравнение	< , > , <= , >=	
Неравенство	< >	
Равенство	=	
Логическое И	AND	
Логическое исключающее ИЛИ.	XOR	
Логическое ИЛИ	OR	Самый низкий

Ниже приведены примеры использования инструкций ST:

Тип инструкции	Пример
Присваивание	A: =B; CV: =CV+1; C: =SIN (X);
Вызов функционального блока и	A: = CMD_TMR.Q

использование FB выхода

RETURN	RETURN;
IF	D: =B*B; IF D<0.0 THEN C: =A; ELSIF D=0.0 THEN C: =B; ELSE C: =D; END_IF;
CASE	CASE INT1 OF 1: BOOL1: = TRUE; 2: BOOL2: = TRUE; ELSE BOOL1: = FALSE; BOOL2 END_CASE;
FOR	J: =101; FOR I: =1 TO 100 BY 2 DO IF ARR [I] = 70 THEN J: =I; EXIT; END_IF; END_FOR;
WHILE	J: =1; WHILE J<= 100 AND ARR [J] <> 70 DO J: =J+2; END_WHILE;
REPEAT	J: =-1; REPEAT J: = J+2; UNTIL J= 101 OR ARR [J] = 70 END_REPEAT;
EXIT	EXIT;
Пустая инструкция	;

Оператор присваивания

Перед оператором присваивания находится операнд (переменная или адрес), которому присваивается значение выражения, стоящего после оператора присваивания.

Пример:

Var1: = Var2 * 10;

После выполнения этой операции Var1 принимает значение в десять раз большее, чем Var2.

Вызов функционального блока в ST

Функциональный блок вызывается с помощью имени экземпляра функционального блока и списка входных параметров с присваиванием данных в круглых скобках. В следующем примере вызывается таймер с параметрами IN и PT. Значение выходной переменной Q присваивается переменной A.

К выходной переменной, как и в PL, можно обратиться с помощью имени экземпляра функционального блока, точки, следующей за ним и имени выходной переменной:

```
CMD_TMR (IN: = %IX5, PT: = 300);
A: =CMD_TMR.Q
```

Инструкция RETURN

Инструкция RETURN позволяет выйти из POU, например, в зависимости от условия.

Инструкция IF

Используя инструкцию IF, можно проверить условие, и в зависимости от этого условия выполнить какие-либо действия.

Синтаксис:

```
IF <Boolean_expression1> THEN
    <IF_instructions>
{ELSIF <Boolean_expression2> THEN
    <ELSIF_instructions1>
.
.
.ELSIF <Boolean_expression n> THEN
    <ELSIF_instructions n-1>
ELSE
    <ELSE_instructions>}
END_IF;
```

Часть конструкции фигурных скобках не обязательна.

Если <<Boolean_expression1> возвращает истину, тогда <IF_Instructions> выполняется.

В противном случае будут выполняться остальные логические выражения одно за другим, пока одно из них не возвратит истину. Тогда выполняются инструкции, стоящие после этого логического выражения до следующего ELSIF или ELSE.

Если все логические выражения ложны, то выполняются инструкции, стоящие после ELSE.

Пример:

```
IF temp < 17
THEN    heating_on: = TRUE;
ELSE    heating_on: = FALSE;
END_IF
```

В этом примере нагревание (heating) включается, когда температура опустится ниже 17 ° градусов, иначе оно останется выключенным.

Инструкция CASE

С помощью инструкции CASE можно нескольким различным значениям целочисленной переменной сопоставить различные инструкции.

Синтаксис:

```

CASE <Var1> OF
  <Value1>:  <Instruction 1>
  <Value2>:  <Instruction 2>
  <Value3, Value4, Value5>:  <Instruction 3>
  <Value6 .. Value10>:      <Instruction 4>
  ...
  <Value n>:  <Instruction n>
ELSE      <ELSE instruction>
END_CASE;

```

Инструкция CASE выполняется согласно следующим правилам:

- Если переменная <Var1> имеет значение <Value i>, то выполняется инструкция <Instruction i>
- Если <Var1> не принимает ни одного из указанных значений, то выполняется <ELSE Instruction>.
- Чтобы одна и та же инструкция выполнялась при различных значениях переменной <Var1>, необходимо перечислить эти значения через запятую.
- Чтобы одна и та же инструкция выполнялась для целого диапазона значений, необходимо указать начальное и конечное значения, разделенные двумя точками.

Пример:

```

CASE INT1 OF
  1, 5: BOOL1 := TRUE;
        BOOL3 := FALSE;
  2:  BOOL2 := FALSE;
        BOOL3 := TRUE;
  10. 20: BOOL1 := TRUE;
        BOOL3 := TRUE;
ELSE
  BOOL1 := NOT BOOL1;
  BOOL2 := BOOL1 OR BOOL2;
END_CASE;

```

Цикл FOR

С помощью FOR можно программировать повторяющиеся процессы.

Синтаксис:

```

INT_Var :INT;

FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO
  <Instructions>
END_FOR

```

Часть конструкции, заключенная в фигурные скобки, не обязательна.

<Instructions> выполняются, пока счетчик <INT_Var> не больше <END_VALUE>. Это условие проверяется перед выполнением <Instructions>, поэтому раздел <Instructions> не выполняется, если <INIT_VALUE> больше <END_VALUE>.

Всякий раз, когда выполняются <Instructions>, значение <INIT_VALUE>, увеличивается на <Step_size>.

<Step_size> может принимать любое целое значение. По умолчанию шаг устанавливается равным 1.

Пример:

```
FOR Counter: =1 TO 5 BY 1 DO
  Var1: =Var1*2;
END_FOR;
Erg:=Var1;
```

В этом примере предполагается, что начальное значение Var1 равно 1. После выполнения цикла эта переменная будет равна 32.

Замечание: <END_VALUE>: не должно быть равно предельному значению счетчика <INT_VAR>. Например, если счетчик является переменной типа SINT и <END_VALUE> равно 127, то цикл становится бесконечным.

Цикл WHILE

Цикл WHILE может использоваться, как и цикл FOR, с тем лишь различием, что условие выхода определяется логическим выражением. Это означает, цикл выполняется, пока верно заданное условие.

Синтаксис:

```
WHILE <Boolean expression>
  <Instructions>
END_WHILE
```

Раздел <Instructions> выполняется циклически до тех пор, пока <Boolean_expression> дает TRUE. Если <Boolean_expression> равно FALSE уже при первой итерации, то раздел <Instructions> не будет выполнен ни разу. Если <Boolean_expression> никогда не примет значение FALSE, то раздел <Instructions> будет выполняться бесконечно.

Замечание: Программист должен быть уверен, что цикл не станет бесконечным. Для этого в теле цикла значение входящей в условие переменной обязательно должно изменяться. Например, путем инкремента или декремента счетчика.

Пример:

```
WHILE counter<>0 DO
  Var1: = Var1*2;
  counter := counter-1;
END_WHILE
```

Цикл REPEAT

Цикл REPEAT отличается от цикла WHILE тем, что первая проверка условия выхода из цикла осуществляется, когда цикл уже выполнен 1 раз. Это означает, что независимо от условия выхода цикл выполняется хотя бы один раз.

Синтаксис:

```
REPEAT
  <Instructions>
```

UNTIL <Boolean expression>
END_REPEAT

Раздел <Instructions> выполняется циклически до тех пор, пока <Boolean_expression> дает TRUE. Если <Boolean_expression> равно FALSE уже при первой итерации, то раздел <Instructions> не будет выполнен один раз. Если <Boolean_expression> никогда не примет значение FALSE, то раздел <Instructions> будет выполняться бесконечно.

Замечание: Программист должен быть уверен, что цикл не станет бесконечным. Для этого в теле цикла значение входящей в условие переменной обязательно должно изменяться. Например, путем инкремента или декремента счетчика.

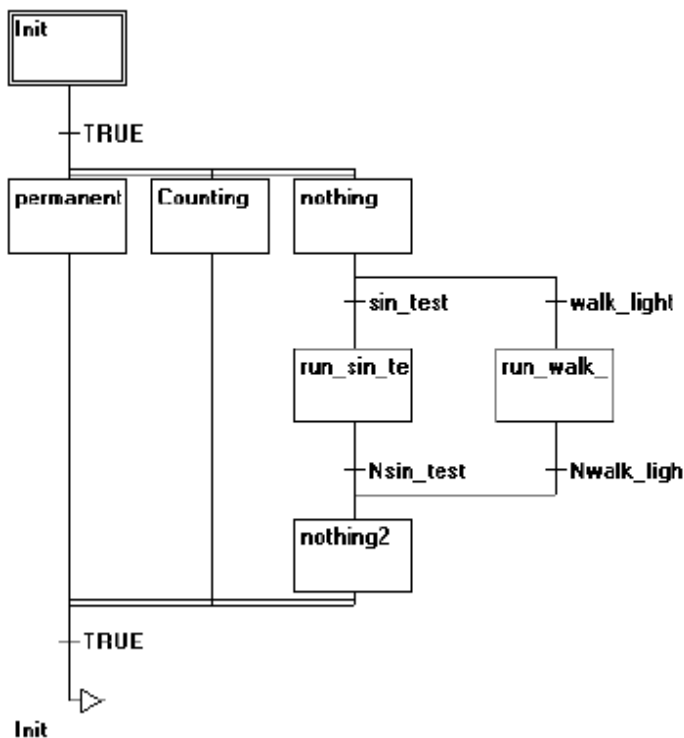
Инструкция EXIT

Если EXIT встречается в циклах FOR, WHILE, REPEAT, то цикл заканчивает свою работу независимо от значения условия выхода.

2.2.4 Язык последовательных функциональных схем (SFC)

SFC – это графический язык, который позволяет описать хронологическую последовательность различных действий в программе. Для этого действия связываются с шагами (этапами), а последовательность работы определяется условиями переходов между шагами.

Пример SFC диаграммы:



Шаг

SFC POU состоит из набора шагов, связанных переходами. Существуют 2 вида шагов:

- Шаг простого типа (упрощенный SFC) может включать единственное действие. Графический флажок (небольшой треугольник в верхнем углу шага) показывает, пустой шаг или нет.
- МЭК шаг (стандартный SFC) связан с произвольным числом действий или логических переменных. Связанные действия располагаются с правой стороны от шага.

Действие

Действие может содержать список инструкций на IL или ST, схемы на FBD или LD, или снова схемы на SFC.

При использовании простых шагов действие всегда связывается с этим шагом. Для того, чтобы редактировать действие, необходимо дважды щелкнуть левой клавишей мышки на шаге. Или выделить шаг и выбрать команду меню "Extras" "ZoomAction/Transition". Помимо основного действия, шаг может включать одно входное и одно выходное действие.

Действия МЭК шагов показаны в Организаторе Объектов, непосредственно под вызывающей их POU. Редактирование действия запускается двойным щелчком мыши или клавишей <Enter>. Новые действия

добавляются командой главного меню "Project" "Add Action". Вы можете сопоставить одному шагу до 9 действий.

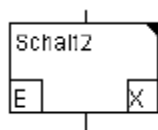
Входное или выходное действие

В шаг можно добавить входное и выходное действие.

Входное действие выполняется один раз при активизации шага, выходное – при деактивизации. Шаг, который имеет входное действие, обозначается буквой "E" в левом нижнем углу, шаг с выходными действиями – буквой "X" в правом нижнем углу.

Входные и выходные действия могут описываться на любом языке. Для того чтобы отредактировать входное или выходное действие, надо дважды щелкнуть мышкой в соответствующем углу шага.

Пример простого шага с входным и выходным действиями:



Переход/условие перехода

Между шагами находятся так называемые переходы. Условием перехода может быть логическая переменная или константа, логический адрес или логическое выражение, описанное на любом языке. Условие может включать серию инструкций, образующих логический результат, в виде ST выражения (т.е. $(i \leq 100) \text{ AND } b$) либо на любом другом языке. Но условие не должно содержать присваивания, вызов программ и экземпляров функциональных блоков!

В редакторе SFC условие перехода можно записать непосредственно около символа перехода либо в отдельном окне редактора для ввода условия (См. раздел 5.4.4, 'Extras' 'Zoom Action/Transition'). Условие заданное окне редактора предпочтительнее!

Замечание: помимо условий переходов, можно использовать тактируемый режим переходов; См. SFCtip и SFCtipmode.

Активный шаг

После вызова SFC POU начальный шаг (шаг, выделенный двойной рамкой) выполняется первым. Шаг, выполняемый в данный момент, называется активным. Действия, связанные с активным шагом, выполняются один раз в каждом управляющем цикле. В режиме online активные шаги выделяются синим цветом. Следующий за активным шагом шаг станет активным, только когда условие перехода к этому шагу будет истинно.

В каждом управляющем цикле будут выполнены действия, содержащиеся в активных шагах. Далее проверяются условия перехода, и, возможно, уже другие шаги становятся активными, но выполняться они будут уже в следующем цикле.

Замечание: выходное действие выполняется однократно в следующем цикле, после того, как условие перехода станет истинным.

Шаг МЭК

В отличие от упрощенного SFC МЭК шаги могут включать несколько действий. Действия МЭК шагов описываются отдельно от них и могут неоднократно использоваться в пределах данного POU, для чего их надо связать с шагом с помощью команды главного меню "Extras" "Associate action".

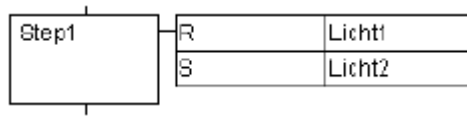
Кроме действий с шагом, можно связывать логические переменные.

С помощью так называемых классификаторов действия и логические переменные могут активироваться и деактивироваться, возможно, с задержкой времени. Например: действие может продолжать работу, даже если запустивший его шаг утратил активность; с помощью классификатора S (установка) можно программировать параллельные процессы и т.д.

Логическая переменная, связанная с шагом, получает значение ИСТИНА при каждой активации шага.

Действие, связанное с МЭК шагом, описывается справа от него в блоке, состоящем из двух частот. Левая часть этого блока содержит классификатор, возможно, с константой времени, а правая часть содержит имя действия или логической переменной.

Пример МЭК шага с двумя действиями:



В режиме online все активные действия выделяются синим цветом, подобно активным шагам. Благодаря чему достаточно легко проследить ход выполнения процесса после каждого управляющего цикла.

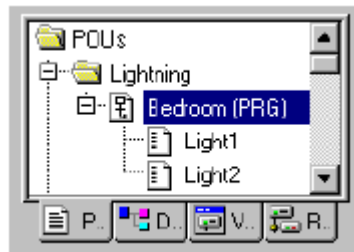
Замечание: Если действие деактивируется, то оно выполняется **еще один раз**. Это означает, что каждое действие выполняется хотя бы два раза (см. далее действие со спецификаторами).

При выполнении шага сначала производится деактивация действий, затем выполняются активные действия в алфавитном порядке.

Для того чтобы использовать шаги с МЭК действиями, необходимо установить опцию "Extras" "Use IEC-Steps" и подключить к проекту специальную библиотеку **Iecsf.lib**.

В Организаторе объектов действия показаны непосредственно под SFC POU's, которые их вызывают. Новые действия можно создавать с помощью команды "Project" "Add Action".

SFC POU со списком действий в Организаторе Объектов:



Классификаторы действий

Для связи шагов и действий применяются классификаторы:

N	Не сохраняемое	Действие активно в течение активности шага
R	Внеочередной сброс	Деактивация действия
S	Установка	Действие активно вплоть до сброса
L	Ограниченное по времени	Действие активно в течение указанного времени, но не дольше времени активности шага
D	Отложенное	Действие активируется по прошествии указанного времени, если шаг еще активен и продолжает быть активным

		ни, если шаг еще активен и продолжает быть активным
P	Импульс	Действие выполняется один раз, если шаг активен
SD	Сохраняемое и отложенное	Действие активно после указанного времени до сброса
DS	Отложенное и сохраняемое	Действие активно после указанного времени, если шаг еще активен, вплоть до сброса
SL	Сохраняемое и ограниченное по времени	Активно после указанного времени.

Классификаторы L, D, SD, DS, SL требуют указания временной константы в формате TIME.

Замечание: В процессе деактивации действие выполняется еще один раз. Это относится и к действиям с классификатором P!

Неявные переменные в SFC

В SFC существуют неявно объявленные переменные, которые могут оказаться полезны.

Каждому шагу соответствует флаг, который хранит информацию о его состоянии. Этот флаг обозначается **<StepName>.x** для МЭК шагов и **<StepName>** для простых шагов. Этот флаг имеет значение ИСТИНА, когда соответствующий шаг активен, и ЛОЖЬ, когда неактивен. Он может использоваться в любом действии или переходе SFC.

Аналогично, с помощью переменной **<ActionName>.x**, можно узнать, активно действие или нет. С помощью неявной переменной **<StepName>.t** можно узнать время активности МЭК шага.

К неявным переменным разрешен доступ даже из другой программы. Например: `boolvar1:=sfc1.step1.x`; Где `step1.x` - неявная логическая переменная, представляющая состояние МЭК шага `step1` в POU `sfc1`.

Флаги SFC

Для управления работой SFC компонента предусмотрены специальные флаги. Для использования этих флагов необходимо объявлять их глобально или локально как входные или выходные переменные.

Пример: Если в SFC POU некоторый шаг активен дольше, чем время, заданное в его атрибутах (см. ниже), устанавливается специальный флаг, доступный через переменную "SFCError" (SFCError принимает значение TRUE в этом случае).

Вы можете использовать следующие переменные-флаги:

SFCEnableLimit: Переменная типа BOOL. Когда значение этой переменной ИСТИНА, задержка времени шага регистрируется в SFCError. Другие задержки времени игнорируются.

SFCInit: Переменная типа BOOL. Когда переменная получает значение ИСТИНА, программа переходит обратно на шаг Init и все SFC флаги сбрасываются. Шаг Init становится активным, но не выполняется, пока переменная имеет значение ИСТИНА. Как только SFCInit примет значение ЛОЖЬ, выполнение программы продолжится.

SFCReset: Переменная типа BOOL. Работает подобно SFCInit. Но последующее выполнение будет продолжено после шага инициализации Init. Поэтому флаг SFCReset можно сбросить в FALSE в самом шаге Init.

SFCQuitError: Переменная типа BOOL. Выполнение программы SFC приостанавливается, пока переменная имеет значение ИСТИНА. Посредством чего сбрасывается возможный таймаут. Время работы любого активного шага сбрасывается, когда эта переменная снова получает значение ЛОЖЬ.

SFCPause: Переменная типа BOOL. Выполнение программы SFC приостанавливается, пока эта переменная имеет значение ИСТИНА.

SFCError: Эта логическая переменная принимает значение ИСТИНА, когда происходит задержка времени. Если следом возникает вторая ошибка, она не фиксируется, пока флаг SFCError не будет сброшен. Для уточнения причины ошибки необходимо использовать флаги: SFCErrorStep, SFCErrorPOU, SFCQuitError, SFCErrorAnalyzation.

SFCTrans: Переменная типа BOOL. Принимает значение ИСТИНА, когда переход активируется.

SFCErrorStep: Переменная типа STRING. В этой переменной хранится имя шага, в котором обнаружена ошибка (задержка времени).

SFCErrorPOU: Переменная типа STRING. В этой переменной хранится имя компонента, в котором обнаружена ошибка (задержка времени).

SFCCurrentStep: Переменная типа STRING. В этой переменной хранится имя активного шага. В случае одновременного выполнения шагов в переменной сохраняется имя того шага который находится в правой ветви SFC диаграммы.

SFCErrorAnalyzationTable: Переменная типа **ARRAY [0..n] OF ExpressionResult** сообщает результаты анализа условного выражения перехода. Для каждого элемента выражения, формирующего значение FALSE и соответственно задерживающего переход, заполняется структура, содержащая наименование, адрес, комментарий и текущее значение.

Допускается максимум до 16 элементов (переменных), поэтому индексы массива имеют значения от 0 до 15.

Структура ExpressionResult и неявно используемые функции анализа включены в библиотеку AnalyzationNew.lib. Модули библиотеки можно использовать явно в других POU, не запрограммированных в SFC.

Предварительным условием анализа является обнаружение задержки в шаге. Поэтому контроль времени выполнения обязателен при анализе. Также обязательно должна быть объявлена переменная SFCError (см. выше).

SFCTip, SFCTipMode: Переменные типа BOOL позволяют задать тактируемый режим выполнения SFC. Если этот режим включен – SFCTipMode=TRUE, то переход к следующему шагу возможен только при SFCTip, равном TRUE. Пока SFCTip имеет значение FALSE, переход не разрешен, даже если условие выполнено.

Альтернативная ветвь

Две и более ветви SFC могут быть альтернативными. Каждая альтернативная ветвь должна начинаться и заканчиваться переходом. Альтернативные ветви могут содержать параллельные ветви и другие альтернативные ветви. Альтернативная ветвь начинается горизонтальной линией (начало альтернативы), а заканчивается горизонтальной линией (конец альтернативы) или переходом на произвольный шаг (jump). Если шаг, который находится перед линией альтернативного начала, активен, то первые переходы альтернативных ветвей начинают оцениваться слева направо.

Таким образом, первым активируется тот шаг, который следует за первым слева истинным переходом.

Параллельные ветви

Две и более ветви SFC могут быть параллельными.

Каждая параллельная ветвь должна начинаться и заканчиваться шагом. Параллельные ветви могут содержать альтернативные ветви и другие параллельные ветви. Параллельная ветвь наносится двойной горизонтальной линией (параллельное начало) и заканчивается двойной горизонтальной линией (конец параллели) или переходом на произвольный шаг (jump).

Если шаг активен, условие перехода после этого шага истинно и за этим переходом следуют параллельные ветви, то активируются первые шаги этих ветвей. Эти ветви выполняются параллельно друг другу. Шаг, находящийся после параллельных ветвей, становится активным только тогда, когда все предыдущие шаги активны и условие перехода истинно.

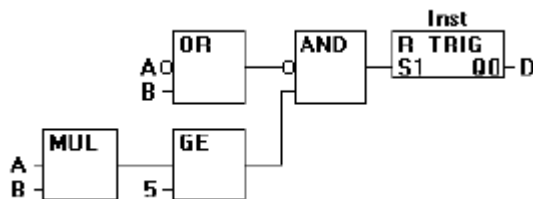
Переход на произвольный шаг (Jump)

Переход на произвольный шаг - это соединение на шаг, имя которого указано под знаком «jump». Такие переходы нужны для того, чтобы избежать пересекающихся и идущих вверх соединений.

2.2.5 Язык функциональных блокковых диаграмм (FBD)

FBD – это графический язык программирования. Он работает с последовательностью цепей, каждая из которых содержит логическое или арифметическое выражение, вызов функционального блока, переход или инструкцию возврата.

Вот пример типичной FBD схемы в CoDeSys:



Более подробно о FBD написано в разделе 5.4.2.

2.2.6 Непрерывные функциональные схемы (CFC)

В отличие от FBD редактор непрерывных функциональных схем не использует цепи, но дает возможность свободно размещать компоненты и соединения, что позволяет создавать обратные связи, как показано в примере.

Пример CFC:



Более подробно о CFC написано в разделе 5.4.5.

2.2.7 Язык релейных диаграмм (LD)

Язык релейных или релейно-контактных схем (РКС) – графический язык, реализующий структуры электрических цепей.

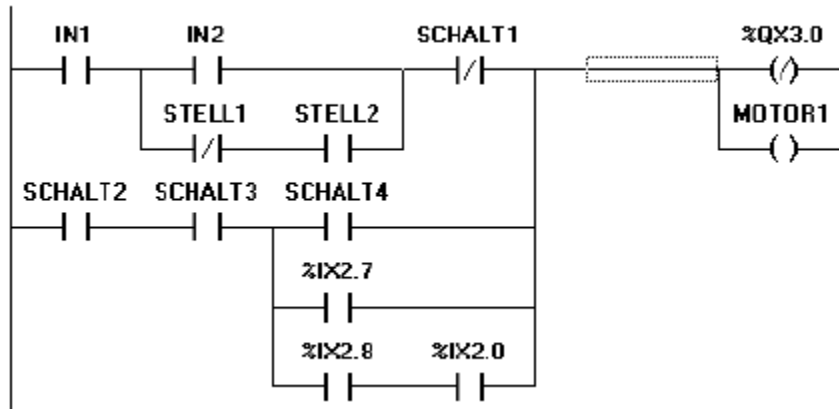
Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать и сложные цепи - как в FBD. Кроме того, LD достаточно удобен для управления другими компонентами POU.

Диаграмма LD состоит из ряда цепей.

Слева и справа схема ограничена вертикальными линиями - шинами питания. Между ними расположены цепи, образованные контактами и обмотками реле, по аналогии с обычными электронными цепями.

Слева любая цепь начинается набором контактов, которые посылают слева направо состояние "ON" или "OFF", соответствующие логическим значениям ИСТИНА или ЛОЖЬ. Каждому контакту соответствует логическая переменная. Если переменная имеет значение ИСТИНА, то состояние передается через контакт. Иначе правое соединение получает значение выключено ("OFF").

Пример типичной LD цепи:



(См. также раздел "Редакторы CoDeSys")

Контакт

Контакты обозначаются двумя параллельными линиями и могут иметь состояния "ON" или "OFF". Эти состояния соответствуют значениям ИСТИНА или ЛОЖЬ. Каждому контакту соответствует логическая переменная. Если значение переменной ИСТИНА, то контакт замкнут.

Контакты могут быть соединены параллельно, тогда соединение передает состояние "ON", когда хотя бы одна из ветвей передает "ON".

Если контакты соединены последовательно, то для того, чтобы соединение передало "ON", необходимо, чтобы оба контакта передавали "ON". Это соответствует электрической параллельной и последовательной схеме.

Контакт может быть инвертируемым. Такой контакт обозначается с помощью символа $\overline{\quad}$ и передает состояние "ON", если значение переменной ЛОЖЬ.

Обмотка

В правой части схемы может находиться любое количество обмоток (реле), которые обозначаются круглыми скобками (). Они могут соединяться только параллельно. Обмотка передает значение соединения слева направо и копирует его в соответствующую логическую переменную.

В целом цепь может быть либо замкнутой (ON), либо разомкнутой (OFF). Это как раз и отражается на обмотке и соответственно на логической переменной обмотки (ИСТИНА/ЛОЖЬ).

Обмотки также могут быть инверсными (в примере - %QX3.0). Если обмотка инверсная (обозначается символом (/)), тогда в соответствующую логическую переменную копируется инверсное значение.

Функциональные блоки в LD

Кроме контактов и обмоток, в LD можно использовать функциональные блоки и программы. Они должны иметь логические вход и выход и могут использоваться так же, как контакты.

SET и RESET обмотка

Обмотки могут быть с «самофиксацией» типов SET и RESET. Обмотки типа SET обозначаются буквой "S" внутри круглых скобок (S). Если соответствующая этой обмотке переменная принимает значение ИСТИНА, то она навсегда (до сброса R) сохраняет его.

Обмотки типа RESET обозначаются буквой R. Если соответствующая переменная принимает значение ЛОЖЬ, то она навсегда (до установки S) сохраняет его.

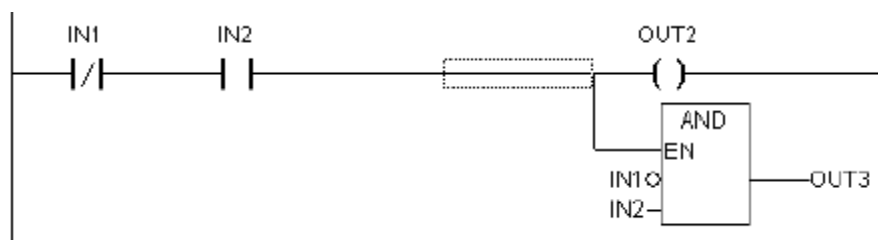
LD в качестве FBD

Весьма вероятно, что при работе с LD вы захотите с помощью контакта управлять другими POU.

Во-первых, можно использовать обмотку для передачи значения глобальной переменной, которая будет использоваться в другом месте. Кроме того, можно вставить вызов прямо в схему LD.

Такой POU может быть оператором, функцией, программой или функциональным блоком, который имеет добавочный вход, обозначаемый EN. Вход EN всегда логического типа, и POU выполняется, только когда значение EN=ИСТИНА. POU встраивается в схему параллельно обмоткам, и вход EN соединяется ответвлением. Использование таких POU делает LD схему похожей на FBD схему.

Пример LD цепи с EN POU:



2.3 Отладка и online функции

Трассировка.

Sampling Trace позволяет отображать последовательные значения переменных, записанные периодически, с возможной привязкой к событию, заданному так называемым триггером трассировки. Sampling

Trace обладает способностью трассировать до 20 переменных одновременно. Для каждой переменной сохраняются 500 последних значений.

Отладка

Специальная опция отладки CoDeSys заставляет компилятор формировать дополнительный код, упрощающий поиск ошибок. Опция включается через меню "**Project**" "**Options**" в диалоговом окне "**Debugging**".

Точки останова

Точки останова – это места, в которых выполнение программы будет приостанавливаться, что позволяет просмотреть значения переменных на определенном этапе работы программы. Точки останова можно задавать во всех редакторах. В текстовом редакторе точка останова устанавливается на номер строки, в FBD и LD - на графический элемент и в SFC - на шаг.

Пошаговое выполнение

Под «шагом» подразумевается:

- В IL: Выполнить программу до следующего оператора CAL, LD or JMP.
- В ST: Выполнить следующую инструкцию.
- В FBD, LD: Выполнить следующую цепь.
- В SFC: Продолжить действие до следующего шага.

Пошаговое выполнение позволяет проверить логическую правильность программы.

Выполнение по циклам

Команда одинарный цикл (Single Cycle) выполняет один рабочий цикл и останавливает контроллер после выполнения.

Изменения значений переменных в режиме Online.

Изменив значение переменной в режиме online, вы можете однократно записать его в контроллер (запись значений) либо включить режим записи заданных значений в каждом рабочем цикле (фиксация значений).

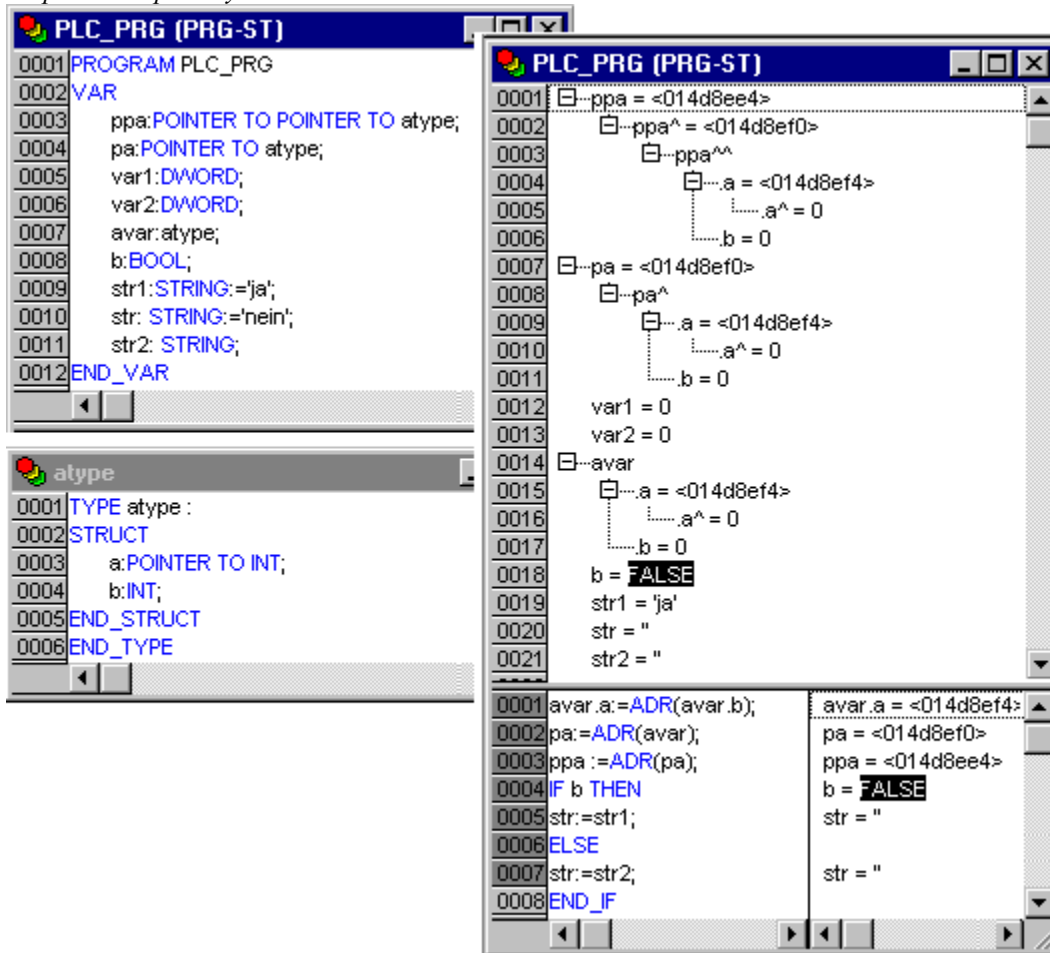
Для изменения текущего значения переменной щелкните по нему дважды мышкой. Если это логическая переменная, то она изменит свое значение на противоположное, если же эта переменная любого другого типа, то новое значение переменной нужно будет ввести в диалоговом окне (**Write Variable**).

Мониторинг

В режиме online значения всех видимых на экране переменных отображаются CoDeSys в реальном времени. Вы можете наблюдать актуальные значения переменных в разделах объявлений и кода редакторов, в менеджере «рецептов» (watch and receipt manager) и на экранах визуализации. Значения переменных экземпляров функциональных блоков отображаются в виде иерархического дерева.

Для указателей, в разделе кода, отображаются только их собственные значения. В разделе объявлений вы можете «раскрыть» указатель и увидеть значение адресуемой им переменной.

Пример мониторинга указателей:



При мониторинге массивов отображаются не только элементы с константой в индексе, но и элементы с индексной переменной:

```

anarray[1] = 5
anarray[i] = 1
    
```

Значение элемента массива не отображается только в случае, если в качестве индекса задано выражение (например [i + j] или [I*2]).

Внимание: Если превышено число одновременно отображаемых переменных, то в строке значений переменных вы увидите сообщение: "Too many monitoring variables".

Эмуляция

Во время эмуляции созданная программа выполняется не в ПЛК, а в компьютере, на котором запущен CoDeSys. В этом режиме допустимы все функции online, что позволяет проверить логическую правильность программ, не используя контроллер.

Внимание: функции внешних библиотек не выполняются в режиме эмуляции.

Бортжурнал

Бортжурнал (log) хронологически записывает действия пользователя, внутренние процессы, изменения состояния и исключения в режиме online. Это необходимо при анализе условий возникновения ошибки при отладке.

3 Пишем простой пример

3.1 Блок управления светофором

В этой главе мы создадим небольшую программу-пример. Это - простой блок управления движением на перекрестке, имеющем светофоры для двух пересекающихся направлений движения. Понятно, что светофоры должны иметь два противоположных состояния – красный и зеленый. Чтобы избежать несчастных случаев, мы добавим общепринятые переходные стадии: желтый и желто-красный. Последняя стадия должна быть длиннее предыдущей.

В этом примере мы покажем, как управляемые по времени процессы можно представить средствами языков стандарта МЭК 61131-3, как легко можно комбинировать различные языки в CoDeSys и познакомимся со средствами моделирования в CoDeSys.

Создайте POU

Начинать всегда легко: запустите CoDeSys и выберите "File" "New".

В окне диалога определим первый POU. По умолчанию он получает наименование PLC_PRG. Не изменяйте его. Тип этого POU, безусловно, должен быть - программа. Каждый проект должен иметь программу с таким именем. В качестве языка программирования данного POU мы выберем язык Continuous Function Chart (CFC).

Теперь создайте еще три объекта. Воспользуйтесь командой "Project" "Object Add" в системном или в контекстном (нажмите правую кнопку мыши в окне Object Organizer) меню. Создайте: программу на языке Sequential Function Chart (SFC) с именем SEQUENCE, функциональный блок на языке Function Block Diagram (FBD) с именем TRAFFICSIGNAL и еще один аналогичный блок - WAIT, который мы будем описывать на языке Список Инструкции (IL).

Что делает TRAFFICSIGNAL?

В POU TRAFFICSIGNAL мы сопоставим определенные стадии процесса соответствующим цветам. То есть мы удостоверимся, что красный свет зажжен в красной стадии и в желто-красной стадии, желтый свет в желтой и желто-красной стадии и т.д.

Что делает WAIT?

В WAIT мы создадим простой таймер, который на вход получает длину стадии в миллисекундах и на выходе выдает состояние ИСТИНА по истечении заданного периода времени.

Что делает SEQUENCE?

В SEQUENCE все будет объединено так, чтобы нужные огни зажигались в правильное время и на нужный период времени.

Что делает PLC_PRG?

В PLC_PRG вводится входной сигнал включения, разрешающий начало работы светофора и 'цветовые команды' каждой лампы связаны с соответствующими выходами аппаратуры.

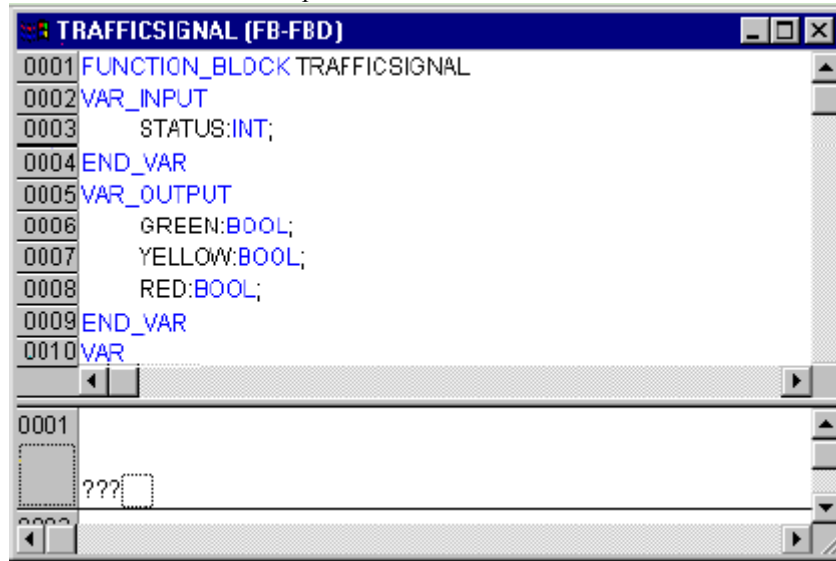
Объявления "TRAFFICSIGNAL"

Вернемся теперь к POU TRAFFICSIGNAL. В редакторе объявлений определите входную переменную (между ключевыми словами VAR_INPUT и END_VAR) по имени STATUS типа INT. STATUS будет

иметь четыре возможных состояния, определяющие соответствующие стадии - зеленая, желтая, желто-красная и красная.

Поскольку наш блок TRAFFICSIGNAL имеет три выхода, нужно определить еще три переменных RED, YELLOW и GREEN. Теперь раздел объявлений блока TRAFFICSIGNAL должен выглядеть так:

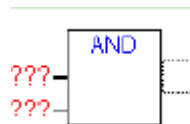
Функциональный блок TRAFFICSIGNAL, раздел объявлений:



Программируем "TRAFFICSIGNAL"

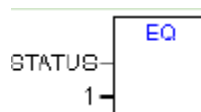
Теперь пора описать, что связывает вход STATUS с выходными переменными. Для этого перейдите в раздел кода POU (body). Щелкните на поле слева от первой цепи (серая область с номером 1). Вы теперь выбрали первую цепь. Теперь дайте команду меню "Insert" "Operator".

В первой цепи будет вставлен прямоугольник с оператором AND и двумя входами:



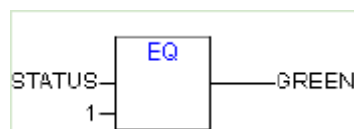
Щелкните мышкой на тексте AND и замените его на EQ.

Три знака вопроса около верхнего из двух входов замените на имя переменной STATUS. Для нижнего входа вместо трех знаков вопроса нужно поставить 1. В результате Вы должны получить следующий элемент:



Щелкните теперь на месте позади прямоугольника EQ. Теперь выбран выход EQ. Выполните команду меню "Insert" "Assignment".

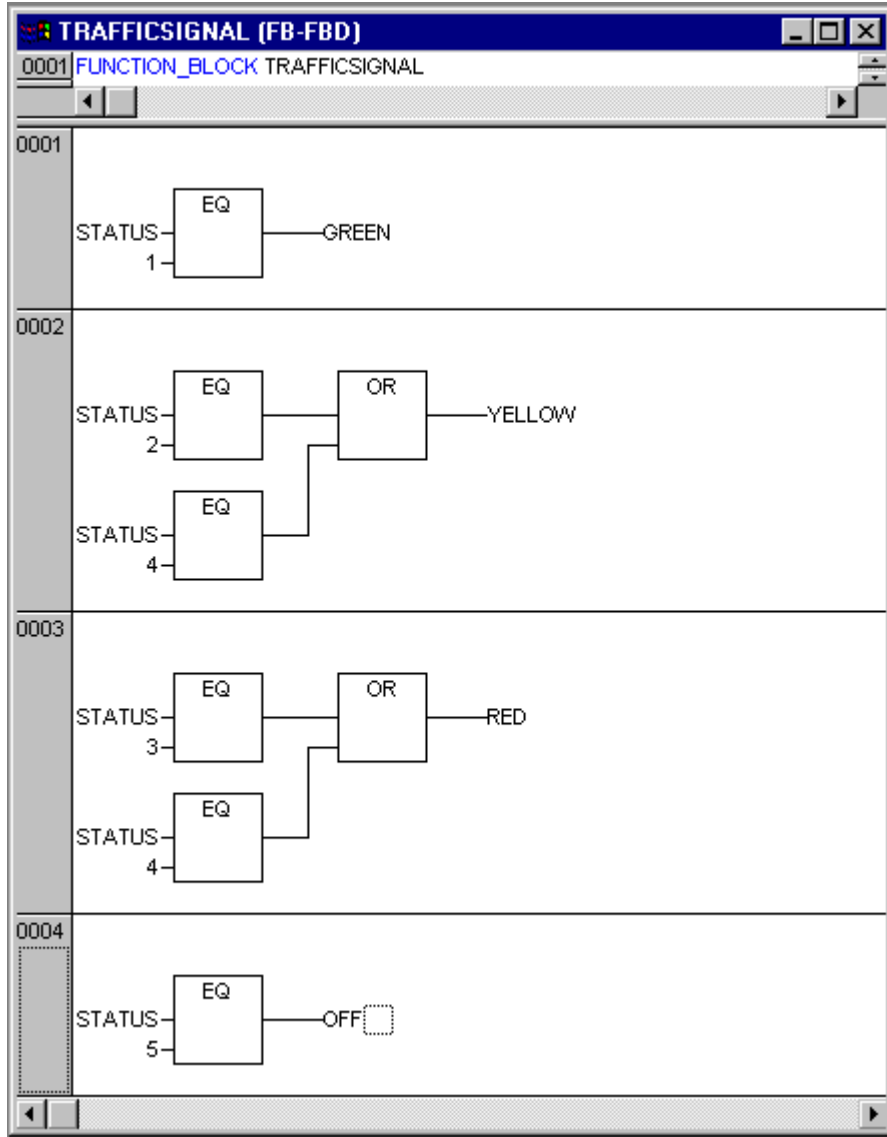
Измените три вопроса ??? на GREEN. Вы теперь создали цепь следующего вида:



STATUS сравнивается с 1, результат присваивается GREEN. Таким образом, GREEN будет включен, когда STATUS равен 1.

Для других цветов TRAFFICSIGNAL нам понадобятся еще две цепи. Создаете их командой "Insert" "Network (after)". Законченный POU должен выглядеть следующим образом:

Функциональный блок TRAFFICSIGNAL:



Чтобы вставить оператор перед входом другого оператора, Вы должны выделить сам вход, а не текст (выделяется прямоугольником). Далее используйте команду "Insert" "Operator".

Теперь наш первый POU закончен. Как и планировалось, TRAFFICSIGNAL будет управлять включением выходов, руководствуясь значением переменной STATUS.

Подключение standard.lib

Для создания таймера в POU WAIT нам понадобится POU из стандартной библиотеки. Итак, откройте менеджер библиотек командами "Window" "Library Manager". Выберите "Insert" "Additional library". Должно открыться диалоговое окно выбора файлов. Выберите standard.lib из списка библиотек.

Объявления "WAIT"

Теперь давайте вернемся к POU WAIT. Как предполагалось, этот POU будет работать таймером, задающим длительность стадий TRAFFICSIGNAL. Наш POU должен иметь входную переменную TIME типа TIME и генерировать на выходе двоичную (Boolean) переменную, которую мы назовем OK. Данная переменная должна принимать значение TRUE, когда желательный период времени закончен.

Предварительно мы устанавливаем эту переменную в FALSE в конце строки объявления (но до точки с запятой, однако) " := FALSE ".

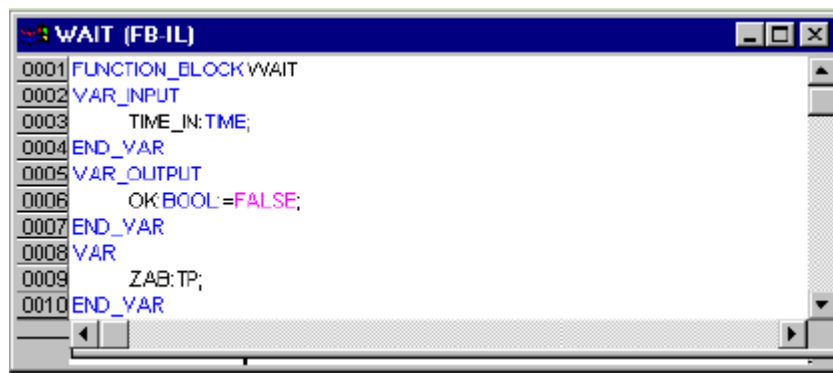
Теперь нам нужен генератор времени POU TP. Он имеет два входа (IN, PT) и два выхода (Q, ET). TP делает следующее:

Пока IN установлен в FALSE, ET будет 0 и Q будет FALSE. Как только IN переключится в TRUE, выход ET начнет отсчитывать время в миллисекундах. Когда ET достигнет значения заданного PT, счет будет остановлен. Тем временем выход Q равен TRUE, пока ET меньше PT. Как только ET достигнет значения PT, выход Q снова переключится в FALSE.

Описание всех POU из стандартной библиотеки приведено в приложении.

Чтобы использовать TP в POU WAIT, мы должны создать его локальный экземпляр. Для этого мы объявляем локальную переменную ZAB (отсчитанное время) типа TP (между ключевыми словами VAR, END_VAR).

Раздел объявлений WAIT теперь должен выглядеть так:



```

0001 FUNCTION_BLOCK WAIT
0002 VAR_INPUT
0003     TIME_IN:TIME;
0004 END_VAR
0005 VAR_OUTPUT
0006     OK:BOOL:=FALSE;
0007 END_VAR
0008 VAR
0009     ZAB:TP;
0010 END_VAR

```


Программируем "WAIT"

Для создания желаемого таймера текст программы должен быть следующим:

```

0001 FUNCTION_BLOCK WAIT
0002 LD ZAB.Q
0003 JMP mark
0004
0005 CAL ZAB(IN:=FALSE)
0006 LD TIME_IN
0007 ST ZAB.PT
0008 CAL ZAB(IN:=TRUE)
0009 JMP end
0010
0011 mark:
0012 CAL ZAB
0013 end:
0014 LDN ZAB.Q
0015 ST OK
0016 RET

```

Сначала проверяется, установлен ли Q в TRUE (возможно, отсчет уже запущен), в этом случае мы не трогаем установки ZAB, а вызываем функциональный блок ZAB без входных переменных - чтобы проверить, закончен ли период времени.

Иначе мы устанавливаем переменную IN ZAB в FALSE и одновременно ET в 0 и Q в FALSE. Таким образом, все переменные установлены в начальное состояние. Теперь мы устанавливаем необходимое время TIME переменной PT и вызываем ZAB с IN:=TRUE. Функциональный блок ZAB теперь будет работать, пока не достигнет значения TIME и не установит Q в FALSE.

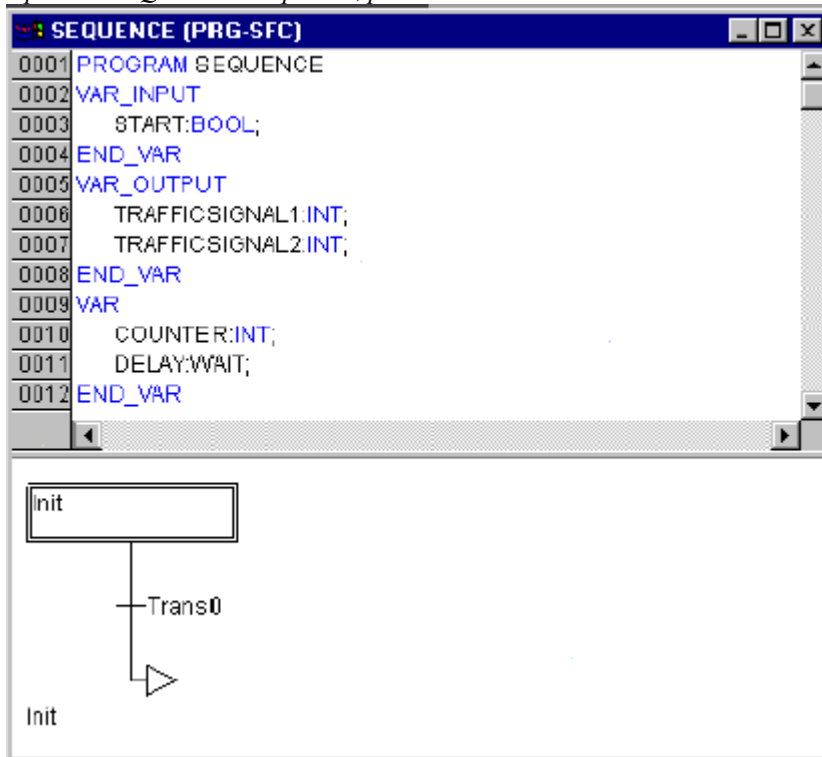
Инвертированное значение Q будет сохраняться в переменной OK после каждого выполнения WAIT. Как только Q станет FALSE, OK примет значение TRUE.

С таймером покончено. Теперь пришло время объединять наши два блока WAIT и TRAFFICSIGNAL в главной программе PLC_PRG.

"SEQUENCE" версия 1

Сначала объявим необходимые переменные. Это входная переменная START типа BOOL, две выходных переменные TRAFFICSIGNAL1 и TRAFFICSIGNAL2 типа INT и одна типа WAIT (DELAY оригинально, не так ли). Программа SEQUENCE теперь выглядит так:

Программа *SEQUENCE* версия 1, раздел объявлений:



Создаем SFC диаграмму

Первоначально SFC граф всегда состоит из этапа "Init" перехода "Trans0" и возврата назад к Init, естественно, нам придется несколько дополнить его.

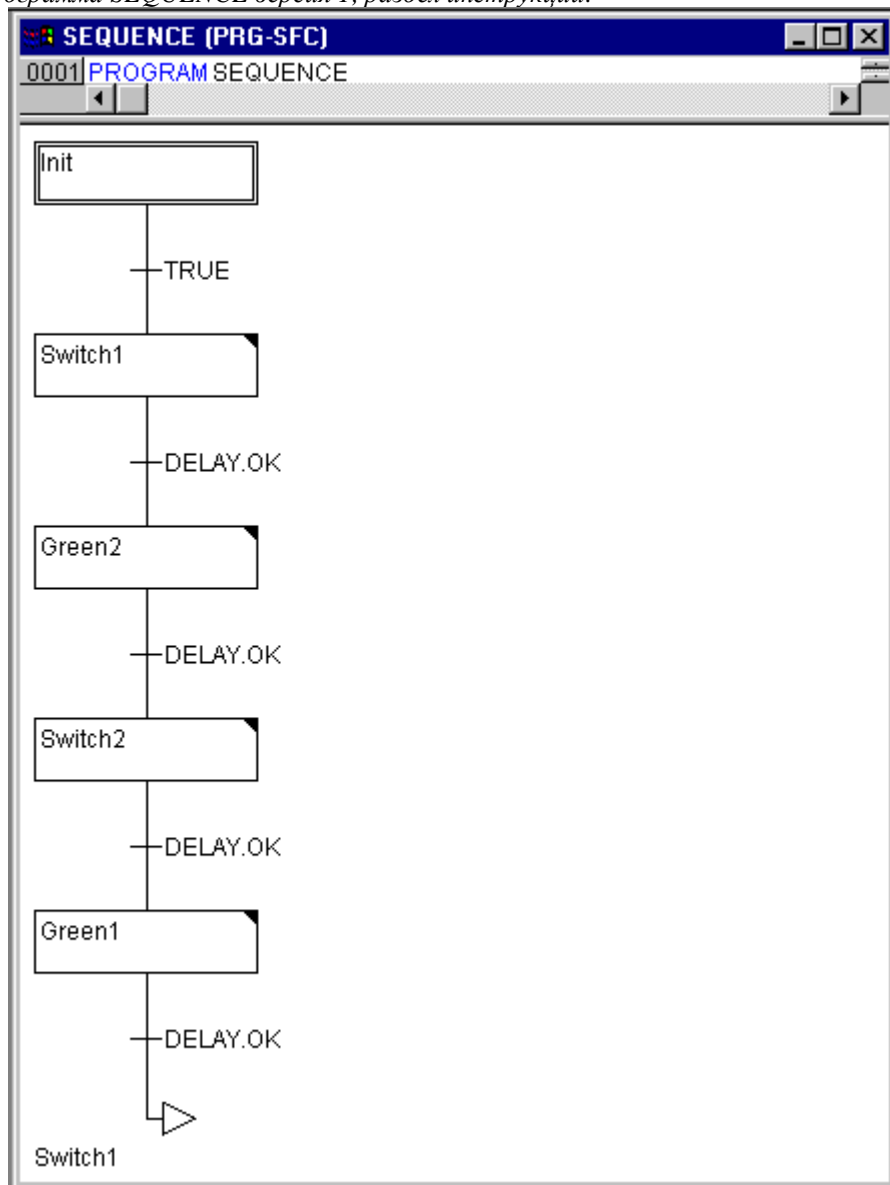
Прежде чем программировать конкретные этапы и переходы, выстроим структуру графа. Сначала нам понадобятся этапы для каждой стадии TRAFFICSIGNAL. Вставьте их, отмечая Trans0 и выбирая команды "Insert" "Step transition (after)". Повторите эту процедуру еще три раза.

Для редактирования названия перехода или этапа нужно просто щелкнуть мышкой на нужном тексте. Назовите первый переход после Init "START", а все прочие переходы "DELAY.OK".

Первый переход разрешается, когда START устанавливается в TRUE, все же прочие - когда DELAY в OK станет TRUE, т.е. когда заданный период закончится.

Этапы (сверху вниз) получают имена Switch1, Green2, Switch2, Green1, ну и Init, конечно, сохранит своё имя. "Switch" должен включать жёлтую фазу, в Green1 TRAFFICSIGNAL1 будет зеленым, в Green2 TRAFFICSIGNAL2 будет зеленым. Наконец, измените адрес возврата Init на Switch1. Если вы всё сделали верно, диаграмма должна выглядеть так:

Программа SEQUENCE версия 1, раздел инструкций:

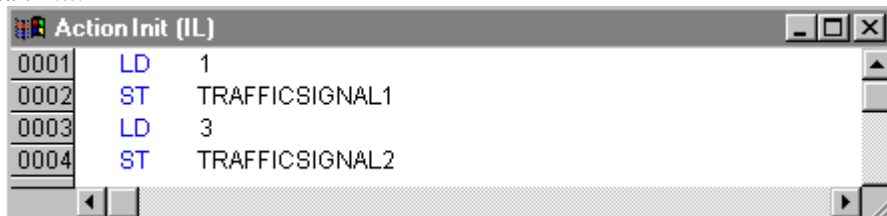


Теперь мы должны запрограммировать этапы. Если вы сделаете двойной щелчок мышью на изображении этапа, то должен открыться диалог определения нового действия. В нашем случае мы будем использовать язык IL (Список Инструкций).

Программирование этапов и переходов

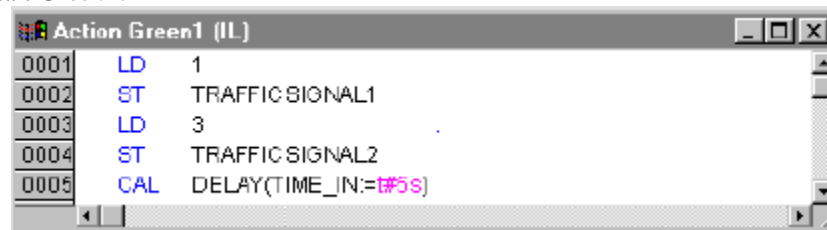
Во время действия этапа Init проверяем, активен сигнал включения START или нет. Если сигнал не активен, то светофор выключается. Этого можно достичь, если записать в переменные TRAFFICSIGNAL1 и TRAFFICSIGNAL2 число 5.

Этап Init:



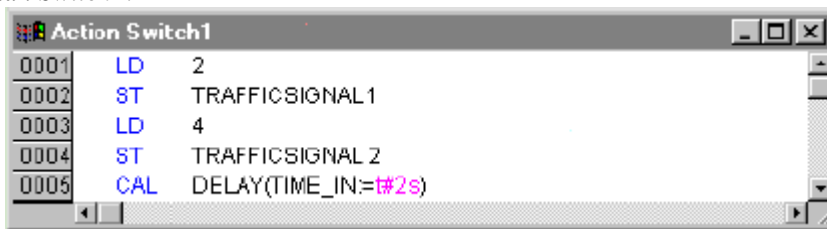
В Green1 TRAFFICSIGNAL1 будет зеленым (STATUS:=1), TRAFFICSIGNAL2 будет красным (STATUS:=3), задержка в 5000 миллисекунд.

Этап Green1:



Switch1 изменяет состояние TRAFFICSIGNAL1 на 2 (жёлтое) и, соответственно, TRAFFICSIGNAL2 на 4 (жёлто-красное). Кроме того, теперь устанавливается задержка в 2000 миллисекунд. Это выглядит так:

Этап Switch1:



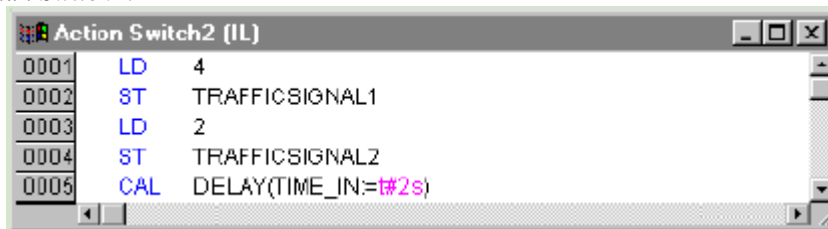
Green2 включает красный в TRAFFICSIGNAL1 (STATUS:=3) и зеленый в TRAFFICSIGNAL2 (STATUS:=1). Задержка устанавливается в 5000 миллисекунд.

Этап Green2:



В Switch2 STATUS в TRAFFICSIGNAL1 изменяется на 4 (жёлто-красный), соответственно, TRAFFICSIGNAL2 будет 2 (жёлтый). Задержка теперь должна быть в 2000 миллисекунд.

Этап Switch2:



Первая версия нашей программы закончена.

Если вы хотите проверить ее работу в режиме эмуляции, сделайте следующее:

Откройте POU PLC_PRG. Каждый проект начинает работу с PLC_PRG. Вставьте в него компонент и замените “AND” на “SEQUENCE”. Входы и выходы оставьте пока свободными.

Теперь вы можете откомпилировать ('Project' 'Build') и её проверить отсутствие ошибок. В окне сообщений вы должны увидеть текст: "0 Errors, 0 Warnings".

Теперь включите флажок 'Online' 'Simulation' и дайте команду 'Online' 'Login'. Запустите программу 'Online' 'Run'.

Откройте программу SEQUENCE. Программа запущена, но не работает, поскольку переменная START должна иметь значение TRUE. Далее это будет делать PLC_PRG, но сейчас вы можете изменить ее вручную. Для этого щелкните дважды мышью по объявлению этой переменной. Ее значение теперь выделено цветом и равно TRUE. Дайте команду записи значений переменных ('Online' 'Write values'). Теперь вы можете понаблюдать за работой программы. Активные шаги диаграммы выделяются голубым цветом.

Для продолжения редактирования программы закройте режим онлайн командой 'Online' 'Logout'.

"SEQUENCE" вторая версия

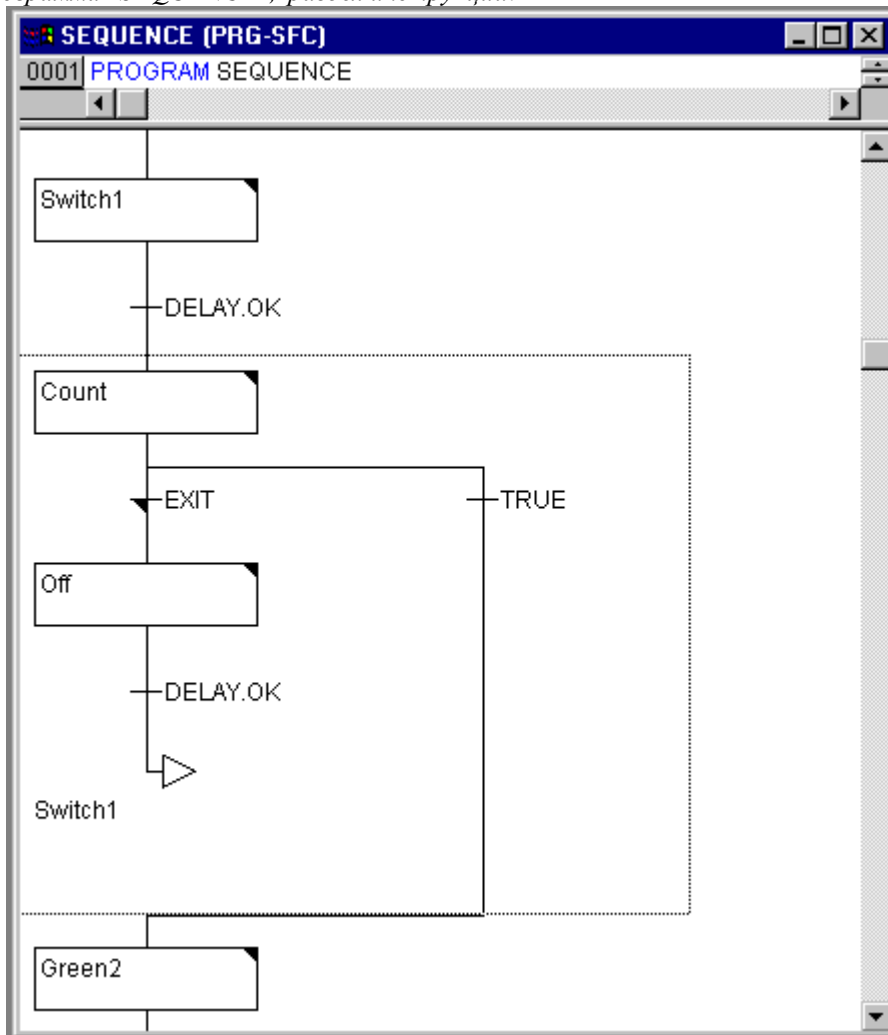
Теперь немного усложним нашу программу. Разумно будет выключать наши светофоры на ночь. Для этого мы создадим в программе счетчик, который после некоторого числа циклов TRAFFICSIGNAL произведет отключение устройства.

Для начала нам нужна новая переменная COUNTER типа INT. Объявите её как обычно в разделе объявлений SEQUENCE.

Теперь выберете переход после Switch1 и вставьте ещё один этап и переход. Выберете результирующий переход и вставьте альтернативную ветвь вправо. После левого перехода вставьте дополнительный этап и переход. После нового результирующего перехода вставьте удаленный переход (jump) на Init.

Назовите новые части так: верхний из двух новых этапов нужно назвать "Count" и нижний "Off". Переходы будут называться (сверху вниз слева на право) EXIT, TRUE и DELAY.OK. Теперь новые части должны выглядеть как фрагмент, выделенный рамкой.

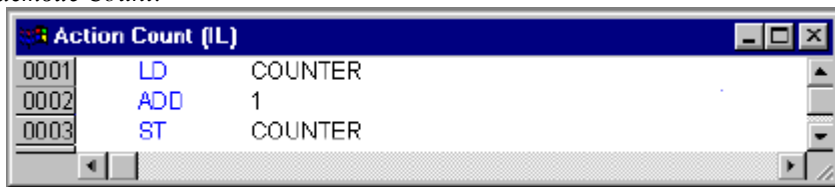
Программа "SEQUENCE", раздел инструкций:



Теперь два новых этапа и перехода необходимо наполнить содержанием.

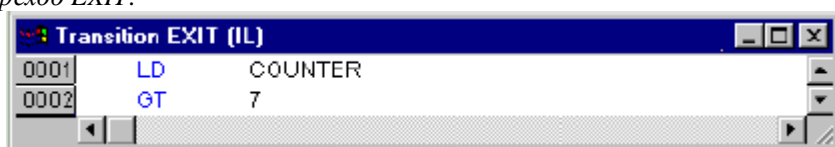
На этапе Count выполняется только одно действие - COUNTER увеличивается на 1:

Действие Count:



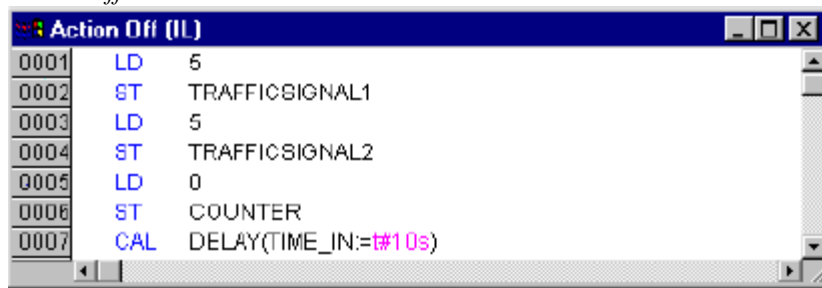
На переходе EXIT1 проверяется достижение счетчиком заданного значения, например 7:

Переход EXIT:



На этапе Off состояние обоих светофоров устанавливается в 5 (светофор выключен), COUNTER сбрасывается в 0 и устанавливается задержка времени в 10 секунд.

Действие Off:



Результат

В нашей гипотетической ситуации ночь наступает после семи циклов TRAFFICSIGNAL. Светофоры полностью выключаются до рассвета, и процесс повторяется снова. При желании вы можете еще раз проверить работу программы в эмуляторе, прежде чем продолжить ее усовершенствование.

PLC_PRG

Мы определили два строго коррелированных во времени светофора в блоке SEQUENCE. Теперь полностью закончим программу. Для этого необходимо распределить входные и выходные переменные в блоке PLC_PRG. Мы хотим дать возможность запустить систему выключателем IN и хотим обеспечить переключение всех шести ламп (2 светофора) путем передачи "команд переключения" на каждом шаге SEQUENCE. Объявим теперь соответствующие Boolean переменные для всех шести выходов и одного входа, затем создадим программу и сопоставим переменные соответствующим IEC адресам..

Следующий шаг - это объявление переменных LIGHT1 и LIGHT2 типа TRAFFICSIGNAL в редакторе объявлений.

Объявление LIGHT1 и LIGHT2:



Для представления шести ламп светофоров нужно 6 переменных типа Boolean. Однако мы не будем объявлять их в разделе объявлений блока PLC_PRG, вместо этого используем глобальные переменные (Global Variables) из ресурсов (Resources). Двоичная входная переменная IN, необходимая для установки переменной START блока SEQUENCE в TRUE, будет определена таким же образом. Выберите вкладку Resources и откройте список Global Variables.

Объявление глобальных переменных:

```

0001 VAR_GLOBAL
0002     IN:BOOL;
0003     L1_GREEN:BOOL;
0004     L1_YELLOW:BOOL;
0005     L1_RED:BOOL;
0006     L2_GREEN:BOOL;
0007     L2_YELLOW:BOOL;
0008     L2_RED:BOOL;
0009 END_VAR
0010
    
```

Закончим PLC_PRG. Для этого мы перейдем в окно редактора. Мы выбрали редактор Continuous Function Chart, и, следовательно нам доступна соответствующая панель инструментов.

Щелкните правой клавишей мыши в окне редактора и выберите элемент Box. Щелкните на тексте AND и напишите "SEQUENCE". Элемент автоматически преобразуется в SEQUENCE с уже определенными входными и выходными переменными.

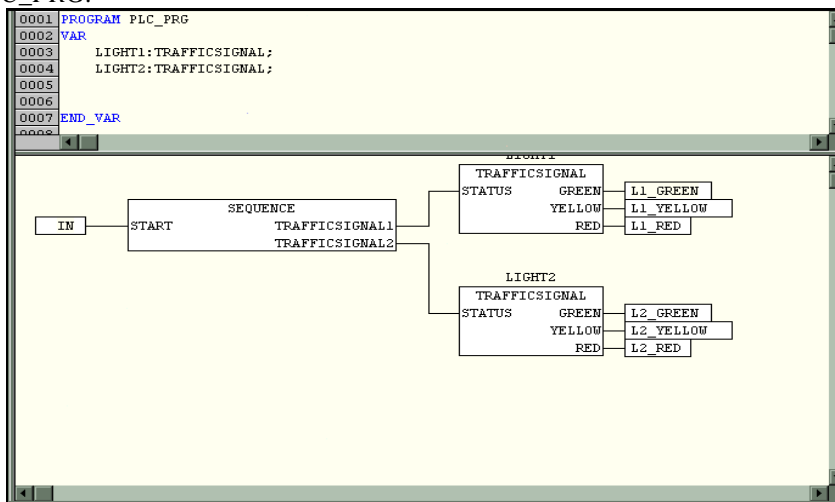
Вставьте далее два элемента и назовите их TRAFFICSIGNAL. TRAFFICSIGNAL - это функциональный блок, и, как обычно, Вы получите три красных знака вопроса, которые нужно заменить уже объявленными локальными переменными LIGHT1 и LIGHT2.

Теперь создайте элемент типа Input, который получит название IN и шесть элементов типа Output, которым нужно дать следующие имена: L1_green, L1_yellow, L1_red, L2_green, L2_yellow, L2_red.

Все элементы программы теперь на месте, и Вы можете соединять входы и выходы. Для этого щелкните мышью на короткой линии входа/выхода и тяните ее (не отпуская клавишу мыши) к входу/выходу нужного элемента.

Наконец Ваша программа должна принять вид, показанный ниже.

PLC_PRG:



Теперь наша программа полностью готова.

TRAFFICSIGNAL эмуляция

Теперь проверьте окончательно вашу программу в режиме эмуляции. Убедитесь в правильности ее работы, контролируя последовательность выполнения и значения переменных в окнах редакторов CoDe-Sys.

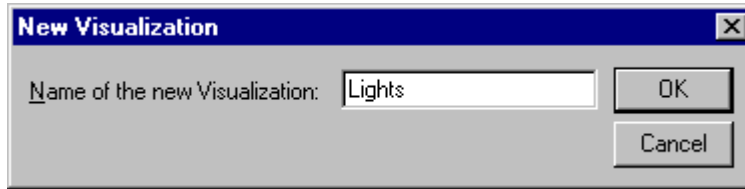
3.2 Визуализация примера

С помощью визуализации можно быстро и легко оживить переменные проекта. Полное описание визуализации Вы найдете в главе 8. Сейчас мы нарисуем два светофора и их выключатель, который позволит нам включать и выключать блок управления светофором.

Создание новой визуализации

Для того чтобы создать визуализацию, выберите вкладку Visualizations в организаторе объектов. Теперь выполните команду 'Project' 'Object Add'.

Диалог для создания новой визуализации:

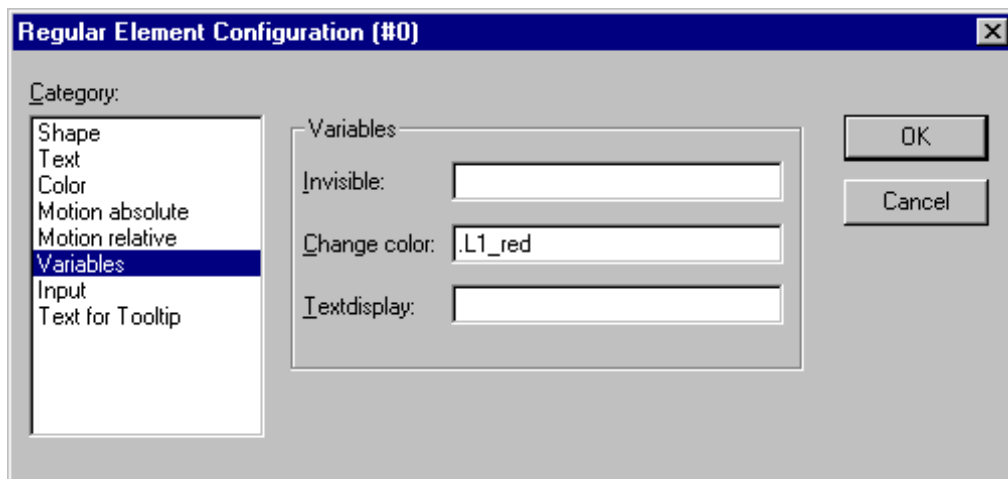


Введите любое имя для визуализации, например Lights. Когда Вы нажмете кнопку Ok, откроется окно, в котором вы будете создавать визуализацию.

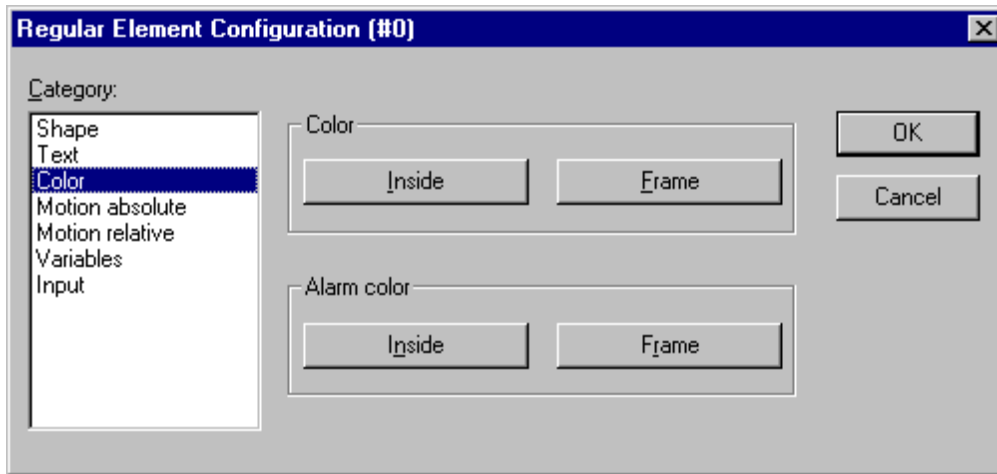
Вставка элемента в визуализацию

Для создания визуализации светофора выполните следующие действия:

- Выберите команду 'Insert' 'Ellipse' и нарисуйте окружность с диаметром около 2 сантиметров. Для этого щелкните мышью на рабочем поле и, удерживая левую кнопку мыши, растяните появившуюся окружность до требуемого размера.
- Дважды щелкните мышью на окружности. Появится диалоговое окно для настройки элемента визуализации.
- Выберите категорию Variables и в поле Change color введите имя переменной .L1_red. Вводить имя переменной удобно с помощью Input Assistant (клавиша <F2>). Глобальная переменная L1_red будет управлять цветом нарисованной Вами окружности.



- Выберите категорию Color. В области Color нажмите кнопку Inside и в появившемся окне выберите любой нейтральный цвет, например, черный.
- Нажмите кнопку Inside в области Alarm Color и выберите красный цвет.



Полученная окружность будет черной, когда значение переменной ложно, и красной, когда переменная истинна.

Таким образом, мы создали первый фонарь первого светофора.

Остальные цвета светофора.

Теперь вызовите команду копирования 'Edit' 'Copy' (<Ctrl>+<C>) и дважды выполните команду вставки 'Edit' 'Paste'(<Ctrl>+<V>). Вы получите две новых окружности. Перемещать эти окружности можно с помощью мышки. Расположите их так, чтобы они представляли собой вертикальный ряд в левой части окна редактора. Двойной щелчок по окружности приводит к открытию окна для настройки свойств элемента визуализации. В поле Change Color окон настройки свойств соответствующих окружностей введите следующие переменные:

для средней окружности: .L1_yellow
 для нижней окружности: .L1_green

В категории Color в области Alarm color установите цвета окружностей (желтый и зеленый).

Корпус светофора.

Теперь вызовите команду "Insert" "Rectangle" и вставьте прямоугольник так, чтобы введенные ранее окружности находились внутри него. Выберите цвет прямоугольника и затем выполните команду "Extras" "Send to back", которая переместит его на задний план. После этого окружности снова будут видны.

Активизируйте режим эмуляции, выполнив команду "Online" "Simulation"(режим эмуляции активен, если перед пунктом меню "Online" "Simulation" стоит галочка).

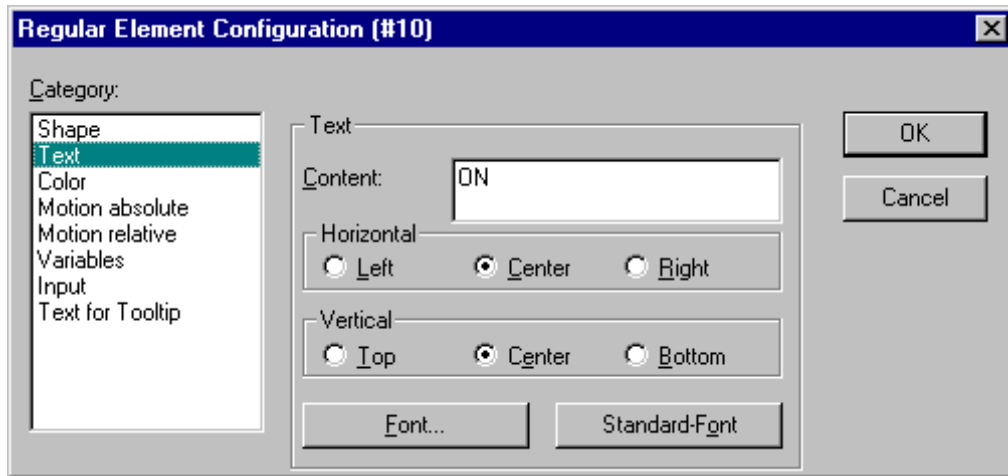
Запустите программу путем выполнения команд "Online" "Login" и "Online" "Run" и вы увидите, как будут меняться цвета светофора.

Второй светофор.

Самый простой способ создать второй светофор – скопировать все элементы первого. Выделите элементы первого светофора и скопируйте их, выполнив команды "Edit" "Copy" и "Edit" "Paste". Замените имена переменных, управляющих цветами (например, .L1_red на .L2_red), и второй светофор будет готов.

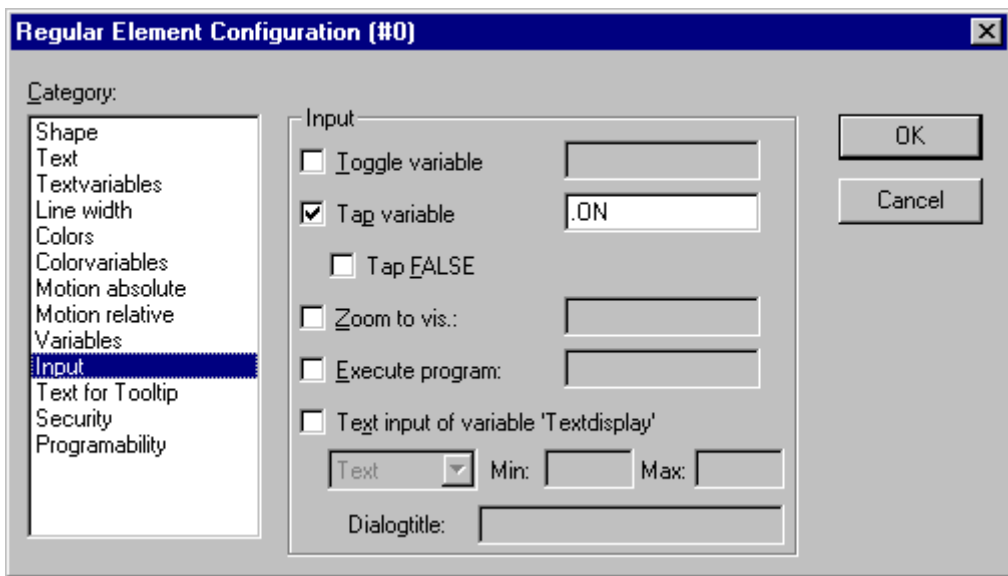
Переключатель ON.

Как описано выше, вставьте прямоугольник, установите его цвет и введите переменную .ON в поле Change Color категории Variables. В поле Content категории Text введите имя "ON".



Для того чтобы переменная ON переключалась при щелчке мышкой на этом элементе, в поле Toggle variable категории Input введите переменную .ON. Созданный нами переключатель будет включать/выключать светофоры.

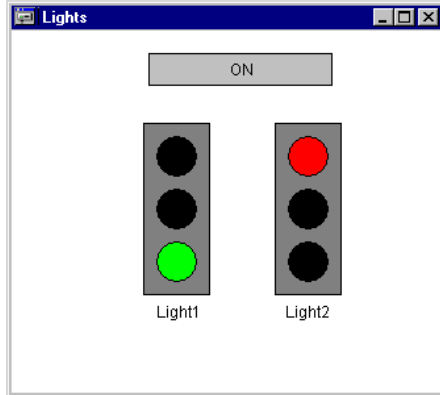
Отобразить включенное состояние можно цветом, как и для светофора. Впишите переменную в поле Change Color.



Надписи в визуализации.

Под светофорами вставим два прямоугольника. В свойствах элемента в категории Color цвет границы(frame)прямоугольника задайте белым. В поле Contents(категория Text) введите названия светофоров "Light1" "Light2".

Визуализация для проекта Traffic Signal:

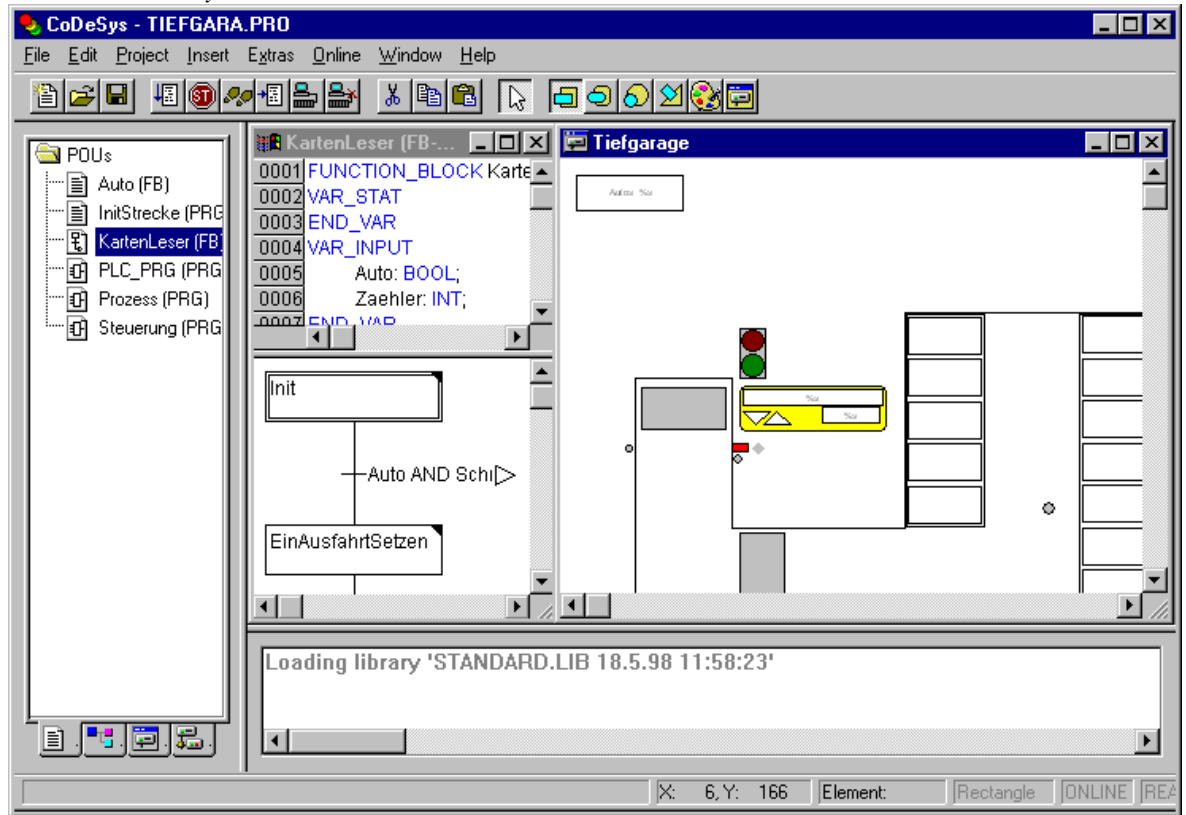


4 Работа в системе программирования CoDeSys

4.1 Главное окно

Элементы главного окна

Главное окно CoDeSys:



Главное окно CoDeSys состоит из следующих элементов (в окне они расположены сверху вниз):

- Меню.
- Панель инструментов. На ней находятся кнопки для быстрого вызова команд меню.
- Организатор объектов, имеющий вкладки POU, Data types, Visualizations и Resources.
- Разделитель Организатора объектов и рабочей области CoDeSys.
- Рабочая область, в которой находится редактор.
- Окно сообщений.
- Строка статуса, содержащая информацию о текущем состоянии проекта.

Панель инструментов, окно сообщений и строка статуса не являются обязательными элементами главного окна.

Меню

Меню находится в верхней части главного окна. Оно содержит все команды CoDeSys.

File Edit Project Insert Extras Online Window Help

Панель инструментов

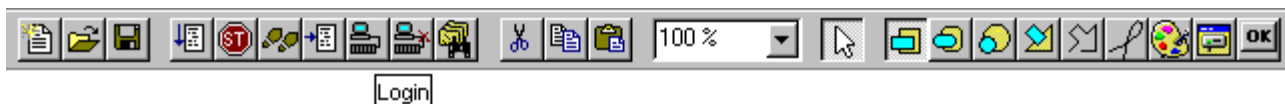
Кнопки на панели инструментов обеспечивают более быстрый доступ к командам меню.

Вызванная с помощью кнопки на панели инструментов команда автоматически выполняется в активном окне.

Команда выполнится, как только нажатая на панели инструментов кнопка будет отпущена. Если вы поместите указатель мышки на кнопку панели инструментов, то через небольшой промежуток времени увидите название этой кнопки в подсказке.

Кнопки на панели инструментов различны для разных редакторов CoDeSys. Получить информацию относительно назначения этих кнопок можно в описании редакторов.

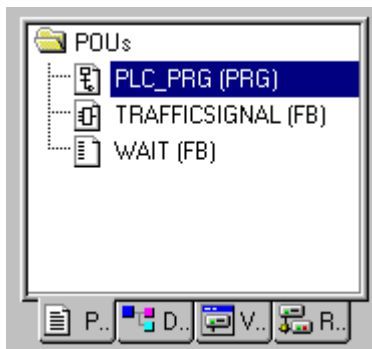
Панель инструментов можно отключить (см. 'Project' 'Options' категория Desktop)



Организатор объектов

Организатор объектов всегда находится в левой части главного окна CoDeSys. В нижней части организатора объектов находятся вкладки POU's, Data types (типы данных), Visualizations (визуализации) и Resources (ресурсы). Переключаться между соответствующими объектами можно с помощью мышки или клавиш перемещения. Правила работы с объектами организатора объектов описаны в главе “Управление объектами”.

Организатор объектов:



Разделитель экрана.

Разделить экрана – это граница между двумя непересекающимися окнами. В CoDeSys есть следующие разделители: между организатором объектов и рабочей областью, между разделом объявлений и разделом кода POU, между рабочей областью и окном сообщений. Вы можете перемещать разделители с помощью мышки, нажав ее левую кнопку.

Разделитель сохраняет свое положение даже при изменении размеров окна. Если вы больше не видите разделителя на экране, значит, стоит изменить размеры окна.

Рабочая область.

Рабочая область находится в правой части главного окна CoDeSys. Все редакторы, а также менеджер библиотек открываются именно в этой области. Имя открытого объекта находится в заголовке окна.

Окно сообщений.

Окно сообщений отделено от рабочей области разделителем. Именно в этом окне появляются сообщения компилятора, результаты поиска и список перекрестных ссылок.

При двойном щелчке мышкой или при нажатии клавиши <Enter > на сообщении будет открыт объект, к которому относится данное сообщение.

С помощью команд “Edit” ”Next error” и ”Edit ” ” Previous error ” можно быстро перемещаться между сообщениями об ошибках.

Окно сообщений можно убрать либо включить с помощью команды “Window” “ Message”.

Статусная строка

Статусная строка находится в нижней части главного окна CoDeSys и предоставляет информацию о проекте и командах меню.

При выборе пункта меню его описание появляется в левой части строки статуса.

Если вы работаете в режиме online, то надпись **Online** в строке статуса выделяется черным цветом. В ином случае надпись серая.

С помощью статусной строки в режиме online можно определить, в каком состоянии находится программа: **SIM** – в режиме эмуляции, **RUN** – программа запущена, **BP**- установлена точка останова, **FORCE** – происходит фиксация переменных.

При работе в текстовом редакторе в строке статуса указывается позиция, в которой находится курсор (например, **Line:5, Col:11**). В режиме замены надпись “**OV**” выделяется черным цветом. Нажимая клавишу <Ins> можно переключаться между режимом вставки и замены.

В визуализации в статусной строке выводятся координаты курсора X и Y, которые отсчитываются относительно верхнего левого угла окна. Если указатель мыши находится на элементе или над элементом производятся какие-либо действия, то указывается номер этого элемента. При вставке элемента в строке статуса указывается его название (например, Rectangle).

Если вы поместили указатель на пункт меню, то в строке статуса появляется его краткое описание.

Статусную строку можно убрать либо включить (см. 'Project' 'Options' категория Desktop)

Контекстное меню

Быстрый вызов: <Shift>+<F10>

Вместо того, чтобы использовать главное меню для вызова команд, можно воспользоваться контекстным меню. Это меню, вызываемое правой кнопкой мыши, содержит наиболее часто используемые команды.

4.2 Опции проекта

В CoDeSys с помощью команды “Project” “Options” вы можете настраивать вид главного окна. Опции делятся на несколько категорий. Выбор нужной категории в левой части диалогового окна производится с помощью мышки или клавиш перемещения, соответствующие опции отображаются в правой части окна.

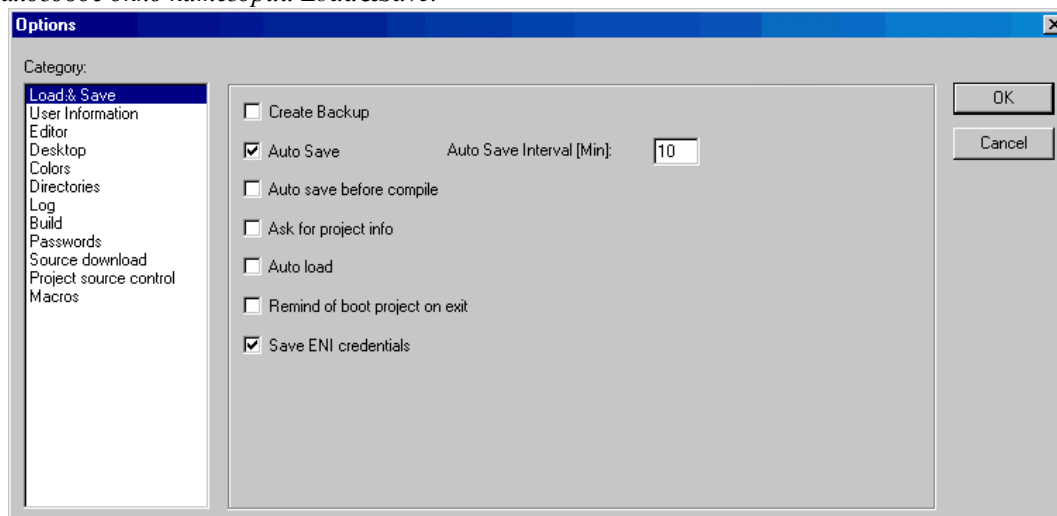
Полный образ опций проекта вы найдете в разделе 'Workspace' на вкладке Resources.

Общие настройки сохраняются в файле "CoDeSys.ini" и восстанавливаются при следующем запуске CoDeSys.

Для каждой категории настройка производится в отдельном диалоговом окне. Если какая-либо опция включена, то перед ней появляется галочка.

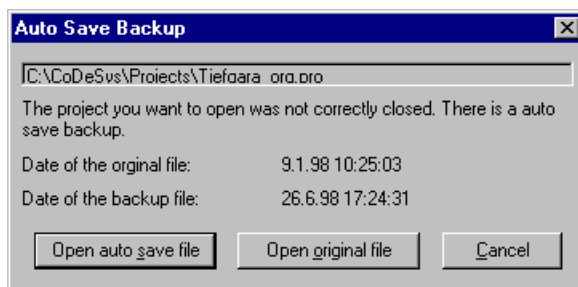
Чтение и Запись (Load&Save)

Диалоговое окно категории Load&Save:



☐ **Create Backup** указывает CoDeSys создавать резервный файл с расширением ".bak" при каждом сохранении. В отличие от ".asd" файла (см. ниже) он не удаляется при закрытии проекта. Благодаря чему вы всегда сможете восстановить предыдущую версию проекта.

☐ **AutoSave** заставляет CoDeSys периодически сохранять проект во временном файле с расширением ".asd", в рабочей директории проекта. Сохранение происходит через промежуток времени, указанный в поле **Auto Save Interval**. При нормальном выходе из CoDeSys этот файл удаляется. Если произошла какая-либо авария (например, у компьютера пропало питание), то этот файл не удаляется. При следующей попытке открыть проект появится следующее сообщение:



Выбор *Open auto save file* даст возможность восстановить заведомо «исправный» проект из автоматически сохраненного файла.

Если для редактирования открыта библиотека, то временный файл имеет расширение ".asl".

☐ **Auto save before compile** создает аналогичные временные файлы перед каждой компиляцией проекта.

• Если выбрана опция **Ask for project info**, то при сохранении проекта под новым именем автоматически вызывается диалоговое окно для ввода информации о проекте. Это же окно появляется при выборе команды “Project” “Project info”.

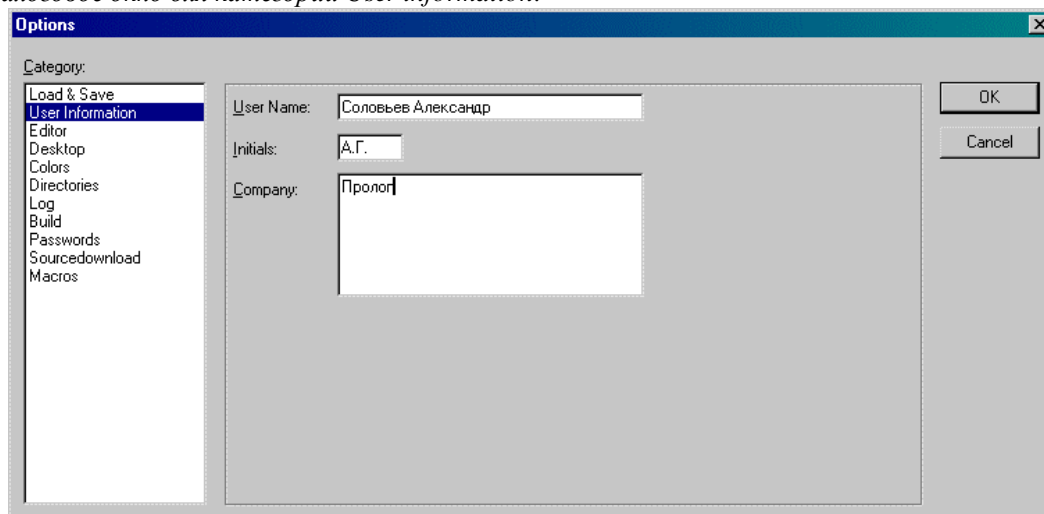
• Опция **Auto Load** обеспечивает автоматическое открытие последнего открытого проекта при запуске CoDeSys. Заметим, что при запуске CoDeSys может открыть и другой проект, если его имя указано в командной строке.

• **Remind of boot project on exit**: Если вы изменили проект и загрузили его в контроллер, не создав после этого загрузочный проект, то при попытке закрыть проект вы получите напоминание: " После последней загрузки загрузочный проект не был создан. Закрыть проект? " (No boot project created since last download. Exit anyway ?).

• **Save ENI credentials** (сохранять удостоверение пользователя): Имя и пароль для диалога Login базы данных ENI будут сохраняться в проекте.

Информация о пользователе (User information)

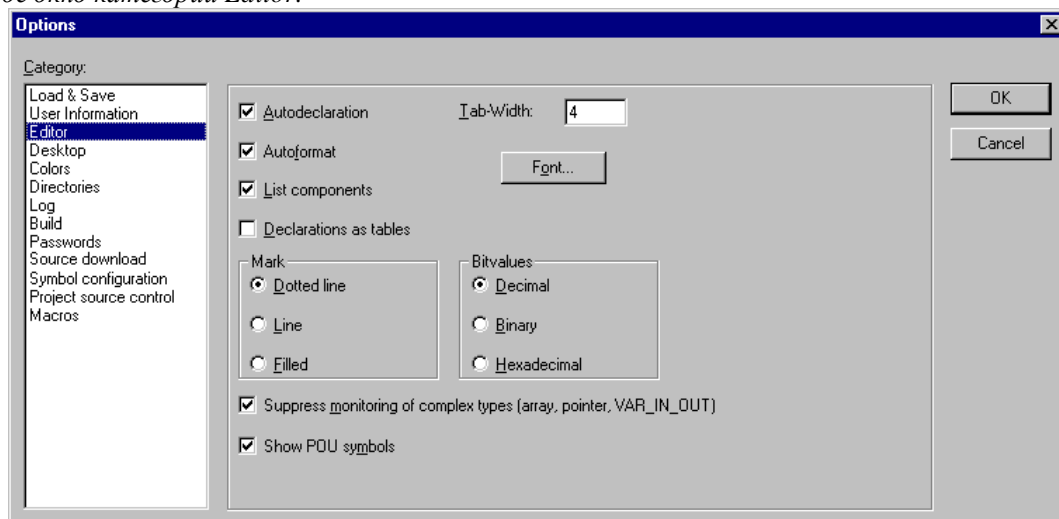
Диалоговое окно для категории User information:



Здесь пользователь может ввести свое имя (Name) и инициалы (Initials), а также название компании (Company). Каждое из полей свободно редактируется. Заданная информация будет автоматически вставлена во все новые проекты, созданные на данном компьютере.

Редактор (Editor).

Диалоговое окно категории Editor:



☐ **Autodeclaration:** при вводе имени новой переменной автоматически будет предложен диалог для ее объявления.

☐ **Autoformat:** CoDeSys будет автоматически выполнять форматирование текста в IL редакторе и разделах объявлений. По окончании ввода строки:

1. Операторы будут отображаться заглавными буквами.
2. Для разделения полей будет добавлен символ табуляции.

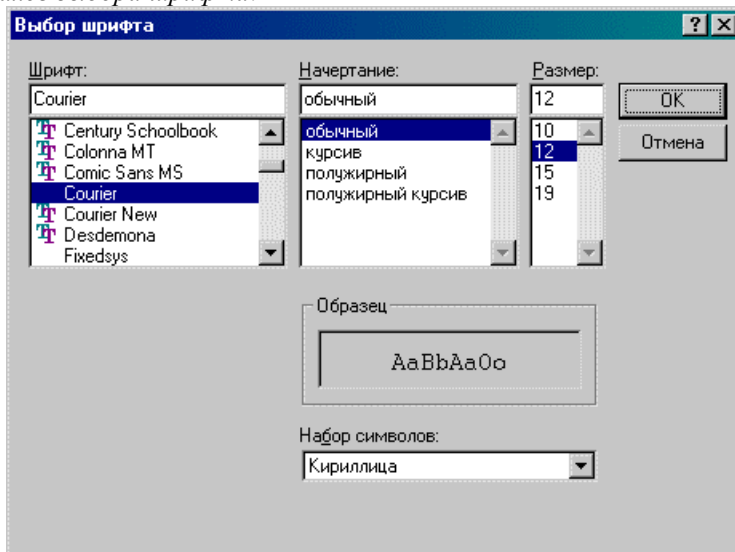
☐ **List components:** включает функцию интеллектуального анализа ввода (*Intellisense*). Работает это так: вы ставите точку в позиции, куда необходимо вставить идентификатор. Затем открывается список глобальных переменных проекта. Если вы вводите имя экземпляра функционального блока, будет открыт список всех входов и выходов блока. Функция *Intellisense* доступна в редакторах, в менеджере рецептов, в визуализации и трассировке.

☐ **Declarations as tables:** раздел объявлений будет отображаться в виде карточек с таблицами. На отдельных карточках будут представлены входные, выходные, локальные переменные и переменные вход-выход (*in_out*). Каждая таблица будет содержать столбцы: имя (**Name**), адрес (**Address**), тип (**Type**), начальное значение (**Initial**) и комментарий (**Comment**).

Поле **Tab-Width** определяет шаг позиций табуляции в окнах редакторов. По умолчанию 4 символа. Ширина одного символа определяется выбранным шрифтом.

Кнопка **Font** открывает стандартный диалог выбора шрифта редакторов. Обратите внимание, что увеличение размера шрифта сказывается на отображении и печати всех элементов редакторов CoDeSys.

Диалог выбора шрифта:

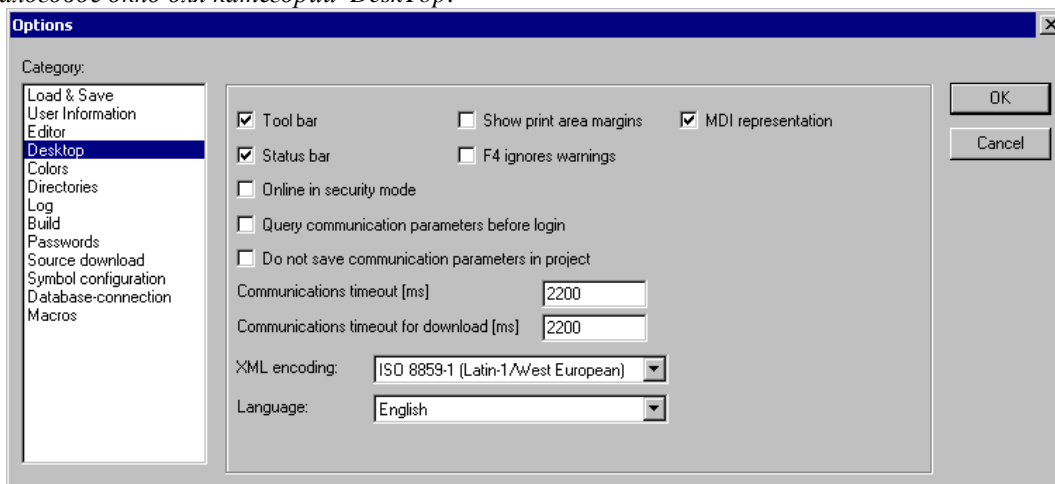


Опции **Mark** определяют, как будет выглядеть выделение в графических редакторах. Если выбрана опция **Dotted**, то курсор – это прямоугольник с пунктирной границей, если **Line**, то выделение – прямоугольник со сплошной границей, **Filled** – прямоугольник закрашен черным.

Опции **Bitvalues** определяют систему счисления (по умолчанию) для отображения значений переменных типа битовых строк (Byte, Word и Dword): двоичные числа (**Binary**), шестнадцатеричные (**Hexadecimal**) и десятичные (**Decimal**).

Рабочий стол (DeskTop)

Диалоговое окно для категории *DeskTop*:



☐ **Tool bar** указывает отображать панель инструментов (под главным меню).

☐ **Status bar** указывает отображать статусную строку.

☐ **Online in Security mode** указывает выводить запрос на подтверждение при выполнении команд '**Run**', '**Stop**', '**Reset**', '**Toggle Breakpoint**', '**Single cycle**', '**Write values**', '**Force values**'. Если поддержано в целевой платформе, то будет дан дополнительный диалог, включающий информацию о проекте и подтверждающий замену уже записанного в контроллере проекта.

Ö **Query communication parameters before login:** перед командой 'Online' 'Login' будет открыт диалог настройки параметров коммуникации. Для перехода в режим online нужно будет выбрать соединение и нажать ОК.

Ö **Do not save communication parameters in project:** настройки параметров канала коммуникации ('Online' 'Communication Parameters') не будут сохраняться в проекте.

Ö **Show print area margins:** в окнах редактора будут показаны ограничители, соответствующие заданным параметрам страницы при печати.

Ö **F4 ignores warnings:** после компиляции клавиша F4 перемещает фокус ввода в окне сообщений только по сообщениям об ошибках, игнорируя предупреждения.

Ö **MDI representation:** указывает использовать MDI интерфейс окна CoDeSys, опция активна по умолчанию. При отключенной опции используется SDI интерфейс.

Communications timeout [ms]: время таймаута для стандартных коммуникационных сервисов. Время в миллисекундах, после которого разрывается связь с системой исполнения, если не выполняется ни каких активных действий. Допустимые значения: 1-10000000 мс.

Communications timeout for download [ms]: время таймаута для длительных коммуникационных сервисов (закружка кода проекта, файлов, создание и контроль загрузочного проекта): Время в миллисекундах, после которого разрывается связь с системой исполнения, если не выполняется ни каких активных действий (Download Wait Time). Допустимые значения: 1-10000000 мс.

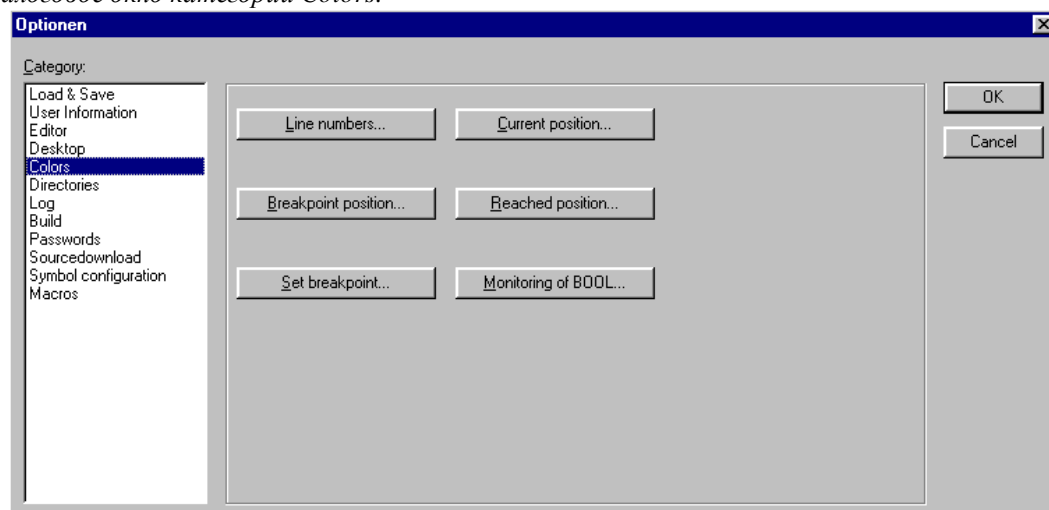
XML-Encoding: кодировка для XML файлов импорта. По умолчанию "ISO 8859-1". Касается работы через ENI, Message Interface и COM Automation Interface, а также пользовательского экспорта из CoDeSys посредством XML. Не влияет на Licensing Manager.

Language: Определяет язык в меню и в окнах диалога.

Замечание: функция недоступна в Windows 98!

Цвета (Colors)

Диалоговое окно категории Colors:



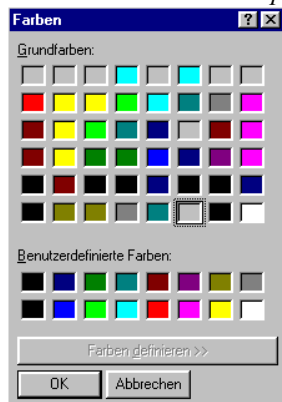
Здесь Вы можете редактировать цветовые установки CoDeSys. Вы можете изменить цвет для номеров строк (Line numbers), текущей позиции (Current position), позиций точек останова (Breakpoint positions),

установленных точек останова(Set breakpoint), пройденных позиций(Reached Positions), цвет при мониторинге значений логических переменных (Monitoring of Bool). По умолчанию установлены следующие цвета:

- номер строки - светло-серый
- текущая позиция - красный
- точки останова – темно-серый
- установленная точка останова - голубой
- пройденная позиция - зеленый
- цвет при мониторинге значений логических переменных – синий

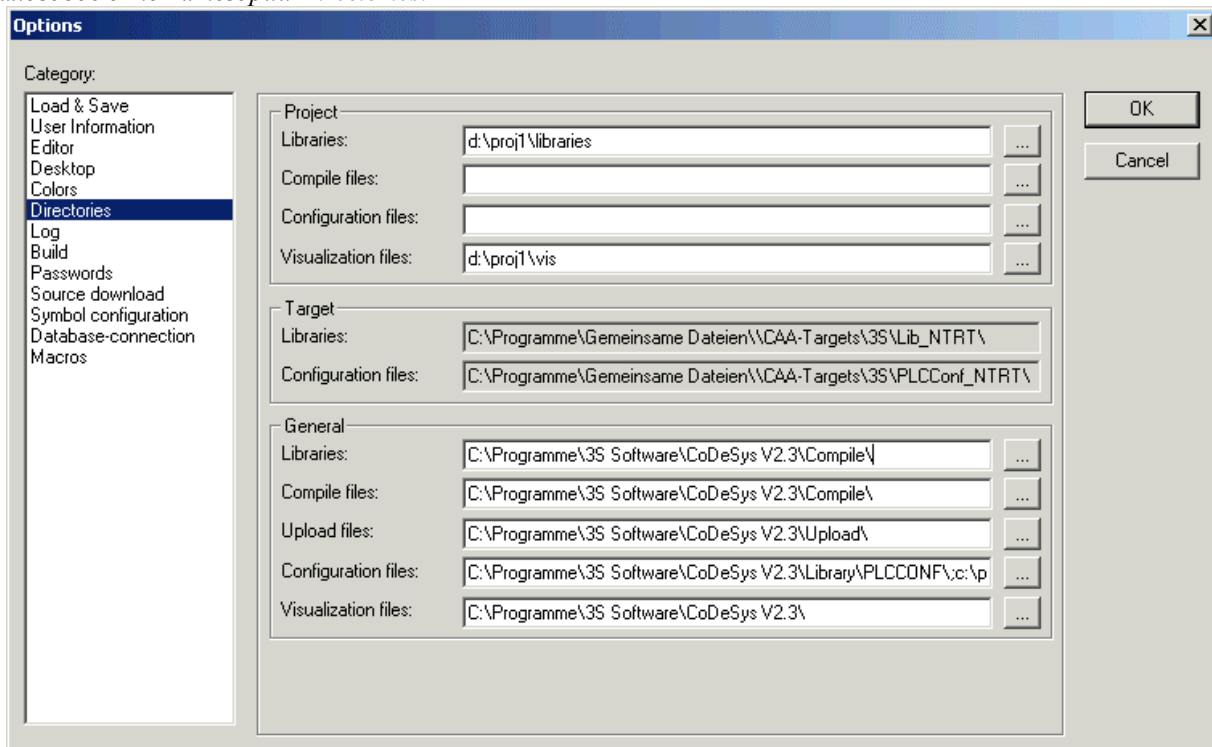
Выбор цвета осуществляется в стандартном диалоговом окне.

Диалоговое окно для выбора цвета:



Директории (Directories)

Диалоговое окно категории Directories:



Здесь следует ввести директории, в которых находятся библиотеки (**Libraries**), файлы конфигурации контроллеров (**Configurations files**) и файлы визуализации (**Visualisation files**). Также нужно указать директории, в которых будут сохраняться файлы компилятора (**Compile files**) (например, map- и list-файлы) и файлы, загруженные из контроллера (**Upload files**).

Директорию можно выбрать с помощью стандартного диалога, который вызывается кнопкой <...>, расположенной справа от поля ввода имени директории. Для библиотек и файлов конфигурации можно задать несколько путей, разделенных точкой с запятой “;”.

Внимание: вы можете указывать относительные пути к библиотекам, начинающиеся от директории проекта. Относительные пути начинаются точкой. Например, “.\libs” соответствует пути 'C:\programs\projects\libs', если текущий проект расположен в директории 'C:\programs\projects'.

Внимание: не используйте пробелы и спецсимволы, за исключением подчеркивания “_” при указании пути.

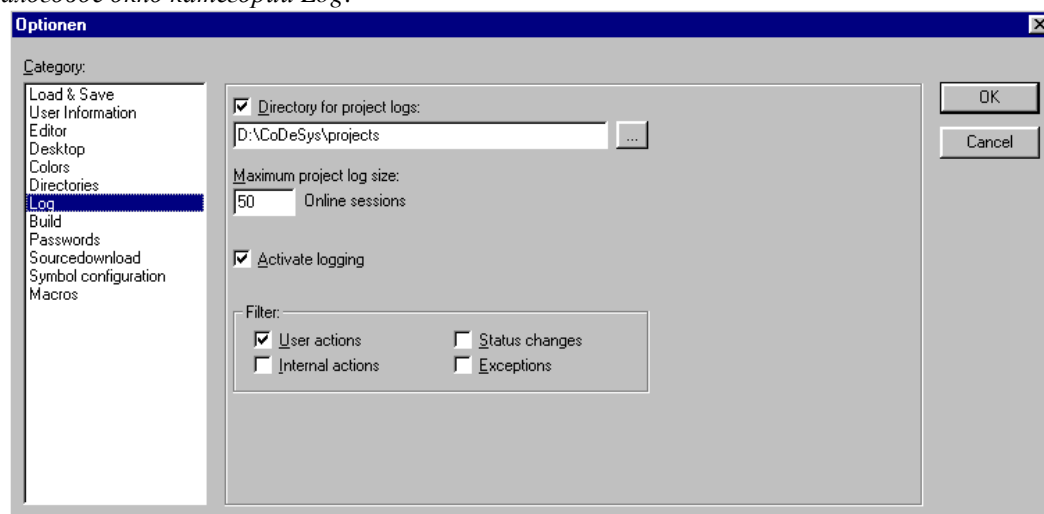
Информация, введенная в разделе **Project**, сохраняется вместе с самим проектом и применима только для него. Установки, сделанные в разделе **Common**, сохраняются в ini –файле CoDeSys и применимы для всех проектов.

В разделе **Target** указываются пути к библиотекам и файлам конфигурации контроллера, которые установлены в целевой системе (файл с расширением **.TNF**). Эти пути нельзя изменять, но можно копировать (с помощью контекстного меню).

В первую очередь **CoDeSys** использует пути в разделе **Project**, затем в разделах **Target** и **Common**. Если файл с одним и тем же именем найден в разных директориях, то используется тот файл, который был найден первым.

Бортжурнал (Log)

Диалоговое окно категории Log:



В этом диалоге вы можете настроить бортжурнал (*.log файл), в который записываются все действия пользователя и действия, выполняемые CoDeSys во время режима **Online**.

Activate logging включает запись в бортжурнал.

Бортжурнал автоматически сохраняется при сохранении проекта в той же директории, что и проект. Если вы хотите, чтобы файл бортжурнала сохранялся в другой директории, выберите опцию **Directory**

for project logs и введите желаемый путь. Для этого удобно использовать диалог, появляющийся при нажатии кнопки <...>.

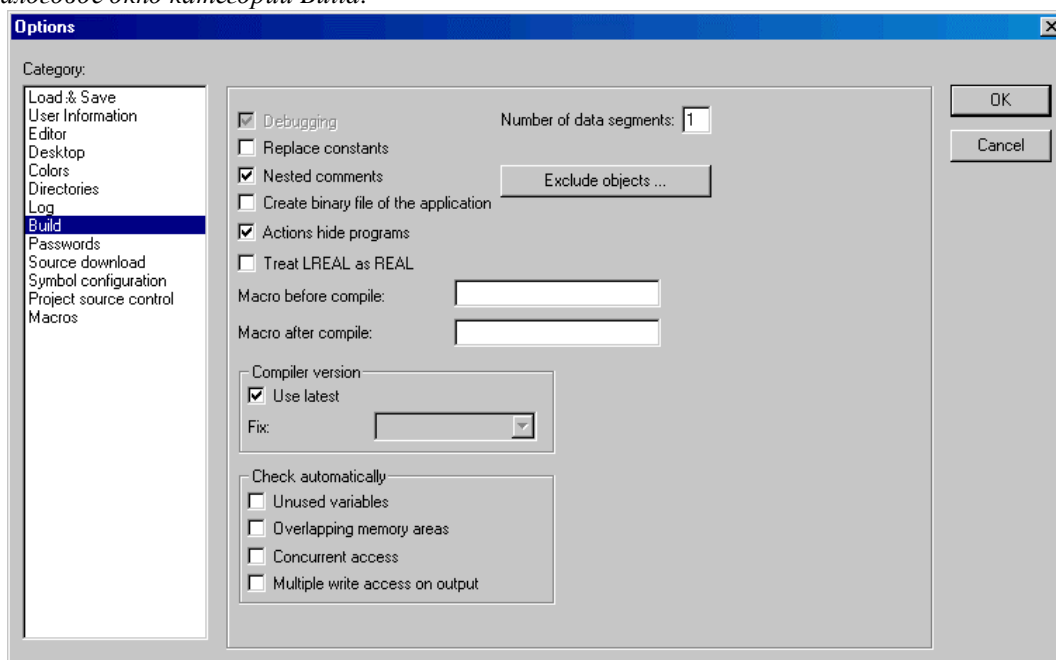
Бортжурнал автоматически получает имя проекта и расширение **.log**. Максимальное число входов в режим **Online**, которые будут описаны в файле, вводится в поле **Online sessions**. Этот показатель определяет размер файла протокола. При превышении этого числа удаляется самая старая запись в файле, освобождая необходимое место.

В области **Filter** можно указать действия, которые нужно записывать в протокол: действия пользователя, внутренние действия, изменения состояния и исключения. Только действия, принадлежащие к выбранным категориям, будут сохраняться в бортжурнале (См. подробнее раздел Бортжурнал).

Окно бортжурнала открывается командой **“Window” “Log”**.

Генератор кода (Build)

Диалоговое окно категории *Build*:



☐ **Debugging** активна в зависимости от установленной целевой платформы. Включает генерацию дополнительного кода для выполнения расширенных функций отладки (например, установка точек останова). Полученный код программы становится ощутимо больше. С выключенной опцией код будет меньше и быстрее. Эта опция сохраняется вместе с проектом.

☐ **Replace constant** разрешает «вшивать» значения констант в машинный код. В режиме Online такие константы изображаются зеленым. Фиксация, запись таких констант невозможна. При отключенной опции константы сохраняются в памяти данных контроллера и с ними можно обращаться так же, как и с обычными переменными (код, естественно, будет медленнее).

☐ **Nested comments** позволяет использовать вложенные комментарии. Например:

```
(*
a := inst.out; (*вложенный комментарий*)
b := b+1;
*)
```

☐ **Create binary file of application** при компиляции будет создан файл, содержащий двоичный код приложения (загружаемый проект). Такой файл имеет имя <имя_проекта>.bin.

• **Actions hide programs:** активируется по умолчанию при создании нового проекта. Опция означает: Если локальное действие имеет имя, совпадающее с именем глобальной переменной или программы, то устанавливается следующая иерархия доступа: локальная переменная, локальное действие, затем глобальная переменная и программа.

Внимание: если открывается существующий проект, созданный в ранних версиях, опция выключается по умолчанию. Устанавливается ранее принятая иерархия доступа: локальная переменная, глобальная переменная, программа, затем локальное действие.

• **Treat LREAL as REAL:** заставляет компилятор использовать тип REAL для LREAL объявлений. Наличие опции зависит от целевой системы, по умолчанию опция выключена. Опция может быть полезна при создании аппаратно независимых проектов.

Число **Number of Data** определяет, сколько сегментов памяти размещается в контроллере под данные. Дополнительное пространство требуется для онлайн коррекции кода. Если во время компиляции появилось сообщение “Out of global data memory...”, увеличьте этот параметр.

Клавиша **Exclude objects** открывает диалог исключения объектов из компиляции (**Exclude objects from build**). Выберите компоненты (POU) которые не нужно компилировать установкой опции **Exclude**. Исключенные POU будут отображаться зеленым цветом. Если хотите отображать только включенные компоненты, нажмите кнопку **Exclude unused**. Отдельный объект, выделенный в Организаторе объектов, можно исключить из компиляции командой '**Exclude from build**' из контекстного меню.

Compiler Version: Здесь вы можете выбрать версию компилятора. Начиная с V2.3.3 (версия, сервис-пак, патч) в CoDeSys, кроме актуальной, устанавливаются и предшествующие версии. По умолчанию, установлен флажок **Use latest**, означающий использование новейшей версии компилятора. Но в этом случае производится контроль версии запущенной системы программирования и компилятора. Если они отличаются, будет использован компилятор, соответствующий версии. Если проект нужно откомпилировать соответствующей версией, выберите ее в списке **Fix**.

Для автоматизации типовых действий, связанных с компиляцией, вы можете использовать два макроса:

- **Macro before compile** выполняется перед компиляцией
- **Macro after compile** выполняется после компиляции

Заметим, что в этих макросах нельзя использовать следующие команды: file new, file open, file close, file save as, file quit, online, project compile, project check, project build, debug, watchlist.

Опции **Check automatically** управляют контролем семантической корректности кода:

- Unused variables: объявленные, но не используемые переменные
- Overlapping memory areas: перекрытие областей памяти
- Concurrent access: одновременный доступ
- Multiple write access on output: многократные присваивания значения выхода

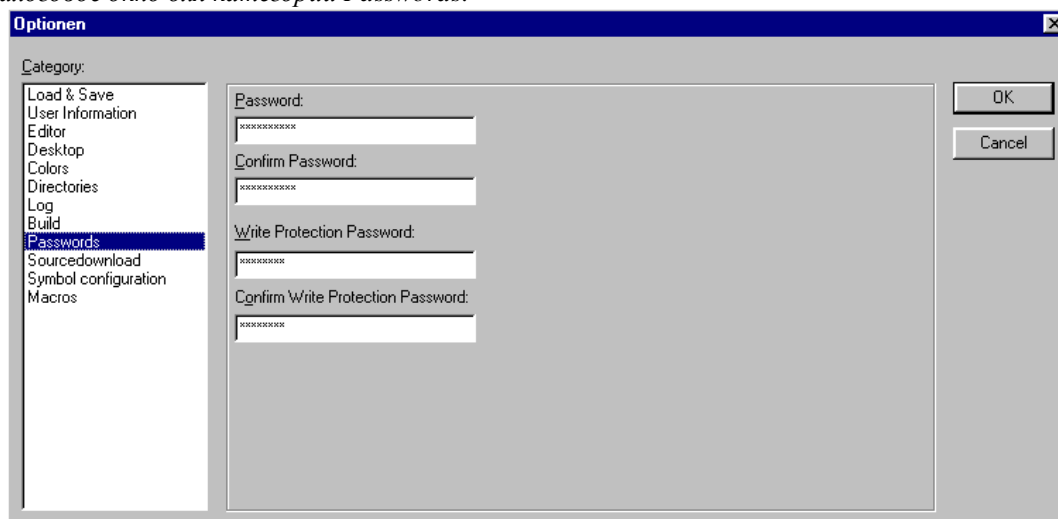
Результаты контроля будут представлены в окне сообщений. Контроль можно запустить отдельно командой 'Check' меню 'Project'.

Негативные результаты проверки могут генерировать ошибки при компиляции, если это поддерживается целевой системой.

Все опции, установленные в этой категории, сохраняются в проекте.

Пароли (Passwords).

Диалоговое окно для категории Passwords:



Для защиты файлов от несанкционированного доступа вы можете установить пароли для открытия файла и пароли разрешения записи.

Введите желаемый пароль в поле **Password**. Теперь вы должны подтвердить введенный пароль в поле **Confirm Password**. Закрывается этот диалог с помощью кнопки **Ok**. Если вы получили сообщение "The password does not agree with the confirmation"(Пароль не подтвержден), значит, вы ошиблись, вводя подтверждение пароля. В этом случае заново ведите пароль и его подтверждение.

Теперь если вы сохраните файл и снова откроете его, то появится диалоговое окно, в котором требуется ввести пароль. Проект будет открыт, если вы ввели правильный пароль. В противном случае будет выведено сообщение "The password is not correct"(неверный пароль).

Вы можете использовать дополнительный пароль разрешения записи. Для этого вы должны ввести пароль в поле **Write Protection Password** и подтвердить его.

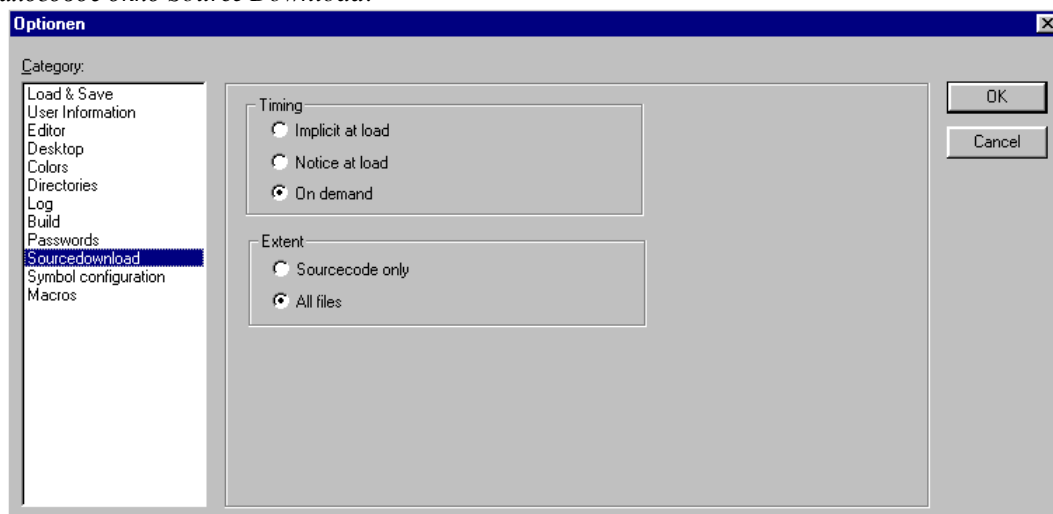
Проект, защищенный таким паролем, можно открыть и без пароля. Для этого при открытии файла, когда CoDeSys потребует пароль, нажмите кнопку **Cancel**. Теперь вы можете компилировать проект, загружать его в контроллер, запускать его, но не изменять.

Пароли сохраняются в проекте.

Для создания отдельных прав доступа к файлам вы можете определить группы пользователей и пароли для них.

Загрузка исходных файлов (Source download)

Диалоговое окно *Source Download*:



Группа опций **Extent** позволяет указать, какие исходные файлы должны загружаться в контроллер.

Опция **Sourcecode only** включает в загрузку только файл проекта. Опция **All files**, кроме того, включает необходимые библиотеки, файлы конфигурации, визуализации и т.д.

Группа опций **Timing** управляет порядком загрузки. Опция **Implicit at load** задает безусловную загрузку исходных файлов по команде "Online" "Download". Опция **Notice at load** приводит к возникновению запроса о необходимости загрузки исходных файлов при загрузке кода. Опция **On demand** приводит к тому, что загрузка исходных файлов будет происходить только по команде "Online" "Sourcecode download".

Проект, сохраненный в памяти ПЛК, можно считать, используя команду "File" "Open" "Open project from PLC". Файл проекта будет считан и распакован.

Символьная конфигурация (Symbol Configuration).

В этой категории Вы можете настроить символьный файл (текстовый файл с расширением *.sym и двоичный с расширением *.sdb). Это необходимо для обмена данными с контроллером через символьный интерфейс, OPC и DDE серверами.

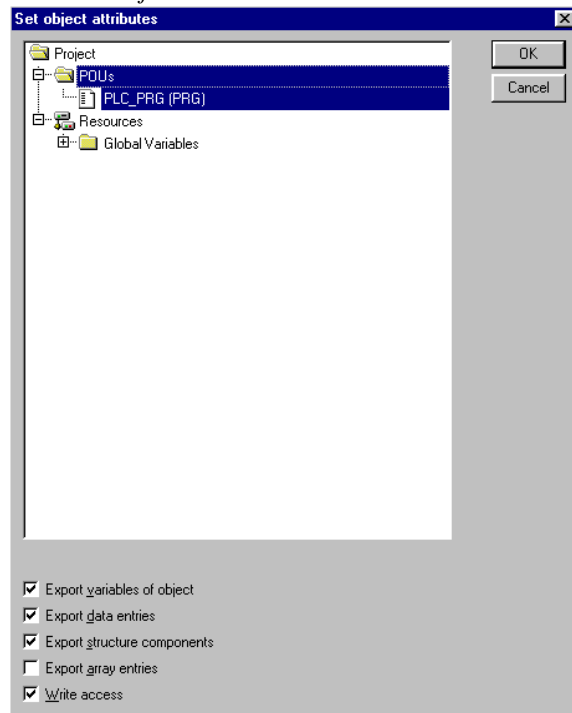
Если выбрана опция **Dumb symbol entries**, то при каждой компиляции в символьном файле будут созданы описания переменных проекта.

Если дополнительно включена опция **Dump XML symbol table**, то в директории проекта будет создан XML файл, содержащий символьную информацию. Он получит название <имя проекта>.SYM_XML.

Обратите внимание: Если в установках целевой платформы (**target settings**) активна опция **Symbol config from INI – file**, то символьная конфигурация будет взята из файла CoDeSys.ini или из другого, определенного для этой цели INI – файла (в этом случае кнопка **“Configure symbol file”** не доступна).

Если опция **Symbol config from INI – file** не активна, то описания переменных в символьном файле будут сделаны в соответствии с установками в диалоге **“Set object attribute”**, который вызывается кнопкой **“Configure symbol file”**.

Диалог “Set object attributes”:



Отметьте в дереве объектов переменные, которые нужно включить в символьный файл. Вы можете выбрать POU (или Global Variables) и целиком включить все его переменные либо выбрать отдельные переменные. Далее в нижней части окна установите нужные опции. Доступны следующие опции:

Export variables of object: Переменные выбранного POU экспортируются в символьный файл. Следующие опции доступны, только если выбрана эта опция.

Export data entries: Создаются описания глобальных переменных и описания структур и массивов выбранного объекта.

Export structure components: Создаются отдельные описания для элементов структур выбранного объекта.

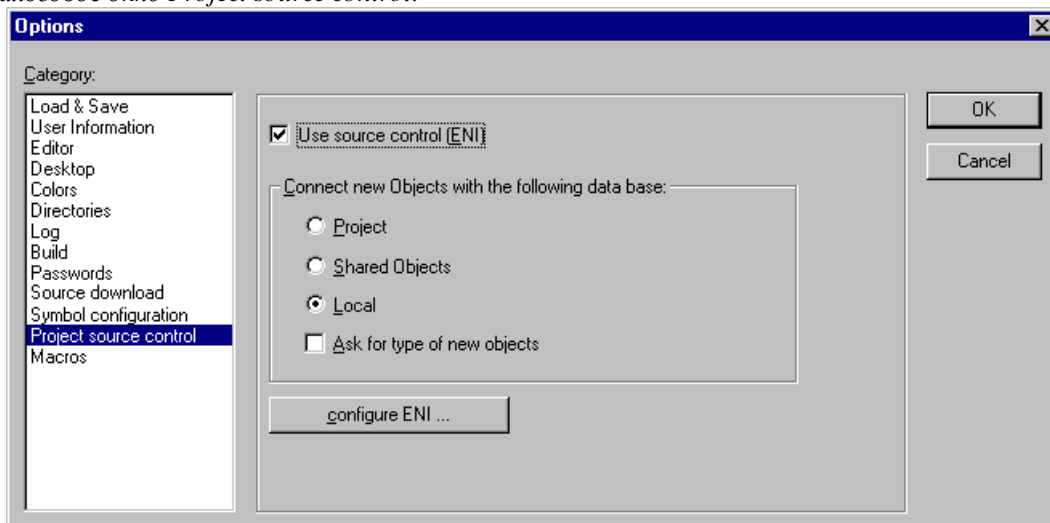
Export array entries: Создаются отдельные описания для элементов массивов выбранного объекта.

Write Access: Переменные выбранного объекта можно изменять через OPC сервер.

Вы можете установить опции для всех POU, не закрывая окно диалога. Сделанные установки сохраняются, как только вы закроете диалоговое окно кнопкой ОК.

Контроль текстов проекта (Project source control)

Диалоговое окно *Project source control*:



Этот диалог включает поддержку управления проектом через ENI интерфейс и соответственно определяет связи с базой данных.

☐ **Use source control (ENI)**: активируя эту опцию, вы включаете доступ к базе данных ENI. Сервер ENI будет управлять всем проектом либо определенными объектами проекта. ENI Server и база данных должны быть установлены заранее. Вы также должны быть зарегистрированным пользователем базы.

Подробнее см. раздел 'ENI'.

При включении данной опции становятся доступными функции базы данных (Check in, Get last version и т.д.) для управления компонентами проекта. Кроме того, некоторые функции будут выполняться автоматически согласно установленным опциям. В меню 'Project' 'Data Base Link' вы найдете команды для явного выполнения этих функций. Страничка 'Data base-connection' будет добавлена в диалог свойств (Properties), где вы сможете сопоставить объектам соответствующие категории базы данных.

Опции **Connect new Objects with the following data base**:

Здесь задаются опции по умолчанию: если в проект добавляется новый объект ('Project' 'Object' 'Add'), он будет автоматически сопоставлен указанной здесь категории. Сопоставление будет отображаться в диалогe свойств объекта ('Project' 'Object' 'Properties') и может быть изменено позднее.

Возможные сопоставления:

- **Project**: POU будет сохраняться в папке, определенной в поле ENI configuration/Project
- **Shared Objects**: POU будет сохраняться в папке, определенной в поле ENI configuration/Shared
- **Local**: POU будет сохраняться локально в проекте без записи в базу данных ENI.

Кроме 'Project objects' и 'Shared objects', существует категория 'Compile files', объекты которой не существуют, пока проект не откомпилирован. Поэтому данная категория здесь не доступна.

☐ **Ask for type of new objects**: включает автоматический вызов диалога 'Object' 'Properties' при вставке нового компонента. Вы сможете установить необходимую категорию индивидуально, не полагаясь на установки по умолчанию.

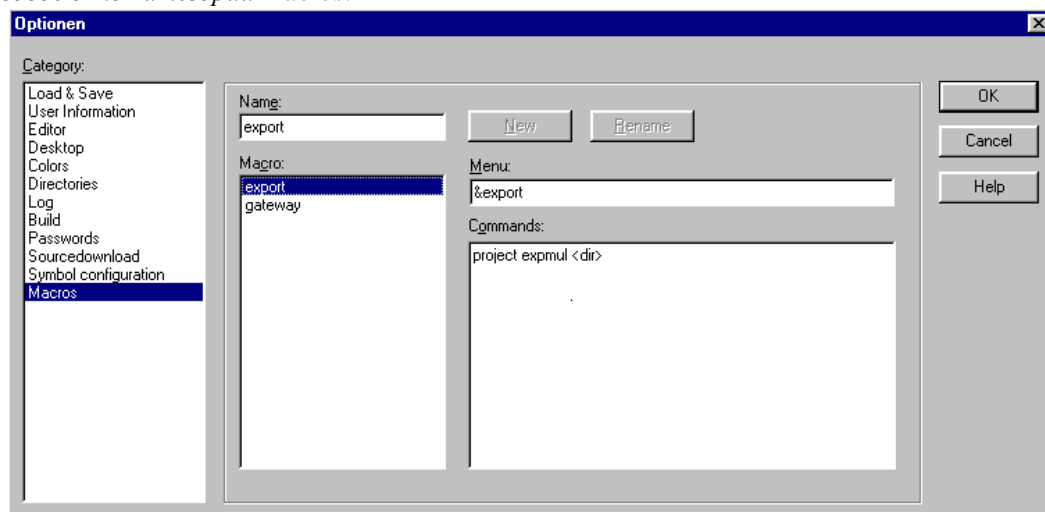
Кнопка **configure ENI** открывает первый диалог конфигурирования ENI:

Каждый компонент проекта, требующий записи в базу данных ENI, может быть сопоставлен одной из трех категорий: 'Project objects', 'Shared objects' или 'Compile files'. Раздел (папка) базы данных для каждой категории задается в отдельном диалоге.

При первой настройке все три диалога открываются последовательно (кнопка **Next** мастера настройки – **Wizard**). Установки первого диалога копируются в следующий, так что вам необходимо выполнить только минимальную коррекцию. Если вы еще не установили соединение с базой, то диалог Login будет открыт автоматически.

Макросы (Macros).

Диалоговое окно категории Macros:



В этом диалоговом окне можно создать макросы (макрокоманды), которые состоят из команд пакетного механизма CoDeSys. Созданные макросы будут добавлены как команды в меню “**Edit**” “**Macros**”.

Для определения нового макроса нужно сделать следующее:

1. Введите имя создаваемого макроса в поле **Name**. После нажатия кнопки **New** это имя помещается в список **Macro**. Список имеет древовидную структуру. Локальные макросы перечислены один за другим. Макро библиотеки (см. ниже) представлены названиями, их содержание можно раскрыть щелчком мыши.
2. В поле **Menu** задайте наименование, которое будет использоваться в качестве пункта меню “**Edit**” “**Macros**”. Для макроса можно определить символ быстрого ввода. Для этого перед соответствующим символом введите “**&**”. Пример: наименование “**Ma&cro 1**”, соответствует пункту меню “**Macro 1**”.
3. В поле **Commands** задайте команды, из которых будет состоять макрос. Здесь доступны все команды пакетного механизма CoDeSys. Вы можете получить список и описание этих команд, нажав кнопку **Help** в панели диалога. Для перехода на новую строку нажмите <Ctrl><Enter>. Контекстное меню (правая клавиша мыши) включает наиболее часто используемые функции текстовых редакторов.
4. Для ввода нескольких макросов повторите шаги 1-3 и закройте диалог кнопкой **OK**.

Выделенный в списке макрос можно удалить, нажав клавишу . Изменить имя макроса можно, нажав кнопку **Rename**.

В меню “**Edit**” “**Macros**” макросы будут расположены в том порядке, в котором они были созданы. Для проверки работы макроса выберите его в меню “**Edit**” “**Macros**”.

Макро библиотеки:

Макросы можно сохранить как библиотеку и использовать в любом другом проекте.

Создание макробиблиотеки на основе текущего проекта: Нажмите кнопку **Create**. В диалоге **Merge project** вы увидите список всех определенных макросов. Отметьте необходимые наименования и нажмите ОК. В следующем диалоге **Save Macro library** задайте имя и директорию для сохранения библиотеки и нажмите кнопку **Save**. Будет создана библиотека <library name>.mac.

Подключение макробиблиотеки к текущему проекту: Нажмите кнопку **Include**. В диалоге **Open Macro library**, задайте соответствующий *.mac файл и нажмите кнопку **Open**. Библиотека будет добавлена в список макросов.

Подсказка: макросы проекта можно экспортировать ('Project' 'Export').

4.3 Управление проектом

Команды управления проектом находятся в пунктах меню “**File**” и “**Project**”. Часть команд доступна через иконки на панели управления.

“File” “New”



Создает новый проект с именем “Untitled”. При сохранении это имя желательно заменить.

“File” “New from template”

Открывает шаблон проекта. Новый проект получает имя “Untitled”.

“File” “Open”



Открывает ранее сохраненный проект. Если в момент вызова этой команды какой-то проект уже открыт и в него были внесены изменения, то CoDeSys спросит, нужно ли сохранить этот проект или нет.

В диалоговом окне открытия проекта вы можете выбрать проект (файл с расширением **.pro**) или библиотеку (файл с расширением **.lib**). Заметим, что с помощью команды “**Open**” нельзя создать новый файл.

Загрузка проекта из контроллера

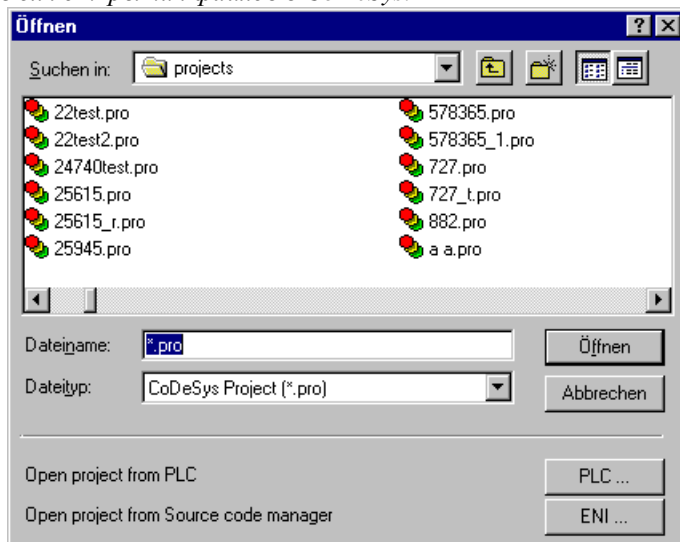
Нажав кнопку PLC, вы сможете загрузить проект из контроллера. Если соединение с контроллером не установлено, то появится диалог **Communication parameters** для настройки параметров связи с контроллером. Как только удалось установить соединение, CoDeSys проверяет, есть ли в текущей директории проект с таким же именем, как и в контроллере. Если такой проект есть, то появится диалог **Load the project from the controller**, в котором вы должны указать, нужно ли заменить файлы на вашем диске на проект из контроллера. (Последовательность действий при выполнении этой команды обратна той, которая выполняется при вызове команды '**Online**' '**Load source code**'. Эта команда сохраняет проект в контроллере. Не пугайте ее с командой '**Create Boot project**!')

Замечание: Примите к сведению, что в случае, если проект в контроллере не назван, то вы должны ввести его имя. В противном случае проект будет сохранен с именем “**Untitled**”.

Замечание: Если это поддерживается целевой системой, то наименование в поле 'Title' диалога **Project info** будет использоваться как начальное имя для файла проекта. В этом случае при загрузке проекта из ПЛК будет открыт диалог записи файла с данным именем, где его можно подтвердить или изменить.

Если проект в контроллере не обнаружен, то будет выведено сообщение об ошибке.

Диалог для открытия файлов в CoDeSys:



Загрузка проекта из ENI базы

Вы должны заранее иметь доступ к базе данных проекта. Нажмите кнопку **ENI**, в следующем диалоге вы подключитесь к категории 'Project objects' базы данных ENI сервера. Задайте соответствующие параметры для доступа (TCP/IP-адрес, порт, имя пользователя, пароль, только чтение) и раздел базы (Project name), в котором находятся необходимые объекты. Нажмите кнопку **Next**. Данный диалог будет закрыт, и открыт следующий, в котором вы вставите данные из категории 'Shared objects'. Нажмите кнопку **Finish**, объекты из заданных разделов будут автоматически считаны и показаны в менеджере объектов CoDeSys. Если вы хотите продолжить сохранение объектов проекта в базе данных, задайте соответствующие параметры в опциях проекта.

Данные доступа сохранены в *codesys.ini* файле, однако имя пользователя и пароль - только если включена опция проекта 'Save ENI credentials'.

Часто открываемые проекты

Список недавно использовавшихся файлов находится в нижней части меню **'File'** под командой **'Exit'**.

Если при записи файла проекта были определены пароли, то CoDeSys потребует ввод пароля при открытии проекта.

“File” “Close”

Закреть открытый в данный момент проект. Если с момента открытия в проект были внесены изменения, то CoDeSys спросит, сохранять его или нет.

Если проект имеет имя “Untitled”, то ему следует дать имя с помощью команды “File” “Save as”.

“File” “Save”

 Быстрый вызов: <Ctrl>+<S>

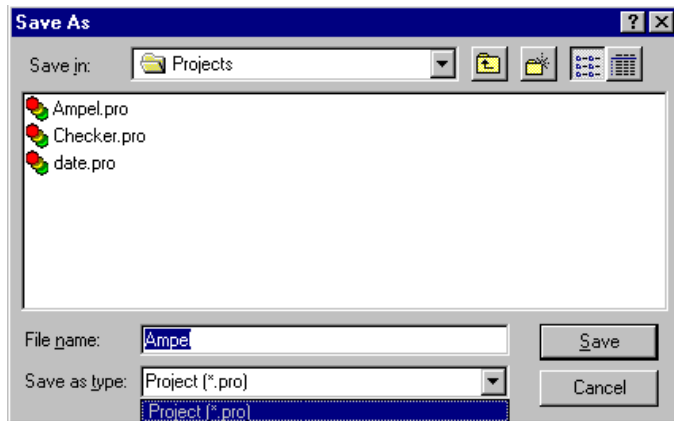
Сохранить проект. Если проект имеет имя “Untitled”, то ему следует дать имя с помощью команды “File” “Save as”.

“File” “Save as”

Эта команда сохраняет проект или библиотеку с новым именем. При этом исходный файл не изменяется.

После выбора этой команды появится диалог сохранения файлов. Задайте имя и тип файла.

Диалог *Save as*:



Проект можно сохранить как проект версии 1.5 (**Project Version 1.5 (*.pro)**), 2.0 (**Project Version 2.0 (*.pro)**), (**Project Version 2.1 (*.pro)**) или 2.2 (**Project Version 2.2 (*.pro)**). При этом данные, свойственные только проекту версии 2.3, будут утеряны. Сохраненные таким образом проекты можно использовать в версиях CoDeSys 1.5, 2.0, 2.1 или 2.2.

Проект можно сохранить как библиотеку, для того чтобы использовать его модули в других проектах. Если код ваших POU целиком создан в CoDeSys, выберите тип файла **Internal library (*.lib)**.

Если вы планируете создать библиотеку модулей проекта на других языках, например на С, выберите тип файла **External library (*.lib)**. В этом случае, кроме библиотеки, создается еще один файл, который имеет имя библиотеки и расширение **.h**. Этот файл представляет собой заголовочный файл языка Си, включающий объявления всех POU, типов данных и глобальных переменных. В режиме эмуляции используется код POU, написанный в CoDeSys. При работе с контроллером выполняется код из внешней библиотеки, написанный на Си.

Лицензирование библиотеки:

Для создания лицензионной библиотеки добавьте соответствующую лицензионную информацию в диалоге 'Edit Licensing Information', который открывается кнопкой **Edit license info....**

Подробнее см. 'Менеджер лицензирования CoDeSys'.

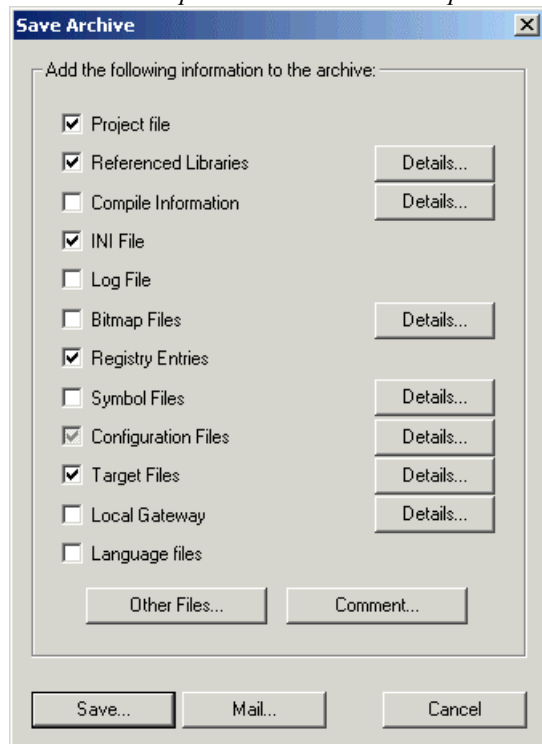
Файл будет сохранен после нажатия кнопки **ОК**. Если файл с таким именем уже существует, то CoDeSys спросит, перезаписать файл или нет.

При сохранении библиотеки она будет скомпилирована. Если при этом будет найдена ошибка, то проект не будет сохранен как библиотека и появится сообщение о том, что проект нужно исправить перед созданием библиотеки.

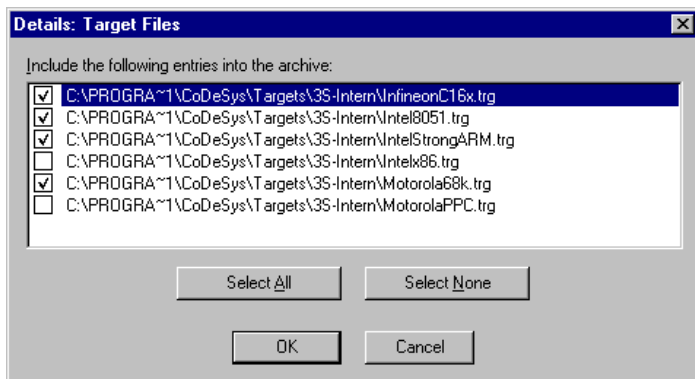
“File” “Save/Mail Archive ”

Создает архив проекта. Все файлы, которые используются проектом CoDeSys, сохраняются и сжимаются в zip файл. Такой файл удобно хранить и пересылать по электронной почте. Также с помощью этой команды можно узнать, какие файлы требуются для правильного функционирования проекта.

Диалоговое окно определения компонент архива:



Здесь вы должны определить, какие категории файлов будут добавлены в архив. Если категория в списке выбрана, то все файлы, относящиеся к этой категории, будут сохранены в архиве. Для того чтобы выбрать отдельный файл, относящийся к категории, нажмите кнопку **Details**. При этом появляется список файлов, входящих в категорию:



В этом диалоге можно использовать кнопку **Select All** для выбора всех файлов в списке и кнопку **Select None** для отмены выбора всех файлов. Можно также выбирать файлы по отдельности.

Для подтверждения сделанных установок нажмите кнопку **Save**.

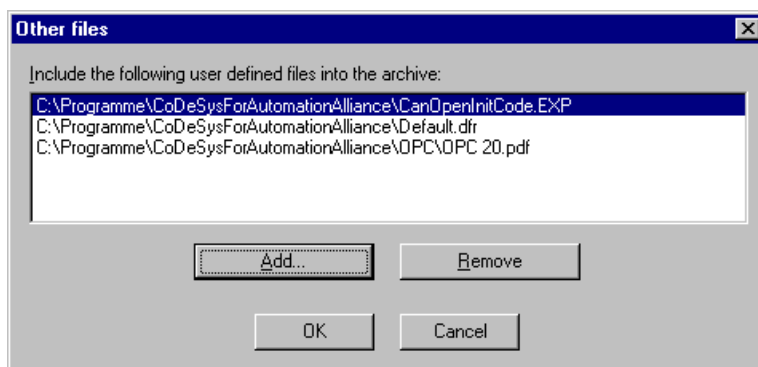
Категории в диалоговом окне "**Save archive**", в которых выбрана только часть файлов, показаны серым цветом.

Ниже показана таблица, в левой части которой находятся возможные категории, а в правой – типы файлов, относящиеся к соответствующим категориям:

Project File	<имя проекта>.pro (файл проекта CoDeSys)
Referenced Libraries	*.lib, *.obj, *.hex (библиотеки и при необходимости объектные и hex-файлы)
Symbol Files	*.sdb, *.sym (файлы, содержащие символьную информацию)

Compile Information	*.ci (файлы, содержащие информацию о компиляции) *.ri (информация о загрузке кода в контроллер и о ссылках) <temp>.*(временные файлы)
Log File	*.log (файл протокола)
INI File	CoDeSys.ini
Configuration files	файлы, используемые для конфигурирования PLC (файлы конфигурации, файлы устройств, пиктограммы и т.д): *.cfg, *.con, *.eds, *.dib,*.ico
Target Files	*.tgr (файлы целевых задач в двоичном формате для всех установленных задач) *.txt (файлы целевых задач в текстовом формате для всех установленных задач)
Registry Entries	Registry.reg (параметры в реестре, используемые CoDeSys, Gateway и PS). Будут добавлены следующие параметры: HKEY_LOCAL_MACHINE\SOFTWARE\3S-Smart Software Solutions HKEY_LOCAL_MACHINE\SOFTWARE\AutomationAlliance“
Bitmap Files	*.bmp (рисунки для POU и для визуализаций)
Local Gateway	Файлы Gateway.exe, GatewayDDE.exe, GClient.dll, GDrvBase.dll, GDrvStd.dll, GHandle.dll, GSymbol.dll, GUtil.dll, и другие DLL используемые программой Gateway.
Language Files	Языковые файлы, используемые для визуализации (*.vis, *.xml)

Для того чтобы добавить какие-либо другие файлы в архив, нажмите кнопку **Other Files**. Будет открыт диалог “**Other files**”, в котором можно составить список нужных вам файлов:



Нажмите кнопку **Add**, чтобы открыть стандартный диалог для выбора файла. После нажатия кнопки “Открыть” выбранный файл будет добавлен в список. Повторите эту процедуру для каждого файла, который нужно добавить в архив. Чтобы удалить файл из списка, выберите его и нажмите кнопку **Remove**. Нажмите кнопку **Ok**, чтобы добавить выбранный список в архив.

В архив можно добавить текстовый файл описания, нажав кнопку **Comment**. Откроется текстовый редактор, в котором вы можете ввести любой текст. Как только вы нажмете кнопку **OK**, в архив будет добавлен файл **readme.txt**. Кроме ваших комментариев, в него будет записана информация о дате создания проекта и версии CoDeSys.

Если вы сделали все необходимые установки, нажмите:

- **Save...** для создания и сохранения архива. Будет открыт стандартный диалог для сохранения файла, и вы должны будете выбрать имя и путь для сохраняемого файла. По умолчанию архив имеет имя

<имя проекта>.zip. Во время создания архива вы будете видеть число сохраняемых файлов, число сохраненных файлов и процент выполнения упаковки.

- **Mail...** для создания временного архива и передачи его по электронной почте. Это функция работает, только если на вашем компьютере установлен MAPI (Messaging Application Programming Interface). В противном случае будет выдано сообщение об ошибке. Если все нормально, то вы увидите точно такое же сообщение, как и при нажатии кнопки **Save...** После того как временный архив создан, будет вызвана программа для работы с E-mail, и вы сможете послать письмо, содержащее созданный архив. Временный архив удаляется автоматически, как только письмо отправлено.
- **Cancel** для того, чтобы отменить создание архива.

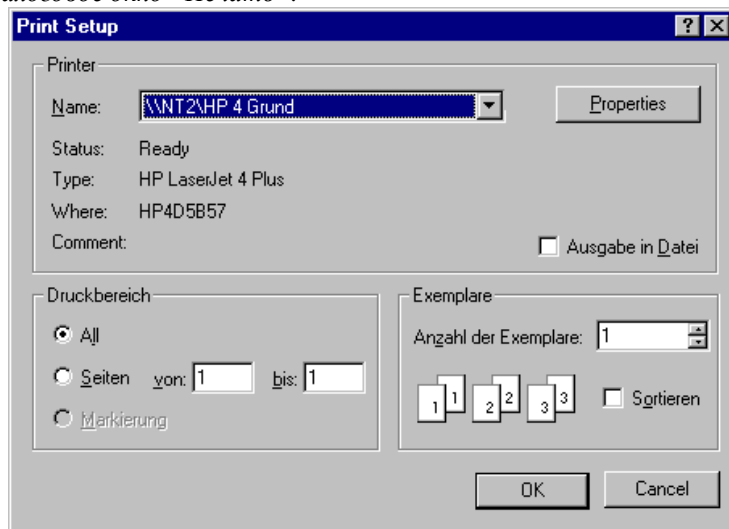
Примечание: После распаковки zip в другой системе может возникнуть необходимость изменить имена директорий!

“File” “Print”

Быстрый вызов: <Ctrl>+<P>

Выводит на принтер содержание активного окна. После вызова этой команды появляется стандартный диалог “Печать”. Выберите необходимые опции, настройте принтер и нажмите кнопку ОК. Содержание активного окна будет распечатано. Цветная печать доступна во всех редакторах.

Диалоговое окно “Печать”:



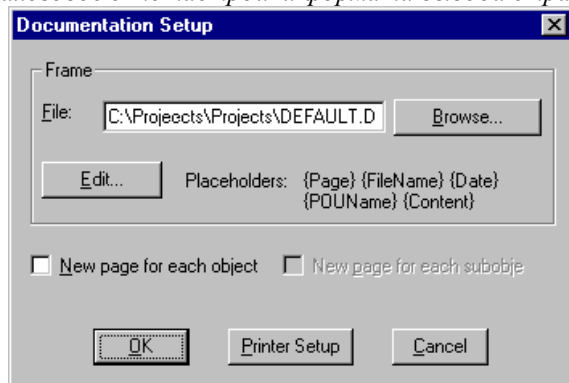
В этом диалоговом окне вы можете выбрать число копий и какие страницы печатать. Формат вывода определяется с помощью команды **Printer Setup**, о которой будет сказано ниже. Во время печати появится диалоговое окно, в котором будет указан номер печатаемой страницы. Если вы закроете это окно, то печать будет приостановлена перед следующей страницей.

Для документирования проекта удобно использовать команду **“Project” “Document”**. Если вы хотите создать документ с комментариями на разных языках, используйте команду **“Extras” “Make docu-frame file”**.

“File” “Printer setup”

Установка формата вывода страниц.

Диалоговое окно настройки формата вывода страниц:

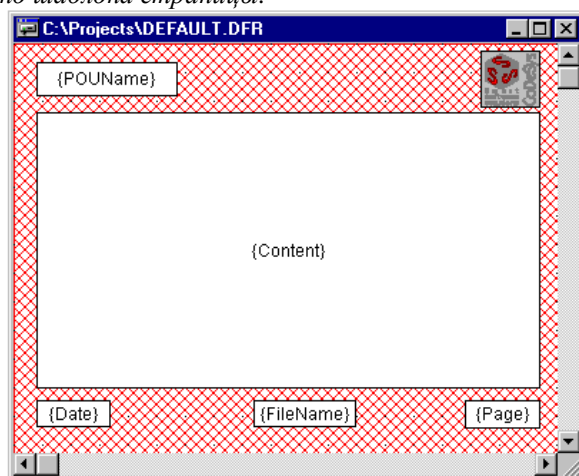


В поле **File** вы можете ввести имя файла с расширением **.dfr**, в котором хранятся установки формата вывода страниц. По умолчанию здесь выбран файл **DEFAULT.DFR**. Можно выбрать требуемый файл, нажав кнопку **Browse**.

Вы можете определить, начинать ли новую страницу для каждого объекта (опция **new page for each object**) и для каждого подобъекта (опция **new page for each subobject**). Используйте кнопку **Printer Setup** для настройки принтера.

Если вы нажмете кнопку **Edit**, то появится окно, в котором вы сможете настроить шаблон страницы. Здесь вы можете установить номера страниц, дату, имя файла и POU, а также разместить на странице рисунок.

Окно шаблона страницы:



В пункте меню "Insert" "Placeholder" вы можете выбрать один из 5 объектов: Page, POU name, File name, Date and Content и расположить их на странице помощью мыши. При выводе на принтер эти объекты будут заменены:

Объект	Шаблон	Результат
Page	{Page}	Номер текущей страницы
POU name	{POU name}	Имя текущего POU
File name	{File name}	Имя проекта
Date	{Date}	Текущая дата
Contents	{Contents}	Содержание POU

С помощью команды **“Insert” “Bitmap”** вы можете вставить рисунок, например логотип вашей компании. После выбора рисунка появится прямоугольник, содержащий этот рисунок. Его можно переместить мышью. Также можно вставлять любые объекты визуализации (см. главу Визуализация).

Если шаблон был изменен, то при закрытии окна CoDeSys спросит, сохранить эти изменения или нет.

Рекомендация: Включите опцию 'Show print area margins' в категории Desktop опций проекта. Это позволит убедиться в том, что заданный формат применим к содержанию редакторов.

“File” “Exit”

Быстрый вызов: <Alt>+<F4>

Закрывает CoDeSys. Если в момент вызова этой команды открыт проект, то вам будет предложено его сохранить, как это описано выше (“File” “Save”).

“Project” “Build”

Быстрый вызов: <F11>

Инкрементальная компиляция проекта. Компилируются только POU, которые были изменены. Перекомпилировать проект целиком можно, предварительно выполнив команду **“Project” “Clean all”**.

В целевых системах, поддерживающих изменения в режиме Online, POU, которые будут загружены в контроллер, после компиляции помечаются синей стрелкой в организаторе объектов.

Если требуется, при соединении с контроллером командой **“Online” “Login”** компиляция выполняется автоматически. Во время компиляции открывается окно сообщения, в котором описывается процесс компиляции и выводятся обнаруженные ошибки и предупреждения. Каждая ошибка и предупреждение имеют уникальный номер. Подробные описания ошибок и предупреждений даны в приложении и в контекстной помощи (<F1>).

Пример сообщений об ошибках и информации компилятора:

```
Interface of POU 'PLC_PRG_TRD'
Error 4024: PLC_PRG_TRD (5): Expecting ';' or '=' before 'bAlarm1'
Error 3781: PLC_PRG_TRD (5): 'END_VAR' or identifier expected
2 Error(s), 0 Warning(s).
```

```
Declarations of the global constants
Declarations of the global library constants
Interface of POU 'PLC_PRG'
Interface of POU 'PLC_PRG_TRD'
Declarations of the global variables
Data allocation
Check task configuration
Implementation of POU 'PLC_PRG'
Implementation of POU 'PLC_PRG_TRD'
Implementation of task 'PLC_PRG_TASK'
Check of the parameter configuration
Hardware-Configuration
POU indices:19 (3%)
Size of used data: 636 of 2097152 bytes (0.03%)
Size of used retain data: 0 of 32768 bytes (0.00%)
0 Error(s), 0 Warning(s).
```

Если выбрана опция **Save before compilation** в категории **Load & Save**, перед компиляцией проект будет сохранен.

Один или несколько объектов, выделенных в Организаторе объектов (Object Organizer), можно исключить из процесса компиляции. Для этого используется команда **'Exclude from build'**, доступная в контекстном меню. Дополнительно можно использовать диалог ('Exclude objects') конфигурации опций компиляции проекта (См. 4.2 Опции проекта, Build).

Замечание: Перекрестные ссылки создаются во время компиляции и сохраняются вместе с информацией о компиляции. Для того, чтобы использовать команды **'Show Call Tree'**, **'Show Cross Reference'**, а также команды **'Unused Variables'**, **'Overlapping memory areas'**, **'Concurrent Access'**, **'Multiple Write Access on output'** в меню **'Project' 'Check'**, нужно сначала скомпилировать проект.

“Project” “Rebuild all”

В отличие от команды **“Project” “Build all”** эта команда компилирует весь проект, даже если он не был изменен. Заметим, что информация о загрузке не удаляется, как это происходит при использовании команды **“Clear All”**.

Относительно исключения объектов из процесса компиляции см. 4.2 (Опции проекта, Build).

“Project” “Clean all”.

Стирает всю информацию о предыдущей компиляции и загрузке проекта в контроллер. После вызова этой команды появляется диалоговое окно, которое сообщает о том, что изменения в режиме Online больше не доступны. В этот момент вы можете подтвердить или отменить команду.

Замечание: После выполнения этой команды работа с проектом в контроллере возможна только в том случае, если файл *.g, с информацией о предыдущей загрузке был сохранен вне директории проекта и

был снова загружен перед соединением с контроллером. Загрузить файл *.gi можно с помощью команды **“Load Download-Information”**.

“Project” “Load Download-Information”

С помощью этой команды вы можете загрузить информацию о загрузке кода в контроллер, если она была сохранена в директории отличной от той, в которой находится проект. После выбора этой команды появится стандартный диалог “Открыть файл”.

При каждой загрузке кода в контроллер эта информация сохраняется в файле с именем <project name><target identifier>.gi в директории проекта. Этот файл открывается вместе с проектом, и при соединении с контроллером он используется для того, чтобы сравнить открытый проект и проект в контроллере. Кроме того, он предназначен для того, чтобы определить, в каких POU проекта код был изменен. В целевых системах, поддерживающих изменения в режиме Online, в контроллер будут загружены только эти POU.

Если вы воспользовались командой **“Project” “Clean all”** файл *.gi в директории проекта будет удален. Но вы можете открыть его с помощью команды **“Project” “Load Download-Information”**, если он был сохранен вами в другой директории.

“Project” “Translate into another language”.

Используется для перевода текстов проекта на другой национальный язык. Здесь используется вспомогательный текстовый файл, созданный в CoDeSys и переведенный в текстовом редакторе на желаемый язык.

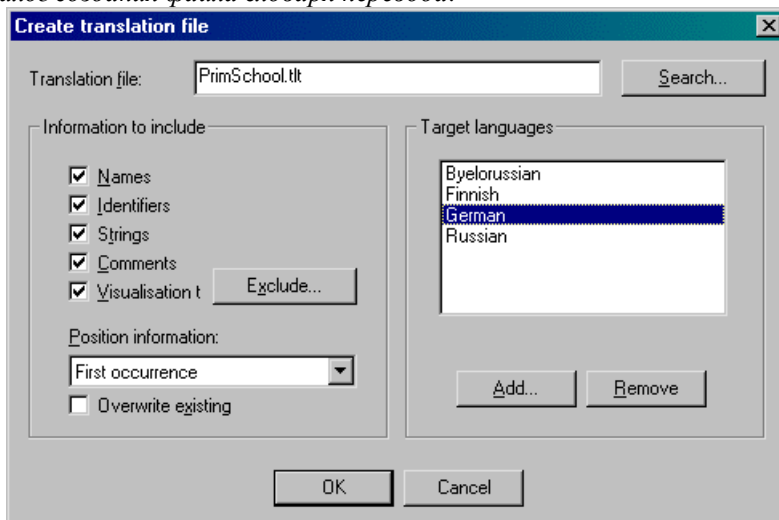
Данное меню распадается на несколько подпунктов:

- Create translation file (создать файл перевода)
- Translate project (перевести проект)
- Show project translated (отображение проекта с переводом)

См. также: 'Редактирование файла перевода'

Создание файла словаря перевода (Create translation file)

Диалог создания файла словаря перевода:



В поле **Translation file** введите путь и имя файла. По умолчанию этот текстовый файл получит имя проекта и расширение .tlt. Вы можете использовать расширение *.txt и применить EXCEL или WORD, поскольку данные удобно редактировать в форме таблицы.

Если файл перевода уже существует, то для его открытия удобно воспользоваться кнопкой **Search**.

В файл перевода будет добавлена следующая информация: имена (Names), например, имена POU, идентификаторы (Identifiers), строки (Strings), комментарии (Comments), текст в визуализации (Visualization texts). Кроме того, сюда добавляется информация о расположении элементов (Position information) в проекте.

Если выбраны соответствующие опции, то эта информация будет сохранена в новом файле или добавлена в уже существующий. Если опция не выбрана, то информация, относящаяся к ней, будет удалена из файла.

Под текстами визуализации понимаются элементы визуализации “Text” и “Tooltip-Text”.

Замечание: Для того чтобы тексты визуализации (элементы визуализации “Text” и “Tooltip-Text”) транслировались в файл перевода, при их вводе в конце и начале текста добавьте символ “#” (например, #text#). Все остальные тексты в визуализациях в файл перевода транслироваться не будут. С помощью команды “Extras” “Settings..” для визуализации можно включить режим перевода в Online . (См. главу «Визуализация»).

Position information – информация о расположении компонентов. Включает в себя имя файла, имя POU и номер строки, в которой находится элемент. В этом разделе есть 3 опции:

None	Информация о расположении компонентов не генерируется
First	Генерируется информация о только о первом компоненте
All	Информация о расположении всех компонентов

Если опция не выбрана, то информация, относящаяся к ней, будет удалена из ранее созданного файла.

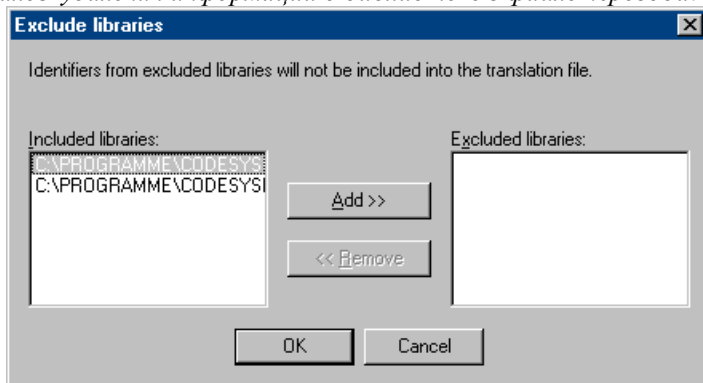
Замечание: Даже если вы выбрали опцию “All”, будет описано не более 64-х позиций одного элемента.

Overwrite existing – перезапись: ранее записанная информация о расположении компонентов будет перезаписана.

Список Target languages: содержит идентификаторы для всех языков, которые находятся в файле перевода.

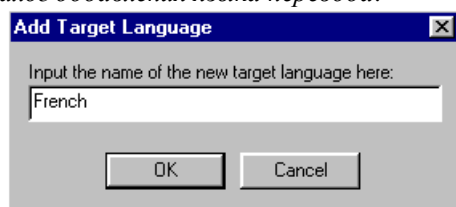
Кнопка **Exclude** открывает диалог “**Exclude libraries**”. Здесь нужно указать библиотеки, используемые в проекте, информация об идентификаторах которых не должна сохраняться в файле перевода. Для того чтобы сделать это выберите библиотеку в таблице **Included libraries**, которая находится в левой части окна, и нажмите кнопку **Add**. Выбранная библиотека появится в таблице **Excluded liberalities**. Чтобы удалить строку в этой таблице, нажмите кнопку **Remove**. При нажатии кнопки **Ok** все изменения сохраняются, и диалоговое окно закрывается.

Диалог удаления информации о библиотеке в файле перевода:



Кнопка Add открывает диалог “Add target Language”.

Диалог добавления языка перевода:



Идентификатор языка нужно ввести в появившемся окне. Он должен быть задан латинским шрифтом и не должен содержать пробелов и непечатаемых символов. Диалог закрывается кнопкой **Ok**, и после этого новый язык становится доступным в списке языков. С помощью кнопки **Remove** язык можно удалить из списка.

Для создания файла перевода нажмите кнопку **Ok**.

Если файл перевода с таким же именем уже существует, то появится сообщение «Указанный файл перевода уже существует. Он будет перезаписан с созданием его резервной копии. Хотите продолжить?» Нажмите кнопку **No** для того чтобы отменить действие, или подтвердите его с помощью кнопки **Yes**. Резервный файл получит имя “Backup_of_<translation file>.xlt”.

При создании файла перевода происходит следующее:

- В каждом новом языке для каждого элемента создается маркер “##TODO”.
- Если файл перевода уже есть, но список языков в нем не соответствует вновь созданному, то все записи для лишних языков удаляются.

Редактирование файла перевода.

Откройте файл словаря перевода в текстовом редакторе. Ключевые слова в файле начинаются с символов ##. Маркер ##TODO в файле надо заменять на текст. Для каждого элемента создается параграф, который начинается ключевым словом ##NAME_ITEM, а заканчивается ключевым словом ##END_NAME_ITEM (для комментариев соответственно ##COMMENT_ITEM и т.д.).

В следующем примере можно увидеть параграф, описывающий имя POU ST_Visualisierung. Здесь показан перевод с немецкого на английский (English) и французский (French) языки. В файл перевода также добавлена информация о расположении элемента.

Шаблон словаря перевода:

```
##NAME_ITEM
[D:\CoDeSys\projects\Bspdt_22.pro::ST_Visualisierung::0]
```

```

ST_Visualisierung
##English :: ##TODO
##French :: ##TODO
##END_NAME_ITEM

```

Готовый словарь перевода:

```

##NAME_ITEM
[D:\CoDeSys\projects\Bspdt_22.pro::ST_Visualisierung:0]
ST_Visualisierung
##English :: ST_Visualization
##French :: ST_Visu
##END_NAME_ITEM

```

Маркер ##TODO заменен на английский и французский эквиваленты.

Заданные идентификаторы должны соответствовать стандарту МЭК, а текстовые строки и комментарии должны быть заключены в кавычки.

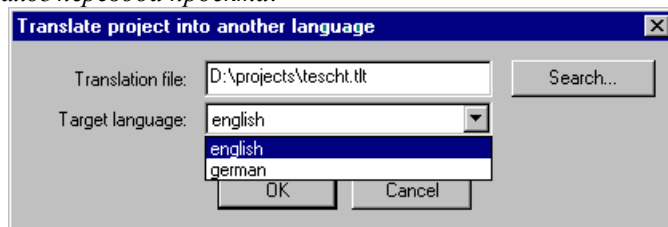
Замечание к русскому переводу: Стандарт не допускает кириллицу в идентификаторах. При необходимости можно применять транслитерацию. В комментариях кириллицу можно использовать без ограничений.

Замечание: Блок Language, блок Flag, информацию о расположении элемента, исходные тексты (например, комментарии) нужно изменять очень осторожно.

Перевести проект (Translate project)

Текущий проект можно перевести на другой язык, используя соответствующий файл словаря перевода: **Translation file**.

Диалог перевода проекта:



Замечание: Если вы хотите в дальнейшем использовать исходный проект, то сохраните его копию под другим именем, т.к. сделанные при переводе изменения не обратимы.

Список **Target language** содержит языки, которые доступны в файле словаря перевода, и вы можете выбрать один из них.

Начните перевод проекта на желаемый язык, нажав кнопку **OK**. Во время перевода выводится окно, в котором описывается процесс перевода, а также возникающие при этом ошибки. Как только перевод закончится, диалоговое окно, а также все открытые окна редактора будут закрыты.

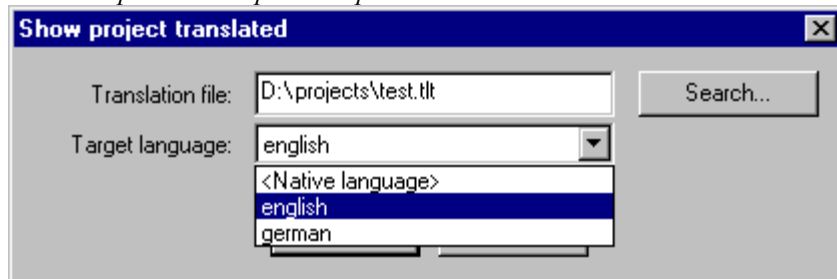
Кнопка **Cancel** отменяет выполнение перевода.

Если файл перевода содержит ошибки, то после нажатия кнопки **OK** будет выведено сообщение об ошибке: путь к файлу перевода, номер строки, в которой произошла ошибка, и ее краткое описание. Например: “[C:\Programs\CoDeSys\projects\visu.tlt (78)]; Translation text expected”.

Отображать перевод (Show project translated)

Если файл словаря перевода уже создан, вы имеете возможность **отображать** проект на экране с заданным переводом, без перезаписи исходной версии проекта. (Вышеописанная команда перевода 'Translate Project' создает отдельную новую версию проекта!)

Диалог отображения перевода проекта:



Задайте необходимый файл словаря перевода в поле **Translation file**. Используйте стандартный диалог открытия файла, кнопка **Browse**.

В поле **Target language** выберите необходимый язык. Строка "<Native language>" соответствует оригиналу проекта. Нажмите ОК. Теперь проект будет отображаться на выбранном языке в режиме **только для чтения!** Для возврата к оригиналу используйте команду '**Switch translation**'.

Переключение перевода (Switch translation)

Если вы просматриваете перевод проекта (См. 'Show project translated'), то вы можете переключаться между просмотром на разных языках и редактированием оригинала. Это делается командой 'Switch translation' из меню 'Project' 'Translate...'

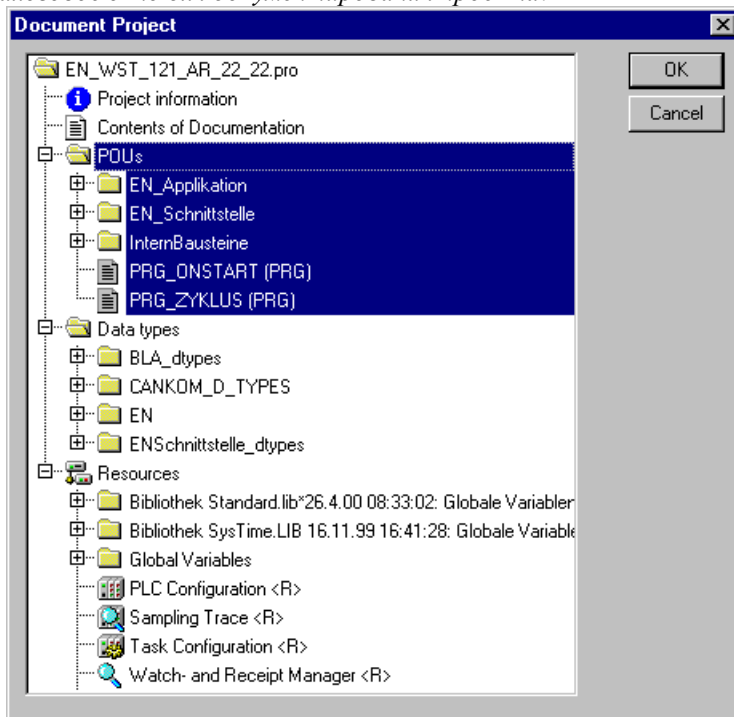
“Project” “Document”

Эта команда позволяет создать печатную версию проекта, в которую могут входить следующие элементы:

- POU
- Содержание документа
- Типы данных
- Визуализации
- Ресурсы: глобальные переменные, переменные конфигурации, трассировка, конфигурация PLC, конфигурация задач, объект мониторинга переменных.
- Дерево вызовов POU.
- Список перекрестных ссылок.

Два последних элемента доступны, если проект скомпилирован без ошибок.

Диалоговое окно для документирования проекта:



Печатаются только элементы, которые вы выбрали.

Если вы хотите выбрать все элементы проекта, выберите имя проекта. Чтобы выделить один элемент проекта, выберите его мышкой или клавишами перемещения. Объект, перед которым находится знак плюс, имеет подобъекты. Щелкнув на знаке плюс, такой объект можно открыть, а на знаке минус – закрыть. При выделении составного объекта все входящие в него объекты также оказываются выделенными. Нажав <Shift> вы можете выделить группу элементов, а <Ctrl> - несколько отдельных элементов.

Как только вы выбрали необходимые элементы, нажмите **OK**. После этого появится стандартный диалог печати. Командой “File” “Printer setup” вы можете настроить формат страниц.

“Project” “Export”

В CoDeSys проекты можно экспортировать и импортировать. Это позволяет переводить программы из одного инструмента МЭК программирования в другой.

Существует стандартный формат файлов обмена для IL, ST и SFC (Common Elements format МЭК).

Для POU на языках LD и FBD CoDeSys использует собственный формат, поскольку стандартный формат не определен. Выбранные объекты сохраняются в текстовом ASCII файле.

Можно экспортировать POU, типы данных, визуализации и другие ресурсы. Также экспортируются описания подключенных к проекту библиотек (но не сами библиотеки).

Важно: Импорт ранее экспортированных POU на FBD и LD выполняется с ошибкой, если комментарии в графическом редакторе содержат символы (‘), которые интерпретируются как начало строки.

Объекты в диалоговом окне выбираются так же, как и при использовании команды “Project” “Document”. С помощью опции **One file for each object** вы можете экспортировать все объекты либо в один файл, либо в разные. Как только вы нажмете кнопку Ok, появится стандартный диалог для сохранения файла. Выберите директорию, если вы экспортируете объекты в разные файлы. В этом случае каждый

объект будет сохранен в файле с именем <имя объекта.exp>. При сохранении одного файла введите его имя.

“Project” “Import”.

В появившемся диалоговом окне выберите импортируемый файл.

Данные из этого файла будут импортированы в проект. Если объект с таким же именем, как в файле, уже существует, то появляется диалоговое окно, в котором вам надо ответить на вопрос: “Do you want to replace it?”(Заменить объект?). Если вы ответите Yes, то объект в проекте будет заменен на объект, импортируемый из файла. Если вы ответите No, то новые объекты будут получать имена из последовательных номеров (“_0”, “_1”, ..). Нажмите Yes, all или No, all для того чтобы выполнить эти действия для всех импортируемых объектов.

Если в файле есть информация о библиотеке, то эта библиотека присоединяется к проекту и помещается в конец списка в менеджере библиотек. В случае, если библиотека уже была соединена с проектом, то заново она не открывается. Однако, если импортируемая библиотека имеет другую дату создания, то она присоединяется к проекту с именем, к которому добавлен символ “*” и дата создания (например, standart.lib*30.3.99 11:20:14). Если импортируемая библиотека не найдена, то выводится сообщение: “Cannot find library {<path>}\<name> <date> <time>”(Не могу найти библиотеку {путь \ имя \ дата \ время}).

“Project” “Siemens Imports”

Здесь вы найдете команды для импортирования переменных и POU из файлов Siemens-STEP5 и STEP7.

Более подробно процесс импортирования описан в приложении .

“Project” “Compare”

Сравнение двух проектов или разных версий одного проекта.

Введение:

Далее мы будем использовать следующие термины:

- “актуальный проект” – открытый проект, с которым вы работаете
- “сравниваемый проект” – проект, который мы сравниваем с актуальным
- “режим сравнения” – режим отображения проекта после выполнения команды Compare. В статусной строке подсвечено слово “COMPARE”.
- “модуль” – минимальный модуль или компонент сравнения. Это может быть строка (в редакторах ST и IL) или цепь (в FBD и LD) или элемент /POU (в CFC и SFC).

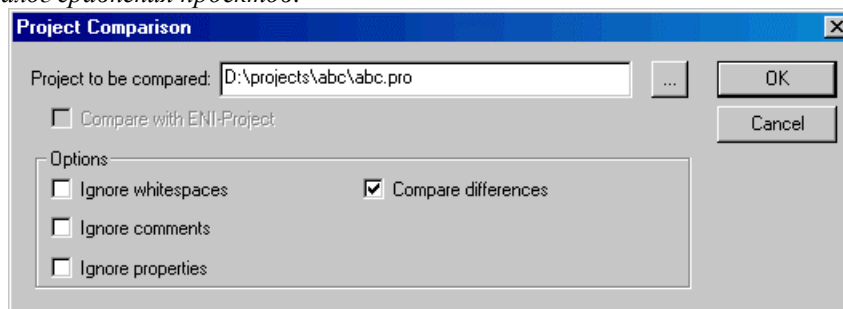
В режиме сравнения актуальный и сравниваемый проект будут открыты в окне, разделенном на две синхронизированные части. Имена POU, в которых найдены различия, помечаются синим. Содержание POU помечается так же. Результаты сравнения и вид их представления зависят от того какие фильтры были задействованы во время сравнения, брались ли в расчет пробелы и комментарии, считать ли модифицированные строки (схемы, элементы) вставленными или нет. Отличия, найденные в сравниваемом проекте, можно принять в актуальном проекте полностью или частично.


Заметьте, что в режиме сравнения проект нельзя редактировать!

Выполнение сравнения:

После вызова команды “Project” “Compare” появится диалог “Project Comparison”.

Диалог сравнения проектов:



Путь к сравниваемому проекту вводится в поле **Project to compare**. При этом удобно использовать кнопку , которая вызывает стандартный диалог открытия файла. Если вы ввели имя актуального проекта, то текущая версия проекта будет сравниваться с ранее сохраненной.

Если проект подключен к управлению версиями в ENI базе данных, то локальная версия проекта будет сравниваться с текущей версией, присутствующей в базе данных. Для этого включите опцию **Compare with ENI-Project**.

Следующие опции влияют на сравнение:

Ignore whitespaces: игнорировать пробелы при сравнении.

Ignore comments: игнорировать комментарии.

Ignore properties: игнорировать свойства объектов.

Compare differences (противопоставлять различия): Если строка, схема или элемент в POU различаются, то они изображаются друг напротив друга в окне сравнения и выделяются красным (смотри ниже). Если опция не активна, то соответствующая строка в сравниваемом проекте изображается как “удаленная”, а в актуальном проекте - как “вставленная” (синий/зеленый, смотри ниже). Это означает, что они не будут изображаться друг против друга.

Пример:

Строка с номером 0005 была изменена в актуальном проекте (левая часть).

(Верхний рисунок – опция *Oppose difference* включена, нижний – выключена)

Option 'Oppose differences' activated:

0001	str_var1:=LREAL_TO_STRING(real_var);	str_var1:=LREAL_TO_STRING(real_var);
0002	boolvar:=TRUE;	boolvar:=TRUE;
0003	count:=count+2;	count:=count+2;
0004		
0005	IF count > 10 THEN	IF count > 20 THEN
0006	switch:=TRUE;	switch:=TRUE;
0007	END_IF;	END_IF;

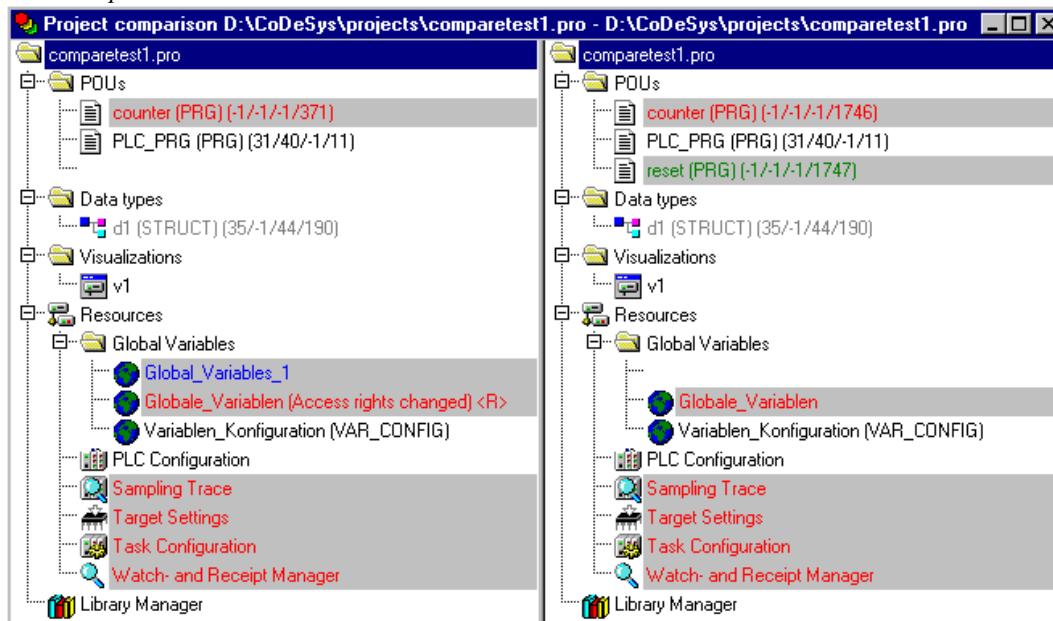
Option 'Oppose differences' not activated:

0001	str_var1:=LREAL_TO_STRING(real_var);	str_var1:=LREAL_TO_STRING(real_var);
0002	boolvar:=TRUE;	boolvar:=TRUE;
0003	count:=count+2;	count:=count+2;
0004		
0005		IF count > 20 THEN
0006	IF count > 10 THEN	
0007	switch:=TRUE;	switch:=TRUE;
0008	END_IF;	END_IF;

Сравнение в соответствии со сделанными установками запускается кнопкой ОК.

Представление результатов сравнения

Сравнение проектов:



1. Сравнение проекта:

В режиме сравнения заголовков окна указывает сравниваемые проекты:

“Project comparison <путь к актуальному проекту> - <путь к сравниваемому проекту>”.

Актуальный проект показан в левой части окна, а сравниваемый - в правой. Здесь проекты представлены в виде дерева. POU, в которых найдены различия, помечаются серым, а цвет шрифта в их именах устанавливается следующим образом:

Красный: Модуль был изменен; выделяются оба модуля.

Синий: Модуль обнаружен только в сравниваемом проекте; в соответствующем месте актуального проекта будет вставлен промежуток.

Зеленый: Модуль обнаружен только в актуальном проекте; в соответствующем месте сравниваемого проекта будет вставлен промежуток.

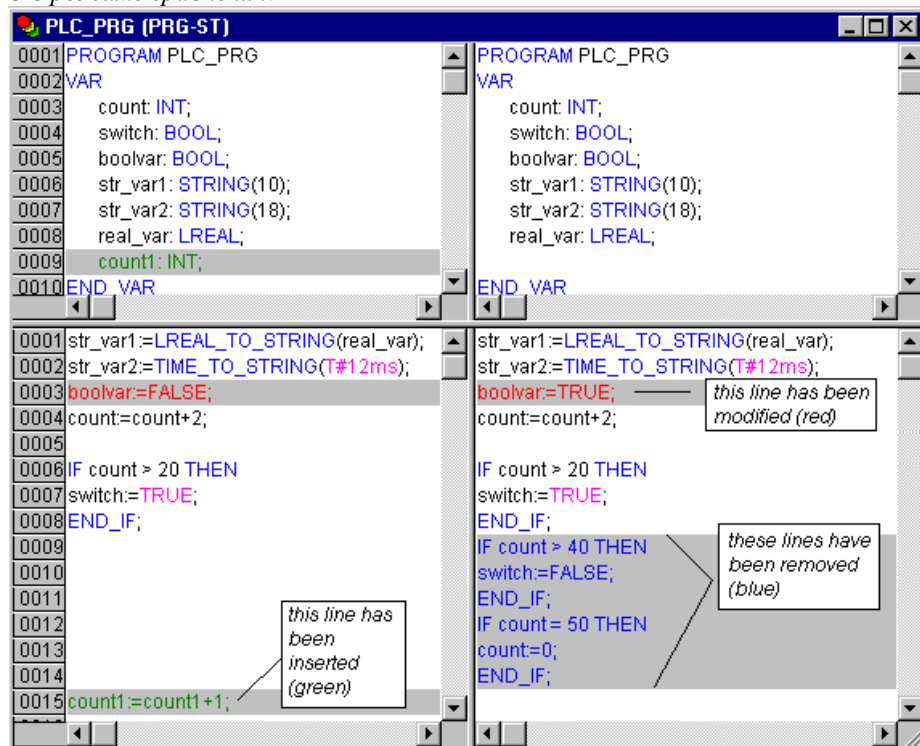
Черный: Модули одинаковые.

К имени POU будет добавлена строка “(Properties changed)”, если различия найдены в свойствах модулей. Строка “(Access right changed)” добавляется в том случае, если различны права доступа для модулей.

2. Сравнение содержания POU

Щелкнув мышкой на POU, который выделен красным в структуре проекта, вы откроете редактор этого POU. Оба POU открываются друг напротив друга. Наименьший сравниваемый модуль - это строка (редактор объявлений, ST,IL), цепь (FBD, LD) и элемент (CFC, SFC). При этом используются цвета, описанные выше.

POU в режиме сравнения:



Если выбран конфигурактор задач целевой системы и т.д., то можно открыть либо объект актуального проекта, либо сравниваемого проекта. Для таких объектов более детальная информация об найденных различиях не выводится.

Работа в режиме сравнения

В зависимости от того, работаете ли вы со структурой проекта или с POU, в контекстном меню и меню “Extras” доступны следующие команды:

Команда	Быстрый ввод	Описание
“Next difference”	<F7>	Курсор перемещается к следующему модулю, в котором найдены различия
“Previous difference”	<Shift><F7>	Курсор перемещается к предыдущему модулю, в котором найдены различия

“Accept change”:	<Space>	Для всех позиций (например, строк), которые имеют один тип разли в актуальный проект копируется версия сравниваемого проекта. Соответствующие строки будут выделены в левой части окна. Если эти строки помечены красным, то после выполнения команды в актуальном проекте они станут желтыми.
“Accept change item”	<Ctrl> <Spacebar>	Производит те же действия, что и предыдущая команда, но действует только на выделенный объект.
“Accept properties”		Свойства выбранного объекта сравниваемого проекта принимаются объектом актуального проекта.
“Accept access right”		(доступна только при работе со структурой проекта): Права доступа выбранного объекта сравниваемого проекта принимаются для объекта актуального проекта.

“Project” “Merge”.

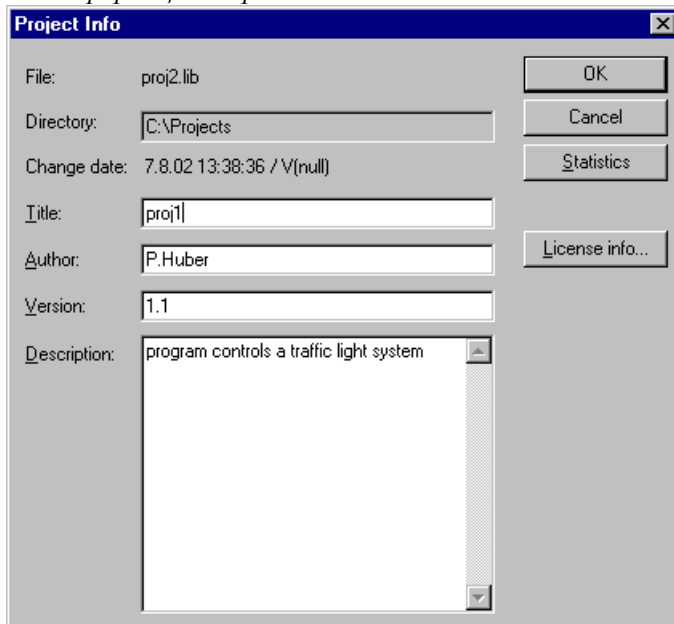
Слить два проекта. При выполнении этой команды появляется диалоговое окно открытия проекта. Когда вы выберете нужный проект, появится диалог для работы с объектами проекта. Принципы его функционирования описаны в “Project” “Document”.

Если при слиянии появляются два объекта с одинаковым именем, то к имени добавляемого объекта присоединяется строка “_1” или “_2” и т.д.

“Project” “Project info”

Сохранить дополнительную информацию о проекте. Выводится диалог, показанный ниже.

Диалог информации о проекте:



В нем содержится следующая информация:

- Имя файла (**File**)
- Имя директории (**Directory**)
- Время последней модификации (**Change date**)

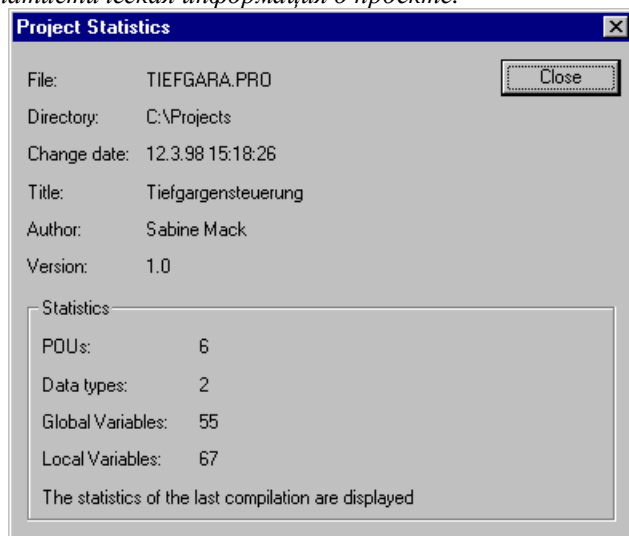
Эту информацию нельзя изменить. Вы можете добавить следующее:

- Заголовок проекта (**Title**)
- Имя автор (**Author**)
- Версию (**Version**)
- Краткое описание проекта (**Description**)

Эта информация не обязательна.

Нажав кнопку **Statistics**, вы получите статистическую информацию о проекте. Она включает число POU, типов данных, локальных и глобальных переменных.

Статистическая информация о проекте:




Кнопка **License info** доступна, если ваш CoDeSys проект уже записан с лицензионной информацией, командой 'File' 'Save as...'. В этом случае кнопка открывает диалог редактирования лицензионной информации (См. 'Управление лицензиями в CoDeSys').

Если выбрана опция Ask for project info в категории Load & Save опций проекта, то при сохранении проекта этот диалог информации будет вызван автоматически.

“Project” “Global Search”

Найти текст в POU, типах данных или разделе глобальных переменных проекта. При вызове этой команды появляется диалоговое окно для работы с объектами проекта (см. “Project” “Document”).

После выбора необходимых объектов появляется стандартное окно поиска. Это же окно появляется при вызове команды  “Global Search” на панели инструментов. В этом случае поиск производится по всему проекту. Строки, которые вы уже искали ранее, доступны в выпадающем списке **Search for**.

Если строка найдена, то открывается соответствующий редактор и искомая строка выделяется. Здесь поиск строк осуществляется так же, как и при использовании команды “Edit” “Search”.

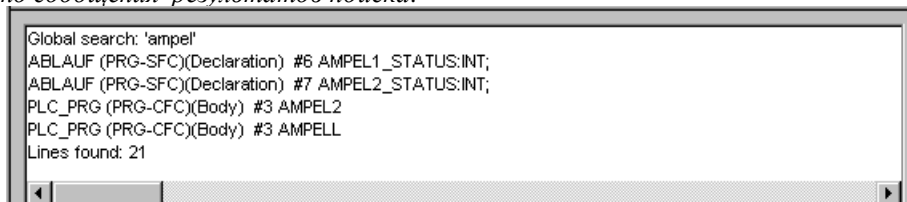
Если вы нажали кнопку **In message window**, то все месторасположения найденной строки будут перечислены в окне сообщений. Кроме того, там выводится количество позиций, в которых эта строка найдена.

Для каждой позиции сообщается:

- Имя объекта
- В какой части объекта найдена строка: в разделе объявлений (Decl) или кода (Impl).

- Номер строки или цепи.
- Сама строка или текстовый элемент в графическом редакторе.

Окно сообщения результатов поиска:



Выбрав соответствующую строку в этом окне, вы откроете POU в том месте, где найдена эта строка. Используя клавиши <F4> и <Shift>+<F4>, можно быстро перемещаться между строками в окне сообщений.

“Project” “Global replace”

Найти заданный текст в POU, типах данных или в глобальных переменных и заменить его другим. Эта команда работает аналогично командам “Project” “Global Search” и “Edit” “Replace”. Поиск по библиотекам, однако, не производится.

Результаты выводятся в окне сообщений.

“Project” “Check”

Команды этого меню используются для дополнительного семантического контроля. Проект должен быть откомпилирован без ошибок, иначе данные команды недоступны. Для получения актуальных данных проект должен быть откомпилирован после любого изменения. Соответствующее предупреждение будет дано в окне сообщений.

Подменю семантического контроля включает следующие команды:

- Unused Variables (не используемые переменные)
- Overlapping memory areas (перекрывающиеся области памяти)
- Concurrent Access (конкуренция доступа)
- Multiple writes to output (множественная запись выходов)

Результаты работы выводятся в окне сообщений.

Внимание: В опциях компилятора вы можете задать автоматическую проверку при каждой компиляции.

Unused Variables

Не используемые переменные. Ищет переменные, которые были объявлены, но ни разу не использовались в проекте. В окне сообщений такие переменные выводятся вместе с именем POU и номером строки, в которой они были объявлены. Например: PLC_PRG(4) – var1. Переменные, объявленные в библиотеках, не проверяются.

Overlapping memory areas

Перекрывающиеся области памяти. Команда проверяет, не перекрывают ли друг друга переменные, объявленные с помощью “AT”. Например, переменные “var1 AT %QB21: INT” и “var2 AT %QD5: DWORD” перекрывают друг друга, так как обе используют 21-й байт области выходов.

В окне сообщений появится следующее сообщение:

%QB21 is referenced by the following variables:

```
PLC_PRG (3): var1 AT %QB21
PLC_PRG (7): var2 AT %QD5
```

Concurrent Access

Конкуренция доступа. Ищет такие области памяти, которые используются сразу в нескольких задачах, но имеют разные виды доступа. Например:

```
%MB28 is referenced in the following tasks :
Task1 – PLC_PRG (6): %MB28 [read-only access]
Task2 – POU1.ACTION2 (1) %MB28 [write access]
```

Multiple writes to output

Множественная запись выходов. Ищет выходы, запись в которые осуществляется более одного раза за управляющий цикл. Например:

```
%QB24 is written to at the following locations:
PLC_PRG (3): %QB24
PLC_PRG.POU1 (8): %QB24
```

Группы пользователей

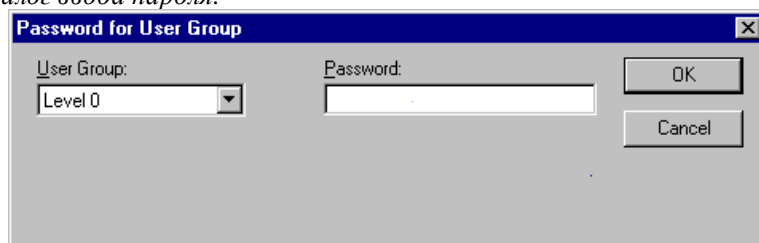
В CoDeSys можно определить до восьми групп пользователей с разными правами доступа к компонентам проекта. Можно установить уровень доступа для каждого объекта в отдельности или для всех сразу. Благодаря этому только член определенной группы пользователей может получить доступ к проекту. Член такой группы пользователей идентифицируется паролем.

Группы пользователей номеруются от 0 до 7. Группа 0 получает права администратора, т.е. только члены этой группы могут назначать пароли для других групп или объектов.

В только что созданном проекте пароли пустые. Пока пароль для группы 0 не задан, каждый пользователь проекта считается членом 0-й группы.

Как только группа 0 получит пароль, можно будет задать пароли для всех остальных групп. Для ввода пароля используется следующий диалог.

Диалог ввода пароля:



В выпадающем меню **User group** выберите номер группы, членом которой вы являетесь, и введите пароль в поле **Password**. Если пароль не верный, то появится сообщение:

"The password is not correct." (не верный пароль)

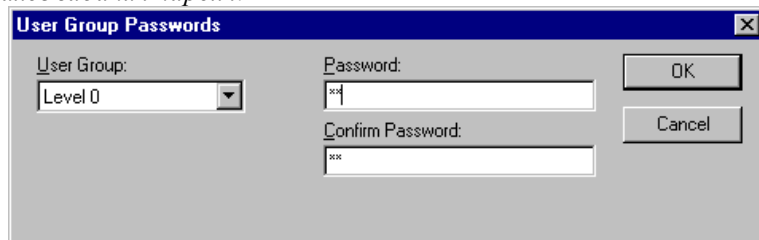
Если вы ввели правильный пароль, то проект будет открыт.

Пароли задаются с помощью команды "Passwords for user group", а права доступа к объектам с помощью команды "Object" "Access rights".

"Project" "Passwords for user group"

Задание паролей групп пользователей. Команда доступна лишь членам группы 0.

Диалог задания пароля:



В выпадающем меню **User group** выберите номер группы и введите пароль для этой группы в поле **Password**. При вводе пароля все символы заменяются звездочками “*”. В поле **Confirm password** вы должны подтвердить пароль. Если при нажатии кнопки **OK** вы увидите сообщение "The password does not agree with the confirmation", то это значит, что вы не правильно подтвердили пароль. В этом случае повторите операцию ввода и подтверждения пароля.

Если надо задать пароль еще для одной группы, вызовите команду снова.

Важно: Если хотя бы одна группа не имеет пароля, то проект можно свободно открыть через эту группу!

“Project” “Data Base Link”

Данное меню доступно, только если включена опция проекта '**Use source control (ENI)**' в категории 'Project source control'. В данное меню входит набор команд управления объектами проекта посредством ENI:

1. Login (подключение к ENI серверу)

Если в **Организаторе объектов** выбран некоторый объект и успешно выполнена команда **Data Base Link** (из контекстного меню, правая клавиша мыши), то для данного объекта становятся доступными ниже перечисленные команды:

2. Define
3. Get Latest Version
4. Check Out
5. Check In
6. Undo Check Out
7. Show differences
8. Show Version History

Если подключение к базе данных не установлено, то диалог 'Data base Login' будет открыт автоматически.

Если команда **Data Base Link** из меню “Project” активна, то становятся доступными дополнительные групповые команды, относящиеся ко всем объектам проекта:

9. Multiple Define
10. Get All Latest Versions
11. Multiple Check Out
12. Multiple Check In
13. Multiple Undo Check Out
14. Project Version History
15. Label Version
16. Add Shared Objects
17. Refresh Status

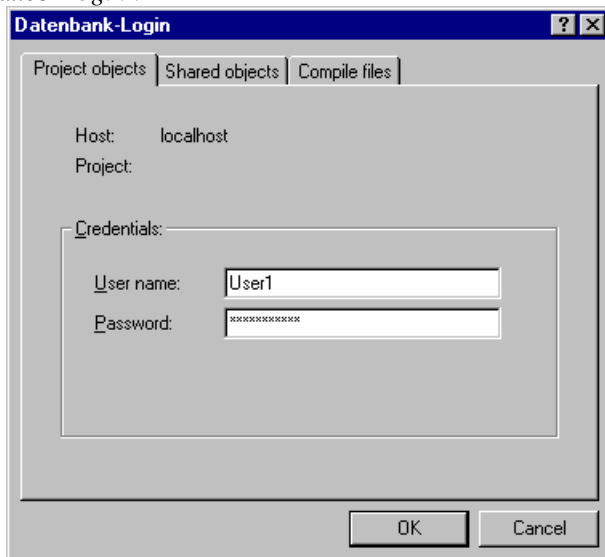
Статус объекта при работе с базой данных отображается в Организаторе объектов :

	<p><u>Иконка закрашена серым:</u> Объект поддерживается в базе данных</p>
	<p><u>Зеленая галочка перед именем объекта:</u> Объект извлечен из базы для редактирования в данном проекте.</p>
	<p><u>Красный x перед именем объекта:</u> Объект извлечен другим пользователем.</p>
	<p><u><R> после имени объекта:</u> Доступ только по чтению.</p>
	<p>Обратите внимание: Некоторые объекты (Task configuration, Sampling Trace, PLC Configuration, Target Settings, Watch- and Receipt Manager) содержат <R> по умолчанию, поскольку не могут быть извлечены. Если вы не имеете права редактировать объект, то вопрос об извлечении не будет задаваться автоматически и команда 'Check out' будет недоступна.</p>

Login (Подключиться)

Команда открывает одноименный диалог 'Login', в котором вы должны задать данные для подключения к базе данных через ENI Сервер. Данные могут отличаться для различных типов базы данных. Настройка самой базы производится в меню ENI Сервера (ENI Admin, User Management).

Диалог 'Login':



Страничка **Project objects** содержит:

Host: имя машины на которой работает ENI Server (определяется в поле 'TCP/IP address' в опциях проекта 'Project source control').

Project: имя базы данных проекта (определяется в поле 'Project name' в опциях проекта 'Project source control' / 'Project Objects').

Удостоверения (Credentials):

- Имя пользователя **User name** и пароль **Password**.
- Когда в проекте активна опция **Use as default**, указанные выше данные будут использоваться по умолчанию автоматически для всех последующих операций с базой данных.
- Нажмите ОК для подтверждения ввода. Далее автоматически будет открыта страничка общих объектов 'Shared objects'. Выберите необходимые объекты и нажмите ОК. Аналогично выполните ввод для странички 'Compile files'.
- Диалог Login вызывается автоматически при попытках доступа к базе данных, если подключение еще не установлено.

Внимание: Если вы хотите сохранить удостоверения доступа в проекте, включите опцию 'Save ENI credentials' опциях проекта, категория 'Load & Save'.

Define (Определить)

Команда: 'Project' 'Data Base Link' 'Define'.

Команда определяет, должен ли выбранный объект поддерживаться в базе данных или только локально в проекте. В предложенном диалоге вы должны указать одну из категорий базы данных 'Project' (объект проекта) или 'Shared objects' (разделяемый объект) либо 'Local' (локальный объект).

Иконки всех разделяемых (поддерживаемых в базе данных) объектов закрашены серым цветом в Организаторе объектов.

Get Latest Version (Считать последнюю версию)

Команда: 'Project' 'Data Base Link' 'Get Latest Version'

Текущая версия выделенного объекта считывается из базы данных, перезаписывая локальную версию. В отличие от команды "Check Out" не блокирует объект для других пользователей базы данных.

Check Out (Извлечь)

Команда: 'Project' 'Data Base Link' 'Check Out'

Выделенный объект извлекается из базы данных и блокируется для других пользователей.

При выполнении этой команды открывается диалог 'Check out object'. Здесь можно задать комментарий, который будет сохранен в истории версий данного объекта. Для вставки перевода строки используйте <Ctrl>+<Enter>.

Если версия объекта отличается от его версии в локальном проекте, будет дано соответствующее сообщение и пользователь сможет решить, нужно ли его извлекать.

Иконки всех извлеченных объектов отмечены зеленой галочкой в Организаторе объектов. Другие пользователи будут видеть его с красным крестом, указывающим на запрет редактирования.

Check In (Поместить)

Команда: 'Project' 'Data Base Link' 'Check In'

Выделенный объект помещается в базу данных как текущая версия. Старые версии не удаляются, оставаясь в базе данных.

При выполнении этой команды открывается диалог 'Check in object'. Здесь можно задать комментарий, который будет сохранен в истории версий данного объекта в базе данных. Для вставки перевода строки используйте <Ctrl>+<Enter>.

После успешного завершения команды зеленая галочка в Организаторе объектов будет убрана.

Undo Check Out (Отменить извлечение)

Команда: 'Project' 'Data Base Link' 'Undo Check Out'

Используется для отмены извлечения выделенного объекта. Все сделанные после извлечения локальные изменения также отменяются. Последняя (до извлечения) версия становится текущей доступной для других пользователей.

Show Differences (Показать различия)

Команда: 'Project' 'Data Base Link' 'Show Differences'

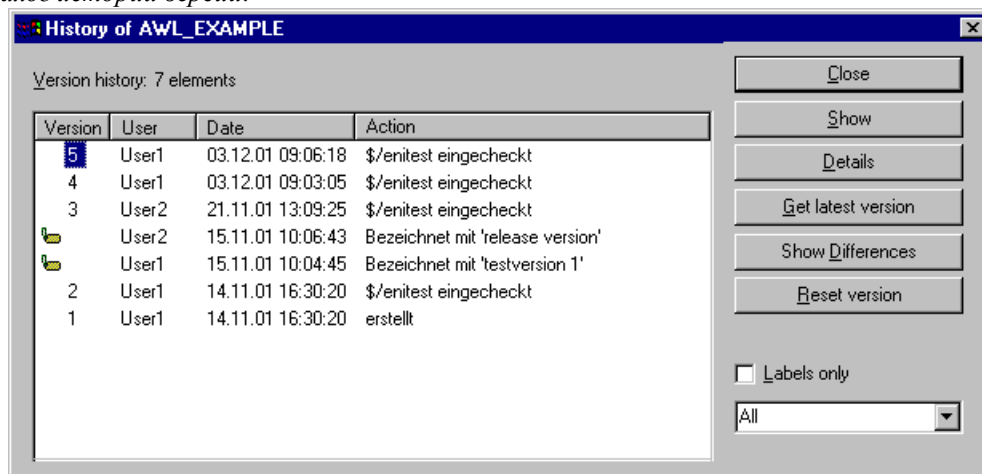
Открытый в настоящий момент объект сравнивается с текущей (последней) версией, сохраненной в базе данных. Окно объекта в CoDeSys разделяется на две части, так же, как и при сравнении проектов (См. 'Project' 'Compare').

Show Version History (Показать историю версий)

Команда: 'Project' 'Data Base Link' 'Show Version History'

Открывает диалог истории версий для объекта, выбранного в Организаторе объектов. Таблица диалога содержит все версии, помещенные в базу, а также маркеры версий:

Диалог истории версий:



Столбцы таблицы содержат:

Version: последовательный номер версии объекта в базе данных. Увеличивается при каждом помещении объекта в базу. Маркировка (label) версии не создает номер, а снабжается специальным значком.

User: имя пользователя, поместившего в базу новую версию или выполнившего маркировку

Date: дата и время действия

Action: тип произведенного действия. Возможные типы: 'created' (объект впервые помещен в базу), 'checked in' (любое помещение объекта в базу, кроме самого первого) и 'labeled with <label>' (маркер присвоен соответствующей версии объекта).

Кнопки:

Close: закрыть диалог.

Display: выбранная в таблице версия будет открыта в окне CoDeSys. Заголовок окна: "ENI: <имя проекта в базе>/<имя объекта>

Details: открывает диалог детальной информации 'Details of Version History':

File (имя проекта и объекта в базе данных), **Version** (см. выше), **Date** (см. выше), **User** (см. выше), **Comment** (комментарий, заданный при помещении новой версии в базу либо при маркировании). Используйте кнопки **Next** и **Previous** для перехода соответственно на следующую и предшествующую записи в таблице диалога истории версий 'Version history of ..'.

Get latest version: последняя версия из таблицы загружается в CoDeSys и замещает локальную версию.

Show Differences: если в таблице выбрана только одна версия, то производится операция сравнения ее с текущей (последней) версией. Если выбраны две версии, то они и будут сравниваться. Отличия отображаются так же, как и при сравнении проектов.

Reset version: выбранная в таблице версия будет считаться текущей (последней). Все версии, помещенные позднее, будут удалены! Команда необходима для возврата к раннему статусу объекта.

Labels only: если включена эта опция, то будут отображаться только маркированные версии.

Selection box работает совместно с опцией 'Labels only': Здесь вы найдете имена всех пользователей, выполнявших действия над объектами данного проекта. Для просмотра истории по всем пользователям выберите строку 'All'.

Multiple Define (Групповое определение)

Команда: 'Project' 'Data Base Link' 'Multiple Define'

По аналогии с командой "Define" данная команда открывает диалог определения категории **'Properties'** для нескольких объектов. После выбора соответствующей категории, будет открыт диалог **'ENI-Selection'**, содержащий все допустимые для данной категории POU. Компоненты представлены в виде древовидной структуры, аналогичной Организатору объектов. Выберите необходимые POU и подтвердите ввод ОК.

Get All Latest Versions (Считать все последние версии)

Команда: 'Project' 'Data Base Link' 'Get All Latest Versions'

Последние версии всех объектов из базы данных замещают локальные версии открытого проекта. Имейте в виду следующее:

- Если за прошедшее время в базе данных проекта были сохранены дополнительные объекты, они будут добавлены в текущий проект CoDeSys.
- Если за прошедшее время некоторые объекты были удалены из базы данных проекта, они не будут удаляться из текущего проекта, но будут автоматически помещены в категорию локальных объектов.

Multiple Check Out (Групповое извлечение объектов)

Команда: 'Project' 'Data Base Link' 'Multiple Check Out'

Позволяет в один прием извлечь несколько объектов. Выберите нужные объекты в диалоге **'ENI-Selection'** и подтвердите выбор кнопкой ОК. Подробности см. в описании команды 'Check Out'.

Multiple Check In (Групповое помещение объектов)

Команда: 'Project' 'Data Base Link' 'Multiple Check In'

Позволяет в один прием поместить несколько объектов. Выберите нужные объекты в диалоге **'ENI-Selection'** и подтвердите выбор кнопкой ОК. Подробности см. в описании команды 'Check In'.

Multiple Undo Check Out (Групповая отмена извлечения)

Команда: 'Project' 'Data Base Link' 'Undo Multiple Check Out'

Позволяет в один прием отменить извлечение нескольких объектов. Выберите нужные объекты в диалоге **'ENI-Selection'** и подтвердите выбор кнопкой ОК. Подробности см. в описании команды 'Undo Check Out'.

Project Version History (История версий проекта)

Команда: 'Project' 'Data Base Link' 'Project Version History'

Позволяет просмотреть историю версий текущего проекта, если выбранный тип базы данных имеет такую возможность.

Команда открывает диалог 'History of <data base project name>'. Он содержит действия (создан, помещен, маркирован) по всем объектам проекта в хронологическом порядке. Общее число объектов указано после надписи **Version history**. Работа с данным диалогом аналогична работе с диалогом команды 'Show Version History'. Обратите внимание:

- 1) Команда 'Reset Version' доступна только для одиночных объектов.

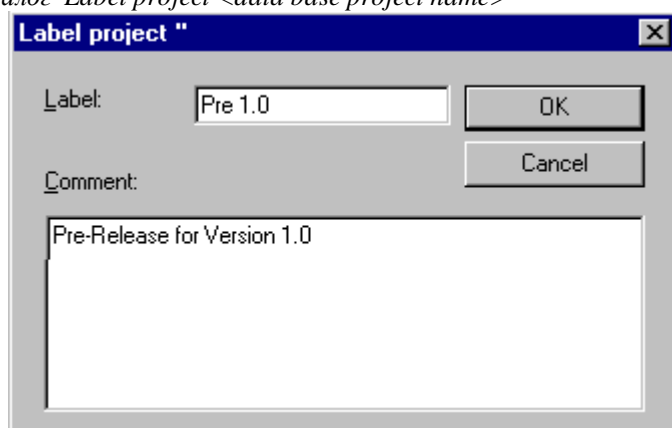
- 2) Команда 'Get latest version' подразумевает, что все объекты выбранной версии будут считаны в локальный проект! Это означает, что объекты в CoDeSys будут перезаписаны старыми версиями. Но: локальные объекты, не входившие в состав старой версии, не будут удалены! Если вызывается маркированная версия, содержащая разделяемые объекты, пользователю будет предложен диалог, позволяющий принять решение о вызове данных объектов.

Label Version (Маркирование версии)

Команда: 'Project' 'Data Base Link' 'Label Version'

Помещает "маркер" (label) на текущие версии всех объектов проекта, так что данный проект может быть в точности восстановлен позднее. В диалоге 'Label <data base project name>' введите имя маркера (**Label**) (например, "Release Version") и, если нужно, комментарий **Comment**. Подтвердите ввод кнопкой ОК. Разделяемые объекты также получают маркеры. Действие маркирования будет сохранено и появится в таблице истории версий. В таблице истории версий объектов маркированные версии объектов получают специальную иконку в колонке 'Version'. При активной опции 'Labels only' будут отображаться только маркированные версии объектов.

Диалог 'Label project <data base project name>'



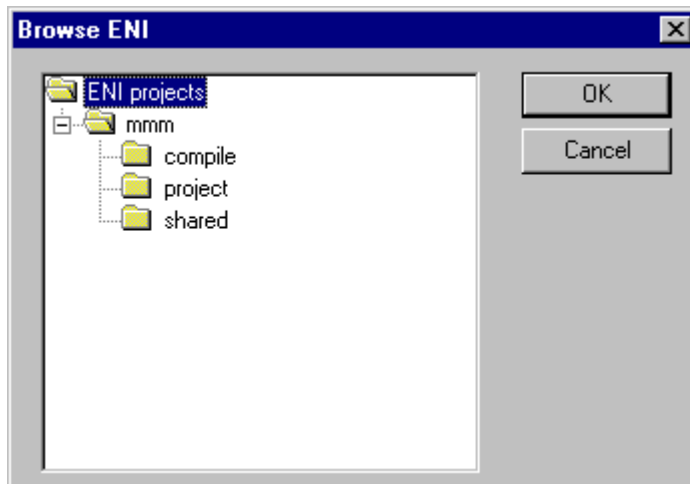
Add Shared Objects (Добавить разделяемые объекты)

Команда: 'Project' 'Data Base Link' 'Add Shared Objects'

Используйте эту команду для явного добавления новых разделяемых объектов 'Shared Objects' в проект. Для объектов категории 'Project Objects' это не нужно, поскольку команда 'Get (all) latest version(s)' автоматически вызывает все объекты из базы данных проекта, даже если они отсутствуют в локальном проекте. Но для объектов категории 'Shared Objects' вызываются только объекты, определенные в проекте.

Выполнение команды 'Add Shared Objects' открывает диалог 'Browse ENI'. Выберите необходимый объект и подтвердите добавление его в открытый проект кнопкой ОК или двойным щелчком мыши.

Диалог 'Browse ENI'



Refresh Status (Обновить статус)

Команда: 'Project' 'Data Base Link' 'Refresh Status'

Обновляет изображение в Организаторе объектов, для отображения актуального статуса связанных с базой данных объектов проекта.

4.4 Управление объектами проекта

В этом разделе приведены сведения о принципах работы с объектами и структуре проекта.

Объект


Слово «объект» используется как общее понятие, включающее программные компоненты (POU), типы данных, визуализации (visualizations), разделы глобальных (global) и конфигурационных переменных (variable configuration), трассировку (Sampling trace), конфигурацию контроллера (PLC configuration), конфигурацию задач (Task Configuration) и менеджер рецептов (Watch and Receipt Manager). Папки используются для структурирования проекта. Все объекты проекта отражены в «Организаторе объектов».

Если вы поместите мышь на POU в организаторе объектов, то всплывающая подсказка укажет его тип (программа, функция или функциональный блок). Для глобальных переменных подсказка включает соответствующее ключевое слово (VAR_GLOBAL и VAR_CONFIG).

Объекты в организаторе объектов можно перетаскивать мышкой (drag&drop). Если при перетаскивании возникает конфликт имен, то к имени нового объекта добавляется серийный номер (*например*, "Object_1").

Папки

Для структурирования больших проектов вы можете объединять POU, типы данных, визуализации и глобальные переменные в папки. Вы можете создать произвольное число вложенных папок.

Если перед папкой стоит знак плюс,  то эта папка содержит объекты или вложенные папки. Открывать и закрывать папки можно, щелкая по знаку "плюс" или "минус" перед ней. Те же функции выполняют команды контекстного меню "Expand node" (открыть папку) и "Collapse node" (закреть папку). Новая папка создается командой "New Folder".

Замечание: Папки не оказывают никакого влияния на программу, а служат исключительно для структурирования проекта.

Пример структуры папок в организаторе объектов:



“New Folder”

Добавляет новую папку в организаторе объектов. Если при вызове этой команды была выделена папка, то новая будет создана в ней. В противном случае она создается на том же уровне вложенности. Если выделено действие, то папка создается на уровне вложенности, к которому принадлежат действия.

Эта команда доступна, когда выбран объект в организаторе объектов, и содержится в контекстном меню, которое вызывается нажатием клавиш **<Shift>+<F10>** или правой кнопкой мыши.

Вновь созданная папка получает имя “New Folder”. Папки можно называть по следующим правилам:

- Папки на одном уровне вложенности должны иметь различные имена. На разных уровнях допускаются одинаковые имена.
- Папка не должна иметь такое же имя, как и какой-нибудь объект на этом уровне.

Если при выполнении команды папка с именем “New Folder” уже существует, то новая папка получит то же имя только с номером, например “New Folder 1”. Нельзя давать папке имя, которое уже используется.

“Expand nodes” и “Collapse nodes”.

С помощью команды “Expand nodes” папку можно открыть. Команда “Collapse nodes” закрывает папку. То же самое можно сделать с помощью мышки.

Эти команды доступны, когда выбран объект в организаторе объектов, и находятся в контекстном меню, которое вызывается нажатием клавиш **<Shift>+<F10>** или правой кнопкой мыши.

“Project” “Object” “Delete”

Быстрый вызов: **<Delete>**

Удаляет выбранный объект или папку со всеми входящими в нее объектами из Организатора объектов. Для надежности вы должны подтвердить удаление. Удаленный объект может быть восстановлен командой **'Edit' 'Undo'**.

Если при удалении объект был открыт в редакторе, то этот редактор будет автоматически закрыт. С помощью команды **“Edit” “Cut”** вы можете переместить объект в буфер.

“Project” “Object” “Add”.

Быстрый вызов: **<Insert>**

Создает новый объект. Тип создаваемого объекта зависит от выбранной в организаторе объектов вкладки. Для объектов типа 'Global Variables', 'Data types', 'Function', 'Function Block' или 'Program' доступно использование шаблонов, см. ниже 'Save as template'.

Имя нового POU определяется в поле **Name of the new POU** диалогового окна. Не забудьте, что имя должно быть уникальным.

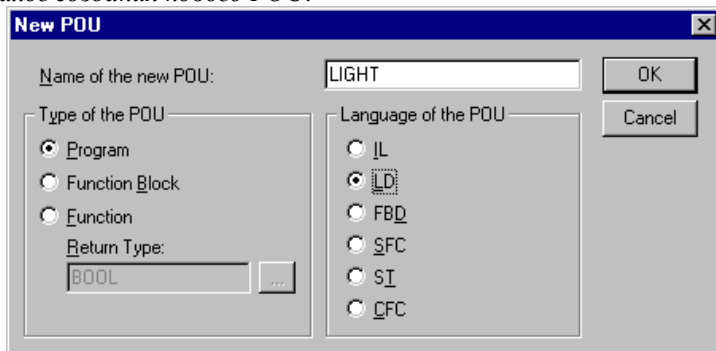
При определении имени придерживайтесь следующих правил:

- Имя не должно содержать пробелов
- Имя нового POU не должно совпадать с именами других POU или типов данных.
- Имя нового типа данных не должно совпадать с именами других POU или типов данных.
- Имя нового списка глобальных переменных не должно совпадать с именами других списков глобальных переменных
- Имя нового действия не должно совпадать с именами других действий в этом же POU.
- Имя новой визуализации не должно совпадать с именами других визуализаций.

Во всех остальных случаях имя будет признано корректным. Поэтому, например, действия, принадлежащие разным POU, могут иметь одинаковые имена. Визуализации могут иметь те же имена, что и POU.

При создании POU в появившемся диалоговом окне нужно задать тип POU (программа, функция или функциональный блок) и язык, на котором этот POU будет выполнен. Тип POU задается в разделе **Type of the POU**, а язык – в разделе **Language of the POU**. Если POU является функцией, то также нужно задать тип возвращаемого значения в поле **Return Type**. Здесь можно вводить любой простой или определяемый пользователем тип (массив, структура, перечисление). При этом удобно пользоваться инструментом **Input assistance**, который вызывается клавишей <F2>.

Диалог создания нового POU:



Если имя POU задано верно, то кнопка ОК становится доступна и, нажав ее, вы создадите новый объект в Организаторе объектов. При этом откроется окно редактора для этого объекта.

При вставке объекта из буфера с помощью команды **“Edit” “Insert”** это диалоговое окно не появляется. Если имя вставляемого объекта конфликтует с имеющимся объектом, то к нему будет добавлен номер (например, “Righ_1”).

Если проект находится под контролем механизма ENI, то, возможно (в зависимости от настроек опций проекта 'Project source control'), вам придется ответить на вопрос о категории, к которой его необходимо отнести. Этот процесс происходит автоматически. Вам нужно только выбрать необходимую категорию базы данных в диалоге 'Properties'.

“Save as template”

Объекты типов 'Global Variables', 'Data types', 'Function', 'Function Block' или 'Program' можно сохранять как шаблоны. Выберите объект в Организаторе объектов и дайте команду **'Save as template'** в контекстном меню (правая клавиша мыши). Теперь каждый новый объект аналогичного типа будет автомати-

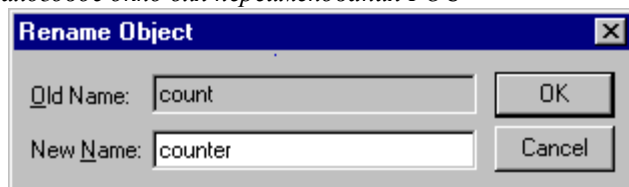
чески включать раздел объявлений из шаблона. Всякий раз используется последний сохраненный шаблон.

“Project” “Object” “Rename”

Быстрый вызов: <Пробел>

Переименовывает выбранный объект или папку. Новое имя должно быть уникальным. Если при переименовании было открыто окно редактирования объекта, то заголовок окна изменится автоматически.

Диалоговое окно для переименования POU



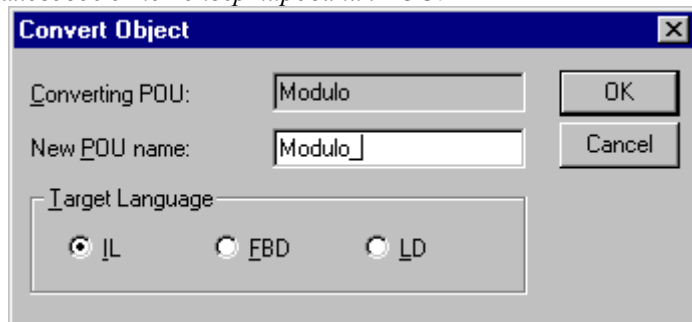
“Project” “Object” “Convert”

Эта команда действует только на POU. Она конвертирует POU с любого языка на один из трех языков IL, FBD и LD.

Перед использованием команды нужно скомпилировать проект. Выберите один из этих трех языков и дайте POU новое имя. Не забудьте, что это имя должно быть уникальным. Теперь нажмите ОК, и новый POU будет добавлен в список POU.

Процесс конвертирования сходен тому, что происходит при компиляции.

Диалоговое окно конвертирования POU.



Внимание: Действия конвертировать нельзя.

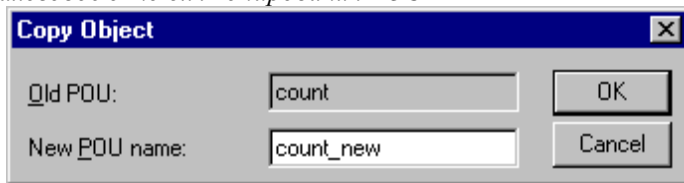
Обратите внимание: POU, созданный в редакторе FBD, можно редактировать в LD без преобразования (см. 'Extras' 'View').

“Project” “Object” “Copy”

При выполнении этой команды объект копируется и сохраняется под новым именем. Имя нового POU введите в диалоговом окне. Не забудьте, что имя должно быть уникальным.

Аналогичную функцию выполняет команда “Edit” “Copy”, но диалоговое окно при этом не появляется.

Диалоговое окно для копирования POU



“Project” “Object” “Open”

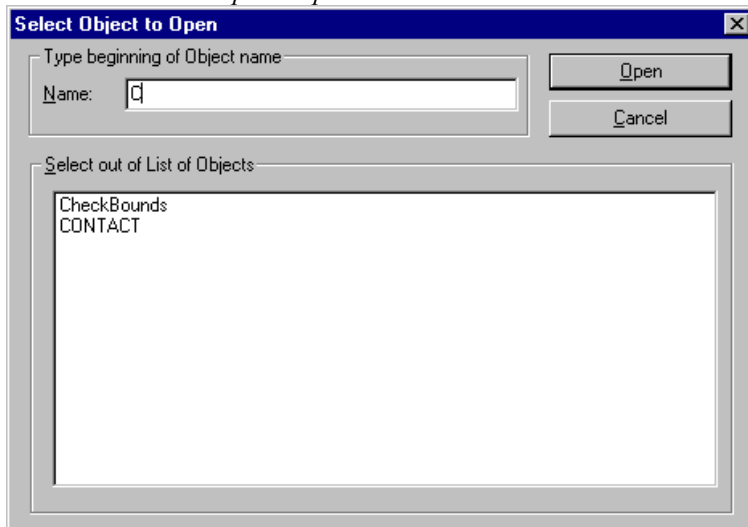
Быстрый ввод: <Enter>

Эта команда открывает выбранный в организаторе объектов POU в соответствующем редакторе. Если окно для этого объекта уже открыто, то оно получает фокус ввода (становится активным).

Есть еще два способа открытия объекта:

- Двойной щелчок мышкой на нужном объекте.
- Введите первую букву имени объекта в Организаторе объектов. Откроется диалоговое окно, в котором будут перечислены все объекты, имя которых начинается с введенной буквы. Выберите объект из списка и нажмите кнопку **Open**. Такой прием удобно использовать в проектах с большим количеством объектов.

Диалоговое окно выбора открываемого объекта

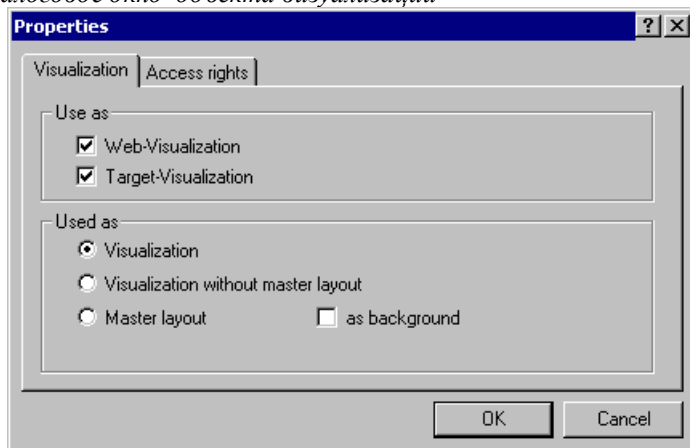


“Project” “Object Properties”

Открывает диалог свойств объекта 'Properties', выбранного в Организаторе объектов. Вид диалога определяется типом объекта:

- для **списков глобальных переменных** будет открыт диалог **Global variable list**. В нем задаются параметры обновления списка и, если возможно, параметры обмена данными посредством сетевых переменных. Этот же диалог появляется и при создании нового списка глобальных переменных. Создание нового списка глобальных переменных выполняется при помощи команды “Add Object”, если выделена папка “Global Variables” или любой уже существующий список глобальных переменных.
- на вкладке '**Visualization**' определяется способ использования объекта визуализации (см. подробнее отдельный документ “Визуализация CoDeSys”).

Диалоговое окно объекта визуализации



Use as: если включены опции **Web visualization** или **Target visualization** в настройках **Target Settings**, то здесь вы задаете включение объекта в Web или Target визуализацию.

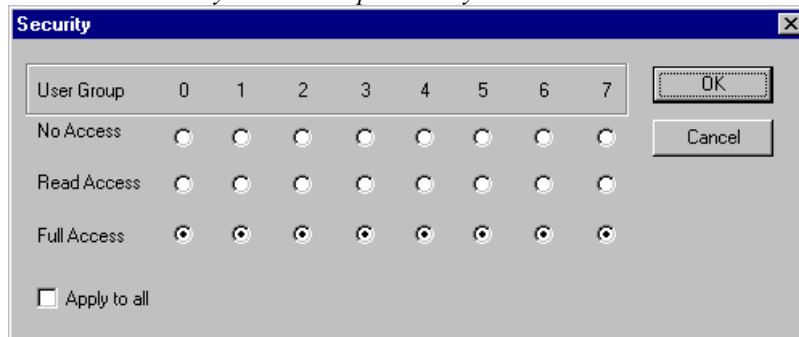
Used as: Активируйте необходимую опцию, относящуюся к возможности использования управляющих панелей (Master layouts):

- **Visualization:** объект использует общие правила.
- **Visualization without Master layout:** если в проекте есть управляющая панель, то она не будет использоваться в данном объекте.
- **Master layout:** данный объект используется как управляющая панель. По умолчанию панель всегда размещается на переднем плане, если не установлена опция **as background**.
- если проект подключен к базе данных **ENI** (см. 'Project' 'Options' 'Project source control'), то будет открыт диалог 'Database-connection'. Здесь вы сможете настроить привязку объекта к необходимой категории базы данных либо к локальной категории 'Local'. (Подробнее см. "Что такое ENI?").

“Project” “Object Access rights”

С помощью этой команды вы можете установить права доступа к объекту для различных групп пользователей. При этом появляется следующее диалоговое окно:

Диалоговое окно для установки прав доступа



Права доступа для групп пользователей могут устанавливать только члены группы 0. Существуют три типа доступа:

- **No Access:** объект не доступен для группы пользователей.
- **Read Access:** объект доступен членам группы только для чтения.
- **Full Access:** группа пользователей может открывать и изменять объект.

Сделанные установки относятся только к выбранному объекту или, если активна опция **Apply to all**, ко всем POU, типам данных, визуализациям и ресурсам проекта.

Права доступа могут устанавливать только члены группы 0.

Обратите внимание на возможность установки прав доступа к операциям объектов визуализации.

“Project” “Add Action”

Используется для создания действия, связанного с выделенным в Организаторе объектов блоком. При этом нужно задать имя действия и язык, на котором оно будет описано.

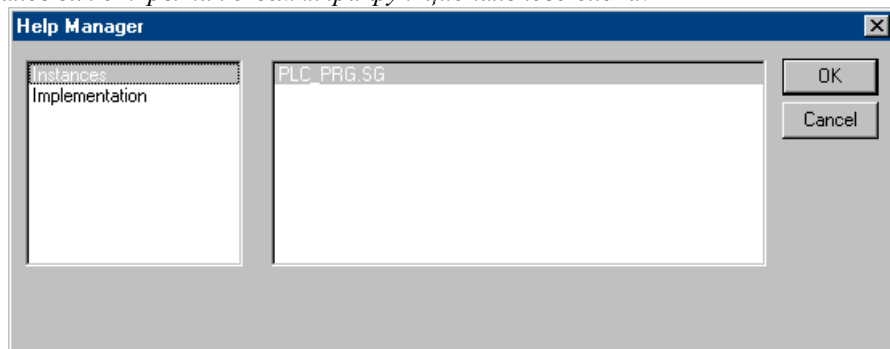
Новое действие будет помещено под выбранным блоком в организаторе объектов. Перед именем этого блока появится знак "плюс". Открывать и закрывать такой блок можно, щелкая по знаку "плюс" или "минус" перед ним. Те же функции выполняют команды контекстного меню **“Expand node”** (открыть блок) и **“Collapse node”** (закрывать блок).

“Project” “View Instance”

Показывает экземпляры выбранного в организаторе объектов функционального блока. К тому же результату приводит двойной щелчок по выбранному функциональному блоку. При этом появляется список всех экземпляров выбранного функционального блока, и его реализация (Implementation). Выберите нужный экземпляр или реализацию и нажмите ОК.

Внимание: Просмотр экземпляров доступен только в режиме Online. (Проект должен быть скомпилирован без ошибок и загружен в контроллер.)

Диалог для открытия экземпляра функционального блока:

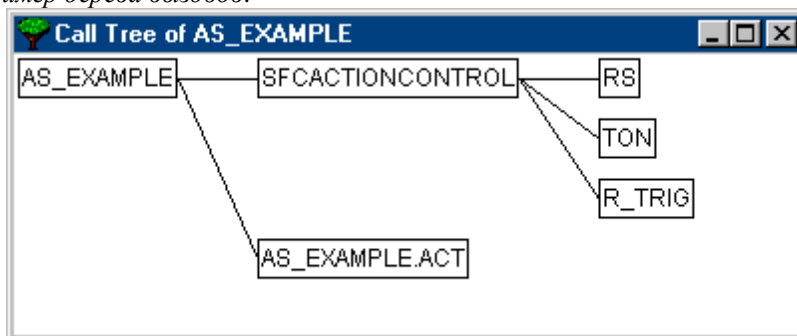


“Project” “Show Call Tree”

Открывает окно, в котором выводится дерево вызовов для выбранного объекта. Для ее использования проект должен быть скомпилирован.

Дерево вызовов содержит вызовы POU и ссылки на типы данных.

Пример дерева вызовов:



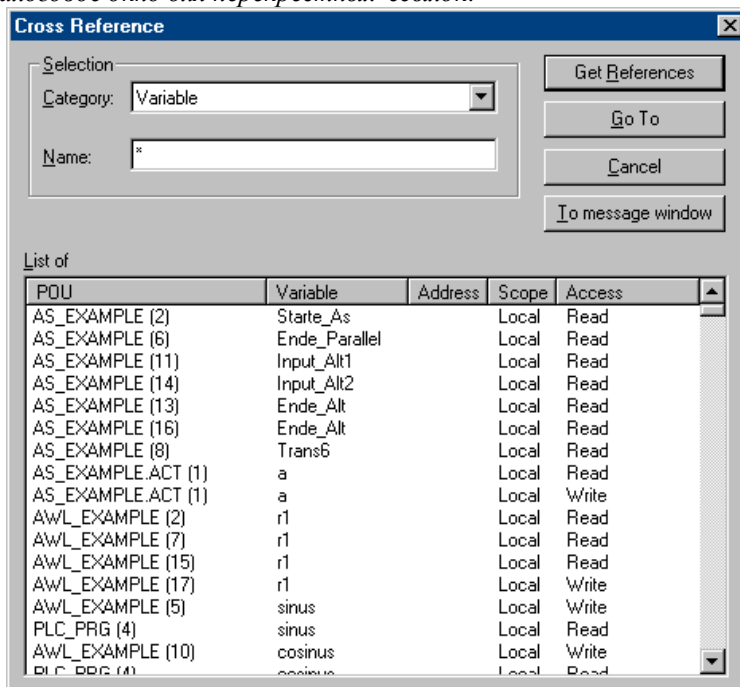
“Project” “Show Cross Reference”.

Открывает диалоговое окно, в котором выводятся адрес, место расположения (POU, номер строки) переменной. Для использования этой команды проект должен быть скомпилирован.

Если проект был изменен после компиляции, то в заголовке окна будет отображено сообщение "Not up to date". Это означает, что измененные связи будут точно соответствовать проекту только после его следующей компиляции.

Выберите категорию “Variable”(переменная), “ Address”(адрес) или “ POU” и в поле **Name** введите имя нужного элемента. Для выбора всех элементов введите символ “*” в поле **Name**.

Диалоговое окно для перекрестных ссылок:



Щелкнув по кнопке **Cross References**, вы получите список всех позиций, где используется переменная.

Для каждой переменной выводится POU, номер строки или схемы, где она расположена, а также адрес, если он задан. В столбце Scope указывается, является ли переменная глобальной или локальной; столбец Access определяет тип доступа к переменной в данной позиции.

Если вы выберете переменную в списке и нажмете кнопку **Go To** или дважды щелкните по ней, то перейдете к той позиции, где она используется. Это очень удобный способ для поиска переменных.

Для того чтобы сделать этот процесс еще проще, нажмите кнопку **Send to message window**, и перекрестные ссылки переместятся в окно сообщений.

4.5 Основные функции редактирования

Все описанные в этой главе команды вы можете использовать в любом редакторе CoDeSys, а некоторые из них даже в Организаторе объектов. Эти команды находятся в меню “Edit” и в контекстном меню.

Если на ваш компьютер установлен соответствующий драйвер, то CoDeSys поддерживает мышь со скроллингом. Во всех редакторах можно менять размер изображения, нажав <Ctrl> и вращая колесико мыши.

“Edit” “Undo”

Быстрый вызов: <Ctrl>+<Z>

Отменяет последнее изменение, сделанное в открытом редакторе или в Организаторе объектов. Используя эту команду, вы можете отменить все изменения, выполненные после открытия окна. Команда может отменять любые изменения, которые были выполнены в редакторах POU, типов данных, визуализаций, глобальных переменных и в организаторе объектов.

С помощью команды “Edit” “Redo” вы можете вернуть отмененные изменения.

Замечание: Команды Undo и Redo действуют только в активном окне. Каждое окно имеет свой список изменений. Если вы хотите отменить изменение в каком-нибудь окне, сделайте его активным. Команды работают и в Организаторе объектов, когда курсор находится в нем.

“Edit” “Redo”


Быстрый вызов: <Ctrl>+<Y>

Возвращает последнее отмененное в открытом редакторе или в Организаторе объектов изменение.

Отменяя изменение командой “Undo”, вы всегда можете его вернуть командой “Redo”.

Замечание: Команды Undo и Redo действуют только в активном окне. Каждое окно имеет свой список изменений. Если вы хотите отменить изменение в каком-нибудь окне, сделайте его активным. Команды работают и в Организаторе объектов, когда курсор находится в нем.

“Edit” “Cut”

Обозначение:  Быстрый вызов: <Ctrl>+<X> или <Shift>+<Delete>

Перемещает выделенный элемент в буфер. При этом выделенный элемент удаляется из редактора.

Аналогично команда работает с объектами в Организаторе объектов, но некоторые объекты нельзя удалить, например, объект PLC Configuration.

Не забудьте, что не все редакторы поддерживают эту команду и в некоторых редакторах ее использование ограничено.

Выделяемые элементы различны для разных редакторов.

В текстовых редакторах IL и ST, а также в редакторе объявлений выделяемый элемент – это текст.

В редакторах FBD и LD – это цепи, блоки и операнды.

В редакторе SFC выделяемые элементы – это последовательность шагов и переходов.

Для того чтобы вставить содержимое буфера, используйте команду **“Edit” “Paste”**. В SFC для этой цели также предназначены команды **“Extras” “Insert parallel branch (right)”** и **“Extras” “Paste after”**.

Для того чтобы скопировать выделенный элемент без удаления, используйте команду **“Edit” “Copy”**.

Команда **“Edit” “Delete”** предназначена для удаления без копирования.

“Edit” “Copy”

Обозначение:  Быстрый вызов: <Ctrl>+<C>

Копирует выделенный элемент в буфер, при этом не изменяя содержимое окна редактора.

Аналогично команда работает с объектами в Организаторе объектов, но некоторые объекты нельзя скопировать, например, объект PLC Configuration.

Не забудьте, что не все редакторы поддерживают эту команду и в некоторых редакторах ее использование ограничено.

Выделяемые элементы различны для разных редакторов, как и при использовании команды **“Edit” “Cut”**.

“Edit” “Paste”

Обозначение:  Быстрый вызов: <Ctrl>+<V>

Вставляет содержимое буфера, начиная с текущей позиции курсора в окне редактора. В графических редакторах команда выполняется только тогда, когда содержимое буфера соответствует выбранному элементу.

Если курсор находится в Организаторе объектов, то из буфера вставляется объект.

Не забудьте, что не все редакторы поддерживают эту команду и в некоторых редакторах ее использование ограничено. Выделяемые элементы различны для разных редакторов.

В редакторах FBD и LD текущая позиция – это схема, и при выполнении команды **“Edit” “Paste”** содержимое буфера вставляется перед выбранной схемой.

В редакторе SFC в зависимости от выбранного элемента и содержимого буфера вставка происходит либо перед выбранным элементом, либо в новую левую ветвь (параллельную или альтернативную).

“Edit” “Delete”

Быстрый вызов:

Удаляет выбранную область, при этом не изменяя содержимое буфера.

Эта команда работает и с объектами в Организаторе объектов, но некоторые объекты нельзя удалить, например, объект PLC Configuration.

Не забудьте, что не все редакторы поддерживают эту команду и в некоторых редакторах ее использование ограничено. Выделяемые элементы различны для разных редакторов, как и при использовании команды “**Edit**” “**Cut**”.

В менеджере библиотек выделяемый объект – это имя библиотеки.

“**Edit**” “**Find**”

Обозначение: 

С помощью этой команды вы можете найти заданный текст в активном окне редактора. При этом открывается диалог для поиска. Он будет оставаться открытым, пока вы не нажмете кнопку **Cancel**.

В поле **Find what** введите искомую строку.

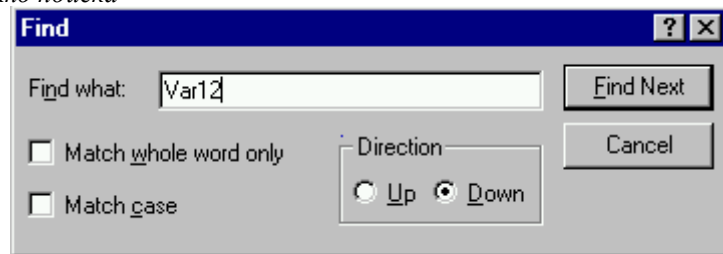
В этом окне вы можете указать, искать ли строку целиком (опция **Match whole word only**) или нет, учитывать ли регистр и направление поиска – вверх (**Up**) или вниз (**Down**).

Кнопка **Find next** начинает поиск с текущей позиции в заданном направлении. Найденная строка выделяется. Если строка не найдена, то будет выведено соответствующее сообщение. Поиск можно повторить несколько раз, до тех пор пока не будет достигнут конец или начало документа.

Учтите, что найденный текст может быть закрыт окном поиска.

В редакторе CFC последовательность поиска определяется геометрическим порядком размещения. Поиск начинается с верхнего левого угла вправо и вниз. Обратите внимание, что FBD компоненты обрабатываются справа налево!

Окно поиска



“**Edit**” “**Find**” “**Next**”

Обозначение:  Быстрый вызов: <F3>

Повторный поиск с теми же параметрами, как и в предыдущий раз.

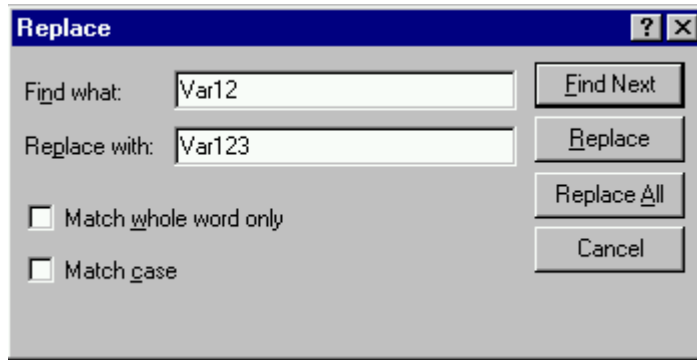
“**Edit**” “**Replace**”

Ищет заданную строку и заменяет ее на другую. После вызова этой команды открывается диалог для поиска и замены.

Кнопка **Replace** заменяет выделенный текст на текст в поле **Replace with**.

С помощью кнопки **Replace all** можно заменить все найденные строки сразу. По окончании обработки будет дано сообщение о числе выполненных замен.

Диалоговое окно для поиска и замены



“Edit” “Input Assistant”

Быстрый вызов: <F2>

Ассистент ввода: диалоговое окно для выбора элемента, который можно ввести в текущей позиции. В левом столбце выберите категорию элементов, а в правом столбце – нужный элемент и подтвердите свой выбор нажатием кнопки ОК. Выбранный элемент появится в текущей позиции.

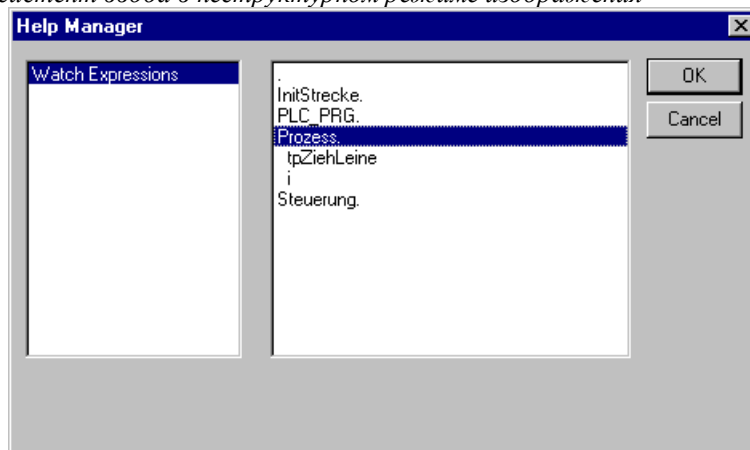
Внимание: Для ввода идентификаторов можно использовать функцию интеллектуального ввода.

Категории элементов, которые доступны в этом диалоговом окне, зависят от текущей позиции курсора в окне редактора, т.е. от того, какой элемент можно ввести в этой позиции (например, переменную, оператор, POU и т.д.).

Если активна опция **With arguments**, то при вставке элемента вместе с ним вставляются и его аргументы. Например, выбрана программа `pr1`, которая имеет один входной параметр `var_in`. Результат вставки будет таким: `pr1(var_in:=);`

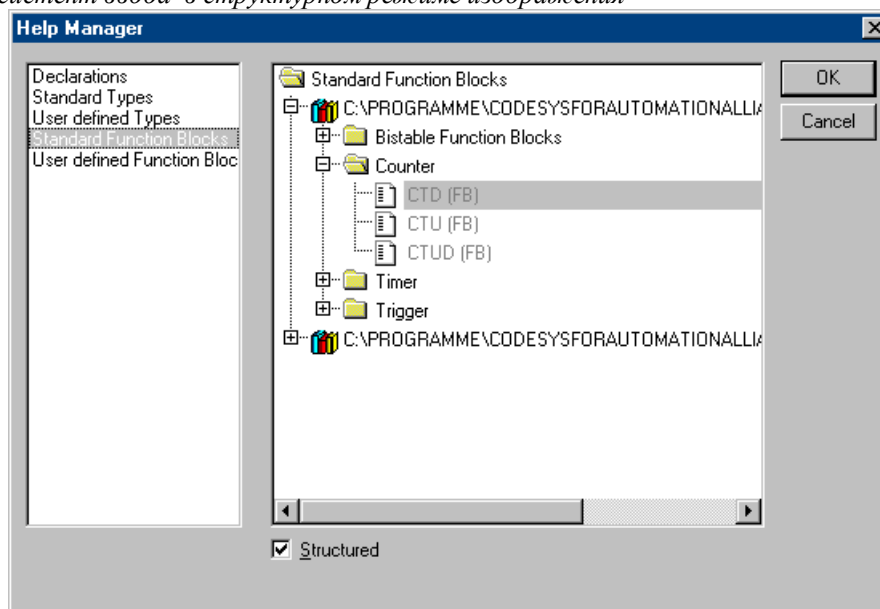
Вставляемая функция `func1` с параметрами `var1` и `var2` после выполнения команды будет выглядеть так: `func1(var1,var2).`

Можно использовать структурный и неструктурный режим изображения доступных элементов. Для переключения этих режимов служит опция **Structured Display**.

Ассистент ввода в неструктурном режиме изображения

POU, переменные или типы данных сортируются в алфавитном порядке. Выберите нужный элемент и нажмите кнопку ОК.

Заметим, что неструктурный режим изображения доступен не всегда. Если в выбранной позиции (например, в объекте Watch and Receipt Manager) используются многоуровневые элементы, то доступен только структурный режим изображения

Ассистент ввода в структурном режиме изображения

Если выбран структурный режим изображения, то POU, переменные и типы данных представлены в виде иерархического дерева. Такой режим можно использовать для вставки стандартных программ, функций, функциональных блоков и определенных пользователем программ, функций, функциональных блоков, глобальных переменных, локальных переменных, типов данных и просматриваемых переменных. Структурный режим изображения похож на тот, что используется в Организаторе объектов. Если к проекту присоединены библиотеки, то их элементы находятся в категориях, начинающихся со слова Standard.

Входные и выходные параметры экземпляров функциональных блоков, которые объявлены глобально или локально, перечислены в категориях Global Variables или Local Variables под именем соответст-

вующего экземпляра. Чтобы получить список входных и выходных параметров экземпляра функционального блока, щелкните по нему мышкой.

Если вы вставляете экземпляр функционального блока, то можно воспользоваться опцией **With arguments**. При этом в языках IL и ST вставляется имя экземпляра и его входные параметры.

Например, вставляем экземпляр Inst функционального блока TON. Результат выглядит так:
Inst(IN:=,PT:=)

Если опция **With arguments** не активна, то вставляется только имя экземпляра. В графических редакторах можно вставить только имя экземпляра.

Вставка элемента структуры выполняется так же, как вставка локальных переменных экземпляра функционального блока

В категории **enumerations** описаны используемые в проекте перечисления. Каждое перечисление задается последовательностью его значений. Перечисления расположены в таком порядке: сначала перечисления из библиотек, потом перечисления из типов данных и, наконец, локальные перечисления из POU.

Многоуровневые элементы в окне Ассистент ввода нельзя вставить (за исключением экземпляров функциональных блоков, см. выше), но их можно раскрыть и вставить входящие в них элементы более низкого уровня.

Если Ассистент ввода вызван из **Watch and Receipt Manager** или в диалоге для настройки конфигурации трассировки, то можно выбрать сразу несколько элементов.

Нажав <Shift>, вы можете выбрать группу элементов, а <Ctrl> - несколько отдельных элементов.

Если вы попытаетесь выделить группу элементов, в которой есть не выделяемые элементы (например, имена POU), то эти элементы будут пропущены. Если вы попытаетесь выделить такой элемент, у вас ничего не получится.

В объекте **Watch and Receipt Manager** или в диалоге для настройки конфигурации трассировки можно вставлять структуры, массивы и экземпляры функциональных блоков. Для того чтобы выбрать такой элемент, выделите его и нажмите ОК.

При вставке группы элементов в объекте **Watch and Receipt Manager** каждый элемент помещается на отдельную строку. Если вы делаете это в диалоге для настройки конфигурации трассировки, то каждая переменная будет вставлена в отдельную строку списка трассируемых переменных.

Так как максимальное число трассируемых переменных равно 20, то при вставке большего числа переменных появится сообщение "A maximum of 20 variables is allowed" и в список добавятся только 20 переменных.

Замечание: Элементы некоторых категорий (например, категории Global Variables) обновляются в Ассистенте ввода только после компиляции.

“Edit” “Declare Variable”

Быстрый вызов: <Shift>+<F2>

Открывает диалог для объявления переменных. Этот же диалог открывается при использовании еще не объявленной переменной, если активна опция 'Project' 'Options' 'Editor' 'Autodeclaration'.

“Edit” “Next error”

Быстрый вызов: <F4>

Если проект скомпилирован с ошибками, то эта команда показывает следующую ошибку. При этом открывается соответствующий редактор в том месте, где произошла ошибка. В то же время в окне сообщений появляется краткое описание этой ошибки.

“Edit” “Previous error”

Быстрый вызов: <Shift>+<<F4>

Если проект скомпилирован с ошибками, то эта команда показывает предыдущую ошибку. При этом открывается соответствующий редактор в том месте, где произошла ошибка. В то же время в окне сообщений появляется краткое описание этой ошибки.

“Edit” “Macros”

Содержит список всех определенных в проекте макросов (см. 'Project' 'Options' 'Macros'). При выполнении макроса появляется окно “Process Macro”, в котором выводится имя макроса и имя активной команды. Нажав кнопку Cancel, вы можете приостановить работу макроса. В этом случае выполняемая команда будет прервана. В окне сообщений и в файле протокола появится следующее сообщение: “<Macro>: Execution interrupted by user”.

Макрос можно выполнять в режиме Online и Offline, но в каждом случае выполняются только те команды, которые доступны в этом режиме.

4.6 Основные функции Online

Функции Online сосредоточены в пункте главного меню “Online”. Доступность некоторых команд зависит от активного редактора.

Команды режима Online становятся доступными только тогда, когда будет установлено соединение с контроллером.

“Online” “Login”

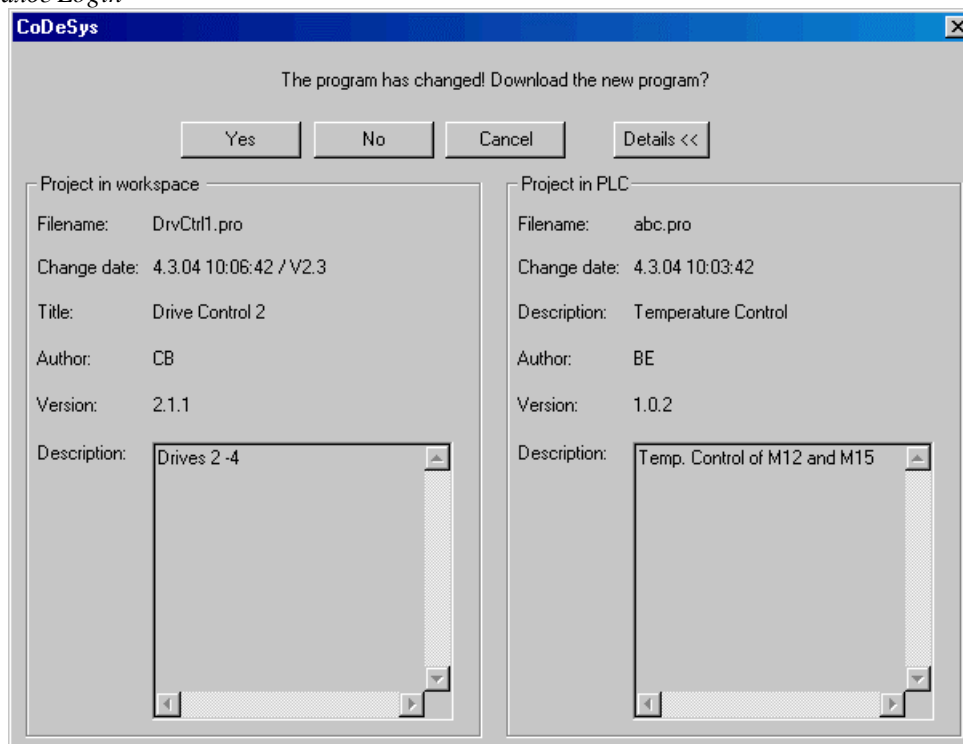
Обозначение:  Быстрый вызов: <Alt>+<F8>

Команда устанавливает соединение системы программирования CoDeSys с контроллером (или запускает программу эмуляции) и включает режим Online.

Если при этом проект не откомпилирован, то он компилируется (то же самое выполняет команда “Project” “Build”). Если при компиляции будут обнаружены ошибки, то CoDeSys не выполняет Login.

Если проект, открытый в CoDeSys, отличается от того, который находится в контроллере и информация о загрузке не стерта командой “Project” “Clear All”, то при выполнении команды “Login” появится сообщение “The program has been modified! Should the new program be loaded?” (Программа изменена. Загрузить новую программу?) Если вы ответите **Yes**, то в контроллер будет загружена новая программа. Ответ **No** приводит к тому, что продолжает работать старая программа и система входит в режим Online. Кнопка **Cancel** отменяет команду.

Диалог Login



Если в разделе Desktop опций проекта включена опция 'Online in security mode' и целевая система поддерживает данную возможность, то диалог будет иметь кнопку **Details**. С ее помощью вы сможете просмотреть и сравнить информацию о проекте, загруженном в контроллер, и о текущем проекте.

Обратите внимание: Доступность данной информации и наличие самой кнопки определяется целевой платформой.

Обратите внимание: Online изменение кода становится невозможным после модификации конфигурации задач или PLC, после включения библиотеки и после команды 'Project' 'Clean all' (см. ниже). Online изменение не приводит к переинициализации переменных, таким образом, модификация начальных значений не приведет к изменению Retain переменных. В этих случаях необходимо перезагрузить проект целиком командой 'Online' 'Download'.

После удачного соединения станут доступны все функции Online. Будет осуществляться мониторинг всех объявленных переменных.

Используйте команду "Online" "Logout" для отключения режима Online.

Во время выполнения команды возможны следующие сообщения:

- "The selected controller profile does not match that of the target system..." (конфигурация контроллера не соответствует настройкам целевой системы.)

Проверьте настройки целевой системы в Target settings (вкладка Resources) и параметры соединения (команда 'Online' 'Communications parameters').

- „Communication error. Log-out has occurred“ (ошибка соединения. Связь с контроллером прервана).

Проверьте, включен ли контроллер и соответствуют ли параметры, установленные в 'Online' 'Communications parameters', параметрам контроллера (номер порта и скорость передачи (baud rate)). Если вы используете удаленный Gateway server, то проверьте его параметры.

- “*The program has been modified! Should the new program be loaded?*” (Программа изменена. Загрузить новую программу?)

Проект, открытый в CoDeSys, отличается от того, который находится в контроллере. Мониторинг в этом случае не возможен. Если вы ответите Yes, то в контроллер будет загружена новая программа. Ответив No, вы сможете выйти из режима Online и загрузить соответствующий проект.

- „*The program has been changed. Load changes? (ONLINE CHANGE)*” (Программа изменена. Загрузить изменения?)”.

Проект работает в контроллере. Целевая платформа поддерживает Online коррекцию кода, и проект был изменен со времени последней загрузки или обновления. Вы должны решить, загрузить ли в контроллер последние изменения или нет. Кроме того, вы можете загрузить новый код целиком с помощью кнопки **Load all**.

Особенности Online Change:

Некоторые целевые системы поддерживают возможность горячего обновления кода (при выполнении Online Login), посредством чего можно изменять часть кода проекта прямо в работающем контроллере.

Здесь важно иметь в виду следующее:

- Функция Online Change недоступна после изменения конфигурации задач или ПЛК, после вставки библиотеки.
- Если загрузочная информация (файл <projectname><targetidentifier>.ri), сохраненная при последней загрузке проекта, будет удалена (например командой 'Project' 'Clean all'), то функция Online Change станет недоступной. Однако если ri-файл был сохранен в другом месте, то загрузочную информацию можно восстановить явно командой 'Project' 'Load download information'.
- Online Change не вызывает переинициализацию переменных. Поэтому изменение начальных значений переменных не приведет к желаемому результату!
- Retain переменные сохраняют свои значения при Online Change в отличие от полной загрузки нового кода (см. 'Online' 'Download').

Горячее обновление кода в нескольких ПЛК:

Допустим, вы хотите запустить проект proj.pro на двух идентичных контроллерах PLC1 и PLC2 (целевые системы идентичны) и при этом необходимо осуществлять горячее обновление кода в обоих контроллерах. Для этого нужно сделать следующее:

(1) Загрузка исходного проекта в PLC1 и сохранение загрузочной информации:

1. Соедините CoDeSys с PLC1 (См. Online/Communication parameters) и загрузите *proj.pro* (Online/Login, Download). В директории проекта образуется файл *proj00000001.ri*, содержащий загрузочную информацию.
2. Переименуйте *proj00000001.ri*, например в *proj00000001_PLC1.ri*. Это исключит потерю данной информации при загрузке *proj.pro* во второй контроллер.
3. Запустите проект в PLC1 и отключите соединение с ним ('Online' 'Start', 'Online' 'Logout').

(2) Загрузка исходного проекта в PLC2 и сохранение загрузочной информации:

1. Соедините CoDeSys с PLC2 (без изменения целевой системы) и загрузите *proj.pro*. В директории проекта образуется файл *proj00000001.ri*, содержащий загрузочную информацию.

2. Переименуйте *proj00000001 ri*, например в *proj00000001_PLC2 ri*.
3. Запустите проект в PLC2 и отключите соединение с ним ('Online' 'Start', 'Online' 'Logout').

(3) Изменение проекта в CoDeSys:

Отредактируйте проект *proj.pro* в CoDeSys, который нужно будет обновить в контроллерах.

(4) Горячее обновление кода в PLC1, восстановление загрузочной информации PLC1:

1. После записи в PLC2, файл *proj00000001 ri* соответствует второму контроллеру. Необходимо восстановить данные, сохраненные для первого контроллера из файла *proj00000001_PLC1 ri*.

Для этого есть 2 возможности:

(a) Переименуйте *proj00000001_PLC1 ri* обратно в *proj00000001 ri*. В этом случае при соединении с PLC1 соответствующая информация будет доступна автоматически, и вы получите запрос на Online Change.

(b) Считайте файл *proj00000001_PLC1 ri* до выполнения login командой 'Project' 'Load Download Information'. В этом случае переименовывать файл ненужно.

(5) Горячее обновление кода в PLC2, восстановление загрузочной информации PLC2:

Восстановите данные, сохраненные для второго контроллера из файла *proj00000001_PLC2 ri* (3) как описано в (4).

“Online” “Logout”

Обозначение:  Быстрый вызов: <Ctrl>+<F8>

Соединение с контроллером разрывается или, если работа происходит в режиме эмуляции, программа заканчивает работу. Система переходит в режим Offline.

Для того чтобы снова соединиться с контроллером, используйте команду “Online” “Login”.

“Online” “Download”

Загружает код проекта в контроллер.

Информация о загрузке сохраняется в файле <имя проекта>0000000ag.rg, который используется, если система поддерживает возможность Online Change (изменения в режиме Online). Этот файл удаляется командой “Project” “Clear all”.

В зависимости от целевой платформы при каждом создании загрузочного проекта *.rg файл может генерироваться заново.

Только persistent переменные сохраняют свои значения при загрузке нового кода проекта.

“Online” “Run”

Обозначение:  Быстрый вызов: <F5>

Запускает программу в контроллере или в режиме эмуляции.

Эта команда доступна сразу после загрузки программы в контроллер, после того как программа остановлена командой **“Online” “Stop”**, после остановки программы на точке останова или при выполнении программы по циклам командой **“Online” “Single Cycle”**.

“Online” “Stop”

Обозначение:  Быстрый вызов: <Shift>+<F8>

Останавливает программу при ее выполнении в контроллере или в режиме эмуляции. При вызове этой команды программа заканчивает цикл и останавливается.

Чтобы продолжить выполнение программы, используйте команду **“Online” “Run”**.

“Online” “Reset”

Сброс. Заново инициализирует все переменные, за исключением VAR RETAIN. Если вы определили начальные значения переменных, они будут присвоены (включая VAR PERSISTENT). Прочие переменные приобретут стандартные значения по умолчанию (например, 0 для целых типов). Перед тем как переменные будут инициализированы, вы должны будете подтвердить это. Данный сброс аналогичен выключению и включению питания ПЛК при работающей программе. См. также 'Online' 'Reset (original)' и 'Online' 'Reset (cold)'.

Для запуска программы используйте команду **“Online” “Run”**.

“Online” “Reset (cold)”

Холодный сброс. Выполняет те же действия, что и команда **“Online” “Reset”**, и дополнительно выполняет инициализацию энергонезависимой области памяти RETAIN.

Для запуска программы используйте команду **“Online” “Run”**.

“Online” “Reset (original)”

Заводской сброс. Выполняет Reset Cold. Инициализирует PERSISTENT область и удаляет программу пользователя. Иными словами, восстанавливает состояние контроллера, в котором он поступает с завода изготовителя.

Примечание: В определенных целевых платформах команды Reset могут выполнять некоторые дополнительные действия. См. документацию на контроллеры.

“Online” “Toggle Breakpoint”

Обозначение:  Быстрый вызов <F9>

Устанавливает точку останова в текущей позиции активного окна. Если в этой позиции уже стоит точка останова, то она будет удалена.

Позиция, в которой можно установить точку останова, зависит от редактора.

В текстовых редакторах IL и ST точка останова устанавливается в строке, в которой находится курсор. В строке можно поставить точку останова, если ее номер отмечен темно-серым цветом. Точку останова также можно установить или удалить, щелкнув по номеру строки мышкой.

В редакторах FBD и LD точка останова устанавливается на выбранной цепи. Здесь точку останова также можно установить или удалить, щелкнув по номеру цепи мышкой.

В SFC точка останова устанавливается на выбранном шаге. Точку останова также можно установить или удалить, щелкая по шагу мышкой при нажатой клавише <Shift>.

Позиция, в которой установлена точка останова, выделяется синим цветом.

Если программа была остановлена на точке останова, то позиция точки останова становится красной. Для того чтобы продолжить выполнение программы, используйте команды Online 'Run', 'Online' 'Step in', или 'Online' 'Step Over'.

Для установки или удаления точек останова можно использовать диалог Breakpoints (команда “Online” “Breakpoint Dialog Box”).

“Online” “Breakpoint Dialog”

Открывает диалог управления точками останова в проекте. В нем указаны все установленные точки останова.

Для того чтобы установить точку останова, выберите требуемый POU в выпадающем списке **POU** и номер строки или цепи в списке **Location**. Нажмите кнопку **Add**, и в выбранной вами позиции будет установлена точка останова. После этого в списке точек останова появится новая запись.

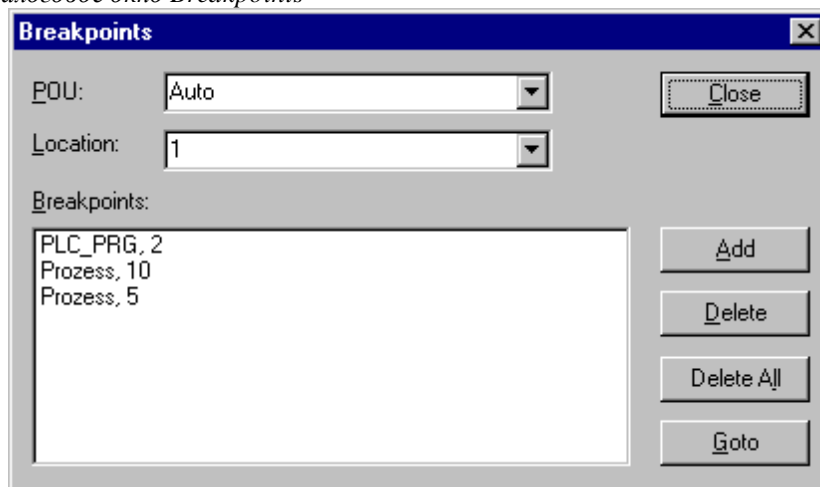
Для удаления точки останова выберите в списке нужную точку останова и нажмите кнопку **Delete**.

Кнопка **Delete All** удаляет все точки останова.

Переход к позиции точки останова осуществляется так: выберите точку останова и нажмите кнопку **Go to**.

Для установки и удаления точек останова также можно использовать команду “Online” “Toggle Breakpoint”.

Диалоговое окно Breakpoints



“Online” “Step over”

Обозначение:  Быстрый вызов <F10>

Выполняет одну инструкцию программы. Если это инструкция вызова POU, то при выборе этой команды целиком выполняется данный POU и после этого программа останавливается. Для того чтобы выполнить этот POU по шагам (зайти в POU), используйте команду “Online” “Step in”.

Как только будет выполнена последняя инструкция вызванного POU, управление возвращается в вызывающий блок.

При выполнении команды **“Step over”** в SFC выполняется один шаг.

“Online” “Step in”

Быстрый вызов <F8>

Выполняет программу по шагам с заходом в вызываемые блоки. Вызываемые POU открываются в отдельных окнах.

Если текущая позиция – это инструкция вызова функции или функционального блока, то выполнение программы останавливается на первой инструкции вызванного блока.

Во всех остальных случаях эта команда работает так же, как и команда **“Online” “Step over”**.

“Online” “Single Cycle”

Быстрый вызов: <Ctrl> +<F5>

Данную команду можно повторять многократно при отслеживании работы программы по рабочим циклам.

Выполняет один рабочий цикл контроллера и останавливается.

Продолжить выполнение программы можно, вызвав команду **“Online” “Run”**

“Online” “Write values”

Быстрый вызов: <Ctrl> +<F7>

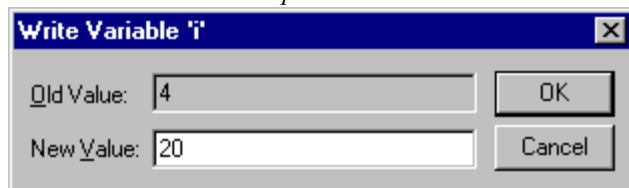
С помощью этой команды вы можете перед началом рабочего цикла записать в переменную или в несколько переменных заранее определенные вами значения (для того чтобы зафиксировать значения переменных, используйте команду **“Online” “Force values”**).

Можно менять значения всех переменных простых типов, которые просматриваются в окне мониторинга.

Перед тем как выполнить эту команду, вы должны определить записываемые значения.

Для установки значения не логической переменной, щелкните по ней мышкой в разделе объявлений или в окне монитора. Есть еще один способ: выделите переменную и нажмите <Enter>. Появится диалоговое окно **“Write variable <x>”**, в котором вы должны ввести новое значение переменной.

Диалог записи значений переменных



Для того чтобы поменять значение логической переменной, по ней достаточно дважды щелкнуть мышкой в разделе объявлений или окне мониторинга. Диалоговое окно при этом не появляется.

Установленное значение выводится в скобках шрифтом бирюзового цвета после текущего значения переменной, например $a = 0 <:= 34 >$

Замечание: Исключения составляют редакторы FBD и LD, в которых установленное значение выводится бирюзовым цветом без скобок сразу после имени переменной.

Можно определить значения любого числа переменных.

Если вы уже ввели новое значение, то вы можете тем же образом его удалить или исправить. То же можно сделать с помощью команды “Online” “Write/Force dialog”(см. ниже).

Прежде чем значения переменных будут записаны в контроллер, они сохраняются в списке записываемых переменных (Writelist), где они хранятся до собственно записи, удаления или перевода в список фиксированных переменных (Forcelist), командой “Force values”.

Команду **Write Values** можно запустить из двух мест:

- § Из меню Online
- § Из диалога Editing the writelist and the forcelist.

При выполнении этой команды все значения из списка Writelist записываются в контроллер в начале цикла, и после этого список Writelist очищается (при выполнении команды “Force values” значения переменных копируются в список Forcelist и список Writelist очищается).

Замечание: В SFC отдельные переменные, из которых состоит выражение, определяющее переход, нельзя изменить командой “Write values”, потому что в этом случае осуществляется мониторинг не отдельных переменных, а всего логического выражения. Например, для выражения: "a AND b" отображается одно итоговое значение TRUE, хотя обе переменные имеют значения TRUE.

В FBD, напротив, просматривается только первая переменная в выражении, записанном, например, на входе функционального блока. Поэтому команду **“Write values”** можно применить только к этой переменной.

“Online ” “Force values”

Быстрый вызов: **<F7>**

С помощью этой команды можно зафиксировать значения одной или нескольких переменных. Запись заданного вами значения осуществляется в начале и в конце каждого управляющего цикла: 1. Чтение входов. 2. Фиксация переменных. 3. Выполнение кода программы. 4. Фиксация переменных. 5. Запись выходов.

Фиксация будет осуществляться, пока вы ее не отмените командой “Online” “Release force”.

Для определения новых значений переменных создайте список Writelist, как это описано выше (См. “Online” “Write values”). В списке Writelist содержатся переменные, отмеченные в окне мониторинга. При выполнении команды “Force” список Writelist копируется в список Forcelist. После этого список Writelist очищается, и соответствующие переменные в окне мониторинга становятся красными. Фиксируемые значения этих переменных сохраняются в списке Forcelist и загружаются в контроллер.

Замечание: Список фиксируемых переменных Forcelist создается при первом выполнении команды “Force Values” из списка Writelist.

Команду фиксации значений переменных можно вызвать:

- Из меню Online

- Из диалога “Editing the writelist and the forcelist”.

Замечание: В SFC отдельные переменные, из которых состоит выражение, определяющее переход, нельзя изменить командой “Write values”, потому что в этом случае осуществляется мониторинг не отдельных переменных, а всего логического выражения.

В FBD, напротив, просматривается только первая переменная в выражении записанном, например, на входе функционального блока. Поэтому команду “**Write values**” можно применить только к этой переменной.

“Online ” “Release force”

Быстрый вызов: <Shift>+<F7>

Отменяет фиксацию переменных. После выполнения этой команды переменные работают в программе как обычно.

Фиксированные переменные выделяются при мониторинге тем, что их значения показаны красным цветом. Вы можете отменить фиксацию сразу всех переменных или выборочно.

Для того чтобы отменить фиксацию всех переменных, применимы следующие способы:

- Команда “Release Force” в “Online”
- Кнопка “Release Force” в диалоге Editing the writelist and the forcelist”
- Диалог “Remove Write-/Forcelist”(см. ниже)

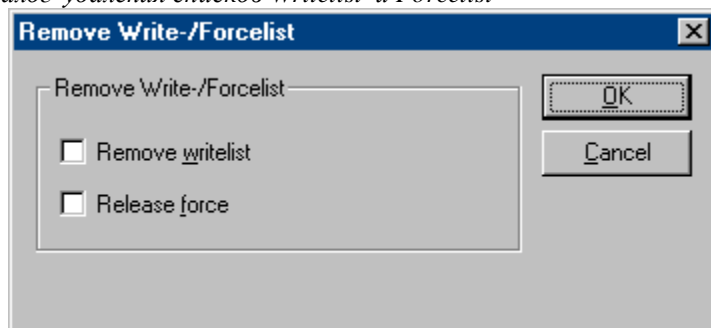
Для того чтобы отменить фиксацию отдельных переменных, их сначала нужно выбрать. Отмеченные переменные обозначаются словом <Release Force> бирюзового цвета. Сделайте это одним из нижеописанных способов:

- Щелкните по зафиксированной нелогической переменной в окне монитора и в появившемся диалоге “Write variable <x>” нажмите кнопку <Release Force for this variable>.
- Щелкайте по зафиксированной логической переменной до появления надписи <Release Force>
- В меню “Online” с помощью команды **Write/Force-Dialog** откройте диалог и удалите значение переменной в столбце **Forced value**.

Когда все необходимые переменные отмечены "<Release Force>" в окне объявления, используйте команду 'Force values' для передачи изменений в контроллер.

Если во время выполнения команды “**Release Force**” список Writelist не пуст, то будет открыт диалог “Remove Write-/Forcelist”. В нем можно указать, какой список удалять - либо **Writelist** (Remove writelist), либо **Forcelist** (Release force). Можно удалить оба списка.

Диалог удаления списков Writelist и Forcelist



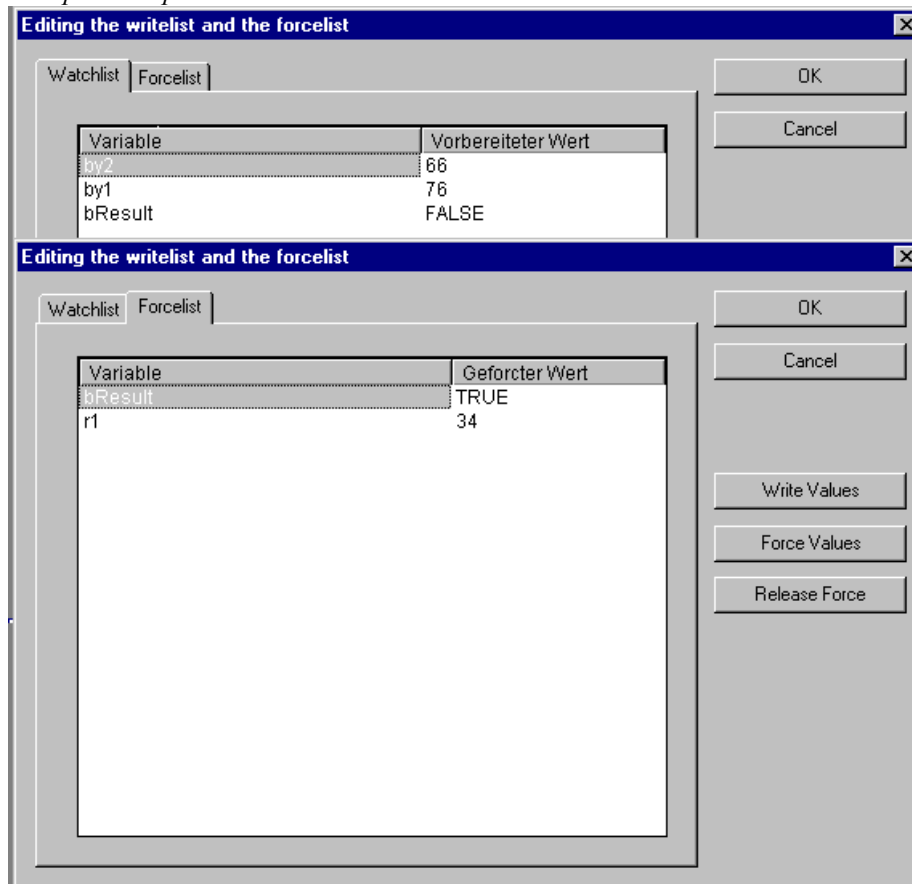
“Online” “Write/Force Dialog”

Быстрый вызов: <Ctrl>+<Shift>+<F7>

Открывает окно, содержащее таблицы записываемых (*Writelist*) и фиксируемых переменных (*Forcelist*). В левом столбце таблиц находятся имена переменных, а в правом - их установленные значения.

Переменные попадают в список *watchlist* посредством команд 'Online' 'Write Values' и передаются в список *forcelist* командой 'Online' 'Force Values'.

Диалог редактирования списков *Writelist* и *Forcelist*



Значения переменных из списка *Writelist* записываются в контроллер с помощью кнопки **“Write Values”**. Чтобы переместить переменные из этого списка в список *Forcelist*, нажмите кнопку **“Force Values”**. В колонках **“Prepared Value”** и **“Forced Value”** вы можете изменить значения переменных из списков *Writelist* и *Forcelist*, щелкнув по ним мышкой. Если при вводе значения переменной вы допустите ошибку, то будет выдано соответствующее сообщение. Если вы удалите значение переменной, то эта переменная будет удалена из списка при выходе из диалога при помощи любой кнопки, кроме **Cancel**.

В этом диалоге доступны следующие соответствующие пунктам меню *Online* команды:

Force Values: Все элементы списка *Writelist* перемещаются в список *Forcelist*. и фиксируются. Для всех переменных, помеченных строкой *<Release Force>*, фиксация отменяется. После этого диалог закрывается.

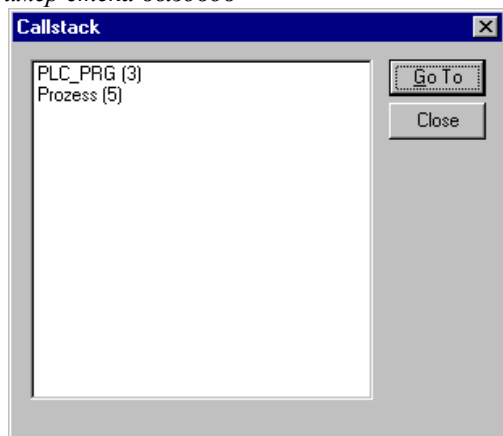
Write Values: Все значения переменных списка *Writelist* записываются в контроллер. После этого диалог закрывается.

Release Force: Удаляются все элементы списка *Forcelist*. Если список *Writelist* не пустой, то открывается диалог **“Remove Write-/Forcelist”**, в котором вы должны указать, какой список нужно удалить. После этого диалог закрывается.

“Online” “Show Call Stack”

Вы можете использовать эту команду в режиме Online, когда программа остановлена в точке останова. Вы увидите диалог со списком вызванных на этот момент POU:

Пример стека вызовов



На первом месте всегда стоит POU PLC_PRG, так как выполнение программы всегда начинается с него.

На последнем месте стоит POU, в котором программа была остановлена.

Если вы выберете POU и нажмете кнопку Go to, то этот POU будет открыт в редакторе на текущей команде.

“Online” “Display Flow control”

Включение режима контроля потока исполнения индицируется галочкой перед командой Display Flow control. Если данная возможность поддерживается в вашей целевой платформе, то каждая строка или цепь программы, которая была выполнена в контроллере в предыдущем управляющем цикле, будет выделена.

Номер каждой такой строки или цепи выделяется зеленым цветом. В редакторе IL, кроме этого, выводится предшествующее значение аккумулятора. В графических редакторах FBD и LD рядом с линиями связи, передающими нелогическое значение, появляются дополнительные поля. В этих полях указываются значения, передаваемые по соответствующим линиям связи. Линии связи, передающие логическое значение TRUE, изображаются синим цветом.

“Online” “Simulation”

Эта команда включает режим эмуляции. Если режим эмуляции включен, то команда Simulation в меню отмечена галочкой.

В режиме эмуляции программа выполняется в ПК. Этот режим используется для тестирования проекта. Взаимодействие с эмулятором опирается на механизм сообщений Windows

Если режим эмуляции выключен, то программа будет запущена в контроллере. Обмен данными между ПК и ПЛК обычно осуществляется по последовательному интерфейсу.

Состояние режима эмуляции (включен/выключен) сохраняется вместе с проектом.

Обратите внимание: POU из внешних библиотек не выполняются в режиме эмуляции.

Длительность рабочего цикла в режиме эмуляции не соответствует (как правило, существенно больше) длительности цикла в реальном контроллере.

“Online” “Communication Parameters”

Выводит диалог для настройки параметров связи ПК и ПЛК. (Если вы используете OPC или DDE серверы, то эти параметры можно настроить из их конфигурации).

Ниже рассмотрены:

- Механизм работы шлюза связи (gateway)
- Установка параметров связи
- Создание нового канала на локальном сервере связи
- Диалог параметров связи на локальном ПК
- Техника редактирования параметров канала
- Устранение проблем связи при работе с удаленным сервером

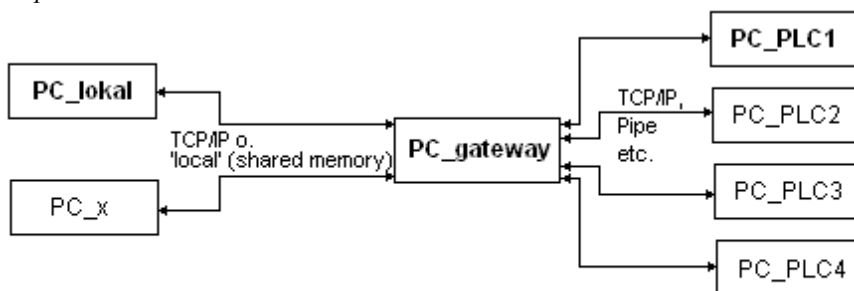
Механизм работы шлюза связи

Взаимодействие ПК с системой программирования происходит посредством вспомогательного сервера связи (gateway). Прежде чем описывать диалог настройки, рассмотрим механизм работы шлюза связи.

Сервер связи позволяет осуществить взаимодействие с одной или несколькими системами исполнения ПЛК. Параметры каналов связи с ПЛК и метод взаимодействия сервера связи с системой программирования необходимо настроить заранее. Возможно, что система программирования и сервер связи являются приложениями, работающими на одной машине. В этом случае сервер связи запускается автоматически при выполнении команды Login. Если сервер связи расположен на другой машине в сети, то он должен быть запущен заранее. Запущенный сервер отображает иконку CoDeSys в правой части панели задач Windows. Изображение иконки подсвечивается, когда установлена связь с ПЛК. Сервер связи продолжает работать независимо от системы программирования. Отключить его можно командой Exit во всплывающем меню (щелкните правой клавишей мыши по иконке сервера связи).

На следующем рисунке показана схема, представляющая работу шлюза связи.

Пример системы связи



На рисунке **PC_lokal** обозначает ваш локальный ПК, **PC_x** – другой ПК, на котором запущен сервер связи **PC_gateway**. **PC_PLC1 ... PC_PLC4** - системы исполнения. На рисунке все элементы показаны отдельно, но в принципе все это может работать на одной машине.

ВАЖНО: Подключение к серверу связи возможно только по TCP/IP. Поэтому убедитесь, что ваш ПК настроен правильно.

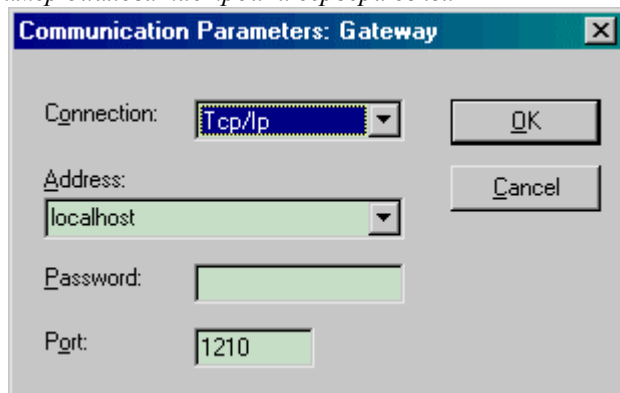
Связь сервера с системами исполнения, напротив, может опираться на различные механизмы (TCP/IP, именованные каналы и т.д.).

Установка параметров связи

1. Выбор сервера связи и канала

Для соединения с сервером связи необходимо использовать диалог "Communication Parameters Gateway". Для его вызова нажмите кнопку "Gateway" в диалоге настройки канала связи.

Пример диалога настройки сервера связи



Здесь вы должны настроить:

- **Connection** - тип соединения с сервером связи. Если сервер локальный, соединение может быть выполнено через разделение памяти (local) или через TCP/IP. Для удаленного сервера только TCP/IP.
- **Address IP** - адрес компьютера, на котором запущен сервер связи или соответствующее символьное имя. По умолчанию используется Localhost, что эквивалентно указанию IP адреса 127.0.0.1. Для соединения с удаленным сервером связи задайте здесь необходимый адрес или символьное имя.
- **Password** – пароль, необходимый для подключения к удаленному серверу. Задать пароль на сервере можно, щелкнув правой кнопкой мыши на иконке сервера связи в панели задач. Во всплывающем меню выберите команду **Change password**. При работе с локальным сервером пароль не нужен.
- **Port** - порт, на который настроен сервер связи. Обычно правильное значение здесь уже задано.

По окончании ввода закройте диалог клавишей ОК. Заданный сервер, включая доступные ему каналы, должен появиться в поле Channels диалога настройки канала связи.

2. Установка нужного канала на выбранном сервере:

Выберете один из каналов сервера. Если соединение с сервером не устанавливается (сообщение "not connected"), возможно, сервер не запущен или настроен неправильно.

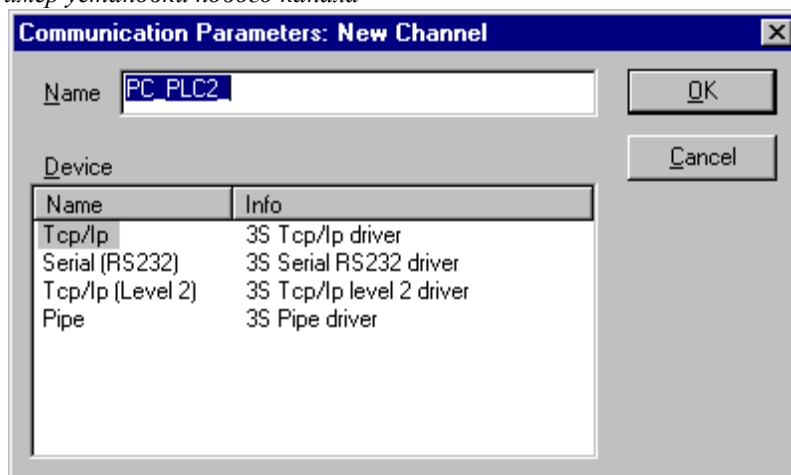
При удачном исходе закройте диалог кнопкой ОК. Указанные настройки будут сохранены в проекте.

Создание нового канала на локальном сервере связи

При наличии соединения с сервером вы можете создавать новые каналы, которые будут использоваться для связи с ПЛК.

Нажмите кнопку **New**. Диалог настройки канала зависит от его типа и определяется установленными в вашей системе драйверами.

Пример установки нового канала



Поле **Name** автоматически содержит имя последнего заданного канала. Если ни одного канала еще не задано, то имя образуется из имени сервера и символа подчеркивания (например, localhost). Вы можете задать любое имя. Имя играет исключительно информационную роль и даже не обязано быть уникальным.

В разделе **Device** выберите необходимый драйвер связи. Список доступных драйверов отражает все установленные в системе драйверы. Если вы закончите ввод кнопкой **OK**, то новый канал появится в разделе **Channels** диалога "Communication Parameters". Пока еще он доступен только локально для данного проекта.

Чтобы внести новый канал в число доступных серверу и сделать возможным его удаленное применение, необходимо выполнить соединение (login). Когда после этого вы вновь раскроете диалог настройки, канал появится в иерархии соответствующего сервера. Теперь вы можете подключиться через этот канал с любого компьютера.

Если возникает ошибка при соединении, возможно, что данный интерфейс (например, COM1) уже занят. Также возможно, что ПЛК просто не подключен.

Параметры каналов, определенные на сервере, недоступны для редактирования и показываются серым. Однако вы можете удалить такой канал, если он не активен.

ВНИМАНИЕ. После удаления канала откат будет невозможен. Канал действительно удаляется в момент нажатия кнопки **Remove**.

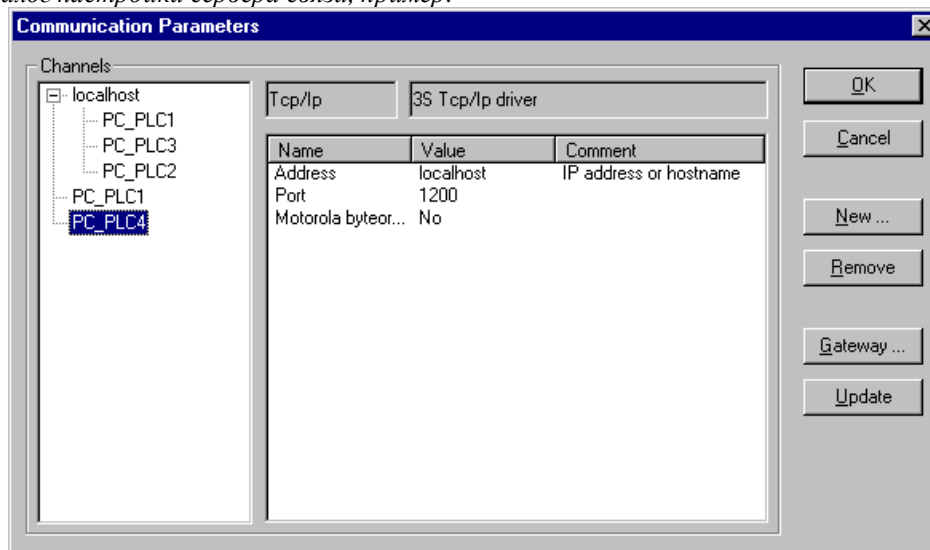
Диалог параметров связи на локальном ПК

Данный диалог служит для выбора сервера связи с ПЛК. Кроме того, он позволяет настроить каналы сервера, установленного на данном компьютере, которые должны быть доступны с других компьютеров сети.

Для просмотра текущих настроек используйте кнопку **Update**.

На рисунке показан пример настройки возможной конфигурации.

Диалог настройки сервера связи, пример:



Под заголовком **Channels** показаны две категории соединений.

- § Первыми показаны каналы подключенного в настоящее время сервера связи. В окне списка показывается его IP адрес или имя в сети. В данном примере это "localhost, что соответствует локальному серверу. Для обращения машины к самой себе в IP сетях предусмотрен адрес 127.0.0.1. Если сервер связи расположен на другой машине, IP адрес, естественно, будет иным. Раскрытый список PC_PCL1 ...PC_PCL3 показывает каналы связи данного сервера с системами исполнения.
- § Вторая категория каналов отображает все возможные соединения вашего локального компьютера. Ветви, показывающие эти каналы, тянутся на рисунке непосредственно от знака "-". В данном примере это PC_PCL1 и PC_PCL4. Здесь можно настроить параметры каналов, как будет описано ниже. Не все показанные здесь каналы обязаны быть известными серверу связи. Так, в примере параметры PC_PCL1 хранятся локально в проекте. Они станут известны серверу связи только при выполнении соединения. Что и произошло в нашем примере для PC_PCL4.

В центральной части окна диалога показаны параметры настройки выбранного канала. Они собраны в таблицу, имеющую 3 столбца: Name, Value и Comment (Имя, Значение и Комментарий).

Техника редактирования параметров канала

Для редактирования с клавиатуры доступны текстовые поля в колонке Value. Войти в режим редактирования можно щелчком мыши или клавишей <пробел>. Клавиша <Enter> заканчивает редактирование.

Клавиши <Tab> и <Shift+Tab> дают возможность перемещаться по полям вверх и вниз.

Для цифровых полей можно выбирать нужное значение. Для перебора значений используйте клавиши <Up>, <Down>, <PgUp>, <PgDn>, <Ctrl+Home> (наименьшее) и <Ctrl+End> (наибольшее) или двойной щелчок мыши.

Устранение проблем связи при работе с удаленным сервером

Если связь с сервером не устанавливается, вы увидите надпись "not connected" в диалоге настройки. В этом случае проверьте:

- § Запущен ли сервер. Иконка (триколор CoDeSys) должна присутствовать в панели задач.
- § Заданный IP адрес доступен (используйте "ping" для проверки).
- § Драйверы TCP/IP установлены, и сеть исправна.

“Online” “Sourcecode download”

Загружает исходные тексты проекта в контроллер. Не путайте исходные тексты проекта с кодом проекта, который создается при компиляции. Параметры этой команды можно установить в опциях проекта “Sourcedownload”.

“Online” “Create bootproject”

Используется для того, чтобы сделать код проекта автоматически загружаемым при перезапуске ПЛК. При перезапуске контроллера этот проект будет выполняться автоматически. Выполнение команды зависит от целевой системы. Например, для 386 системы создаются 3 файла: **default.prg**, содержащий код проекта, **default.chk**, содержащий контрольную сумму кодов, **default.sts**, содержащий статус контроллера.

Эта команда также доступна в режиме Offline, если проект скомпилирован без ошибок. В этом случае в директории проекта создаются следующие файлы: <имя проекта> .prg, в котором сохраняется код проекта, и <имя проекта> .chk – хранит контрольную сумму. При необходимости эти файлы можно переименовать и записать в контроллер.

В зависимости от настроек целевой системы в режиме offline создается новый *.ri-файл (загрузочная информация). Также в зависимости от настроек будет дано сообщение, если такой файл уже существует.

Примечание: Если активна опция **Implicit at create boot project** (категория Sourcedownload), то исходные тексты автоматически загружаются в контроллер при выполнении команды 'Online' 'Create boot project'.

“Online” “Write file to controller”

Используется для записи любых файлов в контроллер. Для выбора загружаемых в контроллер файлов открывается дополнительный диалог.

После нажатия кнопки “Open” выбранный файл загружается в контроллер и сохраняется в нем под тем же именем. Процесс загрузки сопровождается диалогом, в котором выводится процент сделанной работы.

С помощью команды “Load file from controller” вы можете считать ранее сохраненный в контроллере файл.

Доступность этой команды зависит от возможностей целевой системы.

“Online” “Load file from controller”

С помощью этой команды вы можете считать ранее сохраненный в контроллере файл. При этом используется вспомогательный диалог “Load file from controller”. Выберите имя загружаемого из контроллера файла и директорию, в которую вы хотите его поместить. Файл будет сохранен на вашем ПК, когда вы нажмете кнопку “Save”.

4.7 Работа с окнами

Все команды для управления окнами можно найти в меню Windows. Это команды для упорядочивания окон, для открытия менеджера библиотек и для переключения между открытыми окнами. Нижнюю часть меню занимает список открытых окон, упорядоченных в порядке их открытия. Щелкнув на имени окна, вы сделаете его активным. Перед именем активного окна стоит галочка.

“Window” “Tile Horizontal”

Упорядочивает окна по горизонтали так, чтобы они не перекрывали друг друга и полностью занимали рабочую область.

“Window” “Tile Vertical”

Упорядочивает окна по вертикали так, чтобы они не перекрывали друг друга и полностью занимали рабочую область.

“Window” “Cascade”

Упорядочивает окна каскадом.

“Window” “Arrange Symbols”

Выстраивает свернутые окна в ряд внизу рабочей области.

“Window” “Close All”

Закрывает все окна.

“Window” “Messages”

Быстрый вызов: <Shift>+<Esc>

Открывает или закрывает окно сообщений, которое содержит информацию о предыдущей компиляции, проверке или сравнении проекта. Если окно сообщений открыто, то перед этим пунктом меню стоит галочка.

4.8 Помощь

'Help' 'Contents' и 'Search'

Система оперативной помощи CoDeSys включает всю информацию, которая содержится в печатном руководстве.

Команды Contents и Search из меню Help открывают окна подсказки. Для их просмотра используется HTML Help Viewer (Internet Explorer V4.1 и старше).

Contents дает каталог документов системы оперативной помощи. Тома подсказки раскрываются и закрываются двойным щелчком мыши или соответствующей кнопкой.

Search позволяет перейти к контекстному поиску по текстам документов подсказки. Следуйте инструкциям в окне системы подсказки.

С целью оптимизации информации в документах часть иллюстраций и подробностей не отображается в текстах по умолчанию. Их заголовки выделены синим жирным шрифтом, они раскрываются щелчком мыши на заголовке.

Контекстно-зависимая подсказка

Быстрый вызов: <F1>

Вы можете использовать <F1> в активном окне, в диалогах либо для команд меню. Система помощи автоматически находит и показывает текст документации с наибольшей вероятностью, соответствующую

щей ситуации. Кроме того, вы можете выделить необходимый текст (например, ключевое слово или имя стандартной функции) и воспользоваться подсказкой.

5 Редакторы CoDeSys

5.1 Общие элементы редакторов

Составляющие редакторов

Все редакторы программных компонентов POU (Program Organization Units) содержат область кода и раздел объявлений. Область кода может включать графический или текстовый редактор; раздел объявлений - это всегда текст. Разделы кода и объявлений разделены горизонтальной границей, которую можно перетаскивать мышкой.

Границы печатного листа

Ширина окон редакторов в CoDeSys не ограничена, но для того чтобы получить правильное представление о разбивке на листы при печати, вы можете включить отображение границ печатных листов. Границы - это красные пунктирные линии. Они отображаются при включенной опции 'Show print range' в диалоге опций проекта 'Workspace'. Здесь используются параметры листа, заданные в стандартном диалоге настройки принтера. Если принтер не установлен, используется конфигурация по умолчанию (Default.DFR). Горизонтальные границы включаются опцией 'New page for each object' или 'New page for each sub-object' в настройках 'Documentation settings'. Самая нижняя граница не отображается.

Примечание: Точное отображение границ печатного листа возможно только при масштабе 100%.

Комментарии

Пользовательские комментарии должны заключаться между специальными символами “(*)” и “(*)”. Например: (*Это комментарий.*). Комментарии допускаются во всех текстовых редакторах, в любой позиции. Это относится к разделу объявлений, текстам IL и ST и определению типов данных.

В графических FBD и LD редакторах комментарий допускается в каждой цепи. Для его вставки предназначена команда меню **'Insert' 'Comment'**. Дополнительно в редакторе релейных диаграмм можно снабдить каждую обмотку и контакт, если включена соответствующая опция в меню 'Extras' 'Options'.

В CFC предусмотрен специальный элемент (Comment), который можно свободно размещать в произвольном месте.

В SFC вы можете разместить комментарий около шага, введя его в диалоге атрибутов шага.

Вложенные комментарии разрешены, если включена соответствующая опция (Nested comments) в 'Project' 'Options' 'Build Options'.

В режиме Online, если подвести курсор мыши на короткое время к переменной, то адрес и комментарий отображаются в виде всплывающей подсказки.

Zoom to POU

Быстрый ввод: <Alt>+<Enter>

Раскрыть компонент. Открывает соответствующий редактор для выбранного POU. Команда доступна в контекстном меню или в меню 'Extras', если курсор стоит на имени POU в текстовом редакторе либо если выбран данный POU в графическом редакторе.

Если вы имеете дело с библиотечным POU, то вызывается менеджер библиотек и отображает данный POU.

Open instance

Открыть экземпляр. Соответствует команде 'Project' 'Open instance' . Доступна в контекстном меню (<F2>) и меню 'Extras', если курсор стоит на имени функционального блока в текстовом редакторе либо если выбран функциональный блок в графическом редакторе.

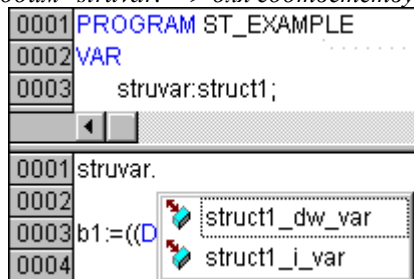
Интеллектуальный ввод

Если соответствующая опция (List components) активирована в категории 'Editor' опций проекта, то «интеллектуальный» ввод поддерживается во всех редакторах в менеджере рецептов, в визуализации и трассировке:

- Если вы вводите точку "." вместо идентификатора, вызывается окно селектора с перечислением всех локальных и глобальных переменных проекта. Вы можете выбрать один из элементов и нажать кнопку 'Return' для вставки его после точки либо вставить элемент двойным щелчком мышки.
- При вводе имени экземпляра функционального блока или структуры за точкой, селектор содержит имена входных и выходных переменных или элементов структуры. Вы вставляете нужный элемент кнопкой 'Return' или двойным щелчком мышки.

Пример:

Вводим "struvar." -> для соответствующей структуры struct1 будет предложено:



- Если при вводе любой строки вы нажмете <Ctrl> + <Space Bar>, то появится диалог выбора POU и глобальных переменных, доступных в проекте. Для вставки выбранного элемента в текущую позицию нажмите клавишу <Enter>.

Offline подсказки для идентификаторов

Во всех редакторах в режиме offline, если курсор помещается на доступный для редактирования идентификатор, то во всплывающем окне подсказки будет указано имя и класс переменной (например, VAR_GLOBAL), тип данных, атрибуты переменной (например, RETAIN), адрес и комментарий.

5.2 Редактор раздела объявлений

5.2.1 Работа в редакторе объявлений

Редактор объявлений используется для объявления переменных POU, глобальных переменных, описания типов данных, а также в Менеджере рецептов. Он позволяет использовать основные функции редактирования Windows и при наличии соответствующего драйвера поддерживает мышь со скроллингом.

Переключаться между режимом вставки и замены можно с помощью клавиши <Ins>. В режиме замены в строке статуса надпись “OV” изображается черным цветом.

В разделе объявлений зарезервированные слова, типы данных и сами переменные выделяются разными цветами.

Наиболее важные команды можно найти в контекстном меню, которое появляется при нажатии правой кнопки мыши или сочетанием клавиш <Ctrl>+<F10>.

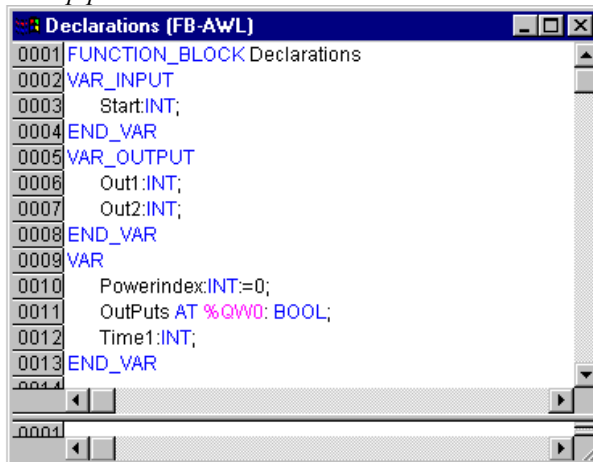
Раздел объявлений

Локальные переменные POU объявляются в разделе объявлений редактора программного компонента. Такими переменными могут быть входные и выходные переменные, переменные, одновременно являющиеся входными и выходными, локальные переменные, сохраняемые переменные и константы.

Обратите внимание на возможность создания шаблонов для глобальных переменных, типов данных, функций, функциональных блоков и программ.

Синтаксис, используемый при объявлении переменных, соответствует стандарту МЭК61131-3. Вот пример правильного объявления переменных в редакторе.

Редактор раздела объявлений



Входные переменные

Входные переменные POU объявляются между ключевыми словами **VAR_INPUT** и **END_VAR**. Значения этих переменных передаются в POU, при его вызове.

Пример:

```
VAR_INPUT
    in1:INT      (*Входная переменная*)
```

END_VAR

Выходные переменные

Выходные переменные POU объявляются между ключевыми словами **VAR_OUTPUT** и **END_VAR**. Через эти переменные POU передает данные в вызывающий его блок, который может читать их, даже не вызывая POU.

Пример:

```
VAR_OUTPUT
    out1:INT; (* Выходная переменная*)
END_VAR
```

Входные и выходные переменные

Объявляются между ключевыми словами **VAR_IN_OUT** и **END_VAR**. Эти переменные можно использовать как входные и как выходные.

Внимание: В отличие от ранее описанных переменных эти переменные передаются по ссылке, а не по значению. Поэтому такая переменная не должна быть константой и не допускает прямого обращения: <имя экземпляра >.<in/out имя>.

Пример:

```
VAR_IN_OUT
    inout1:INT; (* Входная и выходная переменная *)
END_VAR
```

Локальные переменные

Объявляются между ключевыми словами **VAR** и **END_VAR**. Их можно использовать только в том POU, в котором они объявлены. При выходе из POU значения этих переменных пропадают и не доступны извне.

Пример:

```
VAR
    loc1:INT; (* Локальная переменная*)
END_VAR
```

Реманентные переменные

Такие переменные сохраняют свои значения при определенных сбоях в системе. Они бывают сохраняемые и постоянные.

Сохраняемые переменные обозначаются ключевым словом **RETAIN**. Эти переменные сохраняют свои значения, даже если произошла авария питания (выключение и включение) контроллера, что равносильно команде сброс ("Online' 'Reset"). Значения **RETAIN** переменных **сохраняются в энергонезависимой памяти**.

Сохраняемые (Retain) переменные будут инициализированы заново при 'Online' 'Reset (cold)' и 'Online' 'Reset (original), а также при загрузке программы в отличие от постоянных (Persistent) переменных.

Постоянные переменные обозначаются ключевым словом **PERSISTENT**. В отличие от сохраняемых переменных эти переменные сохраняют свои значения только при загрузке кода новой программы, но не при выключении питания или любом сбросе. Значения постоянных переменных **размещаются вне энергонезависимого ОЗУ**.

Переменные **PERSISTENT** можно включить в **RETAIN** объявления.

- = переменная переинициализируется, **x** = значение сохраняется

Online команда	VAR	VAR RETAIN	VAR PERSISTENT	VAR RETAIN PERSISTENT VAR PERSISTENT RETAIN
Reset	-	X	-	x
Reset cold	-	-	-	-
Reset origin	-	-	-	-
Download	-	-	x	x
Online Change	x	X	x	x

Пример:

```

VAR RETAIN
    rem1:INT; (* Сохраняемая переменная*)
END_VAR

```

Внимание:

- Если локальная переменная объявлена как **VAR RETAIN**, то она будет размещена в энергонезависимой области как глобальная.
- Если локальная переменная функционального блока объявлена **VAR RETAIN**, то все его данные целиком помещаются в энергонезависимую область памяти, но обслуживаться как **VAR RETAIN** будет только данная переменная.
- Если локальная переменная функции объявлена **VAR RETAIN**, то это не дает никакого эффекта. Переменная не будет помещена в энергонезависимую область памяти! Аналогичное объявление **PERSISTENT** также не дает желаемого эффекта!

Константы, типизированные литеры

Константы обозначаются ключевым словом **CONSTANT**. Их можно объявлять локально и глобально.

Синтаксис:

```

VAR CONSTANT
    <Идентификатор>:<Тип>:=<начальное значение>
END_VAR

```

Пример:

```

VAR CONSTANT
    con1:INT:=12; (* Константа*)
END_VAR

```

Правила создания констант и использования типизированных литер можно найти в приложении В.

Внешние переменные

Глобальные переменные, которые должны быть импортированы в POU, объявляются с помощью ключевого слова `EXTERNAL`. Такая переменная появляется в окне просмотра переменных в режиме Online.

Если описания внешней переменной в POU и в разделе глобальных переменных не совпадают, то при компиляции появляется ошибка "Declaration of '<var>' does not match global declaration!"

Если внешняя переменная не описана в разделе глобальных переменных, то при компиляции появляется ошибка "Unkown global variable: '<var>!'"

Пример:

```
VAR_EXTERNAL
    var_ext1:INT:=12;    (* Внешняя переменная *)
END_VAR
```

Зарезервированные слова

Все зарезервированные слова записываются заглавными буквами. Их нельзя использовать в качестве имен переменных. Например: `VAR`, `VAR_CONSTANT`, `IF`, `NOT`, `INT`.

Объявление переменных

Переменные объявляются следующим образом:

```
<Идентификатор> {AT <Адрес>}:<Тип> {:=<начальное значение>};
```

Части, заключенные в фигурных скобках, не обязательны.

Имена переменных не должны содержать пробелов и специальных символов, должны объявляться только один раз и не должны совпадать с зарезервированными словами. Регистр букв в имени переменной не имеет значения, т.е. переменные `Var1`, `VAR1` и `var1` не различаются. В именах переменных допустим знак подчеркивания. Переменные `A_BCD` и `AB_CD` считаются разными. Идентификатор не должен содержать подряд более одного символа подчеркивания. Длина идентификатора не ограничена, все символы являются значимыми.

Все переменные и типы данных можно инициализировать. Для этого используется оператор “:=”. Переменные простейших типов инициализируются константами. По умолчанию все переменные инициализируются нулем.

Пример:

```
var1:INT:=12; (* Переменная типа INT, инициализируемая числом 12*)
```

Если вы хотите поместить переменную по определенному адресу, то нужно объявить ее с ключевым словом `AT`.

Для быстрого объявления переменных используйте режим быстрого ввода.

В функциональных блоках можно объявлять переменные, используя неполный адрес. Чтобы использовать такую переменную в экземпляре функционального блока, она должна быть описана в разделе `Variable configuration`.

Обратите внимание на возможность использования автоматического объявления переменных.

Ключевое слово AT

Если вы хотите поместить переменную по определенному адресу, то нужно объявить ее с ключевым словом AT. Преимущество такого объявления состоит в том, что можно дать значащее имя любому адресу и изменять значение по этому адресу где угодно (например, в разделе объявлений).

Заметьте, что переменная, описывающая вход, не доступна для записи. Кроме того, объявление AT допустимо лишь для глобальных и локальных переменных, но не для входных или выходных переменных POU.

Пример:

```
counter_heat7 AT %QX0.0: BOOL;
lightcabinetimpulse AT %IX7.2: BOOL;
download AT %MX2.2: BOOL;
```

Если логическая переменная имеет адрес типа Byte, Word или Dword, она занимает целый байт, а не один бит!

“Insert” ”Declaration keywords”

Вы можете использовать эту команду для получения списка и быстрого ввода ключевых слов, допускаемых в разделе объявлений POU. После того как ключевое слово будет выбрано из списка, оно будет вставлено в текущую позицию курсора.

Также этот же список можно получить, если открыть Input Assistant <F2> и выбрать пункт **Declaration**.

“Insert” ”Type”

С помощью этой команды можно получить список и осуществить быстрый ввод доступных типов. Этот же список можно получить и при использовании Input Assistant <F2>.

Типы разделены на несколько категорий:

- § Standard types (стандартные типы) BOOL, BYTE и т.д.
- § Defined types (типы, определяемые пользователем: структуры, перечисления и т.д.)
- § Стандартные функциональные блоки.
- § Определенные пользователем функциональные блоки.

CoDeSys поддерживает все стандартные типы МЭК1131-3.

Выделение цветом

Все редакторы выделяют переменные соответствующим цветом при их объявлении или использовании. Это позволяет быстро находить синтаксические ошибки в программе. Если вы случайно забыли закрыть комментарий или ошиблись, вводя зарезервированное слово, то вы сразу заметите это.

Используются следующие цвета:

Синий	Ключевые слова
Зеленый	Комментарии в текстовых редакторах
Розовый	Специальные константы (например, TRUE/FALSE, T#3s,%IX0.0)

Красный Ошибки ввода (например, неверные временные константы, ключевые слова, записанные строчными буквами)

Черный Переменные, константы, операторы.

Режим быстрого ввода

Редактор раздела объявлений позволяет использовать режим быстрого набора. Эта функция выполняется, если вы заканчиваете ввод строки сочетанием клавиш **<Ctrl>+<Enter>**.

Поддерживаются следующие сокращения:

- Все идентификаторы в этой строке преобразуются в объявления
- Тип этих идентификаторов определяется словом, которым заканчивается строка.

V или Bool переменные будут объявлены как BOOL

I или Int переменные будут объявлены как INT

R или Real переменные будут объявлены как REAL

S или String переменные будут объявлены как STRING

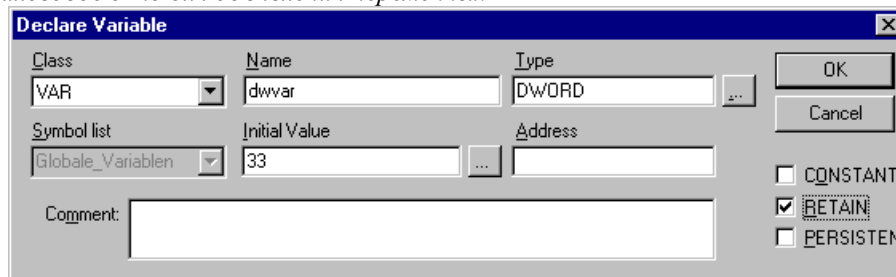
- В других случаях (тип не указан) переменные будут объявлены как BOOL (см. пример 1). Последнее слово в строке понимается как идентификатор.
- Любая константа, в зависимости от типа объявления, будет преобразована в начальное значение переменной (примеры 2 и 3).
- К адресу будет добавлено ключевое слово AT (пример 4).
- Текст после точки с запятой будет превращен в комментарий (пример 4).
- Все другие символы игнорируются (пример 5).

Примеры:

	Быстрый ввод:	Объявление:
1	A	A: BOOL;
2	A B I 2	A, B: INT := 2;
3	ST S 2; A string	ST:STRING(2); (* A string *)
4	X %MD12 R 5 Real Number	X AT %MD12: REAL := 5.0;(* Real Number*)
5	B !	B: BOOL;


Автоматическое объявление переменных

Если поставлен флаг **Autodeclaration** в диалоговом окне Editor Options, то при вводе имен еще не объявленных переменных в любом редакторе будет появляться окно объявления переменных. С помощью этого диалогового окна можно объявить новую переменную.

Диалоговое окно для объявления переменных

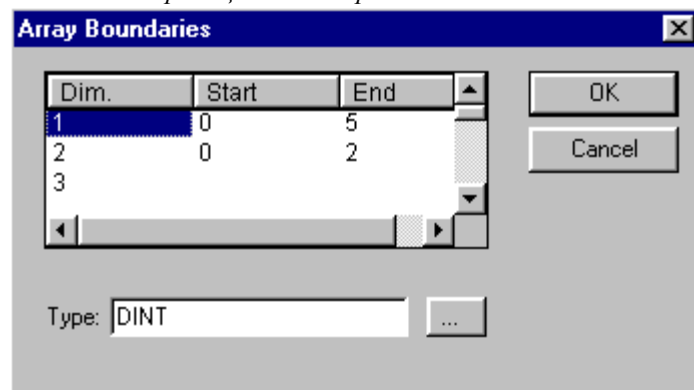
В списке **Class** выберите класс объявляемой переменной: локальная переменная (VAR), входная переменная (VAR_INPUT), выходная переменная (VAR_OUTPUT), входная/выходная переменная (VAR_IN_OUT), глобальная переменная (VAR_GLOBAL).


Устанавливая флаги **CONSTANT**, **RETAIN** или **PERSISTENT**, вы помещаете переменную в соответствующий раздел.

Имя объявляемой переменной нужно ввести в поле **Name**. По умолчанию тип переменной будет BOOL (поле **Type**). Кнопка  открывает диалог Input Assistant, который позволяет выбрать тип из списка предложенных.

Объявление массивов.


Если выбран тип данных **ARRAY**, то появиться диалог для ввода границ массива.

Диалог задания границ массива при автоматическом объявлении переменных

Для каждого из трех возможных индексов массива (столбец **Dim.**) можно вводить начальный (**Start**) и конечный (**End**) индекс соответствующего измерения массива. Тип элементов массива вводится в поле **Field**. При этом удобно использовать Input Assistant, который вызывается при нажатии кнопки .

После закрытия этого диалога объявленная переменная появится в поле Type.

Пример: ARRAY [1..5, 1..3] OF INT.

В поле **Initial Value** вы можете задать начальное значение переменной. Если переменная это массив или структура, вы можете открыть специальный диалог, нажав кнопку  или <F2>.

Для массива такой диалог представляет собой список элементов; при щелчке мышкой после надписи “:=” появляется поле для ввода начального значения элемента.

При инициализации структуры ее элементы изображаются в виде дерева. Тип и начальное значение каждого элемента находятся в скобках. При щелчке мышкой после надписи “:=” появляется поле для ввода начального значения элемента. Если компонента структуры – массив, то, чтобы увидеть его элементы, щелкните мышкой на знаке "плюс" перед именем массива. Теперь начальные значения элементов этого массива можно редактировать.

После закрытия диалога начальные значения элементов массива или структуры изображаются в поле Initial value в соответствии со стандартом МЭК.

Пример: `x:=5,field:=2,3,struct2:=(a:=2,b:=3)`

В поле **Address** задается адрес переменной (объявление AT)

При необходимости можно вставить комментарий. Переход к следующей строке осуществляется нажатием сочетания клавиш <Ctrl>+<Enter>.

После нажатия кнопки ОК переменная будет объявлена в соответствующем разделе объявлений.

Примечание: Диалог объявления переменной можно также вызвать командой 'Edit' 'Declare Variable'. Если в режиме Online курсор стоит на имени переменной, то можно вызвать окно Autodeclare <Shift><F2> для просмотра деталей объявления данной переменной.

Номера строк в редакторе объявлений.

В режиме Offline, щелкнув по номеру строки, можно выделить строку целиком.

В режиме Online, щелкнув по номеру строки, можно установить новое значение переменной или, если это сложная переменная, просмотреть ее поля.

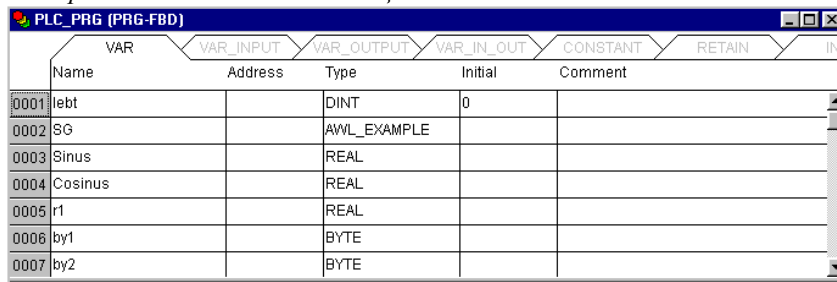
Объявление переменных в виде таблицы

Если флаг **Declarations as tables** в диалоговом окне Editor установлен, то раздел объявлений будет выглядеть как таблица. В таблице можно выбирать вкладки, отвечающие за классы переменных (локальные, глобальные и т.д.), и редактировать переменные.

Для каждой переменной доступны следующие поля:

Name:	Имя переменной
Address:	Адрес переменной (объявление AT), если он нужен.
Type:	Тип переменной. (При создании экземпляра функционального блока введите имя функционального блока)
Initial:	Начальное значение переменной (вводится после “:=”)
Comment:	Комментарии

Оба типа редактора объявлений можно переключать без всяких проблем. В режиме Online редактор всегда выглядит как обычно.

Редактор объявлений в виде таблицы


	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CONSTANT	RETAIN	INIT
	Name	Address	Type	Initial	Comment		
0001	lebt		DINT	0			
0002	SG		AWL_EXAMPLE				
0003	Sinus		REAL				
0004	Cosinus		REAL				
0005	r1		REAL				
0006	by1		BYTE				
0007	by2		BYTE				

“Insert” ” New declaration ”

С помощью этой команды вы можете добавить новую переменную в таблицу редактора объявлений. Если курсор находится в каком-либо поле таблицы, то новая переменная помещается в предыдущую строку, иначе новая переменная помещается в конец таблицы. Кроме того, вы можете добавить новую переменную в конец таблицы, если нажмете TAB или клавишу «Направо» в последнем поле таблицы.

По умолчанию вы получите переменную с именем “Name”, которое записано в поле Name, и с типом “Bool” в поле Type.

Вы должны поменять значение этих полей на те, которые вам нужны. Имя и тип переменной – это все, что необходимо для объявления переменной.

5.2.2 Редактор раздела объявлений в режиме Online

В режиме Online редактор раздела объявлений превращается в окно монитора. В каждой строке находится переменная и ее значение после знака “=”. Если значение переменной не определено, то после знака “=” находятся 3 знака вопроса. Для функциональных блоков изображаются значения только его открытых экземпляров (команда "Project" "Open instance").

Перед каждой составной переменной находится знак плюс. Если нажать <Enter> или дважды щелкнуть мышкой на такой переменной, то она раскроется, и вы увидите значения полей этой переменной. Переменная закрывается аналогичным образом.

```

+---AMPPEL1
  |.....STATUS = 3
  |.....GRUEN = FALSE
  |.....GELB = FALSE
  |.....ROT = TRUE
  |.....AUS = FALSE

```

Нажмите <Enter> или дважды щелкните мышкой на простой переменной. Появится диалоговое окно, в котором вы сможете назначить новое значение для переменной. Если вы изменяете значение логической переменной, то диалоговое окно не появляется, а значение переменной изменяется на противоположное.

Новое значение переменной обозначается красным цветом. Записать измененные значения переменных в контроллер можно, выбрав команду меню “Insert” ”Write values ”. После этого переменная вновь будет обозначаться черным.

После использования команды “Online” “Force values” все выбранные переменные получают установленные значения и будут сохранять их вплоть до вызова команды “Release force”. Фиксированные значения отображаются красным.

5.2.3 Директивы компилятора

Директивы используются для установки свойств переменных, касающихся компиляции. Их можно использовать в текстах программ и в разделе объявлений переменных.

Директива записывается в фигурных скобках, причем регистр букв здесь не важен: {<Директива >}

Если компилятор не понимает эту директиву, то она интерпретируется как обычный комментарий. Однако появляется предупреждение: „Ignore compiler directive ‚<Instruction text>’!“

В зависимости от типа и параметров директивы она действует на строке, на которой она написана, или на всех последующих строках до директивы, ее отменяющей. Если отменяющей директивы нет, то она действует во всем разделе.

Директива может следовать сразу за именем переменной, причем открывающие и закрывающие скобки должны находиться на одной строке.

В настоящее время CoDeSys принимает следующие директивы:

- {flag} инициализация, мониторинг и управление символьным файлом
- {bitaccess...} битовый доступ
- {parameter..}, {template...}, {instance...} создание вхождений для Менеджера параметров
- {nonpersistent} исключение данных из PERSISTENT

Директивы инициализации, мониторинга и управления символьным файлом

Синтаксис директивы:

{flag [<flags>] [off|on]}

<flag> - комбинация следующих флагов:

Noinit	Переменная не инициализируется
Nowatch	Не производится мониторинг переменной
Noread	Переменная экспортируется в символьный файл без разрешения чтения
Nowrite	Переменная экспортируется в символьный файл без разрешения записи
noread, nowrite	Переменная не экспортируется в символьный файл

С помощью модификатора “on” директива включается и действует на все последующие объявления до ее отключения другой директивой или с помощью ”off” {flag **off**}. Без этих модификаторов директива действует только на текущей строке.

Примеры:

Инициализация и мониторинг:

Переменная **a** не инициализируется и не просматривается. Переменная **b** не инициализируется:

```
VAR
    a : INT {flag noinit, nowatch};
    b : INT {flag noinit };
```

```

END_VAR

VAR
    {flag noint, nowatch on}
    a : INT;
    {flag noint on}
    b : INT;
    {flag off}
END_VAR

```

Ни одна из переменных не инициализируется:

```

{flag noint on}
VAR
    a : INT;
    b : INT;
END_VAR
{flag off}
VAR
{flag noint on}
    a : INT;
    b : INT;
{flag off}
END_VAR

```

Размещение переменных в символьном файле:

Флаги “noread” и “nowrite” используются для ограничения прав доступа к переменным. По умолчанию переменная обладает теми свойствами доступа, которые определены для POU, в котором она объявлена. Если переменная не доступна ни для чтения, ни для записи, то в символьный файл она не экспортируется.

Пример:

Если для POU установлено разрешение записи и чтения переменных, то переменная **a** экспортируется с разрешением записи, а переменная **b** не экспортируется вообще:

```

VAR
    a : INT {flag noread};
    b : INT {flag noread, nowrite};
END_VAR

VAR
{ flag noread on}
    a : INT;
{ flag noread, nowrite on}
    b : INT;
{flag off}
END_VAR

```

Ни переменная **a**, ни переменная **b** не экспортируются в символьный файл:

```

{ flag noread, nowrite on }
VAR
    a : INT;
    b : INT;
END_VAR

{flag off}
VAR
{ flag noread, nowrite on}
    a : INT;
    b : INT;
{flag off}
END_VAR

```


Действие одной директивы накладывается на действие другой:

Пример: (Для всех POU установлено разрешение записи и чтения переменных)

```
a : afb;
...
FUNCTION_BLOCK afB
VAR
    b : bfb {flag nowrite};
    c : INT;
END_VAR
...
FUNCTION_BLOCK bfB
VAR
    d : INT {flag noread};
    e : INT {flag nowrite};
END_VAR
```

Переменные **a**, **c** экспортируются с разрешением чтения и записи.

Переменные **b**, **e** экспортируются без разрешения записи.

Переменная **d** экспортируется без разрешения чтения.

Директива битового доступа {bitaccess...}

Используется для корректного отображения переменной, осуществляющей битовый доступ посредством глобальной константы. Она применяется в ассистенте ввода, в функции интеллектуального ввода и разделах объявлений. Директива влияет на мониторинг переменной в окне объявления соответствующего POU. Используемая глобальная константа показывается под соответствующей структурной переменной.

Данную директиву нужно вводить в отдельной строке объявления структуры. Точка с запятой в конце этой строки не нужна.

Синтаксис:

```
{bitaccess <Global Constant> <Bitnumber> '<comment>')}
```

<Global Constant>: Идентификатор глобальной константы, которая должна быть определена в списке глобальных переменных.

<Bitnumber>: Значение глобальной константы.

<comment>: Комментарий

Обратите внимание: опция 'Replace constants' (категория Build) должна быть активна!

Пример применения данной директивы см. «Приложение В: Операнды в CoDeSys» раздел «Переменные», «Доступ к битам в переменных».

Директивы создания вхождений для Менеджера параметров

Включение таких директив в объявление переменных позволяет добавлять элементы списка управляемого Менеджером параметров (Parameter Manager). Наличие Менеджера параметров в среде программирования CoDeSys определяется целевой платформой. Поддержка Менеджера параметров включается

специальным флагом на страничке сетевых опций Networkfunctionality в настройках целевой платформы.

Синтаксис:

- Директива заключена в фигурные скобки, регистр ввода не учитывается: { <Инструкция>}. Если она включена в «нормальное» объявление переменной, то должна стоять до завершающей точки с запятой.
- Директивы, использующие интерфейс VAR_CONFIG, должны вводиться отдельной строкой, без точки с запятой.
- Описание ключей записывается через пробел, внутри фигурных скобок.
- <name>: имя списка параметров. Если указанный список отсутствует, он будет создан.
- <key>: ключ или имя атрибута, т.е. название столбца в списке параметров ("Name", "Value", "Accesslevel" и т.д.). Ключи зависят от типа списка. Определение ключей записывается через пробелы, все вместе заключенные в прямоугольные скобки. Аналогичный синтаксис используется для включения элементов массивов, структур или функциональных блоков (см. 3, ниже).
- <value>: значение атрибута, заданное <key>. Если значение <value> имеет пробелы, то его необходимо заключить в двойные кавычки. Например: accessright="read only".

Обратите внимание: директивы компиляции «срабатывают» только при предварительном анализе текста, который выполняется после изменения фокуса ввода, то есть при выходе из окна редактора. Ошибки ввода сообщаются только при полной компиляции.

Директивы определяют следующие вхождения:

1. Вхождения в списки типа 'Variables'

(а) из раздела объявлений программ и списков глобальных переменных:

Вхождение в список типа 'Variables' для переменной из объявлений PROGRAM или VAR_GLOBAL задается следующим образом:

```
{parameter list=<name> [ <key>=<value> <key>=<value> ...прочие ключи ] }
```

Пример: Переменная bvar объявлена в программе. Она должна входить в список параметров parlist1 (типа 'Variables') с именем bvar1, значением 102, индексом 16#1200 и подиндексом 16#21.

```
VAR
  bvar:INT{parameter list=parlist1 [name=bvar1 value=102 index=16#1200
    subindex=16#1 ] };
END_VAR
```

(б) из объявления интерфейса VAR_CONFIG:

Вхождение в список типа 'Variables' для переменной из окна VAR_CONFIG (вне зависимости от определения конфигурационной переменной) задается следующим образом:

```
{parameter list=<name> path=<path> [ <key>=<value> <key>=<value> ... прочие ключи ] }
```

<path> путь к переменной, например, "PLC_PRG.act1.var_x"

Пример: Для переменной var_x an создается вхождение в список "varlist1", символьное имя – "xvar".

```
VAR_CONFIG
```

```
{parameter list=varlist1 path=PLC_PRG.act1.var_x [ name=xvar ] }
END_VAR
```

2. Вхождения в списки типа 'Template' через функциональные блоки и структуры

Директива в объявлениях функциональных блоков или структур позволяет создавать вхождения в списки типа: 'Template':

```
{template list=<name> [ <key>=<value> <key>=<value> ... прочие ключи ] }
```

Пример: переменная strvar, являющаяся элементом структуры "stru1", должна входить в список "templ1" типа 'Template'; символьное имя "struvar1", уровень доступа "low":

```
TYPE stru :
STRUCT
  ivar:INT;
  strvar:STRING{template list=vor11 [member=struvar1 accesslevel=low]
};
END_STRUCT
END_TYPE
```

3. Вхождения в списки типа 'Instance'(для массивов переменных структуры или функционального блока)

(а) из раздела объявлений программ и списков глобальных переменных:

При объявлении массивов, функциональных блоков или элементов структуры можно задать вхождение в список типа 'Instance':

```
{instance list=<name> template=<template> baseindex=<index> basesubindex=<subindex> [
<key>=<значение первого элемента> <key>=< значение первого элемента> ...прочие ключи пер-
вого элемента ] | [ <key>=<значение второго элемента> <key>=< значение второго элемента > .. прочие
ключи второго элемента ] | [ключи последующих элементов]}<key>=<value> <key>=<value>
...следующие ключи ] }
```

Для массивов ключ "template" будет определен автоматически с помощью неявного шаблона "ARRAY". Для структур и функциональных блоков соответствующий шаблон должен быть создан в Менеджере параметров и должен быть частью заданного здесь определения.

Для каждого отдельного элемента структуры, массива или функционального блока может быть предопределено собственное индивидуальное вхождение в список параметров. Например: вместо единого определения "name" можно указать собственное определение [name=<имя_элемента>] для каждого элемента.

Определение ключей для каждого отдельного элемента (заключается в квадратные скобки) выполняется в строке, разделенной пробелами. Элементы такого пакета определений автоматически получают индексы в возрастающем порядке. Если нет достаточного числа определений ключей для всех элементов массива, структуры или функционального блока, то оставшиеся элементы получают значения от последнего описанного индивидуального элемента (исключение для ключа "name" описано выше)! (см. пример 1b).

Автоматизация ключа "name" при включении массивов в список параметров:

- Если вы не определили имя элемента массива в директиве, то этот и все последующие элементы автоматически получают имена:

<Имя POU>_<Имя переменной массива>_<соответствующий числовой индекс массива >.

Пример: Переменная ARRVAR [1..8] типа массив INT в PLC_PRG включается в список директивой pragma. Если ключ "name" не определен в директиве, то элементы массива в списке параметров автоматически получают имена от "PLC_PRG_arrvar_1" до "PLC_PRG_arrvar_8".

- Если вы определяете директивой некоторое имя "<имя>_<первый числовой индекс массива>" для первого элемента, то последующие элементы массива будут автоматически получать имена "<имя>_<соответствующий числовой индекс>".
Пример: Переменная ARRVAR [1..8] типа массив INT включается в список посредством директивы pragма. Директива указывает [name=xyz_1]" только для первого элемента массива. Последующие элементы автоматически получают имена от "xyz_2" до "xyz_8".

Внимание: Не нужно задавать значения для ключа "Member"; данная колонка заполняется автоматически, при помощи значений индексов массива.

Примеры:

Пример 1а:

Массив "arr_1" включает список "arrinst" типа 'Instance'; все элементы массива в этом списке получают символическое имя "xname_<index number>" (может быть изменено в Менеджере параметров), подиндекс увеличивается на единицу для каждого элемента, начиная с 0. Accesslevel=low будет применен ко всем элементам.

```
arr_1: ARRAY [1..8] OF INT{instance list=arrinst template=ARRAY baseindex=16#0
basesubindex=16#0 [name=xname_1 accesslevel=low ]};
```

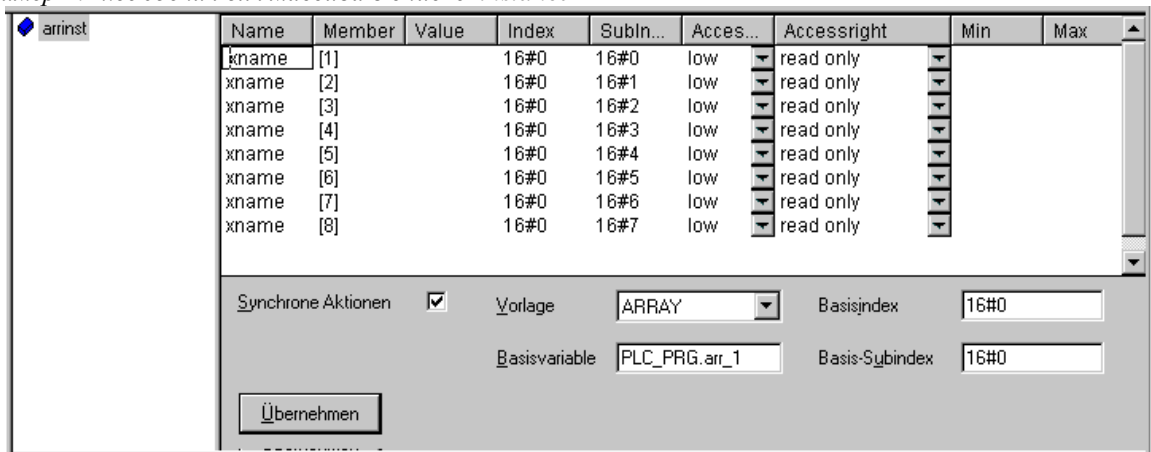
Пример 1б:

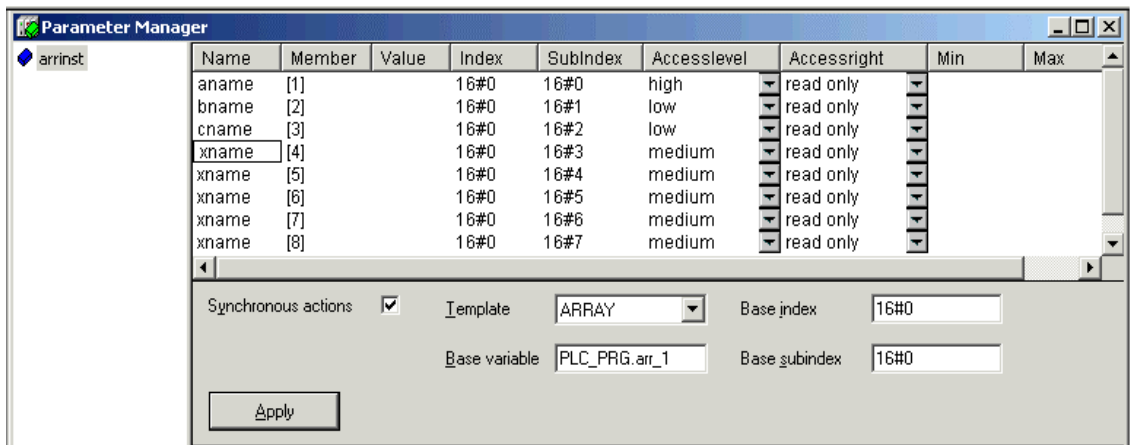
Для массива arr_1 только имена для элементов с 1 по 4 определяются директивой, элементы с 5 по 8 получают имена автоматически на основе определения для элемента 4, то есть "xname_5" и т.д. до "xname_8".

Заметьте, что задания последующих определений ключей для отдельных элементов должны быть включены в эти же квадратные скобки, как показано здесь для прав доступа первого и четвертого элементов:

```
arr_1: ARRAY [1..8] OF INT{instance list=arrinst template=ARRAY baseindex=16#0
basesubindex=16#0 [name=aname accesslevel=high] [name=bname]
[name=cname] [name=xname accesslevel=medium]};
```

Пример 1: Вхождения для массива в списке instance



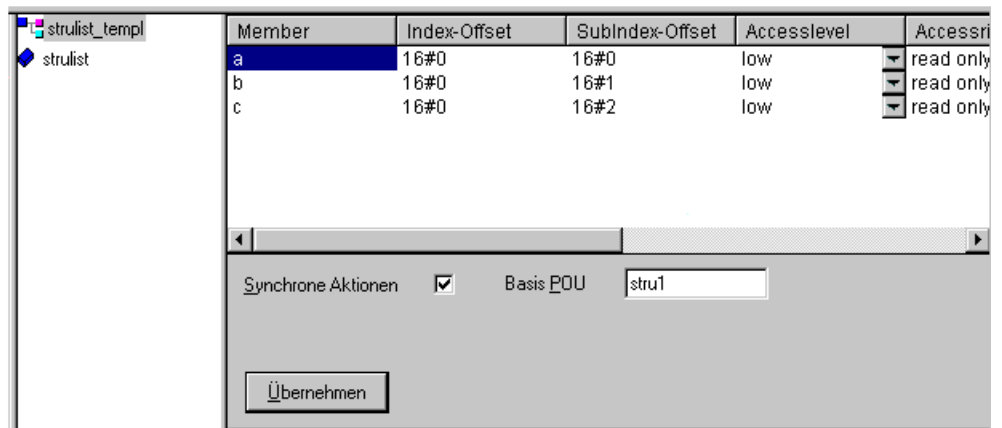
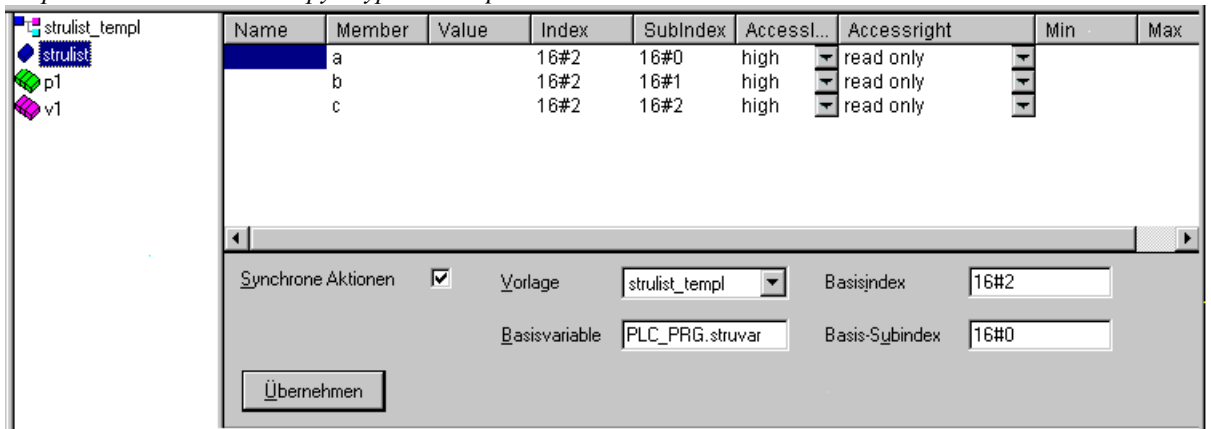


Пример 2:

Элементы структуры типа "stru1" (состоящей из переменных a,b,c) входят в список типа 'Instance', базирующийся на шаблоне "strulist_temp"; список включает вхождения a,b,c, символические имена не присваиваются, уровень доступа "high" и значение каждого индекса определяется по шаблону, через 2. Убедитесь, что определенный в директиве шаблон доступен Менеджеру параметров:

```
struvar:stru1{instance list=strulist template=strulist_temp1 baseindex=16#2
basesubindex=16#0 [accesslevel=high] };
```

Пример 2: Вхождения для структуры в Template



(б) из объявления интерфейса VAR_CONFIG:

Вы можете определить вхождения определяемых переменных в список Instance' непосредственно в окне VAR_CONFIG (вне зависимости от других определений конфигурационных переменных).

Убедитесь, что определенный в директиве шаблон доступен Менеджеру параметров:

```
{instance list=<name> path=<path> template=<template> baseindex=<index>
```

```
basesubindex=<subindex>[ <key>=<value> <key>=<value> ...further keys ] }
```

<path> путь к переменной, например "PLC_PRG.fb1", где fb1 функциональный блок

Пример: Следующая директива в окне VAR_CONFIG будет создавать вхождения для всех переменных для функционального блока "fb1" в списке экземпляров "varinst1" по шаблону "fb1_tmpl". Для каждого вхождения индекс в соответствии с шаблоном увеличивается на 2 (baseindex), подиндекс не изменяется (basesubindex). Каждое вхождение получает символьное имя "fb1var", которое вы можете изменить в Менеджере параметров:

```
VAR_CONFIG
{instance list=varinst1 path=PLC_PRG.fb1 template=fb1_tmpl baseindex=16#2
basesubindex=16#0 [ name=fb1var ]}
END_VAR
```

Директива исключения данных из PERSISTENT

По умолчанию даже если только одна локальная переменная функционального блока или структуры объявлена как PERSISTENT, то все данные экземпляры автоматически будут сохраняться системой исполнения в энергонезависимой области persist.dat. С целью ее экономии можно применить директиву: *{nonpersistent}* в объявлении функционального блока или структуры. В этом случае, объявление PERSISTENT будет влиять только на указанные в нем переменные.

Пример:

Для экземпляров данного функционального блока только значения переменных local и fblevel3 будут записываться в энергонезависимую область.

```
FUNCTION_BLOCK FB_Level_2
{nonpersistent}
VAR_INPUT
bvar_in : BOOL;
END_VAR
VAR_OUTPUT
bvar_out : BOOL;
END_VAR
VAR
ivar2 : INT;
END_VAR
VAR PERSISTENT
local : INT := 33;
fblevel3 : FB_Level_3;
END_VAR
```

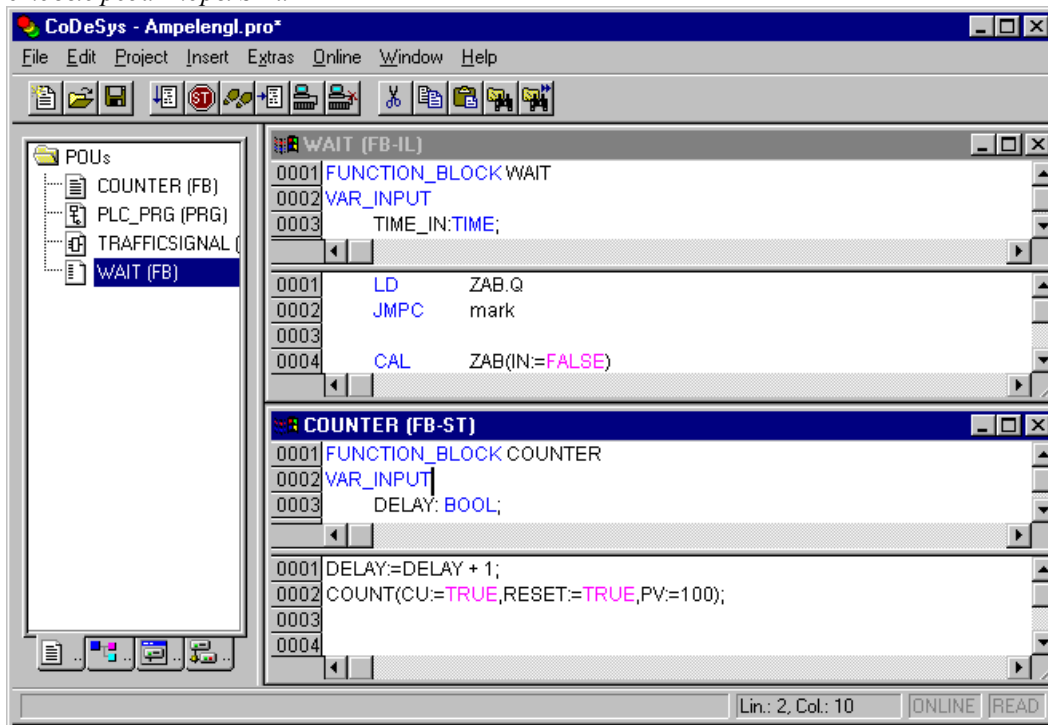
5.3 Текстовые редакторы

5.3.1 Работа в текстовых редакторах

Текстовые редакторы (используемые для написания текстов программ IL и ST) в CoDeSys обеспечивают обычные функции текстовых редакторов Windows. Текстовые редакторы поддерживают цветовое синтаксическое выделение. Корректно введенные инструкции выделяются цветом.

В режиме замены надпись **OV** в статусной строке становится черной. Нажимая клавишу <Ins>, можно переключаться между режимами вставки и замены.

Текстовые редакторы ST и IL



Наиболее важные команды находятся в контекстном меню, которое появляется при щелчке правой кнопки мыши или при нажатии сочетания клавиш <Ctrl> +<F10>.

В текстовых редакторах доступны следующие команды меню:

“Insert” “Operator”

Вызывает список всех доступных для соответствующего языка операторов. Если выбрать оператор из списка и нажать кнопку ОК, то выбранный оператор будет добавлен в текущую позицию курсора.

“Insert” “Operand”

Выводит на экран список всех доступных переменных. Можно выбрать категорию переменных (глобальные, локальные, системные), которые будут изображены в списке.

Если операнд выбран и нажата кнопка ОК, то выбранный операнд будет вставлен в текущую позицию курсора (аналогично работе Input Assistant).

“Insert” “Function”

Выводит диалоговое окно, в котором вы можете выбрать функцию из списка стандартных или определенных пользователем функций.

Выбранная функция помещается в текущую позицию курсора после нажатия кнопки ОК.

Если выбран флаг With Argument, то также будут вставлены необходимые входные и выходные переменные.

“Insert” “Function Block”

Выводит список всех доступных в проекте функциональных блоков. Вы можете выбрать, какие функциональные блоки будут отображены: либо стандартные, либо определенные пользователем.

Выбранный функциональный блок помещается в текущую позицию курсора при нажатии клавиши ОК.

Если выбран флаг With Arguments, то появятся необходимые входные и выходные переменные.

Вызов POU с выходными параметрами

В текстовых языках ST и IL выходные параметры POU можно связать с какими-либо переменными прямо при вызове POU.

Пример: Выходной параметр out1 присваивается переменной a.

```
IL:    CAL afbinst(in1:=1, out1=>a)
ST:    afbinst(in1:=1, out1=>a);
```

Если POU вводится посредством Ассистента ввода (<F2>) с опцией 'With arguments', то вызов в ST или IL автоматически отображается с таким синтаксисом для всех параметров. Однако вы не обязаны все их использовать.

Текстовые редакторы в режиме Online

Текстовые редакторы CoDeSys совмещают типовые функции современных отладчиков. В текстовых редакторах поддерживаются такие Online-функции, как установка точек останова и выполнение программы по шагам.

В режиме Online окно текстового редактора разделяется по вертикали на две части. В левой части окна вы найдете текст программы, а в правой вы увидите значения переменных. Ширину частей можно изменять, перетаскивая мышкой границу между ними.

Просмотр значений переменных осуществляется так же, как и в редакторе раздела объявлений.

Когда связь с контроллером установлена, на экран выводятся текущие значения переменных:

При мониторинге выражений выводится итоговое значение. Например: a AND b отображается со строкой “:=TRUE”, если a и b истинны.

Для бит адресуемых переменных выводится значение соответствующего бита (например, a.2 изображается со строкой “:=TRUE”, если a имеет значение 4).

Если поместить указатель мыши на переменную, то во всплывающей подсказке будет выведен комментарий, тип и адрес переменной.

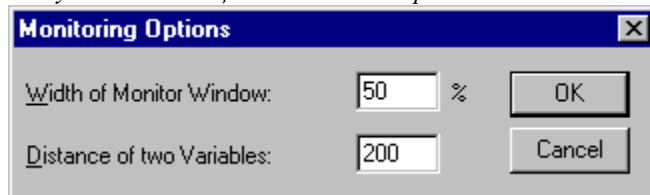
“Extras” “Monitoring Options”

Эта команда позволяет изменить настройки окна, в котором вы просматриваете значения переменных. В текстовых редакторах во время мониторинга окно разделяется на две части. Текст программы находится в левой части, а просматриваемые переменные в правой части окна.

Вы можете установить ширину Width окна монитора и интервал Distance между двумя переменными в строке. Значение интервала, равное 1 соответствует высоте выбранного шрифта.

Обратите внимание, что ширину окон можно оперативно изменять, перетаскивая мышкой границу между ними.

Диалог установки опций окна мониторинга



Точки останова

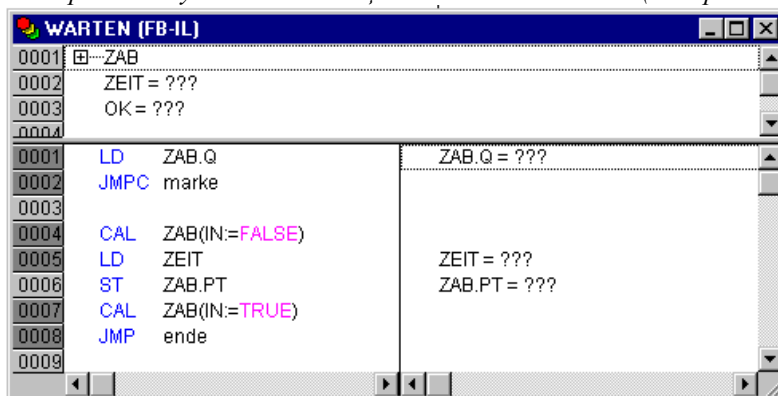
Т.к. в CoDeSys несколько строк на ПЛ объединяются при компиляции, то точки останова нельзя устанавливать в произвольной строке. Точки останова устанавливаются там, где могут измениться либо значения переменных, либо направление выполнения программы. Исключение составляют точки вызова функции. Здесь также можно поставить точку останова. В позициях, находящихся между выше перечисленных, точка останова не имела бы смысла, т. к. здесь не изменяются ни данные, ни направление выполнения программы.

В языке ПЛ точки останова можно ставить в следующих позициях:

- § В начале каждого POU.
- § На каждом операторе LD,LDN
- § На каждом операторе JMP, JMPC, JMPCN
- § На каждой метке
- § На каждом операторе CAL, CALC, CALCN
- § На каждом операторе RET, RETC, RETCN
- § В конце каждого POU
- § Язык ST допускает следующие позиции точек останова:
- § На каждой инструкции присваивания
- § На любой инструкции RETURN и EXIT.
- § В позициях, где вычисляются условия(WHILE, IF,REPEAT)
- § В конце POU

При установке точки останова, номер соответствующей строчки выделяется цветом, выбранным в опциях проекта.

Редактор IL с допустимыми позициями точек останова (номера таких строк выделены темно-серым)



Как поставить точку останова?

Для того чтобы поставить точку останова, щелкните мышкой по номеру строки, в которой вы хотите поставить точку останова. Цвет номера строки поменяется с темно-серого на голубой, и точка останова будет установлена в PLC.

Удаление точек останова.

Для этого щелкните по номеру строки, в которой установлена точка останова.

Устанавливать и удалять точки останова также можно через меню (“Online” “Toggle Breakpoint”), нажимая кнопку <F9> или кнопку на панели инструментов.

Что происходит в точках останова.

Когда точка останова будет достигнута, номер выделенной строки станет красным. Программа будет остановлена в PLC.

Если программа остановлена, то ее выполнение можно продолжить командой “Online” “Run”.

Кроме того, вы можете воспользоваться командами “Online” “Step over” и “Step in” для выполнения программы по шагам. Если пользоваться командой “Step over”, программа не будет останавливаться в точках, вызываемых POU. При вызове команды “Step in” вы будете по шагам проходить все вызываемые POU.

Номер строки в текстовом редакторе.

Номер строки в текстовом редакторе определяет номер строки текста POU.

В режиме Offline щелчок по определенному номеру строки приводит к выделению текстовой строки.

В режиме Online цвет номера строки определяет, установлена точка останова в этой строке или нет. Вот стандартные установки для каждого цвета:

темно-серый: Строка, в которой можно установить точку останова.

голубой: Точка останова установлена в этой строке.

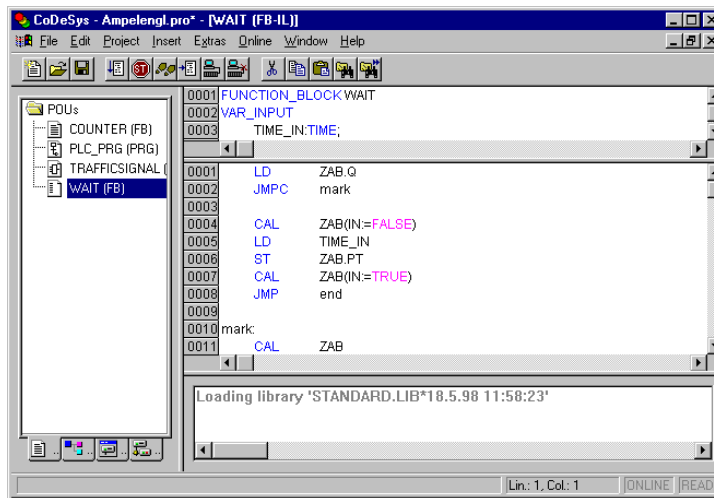
красный: Программа остановлена в этой точке.

В режиме Online щелчок мышки по номеру строки позволяет установить или удалить точку останова.

5.3.2 Редактор языка IL

Вот так выглядит программа, написанная на IL в соответствующем редакторе CoDeSys:

Редактор IL



Все редакторы POU состоят из раздела объявлений и собственно тела программы, отделенных друг от друга разделителем.

Редактор IL – это текстовый редактор с обычными функциями текстового редактора Windows.

В CoDeSys допустим вызов POU с несколькими вложенными вызовами:

Пример:

```

CAL CTU_inst(
    CU:=%IX10,
    PV:=(
        LD A
        ADD 5
    )
)

```

Информацию, касающуюся языка программирования, можно найти в главе 2.2.1. Instruction List (IL).

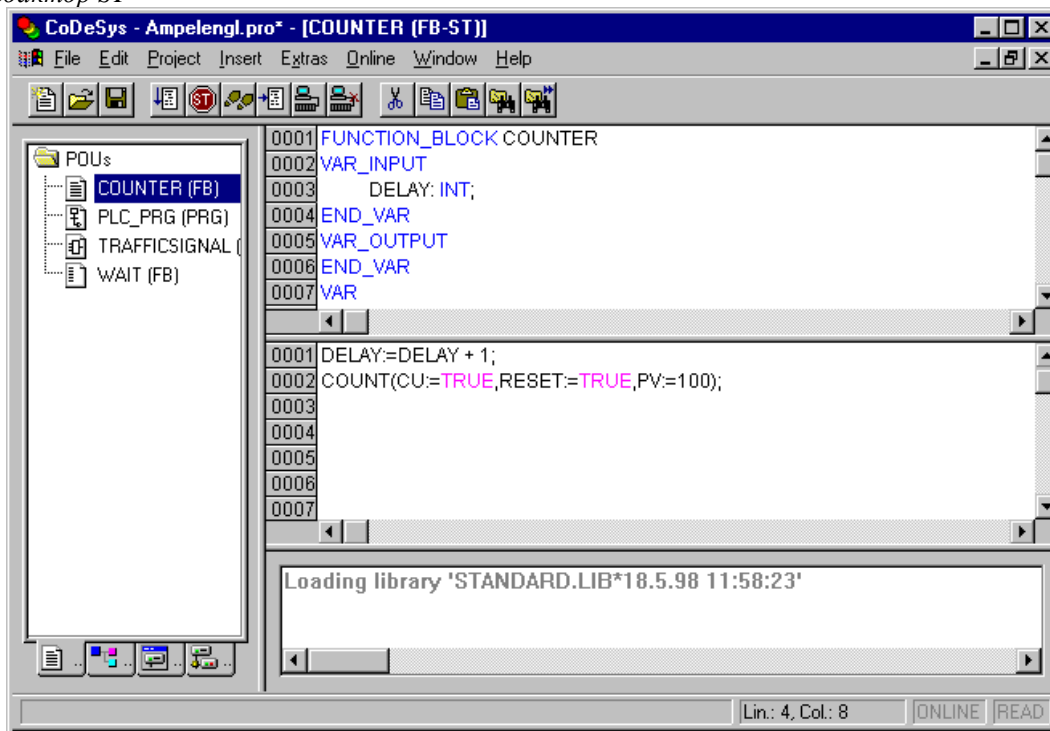
IL в режиме Online.

Если вызвать команду “Online” “Flow control”, в левой части каждой строки появится поле, содержащее значение аккумулятора. Более подробная информация о редакторе IL в режиме Online описана выше в разделе “Текстовые редакторы в режиме Online”.

5.3.3 Редактор языка ST

Вот так выглядит программа, написанная на ST в соответствующем редакторе CoDeSys

Редактор ST



Все редакторы POU состоят из раздела объявлений и собственно тела программы, отделенных друг от друга разделителем.

Редактор ST – это текстовый редактор с обычными функциями текстового редактора Windows.

Информация о редакторе ST в режиме Online описана выше в пункте “Текстовые редакторы в режиме Online”.

Информацию, касающуюся языка программирования, можно найти в главе 2.2.2. Structured Text (ST).

5.4 Графические редакторы

5.4.1 Работа в графических редакторах

Графические редакторы предназначены для графических языков SFC, LD, FBD и CFC, они имеют много общих черт. В следующих параграфах будут описаны эти черты и отдельно будет рассказано о редакторах языков LD, FBD, CFC и SFC.

Масштаб (Zoom)

Графические элементы в языках SFC, LD, FBD, CFC и в визуализациях могут менять свои размеры. Все элементы, относящиеся к исполняемой части программы, в отличие от раздела объявлений меняют свои размеры при использовании функций масштабирования.

По умолчанию любой объект изображается с коэффициентом масштабирования 100%. При сохранении проекта коэффициент масштабирования сохраняется.

Распечатка проекта на принтер всегда происходит с масштабом 100%.

Коэффициент масштабирования можно выбрать на панели инструментов в выпадающем списке, в котором доступны коэффициенты от 25% до 400%. Вручную можно вводить коэффициенты от 10% до 500%.

Устанавливать масштаб можно, только если выбран графический объект или объект визуализации.

Размер текста изменяется пропорционально коэффициенту масштабирования и установленному размеру шрифта.

Объект, получаемый при выполнении любых функций меню, например, вставка объекта, будет иметь текущий масштаб. В режиме Online каждый объект изображается в соответствии с установленным коэффициентом масштабирования; все функции Online доступны без ограничений.

При использовании мыши со скроллингом изменять масштаб объекта можно, одновременно нажимая клавишу <Ctrl> и вращая колесико.

Цепь

В редакторах LD и FBD программа представлена в виде списка цепей. Каждая цепь состоит из двух частей: в левой записан номер цепи, а в правой – структура, состоящая из логических или арифметических операций, вызовов программ, функций или функциональных блоков, инструкций перехода или возврата.

Метка

Каждая цепь может иметь метку, по умолчанию она отсутствует. Метку можно поставить, если щелкнуть по первой строке цепи, прямо за номером цепи. После этого можете вводить имя метки, оканчивающееся двоеточием.

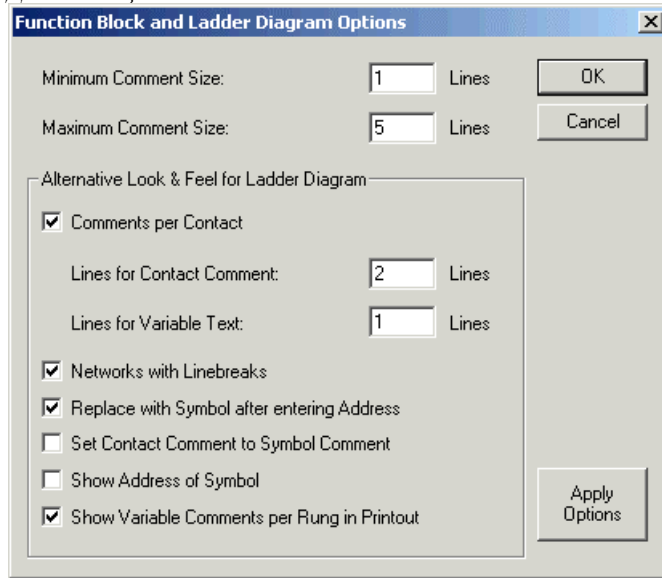
Комментарии к схеме, команда “Extras” ‘Options’

В редакторах релейных и функциональных блоковых диаграмм любая цепь (схема) может иметь комментарий в одну или несколько строк. В настройках “Extras” ”Options” задаются опции отображения комментариев.

В поле ввода **maximum comment size** вы можете ввести максимальное количество строк под комментарий (по умолчанию это значение равно 4). В поле **minimum comment size** можно ввести минимальное количество строк, выделяемых под комментарий. Если, например, в этом поле ввести число 2, то в начале схемы после строки, где располагается метка, будут находиться две пустых строки. По умолчанию это значение 0, что позволяет разместить больше схем на экране.

Если минимальное число строк комментария больше нуля, то щелкнув мышью по строке комментария, выберите ее и введите нужный текст. Есть и другой способ: выделите схему и выберите команду “**Insert**” “**Comment**”. Для зрительного выделения в тексте комментария изображаются серым цветом.

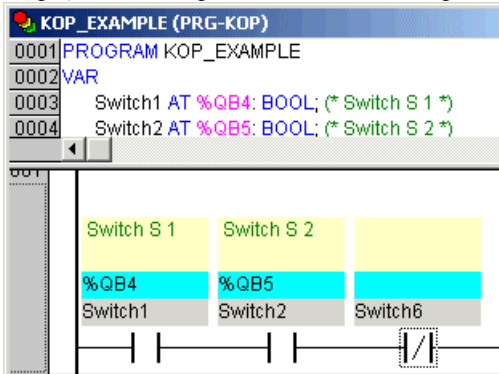
Диалог опций LD и FBD:



Alternative Look & Feel: данные опции позволяют определить альтернативный вид отображения цепей.

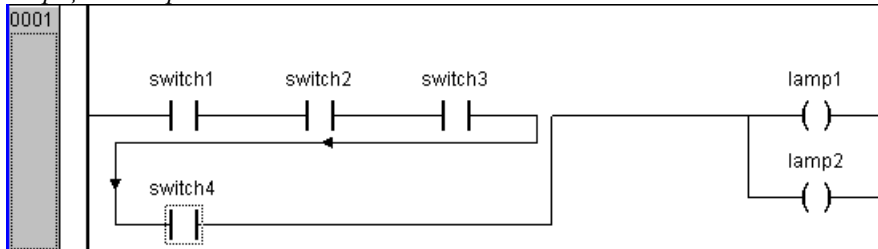
В редакторе релейных схем существует возможность снабдить комментариями отдельные контакты и обмотки. Для этого необходимо включить опцию **Comments per Contact** и вставить в поле **Lines for Contact Comment** число строк, которые нужно зарезервировать для отображения таких комментариев. **Lines for Variable text** определяет число строк, которое отводится под имя переменной, связанной с контактом или обмоткой. Это необходимо при отображении длинных имен путем переноса текста на новую строку.

Пример цепи с отображением комментария и адреса переменной:



Networks with Linebreaks включает в редакторе релейных схем режим автоматического переноса цепи при отображении, если она не умещается в заданном окне.

Пример цепи с переносами



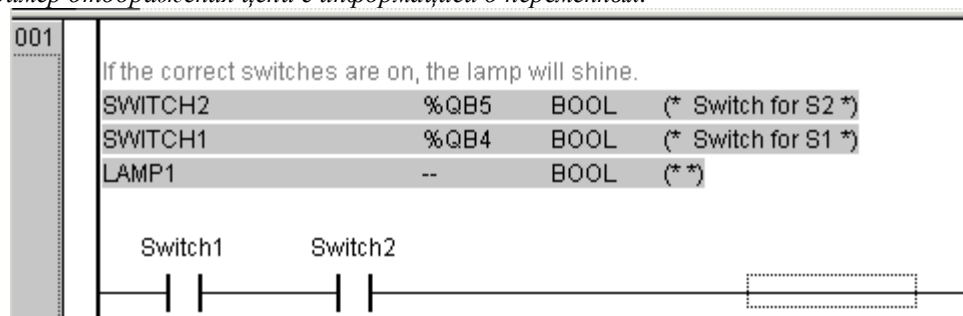
Replace with Symbol after entering Address: (только для релейных схем): Если данная опция активна, то для контакта или обмотки в соответствующем поле можно вводить адрес (например "%QB4"). Адрес будет сразу же заменен на имя переменной, имеющей указанный адрес. Если такая переменная неопределена то, будет отображаться адрес.

Set Contact Comment to Symbol Comment: Если данная опция активна, то в поле комментария контакта или обмотки будет отображаться комментарий соответствующей переменной. (См. выше: «Пример цепи с отображением комментария и адреса переменной») Такой комментарий можно редактировать, если включена опция 'Comments per Contact'. Обратите внимание, если комментарий для обмотки или контакта уже был задан локально, то он будет заменен автоматически на комментарий переменной, даже если переменная не получила комментария при объявлении!

Show Address of Symbol: (только для релейных схем): Если данная опция активна и переменная обмотки или контакта имеет адрес, то данный адрес будет отображаться над именем переменной. (См. выше: «Пример цепи с отображением комментария и адреса переменной»).

Show Variable Comments per Rung in Printout: Если данная опция активна, то в каждой цепи для всех используемых в ней переменных будут добавлены строки, включающие имя, адрес, тип и комментарий переменной. Это удобно при создании печатной документации.

Пример отображения цепи с информацией о переменных:



Подтверждение ввода:

OK: нажмите эту кнопку для того, чтобы принять заданные опции для текущего POU и закрыть диалог.

Apply options: нажмите эту кнопку для того, чтобы принять заданные опции для всего проекта. Такое действие требует подтверждения. Для этого будет открыто вспомогательное диалоговое окно.

“Insert” “Network(after)” или “Insert” “Network(before)”

Быстрый ввод команды **“Network(after)”**: <Shift> +<T>

Если вы хотите вставить новую цепь в редакторах FBD или LD, то используйте команды **“Insert” “Network(after)”** и **“Insert” “Network(before)”** для вставки цепи после или перед выбранной цепью соответственно. Чтобы выбрать схему, щелкните мышью на нужной вам цепи. Номер текущей цепи выделяется прямоугольником с пунктирной границей. Если нажать <Shift>, то можно выделить сразу несколько схем, щелкая мышью по каждой.

Графические редакторы в режиме Online

В режиме Online в графических редакторах FBD и LD вы можете устанавливать точки останова. Номер цепи, в которой вы установили точку останова, изображается синим. Программа останавливается перед такой цепью, после чего номер схемы становится красным. Если вы используете команду **Step in** или **Step out**, то выполняется одна цепь и программа останавливается.

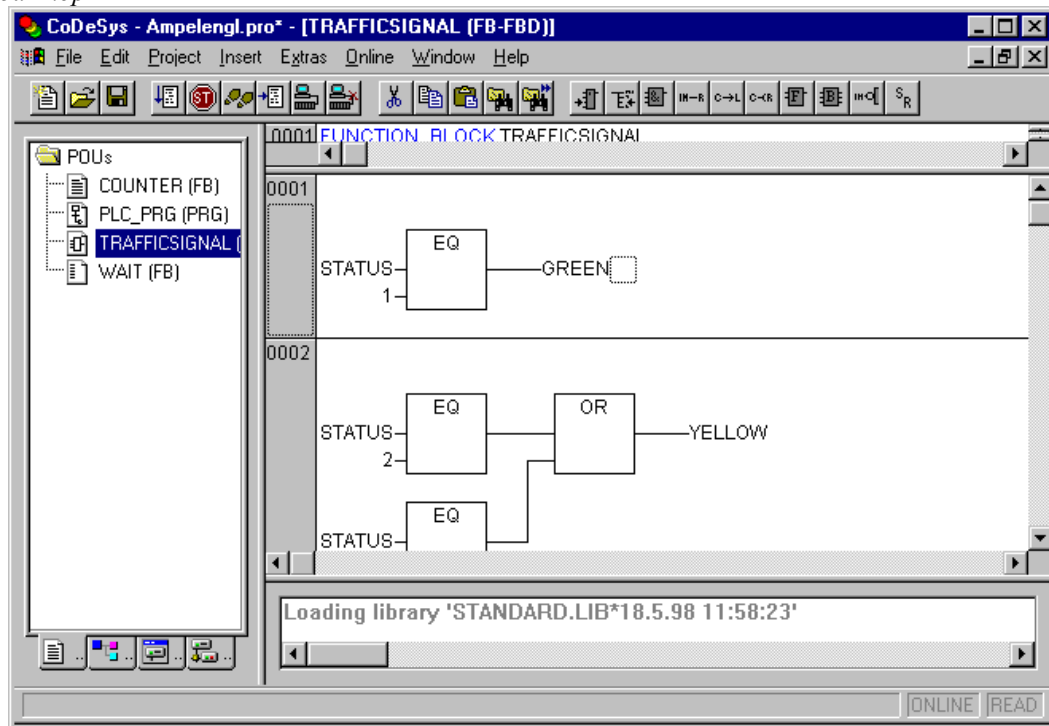
При мониторинге выражений выводятся значения переменных, входящих в выражение. Например: $a \text{ AND } b$ выводится со строкой “:=TRUE”, если a и b истинны. В случае бит адресуемых переменных выводится значение соответствующего бита (например, **a.2** изображается со строкой “:=TRUE”, если **a** имеет значение 4).

Контроль потока выполнения программы запускается с помощью команды “Online” ”Display Flow Control”. Используя эту команду, вы можете просмотреть значения, передаваемые по линиям соединения. Если линии соединения передают не логические значения, то эти значения изображаются в отдельных полях. Поля для переменных, которые не используются, изображаются серым. Если линия передает значение TRUE, то она изображается синим.

В режиме Online, если вы поместите указатель мыши на переменную, то в подсказке появятся тип, комментарии и адрес этой переменной (если они определены).

5.4.2 Редактор FBD

Редактор FBD



Редактор FBD - графический редактор. Он работает со списком цепей, каждая из которых состоит из логических или арифметических выражений, вызовов функций, программ или функциональных блоков, инструкций возврата и перехода.

Наиболее важные функции вы можете найти в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl>+<F10>.

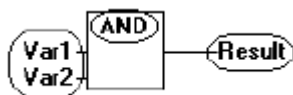
Обратите также внимание на возможности управления отображением комментариев, описанные выше в разделе 5.4.1 «Комментарии к схеме, команда “Extras” ‘Options’».

Позиция курсора в FBD

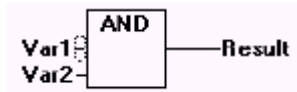
Текстовый курсор может устанавливаться в любую часть FBD цепи, содержащую текст. Выбранный текст выделяется синим и может быть изменен.

Текущую позицию графического курсора можно увидеть по прямоугольнику с пунктирной границей. Далее на примере приводится список всех возможных позиций курсора:

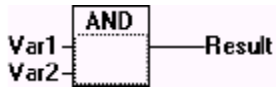
- 1) Любое поле с текстом (обведены черным):



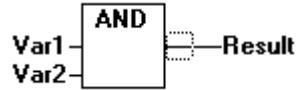
- 2) Любой вход блока:



- 3) Любой оператор, функция или функциональный блок.



- 4) Выход блока, если к нему присоединена переменная или инструкция перехода.



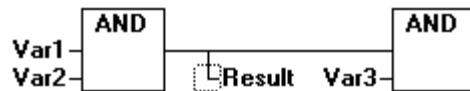
- 5) Пересечение линий над присваиванием, переходом или возвратом:



- 6) Место за самым правым объектом схемы (“последняя позиция курсора”; используется для выбора всей схемы)



- 7) Пересечение линий перед переменной:



Установка позиции курсора

Позицию курсора можно установить с помощью левой кнопки мыши или с помощью клавиатуры.

При использовании клавиш перемещения вы будете менять текущую позицию курсора на соседнюю в заданном направлении. При использовании этого способа можно выбрать любую позицию курсора, в том числе и текстовое поле. Клавиши вверх и вниз позволяют выбрать предыдущую и следующую позицию курсора.

Пустая схема содержит только три знака вопроса "???". Такую схему можно выбрать, щелкнув на них мышью.

“Insert” “Assign”

Обозначение:  Быстрый ввод: <Ctrl>+<A>

Эта команда вставляет инструкцию присваивания в схему.

В зависимости от позиции курсора (см. выше “Позиции курсора в FBD”) присваивание будет вставлено прямо перед выбранным входом (позиция курсора №2), перед выходом (позиция курсора №4) или в конце схемы (позиция курсора №6). После вставки присваивания появятся три знака вопроса, выделив которые, можно вводить имя переменной. Имя переменной удобно вводить с помощью Ассистента ввода (F2).

Обратите также внимание на возможность ввода адресов вместо имен переменных (См. 5.4.1 Комментарии к схеме, команда “Extras” ‘Options’).

Чтобы ввести дополнительное присваивание к существующему, используйте команду “Insert” “Output”.

“Insert” “Jump”

Обозначение:  Быстрый ввод: <Ctrl>+<L>

Эта команда вставляет инструкцию перехода.

В зависимости от позиции курсора (см. выше “Позиции курсора в FBD”) инструкция перехода будет вставлена прямо перед выбранным входом (позиция курсора №2), перед выходом (позиция курсора №4) или в конце схемы (позиция курсора №6).

После вставки инструкции перехода появятся три знака вопроса, выделив которые, можно вводить имя метки.

“Insert” “Return”

Обозначение:  Быстрый ввод: <Ctrl>+<R>

Эта команда вставляет инструкцию возврата Return.

В зависимости от позиции курсора (см. выше “Позиции курсора в FBD”) инструкция возврата будет вставлена прямо перед выбранным входом (позиция курсора №2), перед выходом (позиция курсора №4), перед пересекающимися линиями из прибора № 5 или в конце схемы (позиция курсора №6).

“Insert” “Box”

Обозначение:  Быстрый ввод: <Ctrl>+

С помощью этой команды в схему можно вставлять операторы, функции, функциональные блоки и программы. Сразу после выполнения этой команды в схеме появляется оператор “AND”. Выбрав текстовое поле, где написано “AND”, этот оператор можно превратить в любой другой объект (функцию, функциональный блок, программу, оператор), написав имя желаемого объекта. Это имя удобно выбирать, используя Input Assistant (<F2>). Если новый блок имеет другое число входов, чем оператор AND, то будут добавлены новые входы или удалены ненужные.

В функциях и функциональных блоках изображаются формальные входные и выходные параметры.

Над функциональными блоками находится поле, в котором нужно ввести имя экземпляра функционального блока. Если тип функционального блока введен не корректно (функциональный блок не описан), то появляется блок, имеющий два входа. Если выбрано поле ввода имени экземпляра функционального блока, то с помощью клавиши <F2> можно вызвать Input Assistant.

Новый POU вставляется в выбранную позицию:


- § Выбран вход блока (позиция курсора №2). В этом случае POU вставляется в позицию перед входом. Первый вход этого POU соединяется с ветвью, ранее соединенной с выбранным входом. Выход POU соединяется с выбранным входом.
- § Выбран выход (позиция курсора №4), тогда POU вставляется после этого выхода. Первый вход этого POU соединяется с выбранным выходом. Выход вставленного POU соединяется с ветвью, ранее соединенной с выбранным выходом.
- § Выбран POU (позиция курсора №3), тогда старый блок будет заменен на новый. Насколько это возможно, новый блок будет присоединен к схеме так же, как и старый. Если новый элемент имеет меньше входов, чем старый, то ненужные ветви будут удалены. То же верно и для выходов.

- § Выбрана инструкция перехода или возврата, тогда POU будет вставлен перед ней. Первый вход этого POU соединяется с ветвью, ранее соединенной слева с выбранным элементом. Первый выход этого POU соединяется с ветвью, ранее соединенной справа с выбранным элементом.
- § Выбрана последняя позиция схемы (позиция курсора №б). Новый POU соединяется с последним блоком схемы.

Все входы POU, которые не удалось соединить автоматически, соединяются с тремя знаками вопроса. Этот текст можно заменить на имя переменной или константу.

Если справа от вставленного POU находится ветвь, то она будет соединена с первым выходом этого POU.

“Insert” “Input”

Обозначение:  Быстрый ввод: <Ctrl>+<U>

Добавляет вход оператора. Некоторые операторы могут иметь переменное число входов (например, ADD может иметь два и более входа).

Для того чтобы добавить вход, выберите уже существующий вход (позиция курсора №1), перед которым вы хотите вставить новый и выполните команду “Insert” “Input”. Есть другой способ: выберите оператор (позиция курсора №3) и выполните команду “Insert” “Input”, тогда новый вход будет самым нижним.

Слева от вставленного входа появится строка “???”. Вместо нее нужно ввести имя переменной или константу, для чего можно воспользоваться Input Assistant.

Обратите также внимание на возможность ввода адресов вместо имен переменных (См. 5.4.1 Комментарии к схеме, команда “Extras” ‘Options’).

“Insert” “Output”

Обозначение: 

Добавляет новое присваивание к уже существующему. Это позволяет передать одно значение сразу нескольким переменным.

Если вы выберете пересечение линий над присваиванием (позиция курсора №5, см. выше “Позиция курсора в FBD”) или выход прямо перед ним (позиция курсора №4), то после уже существующего присваивания будет вставлено новое.

В случае, если линии пересекаются прямо перед выбранным присваиванием (позиция курсора №4), то новое присваивание будет вставлено перед выбранным.

Слева от вставленного присваивания появится строка “???”. Вместо нее нужно ввести имя переменной или константу, для чего можно воспользоваться Input Assistant.

Обратите также внимание на возможность ввода адресов вместо имен переменных (См. 5.4.1 Комментарии к схеме, команда “Extras” ‘Options’).

“Extras” “Negate”

Обозначение:  Быстрый ввод: <Ctrl>+<N>

С помощью этой команды можно инвертировать входы, выходы, инструкции перехода или возврата. Символ отрицания – небольшая окружность на месте соединения.


Если выбран вход (позиция курсора №2), то этот вход будет инвертирован.

То же верно и для выхода.

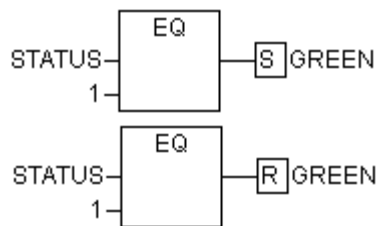
При инвертировании инструкций перехода или возврата они выполняются, если ветвь, к которой они присоединены, передает FALSE.

Снять отрицание можно через повторное отрицание.

“Extras” “Set/Reset”

Обозначение: 

При помощи этой команды вы можете определить Set- и Reset-выходы. Set-выход обозначается буквой S, а Reset-выход – буквой R.

Set/Reset выходы в FBD

Set-выход принимает значение TRUE, а Reset-выход – значение FALSE, если ветвь, к которой они присоединены, передает TRUE. Если эта ветвь передает FALSE, то переменные сохраняют свои значения.

При многократном выполнении этой команды можно получить Set-выход, Reset-выход и обычный выход.

“Extras” “View”

Используя эту команду, можно использовать редактор LD или FBD для программных компонентов (POU) созданных в FBD редакторе. Это возможно как в Offline так и в Online режимах.

Open instance

Команда аналогична команде 'Project' 'Open instance'. Она присутствует в контекстном меню (<F2>) и в меню 'Extras', если курсор установлен на имени функционального блока в графическом или текстовом редакторе.

Команды вырезать (Cut), копировать (Copy), выделить (Paste) и удалить (Delete) в FBD

Эти команды можно найти в меню Edit.

Если выбрано пересечение линий (позиция курсора №5), то присваивания, инструкции перехода или возврата, расположенные под пересекающимися линиями, будут удалены, вырезаны или скопированы.

Когда выбрано POU (позиция курсора №5), то эти действия будут выполнены над выбранным объектом и всеми ветвями, которые соединяют этот объект со схемой.

Кроме того, ветви, полностью расположенные перед позицией курсора, будут вырезаны, удалены или скопированы.

Скопированные или вырезанные части схемы находятся в буфере и могут быть вставлены в нужное место, которое перед этим нужно выбрать. Можно выбирать входы и выходы.

Если POU вставляется из буфера (не забудьте, что в этом случае все соединяющие ветви, кроме первой, хранятся в буфере как единое целое), первый вход соединяется с ветвью перед выбранной точкой.

В другом случае (из буфера вставляется не POU), ветвь, находящаяся перед выбранной точкой, полностью заменяется на содержимое буфера.

В обоих случаях последний вставляемый элемент соединяется с ветвью, расположенной справа от выбранной точки.

Замечание: С помощью вырезания и вставки решается следующая проблема: новый оператор вставляется в середину схемы; ветвь, расположенная справа от оператора, теперь соединяется с первым входом, но может быть соединена со вторым. Вы должны выбрать первый вход и выполнить команду "Edit" "Cut". Затем, выделите второй вход и выполните команду "Edit" "Paste". Теперь ветвь соединится со вторым входом.

FBD диаграмма в режиме Online

В режиме Online в редакторе FBD можно устанавливать точки останова. Если в цепи была установлена точка останова, то номер соответствующей цепи станет синим. Выполнение программы останавливается перед цепью, в которой установлена точка останова. В этом случае номер цепи становится красным. Используя команду "Step in" или "Step over", можно последовательно выполнять цепи, останавливаясь после каждой.

На экран выводится текущее значение каждой переменной. Исключение составляет тот случай, когда вход функционального блока – это выражение. Тогда выводится только значение первой переменной в выражении.

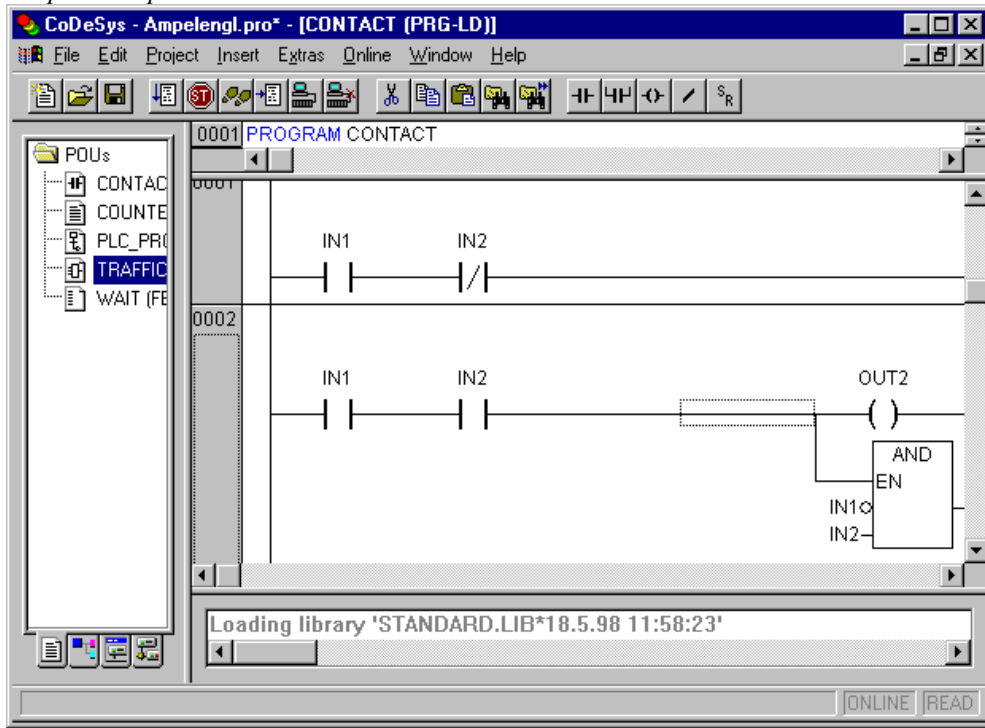
Двойной щелчок мышью по переменной выводит диалоговое окно для ввода нового значения переменной. Если переменная является логической, то диалоговое окно не выводится, а значение переменной просто переключается. Для записи значения переменных в контроллер используется команда "Online" "Write values". После этого переменные снова становятся черными.

Контроль потока выполнения программы запускается с помощью команды "Online" "Display Flow Control". Используя эту команду, вы можете просмотреть значения, передаваемые по линиям соединения. Если линии соединения передают не логические значения, то эти значения изображаются в отдельных полях. Поля для переменных, которые не используются, изображаются серым цветом. Если линия передает значение TRUE, то она изображается синим. Эта команда позволяет наблюдать за потоком информации во время выполнения программы.

В режиме Online, если вы переместите указатель мыши на переменную, то в подсказке появится тип, комментарии и адрес этой переменной.

5.4.3 Редактор LD

POU в редакторе LD



Все редакторы POU состоят из раздела объявлений и собственно тела POU. Они отделены друга от друга разделителем.

Редактор LD – это графический редактор. Наиболее важные команды находятся в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl>+<F10>.

Обратите также внимание на возможности управления отображением комментариев, описанные выше в разделе 5.4.1 «Комментарии к схеме, команда “Extras” ‘Options’».

Информацию по языку можно найти в главе 2.2.6 Ladder Diagram(LD).

Позиции курсора в редакторе LD

Курсор может находиться в следующих позициях, причем для любой позиции контакт можно заменить функциональным блоком или программой. POU, имеющий вход EN, используется также как и в FBD. Информация о редактировании таких схем находится в главе “Редактор FBD”.

Любое текстовое поле (позиция курсора обозначена черной рамкой)



Любой контакт или функциональный блок.



Любая обмотка.



Линия, соединяющая контакт и обмотку.

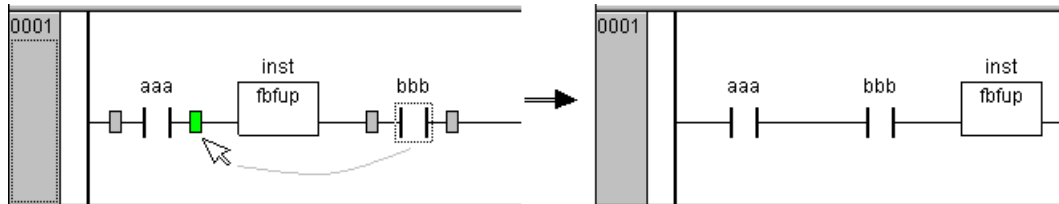


В редакторе LD используются следующие специальные команды:

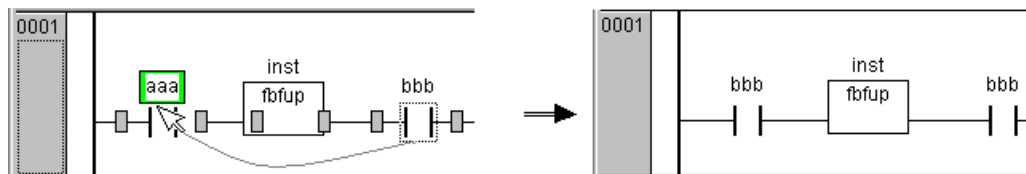
Перемещение элементов и наименований в редакторе LD

С помощью перетаскивания мышкой (drag&drop) элементы (контакт, обмотку или функциональный блок) или их наименования в LD можно перемещать в другие позиции.

Выберите нужный элемент (контакт, обмотку или функциональный блок) и перетаскивайте его, удерживая нажатой клавишу мышки. В процессе этого все допустимые места для помещения элемента будут показаны серыми прямоугольниками. Перетащите элемент в одну из этих позиций и отпустите клавишу. Элемент будет перемещен.



Если вы перетащите элемент в поле имени другого элемента, то данное поле будет подсвечено зеленым цветом. Если теперь отпустить клавишу мышки, то имя в поле будет заменено «перетаскиваемым» именем. Если включено отображение адреса и комментария (опция), то они также будут скопированы.




'Insert' 'Network (before)'

Обозначение: 

Используйте эту команду для вставки цепи, выше выбранной в редакторе LD.

'Insert' 'Network (after)'

Обозначение: 

Используйте эту команду для вставки цепи, ниже выбранной в редакторе LD.

'Insert' 'Contact'

Обозначение:  Быстрый ввод: <Ctrl>+<K>


Используйте эту команду для вставки контакта перед выбранной позицией в цепи.

Если выбрана обмотка (позиция курсора №3) или линия, соединяющая контакт и обмотку (позиция курсора №4), то новый контакт вставляется последовательно с предыдущим.

Текстовое поле над контактом заполняется знаками вопроса. В этом поле надо ввести нужную переменную или константу. Имя переменной удобно вводить с помощью Input Assistant.

Обратите также внимание на возможность управления отображением и ввода адресов вместо имен переменных (См. 5.4.1 Комментарии к схеме, команда "Extras" "Options").

'Insert' 'Contact (negated)'

Обозначение:  Быстрый ввод: <Ctrl> + <G>

Используйте эту команду для вставки инверсного контакта. Она заменяет последовательность команд 'Insert' 'Contact' и 'Extras' 'Negate'.

'Insert' 'Parallel Contact'

Обозначение:  Быстрый ввод: <Ctrl>+<R>

Используйте эту команду для вставки контакта, параллельного выделенной позиции схемы.

Если выделена обмотка (позиция курсора №3) или линия, соединяющая обмотку и контакт (позиция курсора №4), то новый контакт будет параллелен всем контактам, находящимся слева от выбранной позиции.

В текстовом поле над вставленным контактом записаны три знака вопроса. В этом поле надо ввести нужную переменную или константу. Имя переменной удобно вводить с помощью Input Assistant.

'Insert' 'Parallel Contact (negated)'

Обозначение:  Быстрый ввод: <Ctrl> + <O>

Используйте эту команду для вставки инверсного контакта. Она заменяет последовательность команд 'Insert' 'Parallel Contact' и 'Extras' 'Negate'.

'Insert' 'Coil'

Обозначение:  Быстрый ввод: <Ctrl>+<L>

Вы можете использовать эту команду для вставки обмотки, параллельной выбранной.

Для этого нужно выделить обмотку (позиция курсора №3) или линию, соединяющую контакты и обмотки (позиция курсора №4), и выполнить команду. В этом случае вставленная обмотка будет самой нижней. Если выделена обмотка, то новая вставляется прямо над выбранной.

По умолчанию переменная, связанная с обмоткой, получает имя “???”, которое можно заменить на любую константу, переменную или адрес. Для этого удобно использовать Input Assistant.

Существует возможность отображать имена переменных с разрывом строк. Обратите также внимание на возможность ввода адресов вместо имен переменных (См. Комментарии к схеме, команда "Extras" 'Options').

'Insert' "Set' coil'

Обозначение:  Быстрый ввод: <Ctrl> + <I>


Используйте эту команду для вставки 'Set' обмотки, параллельной выбранной. Она заменяет последовательность команд 'Insert' 'Coil' и 'Extras' 'Set/Reset'.

'Insert' 'Reset' coil'

Обозначение: 

Используйте эту команду для вставки 'Reset' обмотки, параллельной выбранной. Она заменяет последовательность команд 'Insert' 'Coil' и 'Extras' 'Set/Reset'.

'Insert' 'Function Block'

Обозначение:  Быстрый ввод: <Ctrl>+

Эта команда используется для вставки оператора, функционального блока, функции или программы.

Для этого нужно выделить обмотку (позиция курсора №3) или линию, соединяющую контакты и обмотки (позиция курсора №4), и выполнить команду. Новый блок имеет имя AND. При необходимости вы можете поменять имя этого блока на любое другое. Для этого удобно использовать Input Assistant, в котором можно выбрать стандартное или определенное пользователем POU.

Первый вход и первый выход этого POU соединяется с линией связи, поэтому этот выход и выход должны быть типа BOOL. Текстовые поля имен переменных для других входов и выходов POU заполняются тремя знаками вопроса, которые можно заменить на любые константы, переменные или адреса. Для этого удобно использовать Input Assistant.

Обратите также внимание на возможность ввода адресов вместо имен переменных (См. Комментарии к схеме, команда "Extras" 'Options').

POU с входом EN.

Если вы хотите управлять вызовом POU из релейной цепи, то POU должен иметь логический вход разрешения EN.

Команду для вставки POU с EN-входом вы найдете в меню “Insert” “Box with EN”. Операторы, функциональные блоки, программы или функции с EN-входом работают так же, как и в FBD. Вход EN соединяется с линией, связывающей обмотки и контакты. POU выполняется, когда линия, к которой подключен EN-вход, передает значение TRUE.

Разрешенный по EN POU будет работать как обычно. Это означает, что соответствующие данные будут передаваться в POU и обрабатываться им.

Таким образом, если вы хотите вставить FBD фрагмент в релейную цепь, вы должны вставить оператор с EN- входом. После него продолжайте схему как в редакторе FBD.

В итоге составленная цепь будет похожа на схему в FBD.

'Insert' 'Box with EN'

Обозначение: 

Используйте эту команду для вставки функционального блока, оператора, функции или программы с EN-входом в схему LD.

Выделенная позиция должна быть обмоткой (позиция курсора №3) или линией, соединяющей обмотку и контакт (позиция курсора №4). Новое POU вставляется параллельно обмоткам или ниже их и по умолчанию имеет имя "AND". Вы можете поменять это имя на любое другое. Для этого удобно использовать Input Assistant.

'Insert' 'Insert at blocks'

С помощью этих команд вы можете вставить дополнительные элементы в уже существующие POU. Команды, находящиеся в этом пункте меню, применимы в тех же позициях курсора, что и соответствующие команды FBD (см. главу 5.7).

Команда **Input** добавляет новый вход в POU.

Команда **Output** добавляет новый выход в POU.

Команда **POU** добавляет новый POU в схему.

Команда **Assign** вставляет присваивание переменной. Задайте имя переменной вместо трех вопросов или используйте ассистент ввода.

'Insert' 'Rising edge detection'

Обозначение: 

Данная команда вставляет в цепь функциональный блок R_TRIG, который служит для выделения переднего фронта импульса (FALSE -> TRUE) сигнала.

'Insert' 'Falling edge detection'

Обозначение: 

Данная команда вставляет в цепь функциональный блок F_TRIG, который служит для выделения заднего фронта импульса (TRUE -> FALSE) сигнала.

'Insert' 'Timer (TON)'

Обозначение: 

Данная команда вставляет в цепь функциональный блок таймер TON, который служит формирования задержки сигнала.

‘Insert’ ‘Jump’

С помощью этой команды вы можете вставить инструкцию перехода параллельно обмоткам, причем эта инструкция размещается в позиции, следующей за последней обмоткой. Если линия, с которой связана инструкция перехода, передает значение On, то осуществляется переход на указанную метку.

Выделенная позиция должна быть обмоткой (позиция курсора №3) или линией, соединяющей обмотку и контакт (позиция курсора №4).

Сразу после выполнения этой команды в поле ввода имени метки появляется строка “???”. Вы можете изменить ее на имя нужной метки.

‘Insert’ ‘Return’

В редакторе LD с помощью этой команды вы можете вставить инструкцию возврата параллельно обмоткам, причем эта инструкция размещается в позиции, следующей за последней обмоткой. Если линия, с которой связана инструкция перехода, передает значение On, то осуществляется переход на начало выполняемого POU.

Выделенная позиция должна быть катушкой (позиция курсора №3) или линией, соединяющей обмотку и контакт (позиция курсора №4).

‘Extras’ ‘Paste after’

Используйте эту команду для вставки содержимого буфера за контактом выделенной позиции. Эта команда доступна, только если содержимое буфера и выделенная позиция – схема, состоящая из контактов.

‘Extras’ ‘Paste below’

Используйте эту команду для вставки содержимого буфера ниже выделенной позиции. Эта команда доступна, только если содержимое буфера и выделенная позиция – схема, состоящая из контактов. Эта схема вставляется параллельно выбранной.

‘Extras’ ‘Paste above’

Используйте эту команду для вставки содержимого буфера выше выделенной позиции. Эта команда доступна, только если содержимое буфера и выделенная позиция – схема, состоящая из контактов. Эта схема вставляется параллельно выбранной.

‘Extras’ ‘Negate’

Обозначение:  Быстрый ввод: <Ctrl>+<N>


Используйте эту команду для инвертирования выбранного контакта, обмотки, инструкции перехода или возврата, входа или выхода POU (позиция курсора 2 или 3). При этом в символе обмотки или контакта появляется слеш (/) или \|. При инвертировании инструкции перехода или возврата, входов или выходов POU появляется кружок в точке соединения, как и в редакторе FBD.

Инверсная обмотка записывает в соответствующую логическую переменную значение, обратное своему. Инвертированный контакт замыкает схему, если соответствующая логическая переменная имеет значение False.

Если инвертирована инструкция возврата или перехода, то она выполняется, когда соединенная с ней линия передает значение Off.

Снять инвертирование с элемента можно, переинвертировав этот элемент.

‘Extras’ ‘Set/Reset’

Обозначение: 

Если выделить обмотку и выполнить эту команду, то можно получить Set-обмотку. Такая обмотка записывает в соответствующую логическую переменную значение True, когда на входе этой обмотки имеется сигнал On, и сохраняет значение этой переменной, когда на входе сигнал Off.

Такая обмотка обозначается буквой “S”.

Выполнив эту команду еще раз, вы получите Reset-обмотку. Такая обмотка записывает в соответствующую логическую переменную значение False, когда на входе этой обмотки имеется сигнал On, и сохраняет значение этой переменной, когда на входе сигнал Off.

Такая обмотка обозначается буквой “R”.

Выполнив эту команду несколько раз, вы можете получить Set-, Reset- и обыкновенную обмотку.

Редактор LD в режиме Online

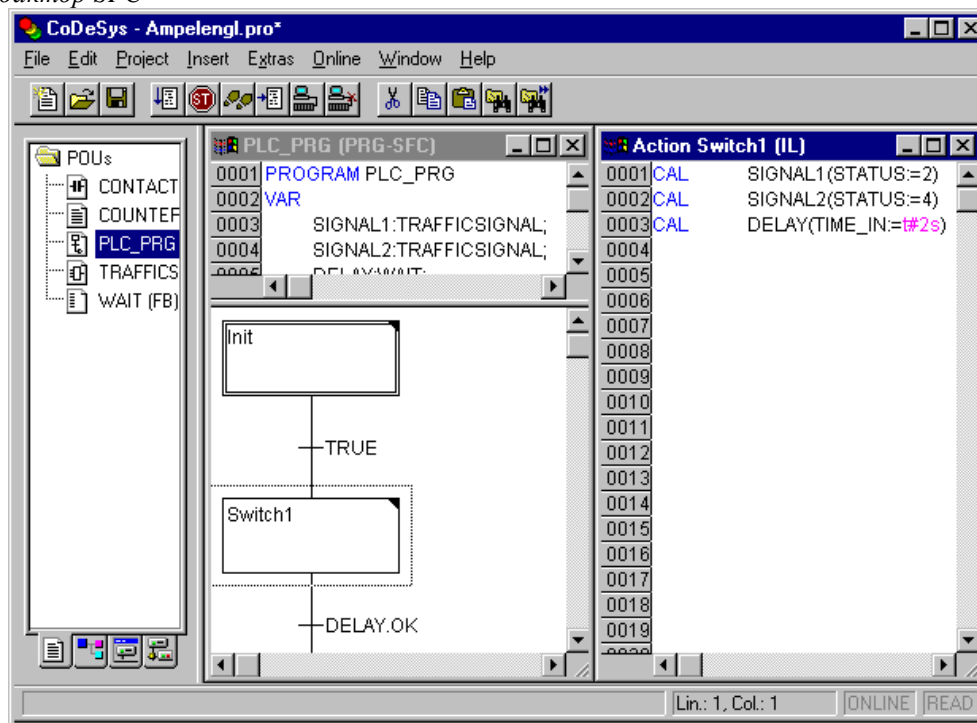
В режиме Online контакты и обмотки, которые находятся в состоянии On, изображаются синим цветом. Кроме того, все линии, передающие состояние On, также окрашиваются синим. Указываются значения всех входов и выходов функциональных блоков.

В режиме Online можно устанавливать точки останова и выполнять программу по шагам.

Если вы переместите указатель мыши на переменную, то в подсказке появятся тип, комментарии и адрес этой переменной.

5.4.4 Редактор SFC

Редактор SFC



Все редакторы POU состоят из раздела объявлений и собственно тела POU. Они отделены друга от друга разделителем.

Редактор SFC – это графический редактор. Наиболее важные команды находятся в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl>+<F10>. Подсказки, включающие полные имена или выражения переходов, шагов, прыжков, меток, классификаторов или связанных действий, показываются в режимах Online и Offline.

Информацию по языку можно найти в главе Sequential Function Chart (SFC).

Редактор SFC удовлетворяет требованиям языка SFC. Для этого обеспечиваются следующие возможности:

Выделение блоков в SFC

Выделенный блок – это совокупность элементов SFC, заключенных в прямоугольник с пунктирными границами.

Вы можете выбрать элемент (шаг, переход, прыжок) левой клавишей мышки или с помощью клавиш перемещения. Для того, чтобы выбрать группу элементов, нажмите клавишу <Shift> и выбирайте элементы. Наименьшая связанная группа состоит из двух элементов.

Обратите внимание: Шаг можно удалить, только выделив его вместе с предшествующим или последующим переходом !

“Insert” ”Step Transition (before)”

Обозначение: : Быстрый ввод: <Ctrl>+<T>

С помощью этой команды вы можете вставить шаг, следующий за переходом, перед выделенным блоком.

“Insert” ”Step Transition (after)”

Обозначение: : Быстрый ввод: <Ctrl>+<E>

С помощью этой команды вы можете вставить шаг, следующий за переходом, после выделенного блока.

Удаление шага и перехода

Шаг можно удалить, только выделив его вместе с предшествующим или последующим переходом. Для этого сделайте выделение вокруг шага вместе с переходом и дайте команду 'Edit' 'Delete' либо нажмите клавишу .

“Insert” ”Alternative Branch (right)”

Обозначение: : Быстрый ввод: <Ctrl>+<A>

Вставляет альтернативную ветвь вправо от выделенного блока. Для этого выделенный блок должен начинаться и заканчиваться переходом. После выполнения этой команды новая ветвь состоит только из одного перехода.

“Insert” ”Alternative Branch (left)”

Обозначение: 

Вставляет альтернативную ветвь влево от выделенного блока. Для этого выделенный блок должен начинаться и заканчиваться переходом. После выполнения этой команды новая ветвь состоит только из одного перехода.

“Insert” ” Parallel Branch(right)”

Обозначение: : Быстрый ввод: <Ctrl>+<L>

Вставляет параллельную ветвь вправо от выделенного блока. Для этого выделенный блок должен начинаться и заканчиваться шагом. После выполнения этой команды новая ветвь состоит только из одного шага. Разрешены произвольные переходы (jump) на параллельные ветви. Для этого нужно сопоставить параллельным ветвям метку с помощью команды Add Label To Parallel Branch.


“Insert” ” Parallel Branch(left)”

Обозначение: : Быстрый ввод <Ctrl>+<U>

Вставляет параллельную ветвь влево от выделенного блока. Для этого выделенный блок должен начинаться и заканчиваться шагом. После выполнения этой команды новая ветвь состоит только из одного шага. Разрешены также произвольные переходы (jump) на параллельные ветви. Для этого нужно сопоставить параллельным ветвям метку с помощью команды Add Label To Parallel Branch.

“Insert” ” Jump”Обозначение: 

Вставляет произвольный безусловный переход (jump) в конец ветви, к которой принадлежит выделенный блок. По умолчанию в поле имени метки стоит строка “Step”. Вы можете заменить ее на имя шага или метку параллельной ветви, на которую должен осуществляться произвольный переход.

“Insert” ”Transition-Jump”Обозначение: 

Данная команда вставляет переход вместе со следующим после него произвольным переходом (jump) в конец выбранной параллельной ветви.

По умолчанию в поле имени метки стоит строка “Step”. Вы можете заменить ее на имя шага или метку параллельной ветви, на которую должен осуществляться переход.

“Insert” ”Add Entry-Actions ”

Добавляет **входное** действие в шаг. Такое действие выполняется только один раз при активации шага и описывается на любом из языков МЭК.

Шаг с входным действием имеет букву “E” в левом нижнем углу.

“Insert” ”Add Exit-Actions ”

Добавляет **выходное** действие в шаг. Такое действие выполняется только раз при деактивации шага и описывается на любом из языков МЭК.

Шаг с выходным действием имеет букву “X” в правом нижнем углу.

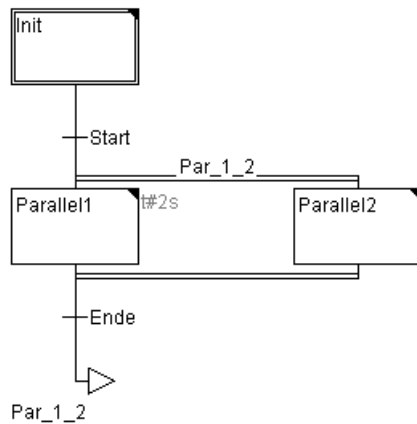
“Extras” ”Paste Parallel Branch (right)”

Данная команда вставляет содержимое буфера как правую параллельную ветвь для выбранного блока. Для этого выделенный блок должен начинаться и заканчиваться шагом. Содержимое буфера должно быть блоком, написанном на SFC, который также должен начинаться и заканчиваться шагом.

“Extras” ”Add label to parallel Branch”

Для того чтобы связать метку с параллельной ветвью, нужно выбрать переход перед разветвлением и выполнить команду **”Add label to parallel Branch”**. По умолчанию параллельная ветвь получит имя, состоящее из слова **“Parallel”** и порядкового номера. Это имя можно заменить на требуемое.

В следующем примере имя “Parallel” заменено на “Par_1_2” и после перехода “Ende” осуществляется произвольный безусловный переход на эту параллельную ветвь.



Удаление метки

Удалить метку параллельной ветви можно, удалив ее имя.

“Extras” ”Paste after”

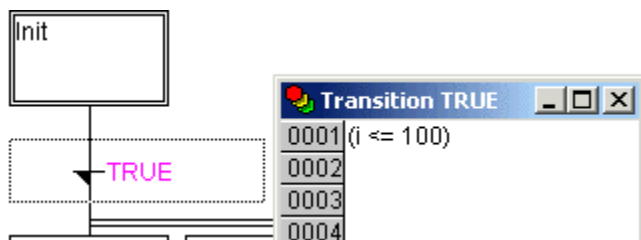
Вставляет блок SFC из буфера после первого шага или перехода выделенного блока. Эта команда выполняется, если результирующая структура соответствует правилам языка SFC.

“Extras” ”Zoom Action/Transition”

Быстрый ввод: <Alt>+<Enter>

Действие первого шага или условие перехода выделенного блока выводятся в окне редактора того языка, на котором написан данный переход или шаг. Если же действие или переход не описаны, то появится диалоговое окно, в котором можно выбрать один из языков.

Обратите внимание, что условие, заданное в окне редактора, имеет превосходство над условием, заданным непосредственно рядом с переходом. Пример: здесь условие $i > 100$ не выполняется, поэтому условие перехода FALSE, несмотря на то, что рядом с переходом задано TRUE!



“Extras” ”Clear Action/Transition”

Действие первого шага или условие первого перехода выделенного блока удаляются.

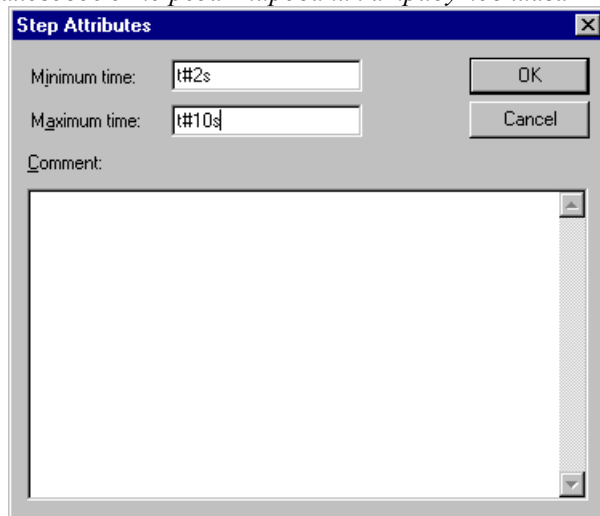
Если шаг включает только одно действие (либо входное, либо выходное, либо основное действие), то удаляется именно это действие. В другом случае, появляется диалоговое окно, в котором можно выбрать действие, которое нужно удалить.

В случае, когда выбран МЭК - шаг, можно удалять действия, связанные с этим шагом. Если шаг связан с несколькими действиями, то появляется диалоговое окно, в котором показан список всех действий. Если шаг связан только с одним действием, то это действие удаляется автоматически.

“Extras” ”Step Attributes”

Открывает диалоговое окно редактирования атрибутов выделенного шага.

Диалоговое окно редактирования атрибутов шага

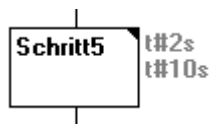


Вы можете воспользоваться тремя полями ввода в этом окне. В поле **Minimum Time** введите минимально возможное время в активности шага. В поле **Maximum Time** должно стоять максимально возможное время активности шага. Заметим, что вводить можно либо константы в формате Time (например, T#3s), либо переменные типа Time.

В поле ввода **Comment** вы можете ввести комментарии для шага. В диалоговом окне “Sequential function chart options”, которое открывается при вызове команды “Extras” “Options”, вы можете установить, показывать ли комментарии и атрибуты времени для шагов в SFC. С правой стороны шага появляются комментарии или атрибуты времени.

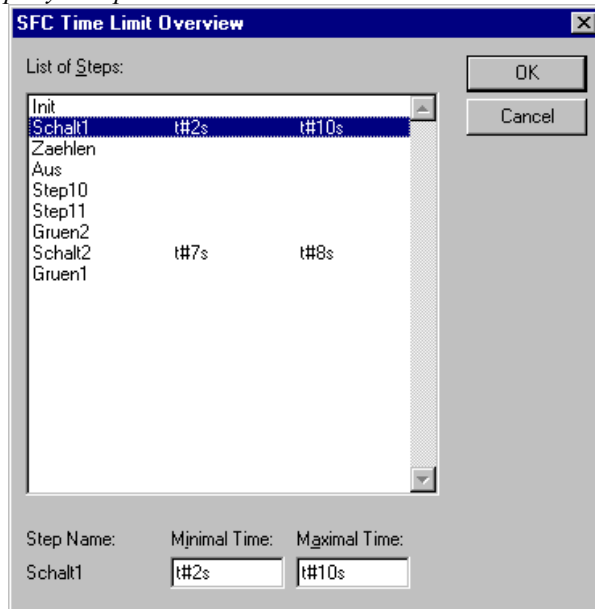
Если время выполнения шага больше, чем заданное максимальное время, то устанавливается соответствующий флаг SFC, доступный программно.

На примере показан шаг время выполнения которого должно быть не меньше 2 секунд и не больше 10 секунд. В режиме Online, кроме этих двух времен, выводится время активности шага.



“Extras” ”Time Overview”

Открывает диалоговое окно редактирования атрибутов времени шагов SFC.

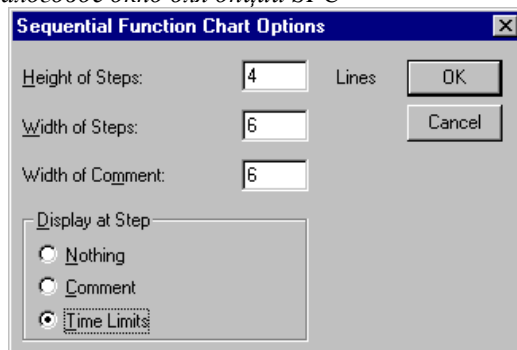
Атрибуты времени SFC POU

В этом диалоговом окне показаны все шаги POU. Если вы ввели атрибуты времени для шага, то они будут показаны справа от имени шага (сначала минимальное, а затем максимальное время). Для редактирования атрибутов времени щёлкните мышкой на имени желаемого шага в списке, после чего в полях **Minimal Time** и **Maximal Time** введите нужные значения. Если вы закроете окно, нажав кнопку ОК, то все изменения будут сохранены.

В этом примере, шаги 2 и 6 имеют атрибуты времени. Для шага 2 минимальное значение времени 2 секунды, а максимальное – 10 секунд. Для шага 6 атрибуты времени 7 и 8 секунд соответственно.

“Extras” ”Options”

Открывает диалоговое окно редактирования опций отображения SFC диаграммы.

Диалоговое окно для опций SFC

Вы можете установить 6 опций SFC. Опция **Height of Step** определяет высоту шага в строчках (по умолчанию 4). Опция **Width of Step** определяет ширину шага в строчках (по умолчанию 6). На панели **Display at Step** вы можете установить, что показывать справа от шага: ничего (**Nothing**), комментарии (**Comment**) или атрибуты времени (**Time Limits**)

“Extras” ”Associate Action”

С помощью этой команды можно связать действие или логическую переменную с шагом МЭК. Справа от шага появляется еще один блок, в котором описываются действия, связанные с этим шагом. По

умолчанию в этом блоке задается классификатор “N” и действие “Action_1”. Классификатор и действие можно изменить. Для этого удобно использовать Input Assistant.

Новые действия для шагов МЭК можно создать в Object Organizer, выбрав необходимое POU и выполнив команду “Project” “Add Action”.

“Extras” ”Use IEC-Steps”

Обозначение: 

Если эта команда активна (стоит галочка в пункте меню), то при выполнении команд вставки шагов-переходов или параллельных ветвей вместо обычных шагов будут появляться МЭК-шаги.

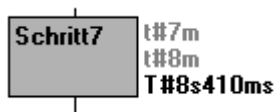
В случае, если эта опция активна при создании SFC POU, то шаг Init будет шагом МЭК.

Эта установка сохраняется в файле “CoDeSys.ini” и продолжает действовать при перезапуске CoDeSys.

SFC в режиме Online

В режиме Online активные шаги изображаются синим цветом. Если вы установили опцию **Time Limits** в пункте меню “Extras” “Options”, то рядом с шагом будут выводиться атрибуты времени этого шага и время его активности.

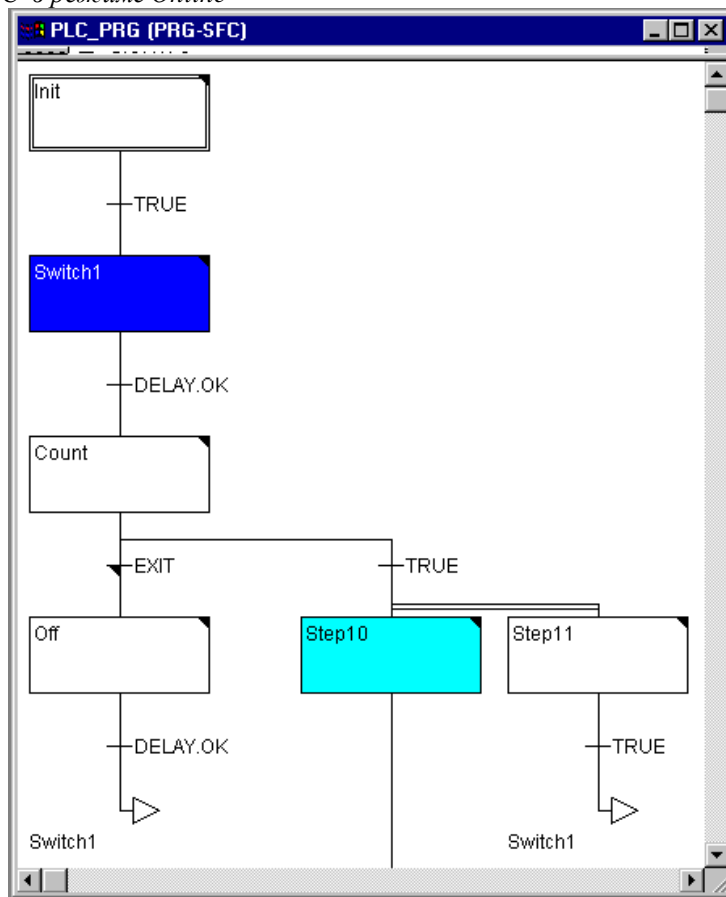
На примере показано, что шаг активен 8 секунд и 410 миллисекунд. Заметим, что шаг должен быть активен не более 7 минут.



С помощью команды “Online” ”Toggle Breakpoint” можно установить точку останова либо на самом шаге, либо в действии, связанном с ним. Выполнение программы будет приостановлено перед выполнением этого шага или инструкции в действии. Шаг или инструкция в действии, где установлена точка останова, изображается голубым цветом.

Если выполнение программы приостановлено, то точка останова становится красной.

При использовании шагов МЭК в режиме Online все активные действия изображаются синим.

SFC в режиме Online

Шаг Switch на рисунке активен, а точка останова установлена на шаге Step10.

С помощью команды **“Online”** **”Step over”** можно выполнить программу по шагам. Если активная точка - это:

- шаг в POU, где нет разветвлений, или шаг в самой правой параллельной ветви в POU, то управление возвращается в блок, вызвавший это POU. Если POU – это PLC_PRG, то программа выполняется по циклам;
- шаг не в самой правой параллельной ветви, то выполняется активный шаг следующей параллельной ветви;
- последняя возможная точка останова в действии шага, то управление возвращается в блок, вызвавший это POU. То же самое относится и к МЭК-шкагам;
- последняя возможная точка останова во входном или выходном действии шага, то управление передается следующему активному шагу.

С помощью команды **“Online”** **”Step in”** можно выполнять программу по шагам, заходя в вызываемые блоки. Однако если вы хотите заходить во входные, выходные действия или в действия, связанные с МЭК-шагами, вы должны поставить в них точки останова. Внутри действий доступны все функции отладки соответствующих редакторов.

Если вы переместите указатель мыши на переменную, то в подсказке появятся тип, комментарии и адрес этой переменной.

Внимание: Если вы переименуете шаг и выполните горячее обновление (Online Change) во время активности этого шага, то программа приобретет неопределенное состояние и будет остановлена!

Порядок выполнения действий в управляющем цикле

1. Во-первых, сбрасываются все флаги управления действиями МЭК-шагов(это не относится к флагам, обрабатываемым внутри действий).
2. Все шаги проверяются сверху вниз и слева направо для того, чтобы определить, нужно ли выполнять выходное действие, если да, то это действие выполняется.
3. Все шаги проверяются сверху вниз и слева направо для того, чтобы определить, нужно ли выполнять входное действие, и если да, то это действие выполняется.
4. Для всех шагов выполняется следующая последовательность действий:
 - Если необходимо, то в соответствующую переменную копируется время активности шага.
 - Если произошел тайм-аут, то устанавливаются соответствующие SFC-флаги.
 - Если шаг не является МЭК-шагом, то выполняется связанное с ним действие.
5. Действия, связанные с МЭК-шагом, выполняются в два этапа в алфавитном порядке. На первом этапе выполняются те действия, которые должны быть деактивированы в этом цикле. На втором этапе выполняются активные в этом цикле действия.
6. Вычисляются условия переходов: если шаг в данном цикле был активен и переход имеет значение True (и при необходимости, если время выполнения шага больше минимального), то следующий шаг становится активным.

Относительно реализации шагов заметим следующее:

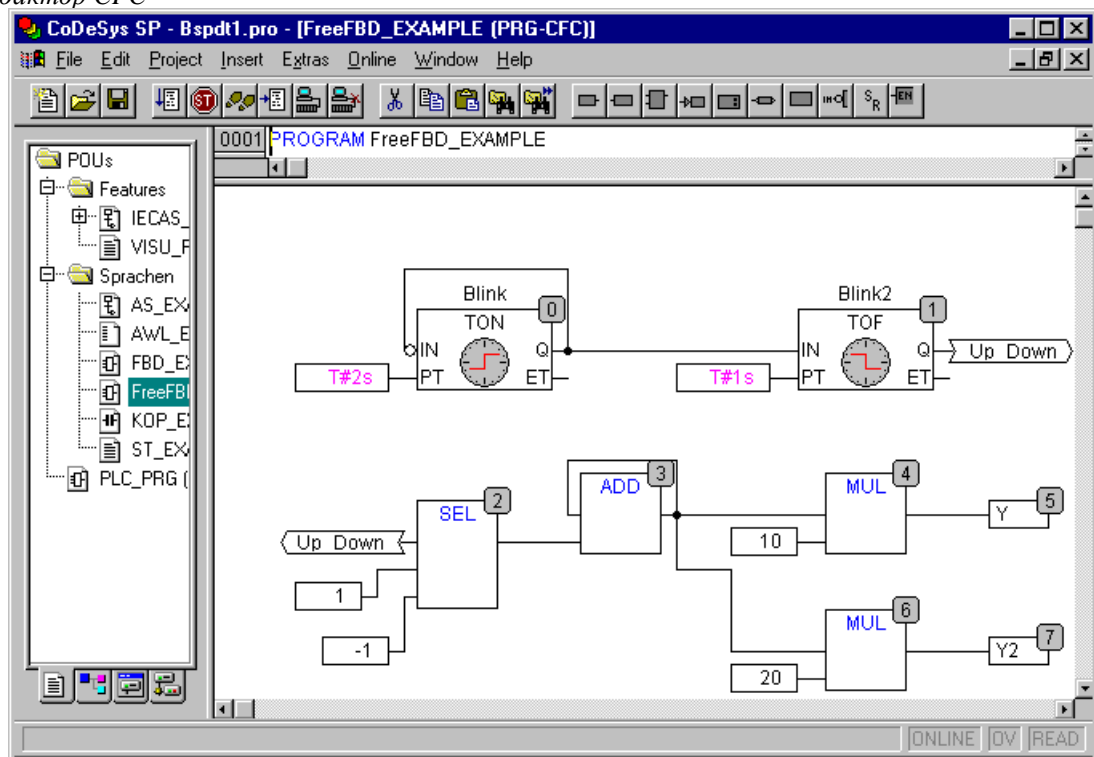
Одно действие может выполняться несколько раз в одном управляемом цикле, если оно связано с разными шагами (например, программа SFC имеет два действия А и В, которые оба выполнены на SFC, и оба этих действия вызывают третье действие С. Если в одном цикле действия А и В активны, то действие С выполнится дважды).

Описанная выше ситуация может приводить к неопределенности и сопровождается сообщением об ошибке. Ошибки могут появиться, если вы используете проект, созданный в ранней версии CoDeSys.

Замечание: При мониторинге выражения перехода (например, Bool1 AND Bool2) выводится только конечное значение выражения.

5.4.5 Редактор CFC

Редактор CFC



В этом редакторе нет сетки, и поэтому элементы могут располагаться где угодно. К элементам языка CFC относятся блоки, входы, выходы, возвраты, произвольные переходы, метки и комментарии. Входы и выходы этих элементов можно соединять, перетаскивая линии соединения мышкой. Эти линии будут перерисовываться автоматически при перемещении элементов. В случае, если линия соединения не может быть перерисована, то она становится красной, и как только вы переставите элемент так, чтобы можно было соединить вход и выход линией без пересечений с другими элементами, линия становится нормальной.

Основное преимущество CFC редактора перед FBD заключается в том, что в схемы можно непосредственно добавлять линии обратной связи.

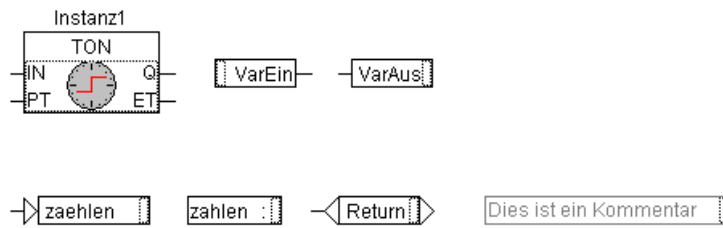
Наиболее важные команды можно найти в контекстном меню.

Позиции курсора

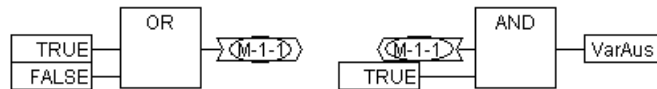
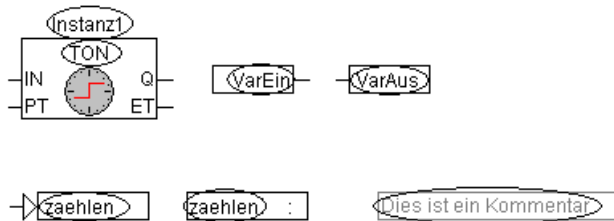
Курсор может располагаться в любой текстовой в строке CFC схемы. Выделенный текст можно редактировать.

Во всех остальных случаях позиция курсора выделяется прямоугольником с пунктирной границей. Далее вместе с примерами показаны все возможные позиции курсора.

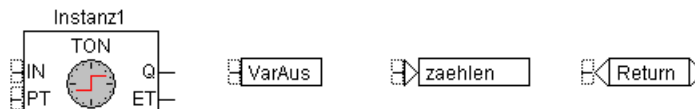
1. Блоки, входы, выходы, возвраты, произвольные переходы и комментарии.



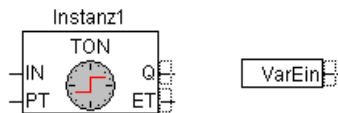
2. Все текстовые поля.



3. Входы элементов: блок, вход, выход, возврат и переход на метку.



4. Выходы элементов: блок и вход.



“Insert” “Box”

Обозначение:  Быстрый ввод: <Ctrl> + .

Команда может быть использована для вставки операторов, функций, функциональных блоков и программ. Сразу после ее выполнения появляется блок с именем “AND”. Выбрав текстовое поле внутри этого блока, вы можете изменить его на имя любого другого оператора, функции, функционального блока или программы. Для этой цели удобно использовать Input Assistant. Если новый блок имеет большее минимальное число входов, то будут добавлены новые входы. Если количество входов нового блока меньше, чем количество входов выбранного блока, то последние входы удаляются.

“Insert” “Input”

Обозначение:  Быстрый ввод: <Ctrl> + <E>.

Вставка входа. В текстовом поле входа появятся три знака вопроса, которые нужно заменить на имя переменной или константы. Для этой цели удобно использовать Input Assistant.

“Insert” “Output”

Обозначение:  Быстрый ввод: <Ctrl> + <A>.


Вставка выхода. В текстовом поле выхода появятся три знака вопроса, которые нужно заменить на имя переменной. Для этой цели удобно использовать Input Assistant. Значение этого выхода присваивается введенной Вами переменной.

“Insert” “Jump”

Обозначение:  Быстрый ввод: <Ctrl> + <J>.

Вставка произвольного безусловного перехода на метку. В текстовом поле появятся три знака вопроса, которые нужно заменить на имя метки. Метку можно вставить с помощью команды **“Insert” “Label”**.

“Insert” “Label”

Обозначение:  Быстрый ввод: <Ctrl> + <L>.

Вставка метки. В текстовом поле метки появятся три знака вопроса, которые нужно заменить на имя метки. В режиме Online в конец POU автоматически добавляется метка Return.

Для того чтобы вставить переход на метку, используйте команду **“Insert” “Jump”**.

“Insert” “Return”

Обозначение:  Быстрый ввод: <Ctrl> + <R>.

Вставка возврата RETURN. Заметим, что в режиме Online в конец схемы добавляется метка с именем RETURN и при выполнении программы по шагам, перед тем как выйти из POU, выполнение останавливается на этой метке.

“Insert” “Comment”

Обозначение:  Быстрый ввод: <Ctrl> + <K>.

Вставка комментариев. При вводе комментария к новой строке можно перейти, нажав <Ctrl>+<Enter>.

“Insert” “Input of box”

Быстрый ввод: <Ctrl> + <U>.

Добавить вход блока. У некоторых операторов число входов можно изменять (например, ADD может иметь два или больше входов).

Эта команда выполнима, если выбран оператор (позиция курсора №1).

“Insert” “In-Pin”, “Insert” “Out-Pin”

Обозначения:  

Обе команды доступны при редактировании макроса. Они используются для вставки входов и выходов макроса, которые отличаются от обыкновенных входов и выходов POU способом отображения и отсутствием позиционных индексов.

“Extras” “Negate”

Обозначение:  Быстрый ввод: <Ctrl>+<N>.

Инвертирование входов, выходов, переходов на метку прыжков или возвратов. При инвертировании на месте соединения этих элементов со схемой появляется кружок.

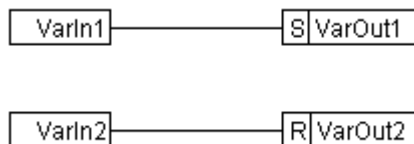
Инверсию можно снять, выполнив команду еще раз.

“Extras” “Set/Reset”

Обозначение:  Быстрый ввод: <Ctrl>+<T>.

Команда доступна, когда выбран вход или выход элемента.

Признак Set обозначается символом S, а Reset - символом R.



VarOut1 получает значение TRUE, когда переменная VarIn1 истинна, и сохраняет свое значение, даже когда переменная VarIn1 уже ложна.

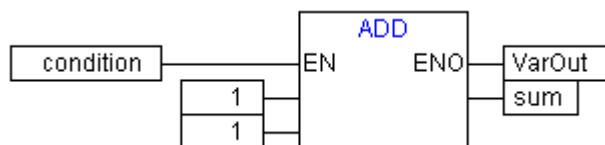
VarOut2 получает значение FALSE, когда переменная VarIn2 истинна, и сохраняет свое значение, даже когда пока переменная VarIn1 уже ложна.

При многократном выполнении этой команды элемент последовательно меняется на Set, Reset и обычный.

“Extras” “EN/ENO”

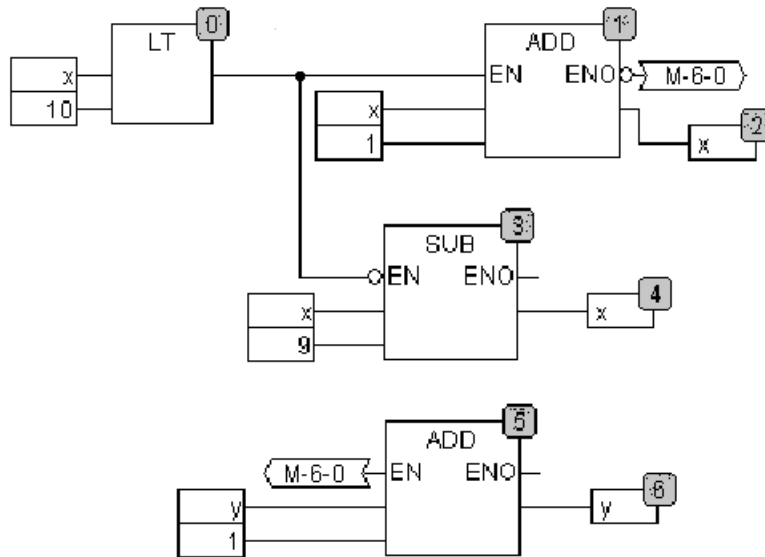
Обозначение:  Быстрый ввод: <Ctrl>+<0>.

Добавить в выбранный блок (позиция курсора №3) вход EN и выход ENO.



VarOut примет значение TRUE после выполнения ADD. Если далее condition изменится в FALSE, то ADD (и все, что за ним) более не будет выполняться. Обратите внимание, что при этом значения на его выходах не изменяются! То есть выход VarOut элемента AND остается в TRUE.

В следующем примере показано, как можно использовать выход ENO.



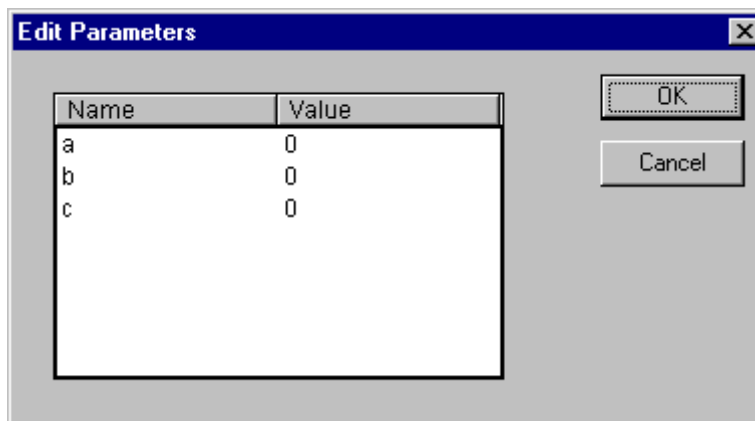
X инициализируется единицей, а Y нулем. Номер в правом углу блока показывает, в каком порядке будут выполняться команды.

X будет увеличиваться на единицу, пока не достигнет значения 10. После этого на выходе блока LT (0) появится значение FALSE и выполнятся операторы SUB(3) и ADD(5). Теперь X снова установится в единицу, а Y увеличивается на единицу, и LT(0) будет выполняться до тех пор, пока X меньше 10. Таким образом Y показывает, сколько раз переменная X пробежала значения от 1 до 10.

“Extras” “Properties...”

Выбрав функцию или функциональный блок, с помощью этой команды можно получить список постоянных входных параметров (VAR_INPUT CONSTANTS) и их значений. То же самое происходит, если выбрать элемент (позиция курсора №1) и дважды щелкнуть по нему мышкой. Непосредственно на CFC диаграмме постоянные параметры не отображаются.

Диалог “Edit Parameters”



Значения постоянных входных параметров (VAR_INPUT CONSTANT) можно изменять. Для этого выберите соответствующий параметр в столбце Value. Для подтверждения изменения значения нажмите <Enter>, а для отмены - <Esc>.

Кнопка ОК сохраняет все преобразования.

Данная функциональность и соответствующее объявление переменных с ключевым словом "VAR_INPUT CONSTANT" имеет существенное влияние только в редакторе CFC. В редакторе FBD все INPUT переменные всегда присутствуют в отображении элемента независимо от объявления VAR_INPUT или VAR_INPUT CONSTANT. Для текстовых редакторов также нет никакой разницы.

Выбор элементов

Для того чтобы выбрать элемент, нужно щелкнуть по нему мышкой (позиция курсора №1).

Чтобы выбрать больше одного элемента, вы должны нажать клавишу <Shift> и выбирать нужные элементы или, щелкнув мышкой на свободном месте, растягивать получившийся прямоугольник. Команда "Extras" "Select all" сразу выбирает все элементы.

Перемещение элементов

Один или несколько элементов можно перемещать с помощью клавиш перемещения, нажав клавишу <Shift>. Это можно сделать иначе: выберите элемент и перемещайте его, не отпуская левую клавишу мыши. Элементы перемещаются до тех пор, пока они не перекрывают другие элементы или не заходят за пределы экрана. В таких случаях элемент будет перемещен в начальную позицию, и вы услышите сигнал тревоги.

Копирование элементов

Выбранные элементы можно скопировать в буфер с помощью команды "Edit" "Copy" и вставить с помощью команды "Edit" "Paste".

Соединение элементов

Вход одного элемента можно соединять с выходом другого. Выход одного элемента может соединяться сразу с несколькими входами других элементов.

Есть несколько возможностей соединения входа элемента E2 с выходом элемента E1.



Поместите указатель мыши на выход элемента E1 (позиция курсора №4), нажмите левую кнопку мыши и, удерживая ее, переместите курсор мыши на вход элемента E2 (позиция курсора №3) и отпустите кнопку мыши. Линия соединения будет создана при перемещении курсора мыши.

Рассмотрим другой способ: поместите курсор мыши на вход элемента E2, нажмите левую кнопку мыши и, удерживая ее, переместите курсор мыши на выход элемента E1. Мы получили такое же соединение.

Переместите один из элементов так, чтобы его вход (выход) соприкоснулся с выходом (входом) другого. Теперь можно как угодно перемещать элементы, и при этом они останутся соединенными.

Если элемент E2 имеет свободный вход, то, переместив указатель мыши с выхода E1 на элемент E2, вы соедините вход и выход. Это произойдет, как только вы отпустите кнопку мыши. В случае, если E2 не имеет свободных входов, то будет добавлен новый, с которым и произойдет соединение.

Этим же методом могут быть соединены вход и выход одного элемента (обратная связь).

Аналогично соединяются входы и выходы макросов.

Если при перемещении линий соединения вы вышли за пределы рабочей области, то окно автоматически прокрутится. Для сложных типов проверка соответствия типов производится при компиляции, а

для простых типов - при соединении. Если вход и выход нельзя соединить по причине несоответствия типов, то изменится форма курсора.

Удаление линий соединения

Есть несколько способов удаления линии, соединяющей выход элемента E1 и вход элемента E2:

Выберите выход элемента E1 или вход элемента E2 (позиция курсора №4) и нажмите <Delete> или выполните команду **“Edit”** **”Delete”**. Если выход элемента E1 связан с несколькими входами, то будут удалены все соединения.

Поместите указатель мыши на вход элемента E2 и, удерживая левую клавишу мыши, переместите его на свободную область экрана. Соединение будет удалено, как только вы отпустите кнопку мыши.

Изменение соединений

Соединение выхода элемента E1 и входа элемента E2 можно легко изменить на соединение выхода элемента E1 и входа элемента E3. Кликните мышкой на входе элемента E2 (позиция курсора №3), удерживая левую кнопку мыши, переместите указатель на вход элемента E3 и опустите кнопку мыши.

“Extras” ”Connectionmark”

Соединять элементы можно не только линией соединения, но и соединяющим маркером. В этом случае вход и выход соединяются с маркером, имеющим уникальное имя.



В нашем примере вход и выход соединяются с помощью соединяющей линии. Теперь выберите начало линии соединения (позиция курсора №3) и выполните команду **“Extras”** **”Connectionmark”**. Схема превратится в эквивалентную ей, но в которой вход и выход соединены с помощью маркеров.

По умолчанию маркер получит имя, начинающиеся с буквы **“M”**. Это имя можно изменить, причем при изменении имени маркера входа изменяется имя маркера выхода и наоборот.

1. Редактирование маркера на выходе: 

Если изменить имя маркера выхода, то изменятся имена всех соответствующих ему маркеров входов. Нельзя выбирать имя маркера, которое уже принадлежит другому маркеру. Оно должно быть уникальным.

2. Редактирование маркера на входе: 

Производится так же, как и в пункте 1.

Для того чтобы перевести маркер в линию соединения, нужно выбрать маркер на выходе (позиция курсора №4) и снова выполнить команду **“Extras”** **”Connection marker”**.

Вставка входов/выходов “на лету”

Выберите вход или выход элемента и введите какую-либо строку с клавиатуры. На схеме появится элемент вход или выход, связанный с выбранным входом или выходом элемента, и с именем переменной или константой, которую вы ввели.

Порядок выполнения схемы

Каждый элемент схемы обладает номером, который указывает порядок его выполнения.

При создании или вставке элемента он автоматически получает номер в соответствии со следующим правилом: слева направо и сверху вниз. Номер элемента не изменяется при его перемещении.

Последовательность действий определяет результат и должна быть изменена при необходимости.

Номер отображается в правом верхнем углу элемента, если включен режим отображения.

“Extras” “Order” ” Show order”

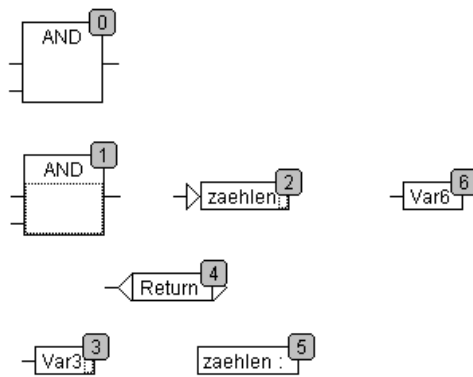
Команда определяет, показывать ли порядок выполнения схемы или нет. По умолчанию порядок выполнения показывается (в соответствующем пункте меню стоит галочка). Порядковый номер отображается в правом верхнем углу элемента.

“Extras” “Order” ”Order topologically”

Автоматическая нумерация элементов схемы в порядке слева направо и сверху вниз. Такой порядок называется топологическим. При этом не имеют значения соединения элементов схемы, а важно лишь расположение элементов.

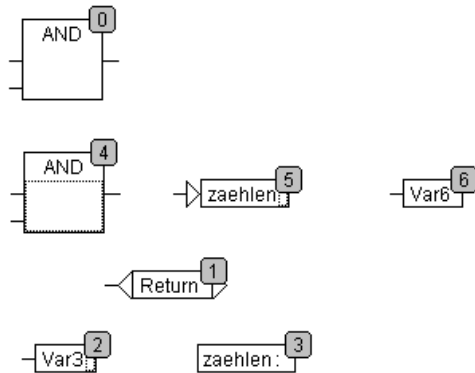
Данная команда применима также к отдельным выбранным элементам, которым присваиваются номера в топологическом порядке при выполнении этой команды. Каждому выбранному элементу присваивается номер так, чтобы он выполнялся перед следующим за ним в топологическом порядке элементом. Поясним это на примере:

Схема до выполнения команды расстановки элементов в топологическом порядке



Выбраны элементы с номерами 1, 2 и 3. При выполнении команды “Order topologically” изменяются номера всех отмеченных элементов. Var3, находящийся перед меткой, получает номер 2, оператор AND получает номер 4, а переход на метку получает номер 5.

Схема после выполнения команды расстановки элементов в топологическом порядке:



Новый вставленный элемент, по умолчанию, получает номер в соответствии с его топологическим расположением.

“Extras” “Order” “Order one up”

При выполнении этой команды выбранные элементы перемещаются на одну позицию вверх по списку порядка выполнения элементов.

“Extras” “Order” “Order one down”

При выполнении этой команды выбранные элементы перемещаются на одну позицию вниз по списку порядка выполнения элементов.

“Extras” “Order” “Order first”

Выбранные элементы перемещаются в начало списка элементов. Порядок нумерации всех остальных элементов остаётся прежними.

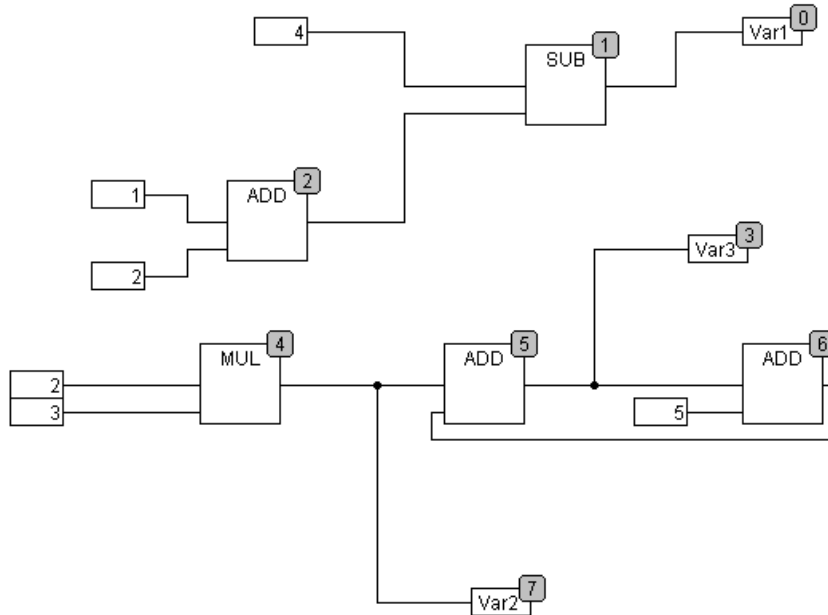
“Extras” “Order” “Order last”

Выбранные элементы перемещаются в конец списка элементов. Порядок нумерации всех остальных элементов остаётся прежним.

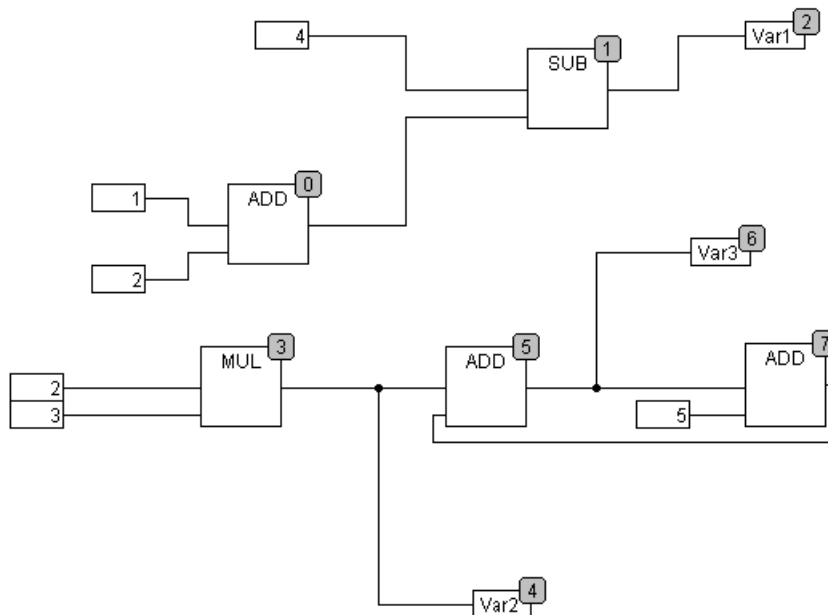
“Extras” “Order” “Everything ordered by data flow”

Эта команда относится ко всем элементам. Порядок выполнения схемы определяется потоком данных, а не позициями элементов. На следующей схеме элементы расставлены в топографическом порядке.

Схема до выполнения команды “Everything ordered by data flow”



После выполнения команды схема выглядит так:




При выполнении этой команды создается новый список элементов. Основываясь на уже известных значениях входов, CoDeSys вычисляет какой из еще не пронумерованных элементов можно выполнить следующим. Например, в приведенной выше схеме оператор AND может быть выполнен сразу же, как только будет известно значение его входов (1 и 2), а оператор SUB выполним после того, как будет вычислен результат оператора ADD.

Обратные связи вставляются последними.

Преимущество такого порядка выполнения элементов заключается в том, что элемент Output выполняется сразу после того, как вычислен связанный с ним блок, тогда как при топологическом порядке это происходит далеко не всегда. Одна и та же схема, выполняемая в данном и в топологическом порядках, может дать различные результаты.

“Extras” “Create macro”

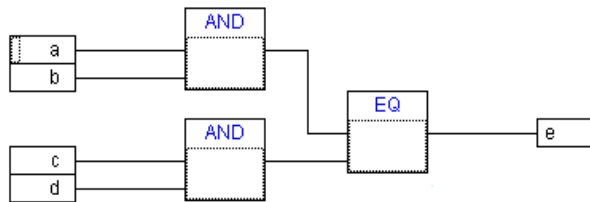
Обозначение: 

С помощью этой команды несколько выбранных ROU можно собрать в один блок, который называется макросом. Макрос можно дублировать с помощью копирования/вставки, при этом имя каждого макроса не должно повторяться. Все соединения, которые вырезаются при создании макроса, превращаются во входы и выходы макроса. По умолчанию входы макросов получают имя In<n>, а выходы – Out<n>. Если соединение осуществлялось посредством маркера, то вход или выход макроса будет соединен с маркером.

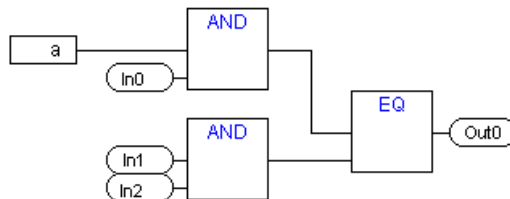
По умолчанию макрос получает имя “Macro”, которое можно изменить. При редактировании макроса имя макроса находится в заголовке окна редактирования.

Пример.

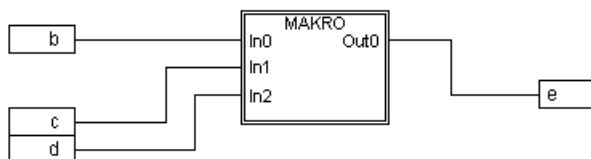
Выбор элементов:



Макрос:





Итоговая схема:



“Extras” “Jump into Macro”

Обозначение: 

С помощью этой команды вы можете редактировать макрос в отдельном окне редактора соответствующего РОУ. Имя макроса находится в заголовке окна редактирования. Двойной щелчок мыши по макросу эквивалентен выполнению этой команды. При редактировании макроса его входы и выходы обрабатываются как обыкновенные входы и выходы РОУ. Они также могут быть перемещены, удалены или добавлены. Отличие только в том, что по-другому выглядят и не имеют номеров. Для добавления

вы можете использовать кнопки  входы,  выходы, доступные на панели команд. Прямоугольники выводов имеют сглаженные углы. Текст в прямоугольниках обозначает наименование выводов макро.

Порядок входов и выходов макроса соответствует порядку выполнения элементов макроса.

В схеме макрос выполняется как обычный блок. Внутри макроса можно выполнять команды изменения порядка действий.

“Extras” “Expand Macro”

Эта команда противоположна по действию команде “Create Macro”. Соединения с входами и выходами макроса снова изображаются как соединения с входами и выходами элементов. Если макрос не удастся развернуть из-за недостатка места, то он перемещается в правый нижний угол схемы до тех пор, пока место не освободится.

Замечание: Если проект сохраняется как проект версии 2.1 или младше, то все макросы будут заманены соответствующими им схемами. То же самое происходит при конвертировании схемы CFC, содержащей макросы.

“Extras” “Back one macro level”, “Back all macro level”

Обозначение:  

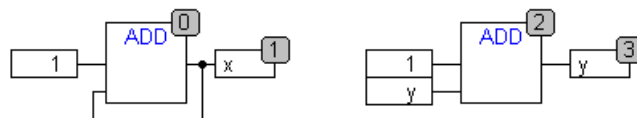
Команды доступны, когда макрос открыт для редактирования. Если макросы вложены один в другой, то с помощью этих команд можно переключаться между разными уровнями вложенности.

Обратные связи CFC

Линии обратной связи непосредственно отражаются в редакторе CFC. Следует заметить, что значение выхода любого блока хранится во внутренней временной переменной, тип данных которой соответствует наибольшему по размеру типу данных входов.

Константы хранятся в переменных с наименьшим из возможных размеров типов данных. Так, например, константа “1” на входе дает тип SINT. Если теперь соединить второй вход с выходом, то его тип будет аналогичным, поскольку ничем более не определен.

Следующие схемы показывают, как можно использовать обратную связь. Переменные x и y типа INT.



Между двумя схемами есть различия:

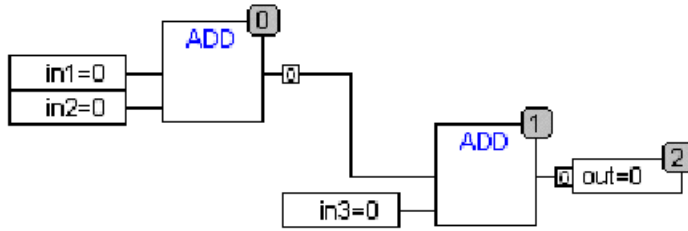
Переменная y может инициализироваться значением, отличным от нуля. Входы левой схемы имеют тип SINT, а правой – INT. Поэтому переменные x и y будут иметь разные значения после 129 вызовов. Хотя переменная x имеет тип INT, она получит значение -127, так как внутренняя переменная переполнится. Переменная y достигнет значения 129 и продолжит увеличиваться дальше.

CFC в режиме Online

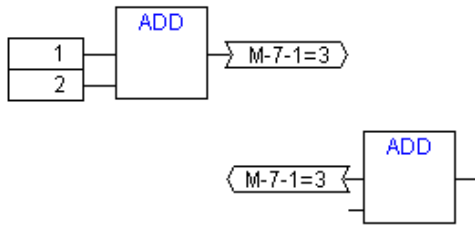
Мониторинг:

Значения входов и выходов изображаются внутри прямоугольных элементов. Мониторинг констант не производится. Для не логических переменных границы элементов расширяются так, чтобы значения этих переменных были видны. Для логических переменных сами элементы и соединенные с ними линии изображаются синим, если значения переменных TRUE, и остаются черными, если значение переменных FALSE.

Внутренние логические соединения изображаются синим, если они передают значение TRUE, и черным в противном случае. Значения внутренних нелогических соединений можно увидеть в квадратиках на выходах элементов.



Мониторинг входов и выходов макросов производится в прямоугольниках выходов.



Если соединения заменены маркерами, то значения, передаваемые по этим соединениям, изображаются внутри маркеров.

Контроль потока:

Если включен контроль потока, то соединения, по которым данные уже были переданы, выделяются цветом.

Точки останова:

Точки останова можно устанавливать только на тех элементах, которые имеют номер. Программа будет остановлена перед выбранным элементом. Номер элемента используется как номер позиции точки останова в диалоге Breakpoints.

Установка точки останова на выбранном элементе осуществляется нажатием клавиши <F9> или с помощью пункта меню "Online" "Toggle breakpoint". Кроме того, точка останова может быть поставлена двойным щелчком на элементе. Точка останова выделяется цветом, определенным в опциях настройки.

Метка Return:

В режиме Online в конце схемы автоматически будет поставлена метка с именем Return. Эта метка обозначает конец POU, и при выполнении программы по шагам прежде, чем выйти из POU, программа будет остановлена на этой метке. В макросах метка Return не ставится.

Выполнение программы по шагам:

При использовании команды “Step over” программа останавливается на следующем элементе с большим номером. Если текущий элемент это макрос или POU, то он будет выполнен по шагам при использовании команды “Step in”.

6 Ресурсы

6.1 Обзор ресурсов

Во вкладке **Resources** Организатора объектов находятся объекты, предназначенные для настройки и управления проектом и распределением переменных:

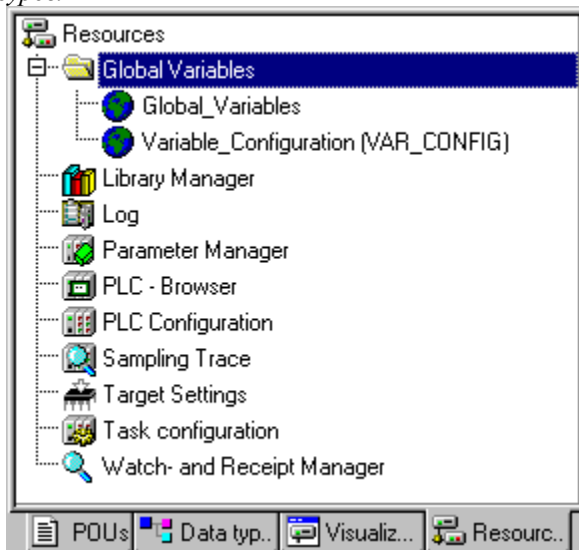
- **Global Variables** - описания глобальных переменных, которые используются в проекте. Здесь же находятся глобальные переменные, описанные в библиотеках.
- **Alarm configuration** – организация системы формирования тревог в CoDeSys проекте.
- **Library Manager** – управление библиотеками, включенными в проект.
- **Log** – запись боржурнала в ходе online сессий.
- **PLC Configuration** - создание описания конфигурации аппаратных средств.
- **Watch and Receipt Manager** – установка и просмотр наборов значений переменных.
- **Task Configuration** – управление программами с помощью задач.
- **Target settings** – выбор аппаратной платформы и настройка ее специфических параметров
- **Workspace** – образ опций проекта.

В зависимости от аппаратной платформы могут быть доступны следующие ресурсы:

- **Parameter Manager** – управление переменными, доступными другим абонентам сети.
- **PLC Browser** – мониторинг и использование специфических особенностей ПЛК.
- **Sampling Trace** – графическое осциллографирование значений переменных.
- **Tools** – подключение внешних инструментов к CoDeSys.
- **SoftMotion** – (при наличии лицензии): CNC и CAM редакторы (см. Отдельный документ по SoftMotion).

Здесь же определяется создание и использование файла комментариев (Docuframe file) на разных языках (английский, русский и т.д.), позволяющего параллельно создавать несколько вариантов печатной документации.

Ресурсы



6.2 Глобальные и конфигурационные переменные, файл комментариев

Объекты глобальных переменных

В папке **Global Variables** вкладки **Resources** (см. рис. 6.1) вы найдете следующие списки (их имена по умолчанию даны в скобках):

- Список глобальных переменных (Global Variables).
- Список конфигурационных переменных (Variable Configuration).

Переменные, определенные в этих списках, можно использовать во всем проекте.

Если папка **Global Variables** не открыта (перед значком папки стоит "+"), то ее можно открыть командой "**Object Open**". Если выбрать список из папки **Global Variables** и открыть его, то появится окно с ранее объявленными глобальными переменными. Редактор глобальных переменных работает точно так же, как и редактор раздела объявлений.

Структурирование глобальных переменных

При большом количестве глобальных переменных их можно структурировать, разделив на несколько именованных списков.

Чтобы создать новый список переменных, нужно выбрать папку **Global Variables** или один из входящих в нее объектов и выполнить команду "**Project**" "**Object Add**". В появившемся диалоговом окне надо задать имя нового списка. Созданный с этим именем список будет иметь ключевое слово **VAR_GLOBAL**. Если вы хотите получить объект с конфигурационными переменными, измените ключевое слово на **VAR_CONFIG**.

6.2.1 Глобальные переменные

Что такое глобальные переменные?

Объявленные как глобальные, «нормальные» переменные, перманентные переменные и константы имеют область видимости, включающую весь проект. Более того, сетевые переменные (Network variables) могут служить для передачи данных между несколькими абонентами сети.

Обратите внимание: Если в некотором программном компоненте проекта объявлена локальная переменная, имя которой совпадает с именем глобальной переменной, то в данном компоненте будет работать локальная переменная!

Нельзя давать одинаковое имя двум глобальным переменным. Так, например, если вы определили переменную "var1" в PLC Configuration, то при объявлении ее в списке глобальных, вы получите сообщение об ошибке.

Сетевые переменные

Примечание: Работа с сетевыми переменными должна быть поддержана целевой платформой и разрешена в ее настройках (категория Network functionality).

Механизм автоматического обмена значениями переменных в сети (альтернативный вариант - это управляемый обмен через Менеджер параметров) дает возможность нескольким контроллерам (CoDeSys совместимых) в сети совместно использовать значения определенных общедоступных переменных. При этом не нужно создавать никаких дополнительных функций. Достаточно, чтобы абоненты сети имели совместимые настройки конфигурации сети и абсолютно идентичные списки сетевых переменных. Поэтому рекомендуется не создавать такие списки вручную в каждом проекте, а загружать их из одного отдельного файла.

Внимание! В настоящее время не поддерживается функция online коррекции сетевых переменных. Изменения сетевых переменных не обнаруживаются как модификация проекта!

Создание списков глобальных переменных

Наименование списка глобальных переменных можно изменить (по умолчанию **Global Variables**). Кроме того, можно создать несколько таких списков.

Откройте вкладку Ресурсы в Организаторе объектов и выберите уже существующий список переменных. Дайте команду **'Project' 'Object' 'Add'**. В появившемся диалоговом окне (см. рис. ниже) надо указать имя и параметры списка.


Этот же диалог открывается командой **'Project' 'Object' 'Properties'**, доступной для определенных в проекте списков.

Задайте имя списка в строке **"Name of the global variable list:"**

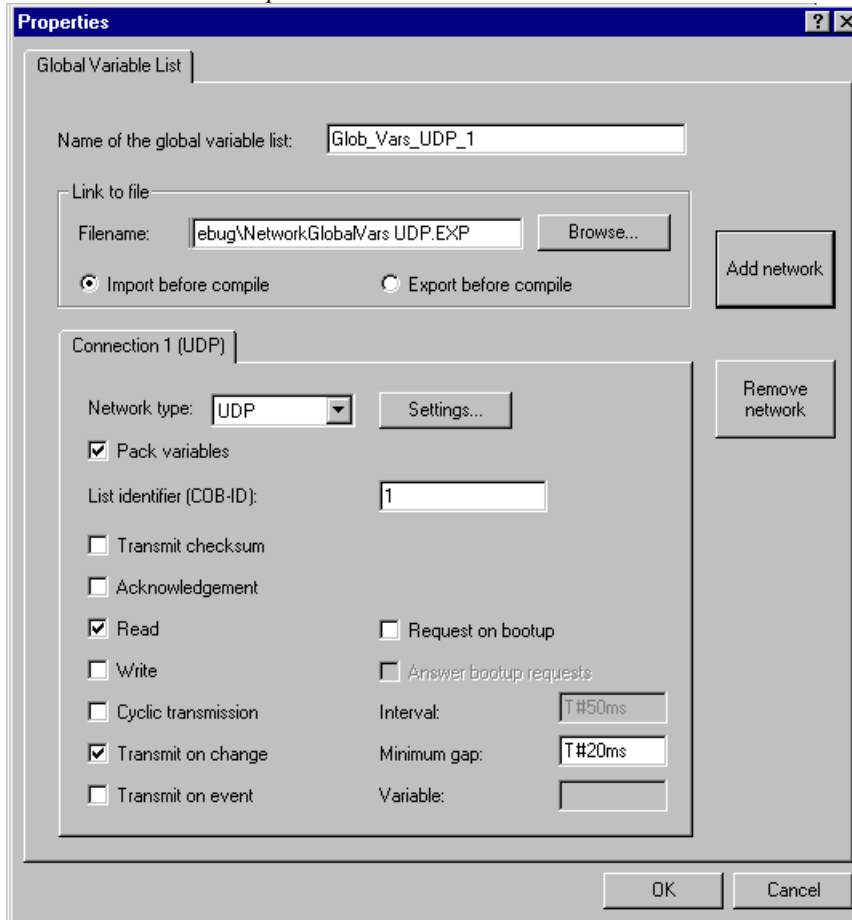
Связать с файлом (Link to file):

Если у вас имеется экспортный файл (*.esp) или DCF файл (*.dcf), содержащий глобальные переменные, вы можете связать с ним данный список. Для этого нужно записать путь и имя файла в поле **Filename**. Используйте кнопку **Browse** для доступа к стандартному диалогу выбора файлов. DCF файлы будут преобразованы согласно МЭК синтаксису при вызове.

Опция **Import before compile** приводит к чтению внешнего файла переменных перед каждой компиляцией. Опция **Export before compile** сохраняет список переменных во внешнем файле перед каждой компиляцией.

После подтверждения ввода кнопкой ОК будет создан новый список. Списки глобальных переменных отмечены иконкой . В любое время вы можете исправить заданные свойства, вызвав данный диалог командой **'Project' 'Object' 'Properties'**.

Диалог свойств списка переменных



Конфигурация сетевых переменных

Если опция **'Support network variables'** включена в настройках целевой платформы, то в данном диалоге будет доступна кнопка **<Add network>**. Нажатие этой кнопки расширяет диалог, и он будет выглядеть, как показано на рисунке. Если данная опция не включена, то кнопка недоступна.

Connection <n> (<Network type>): В нижней части диалога вы можете создать наборы конфигураций до четырех сетевых соединений. Каждая конфигурация определяется на отдельной вкладке и задает параметры сетевого обмена для выбранного списка. Аналогичным образом должны быть определены параметры этого списка переменных для других абонентов сети.

Если никакие конфигурации еще не определены, то для UDP сети будет отображена единственная вкладка **'Connection 1 (UDP)'**. Каждое нажатие кнопки **'Add network'** приводит к созданию новой вкладки с очередным номером после слова "Connection".

Network type: выберете необходимый тип сети из списка доступных для данной целевой платформы. Например: „CAN" или „UDP".

Settings: эта кнопка открывает диалог настроек для соответствующей сети:

UDP:

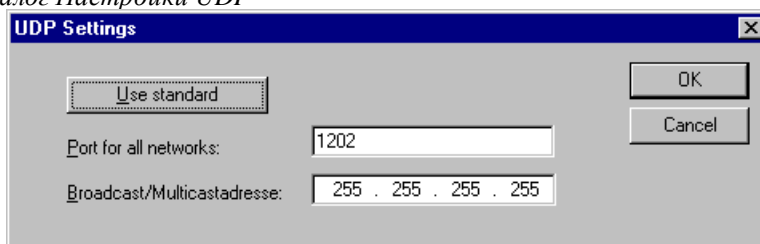
Кнопка **Use standard** определяет использование стандартного порта (Port 1202) для обмена данными с другими абонентами. Адрес **Broadcast/Multicast** устанавливается в "255 . 255 . 255 . 255", что означает: обмен данными будет происходить со всеми абонентами сети.

Port: задайте здесь необходимый порт если он отличается от заданного по умолчанию. Убедитесь, что другие абоненты сети также используют этот порт! Если вы имеете более одного UDP соединения в проекте, то номер порта будет автоматически изменяться для всех наборов конфигураций.

Broadcast/Multicast address: задайте адреса соответствующей подсети, если необходимо изменить настройки по умолчанию (например, "197 . 200 . 100 . 255", если вы хотите взаимодействовать со всеми абонентами с IP-адресами 197 . 200 . 100 . x).

Для Win32 систем, Broadcast/Multicast адреса должны соответствовать маске подсети в конфигурации TCP/IP на PC.

Диалог Настройки UDP

CAN:

Controller Index: индекс контроллера CAN, которым переменные должны передаваться.

Следующие опции могут быть активированы в этой конфигурации:

Pack variables: переменные будут собираться при передаче в пакеты (телеграммы), размер которых зависит от сети. Если данная опция выключена, каждая переменная помещается в отдельный пакет.

Variable telegram number: идентификационный номер первого пакета данных (по умолчанию 1). Последующие пакеты нумеруются по возрастанию.

Include Checksum: контрольная сумма будет помещена в каждый пакет. Данная контрольная сумма будет проверена получателем для проверки отсутствия искажений данных. Пакеты с ошибочной контрольной суммой игнорируются, и, если задано (см. 'Use acknowledge transfer'), то будет дан соответствующий ответ.

Use acknowledged transfer: каждое сообщение подтверждается приемником. Если передатчик не получит хотя бы одно подтверждение, будет сформировано сообщение об ошибке.

Read: значения переменных списка читаются; если опция отключена, то передаваемые в сети значения игнорируются.

Request at Bootup: если локальный узел является «читающим» (опция 'Read' включена), то при перезагрузке актуальные значения переменных будут запрошены из «записывающих» узлов независимо от других условий передачи (время, событие), управляющих нормальной коммуникацией. Обязательное условие: в конфигурации записывающих узлов опция 'Answer Bootup requests' должна быть включена ! (см. ниже).


Write: переменные записываются; применимы следующие опции:

Answer Bootup requests: если локальный узел является «записывающим» (опция 'Write' включена), то будут формироваться ответы на запросы чтения при перезагрузке (см. Request on Bootup). Это означает, что актуальные значения переменных будут переданы даже при отсутствии условий коммуникации (время, событие).

Cyclic transmission: переменные будут записываться с интервалом, заданным в поле **Interval** (нотация длительности, например T#70ms).

Transmit on change: переменные будут записываться, только если их значения изменились. При этом Minimum ограничивает минимальный интервал между передачами.

Transmit on event: переменные списка будут записываться, когда переменная **Variable** принимает значение TRUE.

Списки глобальных сетевых переменных отмечены символом  в Организаторе объектов.

Примечание: Если глобальная сетевая переменная используется в нескольких задачах, то при вызове каждой задачи проверяются условия, заданные в конфигурации. Будет или нет передаваться значение переменной, зависит от того, закончился ли заданный интервал. При каждой передаче интервальный таймер сбрасывается.

Передача выполняется системой исполнения без вмешательства прикладной программы. Никаких дополнительных функций не нужно.

Редактирование списков глобальных и сетевых переменных

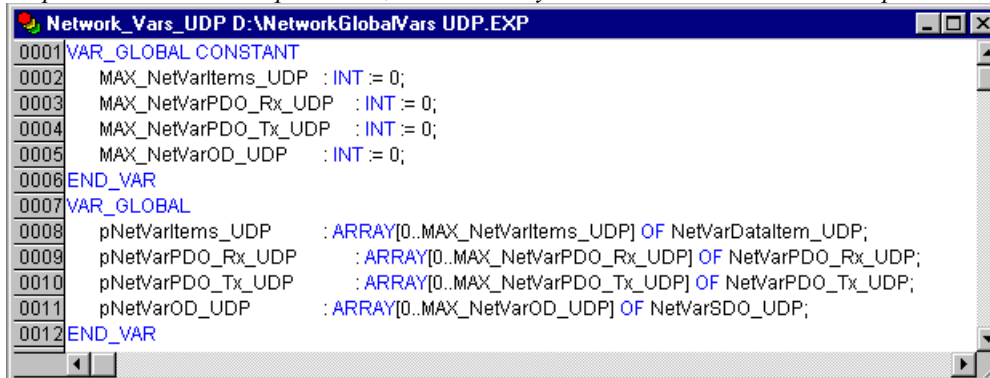
Редактор глобальных переменных работает аналогично редактору раздела объявлений. Если подключен внешний список переменных, то его редактировать нельзя. Внешний список редактируется вне проекта и считывается каждый раз, когда проект открывается или компилируется.

Синтаксис:

```
VAR_GLOBAL
    (*объявление переменных *)
END_VAR
```

Сетевые переменные можно применять, только если это поддержано выбранной целевой платформой.

Пример списка сетевых переменных, созданных путем подключения UDP.EXP файла:



```
Network_Vars_UDP D:\NetworkGlobalVars UDP.EXP
0001 VAR_GLOBAL CONSTANT
0002   MAX_NetVarItems_UDP : INT := 0;
0003   MAX_NetVarPDO_Rx_UDP : INT := 0;
0004   MAX_NetVarPDO_Tx_UDP : INT := 0;
0005   MAX_NetVarOD_UDP : INT := 0;
0006 END_VAR
0007 VAR_GLOBAL
0008   pNetVarItems_UDP : ARRAY[0..MAX_NetVarItems_UDP] OF NetVarDataItem_UDP;
0009   pNetVarPDO_Rx_UDP : ARRAY[0..MAX_NetVarPDO_Rx_UDP] OF NetVarPDO_Rx_UDP;
0010   pNetVarPDO_Tx_UDP : ARRAY[0..MAX_NetVarPDO_Tx_UDP] OF NetVarPDO_Tx_UDP;
0011   pNetVarOD_UDP : ARRAY[0..MAX_NetVarOD_UDP] OF NetVarSDO_UDP;
0012 END_VAR
```

Редактирование списков перманентных переменных

Существуют два типа перманентных переменных:

Retain variables, сохраняющие свои значения при выключении питания и сбросе 'Online' 'Reset' в CoDeSys. **Persistent variables** сохраняют свои значения при сбросе без выключения питания, управляемом останове системы исполнения (стоп, старт).

Реманентные переменные объявляются с дополнительным ключевым словом **RETAIN** или **PERSISTENT**.

Синтаксис:

```
VAR_GLOBAL RETAIN
    (*объявление переменных*)
END_VAR
VAR_GLOBAL PERSISTENT
    (*объявление переменных*)
END_VAR
```

Persistent переменные не являются автоматически Retain переменными! При необходимости используйте комбинированное объявление:

```
VAR_GLOBAL RETAIN PERSISTENT или VAR_GLOBAL PERSISTENT RETAIN
```

Редактирование списков глобальных констант

Глобальные константы дополнительно получают ключевое слово **CONSTANT**.

Синтаксис:

```
VAR_GLOBAL CONSTANT
    (*объявление переменных*)
END_VAR
```

6.2.2 Конфигурационные переменные

Использование прямых адресов в функциональных блоках противоречит идеологии независимости данных разных экземпляров функционального блока. Конфигурационные, или "шаблонные", переменные решают эту проблему.

В показанном ниже примере функциональный блок locio имеет логический вход %I и выход %Q. Прямые адреса переменных заменены символом шаблона "*".

Пример:

```
FUNCTION_BLOCK locio
VAR
    loci AT %I*: BOOL := TRUE;
    loco AT %Q*: BOOL;
END_VAR
```

Далее объявление экземпляров блока Hugo и Otto может выглядеть, например, так:

Пример.

```
PROGRAM PLC_PRG
VAR
    Hugo: locio;
    Otto: locio;
END_VAR
```

Конкретное распределение прямых адресов переменных реализовано в разделе ресурсов Variable Configuration:

```
VAR_CONFIG
```

```

PLC_PRG.Hugo.loci AT %IX1.0 : BOOL;
PLC_PRG.Hugo.loco AT %QX0.0 : BOOL;
PLC_PRG.Otto.loci AT %IX1.0 : BOOL;
PLC_PRG.Otto.loco AT %QX0.3 : BOOL;
END_VAR

```

Объявления переменных состоят из пути (имени экземпляра) и имени переменных. Если задан не существующий путь, будет сформировано сообщение об ошибке. Ошибка возникает и в противоположной ситуации, если отсутствует определение объявленной в функциональном блоке конфигурационной переменной. Все объявленные с использованием "*" переменные должны быть определены в разделе **Variable Configuration**. Типы данных, указанные в шаблоне и при определении адресов, обязаны совпадать.

Для вставки списка всех необходимых определений конфигурационных переменных используйте команду "All Instance Paths" из меню "Insert".

"Insert" "All Instance Paths"

Создает заготовку определений **VAR_CONFIG**, включающую описания всех определенных через шаблоны переменных для всех экземпляров функциональных блоков. В полученной заготовке остается только расставить прямые адреса.

Команда доступна, только если проект откомпилирован. ("Project", "Rebuild All").

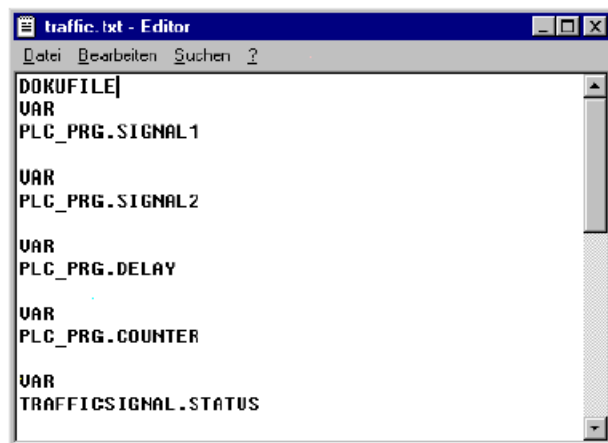
6.2.3 *Файл комментариев переменных*

Если существует задача создать несколько документов, описывающих проект (например, на английском и на русском языке), или надо документировать несколько похожих проектов, использующих одинаковые переменные, то можно сэкономить массу времени, создав отдельный файл комментариев переменных. Для создания шаблона этого файла используются команды: "Extras" "Make Docuframe File". Созданный файл можно редактировать в любом текстовом редакторе. Шаблон начинается строкой **DOCUFILE**. Далее идет список всех переменных проекта. Под каждую из них отводятся 3 строки: **VAR**, которая показывает, что начинается описание переменной, потом строка, содержащая имя переменной, и, наконец, пустая строка. В эту строку нужно вписать комментарий к переменной.

Шаблоны для переменных, не требующих комментариев, можно удалить.

Если нужно, для одного проекта можно создать несколько файлов комментариев.

Окно редактора файла шаблона комментариев



Чтобы использовать созданный ранее файл комментариев, надо выбрать команду "**Extras**" "**Link File**". Теперь при распечатке проекта в тех местах, где описаны переменные, будет добавлен соответствующий комментарий. Эти комментарии появляются только при распечатке.

"Extras" "Make Docuframe File"

Команда используется для создания шаблона файла комментариев и доступна лишь тогда, когда активно одно из окон глобальных переменных. Команда открывает диалоговое окно для создания файла с новым именем. В поле для ввода имени файла всегда предлагается расширение *.txt. После ввода имени файла будет создан текстовый документ, содержащий список всех переменных проекта.

"Extras" "Link Docu File"

Подключить файл комментариев переменных. При ее вызове появится диалоговое окно, где нужно выбрать имя файла. Теперь при распечатке проекта или его частей в тексте программ будут вставлены соответствующие комментарии. Комментарии вставляются только при распечатке.

Для создания файла комментариев используется команда "**Extras**" "**Make Docuframe File**".

6.3 Конфигурация тревог (Alarm Configuration)

6.3.1 Обзор

Сигнальная система, встроенная в CoDeSys, позволяет обнаруживать критические состояния процесса, записывать и визуализировать их для пользователя при помощи элементов визуализации. Механизм работы сигнальной системы может выполняться в CoDeSys или альтернативно в ПЛК. Обработка тревог в ПЛК задается опциями категории 'Визуализация' целевой платформы.

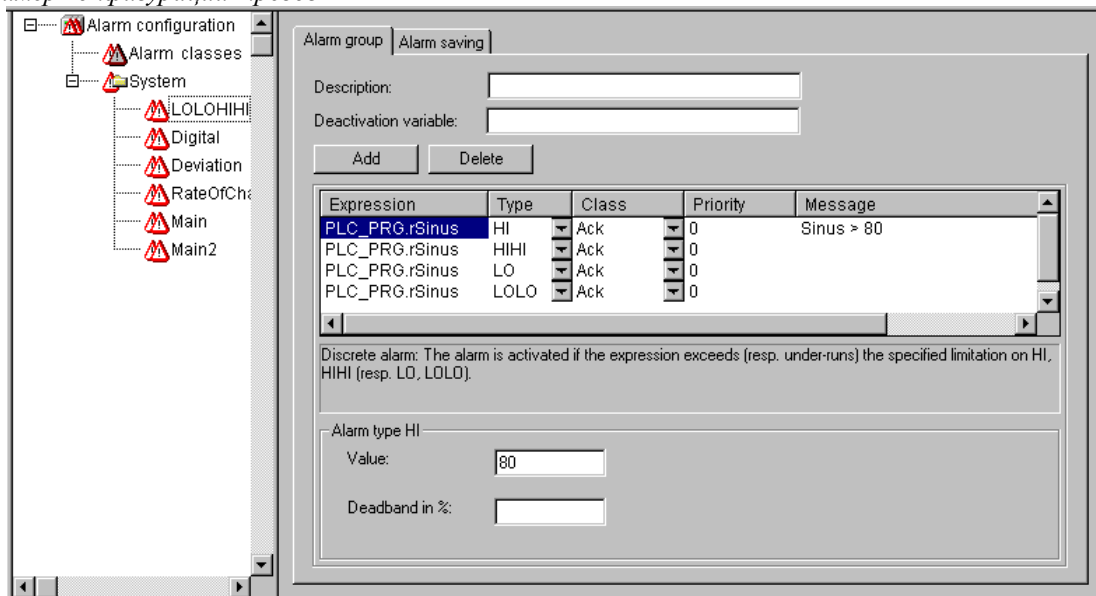
Для конфигурации сигнальной системы используется объект 'Alarm configuration' на вкладке 'Resources'.

Здесь вы определяете **Классы** (alarm classes) и **Группы** тревог (alarm groups). Класс служит для определения параметров, присущих данному виду тревоги. Группа служит для конкретной конфигурации одной или нескольких тревог (которым сопоставлен некоторый класс и соответствующие параметры), используемых в проекте. Таким образом, класс полезен для структурирования тревог. Различные группы тревог формирует пользователь, вставляя соответствующие разделы под головным заголовком '**System**' в дереве конфигурации.

Для **визуализации** тревог в CoDeSys предусмотрен специальный элемент **Alarm table**. Используя такую таблицу, пользователь может наблюдать и подтверждать тревоги.

Если необходима история (**History**), то есть запись тревожных событий (**Alarm-Events**), log-файл должен быть заранее определен и для каждой группы определены параметры записи.

Открыв объект 'Alarm configuration' на вкладке Resources, вы увидите диалог '**Alarm configuration**'. Это разделенное на две части окно, аналогичное по принципам работы окнам PLC Configuration и Task configuration. В левой части представлено дерево конфигурации, в правой - соответствующие диалоги настройки конфигурации.

Пример конфигурации тревог

Щелчок мыши по значку "плюс" раскрывает элементы дерева 'Alarm configuration'. Если вы создали новую конфигурацию тревог, то она состоит только их двух элементов 'Alarm classes' и 'System'.

6.3.2 Общая информация и терминология

Использование сигнальной системы в CoDeSys требует понимания следующих определений:

Тревога (Alarm): тревога - это событие, обусловленное определенными условиями (значением выражения).

Приоритет (Priority): приоритет определяет степень важности тревоги. Самый высокий приоритет - "0". Самый низкий - "255".

Состояние тревоги (Alarm state): выражение / переменная, связанная с тревогой и принимающая следующие значения: NORM - тревоги нет, INTO - тревога только что произошла, ACK - тревога произошла и подтверждена пользователем, OUTOF - условия тревоги "ушли" (но она не подтверждена!).

Промежуточное состояние (Sub-State): условия тревоги могут включать пределы (Lo нижний, Hi верхний) и "чрезвычайные" пределы (LoLo, HiHi). Например: значение выражения растет и достигает предела HI, происходит так называемая HI-тревога. Если значение выражения продолжает расти и достигает чрезвычайного предела прежде, чем она подтверждена пользователем, то HI-тревога станет подтвержденной автоматически и только HIHI-тревога остается в списке (мы говорим о внутреннем списке, используемом механизмом обслуживания тревог). В этом случае HI-состояние называют промежуточным состоянием.

Подтверждение тревог (Acknowledgement of alarms): Главная цель тревог состоит в том, чтобы сообщить пользователю о критических ситуациях. При этом часто необходимо удостовериться, что пользователь заметил эту информацию (возможные действия по тревоге заданы в конфигурации класса). Пользователь должен подтвердить тревогу, чтобы удалить ее из списка.

Тревожное событие (Alarm Event): Тревожное событие нельзя путать с условиями тревоги. Условия тревоги могут иметь силу в течение длительного периода времени. Тревожное событие - это тот момент, при котором мы фиксируем тревогу или изменение ее состояния. В конфигурации тревог CoDeSys для трех типов тревожных событий и соответствующих состояний применяются одинаковые названия (INTO, ACK, OUTOF).

В CoDeSys обеспечиваются следующие особенности:

- Деактивация формирования как отдельных тревог, так и их групп.
- Управление отображением тревог путем задания групп и приоритетов.
- Запись всех тревожных событий в таблицу
- Элемент визуализации 'Alarm table'

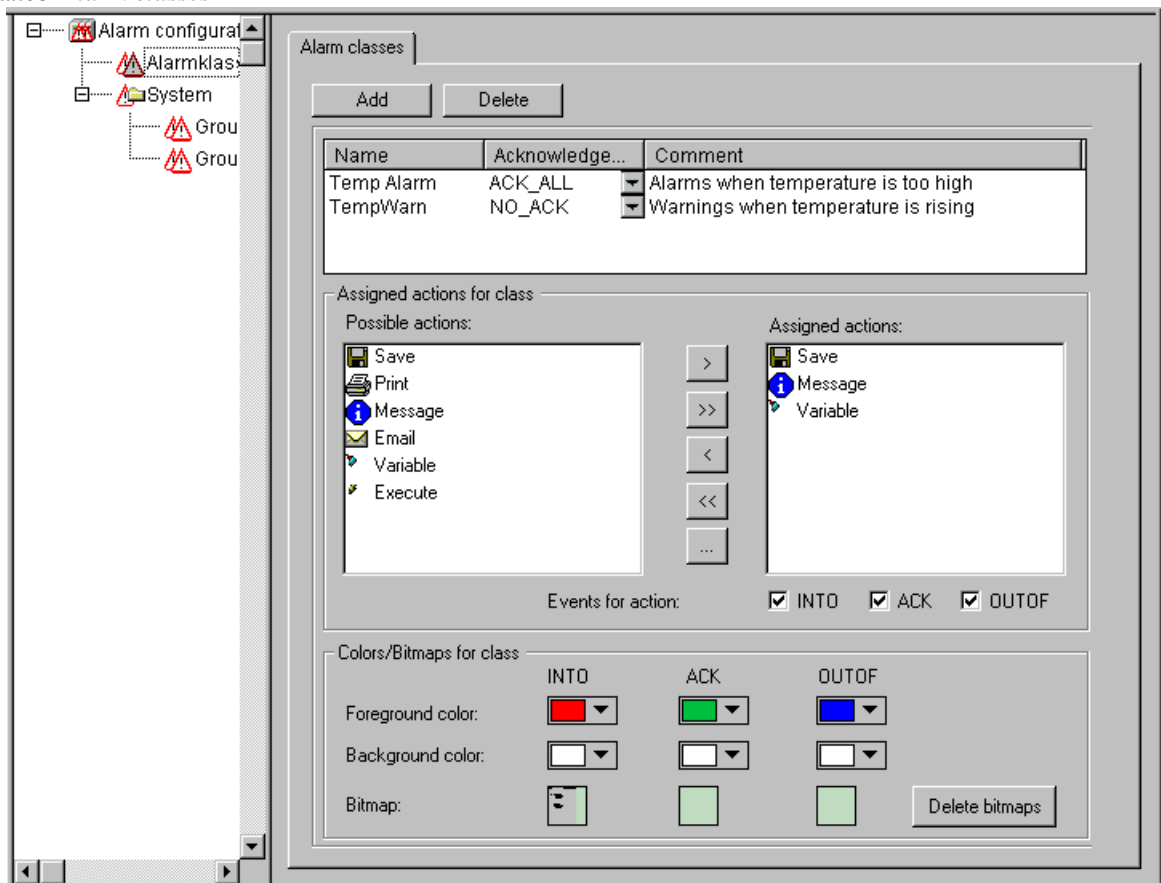
6.3.3 Классы тревог

Классы тревог используются для описания некоторых общих критериев тревог, типа, способа подтверждения пользователем, действий, которые должны автоматически выполняться по тревожным событиям, цветов и картинок, используемых для визуализации. Классы тревог определены глобально в конфигурации тревог и доступны при конфигурировании групп тревог.

Конфигурирование классов тревог:

Выберите объект 'Alarm classes' в дереве конфигурации тревог, и вы увидите диалог конфигурирования классов.

Диалог 'Alarm classes'



Нажмите кнопку **Add** для создания нового класса.

Вслед за этим в верхнем окне будет вставлена строка, изначально с единственным элементом "NOACK" (без подтверждения) в колонке **Acknowledgement** (Подтверждения). Определите название для нового класса в соответствующем поле колонки **Name** (редактирование строки доступно при щелчке мыши). Если необходимо, измените тип подтверждения в колонке Acknowledgement.

Возможны следующие варианты подтверждения:

NO_ACK: подтверждение не требуется.

ACK_INT0: "приход" условий тревоги (статус "INT0") должен быть подтвержден пользователем.

ACK_OUTOF: "уход" условий тревоги (статус "OUTOF") должен быть подтвержден пользователем.

ACK_ALL: "приход" и "уход" условий тревоги должен быть подтвержден пользователем.

Дополнительно вы можете ввести комментарий **Comment**.

Определения новых классов добавляются в конец списка. Используйте кнопку **Delete** для удаления выбранного класса из списка.

Присвоение действий для классов <class name>:

Каждый класс, определенный в верхнем окне, может иметь список действий, которые будут выполнены по тревожному событию.

В списке возможных действий (**Possible actions**) выберите необходимое и нажмите кнопку ">", чтобы перенести его в область Назначенных действий (**Assigned actions**). В итоге эта область будет содержать все действия, назначенные данному классу тревог. Кнопкой ">>" вы можете добавить сразу все действия. Кнопка "<" удаляет одно действие. Кнопка "<<" удаляет сразу все действия из области Назначенных действий. Кнопка "..." открывает соответствующий диалог детализации действия: адрес электронной почты, выбор принтера, переменная состояния или выполняемая программа и, если нужно, текст сообщения.

Допустимы следующие типы действий (**Possible actions**):

<i>Действие</i>	<i>Описание</i>	<i>Установки в соответствующем диалоге:</i>
Save:	Тревожное событие будет записано в log файл. Соответствующий файл должен быть задан в конфигурации группы!	Настройки задаются при определении группы, в диалоге Alarm saving.
Print:	Текст сообщения будет отправлен на печать.	Printer: выбор принтера из доступных в системе; Outputtext: текст сообщения (см. ниже), которое будет отпечатано. <hr/> Данная функция не поддерживается в целевой визуализации.
Message:	В текущей визуализации будет открыто окно сообщений с заданным текстом.	Message: текст сообщения, которое будет дано в окне сообщений. <hr/> Данная функция не поддерживается в целевой визуализации.
E-Mail:	Отправка e-mail сообщения.	From: адрес отправителя; To: адрес получателя; Subject: тема; Message: текст сообщения (см. ниже); Server: имя smtp сервера.

Variable:	Переменная CoDeSys программы получит соответствующий статус.	Variable: имя переменной: вы можете выбрать переменную с помощью Ассистента ввода (<F2>): логическая переменная будет отображать состояние: NORM =0 и INTO=1. Целочисленная переменная будет отображать состояние: NORM =0, INTO =1, ACK =2, OUTF =4; строковая переменная будет получать текст сообщения из поля Message (см. ниже).
Execute:	Запускается выполнение файла внешней программы.	Executable file: имя исполняемого файла (например, notepad.exe), вы можете использовать кнопку "...", вызывающую стандартный диалог выбора файла; Parameter: параметры, которые будут добавлены в командную строку при вызове exe-файла).

Задание текста сообщения:

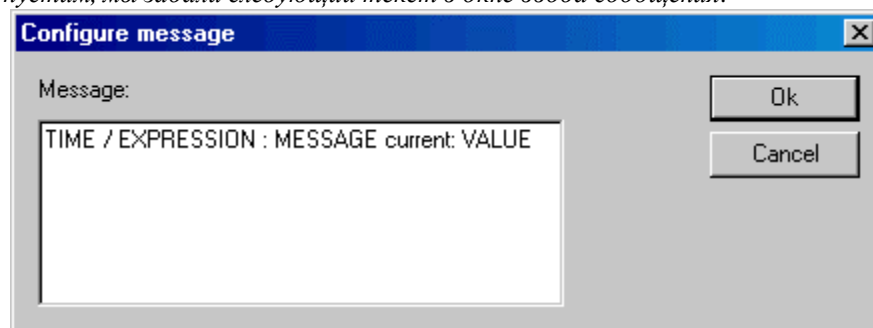
Для действий типа 'Message', 'Print', 'Email' или 'Variable' вы можете задать текст, который будет выводиться по тревожному событию. Перевод строки в тексте формируется комбинацией клавиш <Ctrl>+<Enter>.

В тексте сообщений можно использовать следующие заместители (**placeholders**):

MESSAGE	Текст сообщения, определенный в конфигурации группы или данной тревоги.
DATE	Дата тревоги (INTO).
TIME	Время тревоги.
EXPRESSION	Выражение (определено в группе), вызвавшее тревогу.
PRIORITY	Приоритет тревоги (определен в группе).
VALUE	Текущее значение выражения (см. выше).
TYPE	Тип тревоги (определен в группе).
CLASS	Класс тревоги (определен в группе).
TARGETVALUE	Заданная величина для типов тревоги DEV+ и DEV- (определено в группе).
DEADBAND	«Мертвая зона» тревоги (определена в группе).
ALLDEFAULT	Произвольная информация о тревоге, заданная для записей log файла в группе (History).

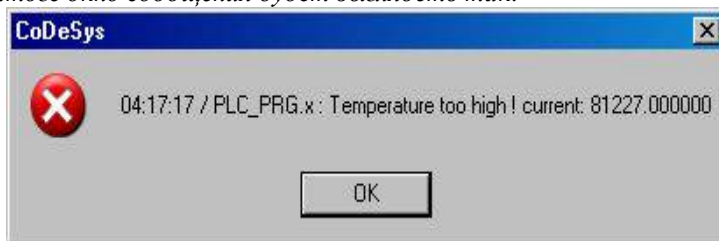
Пример определения сообщения о тревоге:

Допустим, мы задали следующий текст в окне ввода сообщения:



Далее мы определили группу и задали текст (Message): "Temperature too high!".

В итоге окно сообщения будет выглядеть так:



Примечание: Можно сделать так, чтобы текст сообщения выводился на разных языках, если тексты определены в *.vis-файле или файле перевода *.tlf. НО: в этом случае текст должен быть записан между двух символов "#" (например: "#Temperature critical !#" и "TIME /EXPRESSION: MESSAGE #current#: VALUE"). Это необходимо для подстановки текста из раздела ALARM-TEXT_ITEM файла перевода.

Alarm Events for actions (Тревожные события для действий):

Для каждого действия назначается тревожное событие, запускающее данное действие:

INTO Произошла тревога (Status = INTO).

ACK Подтверждение выполнено пользователем (Status = ACK).

OUTOF Условия тревоги закончились (Status = OUT OF).

Colors/Bitmaps for class <class name> (Цвета и рисунки для класса)

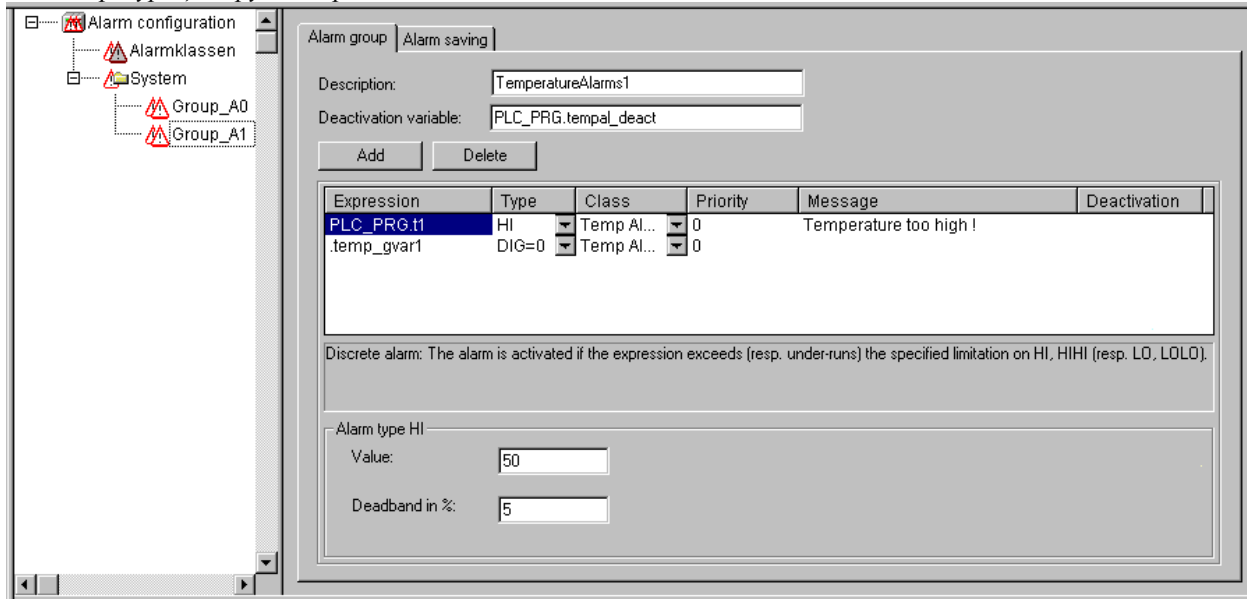
Для каждого класса можно задать цвета и рисунки, позволяющие отличать тревоги разных классов в таблице тревог визуализации. Задайте цвет **Foreground**, фоновый цвет **Background color** для возможных событий INTO, ACK и OUTOF (см. выше). Вы можете использовать стандартный диалог выбора цвета. Щелчок по серому прямоугольнику картинки открывает стандартный диалог выбора файла растрового рисунка (bitmap).

6.3.4 Группы тревог

Группы тревог используются для организации доступных в системе тревог. Каждая тревога назначается определенной группе и управляется по определенным для нее правилам. Все тревоги группы могут быть сопоставлены одной деактивирующей их переменной и имеют общие параметры записи. Обратите внимание, что даже единственная тревога должна быть сконфигурирована в группу.

Группы образуют иерархическую структуру. Диалог **Alarm group** автоматически выводится при выборе группы.

Диалог конфигурации группы тревог:



Имя группы задается в поле **Description**.

Поле **Deactivation variable** может содержать любую логическую переменную проекта. По ее переднему фронту (переход значения в истину) логического сигнала все тревоги группы деактивируются. Тревоги снова активируются по заднему фронту.

Кнопка **Add** добавляет тревогу в группу. Новая строка вставляется в таблицу. Для тревоги определяются следующие параметры:

Expression: выражение, составленное из переменных проекта (например, "a + b"), по которому оцениваются условия тревоги. Используйте Ассистент ввода <F2> и функцию интеллектуального ввода для ускорения работы и исключения ошибок.

Type: тип тревоги из перечисленных ниже. Для каждого типа присутствует определенный комментарий.

DIG=0 дискретная тревога, активна пока выражение дает FALSE.

DIG=1 дискретная тревога, активна пока выражение дает TRUE.

LOLO аналоговая тревога, активна пока значение выражения ниже предела **Alarm type LOLO**. Вы можете определить мертвую зону (Deadband). Пока значение выражения лежит в пределах «мертвой зоны», тревога не будет активизирована, даже если значение было меньше порога LOLO.

LO соответствует LOLO.

HI аналоговая тревога, активна пока значение выражения выше предела **Alarm type HI**. Вы можете определить мертвую зону (Deadband). Пока значение выражения лежит в пределах «мертвой зоны», тревога не будет активизирована, даже если значение было выше порога HI.

HHI соответствует HI.

DEV- отклонение в «-» от заданной величины; тревога активна если значение выражения ниже заданной величины, определенной для **Alarm type DEV-**. Отклонение задается в процентах = заданная величина (target value) * (**deviation in %**) / 100.

DEV+ отклонение в «+» от заданной величины; тревога активна, если значение выражения выше заданной величины, определенной для **Alarm type DEV+**. Отклонение задается в процентах = заданная величина (target value) * (**deviation in %**) / 100.

ROC скорость изменения; тревога становится активной, как только значение выражения начинает изменяться с определенной скоростью. Предел формирования тревоги определяет величина изменения (**Rate of changes**) в единицу времени: в секунду, минуту или час (**units per**).

Class: задайте в этом поле класс тревоги. Выпадающий список содержит определенные в проекте классы. Классы, созданные после последнего сохранения проекта, в список не включаются.

Priority: задайте в этом поле приоритет тревоги в диапазоне 0-152. 0 - это наивысший приоритет. Приоритеты играют роль при сортировке в таблице тревог.

Message: задайте в этом поле текст сообщения о тревоге. Данный текст будет появляться в окне сообщения. Однако нажатие кнопки ОК в этом окне не формирует подтверждение пользователя. Подтверждение (acknowledge) осуществляется через таблицу тревог. Данные могут быть считаны из log файла (если он есть).

Deactivation: логическая переменная проекта деактивирует любое создание тревоги. Ее действие может быть перезаписано переменной, заданной в поле 'Deactivation variable'! (см. выше).

6.3.5 Запись тревог

Для каждой группы тревог может быть определен файл, в который записываются тревожные события, для класса которых задана опция 'Save'.

Выберите группу тревог в дереве конфигурации и откройте вкладку диалога 'Alarm saving':

Диалог 'Alarm saving'

Здесь доступны следующие определения:

Filepath: путь (директория) к файлу, заданному в поле **Filename**; используйте кнопку "..." для доступа к стандартному диалогу выбора директории. Если включена зависящая от платформы опция 'Alarmhandling on PLC', то данный путь игнорируется, а файл записан в загрузочную директорию ПЛК.

Filename: имя файла, в который будут записываться тревожные события. Автоматически создается файл с заданным именем, дополненным индексом, и имеющий расширение ".alm". Индекс это число, которое определяет версию log-файла. Первый файл получает индекс "0"; каждый последующий файл (он создается по условию, определенному в поле **File change event**) будет получать последовательные индексы: 1, 2 и т.д. (Например: "alarmlog0.alm", "alarmlog1.alm").

File change event: здесь определяется условие, при котором необходимо создать новый файл. Возможные варианты: **Never** – никогда, **Hour** – через каждый час, **Day** – каждые сутки, **Week** – каждую неделю, **Month** – каждый месяц, по переднему фронту переменной, определенной в поле **Triggervariable**, по достижении определенного в **Number of records** числа записей.

Поля **Triggervariable** и **Number of records**: описаны выше в File change event.

Delete old files after .. Hours: время хранения «старых» файлов, по истечении этого времени неактуальные log-файлы будут удаляться.

Рассмотрим поля файла истории тревог (log-file) на примере.

Здесь записи двух тревог внесены в таблицу, в заголовках которой пояснено назначение полей:

Date/Time в DWORD	Дата	Время	Событие	Выражение	Тип
1046963332	6.3.03	16:08:52	INTO	PLC_PRG.b	LO
1046963333	6.3.03	16:08:53	ACK	PLC_PRG.n	HIHI

Продолжение таблицы:

Предел	М. зона	Тек. значение	Класс	Приоритет	Сообщение
-30	5	-31	Alarm_high	0	Temperature !
35			Warnng	9	Rising Temp. !

Пример фрагмента log-файла:

```
1046963332,6.3.03 16:08:52,INTO,PLC_PRG.ivar5,HIHI,,,, 9.00,a_class2,0,
1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar4,ROC,2,,, 6.00,a_class2,2,
1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar3,DEV-,,,, -6.00,a_class2,5,
1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar2,LOLO,-35,,3, -47.00,warning,10,warning: low temperature !
1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar1,HI,20,,5, 47.00,a_class1,2,temperature to high! Acknowldge!
```

6.3.6 'Extras' меню: Settings (настройку)

Диалог **Alarm configuration settings** открывается командой 'Extras' 'Settings' в Конфигурации тревог (Alarm Configuration):

Categorie Date/Time:

Здесь настраивается формат записи даты и времени в файл истории тревог. Задайте желаемый формат в соответствии с приведенным ниже синтаксисом. Штрихи и двоеточия необходимо задавать между одинарными верхними кавычками:

дата: dd'-'MM'-'yyyy -> например: "12.Jan-1993"

время: hh'-'mm'-'ss -> например: "11:10:34"

Language:

Задайте здесь языковой файл, который будет использоваться при переключении языков в CoDeSys. В него должны быть включены переводы текстовых строк конфигурации тревог.

Информацию о переключении языков при визуализации вы найдете в отдельном документе «Визуализация в CoDeSys».

Техника перевода проекта на разные языки поясняется в описании команды “Project” “Translate into another language”, в главе «Управление проектом».

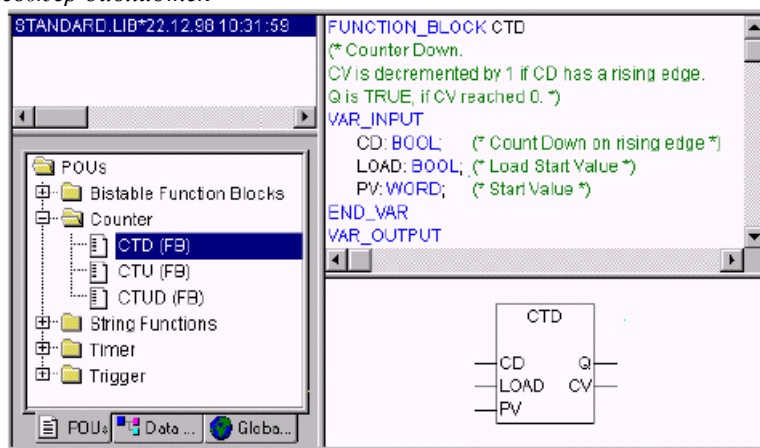
6.4 Менеджер библиотек (Library Manager)

Менеджер библиотек содержит список всех библиотек, которые связаны с проектом. POU, типы данных и глобальные переменные библиотек можно использовать так же, как и определенные пользователем POU типы данных и глобальные переменные.

Информация о включенных библиотеках хранится в проекте, и ее можно увидеть в диалоге 'Informations about external library'. Для вызова этого диалога выберите нужную библиотеку в Менеджере библиотек и дайте команду 'Extras' 'Properties'.

Менеджер библиотек открывается командой "**Window**" "**Library Manager**".

Менеджер библиотек



Использование менеджера библиотек

Окно менеджера библиотек разделено на 3 или 4 области. Список библиотек, соединенных с проектом, находится в левой верхней области. Ниже, в зависимости от выбранной вкладки, показаны переменные POU, типы данных или глобальные переменные выделенной библиотеки.

Папки открываются и закрываются двойным щелчком или нажатием клавиши <Enter>. Перед открытой папкой стоит плюс, перед закрытой – минус.

Если выбрать POU, то в правой верхней части экрана появится раздел объявлений этого POU, а в нижней части – графическое изображение в форме блока с входами и выходами.

При выборе типов данных и глобальных переменных в правой части окна выводится их объявление.

Стандартная библиотека

Библиотека "standard.lib" доступна всегда. Она содержит все функции и функциональные блоки, требуемые стандартом МЭК 61131-3. Разница между стандартными функциями и операторами включает

ся в том, что операторы признаются неявно системой программирования, а стандартные POU должны быть присоединены к проекту (standard.lib).

Исходный текст этих POU находится в C-библиотеке и является компонентом CoDeSys.

Библиотеки, определенные пользователем

Если проект откомпилирован без ошибок, то его можно сохранить как библиотеку. Сам проект при этом не изменится. К созданной библиотеке можно обращаться так же, как и к стандартной библиотеке.

Для библиотек, полностью реализованных в CoDeSys, используйте команду сохранения проекта как **Internal Library**.

Если вы планируете реализовать программные компоненты, объявленные в проекте, на других языках программирования (например C), сохраните проект как внешнюю библиотеку (**External Library**). Вы получите файл библиотеки и дополнительный заголовочный файл с расширением "*.h". Это заголовочный файл C. Он содержит объявления POU, типов данных и глобальных переменных, доступных в данной библиотеке. Если в проекте используется внешняя библиотека, то в режиме эмуляции работает реализация компонентов, описанная в CoDeSys. В реальный ПЛК загружается скомпилированный C код.

Если вы хотите добавить информацию о лицензировании в библиотеку, нажмите кнопку **Edit license info...** и заполните соответствующие поля в диалоге 'Edit Licensing Informationen'. См. 'File' 'Save as...' и отдельный документ «Менеджер лицензирования CoDeSys».

"Insert" "Additional Library"

Этой командой можно присоединять библиотеку к проекту.

В открывшемся диалоговом окне выберите нужную библиотеку с расширением "*.lib". Название библиотеки появится в Менеджере библиотек, и ее объектами можно будет пользоваться как определенными пользователем объектами.

Пути поиска библиотек зависят от состава директорий, определенных в опциях проекта. Если вы присоединяете библиотеку из другой директории, то библиотека будет добавлена в форме полного имени файла. Например: вы присоединяете библиотеку standard.lib из директории "D:\codesys\libraries\standard".

- Если данная директория определена в опциях проекта, то в менеджере проекта будет указано: "standard.lib <дата и время файла >".
- Если в опциях проекта определена директория "D:\codesys\libraries", то в менеджере проекта будет указано: "standard\standard.lib <дата и время файла >".
- Если в опциях проекта нет определения директории, то в менеджере проекта будет указано: "D:\codesys\libraries\standard\standard.lib <дата и время файла >".

При очередном открытии проекта поиск библиотек будет идти в соответствии с записями в менеджере библиотек. Поэтому, если здесь указано только имя файла, то поиск библиотеки будет идти по директориям, указанным в опциях проекта.

Если вы включили **лицензированную библиотеку** и соответствующая лицензия отсутствует, то вы получите сообщение о том, что данная библиотека работает в демонстрационном режиме или о том, что она не лицензирована для выбранной целевой платформы. В это время вы еще можете проигнорировать сообщение или выполнить необходимую процедуру лицензирования. Нарушение лицензии вызовет сообщение об ошибке при компиляции. В этом случае двойным щелчком мыши вы можете открыть диалог 'License information'.

Удаление библиотеки

Удаление библиотеки из проекта в Менеджере библиотек происходит по команде "Edit" "Delete".

'Extras' 'Properties'

Открывает диалог 'Informations about internal (или external) library'. Для внутренних библиотек вы найдете все данные, которые были включены в информацию проекта Project Info (включая информацию о лицензировании). Для внешних библиотек отображается имя библиотеки и путь к ее файлам.

6.5 Бортжурнал (Log)

Бортжурнал - это детальный протокол последовательности действий, которые были выполнены в течение Online сессии. Бортжурнал записывается в двоичный файл (*.log). Пользователь может сохранить какие-нибудь фрагменты этого файла во внешнем протоколе.

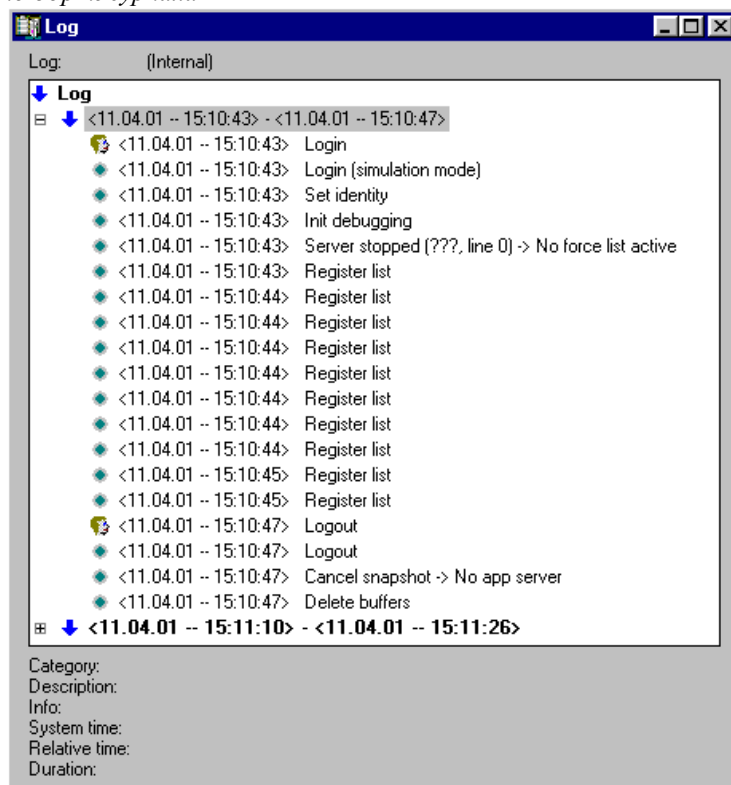
Окно протокола можно открыть в режиме Offline либо Online и использовать его для непосредственного мониторинга действий.

“Window” “Log”

Открывает окно бортжурнала. Аналогично работает выбор 'Log' на вкладке ресурсов.

В окне бортжурнала после слова **Log** дано имя файла. Если используется файл протокола текущего проекта, то здесь присутствует слово “(Internal)”.

Окно бортжурнала



Зафиксированные действия показаны в окне протокола. Последнее действие всегда находится в конце списка. Протоколируются только действия, принадлежащие категориям, выбранным в поле “Field” опций проекта “Log”.

Информация, относящаяся к выбранному действию, находится в нижней части окна:

Category: Категория, к которой относится выбранное действие. Есть 4 категории:

- **User actions:** Пользователь выполнил функцию Online (в основном это команды из меню “Online”)
- **Internal actions:** Внутреннее действие в режиме Online (например, удаление буферов (Delete Buffers) или инициализация отладки (Init Debugging)).
- **Status change:** Статус системы реального времени был изменен (например, выполнение программы было остановлено на точке останова).
- **Exception:** Произошло исключение, например, ошибка связи.

Description: Тип действия. Пользовательские действия имеют те же имена, что и соответствующие пункты меню “Online”. Все остальные действия имеют те же имена, что и соответствующие функции OnlineXXX().

Info: Это поле содержит описание ошибки, которая произошла во время выполнения действия. Оно пусто, если действие выполнено без ошибок.

System time: Системное время начала действия с точностью до секунды.

Relative time: Время начала действия относительно начала Online-сессии с точностью до миллисекунды.

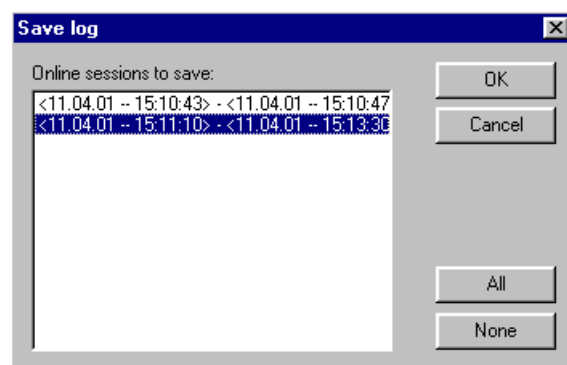
Duration: Продолжительность действия в миллисекундах.

Меню Log

Когда открыто окно боржурнала, в главном меню появляется новый пункт **Log**, включающий следующие команды:

Load.... Загружается внешний файл боржурнала (*.log). Для этого используется стандартный диалог открытия файла. Протокол текущего проекта при этом не удаляется и выводится снова, если началась новая Online сессия или если окно боржурнала закрыто, а потом снова открыто.

Save.... Позволяет сохранить боржурнал целиком или частично во внешнем файле. Для этого выводится диалог выбора Online сессий, которые нужно сохранить.



Выберите необходимые сессии и нажмите кнопку ОК, появится стандартный диалог для сохранения файла.

Display Project Log. Данная команда выполнима, если в данный момент используется внешний файл протокола. Выводит боржурнал текущего проекта.


Хранение файла боржурнала

Независимо от того, используется ли в данный момент информация из внешнего файла или нет (см. выше), протокол текущего проекта автоматически сохраняется в двоичном файле с именем <имя проекта>.log. Путь к этому файлу устанавливается в опциях проекта “Log”. Обычно это та же директория, в которой сохраняется проект.

Максимальное число сохраняемых Online-сессий устанавливается в опциях проекта “Log”. Если число сессий превышает максимальное, то самая старая сессия заменяется новой.

6.6 Конфигуратор ПЛК (PLC Configuration)

6.6.1 Обзор

Объект **PLC Configuration**  (Конфигуратор ПЛК) расположен на вкладке ресурсов Организатора объектов. Конфигурация ПЛК определяет аппаратные средства вашей системы. Здесь задается распределение адресов входов/выходов контроллера, что определяет привязку проекта к аппаратным средствам. На основе описания конфигурации ПЛК CoDeSys проверяет правильность задания МЭК адресов, используемых в программах, на их соответствие фактически имеющимся аппаратным средствам.

Начальный вид окна редактора конфигурации задает файл (файлы) конфигурации *.cfg (см. ниже примечание о совместимости версий) и файлы описания устройств (например, *.gsd, *.eds). Они располагаются в директории, определенной в целевом файле (см. Target Settings), и считываются при открытии проекта в CoDeSys. В любое время вы можете поместить в данную директорию дополнительные файлы.

Файл *.cfg описывает базовую конфигурацию, которая отражается в редакторе ПЛК конфигурации. Здесь же определены элементы, которые пользователь может настраивать самостоятельно.

Внимание: Если файл *.cfg изменен, необходимо перезагрузить CoDeSys чтобы изменения вступили в силу!

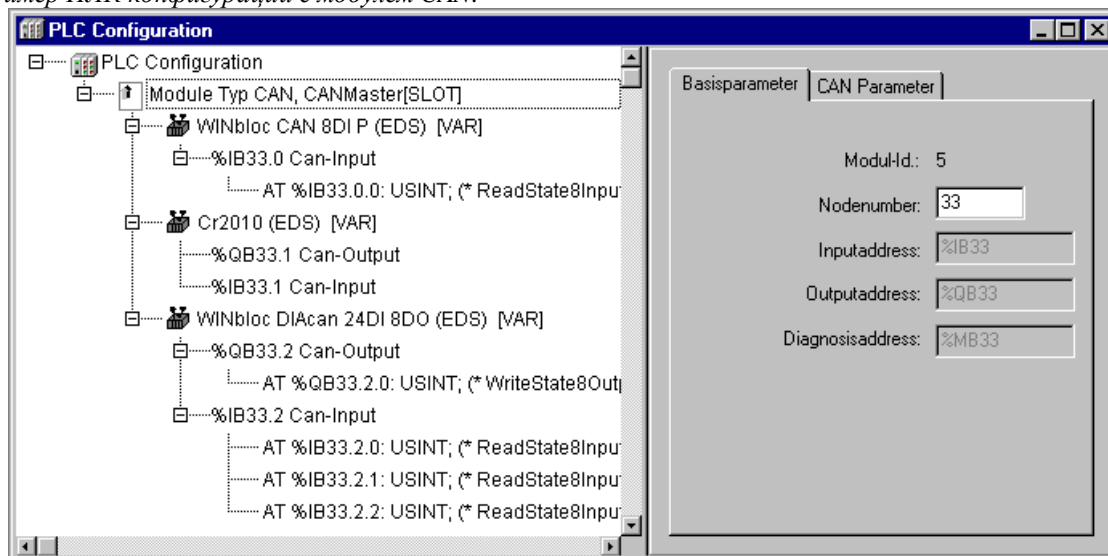
Примечание о совместимости версий: Начиная с версии CoDeSys V2.2 изменился формат файлов конфигурации. Теперь конфигурация задается в файлах с расширением *.cfg. В ранних версиях применялись файлы с расширением *.con. Но: в целевом файле может быть разрешено открытие проектов в старом (old) формате. Это избавляет от необходимости переопределения конфигурации. Вы сможете использовать *.con файлы. Если данная опция отключена, то имеющиеся в проекте конфигурационные данные можно конвертировать в новый формат. Это возможно, только если имеется новый *.cfg файл для данной платформы (См. 'Extras' 'Convert').

Редактор конфигурации **CoDeSys** позволяет подключать удаленные модули ввода/вывода, в том числе CAN и Profibus.

Если поддержано целевой системой, то существует возможность чтения информации из ПЛК: 1) автоматическое сканирование аппаратной структуры и отображение ее в PLC Configuration, 2) отображение диагностических сообщений в CoDeSys, 3) отображение статусной информации в диалогах PLC Configuration.

После окончания настройки двоичный образ конфигурации передается в ПЛК.

Пример ПЛК конфигурации с модулем CAN:



Конфигурация ПЛК отображается в редакторе в виде дерева. Для редактирования элементов применяются команды меню и специализированные диалоги. В конфигурации присутствуют элементы ввода и/или вывода, каждый из которых может содержать вложенные подэлементы (например, CAN-bus или модуль дискретного ввода может содержать 8 входов).

Для входов и выходов могут быть назначены символические имена. Прямые МЭК адреса отображаются в конфигурации для каждого символического имени.

Базовые термины

Конфигуратор ПЛК (PLC Configuration): редактор CoDeSys, в котором определяется состав аппаратных средств и производится настройка определенных параметров ввода-вывода.

Модуль: независимая единица аппаратных средств. Модуль включает набор каналов ввода-вывода. Как и каждый отдельный канал, модуль может иметь параметры. Каждый тип модуля имеет уникальный идентификатор.

Канал: это собственно данные ввода-вывода. Как правило, модуль имеет фиксированный набор каналов или подмодулей. Каждый канал имеет определенный МЭК тип и адрес. Естественно, для каждого канала выделяется определенное пространство памяти. Каждый канал имеет уникальный в пределах данной конфигурации ПЛК идентификатор.

Битовый канал: идентификатор отдельного бита в многобитном канале.

Плоская модель адресации: модель определения МЭК адресов, без спецификации иерархии модулей. Все адресное пространство ввода-вывода представляется в виде плоского набора последовательно пронумерованных ячеек памяти. Если включена опция 'Calculate addresses', то при изменении положения модуля адреса его каналов соответствующим образом смещаются. Альтернативой может служить фиксированная адресация. В этом случае для каждого модуля отводится фиксированное адресное окно, которое определяется физическим расположением (номером слота) модуля. Например: %QB0, %IB26, %MW4.

Иерархическая модель адресации: модель определения МЭК адресов, при котором адрес канала определяется путем указания модуля, подмодуля и номера канала в нем. Например:

%QW2.4.6 : карта CAN bus в VME Slot 2, CAN-модуль Id 4 и канал 6.
 %QX2.4.6.10 : 10й бит этого канала.

Параметр: атрибут канала или модуля. Значение параметра устанавливается интерактивно до компиляции проекта. Оно передается в ПЛК и влияет на работу аппаратуры.

6.6.2 Работа в редакторе конфигуратора ПЛК

Окно редактора конфигуратора ПЛК разделено на две части. В левой части окна показано **дерево конфигурации**. Структура и компоненты дерева определяются главным образом файлом конфигурации, но могут быть изменены пользователем CoDeSys. В правом окне показаны доступные в настоящее время **диалоги конфигурации** в виде одной или нескольких табличных вкладок.

Правая часть окна видна по умолчанию, но может быть скрыта через меню **'Extras' 'Properties'**.

Верхушка конфигурационного дерева начинается с корневого элемента, имя которого определено в файле конфигурации.

Ниже вы увидите другие элементы конфигурации: модули различного типа (CAN, Profibus, I/O, модули последовательного интерфейса и др.).

Выбор элементов

Для выбора элемента щелкните мышкой по его наименованию либо перемещайтесь по элементам с помощью стрелок на клавиатуре. Выбранный элемент обведен прямоугольником из точек.

Элементы, перед которыми стоит значок "плюс", раскрываются на подэлементы. Для развертывания элемента выберите его и щелкните по нему дважды мышкой или нажмите <Enter>. Для свертывания раскрытого элемента (на месте теперь присутствует знак "минус"), нужно выполнить аналогичные действия.

Добавление элементов, 'Insert' 'Insert element', 'Insert' 'Append subelement'

В соответствии с определениями в файле (файлах) конфигурации и файлах описания устройств, считанных, когда проект был открыт, основной состав элементов автоматически помещается в дерево конфигураций. Но если файл конфигурации позволяет, то некоторые дополнительные элементы могут быть добавлены. Для них должны присутствовать файлы описания.

- Команда 'Insert' 'Insert element' вставляет новый элемент перед элементом, выбранным в дереве конфигурации.
- Команда 'Insert' 'Append subelement' добавляет новый подэлемент к выбранному в дереве конфигурации элементу. Подэлемент помещается в последнюю позицию.

Наиболее важные команды присутствуют в контекстном меню (правая клавиша мыши или <Ctrl>+<F10>).

Обратите внимание: Если это поддерживается целевой системой, то может применяться сканирование аппаратных средств и автоматическое представление состава модулей в CoDeSys PLC Configuration.

Замена/переключение элементов, 'Extras' 'Replace element'

В зависимости от определений в файле конфигурации выделенный элемент можно заменить на другой. Аналогичным образом можно переключать каналы элементов на ввод или вывод. Используйте команду 'Extras' 'Replace element'.

Символические имена

Символические имена для модулей и каналов могут быть заданы в конфигурационном файле. В этом случае они будут отражаться в редакторе конфигурации перед определением прямого МЭК адреса (АТ). В конфигурационном файле также определена возможность редактирования и ввода символических имен в редакторе конфигурации ПЛК.

Для ввода символического имени выберите необходимый модуль или канал в дереве конфигурации и щелкните мышкой по текстовому полю перед префиксом прямого МЭК адреса 'АТ'. Аналогично вы можете изменить существующее символическое имя двойным щелчком мыши. Символические имена должны удовлетворять общим правилам создания идентификаторов.

Экспорт/Импорт модулей

Если модуль в конфигурационном файле (*.cfg) определен как «экспортируемый», то в контекстном меню будут присутствовать команды '**Export module**' и '**Import module**'. Контекстное меню относится к модулю, выбранному в дереве конфигурации.

Команда 'Export module' открывает стандартный диалог выбора файла. Здесь необходимо указать имя файла для экспорта. В него будет помещено описание модуля, включая подмодули и их конфигурации в формате XML. Данный файл можно импортировать в другие конфигурации ПЛК с помощью команды 'Import module', если соответствующее определение модуля выделено в дереве конфигурации.

Таким способом можно легко копировать определенный модуль из дерева конфигурации в другие ПЛК конфигурации.

6.6.3 Общие параметры конфигурации ПЛК

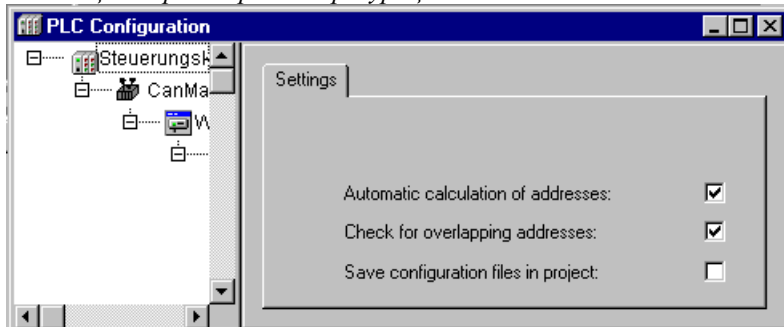
Выберите корневой элемент дерева конфигурации ПЛК. В правой части окна будет отображен диалог 'Settings'. Он содержит общие параметры данной конфигурации ПЛК:

Calculate addresses: адреса для нового вставленного модуля будут вычислены автоматически, в соответствии с его положением в дереве конфигурации и размером. При удалении модуля адреса последующих модулей пересчитываются автоматически. Для пересчета адресов выделенного модуля и последующих модулей применяется команда 'Extras' 'Compute addresses'.

Check for overlapping addresses: при компиляции проекта будет произведена проверка на перекрытие адресов, и при его обнаружении будет дано соответствующее сообщение об ошибке.

Save configuration files in project: информация из конфигурационных файлов *.cfg и файлов описания устройств, относящаяся к текущей ПЛК, конфигурации будет сохраняться в проекте. Таким образом, заданная пользователем конфигурация целиком хранится в проекте. При повторном открытии проекта конфигурация не изменится, даже если **файлы конфигурации утрачены**. Если данная опция не активна, данные из файлов конфигурации будут обновляться при каждом открытии проекта.

Диалог общих параметров конфигурации ПЛК



Глобальный метод адресации (плоская модель или иерархическая модель адресов), применяемый в ПЛК конфигурации, определяется в конфигурационном файле.

Пересчет адресов модуля 'Extras' 'Compute addresses'

Если опция "Calculate addresses" в диалоге 'Settings' активна, то команда 'Extras' 'Compute addresses' запускает пересчет адресов всех модулей. Пересчет идет для всех модулей, начиная с выделенной в дереве конфигурации.

Добавить файл конфигурации 'Add configuration file'

Используйте эту команду из меню 'Extras' для добавления файлов конфигурации проекта. Файлы конфигурации должны помещаться в директории (директориях) заданных в опциях проекта: категория 'Directories', поле 'Configuration files'.

В диалоге **Select configuration file** присутствует фильтр, которым вы можете ограничить открываемые файлы: CAN- (*.eds,*.dcf), Profibus- (*.gsd), конфигурация (*.cfg файлы) или все файлы (*.*) .

После выбора файла производится проверка на наличие данного файла в определенных директориях конфигурации. В этом случае файл не добавляется, о чем будет дано соответствующее сообщение.

Если файл может быть добавлен, то открывается диалог **Select configuration directory**. В нем перечислены все директории конфигураций проекта. Выберите директорию, куда необходимо скопировать файл. Подтвердите выбор кнопкой ОК. Указанный файл сразу же станет доступен в ПЛК конфигурации.

Возврат к стандартной конфигурации 'Extras' 'Standard configuration'

Команда 'Extras' 'Standard configuration' восстанавливает стандартную конфигурацию ПЛК, определенную в *.cfg файле и сохраненную в проекте.

Внимание: В *.cfg файле может быть указано, что стандартная конфигурация должна восстанавливаться при каждом повторном открытии проекта. В этом случае, все изменения, сделанные пользователем будут утрачены!

Преобразование старой ПЛК конфигурации 'Extras' 'Convert'

Команда 'Convert' необходима, если вы открываете проект, содержащий ПЛК конфигурацию, которая создана в CoDeSys до версии **V2. 2**. Если все необходимые файлы конфигурации доступны, команда 'Convert' преобразует существующую конфигурацию в актуальный формат. При выполнении команды будет открыт диалог подтверждения: "Convert the configuration to the new format? Attention: Undo is not possible !" (Конвертировать конфигурацию в новый формат? Внимание: преобразование не обратимо!). Нажмите кнопку **Yes**, конфигурация будет записана в новом формате. Имейте в виду, что обратное преобразование выполнить невозможно!

6.6.4 Диалог специфической настройки параметров

Возможности настройки параметров редактора конфигурации могут быть расширены путем применения специфических для конкретных типов модулей диалогов. Такие диалоги реализуются посредством так называемых 'Hook'-DLL, помещенных в директорию конфигурации и связанных с определенным модулем или каналом. В этом случае, вместо стандартного диалога 'Module parameters' вы увидите диалог, реализованный в DLL.

Пример диалога специфической настройки параметров (Custom Parameters):

Custom Dialog

PgmVersion: 0

LibDir: d:\codesys\lib

ObjectId: 1

SectionName: Module.CPU1

PrmCount: 9

Definierte Parameter

```

1, 10000, Parameter.RateType, XRate, 10
2, 10000, Parameter.RateType, Yrate, 10
3, 10001, Parameter.BoolType, EnableDiags, Yes
4, 10007, Parameter.NameType, Namen, HugoType
5, 107659, Parameter.FileType, DumpFile, D:\CoDeSys\Projekte\Dump.bin
6, 1343, Parameter.NameString, IDString, abc def
7, 1347, Parameter.RealType, Realvalue, 1.876
8, 1348, Parameter.Bool, Boolvalue, TRUE
9, 138740, Parameter.IntTypeHex, Test, 16#A

```

Param1 (Value): 10

6.6.5 Конфигурация модулей ввода/вывода

Базовые параметры модулей ввода/вывода

Диалог базовых параметров модуля ввода/вывода

Base parameters

Module id: 1

Node id: 0

Input address: %IB0

Output address: %QB0

Diagnostic address: %MB4

Comment:

Load module description:

Do not adapt address automatically

Когда модуль ввода/вывода выбран в дереве конфигурации, в правой части окна отображается диалог базовых параметров модуля:

Module id: уникальный идентификатор модуля. Идентификатор задается в конфигурационном файле. Редактировать его нельзя.

Node number: номер позиции модуля задается вхождением в конфигурационном файле или положением в структуре конфигурации.

Input address, Output address, Diagnostic address: адреса областей входов, выходов и диагностики модуля.

Данные адреса отображают модуль в проекте. Они определяются общими установками, режимом адресации и могут допускать редактирование.

Do not adapt address automatically: данная опция доступна, только если это разрешено в конфигурационном файле. Если она активна, то данный модуль не задействован при пересчете адресов. (По умолчанию: опция выключена.)

Comment: текстовый комментарий для модуля.

Load module description: если эта опция отключена, то определение модуля не будет загружаться в проект. По умолчанию эта опция активна. Видимость и доступность редактирования модуля определяется в файле конфигурации *.cfg.

Диагностика в ПЛК конфигурации:

Адрес диагностической области модуля задается в прямоадресуемой маркированной (M) памяти. Для обычных модулей ввода/вывода наличие и функции встроенной диагностики определяются изготовителем. Для сетевых модулей, таких, как CAN или Profibus DP, диагностика работает следующим образом: по указанному адресу размещается информационная структура *GetBusState*, поддерживаемая соответствующей библиотекой изготовителя (например, BusDiag.lib от 3S - Smart Software Solutions).

Каждый раз, когда МЭК программа читает или записывает данные модуля, выполняется запрос модулю на заполнение диагностической структуры. Если хотя бы один модуль в сети вызвал ошибку, то специфическую диагностическую информацию можно получить с помощью функционального блока *DiagGetState*. Он входит в вышеупомянутую библиотеку. Этой возможностью обладают только ведущие сети (masters), определенные в ПЛК конфигурации CoDeSys!

Для доступа к диагностической информации сети, вы должны объявить экземпляр функционального блока **DiagGetState** в своем **CoDeSys** проекте.

Входные переменные DiagGetState:

ENABLE: BOOL;	Функциональный блок начинает работать по переднему фронту ENABLE.
DRIVERNAME: POINTER TO STRING;	Имя драйвера (указатель на имя), к которому должны передаваться запросы на диагностику. Если здесь задан 0, то запросы передаются всем присутствующим драйверам.
DEVICENUMBER:INT;	Идентификатор сети, управляемой драйвером. Например, драйвер Hilscher может поддерживать до 5 сетевых карт. Индекс начинается с 0.
BUSMEMBERID:DWORD ;	Специфический для драйвера идентификатор сетевого модуля (например, для CANopen карты - это NodeID, для PB-DP карты - адрес станции и т.д.).

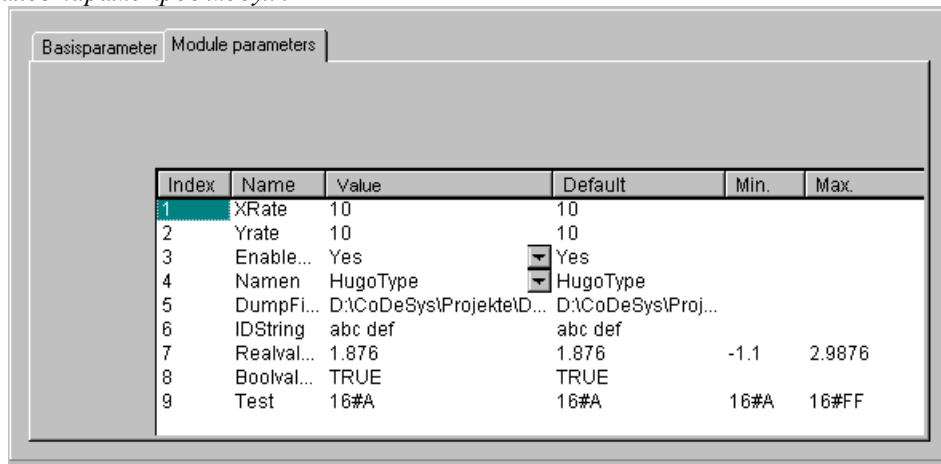
Выходные переменные DiagGetState:

READY:BOOL ;	TRUE: выполнение диагностического запроса было прервано
STATE:INT;	Если READY = TRUE, то STATE принимает следующие значения, отражающие актуальное состояние: -1: ошибочный входной параметр (NDSTATE_INVALID_INPUTPARAM:INT;) 0: функциональный блок не работает (NDSTATE_NOTENABLED:INT;)

	<p>1: функциональный блок читает диагностическую информацию (NDSTATE_GETDIAG_INFO:INT);</p> <p>2: диагностическая информация доступна (NDSTATE_DIAGINFO_AVAILABLE:INT);</p> <p>3: диагностическая информация не доступна (NDSTATE_DIAGINFO_NOTAVAILABLE:INT);</p>
EXTENDED-INFO:ARRAY[0..129] OF BYTE;	<p>До 100 байт специфической диагностической информации изготовителя. Один байт зарезервирован, его биты 0 – 2 используются следующим образом:</p> <p>Bit 0: Сетевой модуль присутствует в конфигурации ПЛК.</p> <p>Bit 1: Сетевой модуль доступен в сети.</p> <p>Bit 2: Сетевой модуль сообщает ошибку.</p>

Параметры модуля / специфические параметры модуля ввода/вывода

Диалог параметров модуля



Index	Name	Value	Default	Min.	Max.
1	XRate	10	10		
2	Yrate	10	10		
3	Enable...	Yes	Yes		
4	Namen	HugoType	HugoType		
5	DumpFi...	D:\CoDeSys\Projekte\ID...	D:\CoDeSys\Proj...		
6	IDString	abc def	abc def		
7	Realval...	1.876	1.876	-1.1	2.9876
8	Boolval...	TRUE	TRUE		
9	Test	16#A	16#A	16#A	16#FF

В этом диалоге отражены параметры, определенные в файле описания устройства. Редактировать можно только значения в столбце 'value'.

Index: последовательный номер параметра модуля.

Name: имя параметра.

Value: значение параметра (редактируемое).

Первоначально устанавливаются значения по умолчанию. Значения могут быть введены непосредственно (в виде чисел) или в символическом представлении. Если данный параметр не отмечен в файле описания как 'Read Only', то его значение можно изменять. Для этого щелкните мышкой на поле ввода или на выпадающем списке (если он есть). Если значение - это имя файла, то откройте диалог 'Open file' двойным щелчком мыши и выберите нужный файл.

Default: значение по умолчанию.

Min.: минимальное значение параметра (только если не используются символические имена).

Max.: максимальное значение параметра (только если не используются символические имена).

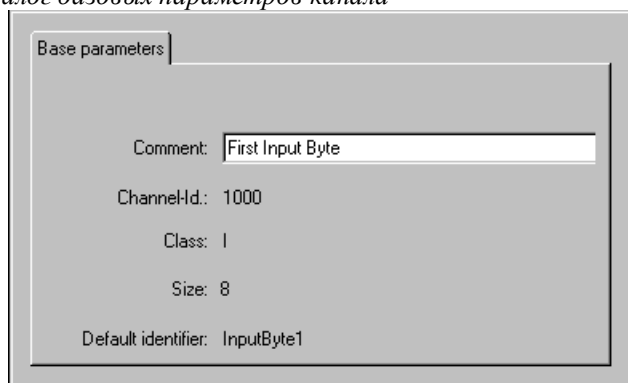
Всплывающая подсказка может содержать дополнительную информацию о выбранном параметре.

Место описанного диалога параметров модуля может занять специфический диалог изготовителя модуля. В этом случае диалогом управляет внешнее приложение (Hook-DLL), прописанное в файле конфигурации.

6.6.6 Конфигурация канала

Базовые параметры канала

Диалог базовых параметров канала



Channel-Id: глобальный уникальный идентификатор канала.

Class: класс определяет работу канала как вход (I), выход (Q), либо вход и выход (I&Q), либо с переключением направления (I|Q). Если направление работы канала переключается, то это можно сделать командой 'Extras' 'Replace element'.

Size: размер канала [Byte].

Default identifier: символьное имя канала.

Имя канала определено в файле конфигурации. Его можно редактировать, если это разрешено в определении модуля.

Comment: комментарий, дополнительная информация.

Address: адрес, данное поле присутствует если оно активировано в конфигурационном файле. Задайте необходимый адрес канала.

Параметры канала

Аналогично диалогу параметров модуля диалог параметров канала включает параметры: **Index, Name, Value, Default, Min., Max.** Данный диалог также может быть заменен специфическим диалогом изготовителя 'Custom Parameters'.

Битовые каналы

Битовые каналы вставляются автоматически, если в конфигурационном файле канала определено CreateBitChannels=TRUE. Диалог параметров для битовых каналов содержит единственное поле **Comment**.

6.6.7 Конфигурирование модулей Profibus

CoDeSys поддерживает конфигурацию аппаратных средств по стандарту Profibus DP. В profibus системах определены модули типа ведущий (master) и ведомый (slave). Каждый ведомый обеспечивается набором параметров и передает данные по запросу ведущего.

Система PROFIBUS DP состоит из одного или нескольких ведущих и их ведомых. Вначале модули должны быть сконфигурированы так, чтобы обмен данных по сети был возможен. При инициализации

сетевой системы каждый ведущий настраивает своих ведомых в соответствии с заданной конфигурацией. При функционировании сети ведущий посылает и/или запрашивает данные у ведомых.

Конфигурация ведущих и ведомых модулей в CoDeSys основана на gsd файлах, поставляемых изготовителем аппаратных средств. С этой целью и gsd-файлы размещаются в директориях конфигурации. Модули, описанные в gsd файле, можно добавлять в дерево конфигураций и редактировать их параметры. Далее к ведущему могут быть добавлены несколько ведомых.

Если DP ведущий выбран в дереве конфигураций, в правой части окна будут доступны следующие диалоги: Base parameters, DP Parameter, Bus parameters, Module parameters.

Если выбран DP ведомый в списке DP ведущего, то справа будут доступны следующие диалоги: (в зависимости от определений в конфигурационном файле): Base parameters, DP Parameter, Input/Output, User parameters, Groups, Module parameters. В зависимости от определений в конфигурационном файле диалог "DP Parameter" может иметь иной заголовок.

Если выбран DP ведомый, вставленный в конфигурацию на уровне ведущих, то будут доступны следующие диалоги: Base parameters, DP parameters, Input/Output, Module parameters.

Базовые параметры ведущего

Диалог базовых параметров (Base parameters) ведущего модуля DP включает параметры: **Module ID**, **Node number**, **Input**, **Output** и адреса диагностики.

Do not adapt address automatically: Данная опция доступна, только если она разрешена в конфигурационном файле. Если она включена, то модуль учитываться в случае пересчета адресов.

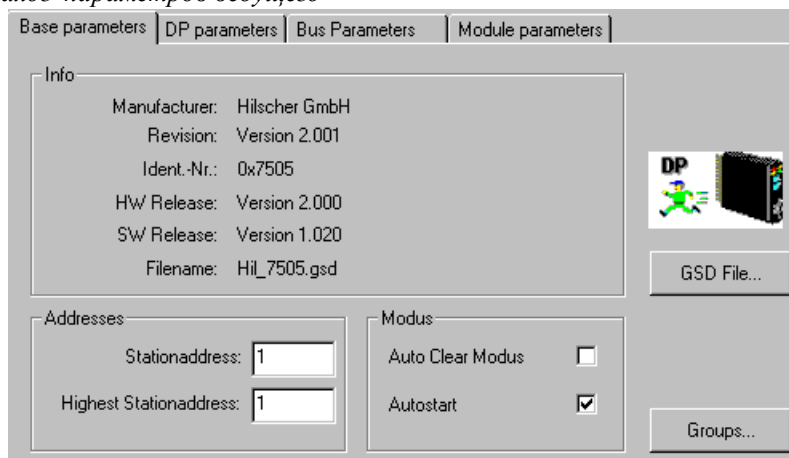
Параметры модуля ведущего

Диалог параметров модуля (Module parameters) ведущего DP включает параметры ответственные за другие модули: Здесь отображаются параметры, присвоенные ведущему в дополнение к параметрам DP, и сети, определенные в конфигурационном файле. Изменение параметров выполняется, как описано выше в разделе «Конфигурация модулей ввода/вывода».

DP параметры ведущего

Данный диалог (DP parameters) отображает параметры, определенные в файле описания устройства ведущего (Диалог может иметь другой заголовок, определенный в конфигурационном файле):

Диалог параметров ведущего



6.6.7..1.1.1 **Info Manufacturer** (изготовитель), **GSD Revision, ID** (идентификационный номер), **HW Release** и **SW Release** (версии аппаратуры и ПО), **GSD**-имя файла

Module name Имя модуля доступно для редактирования.

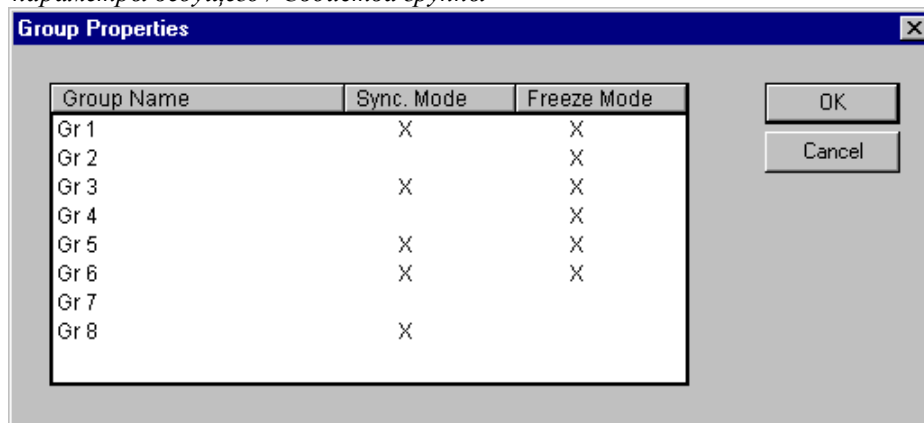
Addresses **Station address:** адрес станции от 0 до 126. Каждое новое устройство, добавленное в сеть получает очередной номер (примечание: адрес DP ведомого 126 по умолчанию). Допускается ручной ввод.
Highest station address: старший адрес станции (**HSA**), присвоенный сети. Здесь же может быть задан и нижний адрес, ограничивающий GAP диапазон (диапазон адресов, где происходит поиск новых подключенных устройств).

Для просмотра GSD файла используйте кнопку **GSD File**.

Путем помещения ведомых (см. 'Параметры ведомого DP' и 'Создание группы ведомых') в различные группы передачу данных ведущего можно синхронизировать через глобальную команду управления. С командой Freeze ведущий вынуждает ведомого или группу „заморозить" входы в их мгновенном состоянии и передавать эти данные при следующем обмене. Командой Sync ведомые принуждаются к одновременному переключению всех выходов по очередной Synch команде в соответствии с данными, полученными от ведущего после первой команды.

Для переключения опций **Freeze** и **Sync** для группы используйте контекстное меню (левая клавиша мыши).

DP параметры ведущего / Свойства группы



Сетевые параметры ведущего

Сетевые параметры (Bus parameters) описывают параметры таймаутов коммуникации. Если включена опция **Optimize**, то значения параметров будут вычисляться автоматически, в зависимости от скорости **Baudrate**, заданной пользователем, и параметров, установленных в GSD файлах.

Внимание: Автоматически рассчитываемые величины имеют приближенное значение!

Сетевые параметры ведущего DP

Parameters	Value	Unity
Slot Time (TSL)	0	tBit
Min.Station Delay (min TSDR)	0	tBit
Max.Station Delay (max TSDR)	0	tBit
Quiet Time (TQUI)	0	tBit
Setup Time (TSET)	0	tBit
Target Rotation Time (TTR)	0	tBit
Gap Update Factor	0	
Max. Retry Limit	0	
Min. Slave Interval	0	100 μ s
Poll Timeout	0	10 ms
Data Control Time	0	ms
Watchdog Time (TWD)	0	ms

Все параметры также можно задавать вручную.

- Baud rate** Допускается выбор значений из заданных в GSD файле. Задавать необходимо только скорость передачи, обеспечиваемую всеми ведомыми.
- Optimize** Если данная опция включена, значения параметров будут вычисляться автоматически. Редактирование параметров доступно, только если опция выключена.
- Slot Time** Максимальное время ожидания ведущим ответа на передачу запроса ведомому.
- Min.Station Delay** min. TSDR (in tbit): минимальное время реакции, после которого станция в сети может отвечать (min. 11 tBit)
- Max.Station Delay** max. TSDR (in tbit): максимальный интервал, за который ведомый должен отвечать.
- Quiet Time** TQUI (in tbit): период простоя (idle), необходимый для переключения передатчика.
- Target Rotation Time** TTR (in tbit): время цикла; интервал, в течение которого ведущий должен получить маркер. Получается в результате суммирования времен владения маркером всех ведущих сети.
- Gap Update Factor** Фактор обновления GAP: число циклов сети, через которое ведущие проводят поиск вновь включенных станций (в диапазоне адресов от своего собственного до адреса следующей станции).
- Max. Retry Limit** Максимальное число запросов, посылаемых ведущим, при отсутствии ответа ведомого.
- Min. Slave Interval** Пауза между двумя циклами сети, которую ведомый может использовать для обработки полученного запроса. Заданное здесь значение должно соответствовать заданным в GSD файле спецификациям.
- Poll Timeout** Максимальное время ответа ведущего на запрос другого ведущего (Class 2 DP master) (дискрета 1 ms).
- Data Control Time** Время сбора ведущего. Ведущий определяет наличие хотя бы одного ответа от каж-

дого своего ведомого за это время и обновляет Data_Transfer_List.

Watchdog Time Значение времени мониторинга обращения (watchdog). Изменение значения поля не поддерживается (фиксированное значение 400 ms).

Базовые параметры ведомого DP

Диалог базовых параметров ведомого (DP-Slaves) аналогичен диалогам любых типов модулей: **Module id**, **Node number**, **Input-**, **Output-** и **Diagnostic address**.

Do not adapt address automatically: Данная опция доступна, только если она разрешена в конфигурационном файле. Если она включена, то модуль учитываться в случае пересчета адресов.

Диалог базовых параметров ведомого DP

DP параметры ведомого

Данный диалог отображает параметры, определенные в файле описания устройства ведомого (Диалог может иметь другой заголовок, определенный в конфигурационном файле):

DP параметры ведомого

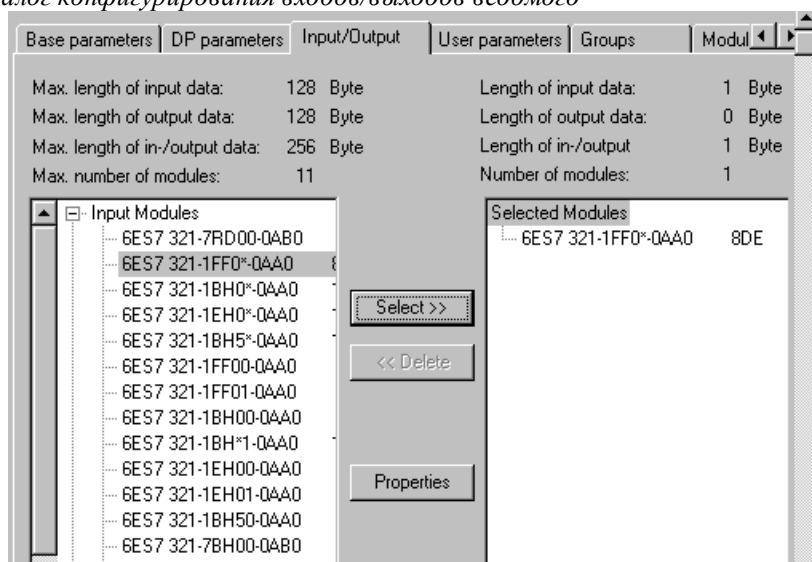
Info **Manufacturer** (изготовитель), **GSD Revision**, **ID** (идентификационный номер), **HW Release** и **SW Release** (версии аппаратуры и ПО), **GSD-имя** файла, тип ведомого

- Standard parameter** **Identnumber:** Уникальный идентификатор, присвоенный PNO данному типу устройств. Обеспечивает однозначное соответствие DP ведомого и GSD файла.
- TSDR (Tbit*):** Time Station Delay Responder: Время реакции, минимальное время, после которого ведомый может отвечать ведущему. (min. 11 TBit)
- * TBit: Время передачи бита PROFIBUS; Определяется скоростью передачи; например 1 TBit на 12MBaud=1/12.000.000 бит/сек = 83нс
- Lock/Unlock:** ведомый блокируется или разблокируется ведущим:
 0: min.TSDR и специфические параметры ведомого могут перезаписываться
 1: Ведомый разблокирован для других ведущих,
 2: Ведомый заблокирован для других ведущих, все параметры приняты;
 3: Ведомый разблокирован для других ведущих
- Identification** **Station address** (см 'Параметры ведущего DP'), **Station name** - имя станции, доступно для редактирования.
- Activation** Активация: Ведомый разрешен или запрещен в данной конфигурации. Если активация снята, то конфигурационные данные передаются коммутатору, но обмен в сети не производится.
- Watchdog** Если **Watchdog Control** активен, то доступна и настройка его работы (мониторинг обращения, дискрета 10 ms). Если ведомый не опрашивается ведущим заданное время, он будет сброшен в начальное состояние.

Для просмотра GSD файла используйте кнопку **GSD File**.

Входы/выходы ведомого DP

Диалог конфигурирования входов/выходов ведомого



Способ конфигурирования ведомого DP определяется наличием у него модульной либо фиксированной структуры.

Выберите в левом окне диалога необходимый модуль ввода-вывода и нажмите кнопку **Select**, он будет помещен в правое окно. Ошибочно добавленные модули удаляются кнопкой **Delete**. Добавленные модули немедленно отображаются в дереве конфигурации. Если затем выбрать такой модуль, то для него показывается соответствующий диалог **Profibus Modul**, описывающий входы, выходы и диагностические адреса. Если вы выберете отдельный канал модуля, то для него будет доступен диалог **Profibus Channel**, описывающий адрес канала. Оба диалога могут иметь специальные заголовки, описанные в конфигурационном файле.

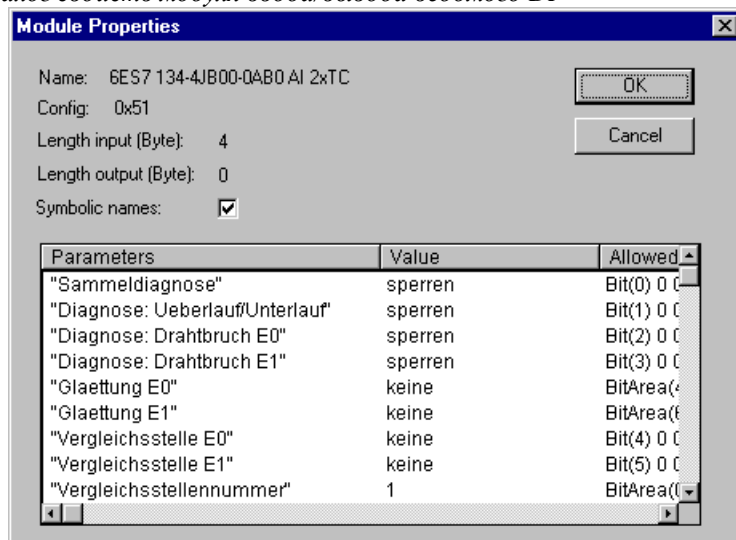
Максимальные размеры данных (**Max. length of input data**, **Max. length of output data**, **Max. length of in-/output data**) и максимальное число модулей (**Max. number of modules**) определяются в GSD-файле. Данная информация отображается для обоих списков модулей. Левый блок отображает максимальные значения для устройства, правый блок отображает суммарные по всей текущей конфигурации показатели. При достижении максимумов будет дано сообщение об ошибке.

Список в левом окне содержит все доступные модули (описанные в GSD файлах), а правое окно дает текущий список установленных для данного устройства модулей ввода-вывода.

Для модульных устройств аналогичным образом можно добавлять модули ввода-вывода к ведомому и удалять их кнопкой **Delete**.

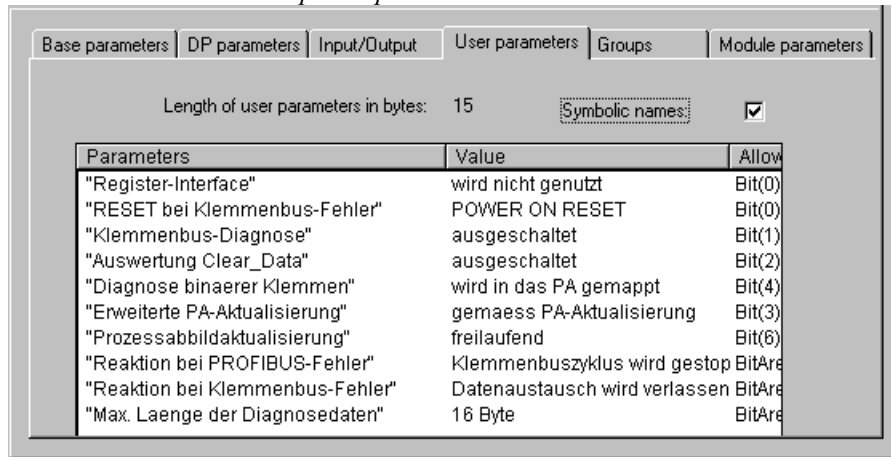
Кнопка **Properties** открывает диалог свойств модуля ввода-вывода 'Module properties', выбранного в правом или левом окне списков. Он включает имя **Name**, **Config** (описание модуля по стандарту PROFIBUS) размер входов и выходов в байтах. Если описание модуля в GSD файле имеет специфические параметры, то они отображаются со своими значениями и диапазонами значений. Если опция **Symbolic names** активна, то используются символьные имена.

Диалог свойств модуля ввода/вывода ведомого DP



Пользовательские параметры ведомого DP

Диалог пользовательских параметров ведомого DP

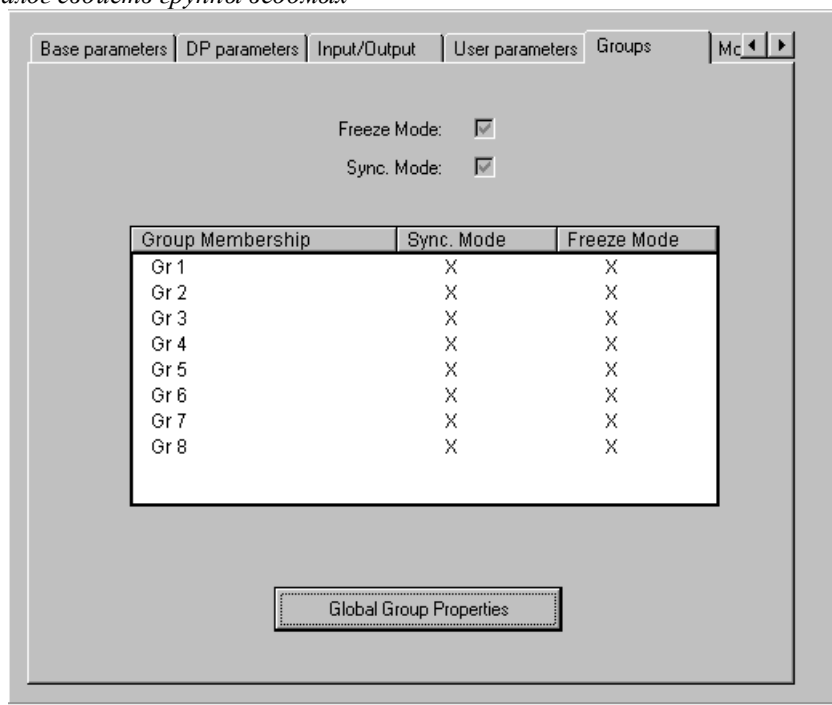


Здесь приведены некоторые дополнительные параметры ведомого, определенные в GSD-файле. Колонка **Parameters** содержит наименование параметра. Значение параметра задается в столбце **Value**. Для его редактирования щелкните дважды по значению мышкой или используйте правую клавишу мыши. Допустимые значения могут быть ограничены **Value range**.

В GSD-файле применяются символьные имена значений параметров. Если включена опция **Symbolic names**, то значения отображаются в виде символьных имен. Для информации над таблицей указан размер пользовательских параметров **Length of user parameters**.

Создание группы ведомых

Диалог свойств группы ведомых



Диалог используется для назначения ведомых в одну или несколько из восьми возможных групп. С другой стороны, общие свойства группы (Sync. и-или Freeze) определяются в конфигурации свойств

ведущего (см. 'Параметры ведущего DP'). Этот же диалог доступен через кнопку **Global Group Properties**.

Группа (группы), в которую включен ведомый, обозначается значком "плюс". Для добавления / удаления ведомого выберите имя группы в колонке **Group Membership** и дайте команду 'Add slave to group' (добавить) / 'Remove slave from group' (удалить) из контекстного меню (правая клавиша мыши).

Ведомое устройство может быть добавлено только в группу, свойства которой оно поддерживает. A slave device can only be assigned to those groups whose properties it supports. Свойства каждого ведомого отображаются над таблицей (**Sync. Mode / Freeze Mode**). В них отмечены свойства, поддерживаемые модулем.

Параметры модуля ведомого DP

Параметры модуля ведомого DP аналогичны параметрам других модулей (см. раздел 6.6.5). Значения параметров редактируются обычным способом.

Параметры ведомых DP в подчиненных сетях Profibus

Если Profibus работает в подчиненном режиме, то ведомые устройства вставляются в дерево конфигурации на уровне ведущих. Конфигурация настраивается в (описанных выше) диалогах: Базовых параметров, DP параметров, параметров модуля и ввода/вывода.

6.6.8 Конфигурирование CANopen-модулей

CoDeSys поддерживает настройку оборудования в соответствии с *CANOpen Draft Standart 301*. Конфигурирование контроллера производится практически так же, как описано выше для обычного аппаратно-зависимого конфигурирования ПЛК.

Все EDS (Electronic Data Sheet) и DCF (Device Configuration File) файлы, которые находятся в директории конфигурационных файлов (обычно PLCCONF), интегрируются в CoDeSys, и их содержание можно просмотреть и изменить с помощью PLC Configuration. В EDS описываются параметры CAN-модуля, которые можно настраивать. Если вы добавите модуль, описанный с помощью DCF, можно будет настраивать только МЭК-адреса, а все остальные параметры модуля будут зафиксированы.

Базовые параметры CAN-мастера

Настройка таких параметров, как **Modul-Id**, **input-/output addresses**, **Diagnostic address**, описана в разделе 6.6.5.

CAN-параметры CAN-мастера

Параметры сети CAN можно настраивать сразу после добавления модуля либо после вызова команды "Extras" "Properties".

Выберите скорость передачи данных из списка **Baud rate**.

PDO (Process Data Object) может передаваться в синхронном и асинхронном режимах. Синхронизирующее сообщение, имеющее уникальный идентификатор **Sync. COB-ID** (Communication Object Identifier), передается с периодом, указанным в поле **Communication Cycle Period**. Для передачи PDO выделяется временной интервал **Sync. Window Length** сразу после передачи синхронизирующего сообщения. Величина этого интервала указывается в микросекундах. Синхронизирующее сообщение не посылается, если **Communication Cycle Period** и **Sync. Window Length** равны 0.

Флажок **activate**: разрешает передачу синхронизирующего сообщения.

Node-Id: идентификатор CAN-устройства (узла). Принимает значения от 1 до 127. Это значение должно быть уникальным для каждого устройства в сети и задается в десятичном виде. (Не путайте поля **Node-Id** и **Node number**.)

Сеть CAN будет автоматически инициализироваться и включаться в работу сразу после загрузки программы, если активна опция **Automatic start**. Если эта опция не активна, то сеть нужно запускать непосредственно из программы.

Если активна опция **Support DSP301, V3.01 and DSP306**, то будут поддерживаться модульные CAN-Slave устройства, а также еще некоторые возможности, описанные в стандарте DSP301, V3.01 и DSP306, в том числе и Heartbeat. В этом случае CAN устройство будет передавать специальные сообщения, сообщающие о том, что оно работает, с периодом, указанным в поле **Heartbeat Master[ms]**. Такой механизм называется Heartbeat (сердцебиение) и является альтернативой механизма Nodeguarding, но отличается тем, что может работать как на Master, так и на Slave-устройствах. Обычно этот механизм запускается на CAN-мастере.

Параметры модуля CAN-мастера

Диалог настройки параметров модуля CAN мастера такой же, как и для других модулей (см. 6.6.5). Дополнительные параметры CAN-мастера, описанные в конфигурационном файле, также доступны пользователю, и их можно редактировать.

Базовые CAN параметры

Информация по настройке таких параметров, как **Modul id**, **Input/Output addresses**, **Diagnostic address**, находится в главе 6.6.5.

МЭК-адреса, по которым можно обращаться к PDO из проекта, вводятся в полях **output address** и **input address** в зависимости от направления передачи данных, которая осуществляется PDO.

В поле **diagnostic address** нужно ввести МЭК-адрес маркированной памяти (M). По этому адресу будет размещена диагностическая информация о модуле.

CAN-параметры CAN-модуля

Диалог CAN параметров CAN-модуля

Раздел General

Node-Id: идентификатор CAN-устройства (узла). Принимает значения от 1 до 127. Это значение должно быть уникальным для каждого устройства в сети и задается в десятичном виде.

Если активна опция **Write DCF**, то при компиляции проекта создается файл с расширением DCF, имя которого состоит из имени соответствующего EDS файла и идентификатора узла, для которого создавался этот файл.

Если активна опция **Create All SDO's**, то SDO создаются для всех объектов, а не только для тех, которые изменены.

Если активна опция **Reset node**, то ведомое устройство сбрасывается перед загрузкой конфигурации.

Опция **Optional device** (ее наличие зависит от специфики целевой платформы) приводит к тому, что мастер будет выполнять только одну попытку чтения из данного узла. Отсутствие ответа игнорируется, то есть мастер продолжит нормальное функционирование.

Опция **No initialization** указывает мастеру немедленно активировать данный узел без отправки конфигурационного SDO. (Данные SDO будут созданы и сохранены в контроллере в любом случае.)

Раздел Node guard (альтернатива механизму Heartbeat):

Если активна опция **NodeGuarding**, то модулю посылается сообщение с периодом **Guard Time** (указывается в миллисекундах). В ответ модуль должен послать сообщение с идентификатором **Guard COB-ID** (Communication Object Identifier). Если этого не происходит, то он получает статус “timeout”. Если модуль не отвечает на **Life Time Factor** сообщений, то он получает статус “not OK”. Статус модуля можно определить, обратившись по диагностическому адресу. Контроль состояния модуля не производится, если переменные Guard Time и Life Time Factor равны 0.

Раздел Heartbeat Settings (альтернатива механизму Nodeguarding):

Если опция **Active Heartbeat Producer** активна, то модуль посылает специальные heartbeat-сообщения с периодом **Heartbeat Producer Time** (указывается в миллисекундах).

Если активна опция **Active Heartbeat Consumer**, то модуль слушает heartbeat-сообщения, посылаемые мастером. Если эти сообщения до него не доходят, то модуль выключает свои входы/выходы.

Раздел Emergency Telegram:

Модуль передает аварийное сообщение с уникальным идентификатором **COB-Id**, когда происходит внутренняя ошибка. Это сообщение, различное для разных модулей, хранится по диагностическому адресу.

Нажав кнопку **Info**, вы можете просмотреть содержание EDS или DCF файлов. Информация разбита на 3 раздела: “FILE INFO”, “DEVICE INFO” и “PDO INFO”.

Выбор CAN-модулей модульных ведомых устройств

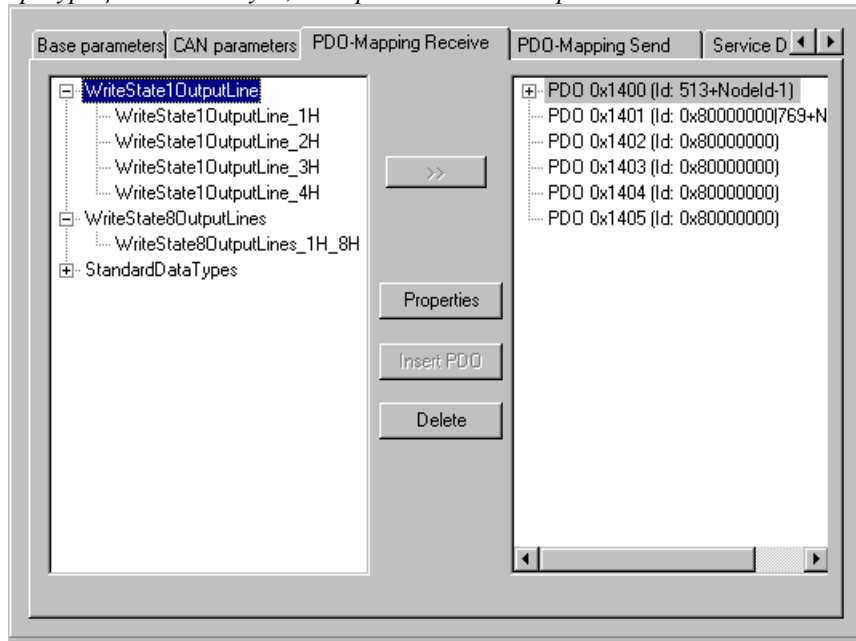
В левой колонке (**Available modules**) находятся все доступные модули. Выберите нужные вам модули и с помощью кнопки **Add** добавьте их в правую колонку (**Selected Modules**). С помощью кнопки **Remove** модуль можно будет удалить. Настройки PDO и SDO изменяются автоматически.

Отображение PDO в CAN-модулях

Вкладки **Receive PDO mapping** и **Send PDO mapping** в диалоге конфигурирования модуля позволяют изменить образ PDO (PDO mapping), описанный в EDS-файле.

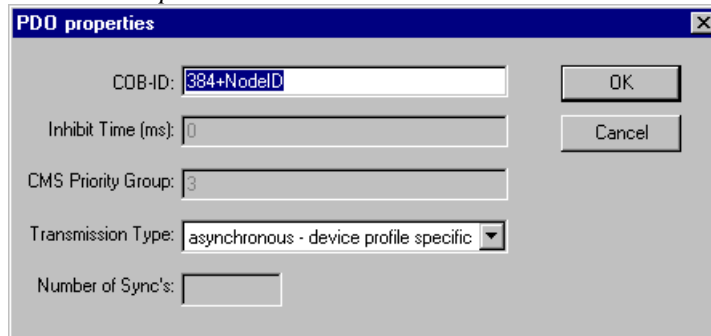
Все доступные объекты располагаются в левой части окна и могут быть отображены в PDO (Process Data Object) с помощью кнопки “>>” и удалены из PDO кнопкой **Remove**. Объекты типа Standard-DataTypes могут быть использованы для заполнения пустых промежутков в PDO.

Конфигурация CAN-модуля, настройка PDO-отображения



Кнопка **Insert PDO** используется для создания дополнительного PDO. В новое PDO вы можете отобразить дополнительные объекты. Распределение памяти входов/выходов для этих объектов производится автоматически, и его можно увидеть в конфигурации контроллера. Кроме того, в конфигурации контроллера появляются символьные имена добавленных объектов. Настройки PDO можно изменить, нажав кнопку **Properties**.

Диалог PDO Properties



Каждое PDO имеет собственный идентификатор **COB-Id** (Communication Object Identifier).

Настройки PDO, недоступные для данного модуля, неактивны.

Inhibit Time – это минимальное время между двумя посылками данного PDO. Нужно выбрать это значение так, чтобы PDO не посылалось слишком часто. Это происходит в том случае, когда значения параметров, отображенных в PDO, передаются чаще, чем меняются их значения, что приводит к необоснованному увеличению загрузки сети.

CMS Priority Group - приоритет PDO при его передаче по сети. Может принимать значения от 0 до 7, причем наивысшему приоритету соответствует значение 0.

Transmission Type-режим передачи PDO. Доступны следующие режимы:

Ў **acyclic-synchronous**: PDO передается синхронно, но не периодически

- **acyclic-synchronous**: PDO передается синхронно, через каждые **Number of Sync** синхронизирующих сообщений.
- **synchronous-RTR only**: PDO обновляется после каждого синхронного сообщения, но передается только после специального запроса (Remote Transmission Request)
- **asynchronous-RTR only**: PDO обновляется и передается только после специального запроса (Remote Transmission Request)
- **asynchronous-device profile specific** и **asynchronous-manufacturer specific**: PDO передается после специального события.
- **Number of Sync**: число синхронизирующих сообщения между передаваемыми PDO в режиме синхронной передачи.

Event-Time: период между двумя сообщениями для соответствующего режима передачи.

Service Data Object (SDO)

На вкладке Service Data Object вы найдете список всех объектов, определенных в EDS и DCF файлах, с индексами от 0x2000 до 0x9FFF. Эти объекты доступны для записи.

Диалог настройки SDO

Index	Name	Value	Type
2100sub1	8bit Output Block No. 1		Unsigned
2100sub2	8bit Output Block No. 2		Unsigned
2100sub3	8bit Output Block No. 3		Unsigned
2100sub4	8bit Output Block No. 4		Unsigned
2100sub5	8bit Output Block No. 5		Unsigned
2100sub6	8bit Output Block No. 6		Unsigned
2100sub7	8bit Output Block No. 7		Unsigned
2100sub8	8bit Output Block No. 8		Unsigned
2100sub9	8bit Output Block No. 9		Unsigned

Каждый объект имеет свойства **Index**, **Name**, **Value** и **Default**. Значения этих свойств можно изменять. Выделите нужное вам значение и нажмите <Пробел>. После этого внесите необходимые изменения и для подтверждения нажмите <Enter>, а для отмены - <Esc>. Эти значения передаются в виде SDO (Service Data Object) при инициализации сети.

Замечание: Все типы данных, поддерживаемые CANopen, но не поддерживаемые МЭК-61131-3, заменяются на типы данных МЭК с более широким диапазоном представления.

6.6.9 Конфигурирование ведомого CANopen-устройства (CANopen Slave)

ПЛК, программируемый с помощью CoDeSys, можно использовать как CANopen Slave-устройство в сети. В дальнейшем будем называть CANopen Slave как CanDevice.

Для этого необходимо определить ПЛК конфигурацию и сохранить ее в EDS-файле. Такой EDS-файл можно в дальнейшем использовать при определении конфигурации CANopen мастера.

Требования для создания CanDevice:

1. Библиотеки
 - a. 3S_CanDrv.lib

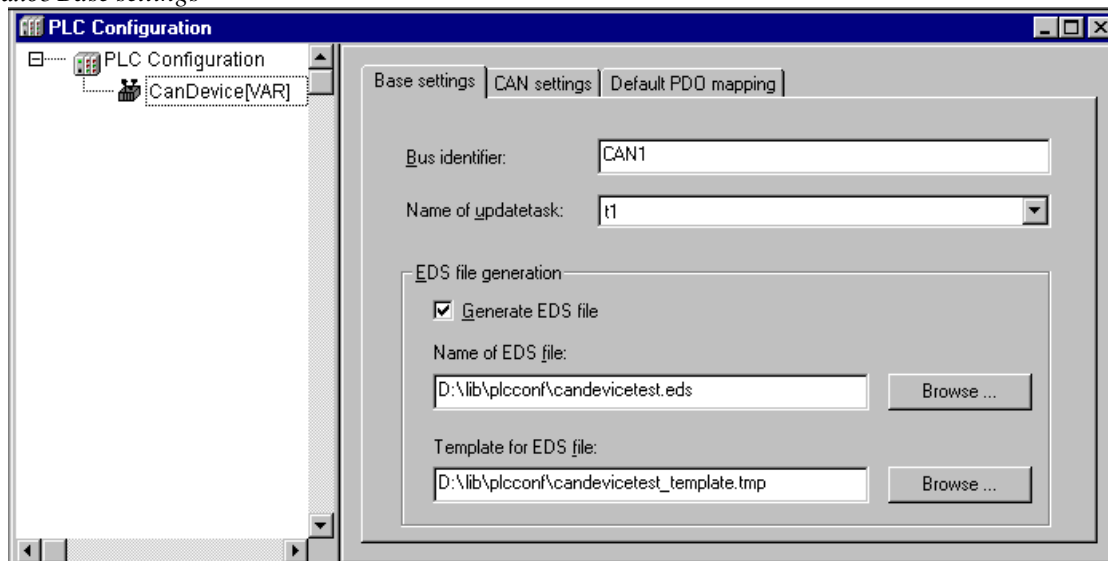
- b. 3S_CanOpenManeger.lib
- c. 3S_CanOpenDevice.lib

должны быть включены в проект. Они необходимы для того, чтобы ПЛК мог работать как устройство CAN.

2. В конфигурационном файле с расширением *.cfg, который описывает конфигурацию ПЛК, должны быть сделаны соответствующие настройки. Только в этом случае в PLC Configuration появится дополнительный подэлемент “CanDevice”. Этот объект настраивается в дополнительном диалоге с 3 вкладками: **Base settings**, **CAN settings**, **Default PDO mapping**.

Базовые настройки CanDevice

Диалог *Base settings*



Bus identifier: пока не используется

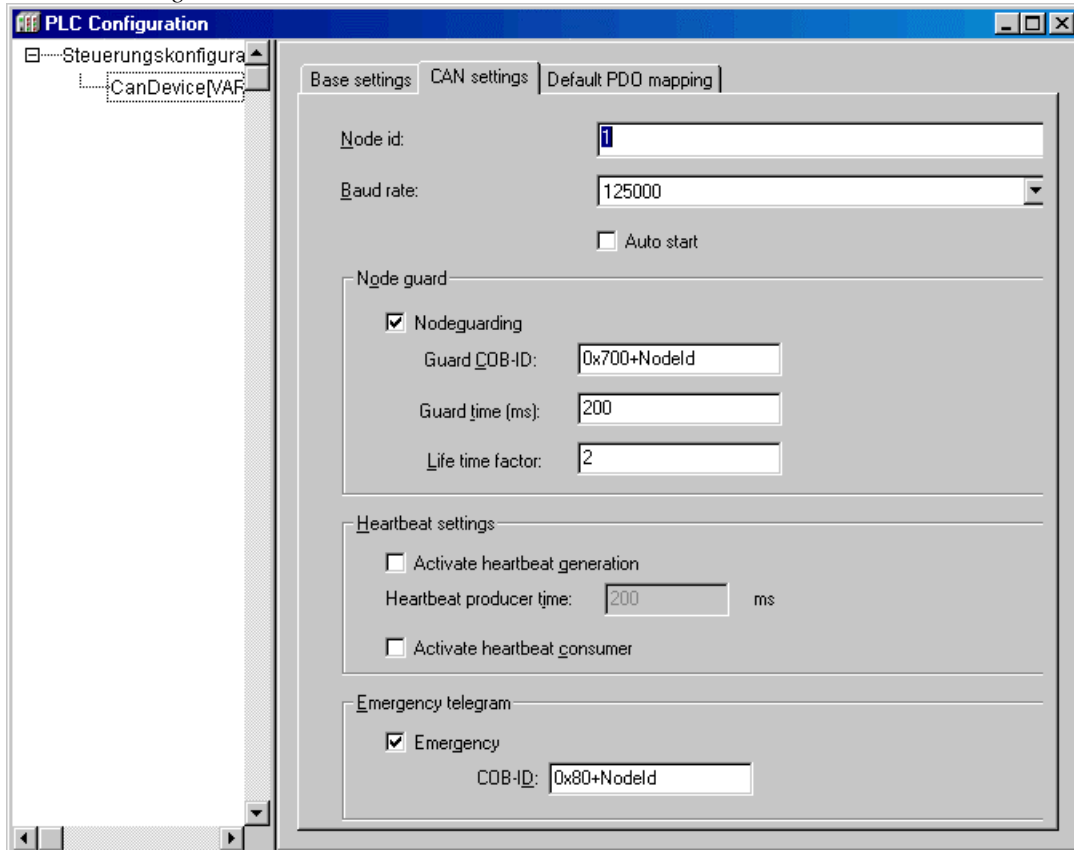
Name of updatetask: название задачи, из которой будет вызываться CanDevice. В выпадающем списке вы можете выбрать необходимую задачу.

EDS file generation: включите эту опцию, если хотите чтобы текущие настройки были сохранены в EDS-файле, который потом можно использовать для настройки любого CAN-устройства. В поле **Name of EDS file** введите имя и путь к файлу. Вы можете вручную создать шаблон для EDS и указать к нему путь в поле **Template of EDS file**. Например, создайте текстовый файл, содержащий элементы EDS-файла, сохраните его как EDS_template.txt и укажите к нему путь в поле **Template of EDS file**. Теперь, если вы создаете EDS-файл “device_xy.eds” для текущего проекта, то настройки, сделанные в проекте, будут объединены с настройками из шаблона и сохранены в файле “device_xy.eds” (расширение шаблона не должно быть “.eds”). Если настройки текущего проекта уже определены в шаблоне, то они не будут перезаписаны.

Для указания пути к файлам используйте стандартный диалог, который открывается при нажатии кнопки **Browse...**

CAN-параметры CanDevice

Диалог CAN settings

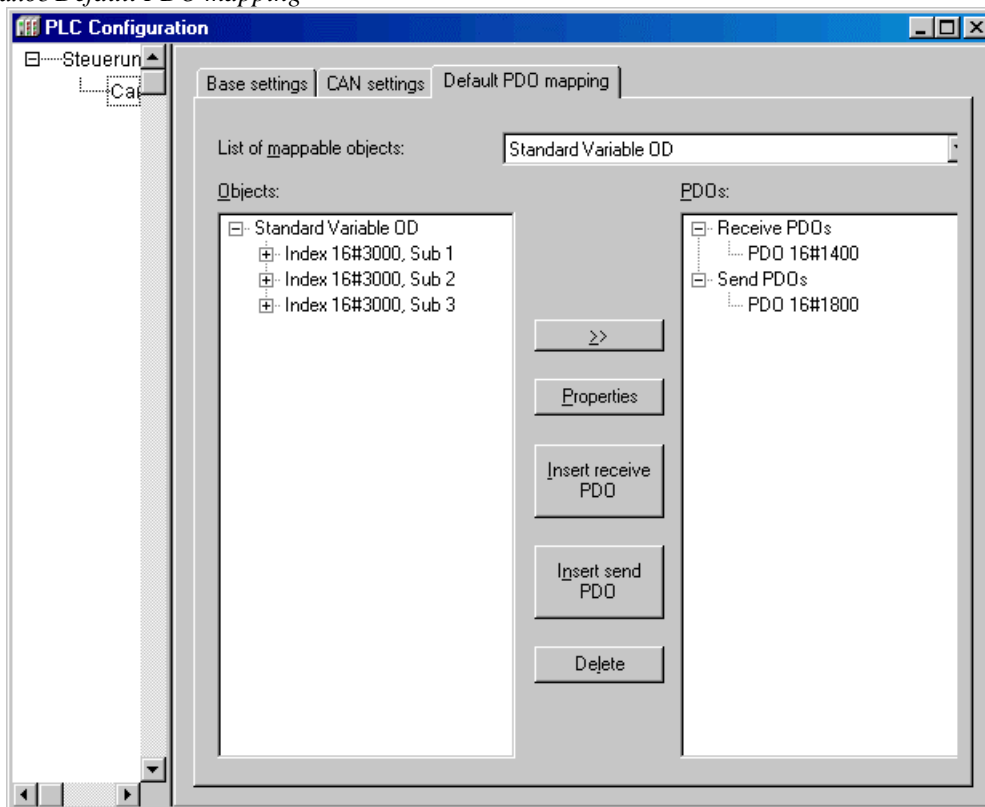


В этом диалоге вы можете определить **Node id** и **Baud rate**. **Node id** – это идентификатор узла, который используется для адресации к устройству по сети CANopen.

Так же можно настроить **Nodeguarding** и **Emergency Telegram** (описание этих механизмов приведено выше). Heartbeat не поддерживается.

Стандартное PDO отображение для CanDevice

Диалог *Default PDO mapping*



В этом диалоге элементы локального Менеджера параметров могут быть сопоставлены с PDO, которые отправляются и принимаются этим CanDevice. Полученное PDO-отображение будет доступно в любом конфигураторе, в который CanDevice интегрируется.

Параметры, описанные в Менеджере параметров, присоединяются к переменным проекта с помощью системы индексов/подиндексов.

Обратите внимание: Подиндекс 0 индекса, который имеет более одного подиндекса, используется для хранения числа подиндексов. Поэтому не используйте подиндекс 0 в менеджере параметров. Не забудьте также, что подиндексы для каждого индекса вводятся в порядке возрастания.

List of mappable objects: Здесь вы можете выбрать список параметров, элементы которого будут отображаться в PDO данного CanDevice. В зависимости от целевой платформы, можно создать список параметров типа “Mapping”, который специально предназначен для отображения в PDO CanDevice. В этом случае для настройки отображения PDO будет доступен только этот список параметров. В противном случае будут доступны все списки параметров типа “Variables” и “Instance”.

Элементы выбранного списка появляются в списке **Objects**. В списке **PDO’s** настраивается конфигурация PDO. Добавить в список принимаемые и передаваемые PDO можно с помощью кнопок **Insert receive PDO** и **Insert send PDO**. Для того чтобы отобразить объект в PDO, выберите объект в левом окне, PDO в правом и нажмите кнопку >>. Параметры PDO настраиваются с помощью диалога, который появляется при нажатии кнопки **Properties**.

С помощью кнопки **Delete** выбранное PDO можно удалить из списка.

Пример:

Цель: В первое принимаемое PDO(COB-Id = 512+NodeId) нужно отобразить переменную PLC_PRG.a

Для этого в менеджере параметров нужно создать параметр с определенным индексом/подиндексом и связать его с переменной PLC_PRG.a. Менеджер параметров подключается на вкладке “Network functionality” в настройках целевой платформы. Там же определяются диапазоны индексов и подиндексов.

Теперь в диалоге “Default PDO-Mapping” можно отобразить этот параметр в принимаемое PDO.

6.6.10 Конфигурирование модулей DeviceNet

CoDeSys поддерживает конфигурирование аппаратуры для распределенных систем, основанных на международном стандарте DeviceNet (EN50325). Главным образом DeviceNet используется в промышленных сетях, обеспечивающих свойства Plug & Play для подключения датчиков и исполнительных устройств (электронные переключатели, заслонки и др.).

Протокол DeviceNet базируется на CAN (Controller Area Network). Обмен данными построен на прямом соединении коммуникационных модулей.

Редактор конфигурации DeviceNet в CoDeSys обеспечивает настройку DeviceNet-мастера, управляющего обменом данными в сети. Поддерживаются различные типы коммуникации для обмена входными и выходными данными между ведомыми модулями (DeviceNet-Slave) в сети. Обычно DeviceNet-мастер выполняет функцию "UCMM" (Unconnected Message Manager для множественных соединений) и обеспечивает запросы от других мастеров к своим ведомым.

Для вставки DeviceNet модуля в CoDeSys ПЛК конфигурацию необходимо иметь соответствующий конфигурационный файл.

Можно использовать все EDS (Electronic Data Sheet) файлы, присутствующие в директории конфигурации и содержащие описание DeviceNet модулей. Описания CAN устройств также используют файлы с расширением ".EDS", но они не применимы в DeviceNet конфигурации!

При выборе DeviceNet-мастера в дереве конфигурации активируется диалог настройки, который содержит вкладки: Базовые параметры, Параметры DeviceNet, Параметры модуля.

Для ведомых (DeviceNet-Slave), которые добавлены к мастеру, доступны диалоги: Базовые параметры, Параметры DeviceNet, Конфигурация входов-выходов, Параметры модуля.

Базовые параметры DeviceNet-Master

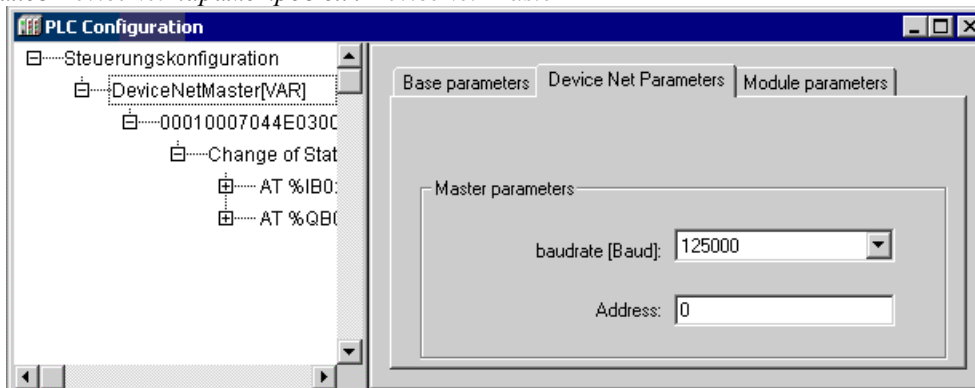
Диалог базовых параметров DeviceNet мастера включает **Module id**, **Node number**, **Input address**, **Output address** и **Diagnosis address**. Назначение данных параметров аналогично для всех типов модулей (см. раздел 6.6.5, Базовые параметры модуля ввода-вывода).

Параметры сети DeviceNet для DeviceNet-Master

Поле **Address** содержит идентификационный номер модуля в сети. По своему смыслу данный ID соответствует "Node-ID" для CAN модуля. (Не путайте его с Node number или адресом модуля в диалоге базовых параметров!) Адрес вводится в десятичном формате, допустимые значения: 0-63, по умолчанию: 0.

В поле **Baudrate [Baud]** задается скорость обмена по сети. Выберите один из вариантов: 125000 (по умолчанию), 250000, 500000.

Диалог DeviceNet параметров для DeviceNet-Master



Параметры модуля DeviceNet-Master

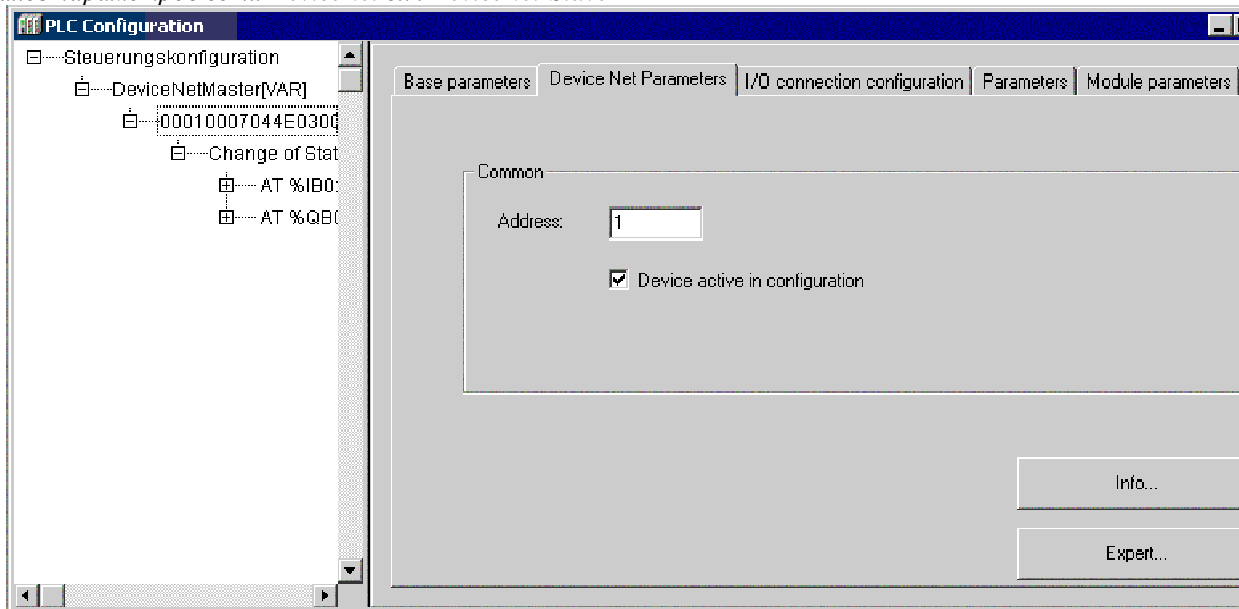
Данный диалог параметров модуля идентичен диалогу параметров любого модуля (см. раздел 6.6.5. Базовые параметры модуля ввода-вывода). Здесь же будут показаны дополнительные параметры, заданные в конфигурационном файле.

Базовые параметры DeviceNet-Slave

Диалог базовых параметров DeviceNet-Slave включает определение диапазонов адресов **Input address** и **Output address**. Задание данных адресов аналогично для всех типов модулей (см. раздел 6.6.5. Базовые параметры модуля ввода-вывода). Направление (вход или выход) определяется с точки зрения модуля.

Параметры сети DeviceNet для DeviceNet-Slave

Диалог параметров сети DeviceNet для DeviceNet-Slave



Здесь задаются общие параметры ведомого сетевого модуля:

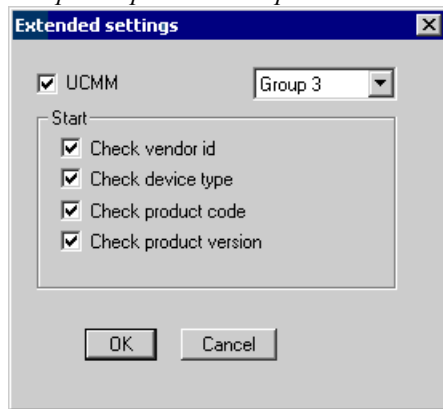
Address: сетевой идентификатор DeviceNet-Slave модуля. По своему смыслу данный ID соответствует "Node-ID" для CAN модуля. (Не путайте его с Node number или адресом модуля в диалоге базовых параметров!) Адрес вводится в десятичном формате, допустимые значения: 0-63, по умолчанию: 0.

Device active in configuration: активируйте данную опцию для того, чтобы сделать данный модуль доступным для обмена данными в сети.

Info...: данная кнопка открывает окно, отображающее содержимое EDS файла. Обратите внимание, что описания CAN устройств также используют файлы с расширением ".EDS", но они не применимы в DeviceNet конфигурации!

Expert...: данная кнопка открывает диалог расширенных настроек **Extended settings**. В нем задаются следующие параметры:

Диалог расширенных настроек



UCMM: (Unconnected Message Manager для множественных соединений). Если данная опция активна (по умолчанию), то ведомый будет способен поддерживать UCMM сообщения. Доступен выбор: **Group1**, **Group2** или **Group3** (по умолчанию).

По умолчанию при старте сети выполняется несколько проверок. В разделе **Start** некоторые из них можно запретить. При проверке всегда сравниваются значения, заданные в EDS файле со значениями в заданном устройстве: **Check vendor id**, **Check device type**, **Check product code**, **Check product version**.

Конфигурация входов-выходов DeviceNet-Slave

Здесь задается конфигурация входов и выходов ведомого, для которых необходим обмен данными в сети (значения параметров). Должен быть определен тип соединения и выбраны входы и выходы, объединенные в соответствии с возможностями модуля.

Selected I/O connection: выберете один из следующих типов соединения, приемлемый для нижеописанных соединений входов-выходов:

Poll: данные опрашиваются циклически (Master-Slave-обработка)

Bit Strobe: мастер DeviceNet передает широковещательные телеграммы всем ведомым, запрашивая передачу текущих данных. Ведомые отвечают один за другим, начиная с первого узла.

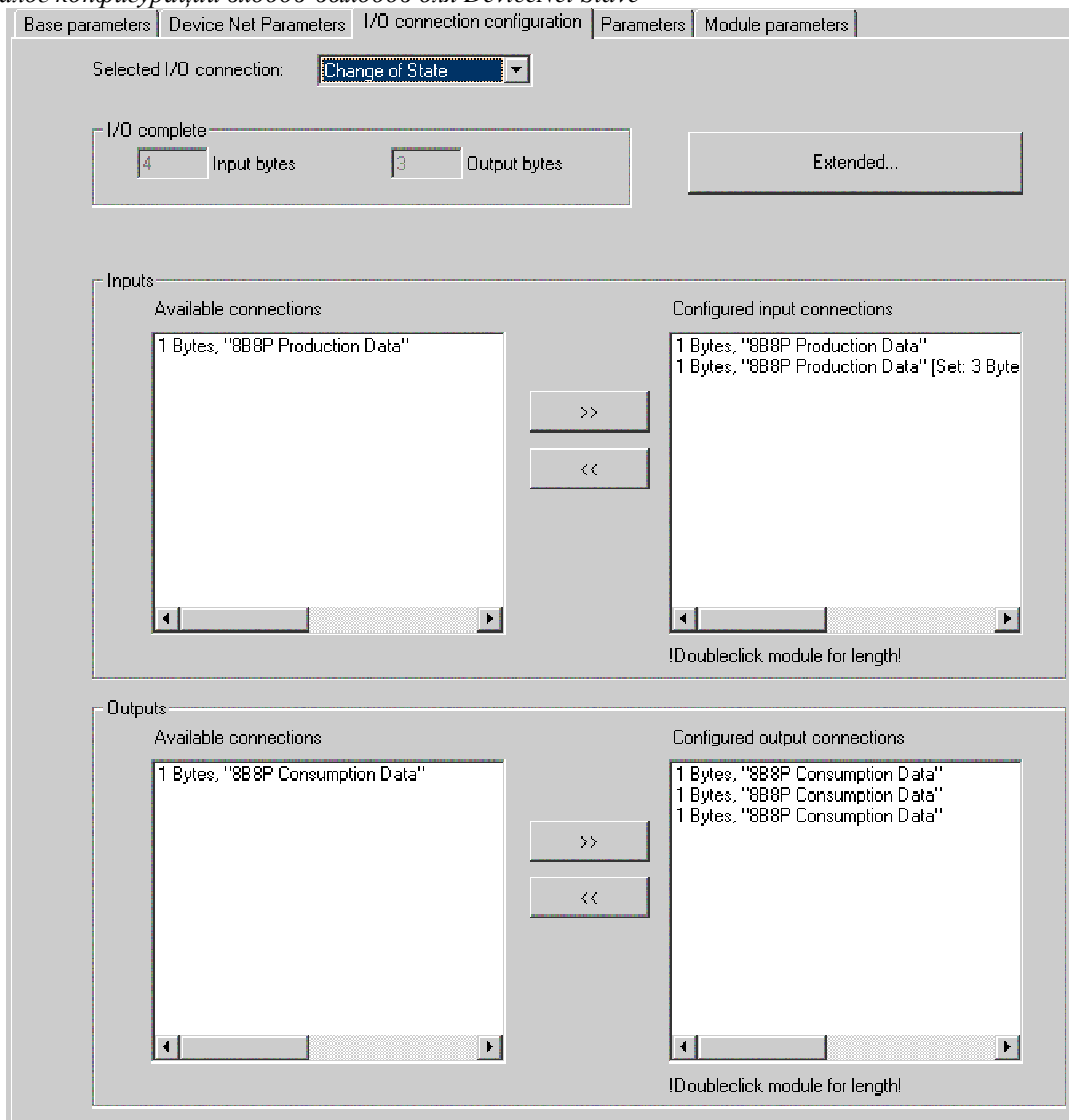
Change of State: ведомый пересылает данные ведомому при любом изменении значений на входах. Явные запросы от ведущего не требуются.

Cyclic: ведомый передает данные через заданные интервалы времени, без запросов со стороны ведущего (функция «сердцебиение»).

Multicast Poll: в настоящее время не поддерживается.

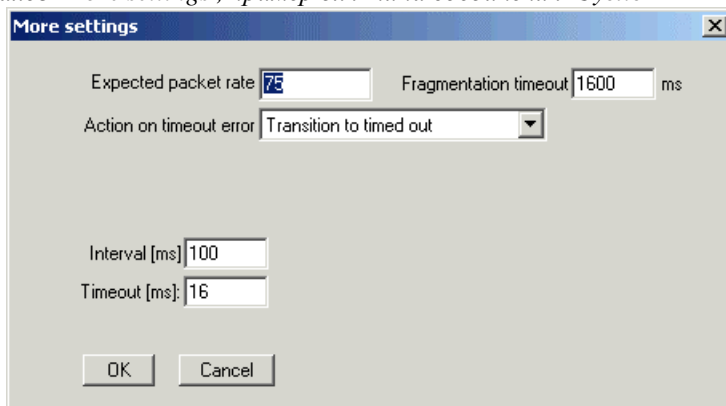
I/O complete: здесь отображается суммарный размер входов **Inputbytes** и выходов **Outputbytes**. Сумма вычисляется на основе размеров областей входов 'Inputs' и выходов 'Output'.

Диалог конфигурации входов-выходов для DeviceNet Slave



Extended: Данная кнопка открывает диалог дополнительных установок **More settings**, позволяющий изменить настройки по умолчанию для выбранного типа соединения:

Диалог 'More settings', пример для типа соединения 'Cyclic'



Expected Packet Rate: по умолчанию: 75 - время ожидания (в миллисекундах) передачи данных ведомого.

Fragmentation timeout: [мс]: по умолчанию 1600 мс. Если размер передаваемых данных превышает 8 байт, то они подлежат фрагментированию, то есть будут разбиты на несколько пакетов. Данный таймаут определяет длительность ожидания мастером пакета фрагментированных данных. При превышении таймаута включается обработка ошибки '*Action on timeout error*'.

Action on timeout error: определяет способ обработки ошибки по таймауту:

Transition to time out: (по умолчанию) действие определяется ведомым.

Auto delete: соединение для входов-выходов будет удалено.

Auto reset: соединение сохраняется, мастер переконфигурирует ведомого, сбрасывается сторожевой таймер.

Дополнительные установки для типа соединения 'Change of state':

Lock time for sending: (по умолчанию:1) минимальный интервал (в миллисекундах) между сообщениями, даже если данные изменились раньше. Этот метод помогает избежать избыточных сообщений в сети. "0" означает отсутствие интервала, в этом случае данные будут переданы так быстро, как только возможно.

Timeout[ms]: (по умолчанию: 16) если «сердцебиение» ожидается более указанного времени, то обнаруживается ошибка по таймауту.

Heartbeatrate[ms]: (по умолчанию 250) интервал в миллисекундах, после которого ведомый должен осуществить передачу данных, даже если изменение данных не произошло.

Дополнительные установки для типа соединения 'Bit Strobe':

Use output bit: при ответе мастеру ведомый будет использовать выходной бит, соответствующий использованному мастером в запросе.

Дополнительные установки для типа соединения 'Cyclic':

Interval [ms]: интервал в миллисекундах, в соответствии с которым ведомый автоматически передает данные (сердцебиение).

Timeout [ms]: если «сердцебиение» ожидается более указанного времени, то обнаруживается ошибка по таймауту.

Inputs (входы):

Выберете нужные входы в поле **Available connections** и переместите их в поле **Configured input connections** с помощью кнопки >> . Кнопка << позволяет удалить элемент списка.

Для изменения размера выбранного входа используйте двойной щелчок мышки. В диалоге **Length of connection** введите необходимую длину в байтах **Length in Bytes**. Длина будет отображаться в скобках после имени входа.

Выбранные входы будут немедленно включены в дерево конфигурации. Они показываются с отступом ниже ведомого с наименованием типа соединения.

Outputs (выходы):

Конфигурирование выходов выполняется точно так же, как и входов.

Параметры DeviceNet-Slave

Перечисленные здесь параметры определяются EDS файлом. В соответствии с определением входов-выходов их значения будут передаваться по сети.

Obj.: идентификатор объекта используется для доступа к параметру в списке параметров (объектный словарь). Номер объекта образуется из соответствующего описания номера параметра в EDS файле (секция [Params], "Param<number>").

Тип: тип данных параметра.

Acc.: права доступа: gw=чтение и запись, go=только чтение.

Min., Max.: диапазон значений параметра, ограниченный по минимуму и максимуму.

Default: значение по умолчанию

Value: значение, как определено в EDS файле. Здесь можно редактировать параметр. Используйте список допустимых значений либо откройте поле редактирования щелчком мышки в ячейке таблицы.

Параметры модуля DeviceNet- Slave

Данный диалог параметров модуля идентичен диалогу параметров любого модуля (см. раздел 6.6.5. Базовые параметры модуля ввода-вывода).

6.6.11 Конфигурация ПЛК в режиме Online

В режиме Online конфигурация ПЛК отображает состояние входов и выходов ПЛК. Если логический вход или выход имеет значение TRUE, то перед его именем в дереве конфигурации отображается маленький прямоугольник, закрашенный голубым цветом. Для других типов переменных в конце строки отображается значение переменной (например, "=12").

Значение логического входа можно изменить щелчком мыши. Для других типов входов щелчок мыши открывает диалог изменения значения. Новое значение записывается в ПЛК сразу же по нажатию кнопки **ОК**.

Кроме того, в конфигурации может отображаться специфическая диагностическая информация (см. ниже).

6.6.12 Сканирование аппаратуры/ Состояние/ Диагностика ПЛК

Если поддерживается целевой системой и допускается в текущей конфигурации (*.cfg файл), то информация о структуре, состоянии модуля и результатах диагностики аппаратуры могут быть считаны из ПЛК и отображены в Конфигурации ПЛК CoDeSys:

Сканирование конфигурации модулей

Если поддерживается целевой системой и допускается в текущей конфигурации, то контекстное меню конфигурации ПЛК содержит команду **Scan module configuration**.

Данная команда доступна только в режиме offline. Команда запускает сканирование актуального состава включенных аппаратных модулей и предлагает включить найденные модули в дерево конфигурации

ПЛК. Это простейший метод создания и отображения реальной конфигурации для существующих аппаратных средств.

Определение состояния модуля

Если поддерживается целевой системой и допускается в текущей конфигурации, то контекстное меню конфигурации модуля содержит **Load module state**.

Данная команда доступна только в режиме online. Команда определяет актуальный (текущий) статус модуля и отображает его в дереве конфигурации цветом:

Черный:	модуль существует и настроен корректно.
Голубой:	модуль существует, но настроен не корректно.
Красный:	модуль не найден.

Обновление статусной информации будет выполняться автоматически при каждой загрузке.

Отображение диагностических сообщений

Если поддерживается целевой системой и допускается в текущей конфигурации, то контекстное меню конфигурации модуля содержит **Show diagnosis messages**.

Данная команда доступна только в режиме online. Если команда активирована, то диагностические сообщения от модулей ПЛК будут отображаться в окне CoDeSys.

6.7 Конфигуратор задач (Task Configuration)

6.7.1 Обзор

Кроме использования специальной «главной» программы PLC_PRG, управлять выполнением проекта можно с помощью задач.


Задача - это единица обработки МЭК программы. Задача имеет название, приоритет и тип. Тип определяет условие вызова задачи. Условием может служить время (циклическое или свободное *freewheeling* выполнение) или событие, внутреннее или внешнее (например, превышение заданного порога глобальной переменной или прерывание в контроллере).

Для каждой задачи назначается ряд программ, которые будут в ней выполняться. Если задача выполняется в текущем цикле, это означает, что выполняются включенные в неё программы (по одному циклу каждая). Комбинация приоритетов и условий вызова определяет хронологический порядок выполнения задач.

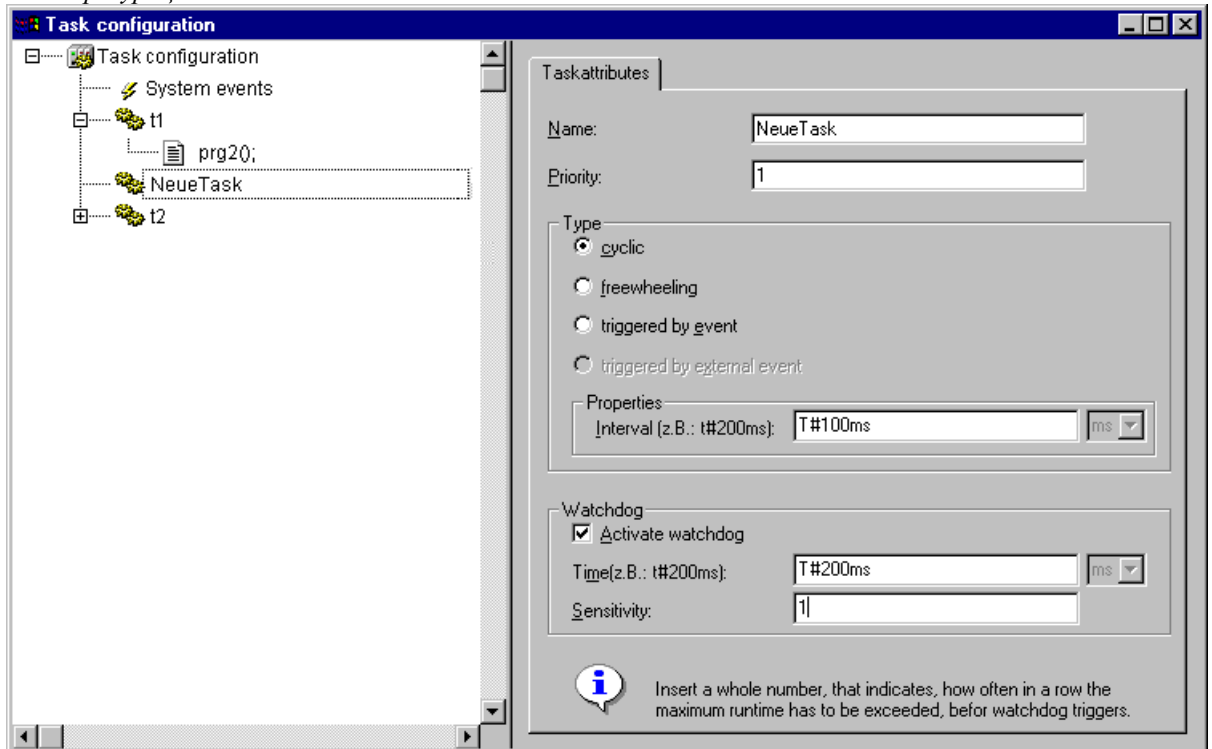
Каждую задачу можно разрешить или запретить независимо от других.

Для каждой задачи можно задать **сторожевой таймер** (контроль времени выполнения). Возможности его использования и настройки определяются целевой платформой.

В режиме Online выполнение задач можно наблюдать в виде графической диаграммы. Помимо этого, существует возможность непосредственно связать системные события **System events** (т.е. Старт, Стоп, Сброс) с выполнением определенных POU проекта.

Раздел **Конфигурации задач**  (**Task Configuration**) находится во вкладке **Ресурсы (Resources)** Менеджера объектов. Окно Конфигурации задач разделено на 2 части.

Окно Конфигурации задач



В левой части окна Task Configuration представлены задачи в виде дерева конфигурации. В корневой позиции обязательно присутствует элемент 'Task configuration'. Под ним раскрывается список конкретных задач, представленных по именам. Под каждой задачей раскрывается список включенных в неё программ. Тип каждого элемента определяется специальной иконкой.

В правой части окна показан диалог, соответствующий выбранному элементу в дереве конфигураций. Здесь вы можете конфигурировать свойства задач (Task properties), вызова программ (Program call), задавать связи с системными событиями (System events). Эта возможность зависит от выбора целевой платформы. Она должна быть поддержана в системе исполнения и разрешена в опциях целевой системы. Если стандартный набор настроек расширен специфическими параметрами, они будут представлены на отдельной вкладке 'Parameter' в правой части окна.

Примечание: Не используйте одну строковую функцию одновременно в нескольких задачах, это может вызвать пересечение данных (см. standard.lib).

6.7.2 Работа в конфигураторе задач

- § Наиболее важные команды находятся в контекстном меню (правая клавиша мыши).
- § В корневой позиции дерева конфигурации задач находится строка "Task Configuration". Если перед этой строкой стоит знак "плюс", то список задач закрыт. Открыть этот список можно двойным щелчком мыши или клавишей <Enter>. При этом появится знак "минус". Если щелкнуть мышкой еще раз, то список снова закроется.
- § Каждой задаче соответствует список вызовов программ. Открывается и закрывается этот список аналогично.
- § С помощью команды "Insert" "Insert Task" можно вставить задачу.
- § Команда "Insert" "Append Task" добавляет задачу в конец списка.

§ Вызов программы в выбранную задачу вставляется командой **"Insert " "Insert Program Call"**.

Для каждого выбранного в левой части окна элемента в правой части окна показывается соответствующий диалог настройки. Некоторые опции можно включить/выключить, некоторые из них требуют ввода строки. В зависимости от выбранного элемента диалог может представлять атрибуты задачи 'Taskattributes' (см. 'Insert Task'), определение вызова программы 'Program Call' (см. 'Insert Program Call') или таблицу системных событий 'System events'. Возможности настройки определяются целевой платформой. Изменения в настройках вступают в силу после выбора другого элемента (потери диалогом фокуса ввода).

§ Изменить имя задачи или программного вызова можно, нажав клавишу <пробел> или щелкнув мышкой на соответствующем объекте.

§ Перемещаться по элементам дерева конфигурации задач можно при помощи клавиш со стрелками (вверх, вниз).

"Insert" "Insert Task" или "Insert" "Append Task"

С помощью этих команд можно вставить новую задачу.

Если выделена задача или элемент 'System events', то доступна команда **"Insert Task"**. Используя эту команду, можно вставить новую задачу перед курсором. Если выделена строка "Task Configuration", то доступна команда **"Append Task"** и новая задача добавляется в конец списка. После этого будет открыто диалоговое окно для установки атрибутов задачи (См. выше рис. Окно конфигурации задач).


Максимально возможное число задач определяется целевой системой. Обратите внимание, что некоторое число задач может оказаться уже зарезервировано для модулей PLC Configuration (определено в cfg-файле).


Диалог атрибутов задачи включает:


Name: имя задачи. Имя отражает задачу в дереве конфигурации. Для редактирования имени нужно щелкнуть по нему мышью или выбрать элемент и нажать <пробел>.


Priority (0-31): приоритет (Priority) (число от 0 до 31 со следующими значениями: 0 - самый высокий, 31 – самый низкий приоритет)

Тип:

cyclic  : задача вызывается циклически через заданный интервал времени 'Interval' (см. ниже).

freewheeling  : задача вновь вызывается сразу же после окончания в непрерывном цикле, без задания каких-либо интервалов.

triggered by event  : задача вызывается по фронту значения логической переменной, определенной в поле **Event**.

triggered by external event  : задача вызывается по событию, которое определено в поле Event . Список доступных событий зависит от целевой платформы.

Свойства:

Interval (для задач типа 'cyclic'): период времени, после которого задача должна быть вызвана в очередной раз. Справа от поля ввода значения задаются единицы измерения. Это могут быть миллисекунды [ms] либо микросекунды [µs]. При установке миллисекунд значение должно быть

дано в формате TIME (например, "t#200ms"). При вводе микросекунд используется числовая форма представления (например, "300").

Event (для задач типа 'triggered by event' и 'triggered by external event'): глобальная переменная инициирующая запуск задачи передним фронтом. Используйте кнопку ... или Ассистент ввода <F2>. Некоторые целевые системы поддерживают **Singleton Events**. Это события, которые позволяют запустить только одну единственную задачу, что контролируется при компиляции проекта. При контроле используется адрес переменной, но не ее имя. Например: если в целевой системе определены %MX1.1 и %IB4 как Singleton-Events, то следующее объявление вызовет 2 ошибки (a и b так же, как c и d, занимают одинаковые адреса):

```
VAR_GLOBAL
  a AT %MX1.1: BOOL;
  b AT %MX1.1: BOOL;
  c AT %MB4: BOOL;
  d AT %MD1: BOOL;
END_VAR
```

Если в обоих полях 'Interval' и 'Event' ничего не задано, то время цикла определяется системой исполнения. Например, для CoDeSys SP NT V2.2 и старше время цикла будет 10 ms.

Сторожевой таймер (Watchdog):

Для каждой задачи можно определить контроль времени выполнения (таймер-сторож). Если целевая платформа поддерживает расширенную конфигурацию таймера-сторожа, то могут быть определены максимальное, минимальное и значение по умолчанию. Также может присутствовать определение времени в процентах.

Activate watchdog: если сторожевой таймер включен, то задача будет прервана с установленным статусом ошибки, если её выполнение заняло больше времени, чем задано в поле 'Time' (см. ниже).


Time (например: t#200ms): время сторожевого таймера; если задача не завершила работу в течение этого времени, сторожевой таймер «срабатывает». В зависимости от целевой платформы время вводится в процентах от интервала задачи. В этом случае селектор единиц измерения показан серым цветом и отображает "%".

Sensitivity: чувствительность, допустимое число превышений времени сторожа без формирования признака ошибки.

Вспомогательные специфические атрибуты:

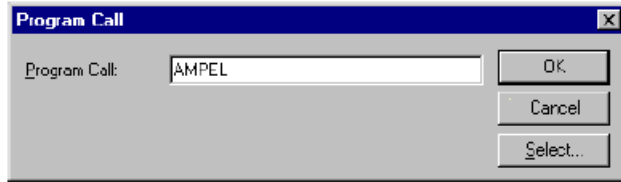
Помимо описанных стандартных атрибутов, изготовителем оборудования задаче могут быть сопоставлены вспомогательные специфические атрибуты. Они отображаются на вкладке "Parameters". В этом случае файлы конфигурации платформы включают специальный файл описания конфигурации задач.

"Insert" "Insert Program Call" или "Insert" "Append Program Call"

С помощью этих команд можно открыть диалоговое окно добавления программ в задачу. Каждая программа в дереве конфигурации задач отмечена иконкой .

При использовании команды **"Insert Call"** можно вставить новый программный вызов перед курсором, **"Append Program Call"** позволяет добавить вызов в конец существующего списка.

Диалоговое окно ввода вызова программы



В поле **Program Call** нужно написать имя программы, которая есть в проекте, или с помощью кнопки **Select** выбрать его из списка предложенных. Если программа имеет входные параметры, то они приводятся в скобках как при обычном вызове программы (например, prg(invar:=17)).

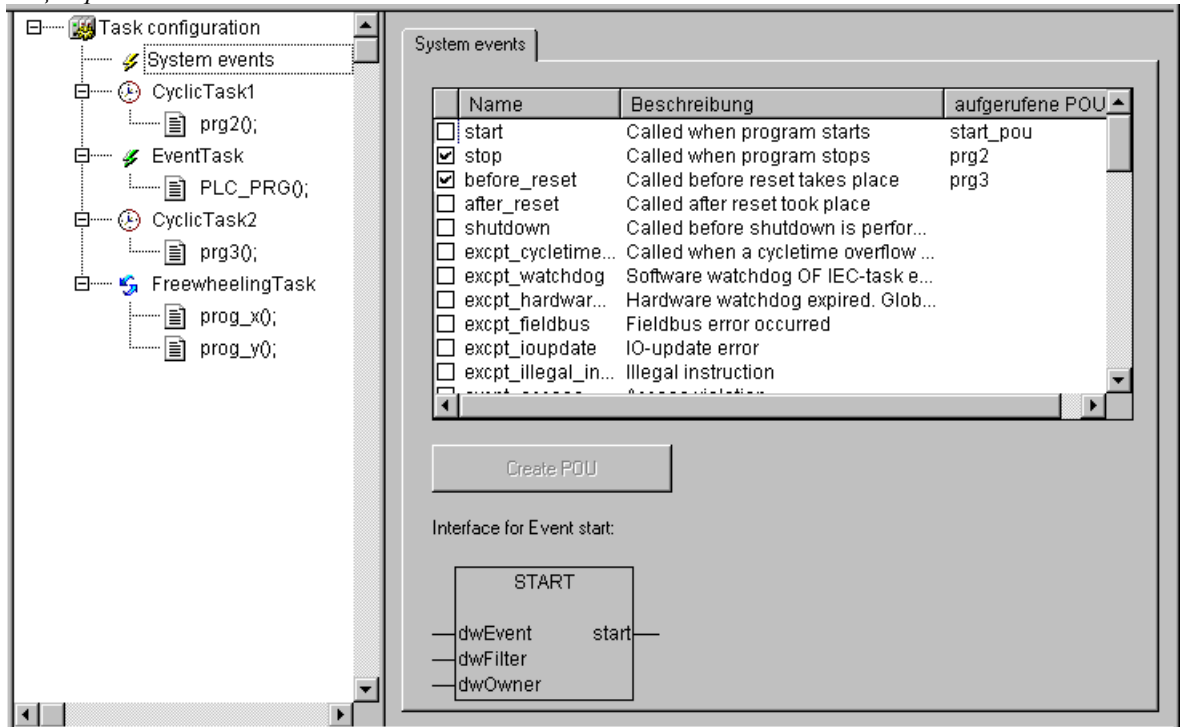
Порядок выполнения программ в задаче определяется их последовательностью в списке (сверху вниз).

6.7.3 Системные события

Помимо "задач", еще и "системные события" могут приводить к вызову POU в проекте. Доступные системные события определяются целевой платформой. Список стандартных событий может быть дополнен специфическими событиями пользователя. Возможные события, например: Старт, стоп, Online коррекция.

Присваивание системных событий POU выполняется в Конфигураторе задач. Диалог 'Events' открывается при выборе элемента ⚡ "System-events" в дереве конфигурации:

Таблица присваивания системных событий POU:



Каждое событие (поддерживаемое в выбранной целевой платформе) представлено в отдельной строке, содержащей поля **Name** и **Description** (описание). Присваивание POU производится в правом столбце таблицы (**called POU**).

Для этого используйте Ассистент ввода (<F2>) или введите вручную имя POU, имеющегося в проекте (например, "PLC_PRG" или "PRG.ACT1"). Если вы задали имя отсутствующего в проекте POU, нажми-

те кнопку Create POU. После чего новый POU будет вставлен в Организатор объектов. Под таблицей графически показан интерфейс программного компонента, соответствующий выбранному событию.

Для включения вызова POU по событию необходимо включить флажок в первой колонке таблицы. Включение/выключение выполняется щелчком мышки.

6.7.4 Конфигуратор задач в режиме online

В режиме online дерево конфигурации задач отображает статус каждой задачи и число отработанных ею циклов. Временная диаграмма работы задач показана в правой части окна. Предварительное условие: для поддержки функции мониторинга времени библиотеки *SysTaskInfo.lib* и *SysLibTime.lib* должны быть включены в проект. Данные библиотеки отсутствуют, если не установлены целевые платформы, поддерживающие мониторинг задач.

Отображение статуса задач в дереве конфигурации:

В режиме online статус задач и число циклов отображаются справа от имени задачи в квадратных скобках. Время обновления экрана такое же, как и при обычном мониторинге переменных.

Возможный статус:

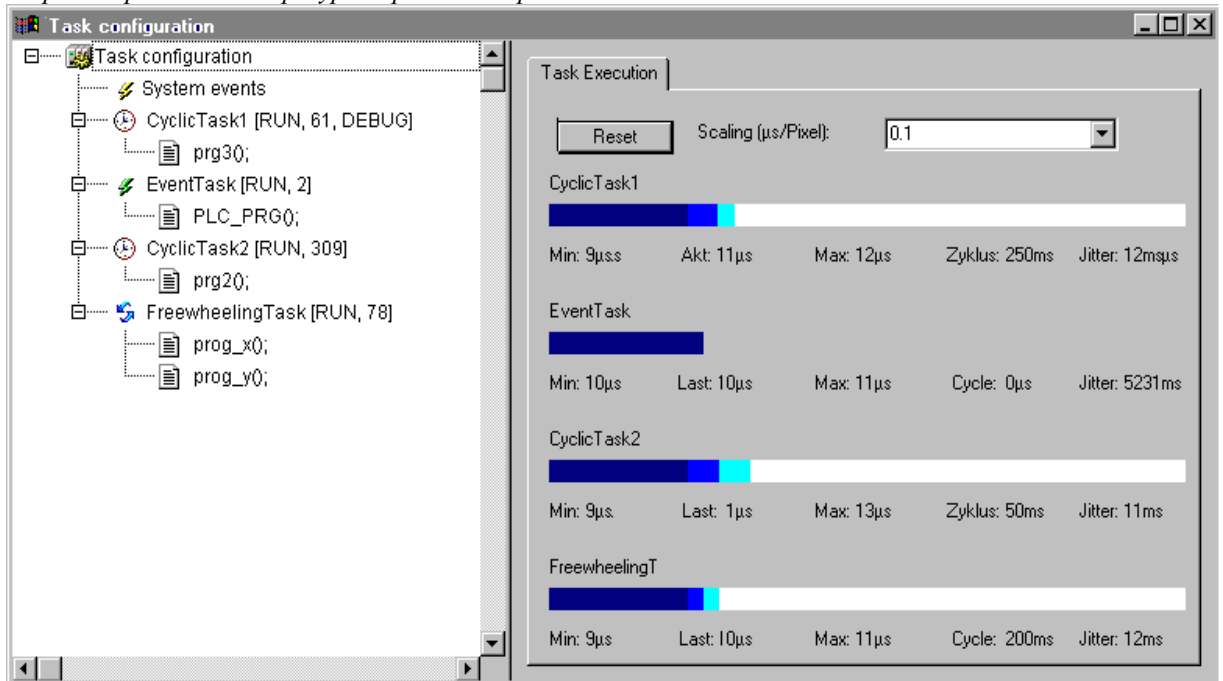
Idle	не выполнялась после последнего обновления; характерно для задач управляемых событиями.
Running	выполнялась не менее одного раза после последнего обновления.
Stop	остановлена
Stop on BP	остановлена, по причине достижения точки останова
Stop on Error	ошибка, например: деление на ноль и др.
Stop Watchdog	выполнение прервано сторожевым таймером

При статусах 'Stop on Error' или 'Stop Watchdog' задача отображается красным цветом .

Отображение диаграмм времени выполнения задач

При выборе 'Taskconfiguration' в дереве конфигурации задач в правой части окна отображаются результаты мониторинга времени выполнения задач.

Пример отображения конфигулятора задач в режиме online



Для каждой задачи отображается столбчатая диаграмма. Длина столбца отражает длительность цикла. Под каждым столбцом диаграммы приведены числовые значения:

- Min:** минимальное измеренное время выполнения в мкс.
- Akt:** последнее измеренное время выполнения в мкс.
- Max:** максимальное измеренное время выполнения в мкс.
- Cycle:** общее время цикла в мкс.
- Jitter:** максимальное измеренное дрожание (jitter) в μs (время между запуском задачи и моментом отображения ее работы операционной системой).

Кнопка **Reset** сбрасывает значения минимумов, максимумов и джиттера в 0.

Масштабирование диаграммы (микросекунд на пиксель) настраивается в поле **Scaling [µs/Pixel]**.

Какие задачи будут выполнены?

При выполнении задач применяются следующие правила:

- Выполняется та задача, условия которой истинны, т.е. прошло указанное время или переменная получила значение ИСТИНА.
- Если несколько задач имеют одинаковые условия, тогда выполняется задача с наивысшим приоритетом.
- Если несколько задач имеют одинаковые условия и приоритет, то выполняется та, которая имеет большее время ожидания.
- Программы одной задачи выполняются в том же порядке, в каком они перечислены в списке конфигурации задачи.

- В зависимости от целевой платформы PLC_PRG может работать как free-wheeling задача без необходимости определять ее в конфигурации задач.

Дополнительные online функции контекстного меню и меню 'Extras':

"Extras" "Set Debug Task"

С помощью этой команды в режиме Online задачу можно сделать отлаживаемой. После имени отлаживаемой задачи появится слово [DEBUG]. Тогда функции отладки применяются только к этой задаче. Другими словами, программа останавливается в точке останова, только если она вызвана такой задачей.

'Extras' 'Enable / disable task'

Запрещает или разрешает выполнение выбранной в дереве конфигурации задачи. Запрещенная задача не вызывает свои программы и отображается в дереве конфигурации серым цветом.


'Extras' 'Callstack'

Данная команда присутствует в меню Extras Конфигуратора задач. Если программа остановлена в точке останова, то с помощью этой команды можно просмотреть стек вызовов. Отлаживаемая задача должна быть выбрана в дереве конфигурации задачи. Стек вызовов показывается в окне 'Callstack of task <task name>'. Здесь вы получите имя POU и позицию точки останова (например: "prog_x (2)" для второй строки prog_x) . Ниже отображается стек вызовов в обратном порядке. Кнопка 'Go To' позволяет переместиться в помеченную в стеке вызовов позицию POU.

6.8 Менеджер рецептов (Watch and Receipt Manager)

6.8.1 Обзор

Менеджер контроля и заказа значений переменных или коротко «Менеджер рецептов» позволяет просматривать значения заданного списка переменных, заранее определять наборы констант для переменных данного списка и моментально присваивать их одной командой записи: "**Write Receipt**". Кроме того, образ текущих значений переменных контроллера можно считать и сохранить (команда "**Read Receipt**"). Эта функция полезна, например, для быстрого восстановления определенного состояния системы.

Для работы с  **Watch and Receipt Manager** откройте соответствующее окно на вкладке **Resource** организатора объектов.

Перечень доступных списков переменных находится в левой части окна. Списки можно выбирать с помощью мышки или клавишами управления курсором. В правой части окна показывается состав выбранного списка.

6.8.2 Менеджер рецептов в режиме Offline

В режиме Offline можно создать несколько именованных списков переменных, используя команду "**Insert**" "**New Watch List**". Сами переменные вводятся с помощью Ассистента ввода или с клавиатуры согласно следующему правилу:

<POUName>.<Variable Name> [:= <Const>]

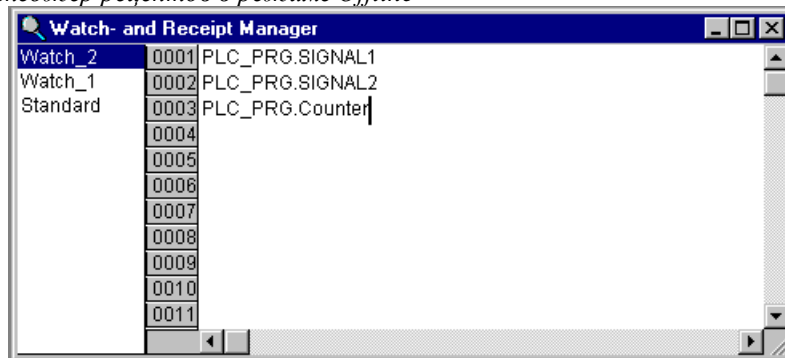
Для глобальной переменной часть **POU Name** (имя программного компонента) не нужна. Ввод глобальной переменной начинается с точки. Имя переменной может быть многоуровневым. Можно использовать прямые адреса.

Пример многоуровневой переменной:

PLC_PRG.Instance1.Instance2.Structure.Componentname

Пример глобальной переменной: **.global1.component1**

Менеджер рецептов в режиме Offline



Для присвоения константы переменной используется стандартный оператор присваивания : =. В режиме online заданные значения констант можно будет записать в контроллер командой 'Write Receipt'.

Пример:

```
PLC_PRG.TIMER:= 50
```

В примере переменная PLC_PRG.TIMER получает значение 50.

По команде "**Extras**" "**Write Receipt**" заданные константы будут переданы в PLC.

Относительно массивов и структур: Вы должны ввести имена отдельных элементов для последующего доступа к ним. Например: вы определили структуру с элементами *a, b, c* объявили переменную *struvar* данного типа в PLC_PRG.

Предварительное помещение в список элементов *a,b,c* вместе со значениями выполняется так:

```
PLC_PRG.struvar.a:=<value>
PLC_PRG.struvar.b:=<value>
PLC_PRG.struvar.c:=<value>
```

Предварительное помещение в список элементов массива выполняется аналогично. Например, для массива переменных типа ARRAY[0..6]:

```
PLC_PRG.arr_var[0]:=<value>
PLC_PRG.arr_var[1]:=<value>
...
```

Для функционального блока *fb*, содержащего переменные *x, y* и имеющего экземпляр *fb_inst*, объявленный в PLC_PRG, введите следующие строки:

```
PLC_PRG.fb_inst.x:=<value>
PLC_PRG.fb_inst.y:=<value>
```

"Insert" "New Watch List"

Создать новый список переменных, имя которого нужно будет ввести в отдельном диалоговом окне.

"Extras" "Rename Watch List"

Изменить имя выбранного списка переменных.

"Extras" "Save Watch List"

Сохранить выбранный список переменных в файле с расширением "*.wtc".

"Extras" "Load Watch List"

Загружает список переменных, сохраненный командой "Save Watch List".

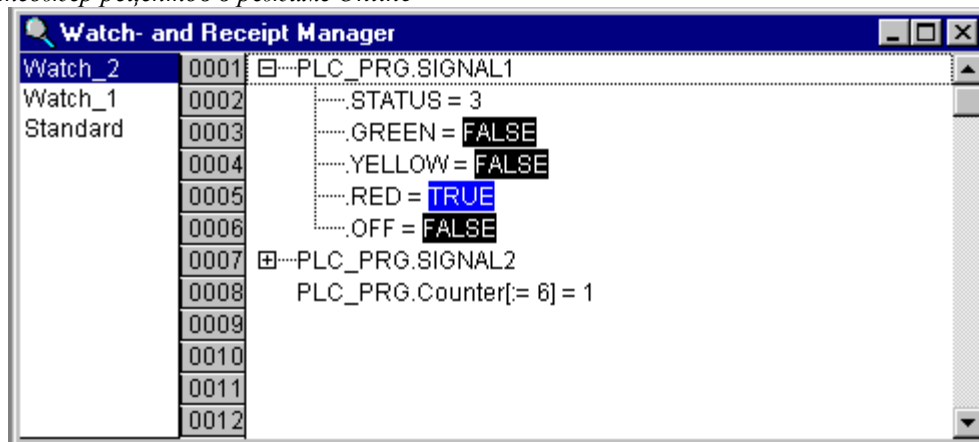
6.8.3 Менеджер рецептов в режиме Online

В режиме Online значения выбранного списка переменных отображаются на экране.

Переменные структурного типа (массивы, структуры, экземпляры функциональных блоков) помечаются значком "плюс" перед именем переменной. Открывается такая переменная двойным щелчком мыши или клавишей <Enter>. Для просмотра значений переменных экземпляров функциональных блоков используйте команды контекстного меню 'Zoom' и 'Open instance'.

Для добавления новых переменных надо отключить режим отображения командой "Extra" "Active Monitoring". Сделайте необходимые изменения и включите отображение этой же командой.

Менеджер рецептов в режиме Online



В режиме Online значения определенных заранее в списке констант записываются в контроллер командой "Extras" "Write Receipt".

Команда "Extras" "Read Receipt" заменяет константы текущими значениями переменных.

Замечание. Загружаются только те переменные, которые выбраны в Watch and Receipt Manager.

"Extras" "Monitoring Active"

В режиме Online эта команда включает или выключает окно показа значений переменных. Если выбран режим показа значений переменных, то команда отмечена галочкой.

Для ввода новых переменных или задания констант (см. режим Offline) режим должен быть выключен с помощью этой команды.

"Extras" "Write Receipt"

С помощью этой команды заданные константы загружаются в контроллер.

"Extras" "Read Receipt"

С помощью этой команды в режиме Online константы, определенные в режиме Offline, заменяются на текущие значения переменных.

Замечание. Загружаются только те переменные, которые выбраны в Watch and Receipt Manager

Фиксация переменных

В окне Watch and Receipt Manager вы можете изменять значения переменных, как и при мониторинге программ. Запись значений в ПЛК и фиксация значений выполняются обычными командами "Write values" и "Force values". Фиксированные значения отображаются красным цветом.

6.9 Трассировка (Sampling Trace)

6.9.1 Обзор

Трассировка или осциллографирование позволяет просмотреть значения переменных в определенном отрезке времени. Числовые значения переменных записываются в циклический буфер (буфер трассировки). Если буфер заполняется полностью, то ранние данные постепенно перезаписываются. Одновременно можно трассировать не более 20 переменных. Максимальное количество значений переменных равно 500.

Так как размер буфера трассировки ограничен, то если переменных очень много и они имеют большой размер (например, DWORD), то трассировка происходит по меньшему количеству значений. Например, если трассируется 10 переменных типа WORD, а объем буфера памяти в контроллере 5000 байтов, то каждая переменная трассируется по 250 значениям.

Для перехода в окно трассировки выберите объект  **Sampling Trace** на вкладке ресурсов Организатора проекта.

Для выполнения трассировки нужно определить список трассируемых переменных **"Extras" "Trace Configuration"** и задать параметры их отображения (цвета, масштаб осей координат и др.). Затем необходимо запустить процесс трассировки **"Start Trace"** и, наконец, считать буфер трассировки **"Read Trace"**. Значения заданных переменных будут отображены в виде графиков.

Готовая трассировка (конфигурация и значения переменных) может быть сохранена (собственный формат *.trc или открытый XML *.mon) и повторно открыта. Конфигурация сохраняется в *.tcf файле. Таким образом, можно сохранить и изучить несколько результатов трассировки.

Обратите внимание: Если используется Менеджер задач для управления программами, функция трассировки относится к отлаживаемой задаче (debug task).

6.9.2 Конфигурация трассировки

Для определения списка трассируемых переменных и настройки параметров трассировки предназначено диалоговое окно **Trace Configuration**.

Оно вызывается при двойном щелчке на серой области окна **Sampling Trace** либо командой:

"Extras" "Trace Configuration"

Диалоговое окно Trace Configuration

Название данной конфигурации трассировки определяется в поле **Trace Name**.

Список трассируемых переменных:

Изначально список трассируемых переменных **Variables** пуст. Для добавления новой переменной используйте поле ввода **Input of trace variable** (по завершению ввода нажать кнопку **"Insert"** или клавишу **<Enter>**). Вы можете интерактивно выбрать необходимые переменные проекта, нажав на кнопку **"Help Manager"**. Чтобы удалить переменную из списка, надо выбрать ее и нажать кнопку **"Delete"**.

Условия окончания процесса записи в циклический буфер:

При необходимости "отловить" определенный фрагмент трассируемых данных следует использовать триггерную переменную: **Trigger Variable**.

В поле **Trigger Variable** вводится логическая или аналоговая переменная (из числа перечисленных в поле **Variables**). Эта переменная определяет условие завершения трассировки. Для аналоговой переменной необходимо указать числовое значение порога. При переходе триггерной переменной через указанный порог процесс трассировки будет остановлен. Если выбрана опция **Trigger edge: positive**,

останов происходит при переходе триггерной переменной через значение **Trigger Level** в сторону увеличения. При выборе опции **Trigger edge: negative** останов происходит при переходе порога в сторону уменьшения. Опция **Bouth** обеспечивает останов при любом переходе значения. Опция **None** запрещает отслеживание триггерной переменной.

Если триггер-переменная не установлена, то буфер трассировки заполняется непрерывно до команды **"Stop Trace"**.

Поле **Trigger Position** определяет, какой процент значений будет записан до условия. Например, если ввести 25, то 25% будет считано до достижения условия, а 75% - после. Затем трассировка будет закончена.

Поле **Sample Rate** используется для установки периода между записями значений переменных. Этот интервал задается в миллисекундах. Значение по умолчанию 0. В этом случае запись значения происходит один раз за цикл.

Выбор режима обновления экрана:

В режиме **Single Number of samples** значения читаются и выводятся один раз. В режиме **Continuous** запрос трассировки инициируется заново каждый раз после чтения. Если, например, Number of samples равно 35, то первое изображение содержит первые 35 вычисленных значений, а вторые 35 записанных значений считываются автоматически. Режим **Manual** используется для ручного чтения результатов трассировки.

Команда **Save** используется для сохранения настроек трассировки в файле. Для этой цели открывается окно "File save as".

Открыть сохраненную конфигурацию можно с помощью кнопки **Load**. При этом открывается окно "File open".

Замечание. Команды Save и Load относятся к конфигурации трассировки, а не к вычисленным значениям.

6.9.3 Управление процессом трассировки

При работе в окне трассировки на панели инструментов отображаются кнопки, предназначенные для управления накоплением и отображением данных. Аналогичный набор команд доступен в меню Extras.

"Extras" "Start Trace"

Символ: 

Конфигурация трассировки передается в ПЛК, и начинается процесс записи значений переменных в кольцевой буфер.

"Extras" "Read Trace"

Символ: 

Значения переменных считываются из буфера трассировки и отображаются в виде графиков.

"Extras" "Auto Read"

Если выбрана эта команда, то текущий буфер трассировки читается автоматически и значения непрерывно изображаются на экране.

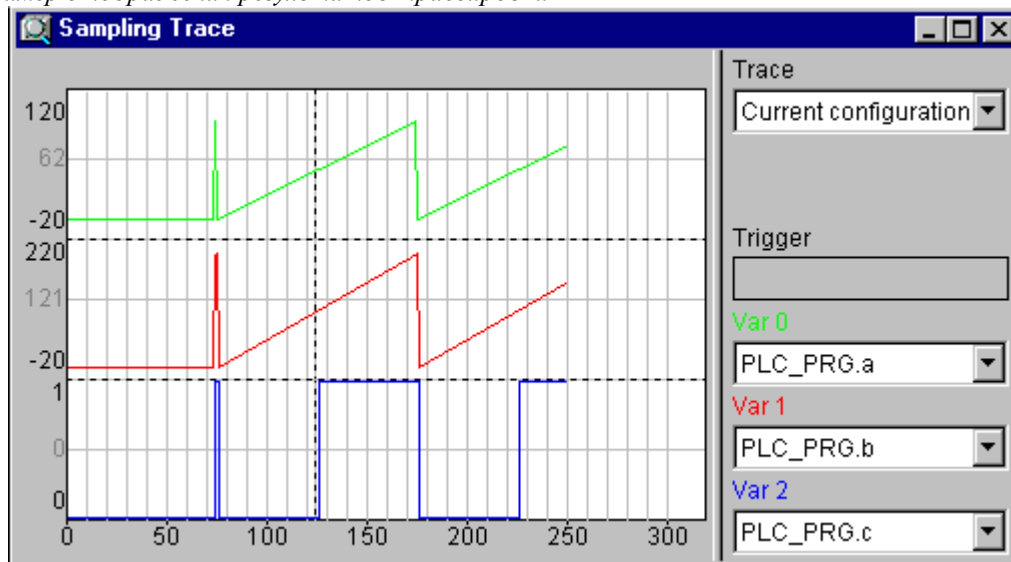
"Extras" "Stop Trace"Символ: 

Останавливает процесс записи данных в кольцевой буфер.

6.9.4 Отображение данных**Выбор изображаемых переменных**

Список изображаемых переменных находится в правой части окна **Sampling Trace**. Над каждой позицией расположена надпись: **Var0**, **Var1** и т.д. Цвет этой надписи определяет цвет, которым будет изображаться график значений соответствующей переменной. Имя переменной выбирается в выпадающем списке из числа определенных в конфигурации трассировки переменных. (На рисунке ниже PLC_PRG.a изображается зеленым). Цвета можно выбирать, даже если кривые уже построены. Одновременно в окне трассировки может изображаться до 8 кривых.

Пример отображения результатов трассировки



Если буфер трассировки заполнен ('Extras' 'Start Trace') и считан успешно ('Extras' 'Read Trace' или 'Extras' 'Auto Read'), то значения трассируемых переменных будут выведены на экран в виде кривых. В статусной строке CoDeSys указывается состояние (**State:**) буфера трассировки. При остановке трассировки ('Extras' 'Stop Trace') буфер данных трассировки очищается.

Если период сканирования установлен, то по оси **X** указывается отметки времени, в противном случае по оси **X** отложены номера отсчетов.

На оси **Y** показаны отметки значений переменной. Формат отметок определяется типом переменной. Вертикальная шкала автоматически масштабируется так, чтобы можно было увидеть наименьшее и наибольшее значения переменной. В примере наименьшее значение для верхней кривой Var 0 равно -20 и наибольшее +120.

Если произошло условие, заданное триггерной переменной, то точку, где оно произошло, указывает вертикальная пунктирная линия (см. рисунок).

Данные графиков сохраняются до выхода из CoDeSys или открытия другого проекта.

"Extras" "Cursor Mode"

Курсоры позволяют детально исследовать графические кривые. Самый простой путь установить курсор в области просмотра – это щелкнуть в ней левой клавишей мышки. Появившийся курсор можно перемещать с помощью мышки. В верхней части окна просмотра выводится текущая X-позиция курсора. В полях, следующих за надписями Var N, указываются мгновенные значения переменных, соответствующие позиции курсора.

Другой путь – это использование команды "**Extras**" "**Cursor Mode**". В окне появятся две вертикальные линии. Изначально они совпадают. Одну из этих линий можно перемещать, используя клавиши управления курсором. Если одновременно с клавишей управления курсором нажать <Control>, то скорость перемещения курсора увеличивается в 10 раз. При нажатии клавиши Shift перемещается другая линия. В этом случае будет выводиться разница значений переменных между этими линиями (dy).

"Extras" "Multi Channel"

Эта команда переключает одноканальный или многоканальный режим изображения кривых. В многоканальном режиме пункт меню "Multi Channel" отмечен галочкой.

Зададим многоканальный режим изображения. В этом случае окно будет разделено максимум на 8 областей, в каждой из которых изображается отдельная кривая. Для каждой кривой на шкале отмечается ее максимальное и минимальное значение.

В одноканальном режиме изображения все кривые отображаются с одной шкалой и в одной области.

"Extras" "Show grid"

Используя эту команду, можно включить или выключить сетку в окне. Если сетка включена, то соответствующий пункт меню отмечен галочкой.

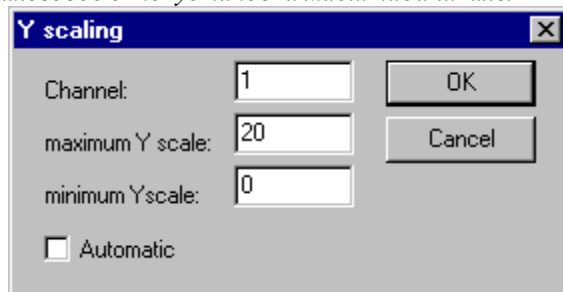
"Extras" "Y Scaling"

С помощью этой команды можно изменить масштаб шкалы Y для любой из кривых.

В диалоговом окне устанавливается номер нужной кривой (поле **Channel**) и минимальное и максимальное значение (поле **maximum y scale** и **minimum y scale**). В автоматическом режиме **Automatic** масштаб шкалы выбирается автоматически, по максимальному и минимальному значениям данных.

При двойном щелчке на кривой появляется то же диалоговое окно с текущими значениями.

Диалоговое окно установки масштаба шкалы




"Extras" "Stretch"

Символ: 

Увеличение масштаба шкалы X. Кривые растягиваются в ширину. Начальная позиция шкалы устанавливается с помощью горизонтального ползунка.

Команда с обратным действием называется **"Extras" "Compress"**.

"Extras" "Compress"

Символ: 

Сжатие по оси X. После выполнения этой команды на экране появляется последовательность значений переменной, соответствующая большему отрезку времени. Возможно многократное использование этой команды.

Команда с обратным действием называется **"Extras" "Stretch"**.

6.9.5 Сохранение данных трассировки ('Extras' 'Save trace values')

С помощью этих команд можно сохранить полученные значения переменных и конфигурацию трассировки в файле с расширением **"*.trc"**.

Эти команды отличаются от присутствующих в окне **"trace configuration"** и сохраняющих только конфигурацию трассировки.

См. также: **'Extras' 'External Trace Configurations'**

"Save Values"

Сохранить полученные значения переменных и конфигурацию трассировки в **"*.trc"** файле. Чтобы загрузить сохраненный файл, следует выбрать команду **"Extras" "Load Values"**.

"Load Values"

Считывает данные и конфигурацию трассировки из файла с расширением **"*.trc"**. Выбор файла производится в стандартном диалоге открытия файлов. Запись данных выполняется командой **"Save Values"**.

"Trace in" "ASCII -file"

Эта команда сохраняет последовательность значений переменной в ASCII файле. Открывается диалог для сохранения файла. Имя сохраненного файла получает расширение **.txt**. Значения хранятся в файле согласно следующей схеме:

```
CoDeSys Trace
D: \CODESYS\PROJECTS\TRAFFICSIGNAL.PRO - проект
Cycle PLC_PRG.COUNTER PLC_PRG.LIGHT1 - имена переменных
0 2 1 - данные
1 2 1
2 2 1
....
```

Если частота сканирования не установлена, то в первом столбце располагаются порядковые номера замеров, так как значения переменных сохраняются один раз в цикл.

В ином случае в этом столбце указывается время записи значений в миллисекундах от начала трассировки. В последующих столбцах сохранены соответствующие значения переменных. Значения, относящиеся к различным переменным, разделены пробелами.

6.9.6 Внешние конфигурации трассировки ('Extras' 'External Trace Configurations')

Команды этого меню используются для записи и восстановления данных и конфигурации трассировок в формате XML, загрузки конфигурации трассировки из внешних файлов в контроллер.

"Save to file"

С помощью этой команды можно сохранить полученные значения переменных и конфигурацию трассировки в XML формате. По умолчанию используется расширение файла: *.mon.

Загрузка *.mon-файла производится командой 'Load from file'.

"Load from file"

С помощью этой команды можно считать записанную ранее конфигурацию и данные трассировки из файла (*.mon) в формате XML. Выбор файла производится в стандартном диалоге открытия файлов. Если вы хотите далее использовать считанную конфигурацию в проекте, дайте команду 'Set as project configuration'.

Файл *.mon создается командой 'Save to file'.

Примечание: Альтернативный вариант сохранения данных дает команда 'Extras' 'Save values'.

'Load from controller'

Считывает из контроллера текущую, используемую им конфигурацию и данные трассировки. Считанные данные можно просмотреть и установить данную конфигурацию в качестве действующей.

'Save to target'

С помощью этой команды в режиме online можно загрузить в контроллер конфигурацию из XML файла (*.mon). Выбор файла производится в стандартном диалоге открытия файлов.

Для сохранения конфигурации в *.mon файлах используется команда 'Save to file'.

'Set as project configuration'

Конфигурация текущей трассировки, выбранная в поле 'Trace' (см. окно трассировки), устанавливается в качестве действующей конфигурации текущего проекта. Выпадающий список в поле 'Trace' представляет трассировки, загруженные командой 'Load from file' из *.mon файлов (с целью просмотра).

6.10 Рабочая область (Workspace)

Workspace

Данный объект вкладки 'Resources' отражает текущие опции проекта (см. 4.2, Project Options). Открывая объект Workspace, вы переходите в диалог 'Options', содержащий все текущие опции по категориям.

6.11 Менеджер параметров (Parameter Manager)

6.11.1 Обзор и подключение

Наличие Менеджера параметров зависит от целевой платформы, он может быть подключен в настройках целевой платформы на вкладке “**Network functionality**” (см. раздел 6.12).

С помощью Менеджера параметров переменные МЭК-программ, константы и системные переменные можно сделать доступными в сети, состоящей из систем, работающих под управлением CoDeSys. Он используется для организации обмена данными в сети, обычно fieldbus. Для этого вы должны создать список переменных и загрузить его в ПЛК.

Обратите внимание: Создать список переменных можно также с помощью директив компилятора (см. 5.2.3)

Что такое параметры?

В данном контексте параметры это:

- Ў Переменные МЭК-программ
- Ў Константы
- Ў Системные параметры, определяемые целевой платформой
- Ў Экземпляры функциональных блоков, структуры и массивы.

Каждый параметр описывается определенным набором атрибутов, таких, как, например, “default value” (значение по умолчанию), “access rights” (права доступа), а также уникальным ключом доступа – “index”, “subindex”, “name” (индекс, подиндекс, имя), который используется для адресации к параметрам при чтении данных из списка параметров или при записи данных в список параметров. Обмен данными выполняется с помощью коммуникационных сервисов и не требует знания адресов переменных и наличия дополнительных функций. Менеджер параметров является альтернативой сетевым переменным при организации обмена данными по сети.

Что такое список параметров?

Список параметров предназначен для описания параметров. Он сохраняется вместе с проектом и загружается в целевую систему, которая работает под управлением МЭК-программы, реализованной в данном проекте. Для каждого типа параметров используется определенный тип списка параметров.

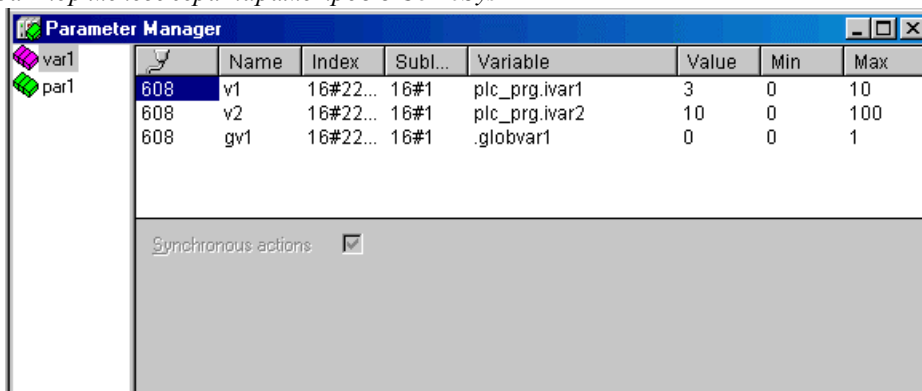
Каждому параметру соответствует отдельная строка в списке параметров. Столбец в списке параметров определяет какой-либо атрибут параметра. Кроме стандартного набора атрибутов, разработчик целевой платформы может определить дополнительные атрибуты для описания параметра в Менеджере параметров.

В файле, описывающем целевую платформу, определяется, какие атрибуты можно редактировать, какие атрибуты будут видимы в менеджере параметров, а также порядок следования атрибутов. Если такой файл отсутствует, то используется стандартный набор атрибутов, каждый из которых принимает значение по умолчанию.

Кроме списков констант и переменных проекта, в менеджере параметров можно определить список системных параметров. Эти параметры определяются целевой платформой. Вы можете создать список экземпляров функциональных блоков или структур. Такой список создается на основе шаблона, который также создается в менеджере параметров.

Так как список параметров хранится независимо от МЭК-программы, список параметров можно, например, использовать для сохранения “рецепта”, который не изменяется даже при изменении самой МЭК-программы.

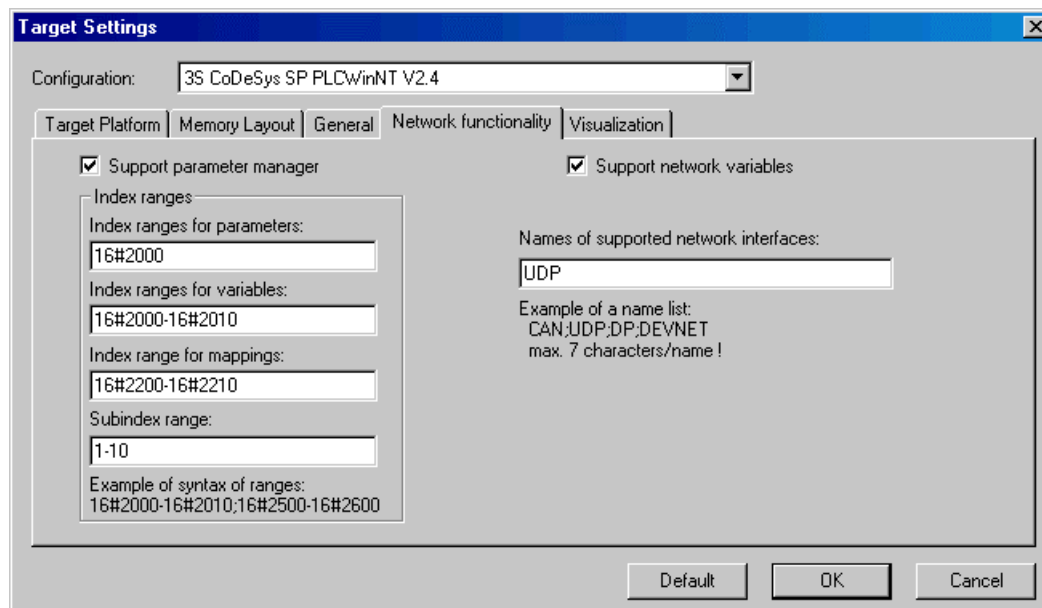
Редактор менеджера параметров в CoDeSys



Примечание: В зависимости от целевой платформы при создании загрузочного проекта список параметров также будет сохраняться в ПЛК.

Подключение менеджера параметров

В настройках целевой платформы откройте вкладку “**Network functionality**”:



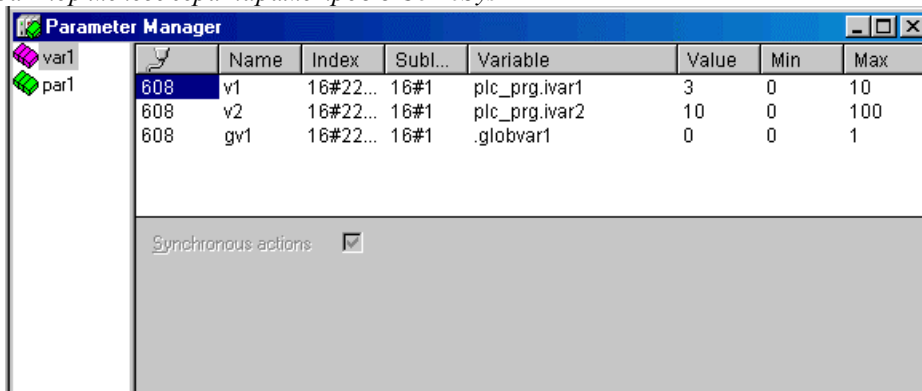
Установите флажок **Support Parameter Manager** и введите необходимые диапазоны индексов и подиндексов для списков параметров типа “Parameter” и “Variable”, а также, если позволяет целевая платформа, заполните поле **Index ranges for mappings** (диапазон индексов для PDO CAN устройств).

6.11.2 Редактор менеджера параметров. Обзор

Выберите объект “**Parameter Manager**” на вкладке “**Resources**”. Откроется окно, в котором вы сможете создавать и редактировать списки параметров, а также загружать их в режиме online в ПЛК и контролировать текущие значения параметров.

Примечание: Не забудьте подключить менеджер параметров и определить необходимые диапазоны индексов и подиндексов в настройках целевой платформы.

Редактор менеджера параметров в CoDeSys



Окно редактора разделено на две части. Левая часть предназначена для навигации по всем спискам параметров, которые созданы в Менеджере параметров. Правая часть (редактор таблицы) содержит таблицу, столбцы которой соответствуют атрибутам, а строки – параметрам.

В окне навигации вы можете вставлять, удалять и переименовывать списки параметров различных типов (Variables, Constant Parameters, Template, Instance, System Parameters).






Редактор таблиц предназначен для добавления параметра в список параметров и редактирования значений атрибутов. Каждый тип списка параметров имеет определенный набор атрибутов. Некоторые атрибуты могут быть невидимыми или недоступными для редактирования. Это определяется целевой платформой.


Для перемещения между окном навигации и редактором таблицы используйте <F6>

В режиме online вы можете загрузить заранее созданный список параметров в ПЛК. Также вы можете использовать менеджер параметров для обмена данными с другими системами. В окне менеджера параметров можно наблюдать текущие значения параметров. В режиме offline созданный список параметров сохраняется вместе с проектом.

6.11.3 Типы списков параметров и их атрибуты

Менеджер параметров может работать со следующими типами списков параметров:

-  **Variables:** список параметров данного типа содержит переменные проекта.
-  **Parameters:** список параметров данного типа содержит константы.
-  **System parameters:** список параметров данного типа содержит системные параметры, которые определяются целевой платформой. Такие списки не могут быть удалены или переименованы.
-  **Template:** Template (шаблон) не содержит параметров, к которым можно обращаться напрямую. Каждый элемент данного списка содержит “базовую конфигурацию атрибутов” для компонент функционального блока или структуры. Такой шаблон может использоваться в списке параметров типа “Instance”.
-  **Instance:** элементами такого списка являются экземпляры функциональных блоков и структуры. Для упрощения добавления новых элементов в список можно использовать шаблоны, которые перед этим нужно создать.

 **Mappings:** Этот тип списка параметров доступен в менеджере параметров только в том случае, если позволяет целевая платформа. Элементы такого списка представляют собой параметры, которые отображаются в PDO устройств CAN. Такой список похож на список типа Variables, но имеет свой собственный диапазон индексов и подиндексов. Этот диапазон определяется в настройках целевой платформы на вкладке “Network functionality”. Параметры из этого списка отображаются в PDO CAN-устройства, а параметры из списков типа “Variable” и “Instance” можно отобразить в PDO в диалоге “Default PDO mapping” объекта PLC Configuration.

Вид каждого списка параметров можно настраивать с помощью специального файла в XML формате. Если такого файла нет, то используются настройки по умолчанию.

Экземпляры и шаблоны

Список параметров типа “Instance”(экземпляр) состоит компонент экземпляров функциональных блоков, структур и массивов. Список экземпляров для функциональных блоков и структур основывается на шаблоне, который также создается в менеджере параметров для соответствующего функционального блока или структуры. При описании массива в менеджере параметров шаблон не используется, а используется массив, который уже описан в проекте.

Список параметров типа “Template”(шаблон) не содержит параметров, к которым можно напрямую обращаться для обмена данными. Шаблон определяет смещение индексов и подиндексов, а также набор атрибутов, которые будут описывать компоненты функционального блока или структуры. Шаблон используется при создании списка параметров типа “Instance” и таким образом упрощает добавление экземпляров функциональных блоков и структур в менеджер параметров.

Создание шаблона: в поле ввода Base POU введите название функционального блока или структуры, для которой вы хотите создать шаблон. Для этого удобно воспользоваться ассистентом ввода. Нажмите кнопку **Apply**, и компоненты выбранного POU будут добавлены в список параметров. Теперь вы можете изменить атрибуты каждого из параметров. Созданный шаблон доступен при создании списка параметров типа “Instance”.

С помощью команды **“Insert missing entries”**, которая вызывается из контекстного меню или меню “Extras”, можно обновить текущие элементы списка параметров. Это может пригодиться в том случае, если было изменено базовое POU или удалены некоторые элементы созданного списка параметров.

При создании списка параметров для массива создавать шаблон не обязательно. Шаблон ARRAY(массив) создается в менеджере параметров автоматически.

Если активирована опция **Synchronous actions** то все операции чтения/записи других POU, определенных для любых элементов списка, будут выполняться системой исполнения синхронно с вызовом соответствующего элемента.

Создание списка параметров типа “Instance”(экземпляр): выберите шаблон из списка “Template”. Это список содержит названия всех созданных шаблонов, а также элемент ARRAY, который нужно выбрать, чтобы добавить в список параметров массив. Нажмите кнопку “Apply”, для того чтобы добавить компоненты POU в список параметров.

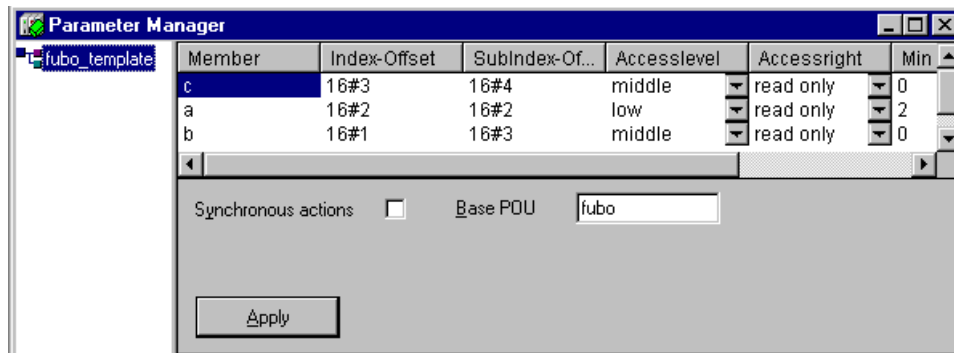
В поле **Base variable** введите имя экземпляра функционального блока, для которого вы создаете список параметров. Тип выбранного экземпляра и шаблон должны соответствовать друг другу.

Также для выбранного экземпляра необходимо ввести базовый индекс (поле ввода Base index) и базовый подиндекс (поле ввода Base subindex). Индекс и подиндекс отдельного компонента экземпляра POU вычисляются автоматически, путем сложения индексов/подиндексов шаблона (для массива их значения равны 0) с только что введенными базовыми индексами/подиндексами экземпляра. Например, вы ввели базовый индекс для компонента, равный 3, а в шаблоне определено смещение индекса, равное 3000. Созданный компонент получит индекс, равный 3003.

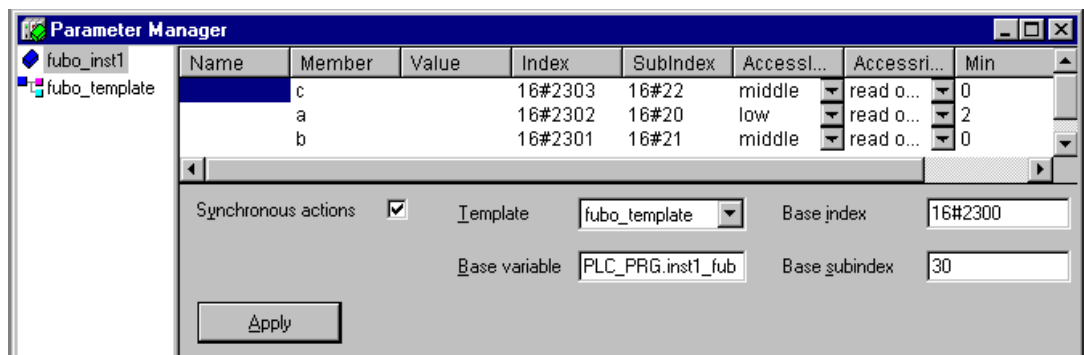
С помощью команды “**Insert missing entries**”, которая вызывается из контекстного меню или меню “Extras”, можно обновить текущие элементы списка параметров. Это может пригодиться в том случае, если было изменен шаблон или удалены некоторые элементы созданного списка параметров.

Пример:

Создайте функциональный блок fudo со входами или выходами a, b, c. В PLC-PRG создайте экземпляры этого функционального блока inst1_fudo и inst2_fudo. Для того чтобы создать списки параметров для переменных inst1_fudo.a, inst1_fudo.b, inst1_fudo.c и inst2_fudo.a, inst2_fudo.b, inst2_fudo.c, откройте менеджер параметров. Добавьте список параметров типа “Template” с именем fudo_template. Введите базовое POU – “fudo”. Нажмите кнопку Apply и введите значения некоторых атрибутов для компонент a, b, c: смещение индекса (Index-offset): для a:16#2, для b:16#1, для c:16#3. Также введите смещение подиндекса (Subindex-offset): для a:16#2, для b:16#3, для c:16#4.



Закройте шаблон и добавьте список параметров типа “Instance”. Выберите шаблон “fudo_template”, базовую переменную “inst1_fudo”, базовый индекс 16#2300, базовый подиндекс 30 (введенные значения должны соответствовать настройкам целевой платформы). После нажатия кнопки Apply в список параметров будут добавлены компоненты a, b, c вычисленными индексами (a:16#2302, b: 16#2301, c: 16#2303) и подиндексами (a:16#20, b: 16#21, c: 16#22).



На основе этих автоматически созданных элементов вы можете продолжить редактирование списка параметров.

6.11.4 Управление списками параметров

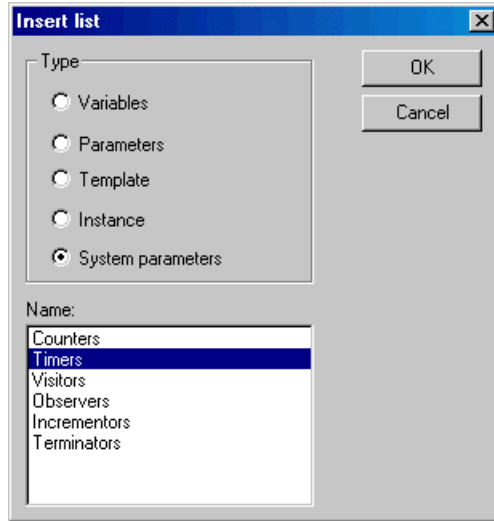
Вставка списка

Быстрый ввод: <Ins>







Чтобы добавить новый список параметров, используйте команду “List...” меню “Insert” или команду “Insert new list...” из контекстного меню. Эти команды доступны, когда фокус ввода находится в окне навигации.

При вызове этой команды появится следующее диалоговое окно:

Диалог “Insert list”



Введите имя нового списка параметров (оно должно быть уникальным в пределах одного типа) и выберите тип списка:

 Variables	Переменные проекта
 Parameters	Константы
 System parameters	Системные параметры, зависящие от целевой платформы
 Template	Шаблон для функциональных блоков или структур
 Instance	Экземпляр функционального блока или структуры, созданный на базе шаблона.
 Mappings	Параметры, которые отображаются в PDO устройств CAN. Наличие списка такого типа зависит от целевой платформы

Нажмите кнопку ОК, и список параметров появится в окне навигации менеджера параметров. Тип списка можно узнать по соответствующей иконке. В редакторе таблицы появятся колонки, которые соответствуют атрибутам параметров этого списка. Набор и порядок этих атрибутов зависит от файла описания целевой платформы. Если такого файла нет, то используются настройки по умолчанию. Теперь вы можете добавлять новые параметры в таблицу (см. раздел 6.11.4. Редактирование списка параметров).

Переименование списка

Для того чтобы переименовать список параметров, выберите его в окне навигации и выполните команду “Rename list ” из меню “Extras” или из контекстного меню. К тому же результату приводит щелчок левой кнопкой мыши по имени списка.

Вырезать/ Копировать/ Вставить список

Быстрый ввод: <Ctrl>+<X>, <Ctrl>+<C>, <Ctrl>+<V>.

Команда “Вырезать”(“Cut” из меню “Edit” и “Cut list” из контекстного меню) перемещает выбранный список параметров в буфер, и вы можете вставить это список в другую позицию в окне навигации командой “Вставить” (“Paste” из меню “Edit” и “Paste list” из контекстного меню). Для этого выберите название списка, перед которым вы хотите поместить список из буфера.

Команда “Копировать” (“Copy” из меню “Edit” и “Copy list” из контекстного меню) копирует выбранный список в буфер.

Удаление списка

Быстрый ввод:

Команда “Удалить” (“Delete” из меню “Edit” и “Delete list” из контекстного меню) удаляет выбранный в окне навигации список.

Обратите внимание: В режиме online эта команда удаляет выбранный список в системе исполнения.

6.11.5 Редактирование списка параметров**Внешний вид столбцов (атрибутов)**

Выбранный в окне навигации список параметров выглядит в редакторе таблицы так, как это определено файлом описания целевой платформы.

Значения атрибутов параметра находятся в одной строке в определенном, зависящем от типа списка, порядке.

Любой атрибут можно сделать невидимым (fade out). Для этого нужно дать соответствующую команду в контекстном меню, когда указатель мыши находится на названии нужного атрибута.

Размер столбца можно изменить с помощью мыши или вызвав команду из контекстного меню. Команда “Standard column width” устанавливает такую ширину столбца, при которой на экране видны все атрибуты. При помощи команды “Maximize width” можно сделать так, чтобы значения выбранного атрибута были видны полностью.

Команда для редактирования элемента списка параметров

Следующие команды находятся в контекстном меню, в меню “Insert ” и в меню “Extras”:

Вставка/Удаление элементов списка (строк таблицы)

Insert line, New line Новый элемент будет вставлен перед выбранным

Line after, New line after Новый элемент будет вставлен после выбранного

Быстрый ввод: <Ctrl>+<Enter>

Delete line Удаление выбранного элемента

Быстрый ввод: <Shift>+

Cut, Copy, Paste line Вырезать, скопировать, вставить выбранный элемент

Редактирование значений атрибутов:

При добавлении нового параметра в список его атрибуты получают значения по умолчанию. Для того чтобы ввести или изменить значение атрибута, щелкните мышкой на соответствующем поле ввода. Если значение атрибута можно редактировать, то введите нужное вам значение атрибута. Для ввода некоторых значений можно использовать ассистент ввода, который вызывается с помощью клавиши <F2>.

По завершению ввода нажмите клавишу <Enter>

Клавиши навигации можно использовать для перемещения между полями.

Для удаления выбранного значения атрибута нажмите клавишу .

Значения атрибутов могут быть представлены как в десятичном, так и шестнадцатеричном формате. Переключение между этими форматами производится командой **“Format Dec/Hex”** меню **“Extras”**.

Для перемещения фокуса ввода в окно навигации используйте клавишу <F6>.

Опции:

Для некоторых типов списков параметров доступны следующие опции:

Download with program: При соединении с контроллером список загружается автоматически.

Synchronous actions: пока не реализована.

Сортировка

Параметры могут быть отсортированы по значению атрибута - как по возрастанию, так и по убыванию.

Сортировать можно в режиме online и offline. Для того чтобы выполнить сортировку, кликните мышкой по названию атрибута. Рядом с названием атрибута появится стрелка, которая показывает порядок сортировки.


6.11.6 Менеджер параметров в режиме online

Обмен списком параметров между редактором и системой исполнения

Если поддерживается целевой системой, то в режиме online список параметров может быть как загружен в систему исполнения, так и считан из нее. Кроме того, вы можете записать значение одиночного параметра в систему исполнения. Максимальные размеры списков типа 'Variable' и 'Parameters' также зависят от целевой платформы.

Обратите внимание: Все списки параметров автоматически загружаются в систему исполнения по команде “Login”, если активна опция “Load with project”.

В режиме Online текущее значение параметра выводится в дополнительном столбце:

	N
608	v1
608	w?

В зависимости от целевой платформы при отображении значений применяются Index и Subindex либо RefID и Offset.

Для обеспечения обмена данными между контроллером и редактором используются следующие команды, находящиеся в меню “Extras”:

Delete list	Выбранный список удаляется из ПЛК
Write list	Открывается диалог, в котором нужно выбрать списки, которые необходимо записать в ПЛК. Запись производится после нажатия кнопки ОК. В зависимости от целевой платформы для перечислений записываются только численные либо дополнительно и символьные значения.
Read list	Из системы исполнения читаются списки параметров типа “Parameter” и помещаются в Менеджер параметров. Чтение списков 'Variables' будет производиться, только если оно явно поддержано в целевой системе.
Write values	Значения атрибута “Value” всех параметров списка записываются в систему исполнения. Для того чтобы записать значение только одного параметра, дважды кликните по соответствующему полю. Появится диалог “Write value”.
Write default values	В систему исполнения будут записаны значения, определенные в столбце “Default”.
Take over values	Текущие значения будут считаны из системы исполнения и записаны в столбец “Value”

Для переключения между десятичным и шестнадцатеричным представлением значений атрибутов используйте команду Format Dec/Hex.

Список параметров в загрузочном проекте

В зависимости от целевой платформы при создании загрузочного проекта список параметров также будет сохраняться в ПЛК.

6.11.7 Экспорт/импорт списков параметров

'Extras' 'Export'

Эта команда экспортирует все списки параметров, созданные в менеджере параметров в XML-файл. Этот файл может быть импортирован в другой проект с помощью функций импорта. Открывается стандартный диалог для сохранения файла с расширением *. prn.

Списки параметров также экспортируются при экспорте проекта в целом (“Project” “Export”).

'Extras' 'Import'

Эта команда используется для импорта списков параметров из XML-файла. Такой файл можно создать с помощью функций экспорта.

Если XML- файл содержит список параметров, который уже есть в Менеджере параметров, то появится диалог, в котором будет предложено перезаписать существующий список.

6.12 Настройка целевой платформы (Target Settings)

Объект "**Target Settings**" расположен на вкладке ресурсов (Resources) Организатора объектов. Здесь вы выбираете, с какой целевой (аппаратной) платформой должен использоваться текущий проект, и задаете настройки выбранной платформы. При создании нового проекта (командой '**Project**' '**New**') диалог выбора целевой платформы открывается автоматически. Выбор платформ ограничен числом установленных на вашем компьютере целевых пакетов (**Target Support Packages: TSP**). Выбор платформы определяет базовые параметры генератора кода и функциональность доступных в системе команд. Некоторые параметры целевой платформы можно изменять в диалоге **Target settings**.

Обратите внимание: Если ни один TSP не доступен, в списке выбора платформ присутствует единственный вариант: 'None'. Это вариант полной эмуляции ПЛК. Никакая его настройка не нужна.

Установка TSP

Необходимые вам TSP должны быть установлены до начала работы. Для этого предназначена утилита **InstallTarget**, включенная в состав комплекса CoDeSys. По умолчанию она автоматически устанавливается на компьютере вместе со средой программирования.

В TSP включены все файлы, необходимые CoDeSys для создания кода, отладки и конфигурирования аппаратуры. Платформа определяет параметры генератора кода, распределение памяти, функциональность ПЛК, модули ввода-вывода. Кроме того, в TSP могут входить дополнительные библиотеки, драйверы связи, ini-файлы сообщений об ошибках и список команд ПЛК-Браузера.

Центральным компонентом TSP является один или несколько целевых файлов (**Target files**). В нем присутствуют данные о всех дополнительных файлах, необходимых для конфигурирования данной платформы. По умолчанию целевой файл имеет расширение *.trg. Он записан в двоичном формате, редактировать его непосредственно нельзя. Для изменения необходимых настроек предназначен диалог **Target settings** в CoDeSys.

В процессе инсталляции каждый TSP устанавливается в отдельную директорию и соответствующий путь регистрируется. Все вспомогательные файлы копируются на жесткий диск. Их состав определен в информационном файле *.tnf. Имя директории определяется названием целевой системы. Рекомендуется создавать целевые директории внутри директорий, названных по имени изготовителя.

Имена установленных платформ определяются при запуске CoDeSys. Выбор платформы происходит в диалоге CoDeSys и сохраняется в проекте.

Обратите внимание: Если вы используете новый целевой файл или изменили существующий, необходимо перезагрузить CoDeSys для обновления его данных.

Диалог настроек целевой системы

Диалог настроек целевой системы (**Target Settings**) открывается автоматически при создании нового проекта. Кроме того, вы можете открыть его в любое время с помощью объекта 'Target Settings' на вкладке ресурсов ('Resources') в Организаторе объектов.

Выберите одну из целевых систем, предложенных в **Configuration**. Если вы выберете целевую систему, не имеющую лицензии на данном компьютере, CoDeSys предложит вам выбрать другой вариант.

Выбор целевой системы определяет возможности ее дополнительного конфигурирования. Определенные поля могут быть не доступны для изменения и показаны серым. Для некоторых целевых систем никакая настройка не предусматривается (Hide Settings). В целом вы можете настраивать:

1. Целевую платформу (Target Platform).

2. Распределение памяти (Memory Layout).
3. Общие параметры (General).
4. Сеть (Networkfunctionality).
5. Визуализацию (Visualization).

Внимание: Будьте осторожны, изменение настроек целевой системы влияет на производительность и работоспособность контроллера!

Нажмите кнопку <Default>, если вы хотите восстановить исходные настройки целевой системы.

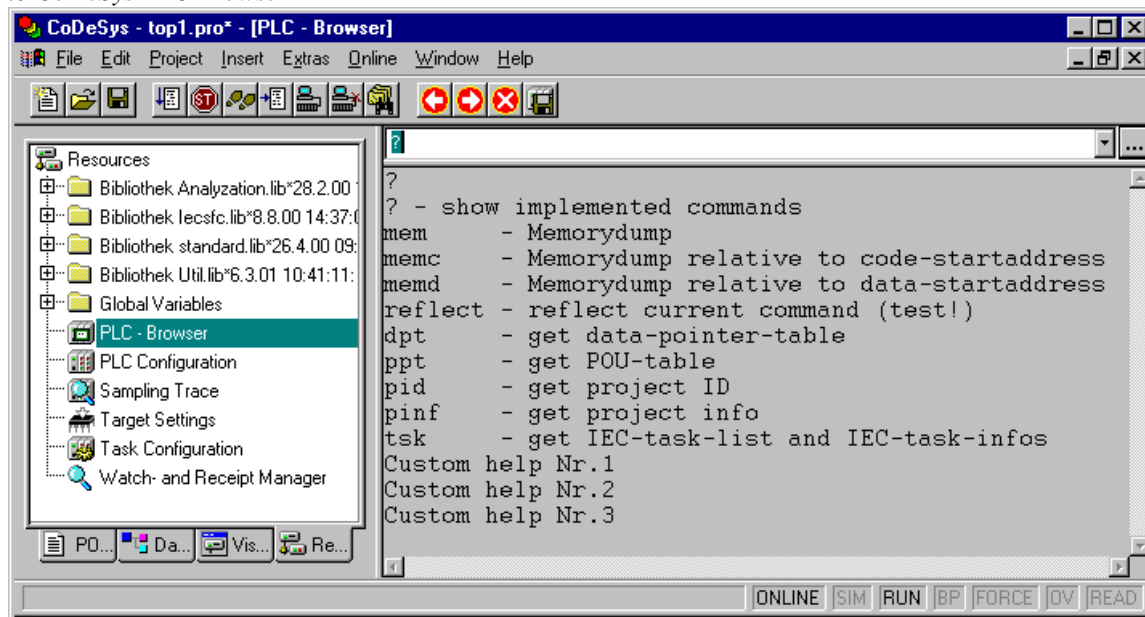
6.13 ПЛК-Браузер (PLC-Browser)

6.13.1 Общие сведения

ПЛК-Браузер - это текстовый монитор (терминал). Команды вводятся в виде текстовых строк и передаются в ПЛК. Ответом может быть запрошенная информация или отчет о результатах выполнения команды. Данный сервис предназначен для диагностики ПЛК и отладки. В CoDeSys предусмотрен определенный набор команд, но он может быть изменен или расширен изготовителем ПЛК. Состав команд задается в ini файле, связанном с целевой системой.

PLC-Browser присутствует на вкладке ресурсов (Resources) Организатора объектов, если он активирован в настройках целевой платформы (категория General).

Окно *CoDeSys PLC Browser*




Окно ПЛК-Браузер состоит из строки команд и окна отображения результатов. Выпадающий список в строке команд содержит все ранее введенные команды со времени запуска проекта. Это упрощает их повторение. Новые команды автоматически добавляются в данный список.

По нажатию клавиши <Enter> команда передается в контроллер. Если Online соединение не установлено, команда отображается в том виде, в котором она передается в контроллер. В режиме Online в окне отображается ответ контроллера.

6.13.2 Набор команд ПЛК-Браузера

Чаще всего ПЛК-Браузер выполняет **стандартный набор команд 3S**, поддержанный в системе исполнения. Он включает функции манипулирования памятью и информационные функции системы исполнения. Список доступных команд задан в ini файле, включенном в состав пакета целевой платформы. Для расширения состава команд необходимо включить их реализацию в систему исполнения и описать в ini файле.

При открытии проекта список доступных команд ПЛК-Браузера считывается из ini файла. Доступ к этим командам можно получить кнопкой  через диалог „**Insert standard command**” или клавишей <F2>. Дать команду можно и через меню 'Insert' 'Standard commands'. Кроме того, команду можно ввести вручную.

Синтаксис команд:

<команда><пробел><параметры>

Список параметров определяется типом команды. Переданная команда повторяется в окне отображения вместе с ответом контроллера.

Пример: Запрос кода идентификатора проекта (Id) командой "pid":

```
pid.....
```

Вывод результата:

```
pid
```

```
Project-ID: 16#0025CFDA
```

Примечание: Для каждой команды можно запросить текст подсказки: ?<пробел><команда>. Текст подсказки находится в ini файле. В контроллер эта команда ничего не передает.

Список стандартных 3S команд PLC-Browser:

Команда	Описание
?	Запрос у системы исполнения актуального списка всех поддерживаемых команд. Данный список не зависит от описаний, включенных в файлы целевой системы.
mem	Hex дамп области памяти Синтаксис 1: mem <start address> <end address> Синтаксис 2: mem <start address>-<end address> Адрес вводится в виде десятичного, шестнадцатеричного числа (префикс 16#) или макро.
memc	Относительный Hex дамп области кода; аналогична mem, адрес задается от начала области кода
memd	Относительный Hex дамп области данных; аналогична mem, адрес задается от начала области данных

reflect	Возврат строки (для тестирования)
dpt	Чтение таблицы указателей данных
ppt	Чтение таблицы POU
pid	Чтение Id проекта
pinf	Чтение информации о проекте
tsk	Показать список ИЕС задач
startprg	Запуск ПЛК программы
stopprg	Останов ПЛК программы
resetprg	Сброс ПЛК программы. Инициализируются только не энергонезависимые переменные.
resetprgcold	Холодный сброс ПЛК программы. Инициализируются в том числе энергонезависимые переменные
resetprgorg	Заводской сброс ПЛК программы. Полная очистка областей кода и данных..
reload	Перезапись загрузочного кода проекта
getprgprop	Свойства программы
getprgstat	Статус программы
filedir	Файловая команда "dir"
filecopy	Копирование файла [from] [to]
filerename	Переименование файла [old] [new]
filedelete	Удаление файла [filename]
saveretain	Запись сохраняемых (retain) переменных
restoreretain	Чтение сохраняемых (retain) переменных
setpwd	Установить пароль на контроллер Синтаксис: setpwd <password> [level] <level> может быть "0" (по умолчанию) действительный для подключения системы программирования или "1" действительный для всех приложений
delpwd	Удалить пароль

Обратите внимание:

ÿ Первое введенное в строке слово воспринимается как ключевое (<KEYWORD>).

ÿ Если ключевое слово не распознано контроллером, в окне результата выводится сообщение 'Keyword not found'.

ÿ Если перед ключевым словом стоит знак вопроса и пробел (например „? mem"), то выполняется поиск и отображение соответствующей подсказки из ini-файла. В контроллер при этом ничего не передается.

6.13.3 Макрорасширения команд ПЛК-Браузера

Если в строке команд введена команда с макрорасширением, оно будет раскрыто до передачи команды контроллеру. Результат выполнения дается в «раскрытом» виде.

Синтаксис: <команда><макрос>

Макросы:.

%P<NAME> Если NAME имя POU, то макрос раскрывается в индекс <POU-Index>, в противном случае текст не изменяется

%V<NAME> Если NAME имя переменной, то макрос раскрывается в #<INDEX>:<OFFSET>, в противном случае текст не изменяется (нотация #<INDEX>:<OFFSET> интерпретируется контроллером как адрес памяти)

%T<NAME> Если NAME имя переменной, то макрос раскрывается в <VARIABLENTYP>, в противном случае текст не изменяется

%S<NAME> Если NAME имя переменной, то макрос раскрывается в <SIZEOF(VAR)>, в противном случае текст не изменяется

Следующий за обратной косой чертой \ символ % игнорируется. Если необходимо передать символ ко-сой черты, введите: \\.

Пример:

Ввод в строке команды (дамп памяти для переменной .testit):

```
mem %V.testit
```

Вывод результата:

```
mem #4:52
```


```
03BAAA24 00 00 00 00 CD CD CD CD ....
```

6.13.4 Вспомогательные команды ПЛК-Браузера

В меню 'Extras' и панели команд PLC-Browser присутствуют вспомогательные команды для ускорения ввода и просмотра истории:

Кнопки просмотра истории Вперед (**History forward**)  и Назад (**History backward**)  дают возможность «прокрутить» результаты выполненных команд. Запись истории сохраняется до закрытия проекта.

Команда **Cancel**  прерывает начатый запрос.

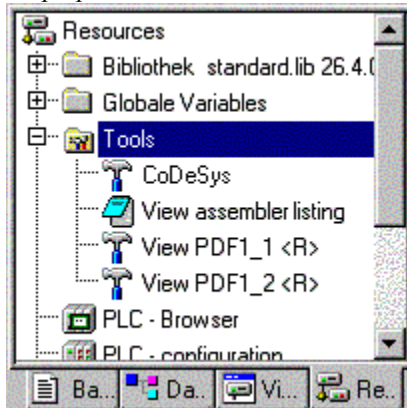
Команда **Save history list**  сохраняет результаты выполненных команд в файле с расширением *.bhl. (**B**rowser **H**istory **L**ist). Команда **Print last command** открывает стандартный диалог печати. На печать будет выведен текущий запрос и его результат.

6.14 Инструменты (Tools)

Объект 'Tools' (инструменты) расположен на вкладке ресурсов (Resources) Организатора объектов, если он разрешен для выбранной целевой системы. Для каждого внешнего инструмента (исполняемый файл)

определен набор команд (shortcuts), которые можно вызывать из CoDeSys двойным щелчком мыши. В качестве внешних инструментов могут служить общедоступные приложения, установленные на данном компьютере (например, Acrobat Reader или notepad.exe) плюс специфический набор инструментов для конкретной целевой платформы. Команда - это вызов инструмента с определенными параметрами. Пользователь может самостоятельно добавлять необходимые ему команды в папку 'Tools' .

Пример представления папки Tools



В данном примере набор инструментов содержит 4 команды. Первая используется для запуска еще одной среды программирования CoDeSys. Вторая открывает редактор с листингом ассемблерного кода. Две последних открывают PDF-файлы. Команды, содержащие в названии "<R>" нельзя изменять в CoDeSys. Такие команды могут вызывать, например, определенный текстовый редактор с конкретным документом или конкретный PDF-файл.

Кроме того, активация команды в разделе инструментов может приводить к загрузке в ПЛК определенных наборов файлов.

6.14.1 Свойства доступных инструментов (*Object Properties*)

Чтобы раскрыть объект Инструменты (Tools), расположенный на вкладке ресурсов Организатора объектов, необходимо щелкнуть мышкой по значку "плюс". Если вы только что создали новый проект, то вы увидите только инструменты определенные в целевом файле. В процессе работы необходимые для инструментов команды можно будет добавить непосредственно в CoDeSys.

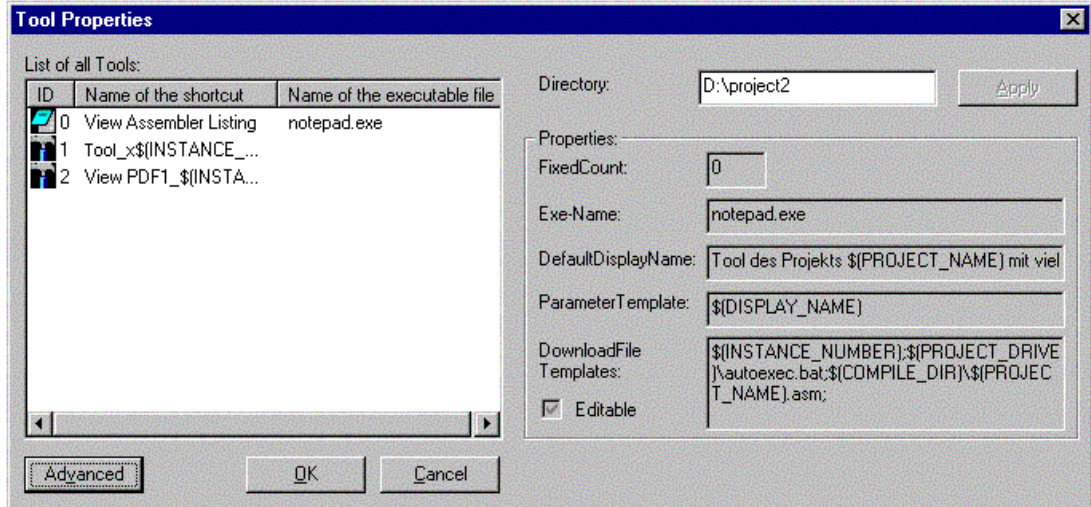
Вы можете наблюдать как глобальные свойства инструмента, так и индивидуальные свойства команд.

1. Свойства инструментов (Tool Properties):

Если папка 'Tools' выбрана в дереве ресурсов, то в контекстном меню будет присутствовать команда 'Object Properties'. Она открывает диалог свойств инструмента (Tool Properties).

В нем вы найдете актуальный список внешних инструментов. Диалог включает следующие параметры: **Id** – уникальный идентификационный номер, **Name of the shortcut** – наименование инструмента (команды) для его отображения в Организаторе объектов, **Name of the executable file** – имя исполняемого файла. Кнопка **Advanced** раскрывает и свертывает дополнительные сведения.

Диалог Свойства инструментов:



Открытый диалог показывает глобальные свойства внешнего инструмента, определенные в файле целевой системы. В поле **Directory** указывается имя рабочей директории, которая будет использоваться данным инструментом. Для изменения рабочей директории инструмента достаточно нажать кнопку **Apply**, не закрывая диалог.

Свойства инструментов:

FixedCount: Число команд для данного инструмента автоматически добавляемых в папку Tools. Если задано число "0", то пользователь может добавлять произвольное число команд.

Обратите внимание: на постоянные команды, определенные как "fix" в целевом файле, число позиций не распространяется и не может быть изменено пользователем CoDeSys. Такие команды отмечены символом "<R>" в Организаторе объектов.

Exe-Name: Имя файла или полный путь исполняемого файла инструмента. Здесь же можно задать определение инструмента через реестр: "[путь в реестре] <элемент, содержащий указание на исполняемый exe-файл>". Если исполняемый файл не задан, то для команд будут применяться зарегистрированные Windows инструменты. Выбор инструмента будет определяться расширением файла.

Примеры: "C:\program\notepad.exe", "345.pdf"

DefaultDisplayName: Имя для отображения инструмента в Организаторе объектов. Здесь может быть использован шаблон \$(INSTANCE NUMBER) (см. see ниже 'Parameter Template').

Parameter Template: Шаблоны параметров определяют файл, открываемый данным инструментом:
 \$(PROJECT_NAME) имя текущего открытого проекта (имя файла без расширения *.pro").
 \$(PROJECT_PATH) имя директории текущего открытого проекта (без диска).
 \$(PROJECT_DRIVE) диск текущего открытого проекта.
 \$(COMPILE_DIR) полное имя директории текущего открытого проекта (включая диск).

\$(TOOL_EXE_NAME) имя exe-файла инструмента.

\$(DISPLAY_NAME) имя текущей команды, отображаемое в папке 'Tools'.

\$(INSTANCE_NUMBER) порядковый номер команды (начиная с "1")

\$(CODESYS_EXE_DIR) полный путь к директории, где расположен Codesys exe-файл (включая диск).

Раскрытый шаблон вы увидите в диалоге свойств команды Shortcut Properties (см. ниже)

Пример:

"\$(PROJECT_NAME)_\$(INSTANCE_NUMBER).cfg"

Будет открыт cfg-файл, имя которого образованно как <имя текущего проекта CoDeSys>_<номер команды>.cfg.

DownloadFile Templates:

Файлы, определенные по имени или через шаблоны, которые будут скопированы в ПЛК при загрузке. Если активна опция **Editable**, то список будет доступен для редактирования в диалоге свойств инструмента. Если путь не указан, поиск файлов будет идти в директории, где расположен codesys-exe файл.

Пример:

"a.up;\$(PROJECT_NAME).zaw;\$(INSTANCE_NUMBER).upp"

Три файла: a.up, <current CoDeSys Projekt>.pro и <shortcut number>.upp будут скопированы в ПЛК при очередной загрузке.

2. Свойства команд (Shortcut Properties):

Выберете необходимое имя команды в дереве 'Tools' Организатора объектов и дайте команду 'Object Properties' из контекстного меню или меню 'Project' 'Object'. Диалог свойств команды (Shortcut Properties) содержит следующие поля:

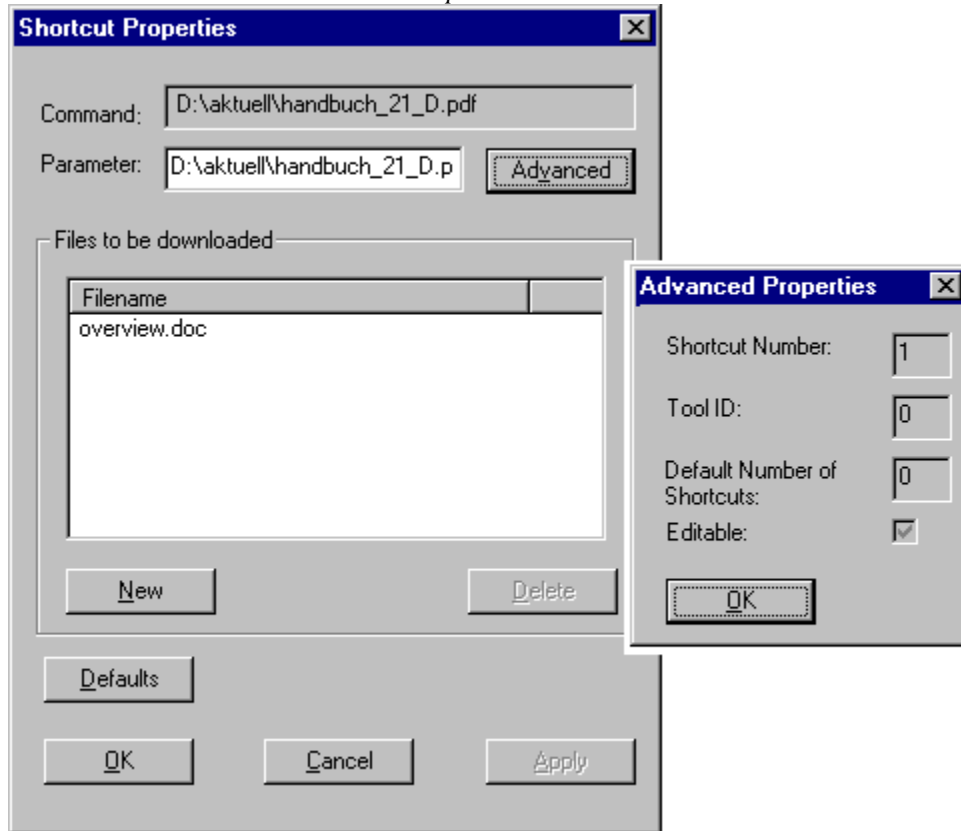
Command	Полная командная строка вызова инструмента; исполняемый файл и передаваемый инструменту параметр (обрабатываемый файл) команды (определенный в поле 'Parameter Template', см. выше) T.e.: C:\programs\notepad.exe D:\listings\textfile.txt
Parameter	Параметр (файл, вызываемый данным инструментом). Задается в целевом файле и может редактироваться здесь, если активна опция 'Editable'.
Files to be downloaded	Изначально здесь перечислены файлы, определенные в целевом файле (target) и в поле свойств инструмента DownloadFileTemplate (см. выше). Если активна опция 'Editable', то в дополнительном диалоге (см. Ниже) этот список можно изменить. Для этого нажмите кнопку New. Задайте имя файла в диалоге 'Filename'. Если вы задали имя без указания пути, поиск файла будет производиться в директории, где расположен codesys-exe файл. Кнопка Delete удаляет элемент списка.

Кнопка **Standard** устанавливает поля диалога по умолчанию, т.е. так, как определено в целевом файле.

Кнопка **Apply** позволяет применить изменения, не закрывая диалог.

Кнопка **Advanced** открывает дополнительный диалог 'Advanced Properties', как показано ниже:

Диалог свойств команды с 'Advanced Properties':



- Shortcut Number:** Порядковый номер, начиная с 1. Каждая команда для данного инструмента получает очередной номер. Если одна из команд будет удалена позднее, то номера остаются неизменными. Порядковый номер можно использовать путем применения шаблона: \$(INSTANCE_NUMBER) (см. 'Parameter Template').
- Tool ID:** Уникальный идентификационный номер инструмента, определенный в целевом файле.
- Default Number of Shortcuts:** Число команд для данного инструмента. Соотносится со значением в поле "FixedCount" целевого файла (См. выше Tool Properties).
- Editable:** Если данная опция активна, то существует возможность редактирования параметров и списка загружаемых файлов.

Кнопка **OK** указывает применить изменения и закрыть диалог.

6.14.2 Настройка команд инструментов

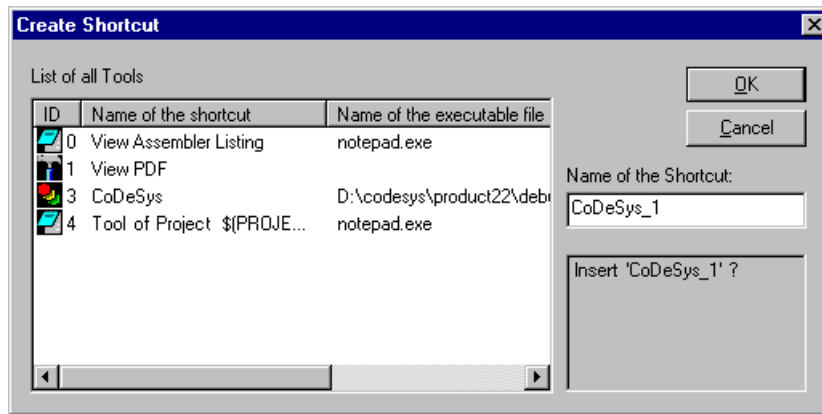
Создание новой команды инструмента

Выделите объект 'Tools' (Инструменты) или одну из уже существующих команд в Организаторе Объектов и дайте команду 'Add Object' из контекстного меню или меню 'Project' 'Object'. Тем самым будет открыт диалог '**Create Shortcut**', показанный ниже.

В таблице показаны все инструменты, для которых можно добавить новые команды. В соответствии с определениями в целевом файле отображаются: **ID** идентификационный номер инструмента, **Name of the shortcut** имя для команды по умолчанию, и имя исполняемого файла **Name of the executable file**.

Выберите инструмент, для которого вы хотите добавить команду и щелкните мышкой по его идентификационному номеру в таблице (ID). Вслед за этим вы можете изменить имя команды, предложенное по умолчанию (**Name of the shortcut**). Одинаковые имена команд использовать нельзя.

Диалог 'Create Shortcut'



Завершите определение кнопкой ОК. Новая команда будет добавлена в дерево ресурсов. Связь отображается по имени, в соответствии с порядковым номером на единицу большим, чем номер последней существовавшей для данного инструмента команды.

Под полем ввода имени команды показана область подсказки, соответствующая выполняемым действиям.

Удаление команды инструмента

Выделите имя необходимой команды и дайте команду 'Delete' из контекстного меню или меню 'Project' 'Object'. Идентификационные номера оставшихся команд не изменяются. Фиксированные команды удалить нельзя.

Выполнение команд

Для выполнения команды щелкните дважды мышкой по соответствующему пункту (названию команды) в Организаторе объектов или дайте команду 'Open Object' из контекстного меню (правая клавиша мыши).

Если произойдет ошибка при открытии заданного в параметрах файла, то будет дано соответствующее сообщение об ошибке. Если параметр не найден, то при запуске exe-файла инструмента будет предложено создать новый файл.

Если исполняемый exe-файл не найден то, будет открыт стандартный диалог для поиска и выбора исполняемого файла. По нажатию ОК указанный путь будет сохранен и доступен в других CoDeSys проектах.

Сохранение набора команд

При сохранении проекта CoDeSys текущие команды папки 'Tools' сохраняются автоматически.

Обратите внимание: Если вы сохраняете проект командой 'Save as' под другим именем, то вы должны учитывать, что при использовании шаблона \$(PROJECT_NAME) для файла параметра инструмента и в загружаемых файлах будет использоваться новое имя. В противном случае изменить имена нужно будет вручную.

6.14.3 Часто задаваемые вопросы по инструментам

Почему я не вижу папки 'Tools' в ресурсах?

Данная папка присутствует, только если это разрешено в целевом файле.

Какие внешне инструменты определены изначально, какие дополнительные команды я могу определить в проекте CoDeSys?

Откройте папку 'Tools' на вкладке 'Resources' Организатора объектов. Щелкните дважды мышкой на значке "плюс". Вы увидите инструменты, подключенные к данному проекту. Если вы только что начали работать с новым проектом и не вносили изменений в инструменты, то здесь будут перечислены только инструменты, определенные в целевом файле. Иначе вы можете увидеть специально настроенный под данный проект набор команд. Чтобы узнать о возможности добавления новых элементов, дайте команду 'Add Object'. Вы увидите диалог, содержащий все инструменты, для которых можно создавать новые команды.

Какие глобальные свойства имеют доступные мне инструменты?

Выделите объект 'Tools' (Инструменты) в Организаторе Объектов и дайте команду 'Object Properties' из контекстного меню (правая кнопка мыши). Расширьте появившийся диалог, нажав кнопку 'Advanced'. Вы увидите список доступных инструментов и соответствующих параметров. Выберите один из инструментов щелчком мыши на символе идентификатора (ID), чтобы, например, посмотреть число доступных команд (shortcuts) для данного инструмента (поле 'Fixed Count') или список файлов которые будут загружены в ПЛК данной командой. Пути и имена файлов могут быть определены с помощью шаблонов. (См. выше “Свойства инструментов”).

Какие индивидуальные свойства имеют команды?

Выделите один из элементов дерева 'Tools' (Инструменты) в Организаторе Объектов и дайте команду 'Object Properties' из контекстного меню (правая кнопка мыши). Для доступа к параметрам выбранной команды нажмите кнопку 'Advanced'. Часть параметров команды определяется вышеописанными глобальными свойствами инструмента. Если это разрешено (в целевом файле), отдельные параметры можно редактировать, задавая им индивидуальные свойства.

Как добавить команду для инструмента?

Выделите объект 'Tools' (Инструменты) в Организаторе Объектов и дайте команду 'Add Object' из контекстного меню (правая кнопка мыши). Вы увидите список инструментов, но только тех, для которых не превышено максимально допустимое число команд. Выберите нужный инструмент и нажмите кнопку ОК. Теперь данный инструмент будет присутствовать в папке 'Tools' Организатора Объектов. Если вам необходимо добавить его вторично, измените его наименование. Например, для команд инструмента Toolxu можно определить "Toolxu_1", "Toolxu_2" и т.д.

Как изменить параметры инструмента?

Для изменения параметров команды (вызов инструмента с определенными параметрами) выберите необходимую команду в Организаторе Объектов и дайте команду 'Object Properties' из контекстного меню. В зависимости от определения данного инструмента в целевом файле для редактирования могут быть доступны разные параметры. Кнопка 'Standard' задает параметры по умолчанию.

Как выполнить необходимую команду внешнего инструмента?

Щелкните дважды мышкой по соответствующему пункту (названию команды) в Организаторе объектов или дайте команду 'Open Object' из контекстного меню для выделенного пункта.

7 ENI

7.1 Что такое ENI?

Инжиниринговый интерфейс ENI (Engineering Interface) позволяет соединять систему программирования CoDeSys с **внешней базой данных**. В ней сохраняются данные, необходимые в ходе проектирования и реализации практических задач автоматизации. Использование внешней базы данных гарантирует целостность данных, которые могут параллельно использоваться несколькими пользователями, проектами и программами. Кроме того, это расширяет функциональные возможности CoDeSys, делая возможным следующее:

- **Управление версиями:** CoDeSys проектов и связанных с ними ресурсов (разделяемые объекты). Если некоторый объект изъять из базы данных, изменить и снова записать (поместить) в базу, то в базе данных создается новая версия (копия) объекта. При этом все «старые» версии будут храниться в базе и могут быть восстановлены в любое время. Для каждого объекта и для целого проекта сохраняется история версий. Любые версии можно легко сравнить и найти различия.
- **Многопользовательская работа:** Самые последние версии объектов, например все POU проекта, могут быть доступны для группы пользователей. Объекты, изъятые в текущий момент одним из пользователей, будут отмечены как находящиеся "в работе" и не будут доступны для редактирования другими пользователями. Таким образом, несколько пользователей могут работать с одним и тем же проектом одновременно без риска нарушить его целостность.
- **Доступ со стороны внешних инструментов:** Помимо среды программирования CoDeSys, другие инструменты, имеющие ENI, могут использовать общую базу данных. Это могут быть внешние средства визуализации, ECAD и другие системы, которым необходим доступ к объектам базы данных (Подробнее см. отдельный документ *ENI Server*).

ENI состоит из двух частей: **клиента** и **сервера**. Таким образом, база данных может располагаться на удаленном компьютере, предоставляющем доступ нескольким пользователям одновременно. Среда программирования CoDeSys выступает в роли одного из независимых клиентов сервера наравне с другими приложениями, нуждающимися в доступе к данным.

В настоящее время ENI поддерживает базы данных 'Visual SourceSafe 6.0', 'MKS Source Integrity', 'PVCs Version Manager' V7.5 и старше, а также может использовать локальную файловую структуру операционной системы. Объекты хранятся в определенных «папках» (категории базы данных) с различными правами доступа. Объект может быть извлечен для редактирования одним из пользователей и будет временно не доступен другим пользователям. Кроме того, некоторые объекты можно хранить только локально в проекте, как в обычных проектах, не использующих контроль версий.

7.2 Условия работы с ENI базой данных в проекте

Обратите внимание: инструкции по установке и использованию *ENI сервера*, поставляемого 3S – Smart Software Solutions GmbH, вы найдете в отдельном документе и в оперативной подсказке. Ниже вы найдете краткое руководство. Кроме того, обратите внимание на *ENI Explorer*, позволяющий работать с объектами базы вне зависимости от конкретной установленной базы данных.

Для использования **ENI** в среде программирования CoDeSys с целью управления объектами проекта, размещенными во внешней базе данных, необходимо:

- для взаимодействия CoDeSys ENI сервера необходима установка **TCP/IP**, поскольку *ENI сервер* использует протокол HTTP.

- *ENI сервер (ENI Server Suite)* должен быть установлен и запущен на удаленном компьютере. Для работы с одним из стандартных драйверов баз данных необходима **лицензия**. Только драйвер локальной файловой системы можно использовать без лицензии.
- С помощью инструмента настройки *ENI сервера (ENI Control)* должно быть корректно определено подключение к базе данных (Data base). Необходимые параметры указываются при установке, но их можно изменить в любое время через *ENI Control*.
- **База данных проекта** (для которой есть соответствующий драйвер) должна быть установлена. Разумно размещать ее на том же компьютере, что и *ENI сервер*. В качестве альтернативы можно использовать локальную файловую систему. Драйвер для этого варианта устанавливается по умолчанию.
- **При администрировании** базы данных может потребоваться разрешить доступ пользователей и самого *ENI сервера*. Для 'Visual SourceSafe' это обязательно. Для других баз данных изучите соответствующую документацию по настройке пользовательской конфигурации.
- Для текущего CoDeSys проекта должен быть активирован **ENI интерфейс** (это делается в диалоге 'Project' 'Options' 'Project data base'). Возможно, потребуется некоторая настройка ENI, например, для более детального описания прав доступа. Но, как правило, достаточно, чтобы пользователь имел право доступа к базе данных.
- Для текущего CoDeSys проекта должно быть корректно определено соединение с базой данных. (Это делается в диалоге 'Project' 'Options' 'Project source control').
- В текущем проекте необходимо установить связь с сервером (**log in to the ENI Server**) с определенным именем пользователя и пароля. Это делается в диалоге Login который открывается командой 'Project' 'Data Base Link' 'Login' либо автоматически при попытке доступа к базе данных.

7.3 Работа с ENI базой данных в проекте CoDeSys

Если соединение с базой данных настроено и установлено (См. 'Условия работы с ENI базой данных в проекте'), то в проекте будут доступны соответствующие команды: Get Latest Version, Check Out, Check In, Version History, Label Version и т.д. Команды расположены в подменю 'Data Base Link' и применяются к объекту, выделенному в Организаторе объектов.

Принадлежность объекта к категории базы данных отображается в свойствах объекта (Object Properties) и может быть изменена здесь же.

Свойства категорий базы данных (коммуникационные параметры, особенности извлечения и записи) определяются в диалоге опций базы данных проекта ('Project' 'Options' 'Project Source Control').

7.4 Категории объектов в базе данных проекта

Существуют 4 категории объектов CoDeSys имеющие отношение к контролю версий:

- В базе данных ENI поддерживаются три различных категории ("категории объектов ENI"): Объекты проекта (Project objects), Разделяемые объекты (Shared objects), Файлы компилятора (Compile files).
- Локальные объекты, сопоставленные категории 'Local', не хранятся в базе данных. Они обслуживаются так же, как объекты проекта, не использующего управление версиями.

Таким образом, объект в системе программирования CoDeSys может быть отнесен к одной из категорий: 'Project objects', 'Shared objects' или 'Local'. Естественно, категорию 'Compile files' для компонентов проекта использовать нельзя. Распределение объектов по категориям выполняется автоматически, при создании объекта, в соответствии с опциями диалога 'Project source control', либо явно командой 'Project' 'Data Base Link' 'Define' или 'Multiple Define'. Существующее распределение можно в любое время изменить в диалоге свойств объекта (Object Properties).

Каждая категория ENI конфигурируется в отдельном диалоге 'Project source control' опций проекта ('Project' 'Options'). Это означает, что каждая категория получает собственные параметры связи с базой данных (директория, порт, права доступа и др.) и определенные свойства хранения и извлечения последних версий объектов. Заданные настройки применяются для всех объектов данной категории. Желательно (но не обязательно) хранить объекты разных категорий в разных папках базы данных. Категория - это свойство объекта, а не часть базы данных.

Отличия трех категорий ENI объектов заключены в следующем:

- Project Objects:** Объекты, значимые (специфичные) для данного проекта, например POU, нуждающиеся в многопользовательских операциях. Команда 'Get all latest versions' автоматически вызывает все объекты данной категории из базы данных в локальный проект, даже новые, отсутствующие в текущем проекте.
- Shared Objects:** Объекты, не несущие специфики отдельного проекта, например библиотечные POU, используемые в нескольких проектах.
Внимание: Команда 'Get all Latest Versions' копирует только существующие в текущем проекте объекты этой категории из базы данных в локальный проект.
- Compile files:** Выходная информация компилятора (т.е. символьные файлы) CoDeSys по данному проекту, которая может быть необходима другим приложениям. Например: для программы внешней визуализации могут потребоваться не только символьные имена, но и адреса переменных, не известные до компиляции.

Любые объекты CoDeSys проекта могут быть исключены из механизма контроля версий и сопоставлены категории 'Local'. Это означает, что они сохраняются только локально в файле проекта, как в проектах, не использующих ENI.

8 DDE интерфейс

Интеграция приложений с CoDeSys посредством DDE

CoDeSys обеспечивает передачу значений переменных другим Windows приложениям, посредством механизма динамического обмена данными (DDE).

Если используется GatewayDDEServer, то для чтения значений из ПЛК и передачи в другие приложения, CoDeSys не нужен.

ВНИМАНИЕ: Прямые адреса нельзя читать через DDE сервер. Используйте для этого переменные, присвоенные прямым адресам.

DDE интерфейс протестирован с Word 97 и Excel 97 под Windows NT 4.0. Если DDE обмен не работает с другими версиями этих программ или с другими программами, фирма 3S— Smart Software Solutions не несет за это ответственность.

8.1 DDE интерфейс CoDeSys

Активизация DDE

Интерфейс DDE автоматически активизируется, как только установлено соединение с ПЛК или режим эмуляции.

Общие принципы

DDE запрос можно разделить на 3 части:

1. Имя приложения (**CoDeSys**).
2. Имя файла.
3. Имя читаемой переменной.

Имя приложения: **CoDeSys**.

Имя файла: полное имя проекта в вашей системе (c:\example\example.pro).

Имя переменной: имя переменной, заданное так, как оно указывается в Watch and Receipt Manager .

Какие переменные будут читаться?

Все адреса и переменные доступны. Например:

%IX1.4.1	(* Вход 1.4.1*)
PLC_PRG.TEST	(* Переменная TEST из программы PLC_PRG*)
.GlobVar1	(* Глобальная переменная GlobVar1 *)

Соединение с WORD

Чтобы получить значение переменной TEST из POU PLC_PRG через DDE в Microsoft WORD, вставьте в текст поле ("Вставка", "Поле,,,").

Код поля должен быть следующим:

```
{DDEAUTO CODESYS "C:\CODESYS\PROJECT\IFMBSP.PRO" "PLC_PRG.TEST" }
```

Проект должен быть загружен и работать в режиме Online. Для активизации поля, дайте в Word команду "Обновить поле".

Примечание переводчика: Word , начиная с версии 6,0 не поддерживает вставку поля DDEAUTO. Тем не менее, механизм DDE работает во всех версиях. Вставьте в текст любое поле (например, DATE). Далее включите отображение кодов полей в тексте (щелкнуть по полю правой клавишей мыши и выбрать "Коды/Значения полей"). Впишите между скобок { } код поля, как описано выше. Вернитесь в режим отображения значений и обновите поле.

Соединение с EXCEL

Введите в ячейку EXCEL, составленную согласно образцу, формулу

```
=CODESYS!C:\CODESYS\PROJECT\IFMBSP.PRO!PLC_PRG.TEST'
```

Далее в меню "Правка", "Связи" вы получите соответствующую информацию:

Исходный файл: C:\CODESYS\PROJECT\IFMBSP.PRO:

Элемент: PLC_PRG.TEST

Тип: CODESYS

Обновление автоматическое.

Соединение с Intouch

Свяжите свой проект с DDE Access Name <AccessName>, где имя приложения CoDeSys и тема DDE (topic name) C:\CODESYS\PROJECT\IFMBSP.PRO

Теперь вы можете сопоставить переменные DDE с <AccessName>. Введите имена переменных как Item Name (например, PLC_PRG.TEST).

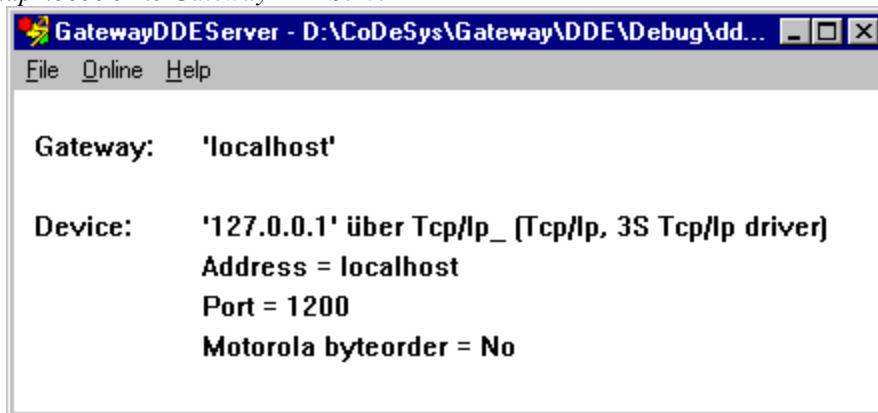
8.2 DDE обмен посредством GatewayDDE Server

Оперирование с сервером GatewayDDE

GatewayDDE сервер использует символы, созданные в CoDeSys для взаимодействия с другими приложениями (см. 'Project' 'Options' 'Symbolconfiguration').

При старте GatewayDDE сервер открывает окно, где выполняется конфигурация. Вы можете задать новые настройки или открыть готовый файл конфигурации.

Стартовое окно GatewayDDE Server



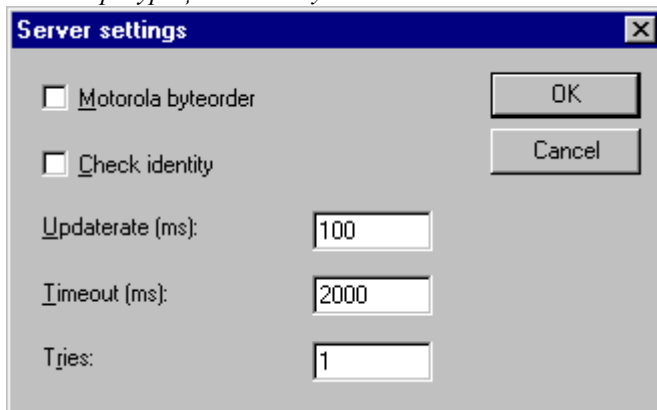
Команда **'File' 'Open'** открывает стандартное окно открытия файла. Файлы конфигурации имеют расширение “.cfg”. После открытия файла конфигурационные параметры и определенная аппаратная платформа будут показаны в окне.

Если опция **'File' 'Autoload'** активирована, то при старте сервера автоматически загружается конфигурация, установленная при отключении.

Если сервер запускается без predetermined configuration, вам необходимо будет настроить новую конфигурацию.

Команда **'File' 'Settings'** открывает диалог **'Server settings'**, в котором устанавливаются следующие параметры:

Диалог конфигурации GatewayDDE Server



Motorola byteorder: порядок байт Motorola

Check identity: проверка соответствия идентификатора проекта (ID) в ПЛК и в символьном файле.

Updaterate [ms]: период чтения всех символьных переменных из ПЛК.

Timeout [ms]: таймаут используемого драйвера.

Tries: число повторных попыток передачи блока данных (поддерживается не всеми драйверами !)

Для подключения к серверу связи нужно открыть диалог настройки **'Communication Parameters'** командой **'Online' 'Parameters'**. Настройка выполняется так же, как и в CoDeSys.

Для сохранения конфигурации используйте команду **'File' 'Save'**

Для подключения к контроллеру выполните команду '**Online**' '**Login**'. Когда соединение установлено, сервер готов к DDE соединению. Символьный файл должен быть заранее подготовлен в CoDeSys..

Команда '**Online**' '**Login**' отключает соединение.

Подключение DDE клиентов

Методология подключения приложений к DDE серверу аналогична описанной выше для CoDeSys.

Имя программы: GATEWAYDDESERVER

Для Word код поля может выглядеть, например, так:

```
{DDEAUTO GATEWAYDDESERVER "BSP.PRO" "PLC_PRG.TEST"}
```

В EXCEL доступ к той же самой переменной TEST будет выглядеть так:

```
=GATEWAYDDESERVER!'bsp.pro'!'PLC_PRG.TEST'
```

Опции командной строки GatewayDDE Server

При запуске GatewayDDE сервера в командной строке можно задать опции:

/n	Не показывать информационное окно при старте		
/s	Запуск в окне	/s=h	нет
		/s=i	минимизированное
		/s=m	максимальное
		/s=n	нормальное
/c	Автоматическая загрузка конфигурации	/c=<config-file>	
/o	Переход в режим online (конфигурация задана 1с)		

Пример:

```
GATEWAYDDE /s=i /c="D:\DDE\conf_1.cfg"
```

Сервер будет запущен в минимизированном окне, конфигурация загружается из файла conf_1.cfg.

9 Менеджер лицензирования CoDeSys

9.1 Обзор

Менеджер лицензирования 3S предназначен для лицензирования модулей 3S, а также любых других модулей, для которых соответствующий файл информации о лицензии присутствует на вашем компьютере. В CoDeSys вы можете создать залицензированную библиотеку на основе любого проекта. Менеджер лицензирования будет установлен автоматически с любым модулем 3S, требующим лицензии.

См. также: отдельный документ 3S Licensing Manager и “Создание лицензированных библиотек” ниже.

9.2 Создание лицензированных библиотек в CoDeSys

В CoDeSys вы можете создавать библиотеку на основе любого проекта. Для библиотек, требующих лицензии, необходимо добавить соответствующую лицензионную информацию (license information). Для этого дайте команду 'File'>'Save as...', выберите внутреннюю 'Internal Library' или внешнюю 'External Library' и нажмите кнопку **Edit license info...**. В диалоге **Edit Licensing Information** введите необходимые данные, как описано ниже. Информация о лицензировании будет добавлена в проект (Project Info). Если позднее данная библиотека будет добавлена в проект CoDeSys, то соответствующие данные будут проверены в диалоге Менеджера библиотек.

Диалог: Edit Licensing Information

General information:	
Name:	Library2_for_RT
Vendor ID:	44g2xw
<input checked="" type="checkbox"/> License free mode:	30 days
<input checked="" type="checkbox"/> Targets:	42

Contact:	
Licensing via phone:	0831/540-310
Licensing via mail:	licm@xyz.com

Optional information:	
Description	Library2 for RT
Manufacturer	
Vendor	
Pricing information	

- **General information** (общие сведения):

Name: введите имя модуля, так как оно должно отображаться в Менеджере лицензирования. Заполнение этих полей обязательно.

Vendor-ID: идентификатор поставщика, определяется специфическим инструментом лицензирования изготовителя.

License free mode: активируйте эту опцию, если данный модуль должен иметь возможность работы без лицензии, в демонстрационном режиме. Введите число дней (**days**), в течение которых допускается работа без лицензии. Число дней автоматически округляется вверх до десятков (10, 20, 30 ...). Если модуль можно использовать без ограничения по времени, то укажите здесь слово “unlimited”, доступное в списке.

Targets: введите идентификационные номера (ID) целевых систем, для которых действительна лицензия. Номера должны быть перечислены через запятую или точку с запятой.

- **Contact** (контактная информация):

Licensing via phone: / Licensing per via mail: номер телефона и e-mail поставщика лицензии. Заполнение этих полей обязательно.

- **Optional information** (дополнительная информация):

В правом окне, вы можете задать дополнительные сведения по пунктам списка, перечисленным в левой части окна: Description (описание), Manufacturer (изготовитель), Vendor (поставщик), Pricing information (цены).

Обратите внимание: Если вы требуете лицензирования библиотеки, есть смысл защитить ее от модификации паролем.

Лицензионная информация 3S сохраняется внутри файлов библиотек и регистрируется на компьютере автоматически, при включении библиотеки в проект. Лицензионная информация для модулей других поставщиков должна быть записана в отдельном XML файле. Подробнее см. документ **3S License Manager**.

10 Приложения

Приложение А: Операторы и функции МЭК

CoDeSys поддерживает все МЭК операторы. В отличие от стандартных функций (см. приложение D, Стандартная библиотека) данные операторы и функции не требуют подключения библиотек. Сверх требований МЭК CoDeSys поддерживает следующие дополнительные операторы, не включенные в стандарт: INDEXOF и SIZEOF (см. Арифметические операторы), ADR и BITADR (см. Адресные операторы).

Обратите внимание, что операторы могут принимать неявную форму. Так оператор сложения (ADD) в языке ST выражается знаком «+».

Внимание: поддержка выполнения операций с плавающей запятой зависит от целевой платформы.

- Арифметические операторы
- Битовые операторы
- Операторы сдвига
- Операторы выборки
- Операторы сравнения
- Адресные операторы
- Оператор вызова
- Преобразования типов
- Математические функции

10.1 Арифметические операторы

ADD

Сложение переменных типов: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL и LREAL.

Две переменных типа TIME можно складывать (напр. t#45s + t#50s = t#1m35s). Результат имеет тип TIME.

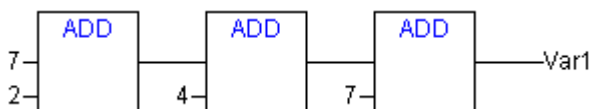
Пример IL:

```
LD    7
ADD   2,4,7
ST    Var 1
```

Пример ST:

```
var1 := 7+2+4+7;
```

Пример FBD:



MUL

Перемножение значений переменных типов: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL и LREAL.

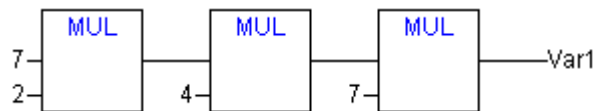
Пример IL:

```
LD 7
MUL 2,4,7
ST Var 1
```

Пример ST:

```
var1 := 7*2*4*7;
```

Пример FBD:

**SUB**

Вычитание значений переменных типов: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL и LREAL.

Переменной TIME можно присвоить результат вычитания двух других переменных типа TIME. Отрицательное время не определено.

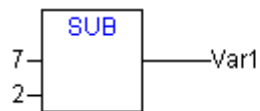
Пример IL:

```
LD 7
SUB 2
ST Var 1
```

Пример ST:

```
var1 := 7-2;
```

Пример FBD:

**DIV**

Деление значений переменных типов: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL и LREAL.

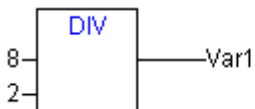
Пример IL:

```
LD 8
DIV 2
ST Var 1 (* Результат равен 4 *)
```

Пример ST:

```
var1 := 8/2;
```

Пример FBD:



Примечание: Определив в своем проекте функции с именами CheckDivByte, CheckDivWord, CheckDivDWord и CheckDivReal вы сможете контролировать делитель и обрабатывать, например, деление на 0.

Внимание: Результат деления на 0 может отличаться на разных целевых платформах.

Рассмотрим простейший пример применения CheckDivReal.

Пример функции CheckDivReal:

```

FUNCTION CheckDivReal : REAL
VAR_INPUT
    divisor:REAL;
END_VAR

IF divisor = 0 THEN
    CheckDivReal := 1;
ELSE
    CheckDivReal:=divisor;
END_IF;
    
```

Оператор DIV использует выход функции CheckDivReal как делитель. В следующей программе этот прием предотвращает деление на 0, делитель (d) заменяется с 0 на 1. В итоге получается результат 799.

```

PROGRAM PLC_PRG
VAR
    erg:REAL;
    v1:REAL := 799;
    d:REAL;
END_VAR

    erg:= v1 / d;
    
```

Внимание: CheckDiv-функции, содержащиеся в библиотеке Check.Lib, представляют собой примеры их реализации. Прежде чем использовать эту библиотеку, убедитесь, что она работает так, как нужно в вашем случае, либо создайте собственные функции непосредственно в вашем проекте.

MOD

Остаток от деления значений переменных типов: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT. Результат всегда целое число.

Пример IL:

```

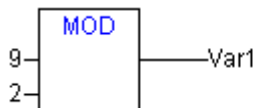
LD    9
    
```

```
MOD 2
ST Var 1 (* Результат 1 *)
```

Пример ST:

```
var1 := 9 MOD 2;
```

Пример FBD:

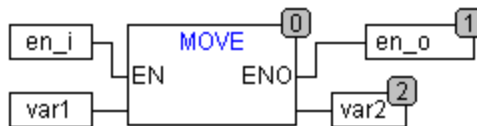


MOVE

Присвоение значения одной переменной другой соответствующего типа. В графических редакторах CFC и LD существует возможность управлять разрешением работы блока (разрешать или запрещать операцию) с помощью входов EN/ENO. В FBD этого делать нельзя.

Пример применения EN/ENO в CFC:

Только если значение en_i равно TRUE, значение переменной var1 будет присвоено var2.



Пример IL:

```
LD ivar1
MOVE ivar2
ST ivar2
```

(! Аналогичный результат дает:

```
LD ivar1
ST ivar2 )
```

Пример ST:

```
ivar2 := MOVE(ivar1);
```

(! Аналогичный результат дает: ivar2 := ivar1;)

10.2 Битовые операторы

AND

Побитное И. Операция применима к типам BOOL, BYTE, WORD или DWORD.

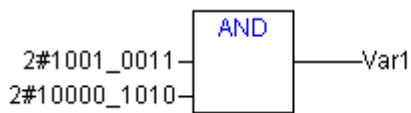
Пример IL:

```
Var1 BYTE
LD 2#1001_0011
AND 2#1000_1010
ST Var 1 (* Результат 2#1000_0010 *)
```

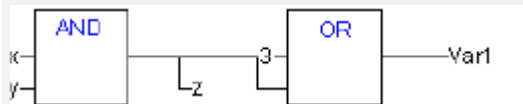
Пример ST:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

Пример FBD:



Внимание: В логических выражениях нельзя гарантировать присваивание промежуточных результатов. Например, если условие SFS перехода выглядит так:



то, значение переменной z может быть не присвоено. Это происходит по причине оптимизации вычислений компилятором. Если значения x и y FALSE, то конечный результат очевиден и остаток выражения вычислять не нужно.

OR

Побитное **ИЛИ**. Операция применима к типам BOOL, BYTE, WORD или DWORD.

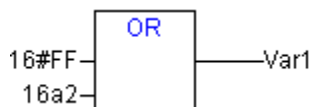
Пример IL:

```
var1 :BYTE;
LD   2#1001_0011
OR   2#1000_1010
ST   var1    (* Результат 2#1001_1011 *)
```

Пример ST:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

Пример FBD:



Внимание: См. примечание к AND.

XOR

Побитное исключающее **ИЛИ**. Операция применима к типам BOOL, BYTE, WORD или DWORD.

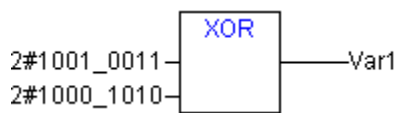
Пример IL:

```
Var1 :BYTE;
LD   2#1001_0011
XOR  2#1000_1010
ST   Var1    (* Результат 2#0001_1001 *)
```

Пример ST:

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

Пример FBD:



Внимание: Допускается расширяемая форма, т.е. XOR имеет более двух входов. В этом случае входы обрабатываются попарно, затем к результатам опять применяется XOR. Такой алгоритм определен стандартом.

NOT

Побитное **НЕ**. Операция применима к типам BOOL, BYTE, WORD или DWORD.

Пример IL:

```

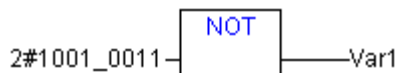
Var1 :BYTE;
LD   2#1001_0011
NOT
ST   Var1   (* Результат 2#0110_1100 *)

```

Пример ST:

```
Var1 := NOT 2#1001_0011
```

Пример FBD:



10.3 Операторы сдвига

Внимание: Количество бит, задействованных в данных операциях, определяется типом входной переменной! Тип переменной результата не влияет на процесс вычисления. (См. ST примеры ниже). Если входная переменная представлена константой, выбирается наиболее компактный из возможных типов данных.

SHL

`res:=SHL(in,n)` Побитный сдвиг операнда *in* влево на *n* бит с дополнением нулями справа.

Входные переменные и результат должны быть типа BYTE, WORD или DWORD.

В следующем примере представлены различные результаты *res_byte* и *res_word* в зависимости от типа входной переменной (BYTE и WORD), хотя числовые их значения равны.

Пример ST:

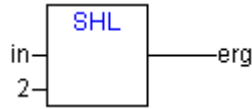
```

PROGRAM shl_st
VAR
  in_byte:BYTE:=16#45;
  in_word:WORD:=16#45;
  res_byte:BYTE;
  res_word:WORD;
  n:BYTE:=2;
END_VAR

```

```
res_byte:=SHL(in_byte,n);    (* Результат 16#14 *)
res_word:=SHL(in_word,n);  (* Результат 16#0114 *)
```

Пример FBD:



Пример IL:

```
LD    16#45
SHL   2
ST    res_byte
```

SHR

res:= SHR(in,n) Побитный сдвиг операнда in вправо на n бит с дополнением нулями слева.

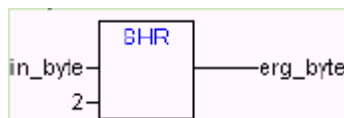
Входные переменные и результат должны быть типа BYTE, WORD или DWORD.

Следующий пример подчеркивает зависимость результата от типа входной переменной (см. примечание выше):

Пример ST:

```
PROGRAM shr_st
VAR
    in_byte:BYTE:=16#45;
    in_word:WORD:=16#45;
    res_byte:BYTE;
    res_word:WORD;
    n:BYTE:=2;
END_VAR
res_byte:=SHR(in_byte,n);    (* Результат 16#11 *)
res_word:=SHR(in_word,n);  (* Результат 16#0011 *)
```

Пример FBD:



Пример IL:

```
LD    16#45
SHR   2
ST    res_byte
```

ROL

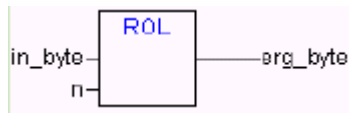
res:=ROL(in,n) Циклический сдвиг операнда in влево на n бит, младшие биты последовательно заполняются старшими. Входные переменные и результат должны быть типа BYTE, WORD или DWORD.

В следующем примере представлены различные результаты *res_byte* и *res_word* в зависимости от типа входной переменной (BYTE и WORD), хотя числовые их значения равны.

Пример ST:

```
PROGRAM rol_st
VAR
    in_byte:BYTE:=16#45;
    in_word:WORD:=16#45;
    res_byte:BYTE;
    res_word:WORD;
    n:BYTE:=2;
END_VAR
res_byte:=ROL(in_byte,n);      (* Результат 16#15 *)
res_word:=ROL(in_word,n);     (* Результат 16#0114 *)
```

Пример FBD:



Пример IL:

```
LD    16#45
ROL   2
ST    res_byte
```

ROR

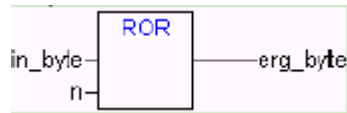
res:=ROR(in,n) Циклический сдвиг операнда *in* вправо на *n* бит, младшие биты последовательно заменяют старшие. Входные переменные и результат должны быть типа BYTE, WORD или DWORD.

Следующий пример подчеркивает зависимость результата от типа входной переменной (см. примечание выше):

Пример ST:

```
PROGRAM ror_st
VAR
    in_byte:BYTE:=16#45;
    in_word:WORD:=16#45;
    res_byte:BYTE;
    res_word:WORD;
    n:BYTE:=2;
END_VAR
res_byte:=ROR(in_byte,n);     (* Результат 16#51 *)
res_word:=ROR(in_word,n);    (* Результат 16#4011 *)
```

Пример FBD:



Пример IL:

```
LD 16#45
ROR 2
ST res_byte
```

10.4 Операторы выборки

Константы в примерах этого раздела используются исключительно для наглядности. Все операции можно выполнять и с переменными.

SEL

Бинарный выбор.

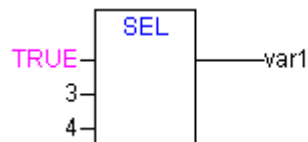
OUT:= SEL(G, IN0, IN1) означает
 OUT:= IN0 если G=FALSE;
 OUT:= IN1 если G=TRUE.

IN0, IN1 и OUT могут быть любого типа, G должно быть типа BOOL. Бинарный выбор возвращает одно из двух: IN0, если G ЛОЖЬ, или IN1, если G ИСТИНА.

Пример IL:

```
LD FALSE
SEL 3,4
ST Var1 (* Результат - 3 *)
```

Пример FBD:



Внимание: Выражение, стоящее перед IN0 или IN1 может не вычисляться, если соответствующий вход не выбран, что определяется значением G.

MAX

Функция максимум возвращает наибольшее из двух значений.

OUT := MAX(IN0, IN1)

IN0, IN1 и OUT могут быть любого типа.

Пример IL:

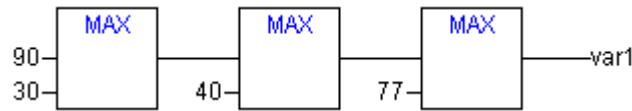
```
LD 90
MAX 30
MAX 40
```



```

MAX 77
ST   Var1  (* Результат - 90 *)
    
```

Пример FBD:



MIN

Функция минимум возвращает наименьшее из двух значений.

$OUT := MIN(IN0, IN1)$

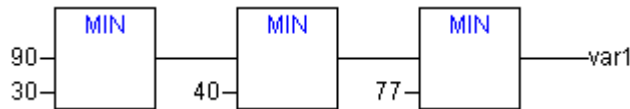
IN0, IN1 и OUT могут быть любого типа.

Пример IL:

```

LD   90
MIN  30
MIN  40
MIN  77
ST   Var1  (* Результат 30 *)
    
```

Пример FBD:



LIMIT

Ограничитель

$OUT := LIMIT(Min, IN, Max)$ означает:

$OUT := MIN (MAX (IN, Min), Max)$

Max задает верхнюю и Min нижнюю границы ограничителя. Если IN больше верхнего или меньше нижнего пределов, результат 'обрезается' соответственно до Max или Min.

IN и OUT могут быть любого типа.

Пример IL:

```

LD   90
LIMIT 30,80
ST   Var 1  (* Результат - 80 *)
    
```

MUX

Мультиплексор. Возвращает K-е значение из входных переменных.

$OUT := MUX(K, IN0, \dots, INn)$ означает:

$OUT := INK.$

IN0, ...,INn и OUT могут быть любого типа. Переменная K должна быть BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT или UDINT.

Пример IL:

```
LD    0
MUX  30,40,50,60,70,80
ST    Var 1          (* Результат - 30 *)
```

Внимание: В результате оптимизации выражение, стоящее перед входом, может не вычисляться, если соответствующий вход не выбран. В режиме эмуляции все выражения вычисляются.

10.5 Операторы сравнения

GT

Больше

Двоичный оператор возвращает TRUE, если значение первого параметра больше второго.

Операнды могут быть типов BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME и STRING.

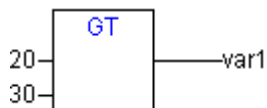
Пример IL:

```
LD    20
GT    30
ST    Var 1          (*Результат - ЛОЖЬ*)
```

Пример ST:

```
VAR1 := 20 > 30 > 40 > 50 > 60 > 70;
```

Пример FBD:



LT

Меньше

Двоичный оператор возвращает TRUE, если значение первого параметра меньше второго.

Операнды могут быть типов BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME и STRING.

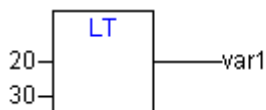
Пример IL:

```
LD    20
LT    30
ST    Var 1          (*Результат - ИСТИНА*)
```

Пример ST:

```
VAR1 := 20 < 30;
```

Пример FBD:



LE

Меньше или равно

Двоичный оператор возвращает TRUE, если значение первого параметра меньше или равно второму.

Операнды могут быть типов BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME и STRING.

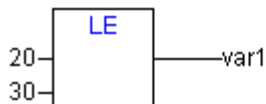
Пример IL:

```
LD    20
LE    30
ST    Var 1          (*Результат - ИСТИНА*)
```

Пример ST:

```
VAR1 := 20 <= 30;
```

Пример FBD:



GE

Больше или равно

Двоичный оператор возвращает TRUE, если значение первого параметра больше или равно второму.

Операнды могут быть типов BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME и STRING.

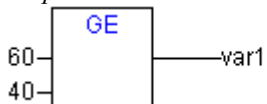
Пример IL:

```
LD    60
GE    40
ST    Var 1          (*Результат - ИСТИНА*)
```

Пример ST:

```
VAR1 := 60 >= 40;
```

Пример FBD:



EQ

Равно

Двоичный оператор возвращает TRUE, если значение первого параметра равно второму.

Операнды могут быть типов BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME и STRING.

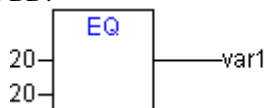
Пример IL:

```
LD    40
EQ    40
ST    Var 1      (*Результат - ИСТИНА*)
```

Пример ST:

```
VAR1 := 40 = 40;
```

Пример FBD:



NE

Не равно

Двоичный оператор возвращает TRUE, если значение первого параметра не равно второму.

Операнды могут быть типов BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME и STRING.

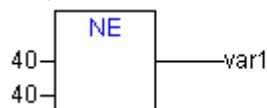
Пример IL:

```
LD    40
NE    40
ST    Var 1      (*Результат - FALSE*)
```

Пример ST:

```
VAR1 := 40 <> 40;
```

Пример FBD:



10.6 Адресные операторы и вспомогательные функции

Внимание: При онлайн коррекции кода, адреса переменных могут измениться. Не забывайте о такой вероятности, используя указатели.

Описанные в этом разделе операторы и функции не предписаны стандартом МЭК.

ADR

Функция адрес

ADR возвращает адрес аргумента в формате DWORD. Полученный адрес может быть передан функции в качестве аргумента или присвоен переменной типа указатель.

Пример IL:

```
LD    Var 1
ADR
ST    Var 2
man_fun1
```

ADRINST

При вызове функции ADRINST из экземпляра функционального блока она возвращает адрес экземпляра в DWORD. Данный адрес можно передавать в функции и использовать как указатель, либо его можно непосредственно присвоить переменной проекта типа указатель.

Пример ST (внутри экземпляра функционального блока):

```
dvar:=ADRINST(); (* Присваивание адреса экземпляра переменной dvar *)
fun(a:=ADRINST()); (* Адрес экземпляра как аргумент функции *)
```

Пример IL:

```
ADRINST
ST dvar
ADRINST
fun
```

BITADR

BITADR возвращает DWORD смещение переменной от начала сегмента данных. Результат зависит от того, включена или нет опция байтовой адресации.

```
VAR
    var1 AT %IX2.3:BOOL;
    bitoffset: DWORD;
END_VAR
```

Пример ST:

```
Bitoffset := BITADR(var1); (* Результат 19, если byte addressing=TRUE, или 35, если
                             byte addressing=FALSE*)
```

Пример IL:

```
LD Var1
BITADR
ST Var2
```

Косвенный оператор (content)

Косвенное обращение через указатель производится путем добавления оператора "^" после имени указателя.

Пример ST:

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

Вспомогательные функции

INDEXOF

Возвращает внутренний индекс POU.

Пример ST:

```
var1 := INDEXOF(POU2);
```

SIZEOF

Возвращает размер переменной в байтах.

Пример IL:

```
arr1:ARRAY[0..4] OF INT;
Var1 INT
LD   arr1
    SIZEOF
ST   Var 1   (* Результат 10 *)
```

Пример ST:

```
var1 := SIZEOF(arr1);
```

TIME

Функция TIME возвращает время в миллисекундах от начала работы системы в формате TIME.

Пример IL:

```
TIME
ST systime (*Результат, например.: T#35m11s342ms *)
```

Пример ST:

```
systime:=TIME();
```

Пример FBD:



INI

INI можно использовать для инициализации retain переменных, содержащихся в экземпляре функционального блока.

Синтаксис: <bool-Variable> := INI(<FB-instance, TRUE|FALSE)

Если второй операнд равен TRUE, то все энергонезависимые переменные, определенные в функциональном блоке FB, будут инициализироваться.

Пример ST: fbinst экземпляр функционального блока fb, в котором определена retain переменная retvar.

Объявление:

```
fbinst:fb;
b:boolean;
```

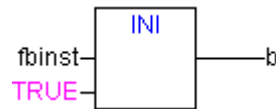
Фрагмент реализации:

```
b := INI(fbinst, TRUE);
ivar:=fbinst.retvar (* => retvar инициализирована *)
```

Пример IL:

```
LD fbinst
INI TRUE
ST b
```

Пример FBD:



10.7 Оператор вызова

CAL

Вызывает функциональный блок или программу.

Применяется в IL для вызова экземпляра функционального блока, входные переменные помещаются в скобках вслед за его именем.

Пример: Вызов экземпляра функционального блока Inst, где входные переменные Par1 и Par2 равны 0 и TRUE соответственно.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

10.8 Явное преобразование типов

Неявное преобразование данных из 'большого' типа в 'меньший' запрещено стандартом (например, из INT в BYTE или DINT в WORD). Если подобное преобразование действительно необходимо, оно должно быть выражено явно. Явные преобразования работают практически для всех базовых типов данных.

Синтаксис:

<баз.Tun1>_TO_<баз..Tun2>

Преобразования в строку ..._TO_STRING используют выравнивание влево. Если строка оказывается слишком короткой, результат будет обрезан.

BOOL_TO

Преобразование типа BOOL в другой тип:

Для числовых типов результат равен 1, когда операнд TRUE, и 0, если операнд FALSE.

Для типа STRING результат - это слово TRUE или FALSE.

Примеры IL:

```
LD TRUE
BOOL_TO_INT
ST I (*Результат: 1 *)

LD TRUE
BOOL_TO_STRING
ST str (*Результат: 'TRUE' *)

LD TRUE
BOOL_TO_TIME
ST t (*Результат: T#1ms *)

LD TRUE
BOOL_TO_TOD
ST (*Результат: TOD#00:00:00.001 *)

LD FALSE
BOOL_TO_DATE
ST dat (*Результат: D#1970-01-01 *)

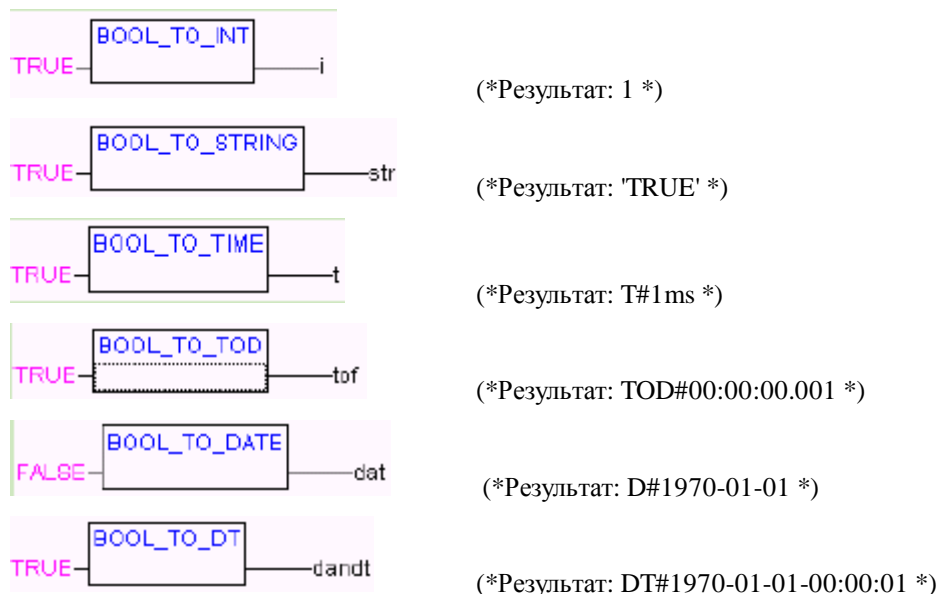
LD TRUE
BOOL_TO_DT
ST dandt (*Результат:DT#1970-01-01-00:00:01 *)
```

Примеры ST:

```
i:=BOOL_TO_INT(TRUE); (* Результат:1 *)
str:=BOOL_TO_STRING(TRUE); (* Результат:"TRUE" *)
t:=BOOL_TO_TIME(TRUE); (* Результат:T#1ms *)
tof:=BOOL_TO_TOD(TRUE); (* Результат:TOD#00:00:00.001 *)
dat:=BOOL_TO_DATE(FALSE); (* Результат:D#1970 *)
```


dandt:=BOOL_TO_DT(TRUE); (* Результат: DT#1970-01-01-00:00:01 *)

Примеры FBD:



TO_BOOL

Преобразование других типов в BOOL:

Результат TRUE, когда операнд не нулевой, иначе FALSE.

Для типа STRING, если строка состоит из слова "TRUE", результат равен TRUE, иначе FALSE.

Примеры IL:

```
LD 213
BYTE_TO_BOOL
ST b (*Результат: TRUE *)

LD 0
INT_TO_BOOL
ST b (*Результат: FALSE *)

LD T#5ms
TIME_TO_BOOL
ST b (*Результат: TRUE *)

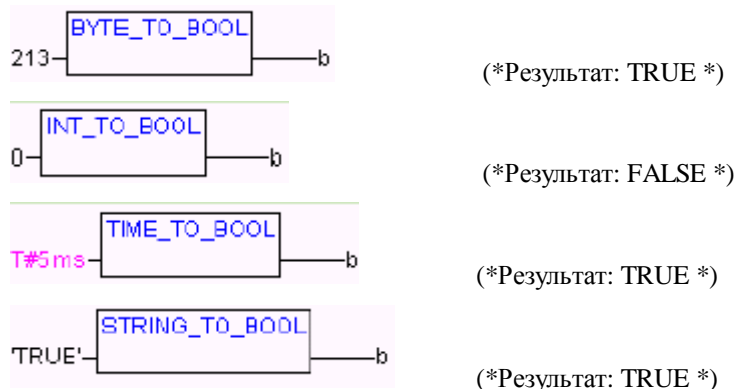
LD 'TRUE'
STRING_TO_BOOL
ST b (*Результат: TRUE *)
```

Примеры ST:

```

b := BYTE_TO_BOOL(2#11010101);    (* Результат: TRUE *)
b := INT_TO_BOOL(0);              (* Результат: FALSE *)
b := TIME_TO_BOOL(T#5ms);        (* Результат: TRUE *)
b := STRING_TO_BOOL('TRUE');     (* Результат: TRUE *)
    
```

Примеры FBD:



Преобразования между целочисленными типами

Преобразования из одного целочисленного типа в другой:

При преобразовании большего типа в меньший, Вы рискуете потерять информацию. Если число превосходит верхний предел данного типа, старшие байты числа игнорируются.

Пример ST:

```

si := INT_TO_SINT(4223);    (* Результат: 127 *)
    
```

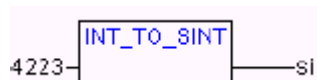
Число 4223 (16#107F в шестнадцатеричной форме) сокращается до переменной типа SINT, теряя старший байт, и принимает значение 127 (16#7F в шестнадцатеричной форме).

Пример IL:

```

LD    2
INT_TO_REAL
MUL  3.5
    
```

Пример FBD:



REAL_TO-/ LREAL_TO

Преобразования типов REAL или LREAL в другие типы:

Значение числа будет округлено вверх до целого и преобразовано в нужный тип. За исключением типов STRING, BOOL, REAL и LREAL. При преобразовании большего типа в меньший, вы рискуете потерять информацию.

При преобразовании в STRING используется до 16 цифр. Если строка имеет не достаточный размер, результат будет обрезан справа.

Пример ST:

```

i := REAL_TO_INT(1.5);    (* Результат: 2 *)
    
```

```

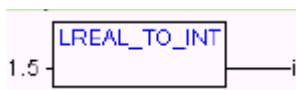
j := REAL_TO_INT(1.4);      (* Результат: 1 *)
i := REAL_TO_INT(-1.5);    (* Результат: -2 *)
j := REAL_TO_INT(-1.4);    (* Результат: -1 *)
    
```

Пример IL:

```

LD    2.7
REAL_TO_INT
GE    %MW8
    
```

Пример FBD:



TIME_TO/TIME_OF_DAY

Преобразования типов TIME или TIME_OF_DAY в другие типы:

Физически значение времени сохраняется в переменной типа DWORD, выраженное в миллисекундах (начиная с 0 часов для TIME_OF_DAY). Собственное это число и будет преобразовано. При преобразовании в меньший тип, вы рискуете потерять информацию.

При преобразовании в STRING образуется соответствующая МЭК текстовая строка.

Примеры IL:

```

LD    T#12ms
TIME_TO_STRING
ST    str                      (*Результат: 'T#12ms' *)

LD    T#300000ms
TIME_TO_DWORD
ST    dw                       (*Результат: 300000 *)

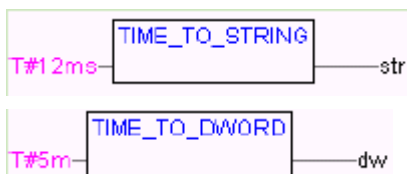
LD    TOD#00:00:00.012
TOD_TO_SINT
ST    si                       (*Результат: 12 *)
    
```

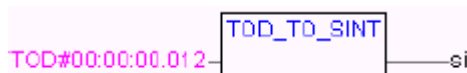
Примеры ST:

```

str :=TIME_TO_STRING(T#12ms);    (* Результат: T#12ms *)
dw :=TIME_TO_DWORD(T#5m);       (* Результат: 300000 *)
si :=TOD_TO_SINT(TOD#00:00:00.012); (* Результат: 12 *)
    
```

Примеры FBD:





DATE_TO/DT_TO

Преобразования типов DATE или DATE_AND_TIME в другие типы:

Физически значение даты сохраняется в переменной типа DWORD, выраженное в секундах начиная с 1 января 1970 г. Это число и будет преобразовано. При преобразовании в меньший тип вы рискуете потерять информацию.

При преобразовании в STRING образуется соответствующая МЭК текстовая строка.

Примеры IL:

```
LD D#1970-01-01      (* Результат FALSE *)
```

```
DATE_TO_BOOL
```

```
ST b
```

```
LD D#1970-01-15     (*Результат 29952 *)
```

```
DATE_TO_INT
```

```
ST i
```

```
LD DT#1970-01-15-05:05:05  (*Результат 129 *)
```

```
DT_TO_BYTE
```

```
ST byt
```

```
LD DT#1998-02-13-14:20     (*Результат 'DT#1998-02-13-14:20' *)
```

```
DT_TO STRING
```

```
ST str
```

Примеры ST:

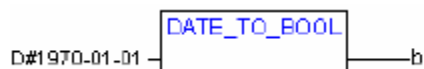
```
b :=DATE_TO_BOOL(D#1970-01-01);      (* Результат: FALSE *)
```

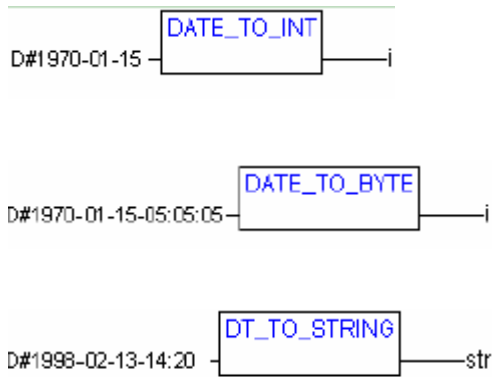
```
i :=DATE_TO_INT(D#1970-01-15);      (* Результат: 29952 *)
```

```
byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05); (* Результат: 129 *)
```

```
str:=DT_TO_STRING(DT#1998-02-13-14:20); (*Результат:'DT#1998-02-13-14:20' *)
```

Примеры FBD:





STRING_TO

Преобразования типа `STRING` в другие типы:

Содержимое строки должно соотноситься с желаемым типом данных, в противном случае преобразование дает 0.

Пример IL:

```
LD 'TRUE'      (*Результат: TRUE *)
STRING_TO_BOOL
ST b

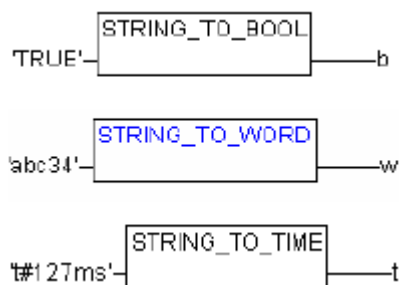
LD 'abc34'     (*Результат: 0 *)
STRING_TO_WORD
ST w

LD 't#127ms'   (*Результат: T#127ms *)
STRING_TO_TIME
ST t
```

Примеры ST:

```
b :=STRING_TO_BOOL('TRUE');      (* Результат: TRUE *)
w :=STRING_TO_WORD('abc34');     (* Результат: 0 *)
t :=STRING_TO_TIME('T#127ms');   (* Результат: T#127ms *)
```

Примеры FBD:



TRUNC

Преобразование из REAL в INT. Используется только целочисленная часть аргумента.

При преобразовании в меньший тип вы рискуете потерять информацию.

Примеры ST:

```

i:=TRUNC(1.9);          (* Результат: 1 *)
i:=TRUNC(-1.4);        (* Результат: 1 *)
  
```

```

LD    2.7
TRUNC
GE    %MW8
  
```

10.9 Математические функции

ABS

Возвращает абсолютное значение числа. Например, ABS(-2) равно 2.

Возможны следующие комбинации типов аргумента и результата:

Параметр	Результат
INT	INT, REAL, WORD, DWORD, DINT
REAL	REAL
BYTE	INT, REAL, BYTE, WORD, DWORD, DINT
WORD	INT, REAL, WORD, DWORD, DINT
DWORD	REAL, DWORD, DINT
SINT	REAL
USINT	REAL
UINT	INT, REAL, WORD, DWORD, DINT, UDINT, UINT
DINT	REAL, DWORD, DINT
UDINT	REAL, DWORD, DINT, UDINT

Пример IL:

```
LD    2
```

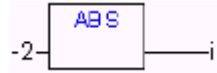
ABS

ST i (*Результат: 2 *)

Примеры ST:

i:=ABS(-2);

Пример FBD:



SQRT

Квадратный корень числа.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

Пример IL:

LD 16

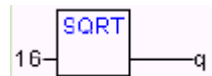
SQRT

ST q (*Результат: 4 *)

Примеры ST:

q:=SQRT(16);

Пример FBD:



LN

Натуральный логарифм числа.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

Пример IL:

LD 45

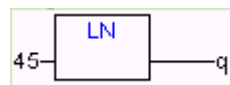
LN

ST q (*Результат: 3.80666 *)

Примеры ST:

q:=LN(45);

Пример FBD:



LOG

Десятичный логарифм числа.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

Пример IL:

```
LD 314.5
LOG
ST q (*Результат: 2.49762 *)
```

Примеры ST:

```
q:=LOG(314.5);
```

Пример FBD:



EXP

Экспонента.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

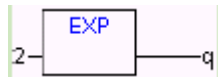
Пример IL:

```
LD 2
EXP
ST q (*Результат: 9.7448e+009 *)
```

Примеры ST:

```
q:=EXP(2);
```

Пример FBD:



SIN

Синус.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL. Аргумент измеряется в радианах.

Пример IL:

```
LD 0.5
SIN
ST q (*Результат: 0.479426 *)
```

Пример ST:

```
q:=SIN(0.5);
```

Пример FBD:



COS

Косинус.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL. Аргумент измеряется в радианах.

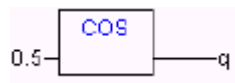
Пример IL:

```
LD    0.5
COS
ST    q          (*Результат: 0.877583 *)
```

Пример ST:

```
q:=COS(0.5);
```

Пример FBD:

**TAN**

Тангенс.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL. Аргумент измеряется в радианах.

Пример IL:

```
LD    0.5
TAN
ST    q          (*Результат: 0.546302 *)
```

Пример ST:

```
q:=TAN(0.5);
```

Пример FBD:

**ASIN**

Арсинус.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

Пример IL:

```
LD    0.5
ASIN
ST    q          (*Результат: 0.523599 *)
```

Пример ST:

```
q:=ASIN(0.5);
```

Пример FBD:



ACOS

Арккосинус.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

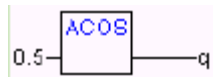
Пример IL:

```
LD    0.5
ABS
ST    q (*Результат: 1.0472 *)
```

Пример ST:

```
q:=ACOS(0.5);
```

Пример FBD:



ATAN

Арктангенс.

Аргумент может быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, результат должен быть типа REAL.

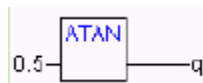
Пример IL:

```
LD    0.5
ABS
ST    q (*Результат: 0.463648 *)
```

Пример ST:

```
q:=ATAN(0.5);
```

Пример FBD:



EXPT

Число в степени:

$$OUT = IN_1^{IN_2}$$

IN1 и IN2 могут быть типов BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT должна быть REAL.

Пример IL:

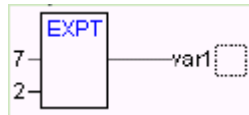
```
LD    7
EXPT  2
```

ST var1 (*Результат: 49 *)

Пример ST:

var1 := (7,2);

Пример FBD:



Приложение В: Операнды в CoDeSys

В качестве операндов могут выступать константы переменные, адреса и вызовы функций.

10.10 Константы

BOOL

BOOL константы могут иметь значение TRUE или FALSE.

TIME

Константы типа TIME в CoDeSys всегда начинаются с префикса "t" или "T" (длинная форма "time" или "TIME") и знака числа "#". Далее следует собственно время, которое может включать дни "d", часы "h", минуты "m", секунды "s" и миллисекунды "ms". Нет необходимости обязательно определять все составляющие времени, но присутствующие поля обязаны следовать именно в таком порядке (d, затем h, затем m, затем s, затем ms).

Правильные примеры TIME констант в ST:

```
TIME1 := T#14ms;
```

```
TIME1 := T#100S12ms; (*Старший компонент может выходить за свой предел*)
```

```
TIME1 := t#12h34m15s;
```

Ошибочные примеры:

```
TIME1 := t#5m68s; (*Младший компонент вышел за предел*)
```

```
TIME1 := 15ms; (*T# пропущено*)
```

```
TIME1 := t#4ms13d; (*Ошибочная последовательность*)
```

DATE

Константы типа DATE начинаются с префикса "d", "D", "DATE" или "date" и последующего "#". Даты задаются в формате Год-Месяц-День.

Примеры:

```
DATE#1996-05-06
```

```
d#1972-03-29
```

См. также 10.14 Переменные типа DATE

TIME_OF_DAY

Константы типа A TIME_OF_DAY начинаются с префикса "tod#", "TOD#", "TIME_OF_DAY#" или "time_of_day#" и последующего времени в формате: Часы:Минуты:Секунды. Секунды можно задавать в виде десятичной дроби.

Примеры:

```
TIME_OF_DAY#15:36:30.123
```

```
tod#00:00:00
```

DATE_AND_TIME

Константы типа DATE_AND_TIME начинаются с префикса "dt#", "DT#", "DATE_AND_TIME#" или "date_and_time#". Дата и время приводятся последовательно через дефис.

Примеры:

DATE_AND_TIME#1996-05-06-15:36:30

dt#1972-03-29-00:00:00

Целочисленные константы

Числовые значения могут быть представлены в двоичной, восьмеричной, десятичной и шестнадцатеричной форме.

Если число не десятичное, необходимо указать основание числа с префиксом # перед числом. Цифры диапазона 10-15 в шестнадцатеричной форме заменяются литерами A-F.

Для удобства чтения в любом месте числа можно вставить подчеркивание.

Примеры:

14 (*десятичное число*)

2#1001_0011 (*двоичное число*)

8#67 (*восьмеричное число*)

16#A (*шестнадцатеричное число*)

Данные числа могут быть представлены типами BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL и LREAL. Неявное преобразование из "большого" в "младший" тип переменной не производится. Это означает, что нельзя просто использовать DINT как INT переменную, необходимо применять преобразование типов (см. раздел «Преобразование типов»).

REAL/LREAL

REAL и LREAL константы представляются в формате с десятичной точкой либо в экспоненциальном формате. Запятая вместо точки не допускается.

Примеры:

7.4 но не 7,4

1.64e+009 но не 1,64e+009

STRING

Константы типа STRING представляются в виде набора символов, заключенных в одинарные кавычки. Строка может содержать пробелы и специальные символы (например, умляуты). Символы, не имеющие печатного образа, могут быть заданы шестнадцатеричным кодом в виде двух цифр, следующих за знаком доллара (\$). Специальные комбинации из двух символов, начинающиеся со знака доллара, интерпретируются следующим образом:

\$\$	Знак доллара
\$'	Одинарная кавычка
\$L или \$l	Line feed
\$N or \$n	New line
\$P or \$p	Page feed
\$R or \$r	Line break
\$T or \$t	Tab

Примеры:

'Полет нормальный'

```
' Abby and Craig '  
' :- )'
```

Типизированные константы

Обычно при использовании МЭК констант подразумевается наименьший из возможных типов данных. Если нужно точно указать тип константы, применяется префикс типа: <Type>#<Literal>

<Type> указывает необходимый тип. Это может быть: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. Тип нужно указывать заглавными буквами.

<Literal> указывает значение константы.

Пример:

```
var1:=DINT#34;
```

Если значение константы не соответствует требуемому типу, CoDeSys выдаст соответствующее сообщение об ошибке.

10.11 Переменные

Переменные могут быть объявлены либо как локальные в разделе определений POU, либо как глобальные - в списке глобальных переменных.

Внимание: Глобальная и локальная переменные могут иметь одинаковое имя. В POU, где объявлена такая локальная переменная, она оказывается «сильнее» одноименной глобальной. Использовать одноименные глобальные переменные нельзя (например, объявленные в конфигурации контроллера и в списке глобальных переменных).

Имя переменной (идентификатор) не должно содержать пробелов и спецсимволов, не должно объявляться более одного раза и не должно совпадать с ключевыми словами. Регистр символов не учитывается, это означает, что VAR1, Var1 и var1 - это одна и та же переменная.

Символ подчеркивания является значимым, т.е. "A_VCD" и "AB_CD" - это разные имена.

Имя должно включать не более одного символа подчеркивания. Ограничений на длину имени нет. Область применения переменной задается ее типом. Список всех объявленных переменных в CoDeSys доступен через ассистент ввода (Input Assistant).

Системные флаги

Системные флаги - это неявно объявленные переменные, различные для конкретных моделей PLC. Для получения списка доступных системных флагов используйте команду "Insert" "Operand". В диалоге ассистента ввода (Input Assistant) флаги собраны в разделе **System Variable**.

Синтаксис доступа к элементам массивов, структур и POU

Элемент двумерного массива:

```
<ИмяМассива>[Индекс1, Индекс2]
```

Переменная структуры:

```
<ИмяСтруктуры>.<ИмяПеременной>
```

Переменная программы или функционального блока:

```
<ИмяФункциональногоБлока>.<ИмяПеременной>
```

Доступ к битам в переменных

В целочисленных переменных существует возможность обращаться к отдельным битам. Для этого указывается номер бита, начиная с 0 через точку после имени.

```
a : INT;
b : BOOL;
...
a.2 := b;
```

В примере значение третьего бита переменной a будет присвоено переменной b.

Если указанный номер бита превышает размер типа, формируется специальное сообщение: «Index '<n>' outside the valid range for variable '<var>'»

Битовая адресация применима для типов :SINT, INT, DINT, USINT, UINT, UDINT, BYTE, WORD, DWORD.

Если битовая адресация для данного типа не поддерживается, CoDeSys формирует сообщение: «Invalid data type '<type>' for direct indexing»

Битовую адресацию нельзя использовать с переменными VAR_IN_OUT!

Битовая адресация через глобальные константы

Если объявить целую глобальную константу, то ее можно будет затем использовать для доступа к битам. Например, так:

Объявление константы

```
VAR_CONSTANT GLOBAL
enable:int := 1;
END_VAR
```

Пример 1, битовая адресация через константу:

Объявление POU:

```
VAR
xxx:int;
END_VAR
```

Битовая адресация:

```
xxx.enable := true; (*установлен в единицу второй бит переменной xxx *)
```

Пример 2, битовая адресация к элементу структуры:

Объявление структуры stru1:

```
TYPE stru1 :
STRUCT
    bvar: BOOL;
    rvar: REAL;
    wvar: WORD;
    {bitaccess: 'enable' 42 'Start drive'}
```


END_STRUCT

END_TYPE

Объявление POU:

VAR

 x:stru1;

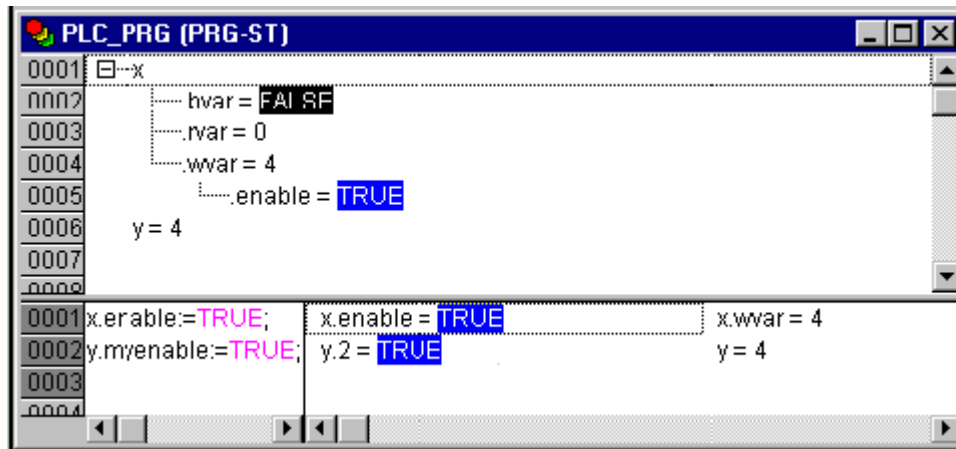
END_VAR

Битовая адресация:

 x.enable := true;

Эта инструкция установит 42й бит переменной x. Поскольку bvar занимает 8 бит, rvar занимает 32 бита, а битовый доступ обращается ко второму биту переменной wvar, получающей в результате значение 4.

Внимание: Для правильного отображения переменной, выполняющей битовый доступ к структурам через глобальную константу, в Ассистенте ввода при мониторинге в окне объявления и при использовании интеллектуального ввода используйте команду компилятора **pragma {bitaccess}**. В этом случае, кроме значения переменной, вы увидите и значение данного бита:



10.12 Адреса

Прямое указание адреса дает способ непосредственного обращения к конкретной области памяти. Прямой адрес образуется из префикса "%", префиксов области памяти и размера, одного или нескольких целых чисел, разделенных точкой.

Префиксы области памяти:

I	Входы
Q	Выходы
M	Память данных

Префиксы размера:

X	Один бит
Отсутствует	Один бит
B	Байт (8 бит)

Приложение В: Операнды в CoDeSys

W	Слово (16 бит)
D	Двойное слово (32 бит)

Примеры:

%QX7.5 и %Q7.5	бит 7.5 в области выходов
%IW215	215е слово в области входов
%QB7	байт 7 в области выходов
%MD48	двойное слово в позиции памяти 48
%IW2.5.7.1	зависит от конфигурации PLC

Является ли последний указанный адрес значимым, будет определяться конкретной конфигурацией PLC.

Примечание: По умолчанию логические переменные занимают один байт, если в объявлении не указан прямой битовый адрес. См. также битовые операции в приложении А.

Распределение памяти

Образование прямых адресов зависит от размера адресуемых данных.

Так, например, адрес %MD48 адресует в области памяти двойное слово 48 или байты 192, 193, 194 и 195 ($48 * 4 = 192$). Нумерация начинается с 0.

Адрес %MX5.0 означает младший бит пятого (считая с нуля) слова памяти.

10.13 Функции в роли операндов

В языке ST результат вызова функции может использоваться как операнд.

Пример:

```
Result := Fct(7) + 3;
```

Приложение С: Типы данных CoDeSys

Тип данных определяет род информации и методы ее обработки и хранения, количество выделяемой памяти. Программист может непосредственно использовать элементарные (базовые) типы данных или создавать собственные (пользовательские) типы на их основе.

10.14 Элементарные типы данных

Логический (BOOL)

BOOL логический тип данных. Переменная может принимать 2 значения ИСТИНА (TRUE) или ЛОЖЬ (FALSE). Занимает 8 бит памяти, если не задан прямой битовый адрес (См. «10.12 Адреса»).

Целочисленные

BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, и UDINT - все это целочисленные типы. Они отличаются различным диапазоном сохраняемых данных и, естественно, различными требованиями к памяти. Подробно данные характеристики представлены в следующей таблице:

Тип	Нижний предел	Верхний предел	Размер памяти
BYTE	0	255	8 Бит
WORD	0	65535	16 Бит
DWORD	0	4294967295	32 Бит
SINT	-128	127	8 Бит
USINT	0	255	8 Бит
INT	-32768	32767	16 Бит
UINT	0	65535	16 Бит
DINT	-2147483648	2147483647	32 Бит
UDINT	0	4294967295	32 Бит

Очевидно, присвоение данных большего типа переменной меньшего типа может приводить к потере информации.

Рациональные

REAL и **LREAL** данные в формате с плавающей запятой, используются для сохранения рациональных чисел. Для типа **REAL** необходимо 32 бита памяти и 64 для **LREAL**.

Диапазон значений **REAL** от: 1.175494351e-38F до 3.402823466e+38F

Диапазон значений **LREAL** от: 2.2250738585072014e-308 до 1.7976931348623158e+308

Строки

Строковый тип **STRING** представляет строки символов. Максимальный размер строки определяет количество резервируемой памяти и указывается при объявлении переменной. Размер задается в круглых или квадратных скобках. Если размер не указан, принимается размер по умолчанию – 80 символов.

Длина строки не ограничена в CoDeSys, но строковые функции способны обращаться со строками от 1 до 255 символов !

Пример объявления строки размером до 35 символов:

```
str:STRING(35):='Просто строка';
```

Размер строк в CoDeSys не ограничен, но библиотека работы со строками поддерживает строки от 1 до 255 символов.

Время и дата

TIME представляет длительность интервалов времени в миллисекундах. Максимальное значение для типа **TIME** : 49d17h2m47s295ms (4194967295 ms).

TOD содержит время суток, начиная с 0 часов. Диапазон значений **TOD** от: 00:00:00 до 23:59:59.999.

DATE содержит календарную дату, начиная с 1 января 1970 года. Диапазон значений от: 1970-00-00 до 2106-02-06.

DT содержит время в секундах, начиная с 0 часов 1 января 1970 года. Диапазон значений от: 1970-00-00-00:00:00 до 2106-02-06-06:28:15.

Типы **TIME**, **TIME_OF_DAY** (сокр. **TOD**), **DATE** и **DATE_AND_TIME** (сокр. **DT**) сохраняются физически как **DWORD**.

Формат представления данных описан в разделе «10.10 Константы».

10.15 Пользовательские типы данных

Массивы

Элементарные типы данных могут образовывать одно-, двух-, и трехмерные массивы. Массивы могут быть объявлены в разделе объявлений **POU** или в списке глобальных переменных. Путем вложения массивов можно получить многомерные массивы, но не более 9 мерных ("ARRAY[0..2] OF ARRAY[0..3] OF ...").

Синтаксис:

```
<Имя_массива>:ARRAY [<ll1>..
```

где ll1, ll2, ll3 указывают нижний предел индексов; ul1, ul2 и ul3 указывают верхние пределы.

Индексы должны быть целого типа. Нельзя использовать отрицательные индексы.

Пример:

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

Пример инициализации простых массивов:

```
arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;
```

```
arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (* сокращение для 3 по 7: 1,7,7,7 *)
```

```
arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3; (*сокращение для 0,0,4,4,4,4,2,3 *)
```

Пример инициализации массива структур:

```
TYPE STRUCT1
```

```
STRUCT
```

```
    p1:int;
```

```
    p2:int;
```

```
    p3:dword;
```

```
END_STRUCT
```

```
ARRAY[1..3] OF STRUCT1:= (p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299),
```

```
(p1:=14,p2:=5,p3:=112);
```

Пример инициализации части массива:

```
arr1 : ARRAY [1..10] OF INT := 1,2;
```

Не инициализированные явно элементы массива принимают значения по умолчанию. Так, в данном примере оставшиеся элементы примут значение 0.

Доступ к элементам массива:

Для доступа к элементам двумерного массива используется следующий синтаксис:

```
<Имя_массива>[Индекс1,Индекс2]
```

Пример:

```
Card_game [9,2]
```

Примечание: Если определить в проекте функцию с именем CheckBounds, вы сможете контролировать нарушение диапазона индексов массивов в проекте.

Функция CheckBounds

Определив в проекте функцию с именем CheckBounds, вы сможете использовать её для контроля за соблюдением границ индексов. Имя функции фиксировано, изменять его нельзя.

Пример функции CheckBounds:

```
FUNCTION CheckBounds : DINT
VAR_INPUT
    index, lower, upper: DINT;
END_VAR

IF index < lower THEN
    CheckBounds := lower;
ELSIF index > upper THEN
    CheckBounds := upper;
ELSE CheckBounds := index;
END_IF
```

В этом примере CheckBounds ограничивает индекс массива заданными границами. Если запрашивается элемент, отсутствующий в массиве, функция CheckBounds возвращает ближайший элемент.

Проверка работы функции CheckBounds:

```
PROGRAM PLC_PRG
VAR
    a: ARRAY[0..7] OF BOOL;
    b: INT:=10;
END_VAR
a[b] := TRUE;
```

Внимание: Функция CheckBounds, содержащаяся в библиотеке Check.Lib, представляет собой пример реализации. Прежде чем использовать эту библиотеку, убедитесь, что данная функция работает так, как нужно в вашем случае, либо создайте собственную функцию непосредственно в вашем проекте.

Указатели

Указатели позволяют работать с адресами переменных или функциональных блоков.

Синтаксис:

<Имя_указателя>: **POINTER TO** <Тип данных/Функциональный блок>;

Указатели применимы для всех базовых типов данных или функциональных блоков, включая определяемые пользователем.

Адреса переменных и функциональных блоков можно получить во время исполнения программы при помощи оператора ADR. Для обращения через указатель необходимо добавить оператор "^" (content) после его имени.

Обратите внимание: Указатели инкрементируются побайтно! Для увеличения указателя, как это принято в C-компиляторах, используйте инструкцию $p=p+SIZEOF(p^)$;

Пример:

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2 := pt^;      (* var_int2 теперь равна 5 *)
```

Функция CheckPointer:

Данная функция позволяет контролировать обращение к допустимой области памяти через указатели. Если определена функция **CheckPointer**, то она будет автоматически вызываться при любом обращении через указатель. Функция должна быть определена в проекте (непосредственно или в библиотеке). Ее имя (CheckPointer) изменять нельзя. Функция имеет следующие параметры:

Для систем с 32-х битными указателями:


```
FUNCTION CheckPointer : DWORD
  VAR_INPUT
    dwAddress : DWORD;
    iSize : INT;
    bWrite: BOOL;
  END_VAR
```

Для систем с 16-и битными указателями:

```
FUNCTION CheckPointer : WORD
  VAR_INPUT
    dwAddress : WORD;
    iSize : INT;
    bWrite: BOOL;
  END_VAR
```

Функция возвращает адрес, который будет использоваться как указатель. Если все хорошо то, это будет входной параметр – dwAddress.

Перечисление

Перечисление - это определяемый пользователем тип данных, задающий несколько строковых псевдонимов для числовых констант. Перечисление доступно в любой части проекта, даже при локальном его объявлении внутри POU. Поэтому наиболее разумно создавать все перечисления на вкладке типы данных (**Data types**) «Организатора Объектов»  (Object Organizer). Объявление должно начинаться с ключевого слова TYPE и заканчиваться строкой END_TYPE.

Синтаксис:

```
TYPE <Имя_перечисления>:(<Элемент_0> ,< Элемент_1>, ...< Элемент_n>);
END_TYPE
```

Переменная типа *<Имя_перечисления>* может принимать только перечисленные значения. При инициализации переменная получает первое из списка значение. Если числовые значения элементов перечисления не указаны явно, им присваиваются последовательно возрастающие числа, начиная с 0. Фактически элемент перечисления - это число типа INT и работать с ними можно точно также. Можно напрямую присвоить число переменной типа перечисление.

Пример:

```
TYPE TRAFFIC_SIGNAL: (Red, Yellow, Green:=10);      (*Каждому цвету
                                                    соответствует свое значение, для red - это 0, для yellow - 1 и для green - 10 *)
END_TYPE
TRAFFIC_SIGNAL1 : TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0;      (*Переменная получила значение red*)
FOR i:= Red TO Green DO
    i := i + 1;
END_FOR
```


Элемент, уже включенный в перечисление, нельзя повторно включать в другое перечисление.

Пример:

```
TRAFFIC_SIGNAL: (red, yellow, green);
COLOR: (blue, white, red);
```

Ошибка: попытка повторного использования элемента TRAFFIC_SIGNAL **red** в COLOR.

Структуры

Структуры создаются на вкладке типы данных (**Data types**) «Организатора Объектов»  (Object Organizer). Объявление должно начинаться с ключевых слов TYPE и STRUCT и заканчиваться строками END_STRUCT и END_TYPE.

Синтаксис:

```
TYPE <Имя_структуры>:
STRUCT
    <Объявление переменной 1>
    .
    .
```

<Объявление переменной n>

END_STRUCT

END_TYPE

<Имя _структуры> образует новый тип данных, который может быть использован в любой части проекта наряду с базовыми типами.

Вложенные структуры допускаются. Единственное ограничение заключается в запрете размещения элементов структуры по прямым адресам (АТ объявления недопустимы!).

Пример объявления структуры по имени *Polygone*:

```
TYPE Polygone:
STRUCT
    Start:ARRAY [1..2] OF INT;
    Point1:ARRAY [1..2] OF INT;
    Point2:ARRAY [1..2] OF INT;
    Point3:ARRAY [1..2] OF INT;
    Point4:ARRAY [1..2] OF INT;
    End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Пример инициализации структуры:

```
Poly_1:polygone := ( Start:=3,3, Point1 =5,2, Point2:=7,3, Point3:=8,5,
Point4:=5,7, End := 3,5);
```

Для доступа к элементам структуры используется следующий синтаксис:


<Имя_структуры>.<Имя_компонента>

Например, структура "Week" содержит компонент "Monday", обращение к которому будет выглядеть так:

Week.Monday

Псевдонимы типов

Псевдонимы типов нужны для создания альтернативных пользовательских наименований типов данных. Это удобно при работе с большим числом однотипных констант, переменных и функциональных блоков.

Псевдонимы типов определены на вкладке типы данных (**Data types**) «Организатора Объектов»  (Object Organizer). Объявление должно начинаться с ключевого слова TYPE и заканчиваться строкой END_TYPE.

Синтаксис:

```
TYPE <Имя псевдонима>: <Исходное имя>;
END_TYPE
```

Пример:

```
TYPE message: STRING[50];
END_TYPE;
```


Ограничение диапазона значений

Ограничение диапазона позволяет объявить переменную, значения которой ограничены в определенных пределах. Существует возможность создать в проекте новые типы данных с ограниченным диапазоном значений либо задать диапазон непосредственно при объявлении переменной.

Создание нового типа выглядит так:

```
TYPE < Имя > : < Целый тип > (<от>..<до>) END_TYPE;
```

< Имя > любой допустимый МЭК идентификатор,

<Целый тип> один из типов SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD).

< от > константа, определяющая начало диапазона значений включительно.

< до > константа, определяющая конец диапазона значений включительно.

Пример:

```
TYPE
    SubInt : INT (-4095..4095);
END_TYPE
```

Ограничение диапазона при объявлении переменной:

```
VAR
    i : INT (-4095..4095);
    ui : UINT (0..10000);
END_VAR
```

При попытке присвоить переменной с ограниченным диапазоном константы, не попадающей в заданный диапазон (например `i := 5000;`), CoDeSys даст сообщение об ошибке.

Для контроля значений «ограниченных» переменных в процессе исполнения применяются функции **CheckRangeSigned** или **CheckRangeUnsigned**. Они позволяют обрабатывать ошибки произвольным образом. Например, ограничить присваиваемое значение или сформировать флаги ошибки. Первая функция работает для переменных со знаком, вторая для переменных без знака (unsigned). Изменять идентификаторы этих функций нельзя.

Пример:

Здесь применяется функция *CheckRangeSigned*, контролирующая переменные со знаком (как, например, объявленная выше `i`). Функция «обрезает» присваиваемые значения по границам диапазона.

```
FUNCTION CheckRangeSigned : DINT
    VAR_INPUT
        value, lower, upper: DINT;
    END_VAR

    IF (value < lower) THEN
        CheckRangeSigned := lower;
    ELSIF(value > upper) THEN
```

```

CheckRangeSigned := upper;
ELSE
    CheckRangeSigned := value;
END_IF
    
```

Функция вызывается автоматически при соответствующих операциях присваивания. Она получает три параметра: присваиваемое значение (*value*) и обе границы диапазона (*lower*, *upper*). Фактически присваивается возвращаемое *CheckRangeSigned* значение.

Например, при присваивании *i:=10*y* происходит неявный вызов:

```
i := CheckRangeSigned(10*y, -4095, 4095);
```

В результате, даже если *y > 1000*, *i* не получит значение более 4095.

Аналогично объявляется и функция *CheckRangeUnsigned*:

```

FUNCTION CheckRangeUnsigned : UDINT
VAR_INPUT
    value, lower, upper: UDINT;
END_VAR
    
```

Замечание: Если функции *CheckRangeSigned* или *CheckRangeUnsigned* не включены в проект, ограничение границ диапазонов во время исполнения работать не будет.

Внимание: Если функции «обрезают» значения, как в вышеописанном примере существует опасность получить бесконечный цикл. Это может случиться, если условие окончания итераций цикла не достигается из-за искусственного ограничения значений переменной!

Пример. Переменная *ui* не превысит 10000, и цикл FOR ни когда не закончится:

```

VAR
    ui : UINT (0..10000);
END_VAR
FOR ui:=0 TO 10000 DO
...
END_FOR
    
```

Внимание: Функции *CheckRange...*, содержащиеся в библиотеке *Check.Lib*, представляют собой пример реализации. Прежде чем использовать эту библиотеку, убедитесь, что данные функции работают так, как нужно в вашем случае, либо создайте собственные функции непосредственно в вашем проекте.

Приложение D: Библиотеки CoDeSys

10.16 Стандартная библиотека Standard.lib

10.16.1 Строковые функции

Внимание: В многозадачных проектах необходимо сосредоточить всю работу со строками в одной задаче. При одновременном вызове одной и той же строковой функции из нескольких параллельных процессов возможно нарушение ее работы (перекрытие данных).

LEN

Возвращает длину строки.

Аргумент STR типа STRING, возвращаемое значение типа INT.

Пример IL:

```
LD          'SUSI'
LEN
ST          VarINT1 (* Результат: 4 *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := LEN ('SUSI');
```

LEFT

Возвращает левую значимую часть строки заданной длины.

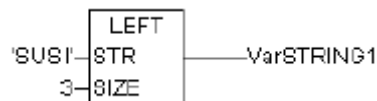
Входная строка STR типа STRING, размер SIZE типа INT, возвращаемое значение STRING.

LEFT (STR, SIZE) означает: взять первых SIZE символов от строки STR.

Пример IL:

```
LD          'SUSI'
LEFT        3
ST          VarSTRING1 (* Результат 'SUS' *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := LEFT ('SUSI',3);
```

RIGHT

Возвращает правую значимую часть строки заданной длины.

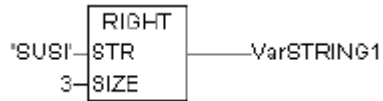
Входная строка STR типа STRING, размер SIZE типа INT, возвращаемое значение STRING.

RIGHT (STR, SIZE) означает: взять последних SIZE символов от строки STR

Пример IL:

```
LD          'SUSI'
RIGHT      3
ST          VarSTRING1    (*Результат: 'USI' *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := RIGHT ('SUSI',3);
```

MID

Возвращает часть строки с указанной позиции указанной длины.

Входная строка STR типа STRING, размер LEN и POS типа INT, возвращаемое значение STRING.

MID (STR, LEN, POS) означает: вырезать LEN символов из STR строки, начиная с POS.

Пример IL:

```
LD          'SUSI'
RIGHT      2,2
ST          VarSTRING1    (* Результат: 'US' *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := MID ('SUSI',2,2);
```

CONCAT

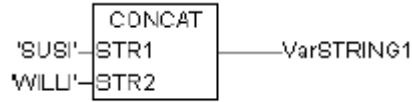
Конкатенация (объединение) двух строк.

Обе входных строки STR1 и STR2 как и результат типа STRING.

Пример IL:

```
LD          'SUSI'
CONCAT     'WILLI'
ST          VarSTRING1    (* Результат: 'SUSIWILLI' *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := CONCAT ('SUSI', 'WILLI');
```

Обратите внимание: Функция CONCAT не работает при более чем 5 вложениях.

INSERT

INSERT вставляет строку в указанную позицию другой строки.

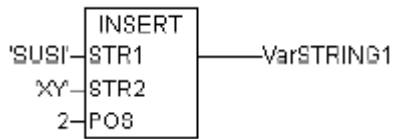
Входные переменные STR1 и STR2 - типа STRING, POS - типа INT, возвращаемое значение - строка STRING.

INSERT(STR1, STR2, POS) означает: вставить STR2 в STR1 в позиции POS.

Пример IL:

```
LD          'SUSI'
INSERT     'XY',2
ST          VarSTRING1    (* Результат: 'SUXYSI' *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := INSERT ('SUSI', 'XY', 2);
```

DELETE

DELETE удаляет часть строки с указанной позиции.

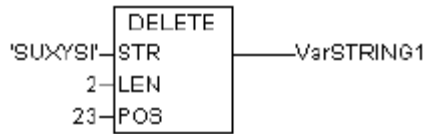
Входные переменные STR типа STRING, LEN и POS типа INT, возвращаемое значение строка STRING.

DELETE(STR, L, P) означает: удалить L символов из STR, начиная с позиции P.

Пример IL:

```
LD          'SUXYSI'
DELETE     2,23
ST          Var1          (* Результат: 'SUSI' *)
```

Пример FBD:



Пример ST:

```
Var1 := DELETE ('SUXYSI',2,3);
```

REPLACE

REPLACE заменяет часть строки другой строкой с указанной позиции заданной длины.

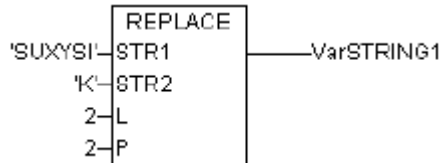
Входные переменные STR1 и STR2 типа STRING, LEN и POS типа INT, возвращаемое значение строка STRING.

REPLACE(STR1, STR2, L, P) означает: заменить L символов строки STR1 на STR2 начиная с позиции P.

Пример IL:

```
LD      'SUXYSI'
REPLACE 'K', 2,2
ST      VarSTRING1    (* Результат: 'SKYSI' *)
```

Пример FBD:



Пример ST:

```
VarSTRING1 := REPLACE ('SUXYSI', 'K', 2,2);
```

FIND

FIND ищет заданный контекст в строке.

Входные переменные STR1 и STR2 типа STRING, возвращаемое значение INT.

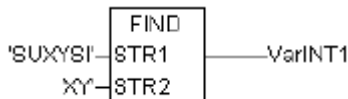
FIND(STR1, STR2) означает: найти позицию в строке STR1, где впервые встречается подстрока STR2.

Нумерация позиций в строке начинается с 1. Если STR2 не найдена, STR1 возвращает 0.

Пример IL:

```
LD      'SUXYSI'
FIND    'XY'
ST      VarINT1        (* Результат: '3' *)
```

Пример FBD:



Пример ST:

VarINT1 := FIND ('SUXYSI', 'XY');

10.16.2 Переключатели

SR

Переключатель с доминантой включения:

Q1 = SR (SET1, RESET) означает:

$Q1 = (\text{NOT RESET AND } Q1) \text{ OR SET1}$

Входные переменные SET1 и RESET - как и выходная переменная Q1 типа BOOL.

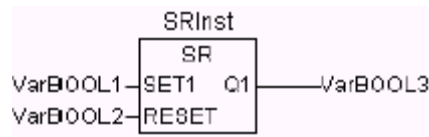
Пример объявления:

SRInst : SR ;

Пример IL:

```
CAL SRInst(SET1 := VarBOOL1, RESET := VarBOOL2)
LD SRInst.Q1
ST VarBOOL3
```

Пример FBD:



Пример ST:

```
SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2 );
VarBOOL3 := SRInst.Q1 ;
```

RS

Переключатель с доминантой выключения:

Q1 = RS (SET, RESET1) означает:

$Q1 = \text{NOT RESET1 AND } (Q1 \text{ OR SET})$

Входные переменные SET и RESET1 - как и выходная переменная Q1 типа BOOL.

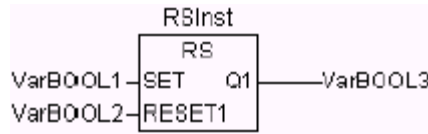
Пример объявления:

RSInst : RS ;

Пример IL:

```
CAL RSInst(SET := VarBOOL1, RESET1 := VarBOOL2)
LD RSInst.Q1
ST VarBOOL3
```

Пример FBD:



Пример ST:

```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 );
VarBOOL3 := RSInst.Q1 ;
```

SEMA

Программный семафор.

BUSY = SEMA(CLAIM, RELEASE) означает:

```
BUSY := X;
IF CLAIM THEN X:=TRUE;
ELSE IF RELEASE THEN BUSY := FALSE; X:= FALSE;
END_IF
```

X - это внутренняя BOOL переменная, изначально имеющая значение FALSE.

Входные переменные CLAIM и RELEASE - как и выходная переменная BUSY типа BOOL. (CLAIM – запрос захвата, RELEASE - освобождение)

Семафор предназначен для организации асинхронного доступа к одному аппаратному ресурсу. Если при вызове семафора с CLAIM = TRUE возвращаемое значение BUSY = FALSE, то ресурс свободен (запрашивается впервые или уже освобожден вызовом RELEASE = TRUE). Возвращаемое значение BUSY = FALSE, это означает, что ресурс занят.

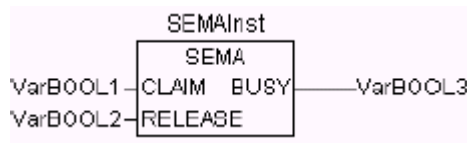
Пример объявления:

```
SEMAInst : SEMA ;
```

Пример IL:

```
CAL SEMAInst(CLAIM := VarBOOL1, RELEASE := VarBOOL2)
LD SEMAInst.BUSY
ST VarBOOL3
```

Пример FBD:



Пример ST:

```
SEMAInst(CLAIM:= VarBOOL1 , RELEASE:=VarBOOL2 );
VarBOOL3 := SEMAInst.BUSY;
```

10.16.3 Детекторы импульсов

R_TRIG

Функциональный блок R_TRIG генерирует импульс по переднему фронту входного сигнала.


```

FUNCTION_BLOCK R_TRIG
VAR_INPUT
    CLK : BOOL;
END_VAR
VAR_OUTPUT
    Q : BOOL;
END_VAR
VAR
    M : BOOL := FALSE;
END_VAR
    Q := CLK AND NOT M;
    M := CLK;
END_FUNCTION_BLOCK
    
```

Выход **Q** равен FALSE до тех пор, пока вход **CLK** равен FALSE. Как только **CLK** получает значение TRUE, **Q** устанавливается в TRUE. При следующем вызове функционального блока выход сбрасывается в FALSE. Таким образом, блок выдает единичный импульс при каждом переходе **CLK** из FALSE в TRUE.

Пример объявления:

```
RTRIGInst : R_TRIG ;
```

Пример IL:

```

CAL RTRIGInst(CLK := VarBOOL1)
LD RTRIGInst.Q
ST VarBOOL2
    
```

Пример FBD:



Пример ST:

```

RTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
    
```

F_TRIG

Функциональный блок F_TRIG генерирует импульс по заднему фронту входного сигнала.

```

FUNCTION_BLOCK F_TRIG
VAR_INPUT
    CLK: BOOL;
END_VAR
VAR_OUTPUT
    Q: BOOL;
END_VAR
    
```

```

VAR
    M: BOOL := FALSE;
END_VAR
    Q := NOT CLK AND NOT M;
    M := NOT CLK;
END_FUNCTION_BLOCK
    
```

Выход **Q** равен **FALSE** до тех пор, пока вход **CLK** равен **TRUE**. Как только **CLK** получает значение **FALSE**, **Q** устанавливается в **TRUE**. При следующем вызове функционального блока выход сбрасывается в **FALSE**. Таким образом, блок выдает единичный импульс при каждом переходе **CLK** из **TRUE** в **FALSE**.

Пример объявления:

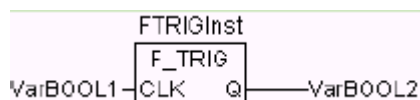
```
FTRIGInst : F_TRIG ;
```

Пример IL:

```

CAL  FTRIGInst(CLK := VarBOOL1)
LD   FTRIGInst.Q
ST   VarBOOL2
    
```

Пример FBD:



Пример ST:

```

FTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := FTRIGInst.Q;
    
```

10.16.4 Счетчики

CTU

Функциональный блок 'инкрементный счетчик'.

Входы **CU**, **RESET** и выход **Q** типа **BOOL**, вход **PV** и выход **CV** типа **WORD**.

По каждому фронту на входе **CU** (переход из **FALSE** в **TRUE**) выход **CV** увеличивается на 1. Выход **Q** устанавливается в **TRUE**, когда счетчик достигнет значения заданного **PV**. Счетчик **CV** сбрасывается в 0 по входу **RESET = TRUE**.

Пример объявления:

```
CTUInst : CTU ;
```

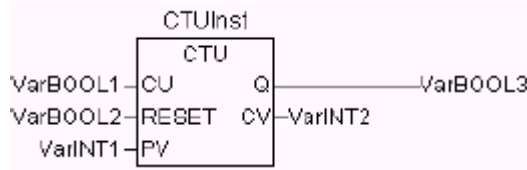
Пример IL:

```

CAL  CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV := VarINT1)
LD   CTUInst.Q
ST   VarBOOL3
LD   CTUInst.CV
    
```

ST VarINT2

Пример FBD:



Пример ST:

```
CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTUInst.Q ;
VarINT2 := CTUInst.CV;
```

CTD

Функциональный блок 'декрементный счетчик'.

Входы CD, LOAD и выход Q типа BOOL, вход PV и выход CV типа WORD.

По каждому фронту на входе CD (переход из FALSE в TRUE) выход CV уменьшается на 1. Когда счетчик достигнет 0, счет останавливается, выход Q переключается в TRUE. Счетчик CV загружается начальным значением, равным PV по входу LOAD = TRUE.

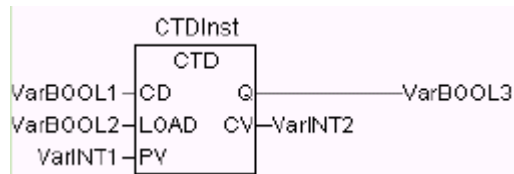
Пример объявления:

```
CTDInst : CTD ;
```

Пример IL:

```
CAL CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV := VarINT1)
LD CTDInst.Q
ST VarBOOL3
LD CTDInst.CV
ST VarINT2
```

Пример FBD:



Пример ST:

```
CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTDInst.Q ;
VarINT2 := CTDInst.CV;
```

CTUD

Функциональный блок 'инкрементный / декрементный счетчик'.

Входы CU, CD, RESET, LOAD и выходы QU и QD типа BOOL, PV и CV типа WORD.

По входу RESET счетчик CV сбрасывается в 0, по входу LOAD загружается значением PV.

По фронту на входе CU счетчик увеличивается на 1. По фронту на входе CD счетчик уменьшается на 1 (до 0).

QU устанавливается в TRUE, когда CV больше или равен PV.

QD устанавливается в TRUE, когда CV равен 0.

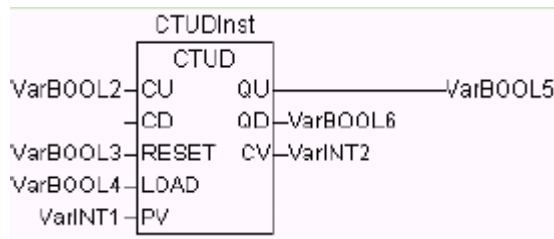
Пример объявления:

```
CTUDInst : CUTD ;
```

Пример IL:

```
CAL CTUDInst(CU := VarBOOL2, RESET := VarBOOL3, LOAD :=
    VarBOOL4, PV := VarINT1)
LD CTUDInst.QU
ST VarBOOL5
LD CTUDInst.QD
ST VarBOOL6
LD CTUDInst.CV
ST VarINT2
```

Пример FBD:



Пример ST:

```
CTUDInst(CU := VarBOOL1, CU:= VarBOOL2, RESET := VarBOOL3,
LOAD:=VarBOOL4 , PV:= VarINT1);
VarBOOL5 := CTUDInst.QU ;
VarBOOL6 := CTUDInst.QD ;
VarINT2 := CTUDInst.CV;
```

10.16.5 Таймеры

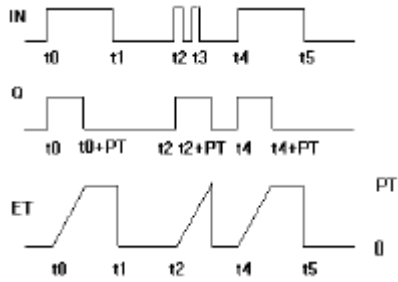
TP

Функциональный блок 'таймер'.

TP(IN, PT, Q, ET) Входы IN и PT типов BOOL и TIME соответственно. Выходы Q и ET аналогично типов BOOL и TIME.

Пока IN равен FALSE, выход Q = FALSE, выход ET = 0. При переходе IN в TRUE выход Q устанавливается в TRUE и таймер начинает отсчет времени (в миллисекундах) на выходе ET до достижения длительности, заданной PT. Далее счетчик не увеличивается. Таким образом, выход Q генерирует импульс длительностью PT по фронту входа IN.

Временная диаграмма работы TP:



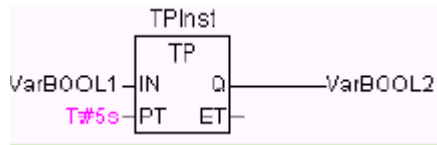
Пример объявления:

```
TPInst : TP;
```

Пример IL:

```
CAL TPInst(IN := VarBOOL1, PT := T#5s)
LD TPInst.Q
ST VarBOOL2
```

Пример FBD:



Пример ST:

```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TPInst.Q;
```

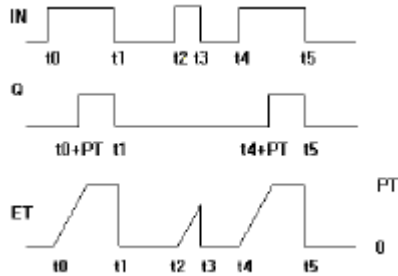
TON

Функциональный блок ‘таймер с задержкой включения’.

TON(IN, PT, Q, ET) Входы IN и PT типов BOOL и TIME соответственно. Выходы Q и ET аналогично типов BOOL и TIME.

Пока IN равен FALSE, выход Q = FALSE, выход ET = 0. Как только IN становится TRUE, начинается отсчет времени (в миллисекундах) на выходе ET до значения, равного PT. Далее счетчик не увеличивается. Q равен TRUE, когда IN равен TRUE и ET равен PT, иначе FALSE. Таким образом, выход Q устанавливается с задержкой PT от фронта входа IN.

Временная диаграмма работы TON:



Пример объявления:

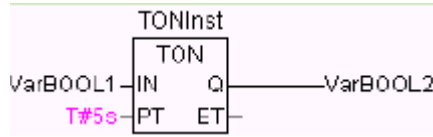
```
TONInst : TON ;
```

Пример IL:

```

CAL TONInst(IN := VarBOOL1, PT := T#5s)
LD TONInst.Q
ST VarBOOL2
    
```

Пример FBD:



Пример ST:

```

TONInst(IN := VarBOOL1, PT:= T#5s);
    
```

ТОФ

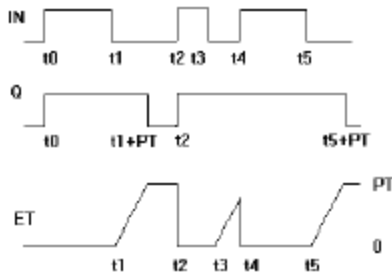
Функциональный блок ‘таймер с задержкой выключения’.

TOF(IN, PT, Q, ET) Входы IN и PT типов BOOL и TIME соответственно. Выходы Q и ET аналогично типов BOOL и TIME.

Если IN равен TRUE, то выход Q = TRUE и выход ET = 0. Как только IN переходит в FALSE, начинается отсчет времени (в миллисекундах) на выходе ET. При достижении заданной длительности отсчет останавливается. Выход Q равен FALSE, если IN равен FALSE и ET равен PT, иначе - TRUE.

Таким образом, выход Q сбрасывается с задержкой PT от спада входа IN.

Временная диаграмма работы TOF:



Пример объявления:

```

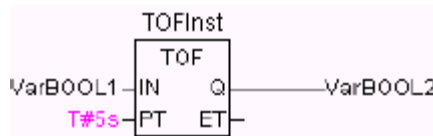
TOFInst : TOF ;
    
```

Пример IL:

```

CAL TOFInst(IN := VarBOOL1, PT := T#5s)
LD TOFInst.Q
ST VarBOOL2
    
```

Пример FBD:



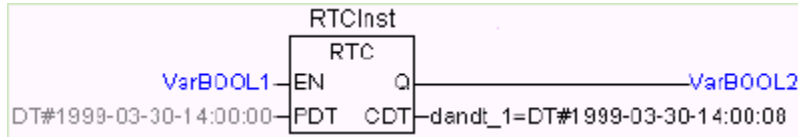
Пример ST:

```

TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TOFInst.Q;
    
```

RTC

Функциональный блок ‘часы реального времени’. После установки начального значения, RTC выдает текущие время и дату.



RTC(EN, PDT, Q, CDT) Входы EN и PDT, выходы Q и CDT типов BOOL и DATE_AND_TIME соответственно.

Пока EN равен FALSE, выход Q равен FALSE и CDT равен DT#1970-01-01-00-00:00:00.

При включении EN = TRUE в часы загружается время PD и начинается отсчет времени. На выходе CDT. Если EN перейдет в FALSE, CDT сбросится в начальное значение DT#1970-01-01-00-00:00:00. Обратите внимание, что установка времени PDT происходит по фронту.

10.17 Библиотека UTIL.LIB

Данная библиотека содержит дополнительный набор различных функций и функциональных блоков, применяемых для BCD и бит/байт преобразований, дополнительных математических функций, а также регуляторов, генераторов и преобразований аналоговых сигналов.

Специальная версия этой библиотеки UTIL_NO_REAL не содержит функций и функциональных блоков, использующих переменные типа REAL.

10.17.1 BCD преобразования

Байт, представленный в формате BCD, содержит числа от 0 до 99. Каждый десятичный знак занимает 4 бита. Биты 4-7 содержат первую цифру – число десятков. Формат BCD подобен шестнадцатеричному представлению с ограничением диапазона чисел 0..99 вместо 0..FF.

Например: Преобразуем число 51 в BCD формат. 5 - это двоичное 0101, 1 - это 0001. В результате получается байт 0101_0001.

BCD_TO_INT

Функция преобразует байт формата BCD в число типа INT.

Входной параметр функции типа BYTE и выход типа INT. Если входное значение не укладывается в формат BCD, функция возвращает -1.

Примеры ST:

```
i:=BCD_TO_INT(73); (* Результат 49 *)
k:=BCD_TO_INT(151); (*Результат 97 *)
l:=BCD_TO_INT(15); (* -1, потому что F0 не BCD формат*)
```

INT_TO_BCD

Функция преобразует INTEGER число в байт формата BCD.

Входной параметр функции типа INT и выход типа BYTE.

Если INTEGER число не может быть представлено в BCD формате, то функция возвращает значение 255.

Примеры ST:

```
i:= INT_TO_BCD(49); (*Результат 73 *)
k:=INT_TO_BCD (97); (*Результат 151 *)
l:= INT_TO_BCD (100);(* Ошибка! Выход: 255 *)
```

10.17.2 Бит/байт функции

EXTRACT

Параметры функции: DWORD X и BYTE N. Выход типа BOOL, отражает значение бита N в числе X. Биты нумеруются с 0.

Примеры ST:

```
FLAG:=EXTRACT(X:=81,N:=4); (* Результат: TRUE, 81 это 1010001, 4й бит 1 *)
FLAG:=EXTRACT(X:=33, N:=0); (* Результат: TRUE, 33 это 100001,бит '0' это 1 *)
```

PACK

Функция сворачивает восемь параметров B0, B1, ..., B7 типа BOOL в один BYTE.

Функциональный блок UNPACK выполняет обратную распаковку.

PUTBIT

Параметры функции: DWORD X, BYTE N и BOOL B.

PUTBIT устанавливает N-й бит числа X в состояние, заданное B. Биты нумеруются с 0.

Примеры ST:

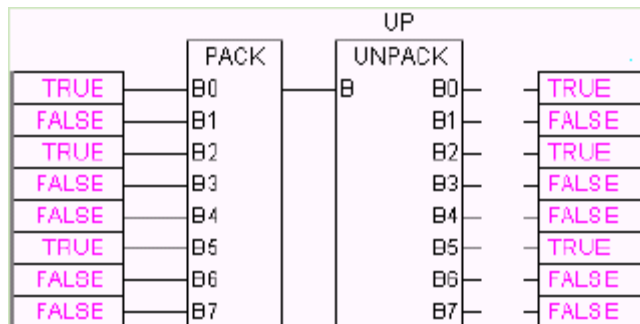
```
A:=38; (* двоичное 100110 *)
B:=PUTBIT(A,4,TRUE); (* Результат : 54 = 2#110110 *)
C:=PUTBIT(A,1,FALSE); (* Результат : 36 = 2#100100 *)
```

UNPACK

UNPACK преобразует вход B типа BYTE в 8 выходов B0,...,B7 типа BOOL.

Обратная упаковка производится с помощью PACK.

Пример FBD:



10.17.3 Дополнительные математические функции

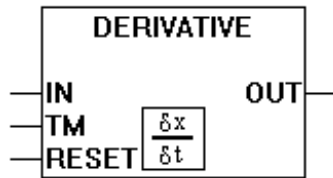
DERIVATIVE

Функциональный блок выполняет численное дифференцирование.

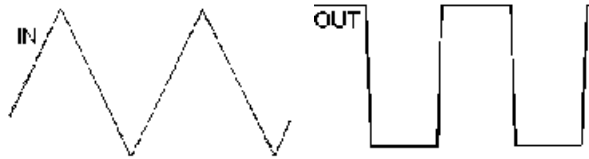
Аналоговый вход IN и выход OUT типа REAL. Вход TM задает время дифференцирования (как правило, в миллисекундах) имеет тип DWORD. В процессе сброса (RESET = TRUE) выход OUT равен нулю.

Алгоритм DERIVATIVE проводит аппроксимацию по четырем точкам, что снижает ошибки при наличии шума во входном сигнале.

Блок FBD:



Пример дифференцирования треугольных импульсов:



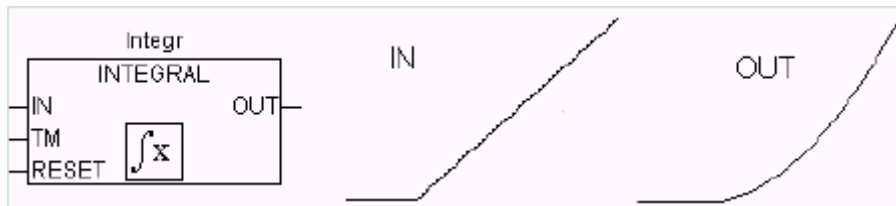
INTEGRAL

Функциональный блок выполняет численное интегрирование.

Аналоговый вход IN типа REAL. Вход TM типа DWORD задает длительность интегрирования (как правило, в миллисекундах). Вход RESET типа BOOL запускает интегрирование при установке в TRUE. Выход OUT типа REAL.

Алгоритм вычисления использует классический двухточечный метод трапеций.

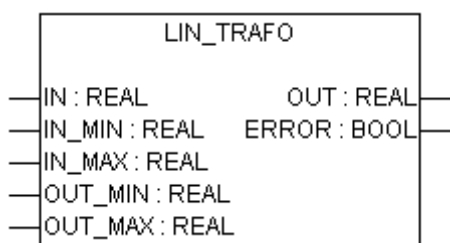
Блок FBD: пример интегрирования линейной функции:



LIN_TRAFO

Данный функциональный блок (util.lib) преобразует значение переменной REAL, принадлежащее одному интервалу в пропорциональное значение, принадлежащее другому интервалу. Интервалы определяются минимальным и максимальным значением. Алгоритм преобразования опирается на следующее равенство:

$$(IN - IN_MIN) : (IN_MAX - IN) = (OUT - OUT_MIN) : (OUT_MAX - OUT)$$



Входные переменные:

Переменная	Тип данных	Описание
IN	REAL	Входное значение
IN_MIN	REAL	Нижнее значение входного диапазона
IN_MAX	REAL	Верхнее значение входного диапазона
OUT_MIN	REAL	Нижнее значение выходного диапазона
OUT_MAX	REAL	Верхнее значение выходного диапазона

Выходные переменные:

Переменная	Тип данных	Описание
OUT	REAL	Выходное значение
ERROR	BOOL	Признак ошибки: TRUE, если IN_MIN = IN_MAX или если значение IN вышло за пределы входного диапазона

Пример использования:

Допустим, датчик температуры выдает некоторое напряжение в вольтах (вход IN). Нам необходимо преобразовать полученное значение в градусы по Цельсию (выход OUT). Входной диапазон (в Вольтах) определяется пределами IN_MIN=0 и IN_MAX=10. Выходной диапазон (в градусах Цельсия) определяется соответствующими пределами OUT_MIN=-20 и OUT_MAX=40.

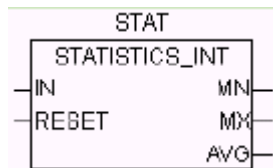
Так, при входном значении 5 Вольт, мы получим на выходе 10 градусов по Цельсию.

STATISTICS_INT

Функциональный блок определяет минимальное, максимальное и среднее значения входной величины.

Аналоговый вход IN типа INT. По входу RESET типа BOOL все переменные инициализируются заново. Выход MN дает минимальное, выход MX максимальное и выход AVG среднее значения входных данных IN. Все три выхода типа INT.

Блок FBD:



STATISTICS_REAL

Функциональный блок, аналогичный STATISTICS_INT. Вход IN и выходы MN, MX, AVG имеют тип REAL.

VARIANCE

Функциональный блок вычисляет дисперсию входных данных.

Вход IN типа REAL, вход RESET типа BOOL и выход OUT типа REAL. Сброс вычисления производится по входу RESET=TRUE.

Среднеквадратичное отклонение может быть получено как квадратный корень VARIANCE.

10.17.4 Регуляторы

PD

Функциональный блок реализует ПД закон регулирования:

$$Y = Y_OFFSET + KP \left(e(t) + TV \frac{de(t)}{dt} \right)$$

где Y_OFFSET – стационарное значение, KP – коэффициент передачи, TV – постоянная дифференцирования (ms), e(t) - сигнал ошибки (SET_POINT-ACTUAL).

Входы ACTUAL, SET_POINT, KP, Y_OFFSET, Y_MIN Y_MAX типа REAL. Вход TV типа DWORD, RESET и MANUAL типа BOOL.

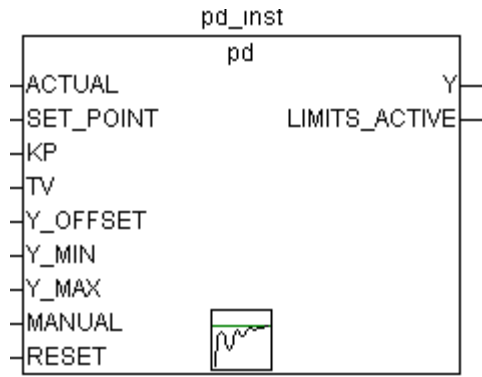
Выходы Y – REAL, LIMITS_ACTVE типа BOOL.

Значение выхода Y ограничено Y_MIN и Y_MAX. При достижении Y границ ограничения, выход LIMITS_ACTVE, (BOOL) принимает значение TRUE. Если ограничение выхода не требуется, Y_MIN и Y_MAX должны быть равны 0.

Если MANUAL равен TRUE, то регулирование выключено, значение Y автоматически не изменяется. При переходе значения MANUAL в FALSE происходит рестарт регулятора.

P-регулятор получается из PD установкой TV в 0.

Пример FBD:



PID

Функциональный блок реализует ПИД закон регулирования:

$$Y = Y_OFFSET + KP \left(e(t) + \frac{1}{TN} \int_0^{TN} e(t) + TV \frac{de(t)}{dt} \right)$$

где Y_OFFSET – стационарное значение, KP – коэффициент передачи, TN – постоянная интегрирования (ms), TV – постоянная дифференцирования (ms), e(t) - сигнал ошибки (SET_POINT-ACTUAL).

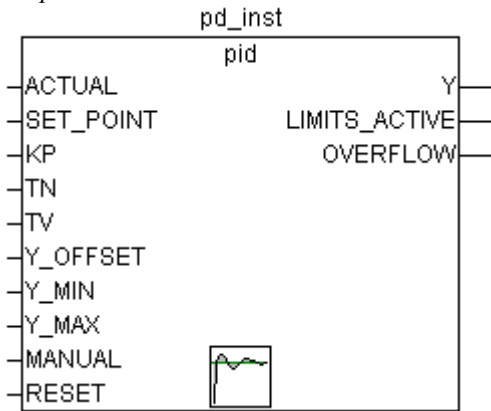
Входы ACTUAL, SET_POINT, KP, Y_OFFSET, Y_MIN Y_MAX типа REAL. Входы TN и TV типа DWORD, RESET и MANUAL типа BOOL.

Выходы Y – REAL, LIMITS_ACTIVE и OVERFLOW типа BOOL.

Механизм ограничения выхода PID аналогичен PD регулятору.

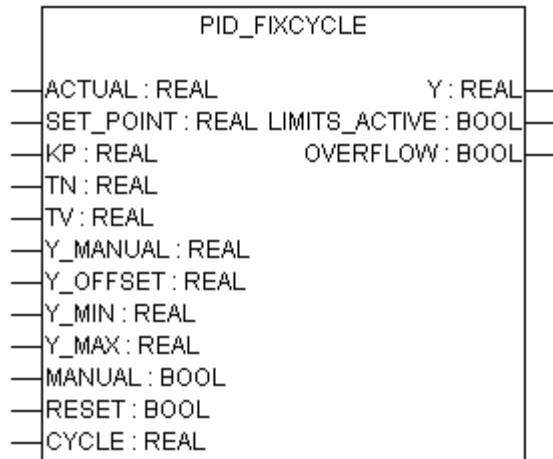
Неправильная настройка регулятора может вызвать неограниченный рост интегральной составляющей. Для обнаружения такой ситуации предназначен выход OVERFLOW. При переполнении он принимает значение TRUE, одновременно останавливается работа регулятора. Для его включения необходимо использовать рестарт.

Пример FBD:



PID_FIXCYCLE

Функциональный блок PID_FIXCYCLE.



Отличается от PID тем, что время цикла не измеряется автоматически встроенным таймером, а задается дополнительной переменной CYCLE, в секундах.

10.17.5 Генераторы сигналов

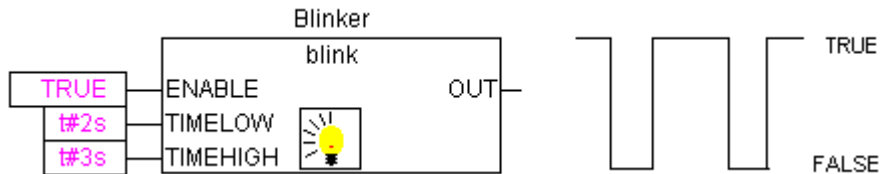
BLINK

Функциональный блок 'генератор прямоугольных импульсов'

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

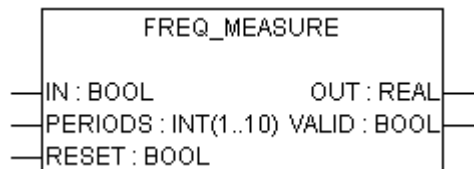
Генератор запускается по входу ENABLE = TRUE. Длительность импульса задается TIMEHIGH, длительность паузы TIMELOW.

Пример CFC:



FREQ_MEASURE

Данный функциональный блок измеряет (усредненную) частоту (в Герцах) входного сигнала типа BOOL. Вы можете задать количество периодов для усреднения. Под периодом понимается время между двумя передними фронтами сигнала.



Входные переменные:

Переменная	Тип данных	Описание
IN	BOOL	Входной сигнал
PERIODS	INT	Число периодов усреднения. Допустимое значение от 1 до 10.
RESET	BOOL	Сброс

Выходные переменные:

Переменная	Тип данных	Описание
OUT	REAL	Результат, частота в Герцах
VALID	BOOL	FALSE до окончания первого замера, либо период > 3*OUT (признак ошибки по входам)

GEN

Функциональный блок 'функциональный генератор'

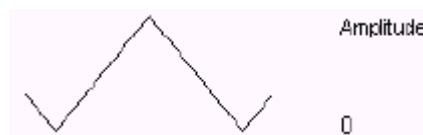
Входы: перечисление MODE predetermined типа GEN_MODE, BASE типа BOOL, PERIOD типа TIME, CYCLES и AMPLITUDE типа INT и RESET типа BOOL. Выход OUT типа INT.

Вход MODE задает вид генерируемой функции. Перечисление включает следующие значения: TRIANGLE и TRIANGLE_POS - треугольники, SAWTOOTH_RISE и SAWTOOTH_FALL - пила, RECTANGLE - прямоугольники, SINE и COSINE - синусоиды:

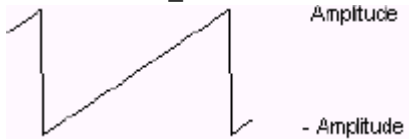
TRIANGLE:



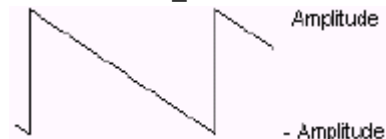
TRIANGLE_POS:



SAWTOOTH_RISE:



SAWTOOTH_FALL:



RECTANGLE:



SINUS:



COSINUS:

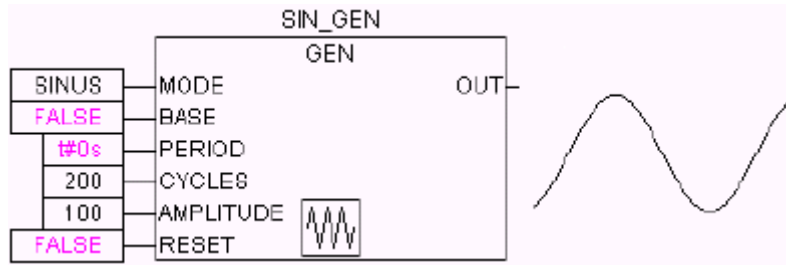


BASE определяет представление единиц периода по времени (BASE=TRUE) или по числу циклов, т.е. по количеству вызовов функционального блока (BASE=FALSE).

Входы PERIOD или CYCLES определяют период выходного сигнала. Вход AMPLITUDE задает амплитуду сигнала.

Сброс генератора происходит при установке RESET=TRUE.

Пример FBD:



10.17.6 Преобразования аналоговых сигналов

CHARCURVE

Функциональный блок осуществляет пересчет входных данных по заданной переходной функции- путем кусочно-линейной аппроксимации.



Вход IN типа INT принимает исходные данные. Вход N типа BYTE определяет количество точек задающих передаточную функцию. Передаточная функция задается массивом точек ARRAY P[0..10], где P - это точка, определенная как структура типа POINT, состоящая из двух переменных INT X и Y.

Выход OUT типа INT, выходные данные. Выход ERR типа BYTE, индикатор ошибки.

Точки P[0]..P[N-1] массива ARRAY должны быть отсортированы по X в порядке возрастания, в противном случае ERR получает значение 1. Если вход IN не лежит в пределах от P[0].X до P[N-1].X, генерируется ошибка ERR=2 и выход OUT приобретает значение соответствующего предела P[0].X или P[N-1].X.

Число N должно быть в пределах от 2 до 11, иначе возникает ошибка ERR=4.

Пример ST:

Прежде всего определим массив ARRAY P:

VAR

...

```
CHARACTERISTIC_LINE:CHARCURVE;
```

```
KL:ARRAY[0..10] OF POINT:= (X:=0,Y:=0), (X:=250,Y:=50), (X:=500,Y:=150),  
(X:=750,Y:=400), 7((X:=1000,Y:=1000));
```

```
COUNTER:INT;
```

```
...
```

```
END_VAR
```

Далее вызываем CHARCURVE для линейно возрастающих значений:

```
COUNTER:=COUNTER+10;
```

```
CHARACTERISTIC_LINE(IN:=COUNTER,N:=5,P:=KL);
```

Последующая трассировка иллюстрирует полученный эффект:



RAMP_INT

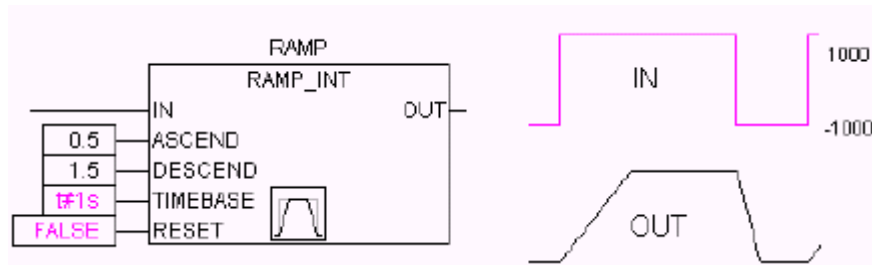
Функциональный блок RAMP_INT ограничивает скорость нарастания и спада сигнала.

Три входа имеют тип INT: IN, входные данные, ASCEND и DESCEND, максимальное нарастание и спад за интервал, заданный TIMEBASE типа TIME. Установка двоичного входа RESET в TRUE вызывает сброс RAMP_INT в начальное состояние.

Выход OUT типа INT, выходные данные.

Если TIMEBASE равен t#0s, ASCEND и DESCEND задают ограничение изменения за один цикл (вызов блока) безотносительно времени.

Пример FBD:



RAMP_REAL

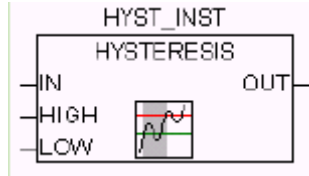
RAMP_REAL аналогичен RAMP_INT, за исключением того, что входы IN, ASCEND, DESCEND и выход OUT типа REAL.

10.17.7 Аналоговые компараторы

HYSTERESIS

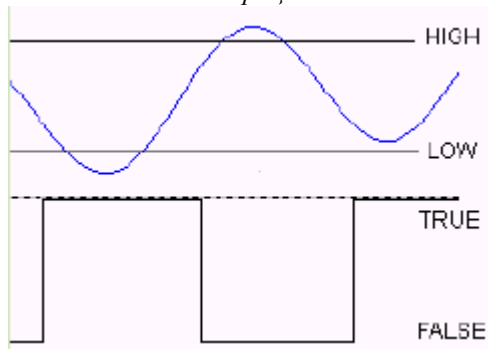
Аналоговый компаратор с гистерезисом.

Входы IN, HIGH и LOW типа INT. Выход OUT типа BOOL.



Если вход IN принимает значение, меньшее LOW, выход OUT устанавливается в TRUE. Если вход IN принимает значение, большее HIGH, то выход равен FALSE. В пределах от LOW до HIGH значение выхода не изменяется.

Пояснительная иллюстрация:



LIMITALARM

Функциональный блок, контролирует принадлежность значения входа IN заданному диапазону. Входы LOW и HIGH задают границу диапазона..

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

Если значение на входе IN:

превышает предел HIGH	выход O = TRUE
меньше предела LOW	выход U = TRUE
лежит в пределах между LOW и HIGH (включительно)	выход IL = TRUE

Пример FBD:



10.18 Библиотека AnalyzationNew.lib

Данная библиотека содержит модули для анализа выражений в SFC. Если сложное выражение дает FALSE, то библиотека позволяет уточнить, какие компоненты условного выражения дали такой результат. Существует специальный флаг `SFCErrorAnalyzationTable`, механизм его работы неявно использует данные функции для анализа условий переходов.

Пример условного выражения:

```
b OR NOT(y < x) OR NOT (NOT d AND e)
```

Во всех модулях используются следующие переменные:

InputExpr: BOOL, анализируемое выражение

DoAnalyze: BOOL, TRUE запускает анализ

ExpResult: BOOL, текущее значение выражения

Функции:

AnalyzeExpression возвращает строку, содержащую компоненты выражения, дающие в итоге значение FALSE. Для этого служит вспомогательная функция **AppendErrorString** добавляющая компоненты, разделенные символом "|".

Выходная строка **OutString** (тип STRING) содержит результат (например: `y < x | d`).

Функция **AnalyzeExpressionTable** записывает компоненты выражения, дающие в итоге значение FALSE, в массив. Для каждого компонента заполняется структура **ExpressionResult**, содержащая наименование, адрес, комментарий и текущее значение.

```
OutTable: ARRAY [0..15] OF ExpressionResult;
```

Например:

```

└─ SFCErrorAnalyzationTable
  └─ SFCErrorAnalyzationTable[0]
    ├── name = 'b'
    ├── address = '%Mx0.0'
    ├── comment = 'false'
    └── value = FALSE
  └─ SFCErrorAnalyzationTable[1]
  └─ SFCErrorAnalyzationTable[2]
  └─ SFCErrorAnalyzationTable[3]

```

AnalyzeExpressionCombined совмещает функции **AnalyzeExpression** и **AnalyzeExpressionTable**.

10.19 Системные библиотеки CoDeSys

В зависимости от целевой платформы применяются разные системные библиотеки. Функции системных библиотек описаны в документе `SysLibs_Overview.pdf`.

Приложение Е: Краткий справочник по операторам и компонентам библиотек

Приведенные ниже таблицы кратко представляют операторы CoDeSys и компоненты библиотек **Standard.lib** и **Util.lib**. Даны нотации для языков ST и IL. Для IL указаны допустимые модификаторы.

Обратите внимание, что для IL инструкций первый операнд должен быть загружен заранее (например, командой LD). Непосредственно в строке за «IL» командой вводятся второй и последующие (если они есть) операнды.

Столбец «Мод.» содержит допустимые IL модификаторы:

- C** Команда выполняется только в случае, если результат предыдущей операции TRUE.
- N** для JMP, CALC, RETC: Команда выполняется только в случае, если результат предыдущей операции FALSE.
- N** прочие: отрицание операнда (не аккумулятора)
- (** Скобки: операторы, заключенные в скобки, выполняются в первую очередь, затем продолжается обычный порядок выполнения.

Детальные описания даны в соответствующих приложениях выше.

Операторы CoDeSys:

ST	IL	Мод.	Описание
'			Разграничение строк (т.е. 'string1')
.. []			Задание индексов массива (т.е. ARRAY[0..3] OF INT)
:			Разделитель операнда и типа при объявлении (т.е. var1 : INT;)
;			Конец инструкции (т.е. a:=var1;)
^			Обращение через указатель (т.е. pointer1^)
	LD var1	N	Загрузить значение var1 а аккумулятор
:=	ST var1	N	Записать в var1 значение аккумулятора
	S boolvar		Установка логического операнда boolvar в TRUE, если значение аккумулятора TRUE
	R boolvar		Сброс логического операнда boolvar в FALSE, если значение аккумулятора TRUE
	JMP label	CN	Переход на метку label
<Program name>	CAL prog1	CN	Вызов программы prog1
<Instance name>	CAL inst1	CN	Вызов экземпляра функционального блока inst1
<Fctname>(vx, vy,..)	<Fctname> vx, vy	CN	Вызов функции fctname передача параметров vx, vy
	(Результат вычислений в скобках используется как операнд.
)		Заканчивает вычисление в скобках.
AND	AND	N,(Битовое AND
OR	OR	N,(Битовое OR
XOR	XOR	N,(Битовое исключающее OR
NOT	NOT		Битовое NOT
+	ADD	(Сложение
-	SUB	(Вычитание
*	MUL	(Умножение
/	DIV	(Деление

Приложение Е: Краткий справочник по операторам и компонентам библиотек

>	GT	(Больше, чем
>=	GE	(Больше или равно
=	EQ	(Равно
<>	NE	(Неравно
<=	LE	(Меньше или равно
<	LT	(Меньше, чем
MOD(in)	MOD		Остаток целочисленного деления
INDEXOF(in)	INDEXOF		Внутренний индекс POU in; [INT]
SIZEOF(in)	SIZEOF		Число байт, занимаемых in
SHL(K,in)	SHL		Поразрядный сдвиг влево на K бит
SHR(K,in)	SHR		Поразрядный сдвиг вправо на K бит
ROL(K,in)	ROL		Циклический сдвиг влево на K бит
ROR(K,in)	ROR		Циклический сдвиг вправо на K бит
SEL(G,in0,in1)	SEL		Мультиплексор на 2 входа in0 (при G FALSE) и in1 (при G TRUE)
MAX(in0,in1)	MAX		Возвращает наибольшее из 2х значений in0 и in1
MIN(in0,in1)	MIN		Возвращает наименьшее из 2х значений in0 и in1
LIMIT(MIN,in,Max)	LIMIT		Ограничивает значение in в пределах от MIN до MAX
MUX(K,in0,...in_n)	MUX		Мультиплексор выбирает K-тое значение из группы (от in0 до In_n)
ADR(in)	ADR		Адрес операнда в [DWORD]
ADRINST()	ADRINST		Адрес экземпляра функционального блока, из которого вызывается ADRINST.
BITADR(in)	BITADR		Битовое смещение операнда в [DWORD]
BOOL_TO_<type>(in)	BOOL_TO_<type>		Преобразование типа из логического
<type>_TO_BOOL(in)	<type>_TO_BOOL		Преобразование типа в логический
INT_TO_<type>(in)	INT_TO_<type>		Преобразование значения операнда типа INT в другой базовый тип
REAL_TO_<type>(in)	REAL_TO_<type>		Преобразование значения операнда типа REAL в другой базовый тип
LREAL_TO_<type>(in)	LREAL_TO_<type>		Преобразование значения операнда типа LREAL в другой базовый тип
TIME_TO_<type>(in)	TIME_TO_<type>		Преобразование значения операнда типа TIME в другой базовый тип
TOD_TO_<type>(in)	TOD_TO_<type>		Преобразование значения операнда типа TOD в другой базовый тип
DATE_TO_<type>(in)	DATE_TO_<type>		Преобразование значения операнда типа DATE в другой базовый тип
DT_TO_<type>(in)	DT_TO_<type>		Преобразование значения операнда типа DT в другой базовый тип
STRING_TO_<type>(in)	STRING_TO_<type>		Преобразование текста строки операнда в другой базовый тип. Строка должна содержать соответствующий текст
TRUNC(in)	TRUNC		Преобразование из REAL в INT
ABS(in)	ABS		Абсолютное значение in
SQRT(in)	SQRT		Квадратный корень из in
LN(in)	LN		Натуральный логарифм из in
LOG(in)	LOG		Десятичный логарифм из in
EXP(in)	EXP		E в степени in

SIN(in)	SIN	Синус in
COS(in)	COS	Косинус in
TAN(in)	TAN	Тангенс in
ASIN(in)	ASIN	Арксинус in
ACOS(in)	ACOS	Арккосинус in
ATAN(in)	ATAN	Арктангенс in
EXPT(in,expt)	EXPT expt	Возведение в степень expt

Компоненты Standard.lib:

ST	IL	Описание
LEN(in)	LEN	Длина строки in
LEFT(str,size)	LEFT	Левая значимая часть строки str из size символов
RIGHT(str,size)	RIGHT	Правая значимая часть строки str из size символов
MID(str,size,pos)	MID	Часть строки str с позиции pos из size символов
CONCAT('str1','str2')	CONCAT 'str2'	Конкатенация (склеивание) 2х строк
INSERT('str1','str2',pos)	INSERT 'str2'.p	Вставить строку str1 в str2 с позиции pos
DELETE('str1',len,pos)	DELETE len,pos	Удалить часть строки, len символов, с позиции pos
REPLACE('str1','str2',len,pos)	REPLACE 'str2',len,pos	Заменить часть строки str1 с позиции pos на len символов из str2
FIND('str1','str2')	FIND 'str2'	Поиск подстроки str2 в str1
SR	SR	FB: Переключатель с доминантой включения
RS	RS	FB: Переключатель с доминантой выключения
SEMA	SEMA	FB: Семафор (interruptable)
R_TRIG	R_TRIG	FB: детектор переднего фронта импульса
F_TRIG	F_TRIG	FB: детектор заднего фронта импульса
CTU	CTU	FB: Инкрементный счетчик
CTD	CTD	FB: Декрементный счетчик
CTUD	CTUD	FB: Реверсивный счетчик
TP	TP	FB: Триггер
TON	TON	FB: Таймер включения
TOF	TOF	FB: Таймер выключения
RTC	RTC	FB: Часы

Компоненты Util.lib:

Компонент	Описание
BCD_TO_INT	Преобразование значения операнда BCD в INT
INT_TO_BCD	Преобразование байта INT в BCD
EXTRACT(in,n)	Определяет значение n-го бита DWORD, результат типа BOOL
PACK	Упаковка значений 8 бит в байт
PUTBIT	Присвоить значение определенному биту в DWORD
UNPACK	Распаковка байта в 8 логических переменных
DERIVATIVE	Производная
INTEGRAL	Интеграл
LIN_TRAFO	Преобразование REAL значений
STATISTICS_INT	Макс., Мин., Среднее значения в INT формате

Приложение Е: Краткий справочник по операторам и компонентам библиотек

STATISTICS_REAL	Макс., Мин., Среднее значения в REAL формате
VARIANCE	Дисперсия
PD	ПД регулятор
PID	ПИД регулятор
BLINK	Генератор импульсов
FREQ_MEASURE	Частотомер
GEN	Функциональный генератор
CHARCURVE	Интерполятор
RAMP_INT	Ограничитель скорости изменения сигнала (INT)
RAMP_REAL	Ограничитель скорости изменения сигнала (REAL)
HYSTERESIS	Гистерезис
LIMITALARM	Компаратор

Приложение F: Командная строка / командный файл

10.20 Командная строка

Параметры, указанные в командной строке при запуске CoDeSys, определяют дополнительные режимы работы программы. Все параметры начинаются символом „/“. Регистр знаков не учитывается. Команды выполняются слева направо.

/online	Непосредственно после запуска CoDeSys перейти в режим online текущего проекта.
/run	После подключения автоматически запускает приложение. Имеет смысл только совместно с /online
/show ...	Вид окна при старте CoDeSys
/show hide	Окно не показывается, нет индикатора в панели задач
/show icon	Окно свернуто
/show max	Развернуто на весь экран
/show normal	Размер окна равен размеру, установленному в предыдущем сеансе работы.
/out <outfile>	Все сообщения программы дополнительно записываются в файл <outfile>.
/noinfo	Не показывать заставку при запуске
/userlevel <group>	Определение пользовательской группы (например, "/userlevel 0" для группы 0)
/password <password>	Прямой ввод пароля пользовательской группы (например, "/password abc")
/openfromplc	Будет загружен проект из целевой системы.
/visudownload	Если CoDeSys HMI запускается с проектом, не соответствующим присутствующему в целевой системе, то будет дан диалог подтверждения загрузки.
/notargetchange	Изменение целевой платформы может быть выполнено только через командный файл (См. ниже описание команды "target...").
/cmd <cmdfile>	После запуска выполнить команды из файла <cmdfile>

Формат командной строки:

```
"<Путь CoDeSys-exe>" "<Путь проекта>" /<команда1> /<с команда2> ....
```

Пример командной строки:

```
"D:\dir1\codesys" "C:\projects\ampel.pro" /show hide /cmd command.cmd
```

Открывается проект *ampel.pro*, все окна закрыты. Далее выполняются команды, указанные в *command.cmd*. Путь указывается в кавычках!

10.21 Командный файл (cmdfile)

Приведенные ниже команды можно использовать в командном файле, который в свою очередь вызывается из командной строки (см. выше). Регистр знаков не учитывается.

Командные строки показывается в окне сообщений и могут быть записаны в файл (за исключением команд, начинающихся с „,@“). Часть строки после точки с запятой (;) игнорируется.

Команды управления выполнением:

- onerror continue** Последующие команды должны выполняться, даже если произошла ошибка выполнения.
- onerror break** Последующие команды не должны выполняться, если произошла ошибка выполнения.

Команды меню Online:

- online login** Подключение и загрузка проекта ('Online' 'Login')
- online logout** Отключение ('Online' 'Logout')
- online run** Запуск приложения ('Online' 'Run')
- online stop** Останов приложения ('Online' 'Stop')
- online bootproject** Создание загрузочного проекта ('Online' 'Create boot project')
- online sourcecodedownload** Загрузка кода проекта в ПЛК ('Online' 'Sourcecode download')
- online sim** Включить режим эмуляции ('Online' 'Simulation')
- online sim off** Выключить режим эмуляции ('Online' 'Simulation')

Команды меню File:

- file new** Создать новый проект ('File' 'New')
- file open <projectfile>** Загрузить проект <projectfile> ('File' 'Open')
- /readpwd:<readpassword>** Задаёт пароль доступа на чтение, диалог ввода пароля не показывается.
- /writepwd:<writepassword>** Задаёт пароль полного доступа, диалог ввода пароля не показывается.
- file close** Закрывает проект ('File' 'Close')
- file save** Сохранить проект ('File' 'Save')
- file saveas <projectfile>** Сохранить проект под именем <projectfile> ('File' 'Save as')
опция: <type><version>
 По умолчанию: сохраняется файл проекта текущей версии CoDeSys <projectfile>. Если вы хотите сохранить проект как внешнюю или внутреннюю библиотеку либо в формате ранних версий, добавьте команду:
 "internallib" сохранить как внутреннюю библиотеку;
 "externallib" сохранить как внешнюю библиотеку;
 "pro" сохранить в формате ранних версий:
 допустимые версии <Version>: 15, 20, 21, 22 (версии 1.5, 2.0, 2.1, 2.2)
Пример: "file save as lib_xy internallib22" -> проект "project xy.pro", сохраняется как "lib_xy.lib" для V2.2.
- file printersetup <filename>.dfr** Определяет dfr файл для печати ('File' Printer setup) и одну из возможных опций 'New page per object' или 'New page per subobject' (См. раздел *опция:* Документация)
- pageperobject** или **pagepersubject**
- file archive <filename>.zip** Архивация проекта в zip-файл с указанным именем ('File' Save/Mail Archive')
- file quit** Завершить работу CoDeSys ('File' 'Exit')

Команды меню Project:

project build	Инкрементальная компиляция текущего проекта ('Project' 'Build')
project rebuild or project compile	Полная компиляция текущего проекта ('Project' "Rebuild all")
project clean	Удалить информацию о компиляции и онлайн коррекции ('Project' 'Clean Project')
project check	Контроль текущего проекта ('Project' 'Check all')
project import <file1> ... <fileN>	Файлы <file1> ... <fileN> импортируются в текущий проект ('Project' 'Import'). Внимание: можно использовать шаблоны, например, "project import C:\projects*.exp."
project export <expfile>	Экспорт текущего проекта в файл <expfile> ('Project' 'Export')
project expmul	Каждый объект текущего проекта экспортируется в отдельный файл, получающий имя объекта.
project documentation	Печать проекта (См. раздел Документация)

Команды управления файлом сообщений:

out open <msgfile>	Открыть файл сообщений <msgfile>. Новые сообщения добавляются.
out close	Закрыть текущий файл сообщений.
out clear	Очистить файл сообщений.

Команды управления сообщениями:

echo on	Отображать команды.
echo off	Скрыть команды.
echo <text>	Отобразить текст <text>.

Команды управления заменой (нужны при выполнении import, export, copy):

replace yesall	Заменять все (отвечает «да» на все запросы замены, диалог не отображается)
replace noall	Запрет замены (отвечает «нет» на все запросы замены, диалог не отображается)
replace query	Показывать диалог замены независимо от 'replace yesall' или 'replace noall'

Команды, изменяющие параметры диалогов CoDeSys по умолчанию:

query on	Отображать диалоги и ждать ввода пользователя
query off ok	Для всех диалогов отвечать 'OK'
query off no	Для всех диалогов отвечать 'No'
query off cancel	Для всех диалогов отвечать 'Cancel'

Вызов вложенных командных файлов:

call <parameter1> ... <parameter10>	Вызов файла команд. Допускается до 10 параметров:\$0 - \$9.
--	---

Задание директорий CoDeSys:

(Диалог опций проекта, категория 'Directories', раздел 'General'): Если в нижеописанных командах нужно задать несколько директорий, то их определения должны быть разделены точкой с запятой и пробелом. Определение директории нужно заключить в кавычки. Например:

```
dir lib "D:\codesys\Libraries\Standard; D:\codesys\Libraries\NetVar"
```

dir lib <libdir>	Задаёт <libdir> как директорию библиотек
dir compile <compiledir>	Задаёт <compiledir> как директорию для компиляции
dir config <configdir>	Задаёт <configdir> как директорию файлов конфигурации
dir upload <uploaddir>	Задаёт <uploaddir> как директорию для загружаемых файлов

Задержка выполнения CMDFILE:

delay 5000 Пауза 5 секунд

Управление Watch and Receipt Manager:

watchlist load <file>	Загрузить список (Watchlist) из файла <file> и открыть соответствующее окно ('Extras' 'Load Watchlist')
watchlist save <file>	Сохранить текущий список в файле <file> ('Extras' 'Save Watchlist')
watchlist set <text>	Присвоить загруженному списку имя <text> ('Extras' 'Rename Watchlist')
watchlist read	Обновить значения в списке ('Extras' 'Read receipt')
watchlist write	Установить значения переменных в соответствии с заданными в списке ('Extras' 'Write receipt')

Компоновка библиотек:

library add <library file1> <library file2> .. <library fileN>	Добавить в текущий проект. Если указан относительный путь, то в качестве корневого, используется директория библиотек.
library delete [<library1> <library2> .. <libraryN>]	Удалить указанные или все (если не указано какие) библиотеки из состава текущего проекта.

Копирование объектов:

object copy <source project file> <source path> <target path>	Копирует объекты (source path) в указанные (target path) объекты текущего проекта. (source path) - это имя объекта. Если указана папка, берутся все ее объекты. В этом случае копируется полная структура папки. Если (target path) отсутствует, будет создан новый объект.
--	---

Доступ только для чтения к определенным объектам:

object setreadonly <TRUE FALSE> <object type> <object name>	Установить доступ только по чтению для заданного объекта. Кроме конкретных идентификаторов, можно задать ограничение для всего типа объектов. Допустимые типы: pou, dut (data type), gvl (global variables list), vis (visualization), cnc (CNC object), liblist (Libraries), targetsettings, toolinstanceobject (particular Tools instance), toolmanagerobject (all instances in the Tools tree), customplconfig (PLC configuration), projectinfo (Project information), taskconfig (task configuration), trace, watchentrylist (Watch-
--	---

and Recipe Manager), alarmconfig (Alarm configuration)

Пример: "object setreadonly TRUE pou plc_prg" задает для PLC_PRG доступ только по чтению.

Ввод параметров коммуникации (gateway, device):

gateway local	Использовать шлюз (gateway) данного локального компьютера.
gateway tcpip <Address> <Port>	Использовать шлюз указанного удаленного компьютера. <Address>: TCP/IP адрес или имя в сети удаленного компьютера. <Port>: TCP/IP порт удаленного шлюза. Внимание: Доступны только удаленные шлюзы, не защищенные паролем доступа!
device guid <guid>	Сделать устройство с указанным GUID текущим. Формат GUID (пример): {01234567-0123-0123-0123-0123456789ABC} Скобки и позиции дефисов важны.
device instance <Instance name>	Установить заданное имя для текущего устройства.
device parameter <Id> <Value>	Присвоить указанное значение параметру (ID) устройства.

Системный вызов:

system <command> Выполнить команду операционной системы.

Выбор целевой платформы:

target <Id> Установить целевую платформу для текущего проекта. Если CoDeSys запущен с опцией командной строки (см . выше) "/notargetchange", то это единственный способ выбора целевой платформы.

Команды управления проектом в базе данных ENI:

В данных командах применяются следующие заместители:

<category>: замещает "project" или "shared" или "compile" в зависимости последующей категории базы данных: Project Objects, Shared Objects, Compile Files

<POUname>: имя объекта, соответствующее именам объектов в CoDeSys.

<Objecttype>: замещает расширение имени POU в базе, соответствующее типу объекта (определяется списком типов, см. ENI Администрирование).

Пример: "GLOBAL_1.GVL" -> имя POU "GLOBAL_1", тип "GVL" (global variables list)

<comment>: замещает комментарий (в одиночных кавычках), сохраняемый с соответствующим действием в истории версий.

Команды настройки ENI сервера:

eni on eni off	Опция 'Use source control (ENI)' включается или выключается, соответственно. ('Project' 'Options' 'Project source control')
eni project readonly on eni project readonly off	Опция 'Read only' для 'Project objects' включается или выключается, соответственно. ('Project' 'Options' 'Project objects')
eni shared readonly on	Опция 'Read only' для категории базы данных 'Shard objects' включается

eni shared readonly off	или выключается, соответственно. (<code>'Project' 'Options' 'Shared objects'</code>)
eni set local <POUname>	Объект включается в категорию 'Local', т.е. не будет сохраняться в базе данных проекта (<code>'Project' 'Object' 'Properties' 'Data base-connection'</code>)
eni set shared <POUname>	Объект включается в категорию 'Shared objects' (<code>'Project' 'Object' 'Properties' 'Data base-connection'</code>)
eni set project <POUname>	Объект включается в категорию 'Project objects' (<code>'Project' 'Object' 'Properties' 'Data base-connection'</code>)
eni <category> server <TCP/IP_Address> <Port> <Projectname> <Username> <Password>	Настраивает соединение ENI Server для категории 'Project objects' (<code>'Project' 'Options' 'Project data base'</code>); Пример: eni project server localhost 80 batchtest\project EniBatch Batch (TCP/IP-Address = localhost, Port = 80, Project name = batchtest\project, User name = EniBatch, Password = Batch)
eni compile sym on eni compile sym off	Опция создания символьного файла ('Create ASCII symbol information (.sym)') для объектов категории включается/выключается (<code>'Project' 'Options' 'Project source control' 'ENI settings' for 'Compile files'</code>)
eni compile sdb on eni compile sdb off	Опция 'Create binary symbol information (.sym)' для категории 'Compile files' соответственно, включается/выключается (<code>'Project' 'Options' 'Project source control' 'ENI settings' for 'Compile files'</code>)
eni compile prg on eni compile prg off	Опция 'Create boot project' для категории 'Compile files' соответственно, включается/выключается (<code>'Project' 'Options' 'Project source control' 'ENI settings' for 'Compile files'</code>)

Команды меню 'Project' 'Data Base Link' для работы с базой данных:

eni set <category>	Объект принимается в названную категорию базы данных ('Define')
'eni set <category>set <Objecttype>:<POUname> <Objecttype>:<POUname>	Перечисленные через пробелы объекты принимаются в названную категорию базы данных. ('Multiple Define') Пример: "eni set project pou:as_fub pou:st_prg" -> объекты (pou) as_fub и st_prg get принимаются в категорию 'Project objects'
eni <category> getall	Последние версии объектов вызываются из базы данных. ('Get All Latest Versions')
'eni <category>get <Objecttype>:<POUname> <Objecttype>:<POUname>	Перечисленные через пробелы объекты вызываются из базы данных. ('Multiple Define'). ('Get latest version') Пример: "eni project get pou:as_fub gvl:global_1" -> компонент (POU) as_fub.pou и список глобальных переменных global_1.gvl вызываются из базы данных.
eni <category> checkoutall "<comment>"	Все объекты названной категории выписываются из базы данных. Указанный комментарий будет сохранен вместе с операцией в истории версий.
eni <category> checkout "<comment>" <Objecttype>:<POUname>	Все объекты (Objecttype:POUname), перечисленные через пробелы, выписываются из базы данных. Комментарий будет сохранен вместе с операцией в истории версий по каждому объекту.

<Objecttype>:<POUname> ей в истории версий по каждому объекту.

Пример:

```
"eni project checkout "for working on xy" pou:as_fub gvl:global_1"
```

-> компонент (POU) as_fub и список глобальных переменных global_1 выписываются из базы данных, комментарий "for working on xy" будет сохранен вместе с этой операцией.

**eni <category>checkinall
"<comment>"**

Все объекты проекта, включенные в контроль, прописываются в базе. Комментарий будет сохранен вместе с операцией.

**eni <category> checkin
"<comment>" <Ob-
jecttype>:<POUname>
<Objecttype>:<POUname>**

Все объекты (Objecttype:POUname), перечисленные через пробелы, прописываются в базе. Комментарий будет сохранен вместе с операцией для каждого объекта.

Ключевые слова параметров команд:

В параметрах команд могут применяться следующие ключевые слова, заключенные между "\$":

- \$PROJECT_NAME\$** Имя текущего проекта CoDeSys (имя файла без расширения ".pro")
- \$PROJECT_PATH\$** Путь или директория расположения текущего проекта CoDeSys (без имени диска и завершающей обратной косой черты, т.е. "projects\sub1").
- \$PROJECT_DRIVE\$** Диск текущего проекта (без имени завершающей обратной косой черты, т.е. "D:")
- \$COMPILE_DIR\$** Полный путь (с диском и без завершающей обратной косой черты, т.е. "D:\codesys\compile")
- \$EXE_DIR\$** Директория codesys.exe (с диском и без завершающей обратной косой черты, т.е. D:\codesys)

Пример командного файла:

```
file open C:\projects\CoDeSys_test\ampel.pro
query off ok
watchlist load c:\work\w.wtc
online login
online run
delay 1000
watchlist read
watchlist save $PROJECT_DRIVE$\PROJECT_PATH$\w_update.wtc
online logout
file close
```

В данном примере открывается файл file ampel.pro, загружается ранее созданный список переменных (watch list) под именем w.wtc, затем проект загружается в контроллер и запускается. Через 1 секунду считываются значения переменных из списка и сохраняются в watch.wtc (в директории "C:\projects\CoDeSys_test"). В конце проект закрывается.

Вызвать командный файл из командной строки можно, например, так:

```
"<path of codesys.exe>" /cmd "<path of cmd file>"
```


Приложение G: Сименс импорт.

В подменю **"Project" "Siemens Import"** вы найдете команды, которые позволяют импортировать компоненты (POU) и переменные из файлов Siemens STEP5/7. Команда **"Import from a SEQ symbol file"** позволяет импортировать глобальные переменные из символьных файлов STEP5. Эта команда выполняется перед командой **"Import from a S5 project file"** для создания удобочитаемых идентификаторов при импортировании компонентов. Две эти команды позволяют импортировать компоненты из файлов STEP5 в открытый проект CoDeSys. Вы можете оставить компоненты на языке STEP5 IL или конвертировать их в МЭК языки.

Мы рекомендуем проводить импортирование в пустой проект CoDeSys. Обязательно включите библиотеку *standard.lib*, иначе вы не сможете импортировать таймеры.

Импорт из символьных файлов SEQ

SEQ - это обычный формат символьных файлов STEP5. Символьные определения считываются из файлов *.seq. Каждое определение включает абсолютный адрес S5 элемента программы (вход, выход, память, и т.д.), соответствующий идентификатор, а также может содержать комментарии. Это текстовый файл, каждое определение начинается с новой строки. Поля определения отделены пробелами. Каждый комментарий начинается с точки с запятой.

Определения из SEQ файла будут переведены в глобальные переменные в формате МЭК 61131-3, включая символьное имя, адрес и комментарий (если есть). Адрес будет приведен к требованиям МЭК 61131-3 (знак процента и т.д.). Так как S5 имена могут содержать не разрешенные МЭК символы, они будут изменены, где это необходимо. Недействительные знаки будут заменены символом подчеркивания. Если потребуется больше чем одно подчеркивание, каждый последующий знак будет заменен допустимым символом (например, "0"). Если имя изменено в процессе преобразования, исходное имя будет добавлено в виде комментария. Исходные SEQ комментарии также импортируются. Может быть создано несколько модулей определения глобальных переменных. Каждый блок включает не более 64КБ текста.

Описанный SEQ формат используется в Siemens STEP5-PG в большинстве версий Siemens STEP7-300/400 и ACCON-PG от DELTALOGIC. Формат поддерживается в STEP7-SEQ файлах версии 3.x или старше. Формат файлов STEP7 версии 2.x не поддерживается. Вместо разделительных табуляций он содержит символические имена фиксированной длины.

Для начала вы указываете необходимый SEQ файл в стандартном диалоге Windows и запускаете процесс импортирования. В течение этого процесса могут возникать ошибки. Это может случиться, если разные STEP5/7 идентификаторы преобразуются в одинаковые МЭК идентификаторы. Например, два STEP5 идентификатора «A!» и «A?» будут конвертированы в «A_». В этом случае возникнет следующее сообщение: «Повторное определение идентификатора A_». Измените одну из переменных.

Ни при каких других обстоятельствах не стоит изменять список глобальных переменных. Если идентификаторы содержат адреса, допустимые в Siemens ПЛК, но не существующие в вашем контроллере, не изменяйте их пока, даже если Вы получаете тысячу сообщений об ошибках при компиляции. Сохранить исходные адреса важно для правильного импортирования компонентов.

Если проект, в который вы импортируете, уже содержит декларацию для глобальной переменной x с прямым адресом (например, "%MX4.0"), то может случиться, что импорт из SEQ содержит переменную с тем же самым адресом. Это допускается в МЭК 61131-3, но, возможно, не запланировано пользователем. Никаких сообщений об ошибке не будет, но ваша программа может функционировать неверно. Чтобы избежать этой проблемы, лучше импортировать в пустой проект или в проект, в котором еще нет никаких абсолютных адресов.

После импортирования SEQ можно переходить к импортированию программных компонентов STEP5/7. Вы можете также определить входы-выходы через конфигурацию ПЛК, это не требуется при импортировании, но позволяет своевременно обнаружить ошибки идентификаторов.

Импорт из файла проекта S5

Компоненты (POU) читаются из файла программы Siemens S5 (*.s5d). Исходным является MC5 код, исполняемый S5 SPS. В целом MC5 код соответствует знакомым программистам инструкциям STEP5 IL (без символьных имен). Кроме того, S5D включает комментарии из исходного текста STEP5. Поскольку S5D файл содержит абсолютные адреса без символьческих имен, CoDeSys отыскивает соответствующие символьческие имена, определенные в проекте. Если таковые отсутствуют, абсолютный адрес остается без изменений. Именно поэтому символьный файл SEQ должен быть обработан до S5 файла.

Для начала вы указываете необходимый S5D файл в стандартном диалоге Windows. В следующем диалоговом окне вы должны просмотреть список POU и выбрать необходимые. Наиболее простой путь – выбрать все. Здесь же вы решаете оставлять компоненты в STEP5 IL либо конвертировать их в IL, LD или FBD.

Везде, где это возможно, CoDeSys будет отыскивать и включать в код символьческие имена. Так, для инструкции "U M12.0" он возьмет имя переменной, расположенной по адресу M12.0. Будет использовано первое объявление в проекте. В результате переменная будет импортирована как, например "U-Name" вместо "U M12.0".

Некоторые дополнительные глобальные объявления будут выполнены в процессе импорта. Например, экземпляр функционального блока R_TRIG будет добавлен при необходимости переключаемого по фронту входа.

Конвертирование языка S5 в МЭК С 61131-3

При конвертировании STEP5 в МЭК языки вы должны знать, что некоторые части вашего проекта не могут быть автоматически преобразованы. Для кода, который не может быть конвертирован в МЭК 61131-3, будет дано сообщение об ошибке, и критическая часть оригинального STEP5 IL будет вставлена как комментарий в МЭК POU. В этом случае вы должны поправить текст вручную. Не конвертируются системные команды, специфичные для конкретного CPU. Базовую часть команд STEP5 можно преобразовать в МЭК нажатием клавиши даже в случае, если при этом меняется смысл кода.

Базовый список команд, которые конвертируются в МЭК 61131-3, содержит все команды, преобразуемые в LD или FBD в системе программирования STEP5, и также команды, допустимые в программных блоках STEP5-PB. Кроме того, общие для всех S5 CPU команды функциональных блоков (например, абсолютные и условные переходы, командах сдвига, и т.д.).

Единственное ограничение связано с переустанавливаемыми таймерами STEP5, которые не имеют прямых аналогов в МЭК 61131-3.

Конвертируемые команды:

U, UN, O, ON, S, R, = с операндами: I (входы), O (выходы), M (память), S (S память), D (блоки данных)

U, UN, O, ON с операндами: T (таймер), C (счетчик)

S, R с операндами: C

SU, RU, P, PN с операндами: E, A, M, D

O, O(, U(,)

L, T со следующим диапазоном операндов: E, A, M, D, T, C, P (Periphery) и размером операндов: B (byte), W (word), D (double word), L (left byte), R (right byte)

L со следующими форматами констант: DH, KB, KF, KH, KM, KT, KZ, KY, KG, KC

SI, SE, SA с операндами: T

ZV, ZR с операндами: C

Приложение G: Сименс импорт.

+, -, X, : с операндами: F (фикс. запятая), G (плав. запятая)

+, - с операндами: D (32 бит фикс. запятая)

!=, ><, >, <, >=, <= с операндами: F, D, G

ADD с операндами: BF, KF, DH

SPA, SPB с операндами: PB, FB (с большинством типов параметров), SB

A, AX с операндами: DB, DX

BE, BEA, BEB

BLD, NOP, ***

UW, OW, XOW

KEW, KZW, KZD

SLW, SRW, SLD, RRD, RLD

SPA=, SPB=

SPZ=, SPN=, SPP=, SPM=

ТАК

D, I

Большинство основных операторов

Не конвертируемые команды:

U, UN, O, ON, S, R, = с битовыми операндами: T0.0, C0.0 (таймер и счетчик)

L, T с диапазоном операндов: Q (расширенная периферия)

LC с операндами: T, C

SV, SS, R, FR с операндами: T

FR с операндами: C

Формальные операторы старта сброса и остановки таймеров

Все команды с диапазоном операндов BA, BB, BS, BT (данные ОС).

SPA, SPB с некоторыми операндами: OB

BA, BAB с операндами: FX

E, EX с операндами: DB, DX

STP, STS, STW

DEF, DED, DUF, DUD

SVW, SVD

SPO=, SPS=, SPR

AS, AF, AFS, AFF, BAS, BAF

ENT

SES, SEF

B с операндами: DW, MW, BS

LIR, TIR, LDI, TDI, TNW, TXB, TXW

MAS, MAB, MSA, MSB, MBA, MBS

MBR, ABR

LRW, LRD, TRW, TRD

TSG

LB, TB, LW, TW с операндами: GB, GW, GD, CB, CW, CD

ACR, TSC

BI

SIM, LIM

Если вы внимательно посмотрите на неконвертируемые команды, то обнаружите, что это дополнительные команды, которые специфичны для некоторых CPU. Из основных команд не конвертируются: BCD таймер или счетчик значений (LC T, LC C), таймер типов SV и SS и переустанавливаемый таймер.

Блоки данных:

Блоки данных STEP5 преобразуются в POU, имеющие объявления, но не содержащие кода.

Некоторые типовые проблемы импорта STEP5, требующие ручной правки:

1. Значения времени в переменных word

В STEP5 значения времени могут размещаться в любом слове памяти (word) или блока данных. В МЭК 61131-3, переменные или константы TIME не совместимы с WORD адресацией. Отсутствие явного указания типа времени может приводить к ошибочным последовательностям команд при импорте. При обнаружении некорректного использования таких переменных вы увидите сообщение "Incompatible Types: Cannot convert WORD to TIME." либо "Incompatible Types: Cannot convert TIME to WORD." В этом случае нужно изменить объявление переменной WORD в TIME.

2. Сложности с блоками данных

В МЭК 61131-3 отсутствуют блоки данных (DB) и нет прямого аналога этой концепции. В STEP5 DB применяются для размещения переменных (по словам, как в области памяти), допускают массивы (B DW), указатели (B MW100 A DB 0) или объединения (доступ как к байту, слову или двойному слову). Конвертирование STEP5 DB проходит успешно, только если их применение упорядочено. При доступе к переменным блока данных, необходимо знать, какой блок сейчас открыт (команда A DB x, где x = номер блока). Это можно определить, если в начале POU стоит команда A DB и номер блока передается в качестве параметра. Если же команда A DB отсутствует, то нет возможности определить, какой блок использовать и соответственно конвертировать такой POU нельзя. Об этой проблеме вас оповестит сообщение "No open data block (insert an A DB)". При конвертировании будут получаться обращения к несуществующим переменным, например "ErrorDW0". Вы должны будете вручную поставить обращение к необходимому DB (заменить "ErrorDW0" на "DB10.DW0"). Конечно, проще заранее явно расставить в самих STEP5 POU необходимые команды A DB.

В случае пропуска команды A BD существует опасность, что при преобразовании сформируется обращение к ошибочному DB.

3. Косвенное обращение к блокам данных

В STEP5 вы можете создавать подобие нескольких экземпляров блоков данных и косвенно (через индекс) выбирать нужный:

L KF +5

T MW 44

B MW 44

A DB 0

Приложение G: Сименс импорт.

В конце этой последовательности будет открыт блок данных DB5 (в целом DB, номер которого записан по адресу %MW44). Такой прием не распознается при конвертировании. Поэтому ручная правка будет необходима в итоговом проекте:

Прежде всего все экземпляры DB должны быть импортированы (как DB5, DB6 и т.д.) в стандартные IL, LD или FBD (по вашему желанию) POU. Каждый такой POU не имеет кода и содержит только объявления. Создайте теперь новый тип данных (например DBType) и перенесите в него объявления данных из соответствующего конвертированного POU. Затем создайте соответствующие глобальные объявления:

```
VAR_GLOBAL
  DB5, DB6 : DBType;
END_VAR
```

Теперь вы можете удалить ненужные более POU (бывшие DBs).

Доступ к нужному DB блоку теперь можно получить путем передачи программному компоненту дополнительного параметра VAR_INPUT типа DBType. Вы можете использовать один программный компонент с разными блоками данных путем задания соответствующего актуального параметра при его вызове.

4. В S5 существуют **встроенные функциональные блоки**. Пользователю доступен только их интерфейс, программный код реализован не в STEP5 (или MC5) или защищен специальным механизмом. Результатом импорта таких блоков являются пустые POU, имеющие только объявление интерфейса. Программный код вам придется написать самостоятельно

5. Помимо этого, могут встретиться блоки (OB) реализованные в виде **ассемблерных вставок** (805xx например). Преимущественно таким способом реализуется ПИД регулятор (OB251), использующий для приема параметров и хранения локальных переменных отдельный блок данных. Естественно, код регулятора не попадет в импортированный проект. Для понимания работы таких блоков используйте документацию по данному процессору.

6. Конфигурационные блоки (такие, как DB1 [S5-95U], DX0, и DX2), применяемые иногда в S5 CPU, конвертируются в бессмысленные МЭК компоненты. Чтобы понять смысл их содержимого, используйте руководство по программированию данного CPU. В крайнем случае можно исследовать определенную данным блоком конфигурацию в системе программирования S5. Она включает настройку коммуникаций, обработку аналоговых значений, мультипроцессорную работу и т.д. Естественно, все эти параметры имеют смысл только в Siemens SPS.

По окончании импортирования внимательно просмотрите полученный код. Не конвертированные автоматически фрагменты снабжены комментарием:

```
(*Warning! Unconvertible STEP5/7 code shown as comment:*)
```

Здесь же в комментарии содержится и исходный код.

Обязательно проверьте адреса. При импорте создаются исходные Siemens адреса, в формате:

Биты: Байтовое-Смещение.Номер-Бита

Не биты: Байтовое-Смещение

Два последовательных адреса слов могут пересекаться. Так %MW32 и %MW33 имеют общий байт %MB33 (только в Siemens SPS). В CoDeSys %MW32 и %MW33 обычно не имеют пресечения.

Ваш ПЛК может иметь иерархическую модель адресации памяти, например "%MW10.0.0. Вы можете сразу переделать все адреса либо попробовать оставить их как есть. Делать это нужно очень осторожно.

Приложение G: Сименс импорт.

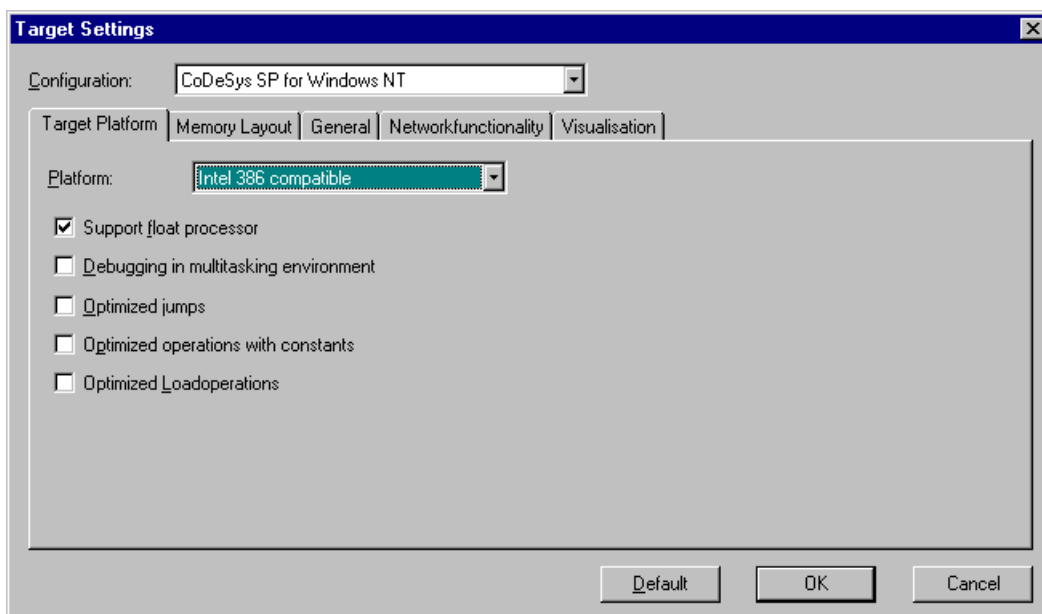
но! В Siemens программах попеременное обращение к одной и той же области памяти - как слову, байту или биту - является обычной практикой. При импортировании блоков данных CoDeSys создает WORD определения для соответствующих слов. Обращение к словам выполняется напрямую. В МЭК нет возможности объявить в общей памяти несколько пересекающихся переменных разного типа. Но это возможно в прямоадресуемой (M) памяти и памяти входов-выходов. Если применяли адреса типа %MX33.3, %MB33, %MW32 или %MD30, их корректное преобразование нужно выполнить вручную.

Используйте список перекрестных ссылок, включающий входы, выходы и распределение памяти. Проверьте пересекающиеся области и исключите их там, где их не должно быть.

Приложение Н: Опции целевых систем

10.22 Системные опции целевых платформ (Target Platform)

Intel 386 совместимые



Пункт диалога

Пояснение

Platform

Тип целевой платформы

Support float processor

если активна: разрешает применять FPU команды в операциях с плавающей запятой.

Debugging in multitasking environment

если активна: генерируется дополнительный код, позволяющий проводить отладку в многозадачной среде.

Optimized jumps

если активна: разрешена оптимизация переходов в операциях сравнения; быстрый + компактный код (особенно на 386/486); Строки, содержащие сравнения перед переходом, будут отображаться серым в режиме контроля выполнения.

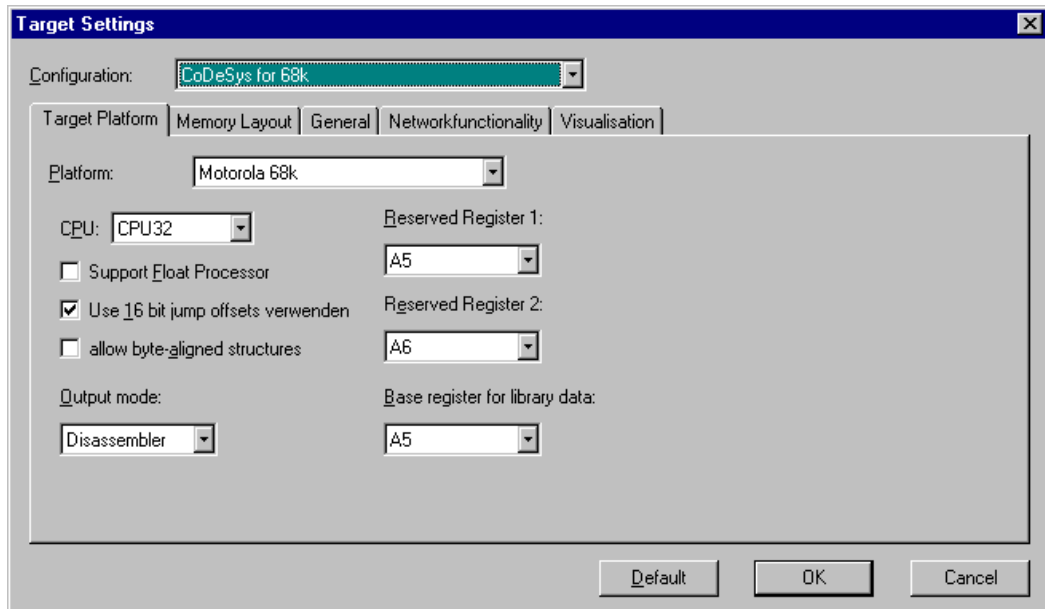
Optimized operations with constants

Оптимизация операций с константами ($A = A + 1$, $A < 500$ etc.); быстрый + компактный код (особенно на 386/486); Константы будут отображаться серым в режиме контроля выполнения.

Optimized Loadoperations

Исключаются повторные операции загрузки при множественном доступе к переменным/константам; быстрый + компактный код

Motorola 68K



Пункт диалога

Пояснение

Platform

Тип целевой платформы.

CPU

Вариант 68k CPU: базовый 68000 либо CPU32 и старше.

Support Float Processor

если активна: разрешает применять FPU команды в операциях с плавающей запятой.

Use 16 bit jump offsets

если активна: переходы при вычислении логических выражений используют относительные 16 бит смещения (более сложные выражения, но больше размер кода).

не активна: 8 бит смещения .

allow byte-aligned structures

если активна: выравнивание только по четным адресам.

не активна: произвольное расположение.

Reserved Register 1

A2,A4,A5,A6: Указанные адресные регистры зарезервированы и не используются

Иначе: регистр используется генератором кода

Reserved Register 2

Дополнительный зарезервированный регистр адреса

Base register for library data

Регистр для адресации статических данных в C библиотеках (перед вызовом библиотечной функции, загружается адресом свободной памяти).

Если "None", используется A5.

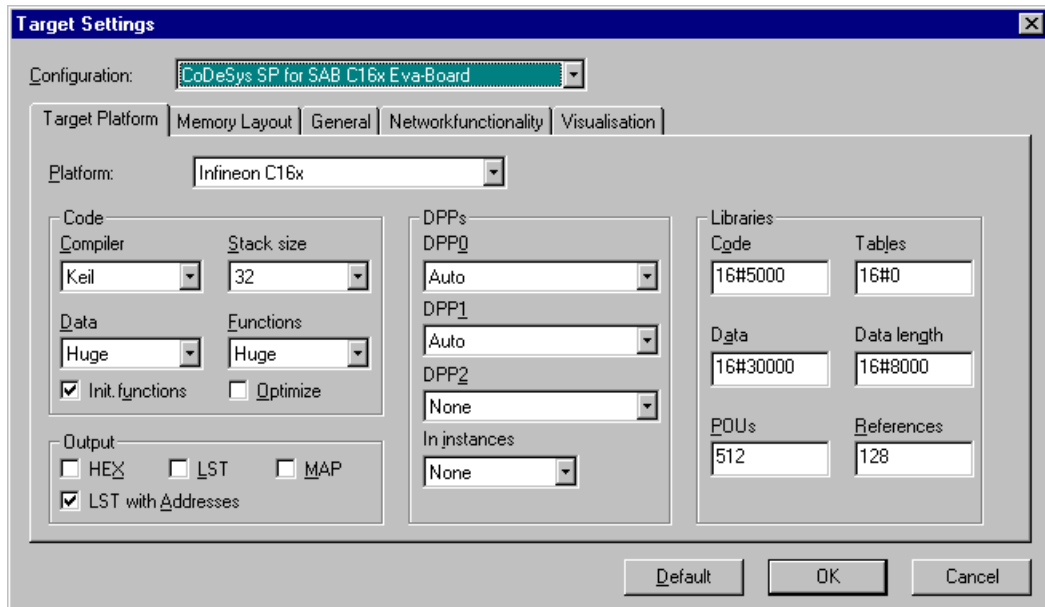
Output-Mode

Nothing = ничего

Assembler = генерируется файл "code68k.hex" в директории компиляции (См. "Project/Options/Directories").

Disassembler = Дополнительно создает файл дизассемблера

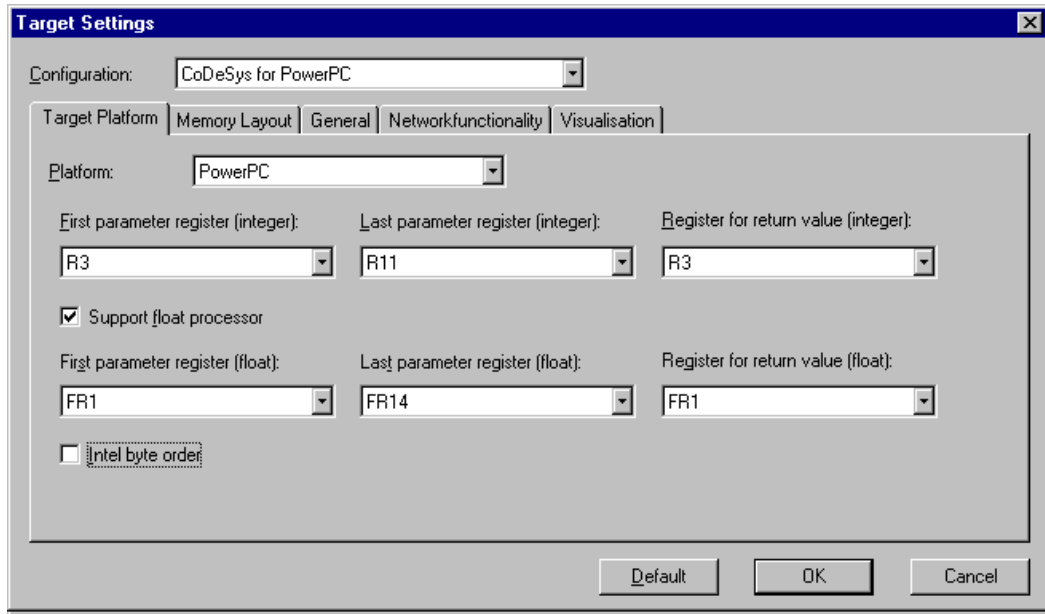
Infineon C16x



<i>Пункт диалога</i>	<i>Пояснение</i>
Platform	Тип целевой платформы
Code / Compiler:	Компилятор, использованный для построения RTS и библиотек (определяет порядок вызова C функций)
Code / Stack size	Максимальная глубина вызовов (вложений)
Code / Data	Модель памяти данных
Code / Functions	Модель памяти кода
Init. functions	если активна: функции включают код инициализации локальных переменных
Optimize	если активна: оптимизация кода, если индекс элемента массива константа
Output HEX-File	если активна: формируется выходной HEX код
Output BIN-File	если активна: формируется двоичный код
Output MAP	если активна: формируется map-файл кода
Output LST	если активна: формируется листинг кода
Output LST , of addresses	если активна: листинг включает адреса
DPPs / DPP0..DPP2	Выбор DPP для DPP0, DPP1, DPP2
In Instances	DPP для короткой адресации функциональных блоков
Libraries / Code	Опции для библиотек
Tables	
Data	
Data length	
POUs	
References	

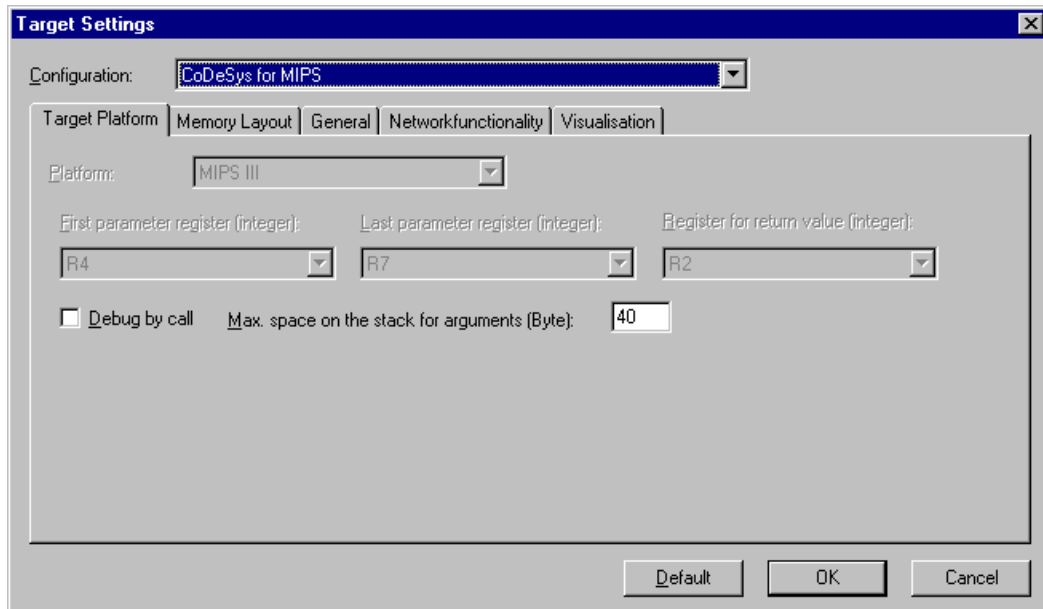
ARM и Power PC

В обеих системах пункты диалога одинаковы.



<i>Пункт диалога</i>	<i>Пояснение</i>
Platform	Тип целевой платформы
Support float processor	если активна: разрешает применять FPU команды в операциях с плавающей запятой
First parameter Register (integer)	Регистр, в котором передается первый целочисленный параметр при вызове C-функции
Last parameter Register (Integer)	Регистр, в котором передается последний целочисленный параметр при вызове C-функции
Register for return values (Integer)	Регистр, в котором возвращается целочисленный результат C-функции
First parameter Register (Float):	Регистр, в котором передается первый параметр в формате с плавающей запятой при вызове C-функции
Last parameter Register (Float):	Регистр, в котором передается последний параметр в формате с плавающей запятой при вызове C-функции
Register for return value (Float):	Регистр, в котором возвращается параметр в формате с плавающей запятой C-функции
Intel byte order	если активна: применяется Intel порядок байт

MIPS



Пункт диалога

Пояснение

Platform

Тип целевой платформы

First parameter Register (integer)

Регистр, в котором передается первый целочисленный параметр при вызове C-функции

Last parameter Register (Integer)

Регистр, в котором передается последний целочисленный параметр при вызове C-функции

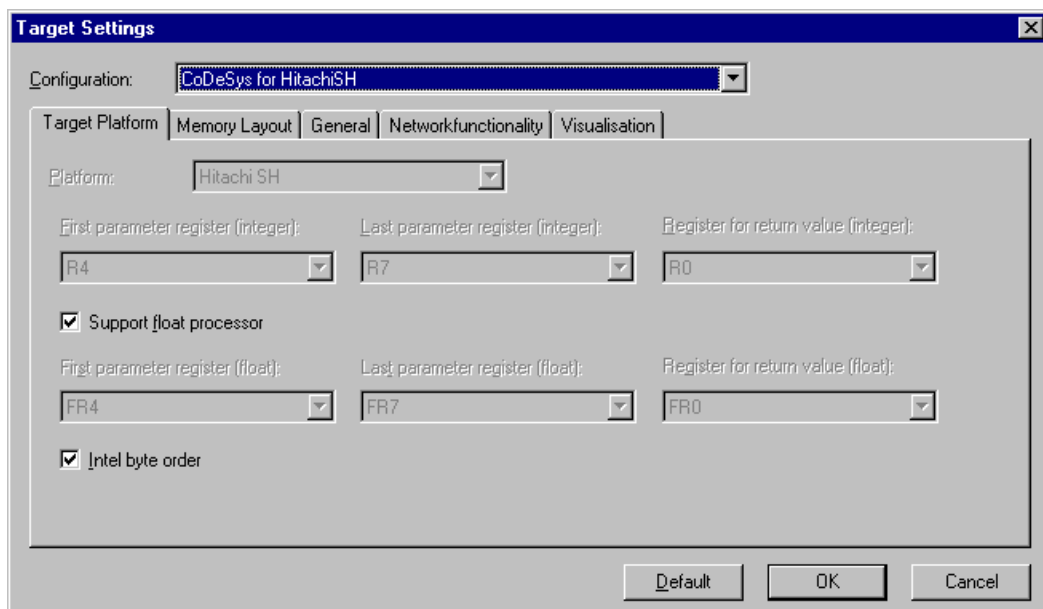
Register for return values (Integer)

Регистр, в котором возвращается целочисленный результат C-функции

Max. space on the stack for arguments (Byte):

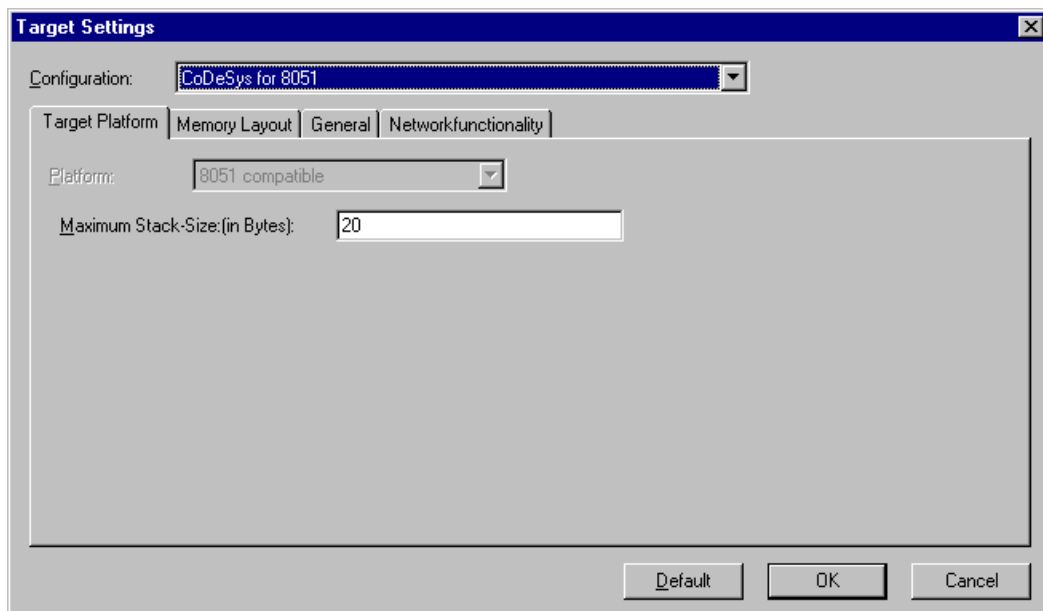
Зависит от ОС: Макс. размер (в байтах) аргументов, помещаемых в стек

'Hitachi SH'



<i>Пункт диалога</i>	<i>Пояснение</i>
Platform	Тип целевой платформы
Support float processor	если активна: разрешает применять FPU команды в операциях с плавающей запятой
First parameter Register (integer)	Регистр, в котором передается первый целочисленный параметр при вызове С-функции
Last parameter Register (Integer)	Регистр, в котором передается последний целочисленный параметр при вызове С-функции
Register for return values (Integer)	Регистр, в котором возвращается целочисленный результат С-функции
Max. space on the stack for arguments (Byte)	Зависит от ОС: максимальный размер аргументов (в байтах), которые можно разместить в стеке.
First parameter Register (Float):	Регистр, в котором передается первый параметр в формате с плавающей запятой при вызове С-функции
Last parameter Register (Float):	Регистр, в котором передается последний параметр в формате с плавающей запятой при вызове С-функции
Register for return value (Float):	Регистр, в котором возвращается параметр в формате с плавающей запятой С-функции
Intel byte order	если активна: применяется Intel порядок байт

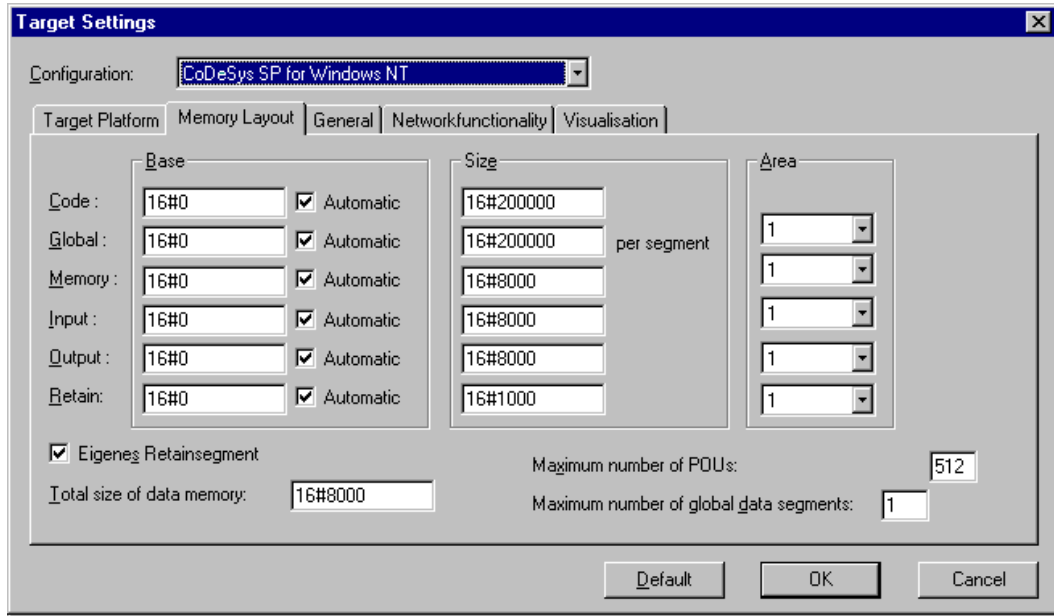
8051 совместимые



<i>Пункт диалога</i>	<i>Пояснение</i>
Platform	Тип целевой платформы
Maximum Stack-Size:(in Bytes)	Максимальный размер стека (в байтах)

10.23 Опции распределения памяти (Memory Layout).

Приведенные здесь описания опции справедливы для всех платформ.

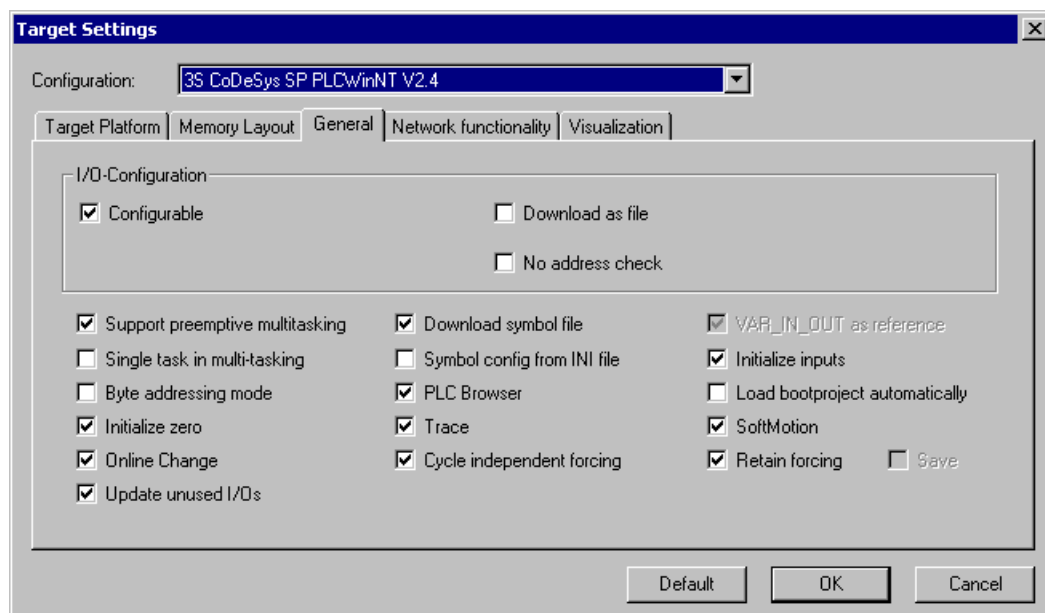


<i>Пункт диалога</i>	<i>Пояснение</i>
Base (Code)	Automatic: автоматическое размещение сегмента кода Иначе: с указанного абсолютного адреса
Base (Global)	Automatic: автоматическое размещение сегмента данных (global data) Иначе: с указанного абсолютного адреса
Base (Memory)	Automatic: автоматическое размещение сегмента прямоадресуемой памяти (M) Иначе: с указанного абсолютного адреса
Base (Input)	Automatic: автоматическое размещение образа входов (I) Иначе: с указанного абсолютного адреса
Base (Output)	Automatic: автоматическое размещение образа выходов (O) Иначе: указанного абсолютного адреса
Base (Retain)	Automatic: автоматическое размещение сегмента Retain памяти Иначе: указанного абсолютного адреса
Area (Code)	Сегментный номер для кода
Area (Global)	Сегментный номер для данных (global data)
Area (Memory)	Сегментный номер для данных прямоадресуемой памяти
Area (Input)	Сегментный номер для входов
Area (Output)	Сегментный номер для выходов
Area (Retain)	Сегментный номер для Retain памяти
Size (Code)	Размер сегмента кода

Size pro Segment (Global)	Размер сегмента данных
Size (Memory)	Размер сегмента прямоадресуемой памяти
Size (Input)	Размер сегмента входов
Size (Output)	Размер сегмента выходов
Size (Retain)	Размер сегмента Retain памяти
Total size of data memory	Общие количество памяти данных
Own retain segment	если активна: разместить Retain в отдельном сегменте
Maximum number of global data segments	Макс. число сегментов данных (global data)
Maximum number of POU's	Макс. число POU в проекте

10.24 Опции общей категории (General)

Приведенные здесь описания опции справедливы для всех платформ.



Пункт диалога

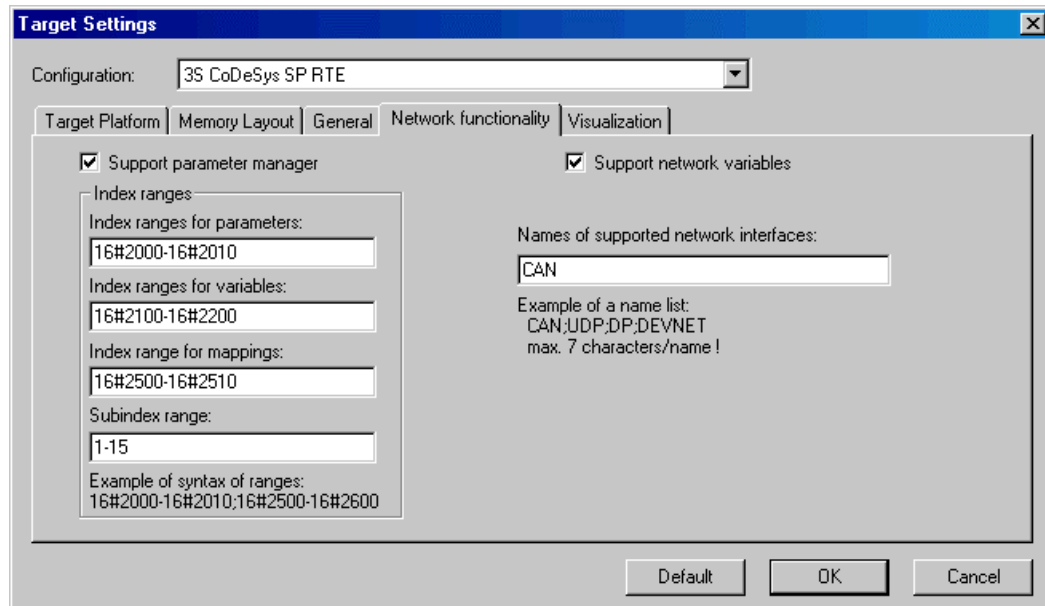
Пояснение

Configurable	если активна: поддерживается конфигурирование I/O и загрузка итогового описания в контроллер
Support CANopen configuration	если активна: поддерживается конфигурирование CANopen и загрузка итогового описания в контроллер
Support Profibus configuration	если активна: поддерживается конфигурирование Profibus и загрузка итогового описания в контроллер
Support preemptive multi-tasking	если активна: поддерживается конфигурирование многозадачности
Download as file	если активна: описание I/O загружается из файла
No address checking	если активна: не проверяется корректность МЭК адресов
Online Change	если активна: разрешена <i>онлайн</i> коррекция кода

Update unused I/O's	если активна: CoDeSys создает задачу, обновляющую значения входов и выходов, не используемых в настоящее время. Таким образом, будут обновляться значения входов/выходов в PLC Configuration.
Singletask in multitasking	не используется в текущей версии
Byte-addressing mode	если активна: байтовая адресация для всех МЭЖ адресов независимо от типа (т.е. var1 AT %QD4 располагается по адресу %QB4)
Initialize zero	если активна: начальное обнуление данных
Download Symbol File	если активна: если символьный файл создан, то он будет загружен
Symbol config from INI file	если активна: символьная конфигурация читается не из проекта (задается в диалоговом окне) а из файла codesys.ini либо из другого файла, указанного в codesys.ini
PLC-Browser	если активна: PLC Browser разрешен
Trace	если активна: трассировка разрешена
VAR_IN_OUT as reference	если активна: переменные типа VAR_IN_OUT передаются через указатели. Соответственно им нельзя присваивать константы и запись/чтение не доступны вне функционального блока.
Initialize Inputs	не активна: с целью оптимизации исключается инициализация входов (так AT %IX может иметь неопределенное значение в первом цикле!)
Automatic boot project load	если активна: после каждого открытия нового проекта автоматически создается загрузочный код и пересылается в ПЛК.
Softmotion	если активна: SoftMotion разрешен и присутствует на вкладке ресурсов (CNC лист и CAM)
Retain forcing	если активна: список фиксированных переменных будет сохранен в системе исполнения, даже после отключения системы программирования. При отключении будет предложен диалог, в котором нужно будет подтвердить сохранение фиксации (в настоящее время поддерживается в CoDeSys SP 32F V2.4).
Save	если активна: системе исполнения продолжает фиксацию даже после перезапуска. Доступна, только если включена опция 'Retain forcing'.
Cycle independent forcing	если активна: фиксация будет выполняться не только перед началом и после рабочего цикла, но и при любой записи в процессе работы программы.

10.25 Опции категории Сетевая функциональность

Приведенные здесь описания опции справедливы для всех платформ.



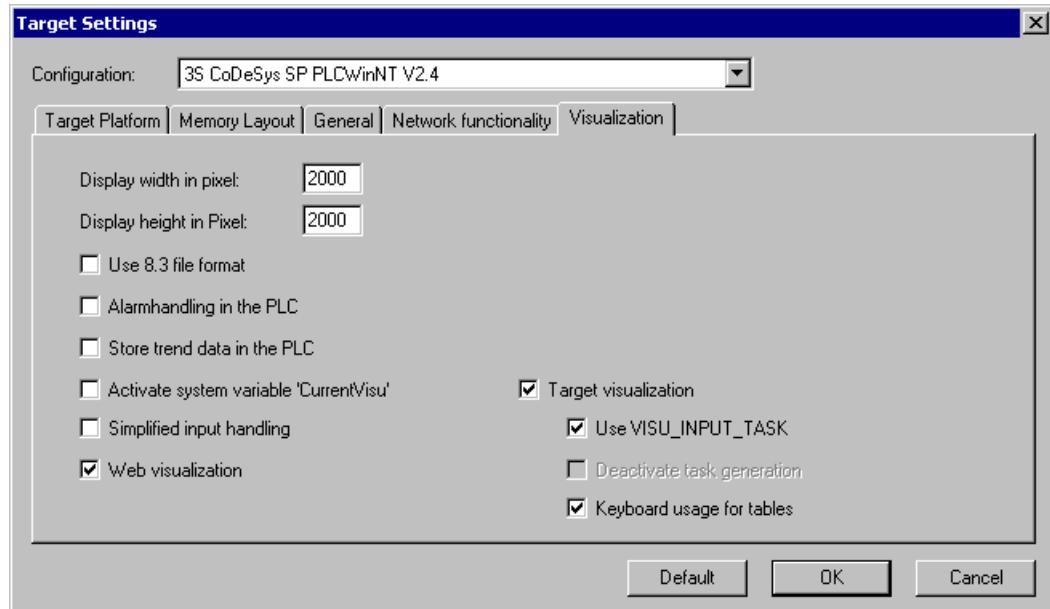
Пункт диалога

Пояснение

Support parameter manager	Если активна: 'Parameter-Manager' присутствует на вкладке ресурсов. Используйте его для создания словаря Object Dictionary для переменных и параметров, доступных для других контроллеров
Support network variables	Активирует применение сетевых переменных, автоматически обновляемых в сети
Names of supported network-interfaces	Список поддерживаемых сетей, т.е.: CAN; UDP; DP
Index ranges for parameters	Диапазон индексов типа 'Parameters'
Index-ranges for variables	Диапазон индексов типа 'Variables'
Index-ranges for Mappings	Диапазон индексов типа 'Mappings'
Subindex range	Диапазон суб-индексов в описанных выше диапазонах для параметров и переменных Object Dictionary

10.26 Опции категории Визуализация

Приведенные здесь описания опции справедливы для всех платформ.



Пункт диалога

Пояснение

Display width in pixel
Display height in pixel

Область указанного размера будет использоваться при редактировании визуализации. Соответственно, размер экрана, на котором будет прокручиваться готовая визуализация.

Use 8.3 file format

Длинные имена файлов растровых рисунков и языковые файлы будут «обрезаны» до формата 8.3 и в таком виде загружены в ПЛК.

Alarmhandling in the PLC

Задача ALARM_TASK автоматически будет добавлена в конфигурацию задач. Она неявно выполняет созданный ST-код, оценивающий статус определенных тревог и вызывает связанные с ними действия (если они заданы). ST-код использует вспомогательную библиотеку SysLibAlarm-Trend.lib. Она загружается автоматически. (Дополнительно включаются библиотеки SysLibSockets.lib, SysLibMem.lib, SysLibTime.lib, SysLibFile.lib. Они должны поддерживаться в вашей целевой системе!)

Внимание: "Alarm handling in the PLC" можно применять даже без Target-или Web-визуализаций.

Store trend data in the PLC	<p>Активируется поддержка трендов в ПЛК. Автоматически создается задача TRENД_TASK, неявно выполняющая ST-код для записи трендов в кольцевой буфер и как опция в «исторические» файлы.</p> <p>ST-код использует функции вспомогательной библиотеки SysLibAlarm-Trend.lib. (Дополнительно включаются библиотеки SysLibSockets.lib, SysLibMem.lib, SysLibTime.lib, SysLibFile.lib. Они должны поддерживаться в вашей целевой системе!)</p> <p>Внимание: 'Store Trend data....' можно применять даже без Target- или Web-визуализаций.</p>
Activate system variable 'CurrentVisu	<p>Системная переменная CurrentVisu используется в качестве переключателя визуализаций.</p>
Supported fonts in the target	<p>Список шрифтов, доступных в целевой системе.</p>
Simplified input handling	<p>если активна: реализуется упрощенный режим ввода в online: не нужно нажимать клавиши <Tab> и <Пробел> для перехода к следующему полю ввода. Переход происходит автоматически по подтверждению ввода клавишей <Return>. Поле ввода можно дополнительно выбирать стрелками или клавишей <Tab> и при этом сразу производить ввод.</p> <p>если не активна: необходимо использовать клавиши <Tab> и <Пробел> для выделения поля и активации режима ввода.</p>
Web visualisation	<p>если активна: Все объекты визуализации компилируются для работы Web визуализации.</p>
Target visualisation	<p>если активна: Все объекты визуализации компилируются для работы Target визуализации.</p>
Use VISU_INPUT_TASK	<p>(доступна только при активной опции Target-Visualisation)</p> <p>если активна: автоматически создаются две задачи для выполнения платформенной (Target) визуализации (VISU_INPUT_TASK, VISU_TASK), иначе создается только VISU_TASK, включающая функции VISU_INPUT_TASK.</p>
Deactivate task generation	<p>(доступна только при активной опции Target-Visualisation)</p> <p>если активна: задачи VISU_INPUT_TASK и VISU_TASK (см. выше) не генерируются автоматически. POU MAINTARGETVISU_PAINT_CODE может вызываться контролируемо из пользовательской программы. Подробнее см. Рук-во по CoDeSys Visualization.</p>
Keyboard usage for tables	<p>(используется только в Target visualization, см. выше)</p> <p>Только если активна данная опция, функции клавиш VK_TAB (табуляция) и VK_SPACE (пробел) определенные для таблиц, можно будет использовать в целевой визуализации. Отключение данной опции приведет к тому, что код для этих функций не будет генерироваться.</p>

Приложение I: Использование клавиатуры

В CoDeSys определены клавиши для быстрой работы с клавиатуры с минимальным использованием команд меню.

- Функциональная клавиша <F6> осуществляет быстрое переключение между разделами объявлений и кода в окнах редакторов.
- <Alt>+<F6> осуществляет быстрый переход по открытым окнам Object Organizer и в окно сообщений (Message window). В режиме поиска <Alt>+<F6> выполняет переключение между Object Organizer и окном поиска.
- Используйте <Control>+<F6> для перехода в следующее открытое окно редактора и <Control>+<Shift>+<F6> для перехода в предыдущее окно.
- Клавиша <Tab> осуществляет переход по полям ввода и кнопкам диалоговых окон.
- «Стрелками» вы можете перемещаться по вкладкам Object Organizer и Library Manager.

Все прочие действия выполняются через меню или соответствующие быстрые комбинации клавиш. Контекстное меню, содержащее наиболее часто используемые в данном контексте команды, вызывается <Shift>+<F10>

Быстрые комбинации клавиш.

Общие функции	
Переключение между разделами объявлений и кода в окнах редакторов	<F6>
Переход по открытым окнам Object Organizer и окном сообщений	<Alt>+<F6>
Контекстное меню	<Shift>+<F10>
Объявление переменной	<Ctrl>+<Enter>
Переход из окна сообщения к исходной позиции в окне редактора	<Enter>
Переход в следующее открытое окно	<Control>+<F6>
Переход в предыдущее открытое окно	<Control>+<Shift>+<F6>
Открытие и закрытие многоэлементных переменных	<Enter>
Открытие и закрытие папок	<Enter>
Перемещение по вкладкам Object Organizer и Library Manager	<Стрелки>
Переход по полям ввода и кнопкам диалоговых окон	<Tab>
Вызов контекстно-зависимой справочной системы	<F1>
Общие команды меню	
'File' 'Save'	<Ctrl>+<S>
'File' 'Print'	<Ctrl>+<P>
'File' 'Exit'	<Alt>+<F4>
'Project' 'Check'	<Control>+<F11>
'Project' 'Build'	<Shift>+<F11>
'Project' 'Rebuild all'	<F11>
'Project' 'Delete Object'	
'Project' 'Add Object'	<Ins>
'Project' 'Rename Object'	<Spacebar>

Приложение I: Использование клавиатуры

'Project' 'Open Object'	<Enter>
'Edit' 'Undo'	<Ctrl>+<Z>
'Edit' 'Redo'	<Ctrl>+<Y>
'Edit' 'Cut'	<Ctrl>+<X> или <Shift>+
'Edit' 'Copy'	<Ctrl>+<C>
'Edit' 'Paste'	<Ctrl>+<V>
'Edit' 'Delete'	
'Edit' 'Find next'	<F3>
'Edit' 'Input Assistant'	<F2>
'Edit' 'Auto Declare'	<Shift>+<F2>
'Edit' 'Next Error'	<F4>
'Edit' 'Previous Error'	<Shift>+<F4>
'Online' 'Log-in'	<Alt><F8>
'Online' 'Logout'	<Ctrl>+<F8>
'Online' 'Run'	<F5>
'Online' 'Toggle Breakpoint'	<F9>
'Online' 'Step over'	<F10>
'Online' 'Step in'	<F8>
'Online' 'Single Cycle'	<Ctrl>+<F5>
'Online' 'Write Values'	<Ctrl>+<F7>
'Online' 'Force Values'	<F7>
'Online' 'Release Force'	<Shift>+<F7>
'Online' 'Write/Force dialog'	<Shift>+<F7>
'Window' 'Messages'	<Shift>+<Esc>
Команды FBD редактора	
'Insert' 'Network (after)'	<Shift>+<T>
'Insert' 'Assignment'	<Ctrl>+<A>
'Insert' 'Jump'	<Ctrl>+<L>
'Insert' 'Return'	<Ctrl>+<R>
'Insert' 'Operator'	<Ctrl>+<O>
'Insert' 'Function'	<Ctrl>+<F>
'Insert' 'Function Block'	<Ctrl>+
'Insert' 'Input'	<Ctrl>+<U>
'Extras' 'Negate'	<Ctrl>+<N>
'Extras' 'Zoom'	<Alt>+<Enter>
Команды CFC редактора	
'Insert' 'POU'	<Ctrl>+
'Insert' 'Input'	<Ctrl>+<E>
'Insert' 'Output'	<Ctrl>+<A>
'Insert' 'Jump'	<Ctrl>+<G>
'Insert' 'Label'	<Ctrl>+<L>
'Insert' 'Return'	<Ctrl>+<R>
'Insert' 'Comment'	<Ctrl>+<K>
'Insert' 'POU input'	<Ctrl>+<U>
'Extras' 'Negate'	<Ctrl>+<N>
'Extras' 'Set/Reset'	<Ctrl>+<T>
'Extras' 'Connection'	<Ctrl>+<M>

Приложение I: Использование клавиатуры

'Extras' 'EN/ENO'	<Ctrl>+<E>
'Extras' 'Zoom'	<Alt>+<Enter>
Команды LD редактора	
'Insert' 'Network (after)'	<Shift>+<T>
'Insert' 'Contact'	<Ctrl>+<K>
'Insert' 'Parallel Contact'	<Ctrl>+<R>
'Insert' 'Function Block'	<Ctrl>+
'Insert' 'Coil'	<Ctrl>+<L>
'Extras' 'Paste below'	<Ctrl>+<U>
'Extras' 'Negate'	<Ctrl>+<N>
'Extras' 'Zoom'	<Alt>+<Enter>
Команды SFC редактора	
'Insert' 'Step-Transition (before)'	<Ctrl>+<T>
'Insert' 'Step-Transition (after)'	<Ctrl>+<E>
'Insert' 'Alternative Branch (right)'	<Ctrl>+<A>
'Insert' 'Parallel Branch (right)'	<Ctrl>+<L>
'Insert' 'Jump'	<Ctrl>+<U>
'Extras' 'Zoom Action/Transition'	<Alt>+<Enter>
Работа в конфигуляторах ПЛК и задач	
Открыть и закрыть элемент	<Enter>
Редактирование	<Spacebar>
'Extras' 'Edit Entry'	<Enter>
Работа редакторе менеджера параметров.	
Переключение между окном навигации и списком	<F6>
Удалить строку в списке	<Ctrl>+ <Shift>+
Удалить поле	

Приложение J: Ошибки и предупреждения компилятора

Если при компиляции проекта обнаружены ошибки или предупреждения, то соответствующие им сообщения будут отображены в окне сообщений. Клавиша <F4> позволяет переходить к следующему сообщению. При этом открывается окно редактора соответствующего POU. Каждая ошибка и сообщение имеет уникальный номер. Клавиша <F1> в окне сообщений открывает соответствующее ошибке окно справочной системы.

10.27 Предупреждения

1100

"Unknown function '<name>' in library."

Используется внешняя библиотека. Проверьте, все ли функции, присутствующие в .hex file, определены в .lib file.

1101

"Unresolved symbol '<Symbol>'."

Генератор кода предполагает POU с именем <Symbol>, неопределенный в проекте. Определите функцию/программу с таким именем.

1102

"Invalid interface for symbol '<Symbol>'."

Генератор кода предполагает функцию с именем <Symbol> и имеющую один скалярный вход либо программу с именем <Symbol> и не имеющую входов и выходов.

1103

"The constant '<name>' at code address '<address>' overwrites a 16K page boundary!"

Строчковая константа превышает 16К границу страницы. Система не имеет таких ограничений. Свяжитесь с изготовителем ПЛК.

1200

"Task '<name>', call of '<name>' Access variables in the parameter list are not updated"

Переменные, которые используются при вызове функциональных блоков в конфигурации задач, не перечисляются в списке перекрестных ссылок.

1300

"File not found '<name>'"

Файл, на который указывает объект глобальных переменных, не существует. Проверьте правильность указания пути.

1301

"Analyze-Library not found! Code for analyzation will not be generated."

Функция анализа используется, но библиотека `analyzation.lib` отсутствует. Добавьте в проект эту библиотеку.

1302

"New externally referenced functions inserted. Online Change is therefore no longer possible!"

Со временем последней загрузки вы подключили библиотеку, содержащую функции, отсутствующие в системе. Перезагрузите проект полностью.

1400

"Unknown Pragma '<Name>' is ignored!"

Данная директива `'pragma'` не поддерживается компилятором. См. "Директивы компилятора".

1401

"The struct '<name>' does not contain any elements."

Структура не содержит элементов, но переменные этого типа занимают 1 байт памяти.

1410

"'RETAIN' and 'PERSISTENT' do not have any effect in functions"

Объявленные внутри функции локальные `RETAIN` и `PERSISTENT` переменные размещаются в обычной области локальных переменных.

1411

"Variable '<name>' in the variable configuration isn't updated in any task"

Верхний уровень экземпляра переменной не участвует в вызове ни одной задачи, поэтому она не копируется из образа процесса.

```
Пример:  
Variable Configuration:  
  VAR_CONFIG  
  plc_prg.aprg.ainst.in AT %IB0 : INT;  
  END_VAR  
plc_prg:  
  index := INDEXOF(aprg);
```

На программу `aprg` есть ссылка, но нет вызова. Поэтому `plc_prg.aprg.ainst` никогда не примет актуальное значение `%IB0`.

1412

"Unexpected token '<Name>' in pragma {pragma name}"

Вы используете не допустимую в данном месте либо некорректно записанную директиву компилятора `'pragma'`.

1413

"'<Name>' is not a valid key for list '<Name>'. The key will be ignored"

Указан несуществующий список параметров. Используйте менеджер параметров для выбора доступных списков

1500

"Expression contains no assignment. No code was generated."

Результат выражения нигде не использован. Поэтому код выражения не генерируется

1501

"String constant passed as 'VAR_IN_OUT': '<Name>' must not be overwritten!"

Такая константа не может быть записана в POU, поскольку нет возможности определить ее размер.

1502

"Variable '<Name>' has the same name as a POU. The POU will not be called!"

Переменная имеет одинаковое с POU имя.

Пример:

```
PROGRAM a
...
VAR_GLOBAL
  a: INT;
END_VAR
...
```

a; (* Вместо вызова POU загружается переменная. *)

1503

"The POU '<name>' has no outputs. Box result is set to 'TRUE'."

Выход POU, не имеющего определенных выходов, соединен в FBD. Автоматически присваивание дает TRUE.

1504

"'<name>' ('<number>'): Statement may not be executed due to the evaluation of the logical expression"

Возможно, не все ветви логического выражения будут вычисляться.

Пример:

```
IF a AND funct(TRUE) THEN ....
```

Если a равно FALSE, то funct не будет вызвана.

1505

"Side effect in '<Name>'! Branch is probably not executed !"

Первый вход POU равен FALSE, поэтому выражения в боковой ветви других входов не будут вычисляться.

1506

"Variable '<name>' has the same name as a local action. The action will not be called!"

Переменная имеет то же имя, что и действие. Действие не вызывается. Переименуйте переменную или действие.

1507

"Instance '<name>' has the same name as a function. The instance will not be called."

Вы вызываете в ST экземпляр, имеющий одинаковое с функцией имя. Будет вызвана функция. Используйте разные имена.

1550

"Multiple calls of the POU '<Name>' in one network may lead to undesired side effects"

Несколько вызовов одного POU в одной цепи могут дать побочные эффекты

1600

"Open DB unclear (generated code may be erroneous)."

Исходная Siemens программа не содержит инструкции выбора DB.

1700

"Input not connected."

Входной элемент CFC не имеет присваивания. Поэтому код не генерируется.

1750

"Step '<Name>': the minimal time is greater than the maximal time!"

Откройте диалог 'Step attributes' данного шага и проверьте задание макс. и мин. времён.

1800

"<name>(element #<element number>): Invalid watchexpression '<name>'"

Элемент визуализации содержит выражение, которое невозможно вычислить. Проверьте имена и заместители в выражении.

1801

"'<name> (number): No Input on Expression '<name>' possible"

В конфигурации визуализации объекта в качестве входного поля указано выражение. Здесь допускается только переменная.

1802

"<Visualization object>(Element number): Bitmap '<name>' was not found"

Проверьте, существует ли указанный растровый рисунок.

1803

""<name>'(<number>): "The print action would not supported for web- and target visualisation"

Вывод на печать задан как действие при тревоге. Это не будет работать в Web- и Target-визуализации.

1804

""<name>'(<number>): The font '<name>' is not supported by the target."

В визуализации вы использовали шрифт, не поддерживаемый целевой платформой. См. категорию Визуализация в опциях платформы

1805

""<name>'(<number>): 'Store trend data in PLC' should be set."

Вы используете элемент сохранения данных трендов, но эта опция отключена в категории Визуализации в опциях платформы.

1806

""<name>'(<number>): The target setting 'Alarm handling in the PLC' should be set."

Вы используете элемент "тревога" в визуализации, но эта опция отключена в категории Визуализация в опциях платформы.

1807

""<name> (<number>): No message window for alarms for target visualization"

Обратите внимание, что действие "сообщение" не поддерживается в целевой визуализации!

1850

"Input variable at %IB<number> is used in task '<name>' but updated in another task"

Проверьте, какие задачи используют данную переменную, и убедитесь, что это не может привести к нежелательным эффектам. Как правило, обновление переменной выполняется в задаче с наивысшим приоритетом.

1851

"Output variable at %IQ<number> is used in task '<name>' but updated in another task"

Проверьте, какие задачи используют данную переменную, и убедитесь, что это не может привести к нежелательным эффектам. Как правило, обновление переменной выполняется в задаче с наивысшим приоритетом.

1852

"CanOpenMaster might not be called cyclically in event task '<name>!' Set modul parameter UpdateTask!"

CanOpen Master вызывается из указанной задачи, управляемой событием. Если вы хотите заставить его работать циклически, то задайте соответствующую задачу посредством параметра UpdateTask в диалоге dialog 'Module parameters' конфигуратора ПЛК.

1853

"A PDO (index: '<number>') might not be updated cyclically in event task '<name>'"

Данный PDO управляется указанной задачей, управляемой событием. Но если вы хотите получить ее циклические вызовы, то нужно присвоить PDO соответствующую циклическую задачу, то есть поместить в нее ссылки на ввод-вывод.

1900

"POU '<name>' (main routine) is not available in the library"

Начальный POU (т.е. PLC_PRG) не доступен, если проект используется как библиотека.

1901

"Access Variables and Variable Configurations are not saved in a library!"

Переменные общего доступа и конфигурируемые переменные не записываются в библиотеку.

1902

"'<Name>': is no Library for the current machine type!"

Файл .obj сгенерирован для другого устройства

1903

"<Name>: is no valid Library"

Данный файл не удовлетворяет требованиям формата библиотеки выбранной платформы

1904

"The constant '<Name>' hides a constant of the same name in a library"

В вашем проекте определена константа, имеющая то же имя, что и в присоединенной библиотеке. Библиотечная переменная будет переопределена!

1970

"Parameter manager: List '<Name>', Column '<Name>', Value '<Name>' could not be imported!"

Проверьте вхождения файла импорта *.rgm, не соответствующие текущей конфигурации Менеджера параметров.

1980

Global network variables '<Name>' '<Name>': simultaneous reading and writing may result in loss of data!"

В конфигурации списка сетевых ('Global variables list' → 'Properties') переменных включена опция 'Read' and 'Write'. Это может привести к потере данных в процессе коммуникации.

1990

"No 'VAR_CONFIG' for '<name>'"

Для данной переменной не определен адрес в Variable_Configuration (VAR_CONFIG). Откройте окно Variable_Configuration в разделе ресурсов (Resources) и введите соответствующее определение (используйте команду 'Insert 'All instance paths').

2500

"Task '<task name>': no cycle time specified for cyclic task"

В Task configuration определена циклическая задача, для которой не задано время цикла. Задайте соответствующее время в диалоге 'Taskattributes' параметр "Interval".

10.28 Ошибки

3100

"Code too large. Maximum size: '<number>' Byte (<number>K)"

Достигнут максимальный размер кода программы. Уменьшите размер проекта.

3101

"Total data too large. Maximum size: '<number>' Byte (<number>K)"

Недостаточно памяти данных. Уменьшите число использованных в приложении данных.

3110

"Error in Library '<Name>'."

Файл .hex не соответствует формату INTEL Hex.

3111

"Library '<Name>' is too large. Maximum size: 64K"

Файл .hex превышает допустимый размер.

3112

"Nonrelocatable instruction in library."

Файл .hex содержит перемещаемые инструкции. Код библиотеки не может быть скомпонован.

3113

"Library code overwrites function tables."

Область кода и таблицы функций пересекаются.

3114

"Library uses more than one segment."

Таблицы и код в .hex файле используют более одного сегмента.

3115

"Unable to assign constant to VAR_IN_OUT. Incompatible data types."

Внутренний формат указателя строковых констант не может быть преобразован во внутренний формат указателя для VAR_IN_OUT, потому что данные используют указатели "near", а строковые константы "huge" или "far". Если можно, измените опции целевой платформы.

3116

"Function tables overwrite library code or a segment boundary."

Код 16бх: Внешняя библиотека не может быть использована с данными опциями платформы Перекомпилируйте библиотеку с соответствующими параметрами.

3117

"<Name> (<Zahl>): Expression too complex. No more registers available"

Данное выражение слишком сложное для обработки посредством доступных регистров. Уменьшите вложенность выражения, используйте промежуточные переменные.

3120

"Current code-segment exceeds 64K."

Полученный код сегмента больше, чем 64К. Возможно, слишком много кода инициализации.

3121

"POU too large."

POU не должен превышать 64К.

3122

"Initialisation too large. Maximum size: 64K"

Код инициализации функции либо переменных не должен превышать 64К.

3123

"Data segment too large: segment '<Number>%s', size <size> bytes (maximum <number> bytes)"

Сообщите проблему изготовителю контроллера.

3124

"String-constant too long: <name1> (Maximum 253)"

Уменьшите длину строковой константы.

3130

"User-Stack too small: '<number>' DWORD needed, '<number>' DWORD available."

Слишком большая глубина вложений. Увеличьте размер стека в опциях платформы или отключите опцию ‚Debug‘ (‘Project’ → ‘Options’ → ‘Build’).

3131

"User-Stack too small: '<number>' WORD needed, '<number>' WORD available."

Сообщите проблему изготовителю контроллера.

3132

"System-Stack too small: '<number>' WORD needed, '<number>' WORD available."

Сообщите проблему изготовителю контроллера.

3150

"Parameter <number> of function '<name>': Cannot pass the result of a IEC-function as string parameter to a C-function."

Используйте промежуточную переменную, которой присваивается результат МЭК функции.

3160

"Can't open library file '<name>'."

Библиотека <name> включена в проект, но отсутствует по указанному пути.

3161

"Library '<name>' contains no codesegment"

Файл .obj библиотеки должен иметь хотя бы одну C функцию. Вставьте пустую функцию в .obj, не объявляя ее в .lib файле.

3162

"Could not resolve reference in Library '<name>'(Symbol '<name>', Class '<name>', Type '<name>')"

Объектный файл .obj содержит внешнюю ссылку (symbol). Проверьте опции C компилятора.

3163

"Unknown reference type in Library '<name>' (Symbol '<name>' , Class '<name>' , Type '<name>')"

Объектный файл .obj содержит ссылку, не разрешенную генератором кода. Проверьте опции C компилятора.

3200

"<name>: Boolean expression to complex"

Не хватает размера временной памяти в целевой системе для вычисления выражения. Разделите выражение на несколько частей с помощью промежуточных переменных.

3201

"<name> (<network>): A network must not result in more than 512 bytes of code"

Внутренние переходы не могут быть разрешены. Активизируйте опцию "Use 16 bit jump offsets" в опциях платформы 68k.

3202

"Stack overrun with nested string/array/structure function calls"

Используются вложенные вызовы CONCAT(x, f(i)). Разделите вызов на два выражения.

3203

"Expression too complex (too many used address registers)."

Разделите выражение на несколько частей.

3204

"A jump exceeds 32k Bytes"

Переход не может быть более, чем 32767 байт кода.

3205

"Internal Error: Too many constant strings"

В ROU можно использовать 3000 строковых констант.

3206

"Function block data exceeds maximal size"

Функциональный блок требует не более 32767 байт кода.

3207

"Array optimization"

Оптимизация доступа к массиву нарушена, поскольку при вычислении индекса используется вызов функции.

3208

"Conversion not implemented yet"

Использована функция преобразования, не поддерживаемая в данной платформе.

3209

"Operator not implemented"

Используется оператор, не реализованный для данных типов в текущем генераторе кода.

3210

"Function '<Name>' not found"

Вызывается функция, отсутствующая в проекте.

3211

"Max string usage exceeded"

Переменная типа строка не должна входить в выражение более 10 раз.

3212

"Wrong library order at POU <POU name>"

Порядок библиотек для этого POU не соответствует cslib.hex файлу. Исправьте порядок (только для r68K).

3250

"Real not supported for 8 Bit Controller"

В данном генераторе кода не реализована поддержка Real.

3251

"date of day types are not supported for 8 Bit Controller"

В данном генераторе кода не реализована поддержка типов дата и время дня.

3252

"size of stack exceeds <number> bytes"

Размер стека превышает установленный предел.

3253

"Could not find hex file: '<Name>' "

Не найден hex файл.

3254

"Call to external library function could not be resolved."

Функция отсутствует во внешней библиотеке.

3255

"Pointers are not supported for 8 bit controllers."

Не используйте указатели на 8-разрядной платформе.

3260

"Function ,<name>' has too many arguments: Increase the size of the argument stack in the target settings."

Функция имеет слишком много параметров. Если это возможно, то увеличьте размер стека в диалоге Target Platform, вкладка Target Settings. Если изменение размера стека запрещено, то обратитесь к изготовителю контроллера.

3400

"An error occurred during import of Access variables"

Файл .exp содержит ошибки в секции Access variables.

3401

"An error occurred during import of variable configuration"

Файл .exp содержит ошибки в секции variables configuration.

3402

"An error occurred during import of global variables"

Файл .exp содержит ошибки в секции global variables.

3403

"Could not import <name>"

Файл .exp содержит ошибки в секции <name>.

3404

"An error occurred during import of task configuration"

Файл .exp содержит ошибки в секции определения задач.

3405

"An error occurred during import of PLC configuration"

Файл .exp содержит ошибки в секции конфигурации ПЛК.

3406

"Two steps with the name '<name>'. Second step not imported."

Два SFC шага с одинаковыми именами в exp файле. Переименуйте один из шагов.

3407

"Predecessor step '<name>' not found"

Шаг <name> отсутствует в .exp файле.

3408

"Successor step '<name>' not found"

Шаг <name> отсутствует в .exp файле.

3409

"No succeeding transition for step '<name>' "

Отсутствует определение перехода шага <name>. Исправьте .exp файл.

3410

"No succeeding step for transition '<name>'"

Отсутствует определение шага для перехода <name>. Исправьте .exp файл.

3411

"Step '<name>' not reachable from initial step"

В .exp файле утрачена связь между шагом <name> и начальным шагом.

3412

"Macro '<name>' not imported"

Проверьте файл экспорта.

3413

"Error during import of the CAMs."

Файл экспорта (*.exp) содержит ошибки в CAM. Проверьте .exp файл.

3414

"Error during import of the CNC program list"

Файл экспорта (*.exp) содержит ошибки в CNC программе.

3415

Error during import of the Alarm configuration"

Вы используете файл экспорта (*.exp), содержащий ошибочные данные в определении тревог в (Alarm Configuration). Проверьте файл экспорта.

3450

"PDO'<PDO-name>': Missing COB-Id!"

Нажмите кнопку ,Properties' в конфигураторе ПЛК и задайте COB ID для <PDO Name>.

3451

"Error during load: EDS-File '<name>' could not be found, but is referenced in hardware configuration!"

Возможно, указан ошибочный путь к EDS файлу. Проверьте путь (,Project'→'Options' →'Directories').

3452

"The module '<name>' couldn't be created!"

Файл описания устройства <name> отсутствует. Возможно, он был изменен после настройки конфигурации в CoDeSys или поврежден.

3453

"The channel '<name>' couldn't be created!"

Файл описания устройства не соответствует текущей конфигурации. Возможно, он был изменен после настройки конфигурации в CoDeSys или поврежден.

3454

"The address '<name>' points to an used memory!"

Опция контроля адресов 'Check for overlapping addresses' включена и обнаружено пересечение адресов. Заметьте, что контроль опирается на типы данных для определения их размера, а не на параметр 'size' в файле конфигурации.

3455

"Error during load: GSD-File '<name>' could not be found, but is referenced in hardware configuration!"

Вероятно, путь к файлу Profibus конфигурации указан не верно. Проверьте настройку директорий (,Project'→ 'Options'→ 'Directories').

3456

"The profibus device '<name>' couldn't be created!"

Файл описания устройства <name> не соответствует текущей конфигурации. Возможно, он был изменен после настройки конфигурации в CoDeSys или поврежден.

3457

"Error in module description!"

Проверьте файл описания устройства.

3458

"The PLC-Configuration couldn't be created! Check the configuration files."

Проверьте, все ли необходимые файлы описания устройств существуют. Возможна ошибка в указании пути.

3459

"The selected baudrate is not supported."

Измените настройки в диалоге параметров CAN в соответствии со скоростью передачи, заданной в GSD файле.

3460

"3S_CanDrv.lib has the wrong version."

Убедитесь, что 3S_CanDrv.lib, включенная в проект, соответствует текущей версии.

3461

"3S_CanOpenMaster.lib has the wrong version."

Убедитесь, что 3S_CanOpenMaster.lib, включенная в проект, соответствует текущей версии.

3462

"3S_CanOpenDevice.lib has the wrong version."

Убедитесь, что 3S_CanOpenDevice.lib, включенная в проект, соответствует текущей версии.

3463

"3S_CanOpenManager.lib has the wrong version."

Убедитесь, что 3S_CanOpenManager.lib, включенная в проект, соответствует текущей версии.

3464

"3S_CanNetVar.lib has the wrong version."

Убедитесь, что 3S_CanNetVar.lib, включенная в проект, соответствует текущей версии.

3465

"CanDevice: Sub indices have to be numerated sequentially"

В списках параметров CanDevice подиндексы должны быть пронумерованы последовательно без разрывов. Проверьте список в Менеджере Параметров.

3466

"CAN network variables: No CAN controller found in the PLC configuration"

Определен ряд сетевых переменных для CAN (Resources, Global Variables), но в конфигурации ПЛК CAN контроллер не определен.

3468

"CanDevice: Update task not available in the task configuration."

Задача обновления, определенная в базовых настройках (Base Settings) CANdevice, должна присутствовать в конфигураторе задач проекта (Task Configuration).

3469

"The CanOpenMaster can not be called. Please assign a task manually."

Присвойте задачу, которая будет вызывать Master через параметр UpdateTask в диалоге параметров модуля PLC Configuration.

3470

"Invalid name in parameter UpdateTask"

Откройте диалог параметров CanMasters Module в PLC Configuration. Проверьте параметр UpdateTask. Указанная задача должна присутствовать в проекте. Если вы не можете установить здесь нужную задачу, проверьте значение UpdateTask в файле описания устройства.

3500

"No 'VAR_CONFIG' for '<Name>'"

Вставьте объявление этой переменной в список глобальных объявлений 'Variable_Configuration'.

3501

"No address in 'VAR_CONFIG' for '<name>'"

Присвойте адрес этой переменной в списке глобальных объявлений 'Variable_Configuration'.

3502

"Wrong data type for '<name>' in 'VAR_CONFIG'"

Определение переменной в списке 'Variable_Configuration' отличается по типу данных от объявления в POU.

3503

"Wrong data type for '<name>' in 'VAR_CONFIG'"

В списке глобальных объявлений 'Variable_Configuration' и в POU переменная объявлена с разными адресами.

3504

"Initial values are not supported for 'VAR_CONFIG'"

Переменная Variable_Configuration объявлена с адресом и начальным значением. Но начальное значение может быть определено для входной переменной только без присвоения адреса.

3505

"'<name>'is no valid instance path"

Variable_Configuration определяет не существующую переменную.

3506

"Access path expected"

В списке глобальных переменных Access Variables путь задан неверно.
Правильно:<Identifier>:<Access path>:<Type> <Access mode>.

3507

"No address specification for 'VAR_ACCESS'-variables"

В списке глобальных переменных Access Variables содержится указание адреса переменной. Это не допустимо.

Корректное определение: <Identifier>:<Access path>:<Type> <Access mode>

3550

"Duplicate definition of identifier '<name>'"

Две задачи имеют одинаковые имена, переименуйте одну из них.

3551

"The task '<name>' must contain at least one program call"

Добавьте вызов программы или удалите задачу.

3552

"Event variable '<name>' in task '<name>' not defined"

Переменная-событие задана в поле 'Single' в диалоге свойств задачи, но не определена глобально в проекте. Объявите переменную как глобальную.

3553

"Event variable '<name>' in task '<name>' must be of type 'BOOL'"

Используйте переменную типа BOOL как переменную-событие в коде 'Single' диалога свойств задачи.

3554

"Task entry '<name>' must be a program or global function block instance"

В поле 'Program call' указана функция или неопределенный POU. Задайте корректное имя.

3555

"The task entry '<name>' contains invalid parameters"

В поле 'Append program call' указаны параметры, не совместимые с объявлением POU.

3556

"Tasks are not supported by the currently selected target"

Указанная конфигурация задач не может использоваться в данной целевой платформе. Измените конфигурацию.

3557

"Maximum number of Tasks ('<number>') exceeded"

Достигнут максимум числа задач для данной платформы. Измените конфигурацию.

Внимание: Не пытайтесь редактировать XML файл конфигурации задач.

3558

"Priority of task '<name>' is out of valid range between '<lower limit>' and '<upper limit>'"

Заданный приоритет задачи не поддерживается в данной целевой системе. Измените конфигурацию.

3559

"Task '<name>': Interval-Tasks are not supported by the current target"

Текущая конфигурация задач содержит интервальную задачу, не поддерживаемую в данной целевой системе. Измените конфигурацию.

3560

"Task '<name>': free wheeling tasks are not supported by the current target"

Текущая конфигурация задач содержит периодическую (free wheeling) задачу, не поддерживаемую в данной целевой системе. Измените конфигурацию.

3561

"Task '<name>': event tasks are not supported by the current target"

Текущая конфигурация задач содержит событийно-управляемую задачу, не поддерживаемую в данной целевой системе. Измените конфигурацию.

3562

"Task '<name>': external event tasks are not supported by the current target"

Текущая конфигурация задач содержит управляемую внешним событием задачу, не поддерживаемую в данной целевой системе. Измените конфигурацию.

3563

"The interval of task '<name>' is out of valid range between '<lower limit>' and '<upper limit>'"

Измените величину интервала в диалоге настройки задач.

3564

"The external event '<name>' of task '<name>' is not supported by the current target"

Заданное внешнее событие не поддерживается в данной целевой системе. Измените конфигурацию.

3565

"Maximum number of event tasks ('<number>') exceeded"

Данная целевая платформа не поддерживает такое количество задач, управляемых событиями. Измените конфигурацию.

3566

"Maximum number of interval tasks ('<number>') exceeded"

Данная целевая платформа не поддерживает такое количество интервальных задач, управляемых событиями. Измените конфигурацию.

3567

"Maximum number of free wheeling tasks ('<number>') exceeded"

Данная целевая платформа не поддерживает такое количество периодических (free wheeling) задач, управляемых событиями. Измените конфигурацию.

3568

"Maximum number of external interval tasks ('<number>') exceeded"

Данная целевая платформа не поддерживает такое количество задач, управляемых внешними событиями. Измените конфигурацию.

3569

"POU '<name>' for system event '<name>' not defined"

POU, который должен вызываться событием '<name>', отсутствует в проекте.

3570

"The tasks '<name>' and '<name>' share the same priority"

Измените конфигурацию задач так, чтобы обе задачи имели разный приоритет.

3571

"The library 'SysLibCallback' is not included in the project! System events cannot be generated."

Для управления задачами по событиям нужна библиотека SysLibCallback.lib. Включите библиотеку в проект или измените конфигурацию.

3575

"Task '<name>': the cycle time has to be a multiple of <number> μs."

Поправьте время цикла задачи в диалоге Taskattributes. Эта величина должна быть кратна времени системного 'тика' вашей целевой системы.

3600

"Implicit variables not found!"

Дайте команду ,Rebuild all'. Если это не поможет, свяжитесь с изготовителем ПЛК

3601

"<name> is a reserved variable name"

Имя данной переменной зарезервировано генератором кода, измените его.

3610

" '<Name>' is not supported"

Данное свойство не поддерживается в установленной целевой системе.

3611

"The given compile directory '<name>' is invalid"

В ,Project' ,Options' ,Directories' задана несуществующая директория для файлов компилятора.

3612

"Maximum number of POU's (<number>) exceeded! Compile is aborted."

В проекте используется слишком много POU. Измените максимум POU в Target Settings / Memory Layout.

3613

"Build canceled"

Компиляция прервана пользователем.

3614

"Project must contain a POU named '<name>' (main routine) or a taskconfiguration"

Создайте главный POU (т.е. PLC_PRG) или задайте конфигурацию задач.

3615

"<Name> (main routine) must be of type program"

Главный POU (т.е. PLC_PRG) должен иметь тип программа.

3616

"Programs musn't be implemented in external libraries"

Проект, который предполагается сохранить как внешнюю библиотеку? содержит программы. Они не будут доступны в библиотеке.

3617

"Out of memory"

Увеличьте размер виртуальной памяти вашего компьютера.

3618

"BitAccess not supported in current code generator!"

Битовый доступ не поддерживается генератором кода данной целевой системы.

3619

"Object file '<name>' and library '<name>' have different versions!"

Убедитесь, что файлы *.lib и *.obj resp. *.hex соответствуют одной версии библиотеки. Проверьте даты создания этих файлов.

3620

"The POU '<name>' must not be present inside a library"

Вы пытаетесь сохранить библиотеку в формате версии 2.1. В этой версии библиотека не может содержать PLC_PRG, удалите или переименуйте его.

3621

"Cannot write compile file '<name>'"

Вероятнее всего, в директории, указанной для файлов компилятора, уже имеется файл с таким именем, имеющий атрибут "Только чтение ". Удалите данный файл либо измените ему права доступа.

3622

"The symbol file '<name>' could not be created"

Вероятнее всего, в директории, указанной для символьных файлов (обычно это директория проекта), уже имеется файл с таким именем, имеющий атрибут "Только чтение ". Удалите данный файл либо измените ему права доступа.

3623

"Cannot write boot project file '<name>'"

Вероятнее всего, в директории, указанной для загрузочных файлов (специфичных для целевой платформы), уже имеется файл с таким именем, имеющий атрибут "Только чтение ". Удалите данный файл либо измените ему права доступа.

3624

"Target setting <targetsetting1>=<set value> not compatible with <targetsetting2>=<set value>"

Проверьте и исправьте данные установки в диалоге Targetsettings dialogs (вкладка Resources). Если они недоступны для редактирования, то обратитесь к изготовителю контроллера.

3700

" POU with name '<name>' is already in library '<name>'"

Имя POU проекта уже использовано в библиотеке, измените его.

3701

"Name used in interface is not identical with POU Name"

Используйте команду **'Project' 'Rename object'** для изменения памяти POU в организаторе объектов либо измените имя в окне объявления POU. Имя POU следует за одним из ключевых слов: PROGRAM, FUNCTION или FUNCTIONBLOCK.

3702

"Overflow of identifier list"

Не более 100 идентификаторов могут быть использованы при объявлении одной переменной.

3703

"Duplicate definition of identifier '<Name>'"

Убедитесь, что только один идентификатор '<Name>' присутствует в разделе объявлений POU.

3704

"data recursion: "<POU 0> -> <POU 1> -> .. -> <POU 0>""

Применен не допустимый вызов экземпляром функционального блока самого себя.

3705

"<Name>: VAR_IN_OUT in Top-Level-POU not allowed, if there is no Task-Configuration"

Создайте конфигурацию задач или убедитесь, что переменные VAR_IN_OUT не используются в PLC_PRG.

3720

"Address expected after 'AT'"

После ключевого слова AT должен быть указан корректный адрес.

3721

"Only 'VAR' and 'VAR_GLOBAL' can be located to addresses"

Поместите объявление в область VAR или VAR_GLOBAL.

3722

"Only 'BOOL' variables allowed on bit addresses"

Только переменные типа BOOL могут адресовать биты. Измените адрес или тип переменной.

3726

"Constants can not be laid on direct addresses"

Константы нельзя располагать по прямым адресам.

3727

"No array declaration allowed on this address"

Объявление может быть произведено по указанному адресу. Измените адрес.

3728

"Invalid address: '<address>'"

Указанный адрес не определен для заданной конфигурации ПЛК. Измените адрес или конфигурацию ПЛК.

3729

"Invalid type '<name>' at address: '<Name>' "

Переменная данного типа не может быть размещена по указанному адресу. Например: адрес AT %I0:WORD; не допустим, если включено выравнивание по четным адресам. Данная ошибка может возникнуть при попытке разместить массив по недопустимому прямому адресу.

3740

"Invalid type: '<Name>' "

Ошибка в типе данных объявления.

3741

"Expecting type specification"

Ключевое слово или оператор использован вместо типа данных

3742

"Enumeration value expected"

В определении перечисления пропущен идентификатор после скобки либо разделитель.

3743

"Integer number expected"

Перечисления можно инициализировать только целыми значениями (INT).

3744

"Enum constant '<name>' already defined"

Проверьте соблюдение следующих правил при объявлении перечислений:

- Все значения в одном перечислении должны быть уникальны.
- Во всех глобальных перечислениях все значения должны быть уникальны.
- Во всех локальных перечислениях все значения должны быть уникальны.

3745

"Subranges are only allowed on Integers!"

Переменные с ограниченным диапазоном образуются только на целочисленных типах.

3746

"Subrange '<name>' is not compatible with Type '<name>'"

Один из пределов диапазона выходит за область значений базового типа.

3747

"unknown string length: '<name>'"

Для определения длины строки используется ошибочная константа.

3748

"More than three dimensions are not allowed for arrays"

Нельзя использовать массивы с размерностью более трех. Используйте ARRAY OF ARRAY при необходимости.

3749

"lower bound '<name>' not defined"

Не задана константа, определяющая нижнюю границу диапазона.

3750

"upper bound '<name>' not defined"

Не задана константа, определяющая верхнюю границу диапазона.

3751

"Invalid string length '<number of characters>'"

Заданный размер строки превышает допустимый в данной целевой системе.

3752

"More than 9 dimensions are not allowed for nested arrays"

Массив может быть 1- 2- или 3-мерный. Размерность можно еще увеличить путем вложений массивов (например, "arg: ARRAY [0..2,0..2,0..2] OF ARRAY [0..2,0..2,0..2] OF ARRAY [0..2,0..2,0..2, 0..2] OF DINT". Максимальная размерность не должна превышать 9. Данная ошибка говорит о превышении этого ограничения. Уменьшите вложенность массивов.

3760

"Error in initial value"

Используйте для инициализации значение, совместимое с типом переменной. Изменяя объявление, воспользуйтесь диалогом объявлений переменных (Shift/F2 или 'Edit'Autodeclare').

3761

""VAR_IN_OUT' variables must not have an initial value."

Удалите инициализацию в объявлении переменной VAR_IN_OUT.

3780

""VAR', 'VAR_INPUT', 'VAR_OUTPUT' or 'VAR_IN_OUT' expected"

В следующей за определением имени POU строке должно быть одно из перечисленных ключевых слов.

3781

""END_VAR' or identifier expected"

Введите корректное определение END_VAR в данной строке окна объявлений.

3782

"Unexpected end"

В разделе объявлений: добавьте ключевое слово END_VAR в конце раздела.

В разделе кода: добавьте инструкцию, заканчивающую команду (например, END_IF).

3783

""END_STRUCT' or identifier expected"

Проверьте правильность окончания определения типа.

3784

""The current target doesn't support attribute <attribute name>"

Данная целевая система не поддерживает переменные такого типа (например, RETAIN, PERSISTENT)

3800

""The global variables need too much memory. Increase the available memory in the project options."

Увеличьте число сегментов в опциях диалога Project' ,Options' ,Build'.

3801

""The variable '<Name>' is too big. (<size> byte)"

Переменная использует тип, занимающий более одного сегмента

Размер сегмента определяется настройкой целевой платформы. Если вы не нашли этого параметра в опциях памяти, свяжитесь с изготовителем ПЛК.

3802

"Out of retain memory. Variable '<name>', <number> bytes."

Израсходована вся память Retain переменных. Размер этой области определяется настройкой целевой платформы. Если вы не нашли этого параметра в опциях памяти, свяжитесь с изготовителем ПЛК (*обратите внимание*: если хотя бы одна переменная функционального блока объявлена как Retain, то все данные экземпляра сохраняются в Retain области!)

3803

"Out of global data memory. Variable '<name>', ,<number>' bytes."

Израсходована вся память глобальных переменных. Размер этой области определяется настройкой целевой платформы. Если вы не нашли этого параметра в опциях памяти, свяжитесь с изготовителем ПЛК

3820

""VAR_OUTPUT' and 'VAR_IN_OUT' not allowed in functions"

В функциях нельзя использовать выходы или входы-выходы.

3821

"At least one input required for functions"

Функция должна иметь как минимум один параметр.

3840

"Unknown global variable '<name>!"

В POU используется глобальная переменная VAR_EXTERNAL, необъявленная в списке глобальных.

3841

"Declaration of '<name>' do not match global declaration!"

Тип переменной в объявлении VAR_EXTERNAL не совпадает с типом в списке глобальных объявлений.

3850

"Declaration of an unpacked struct '<name>' inside a packed struct '<name>' is not allowed!"

Такое определение структуры нарушает распределение памяти. Измените определение соответствующим образом.

3900

"Multiple underlines in identifier"

Удалите повторное подчеркивание в идентификаторе.

3901

"At most 4 numerical fields allowed in addresses"

Попытка присваивания по прямому адресу, содержащему более 4-х уровней (например, %QB0.1.1.0.1).

3902

"Keywords must be uppercase"

Используйте заглавные буквы в ключевых словах или включите опцию ,Autoformat' in ,Project' ,Options'.

3903

"Invalid duration constant"

Нотация константы не соответствует МЭК 61131-3.

3904

"Overflow in duration constant"

Заданное значение длительности нельзя преобразовать во внутренний формат. Максимальное значение длительности: t#49d17h2m47s295ms.

3905

"Invalid date constant"

Нотация константы не соответствует МЭК 61131-3.

3906

"Invalid time of day constant"

Нотация константы не соответствует МЭК 61131-3.

3907

"Invalid date and time constant"

Нотация константы не соответствует МЭК 61131-3.

3908

"Invalid string constant"

Строковая константа содержит недопустимый символ.

4000

"Identifier expected"

В этой позиции должен быть корректный идентификатор.

4001

"Variable '<Name>' not declared"

Объявите переменную локально или глобально.

4010

"Type mismatch: Cannot convert '<Name>' to '<Name>'."

Проверьте, какой тип данных требуется этому оператору, и измените тип или используйте другую переменную.

4011

"Type mismatch in parameter '<Name>' of '<Name>': Cannot convert '<Name>' to '<Name>'."

Тип актуального параметра не может быть автоматически преобразован в тип формального параметра. Используйте другую переменную или явное преобразование типов.

4012

"Type mismatch in parameter '<Name>' of '<Name>': Cannot convert '<Name>' to '<Name>'."

Значение ошибочного типа присваивается входной переменной '<Name>'. Замените переменную или укажите соответствующий префикс константы.

4013

"Type mismatch in output '<Name>' of '<Name>': Cannot convert '<Name>' to '<Name>'."

Значение ошибочного типа присваивается выходной переменной '<Name>'. Замените переменную или укажите соответствующий префикс константы.

4014

"Typed literal: Cannot convert '<name>' to '<name>'"

Тип константы не соответствует заданному префиксу.

Например: SINT#255

4015

"Data type '<name>' illegal for direct bit access"

Прямая битовая адресация допускается только с целыми типами или битовыми строками, но не для прямоадресуемых переменных. Возможно, вы используете переменную типа REAL/LREAL или константу <var1>.<bit>, либо прямоадресуемую переменную.

4016

"Bit index '<number>' out of range for variable of type '<name>'"

Вы пытаетесь использовать бит, не определенный для данного типа переменных.

4017

"'MOD' is not defined for 'REAL'"

Оператор MOD применим только для целых типов.

4020

"Variable with write access or direct address required for 'ST', 'STN', 'S', 'R'"

Замените первый операнд переменной, имеющей доступ на запись.

4021

"No write access to variable '<name>' allowed"

Замените переменную на другую, имеющую доступ на запись.

4022

"Operand expected"

Добавьте операнд команды.

4023

"Number expected after '+' or '-'"

Введите число.

4024

"Expecting <Operator 0> or <Operator 1> or ... before '<Name>'"

Задайте значимый операнд в указанной позиции.

4025

"Expecting ':=' or '=>' before '<Name>'"

Введите один из двух операторов в указанной позиции.

4026

"'BITADR' expects a bit address or a variable on a bit address"

Используйте корректный битовый адрес (т.е. %IX0.1).

4027

"Integer number or symbolic constant expected"

Введите число или идентификатор переменной.

4028

"'INI' operator needs function block instance or data unit type instance"

Проверьте тип переменной, для которой используется оператор №1.

4029

"Nested calls of the same function are not possible."

Функция не должна вызывать сама себя.

Используйте промежуточные переменные.

4030

"Expressions and constants are not allowed as operands of 'ADR'"

Для выражений и констант извлечение адреса невозможно.

4031

""ADR' is not allowed on bits! Use 'BITADR' instead."

Используйте BITADR. Заметьте, что BITADR не дает физический адрес памяти.

4032

""<number>' operands are too few for '<name>'. At least '<number>' are needed"

Проверьте число операндов для данного оператора и добавьте недостающие.

4033

""<number>' operands are too many for '<name>'. At least '<number>' are needed"

Проверьте число операндов для данного оператора и уберите лишние.

4034

"Division by 0"

Вы пытаетесь выполнить деление на 0 с константами. Если вы намеренно хотите спровоцировать деление на ноль при исполнении, используйте переменную.

4035

"ADR must not be applied on 'VAR CONSTANT' if 'replaced constants' is activated"

Извлечение адреса для встроенных констант невозможно. Если необходимо, отключите опцию 'Replace Constants' in 'Project' 'Options' 'Build'.

4040

"Label '<name>' is not defined"

Определите метку с именем <LabelName> либо измените имя на существующее.

4041

"Duplicate definition of label '<name>'"

Метка '<Name>' определена повторно. Удалите одно из определений.

4042

"No more than <number> labels in sequence are allowed"

Число последовательных меток ограничено. Вставьте пустую инструкцию.

4043

"Format of label invalid. A label must be a name optionally followed by a colon."

Недопустимое имя метки или пропущено двоеточие в конце идентификатора.

4050

"POU '<name>' is not defined"

Определите POU с именем '<Name>' командой **'Project' 'Add Object'** или измените '<Name>' на существующий POU.

4051

"'<name>' is no function"

Вместо <Name> используйте функцию, определенную в проекте или библиотеках.

4052

"'<name>' must be a declared instance of FB '<name>'"

Используйте существующий экземпляр функционального блока или объявите его.

4053

"'<name>' is no valid box or operator"

Измените '<Name>' на имя POU или оператор, определенный в проекте.

4054

"POU name expected as parameter of 'INDEXOF'"

Заданный оператор не является именем POU.

4060

"'VAR_IN_OUT' parameter '<name>' of '<name>' needs variable with write access as input"

Для параметра VAR_IN_OUT должен иметь доступ на запись, поскольку его значение может быть изменено в POU.

4061

"'VAR_IN_OUT' parameter '<name>' of '<name>' must be used."

Для параметра VAR_IN_OUT нужна переменная с доступом на запись, поскольку ее значение может быть изменено в POU.

4062

"No external access to 'VAR_IN_OUT' parameter '<name>' of '<name>'."

Параметр VAR_IN_OUT можно использовать только внутри POU, поскольку он передается по ссылке.

4063

""VAR_IN_OUT' parameter '<name>' of '<name>' must not be used with bit addresses."

Биты не имеют физических адресов. Измените переменную или прямой адрес.

4064

""VAR_IN_OUT' must not be overwritten in local action call!"

Удалите параметры VAR_IN_OUT из локального вызова действия.

4070

"The POU contains a too complex expression"

Уменьшите глубину вложений в выражении, разбив его на несколько отдельных выражений. Используйте промежуточные переменные.

4071

"Network too complex"

Разделите цепь на несколько цепей.

4072

"Inconsistent use of an action identifier in FB type ('<name>') and instance ('<name>')."

Вы определили действия в функциональном блоке fb, например a1 и a2, но при вызове действия вы указываете имя блока, а не экземпляра.

Например: fb.a1 вместо inst.a1.

4100

""^' needs a pointer type"

Вы пытаетесь использовать переменную, которая не является указателем.

4110

""[<index>]' needs array variable"

Попытка использовать индексы с переменной, которая не является массивом.

4111

"Index expression of an array must be of type 'INT'"

Используйте целочисленное выражение или переменную в индексах массива.

4112

"Too many indexes for array"

Проверьте число индексов в массиве (1, 2 или 3) и удалите лишние.

4113

"Too few indexes for array"

Проверьте число индексов в массиве (1, 2 или 3) и добавьте недостающие.

4114

"One of the constant indices is not within the array range"

Проверьте, принадлежит ли указанный индекс диапазону, заданному в объявлении массива.

4120

":"' needs structure variable'"

Слева от точки должен быть идентификатор структуры, экземпляра, функционального блока или имя функции либо программы.

4121

" '<Name>' is not a component of <object name>"

Компонент '<Name>' не входит в определение объекта <object name>.

4122

"'<name>' is not an input variable of the called function block"

Проверьте наименования входных переменных функционального блока и используйте один из них.

4200

"LD' expected"

Вставьте хотя бы одну инструкцию LD после jump метки в редакторе IL.

4201

"IL Operator expected"

Каждая инструкция IL должна начинаться с оператора или метки перехода.

4202

"Unexpected end of text in brackets"

Вставьте закрывающую скобку после текста.

4203

"<Name> in brackets not allowed"

Оператор <name> не допустим в скобках IL.

('JMP', 'RET', 'CAL', 'LDN', 'LD', 'TIME')

4204

"Closing bracket with no corresponding opening bracket"

Вставьте открывающую или удалите закрывающую скобку.

4205

"No comma allowed after ')"

Уберите запятую после закрывающей скобки.

4206

"Label in brackets not allowed"

Сместите метку так, чтобы она оказалась вне скобок.

4207

"'N' modifier requires operand of type 'BOOL', 'BYTE', 'WORD' or 'DWORD'"

Модификатор N применим только к логическим переменным.

4208

"Conditional Operator requires type 'BOOL'"

Убедитесь, что выражение дает логический результат? или используйте преобразование типа.

4209

"Function name not allowed here"

Замените вызов функции переменной или константой.

4210

"'CAL', 'CALC' and 'CALN' require a function block instance as operand"

Оставьте экземпляр функционального блока, который вы вызываете.

4211

"Comments are only allowed at the end of line in IL"

Переместите комментарий в конец строки или на отдельную строку.

4212

"Accumulator is invalid before conditional statement"

Значение аккумулятора не определено. Это может быть при выполнении инструкции не формирующей результат (например 'CAL').

4213

""S' and 'R' require 'BOOL' operand"

Используйте логическую переменную.

4250

"Another 'ST' statement or end of POU expected"

Строка должна начинаться с корректной ST инструкции.

4251

"Too many parameters in function '<name>'"

Число параметров больше, чем в объявлении функции.

4252

"Too few parameters in function '<name>'"

Число параметров меньше, чем в объявлении функции.

4253

""IF' or 'ELSIF' require 'BOOL' expression as condition"

Условие IF или ELSIF должно содержать логическое выражение.

4254

""WHILE' requires 'BOOL' expression as condition"

Условие 'WHILE' должно содержать логическое выражение.

4255

""UNTIL' requires 'BOOL' expression as condition"

Условие 'UNTIL' должно содержать логическое выражение.

4256

"'NOT' requires 'BOOL' operand"

Убедитесь, что за 'NOT' следует логическое выражение.

4257

"Variable of 'FOR' statement must be of type 'INT'"

Убедитесь, что счетчик итераций 'FOR' целая переменная или битовая строка (т.е. DINT, DWORD).

4258

"Expression in 'FOR' statement is no variable with write access"

Используйте для счетчика итераций переменную с доступом на запись.

4259

"Start value in 'FOR' statement is no variable with write access"

Начальное значение счетчика 'FOR' должно быть совместимо с типом переменной.

4260

"End value of 'FOR' statement must be of type 'INT'"

Конечное значение счетчика 'FOR' должно быть совместимо с типом переменной.

4261

"Increment value of 'FOR' statement must be of type 'INT'"

Значение приращения 'FOR' должно быть совместимо с типом переменной.

4262

"'EXIT' outside a loop"

Используйте 'EXIT' только в циклах 'FOR', 'WHILE' или 'UNTIL'.

4263

"Expecting Number, 'ELSE' or 'END_CASE'"

После 'CASE' должно быть число либо инструкция.

4264

"'CASE' requires selector of an integer type"

Убедитесь, что оператор имеет тип целое или битовая строка (т.е. DINT, DWORD).

4265

"Number expected after ','"

В перечислении в секторах CASE после запятой должно быть число.

4266

"At least one statement is required"

Вставьте инструкцию или хотя бы точку с запятой.

4267

"Function block call requires function block instance"

Объявите экземпляр вызываемого функционального блока или исправьте идентификатор.

4268

"Expression expected"

Вставьте выражение.

4269

""END_CASE' expected after 'ELSE'-branch"

Закройте 'CASE' после 'ELSE' с помощью 'END_CASE'.

4270

""CASE' constant '<name>' already used"

Селектор 'CASE' должен быть уникален в пределах одной 'CASE' инструкции.

4271

"The lower border of the range is greater than the upper border."

Измените границы диапазона так, чтобы начальный был меньше конечного.

4272

"Expecting parameter '<name>' at place <position> in call of '<name>'!"

При вызове функции можно использовать присваивание значений параметрам, но их последовательность нарушать нельзя.

4273

Parts of the 'CASE'-Range '<range>' already used in Range '<range>'

Убедитесь, что диапазоны начальных CASE селекторов не пересекаются.

4274

"Multiple 'ELSE' branch in 'CASE' statement"

CASE инструкция не должна содержать более одного 'ELSE'.

4300

"Jump requires 'BOOL' as input type"

Убедитесь, что вход jump - логическое выражение.

4301

"POU '<name>' need exactly <number> inputs"

Число входов не соответствует заданному в объявлении VAR_INPUT и VAR_IN_OUT.

4302

"POU '<name>' need exactly %d outputs".

Число выходов не соответствует заданному в объявлении VAR_OUTPUT.

4303

"'<name>' is no operator"

Замените '<name>' на допустимый оператор.

4320

"Non-boolean expression '<name>' used with contact"

Сигнал переключения контакта должен быть логического типа.

4321

"Non-boolean expression '<name>' used with coil"

Выходная переменная обмотки реле должна быть логического типа.

4330

"Expression expected at input 'EN' of the box '<name>' "

Задайте логическое выражение или соединение на вход EN.

4331

"Expression expected at input '<number>' of the box '<Name>' "

Вход <number> оператора не присвоен.

4332

Expression expected at input '<name>' of the box '<Name>' "

Не присвоен вход типа VAR_IN_OUT.

4333

"Identifier in jump expected"

Указанная метка перехода отсутствует.

4334

"Expression expected at the input of jump"

Задайте логическое выражение на вход `jump`. Переход выполняется при значении `TRUE`.

4335

"Expression expected at the input of the return"

Задайте логическое выражение на вход `RETURN`. Переход выполняется при значении `TRUE`.

4336

"Expression expected at the input of the output"

Недопустимое выражение на входе блока.

4337

"Identifier for input expected"

Задайте корректное выражение или идентификатор на входе блока.

4338

"Box '<name>' has no inputs"

Ни один из входов `POU '<Name>'` не содержит корректного присвоения.

4339

"Typemismatch at output: Cannot convert '<name>' to '<name>'."

Тип выхода не может быть преобразован к требуемому.

4340

"Jump requires 'BOOL' as input type"

Убедитесь, что вход `jump` - это логическое выражение.

4341

"Return needs a boolean input"

Убедитесь, что вход `RETURN` - это логическое выражение.

4342

"Expression expected at input 'EN' of the box '<name>'"

Задайте корректное логическое выражение на вход `EN`.

4343

"Values of Constants: '<name>'"

Вход объявлен как VAR_INPUT CONSTANT. Но для данного POU в диалоге 'Edit Parameters' задано выражение несовместимого типа.

4344

""S' and 'R' require 'BOOL' operand"

Задайте допустимое логическое выражение после инструкции Set или Reset.

4345

"Unzulässiger Typ für Parameter '<Name>' von '<Name>': Kann '<Typ>' nicht in '<Typ>' konvertieren."

Выражение на входе '<Name>' несовместимого типа.

4346

"Not allowed to use a constant as an output"

На выходе может быть только переменная или прямой адрес с доступом на запись.

4347

""VAR_IN_OUT' parameter needs variable with write access as input"

Для параметра VAR_IN_OUT допустима только переменная с доступом на запись.

4348

"Invalid program name '<name>'. A variable with the same name exists already."

Вы добавили элемент в редакторе SFC, который имеет то же имя, что и глобальная переменная, уже существующая в проекте. Переименуйте элемент.

4349

"Input or output in POU <name> has been deleted: Check all connections to the box. This error message disappears only after the SFC was edited"

Удален вход или выход POU. Проверьте все соединения данного POU в SFC.

4350

"An SFC-Action cannot be accessed from outside!"

Действие SFC можно вызывать только из POU, в котором оно объявлено. Эта ошибка может возникнуть при правильном вызове действия из SFC POU, если МЭК шаги не используются, а библиотека iecsf.lib включена в проект. В этом случае удалите библиотеку в менеджере библиотек и перекомпилируйте проект.

4351

"Step name is no identifier: '<name>'"

Переименуйте шаг либо выберите допустимый идентификатор.

4352

"Extra characters following valid step name: '<Name>'"

Удалите недопустимые символы в имени шага.

4353

"Step name duplicated: '<Name>'"

Удалите один из шагов.

4354

"Jump to undefined Step: '<Name>'"

Задайте существующий шаг для перехода или создайте такой шаг.

4355

"A transition must not have any side effects (Assignments, FB-Calls etc.)"

Условием перехода должно быть логическое выражение.

4356

"Jump without valid Step Name: '<Name>' "

Используйте правильно определенный идентификатор метки перехода.

4357

"IEC-Library not found"

Убедитесь что библиотека `iecsfc.lib` подключена в Менеджере библиотек и путь, заданный в 'Project' 'Options', определен верно.

4358

"Action not declared: '<name>'"

Убедитесь, что имя действия МЭК шага присутствует в SFC POU в Организаторе объектов и в прямоугольнике справа от классификатора в окне редактора SFC.

4359

"Invalid Qualifier: '<name>'"

В прямоугольнике слева от имени действия введите МЭК классификатор действия.

4360

"Time Constant expected after qualifier '<name>'"

Введите рядом с классификатором слева от имени действия константу времени.

4361

"'<name>' is not the name of an action"

Введите справа от классификатора имя действия или определенной в проекте переменной

4362

"Nonboolean expression used in action: '<name>'"

Вставьте логическую переменную или верное имя действия.

4363

"IEC-Step name already used for variable: '<Name>'"

Переименуйте шаг или переменную.

4364

"A transition must be a boolean expression"

Результат условного выражения должен имеет тип BOOL.

4365

"Time Constant expected after qualifier '<name>'"

Откройте диалог 'step attributes' шага '<Name>' и задайте имя временной переменной или константу времени.

4366

"The label of the parallel branch is no valid identifier: '<Name>'"

Введите корректный идентификатор рядом с треугольником перехода (jump).

4367

"The label '<name>' is already used"

Такая метка или одноименный шаг уже определены. Измените идентификатор.

4368

"Action '<name>' is used in multiple step chains, where one is containing the other!"

Действие '<Name>' используется в POU внутри одного или нескольких действий.

4369

"Exactly one network required for a transition"

В условиях перехода задано несколько FBD или LD цепей. Оставьте только одну цепь.

4370

"Additional lines found after correct IL-transition"

Удалите лишние строки в конце IL текста.

4371

"Invalid characters following valid expression: '<name>'"

Удалите лишние символы в конце выражения.

4372

"Step '<name>': Time limit needs type 'TIME'"

Определите пределы времени шага в атрибутах шага через переменные типа TIME или константы времени (например, "t#200ms").

4373

"IEC-actions are only allowed with SFC-POUs"

Существует действие в не-SFC-POU (см. Организатор объектов), которое содержит МЭК действие. Замените это действие на не содержащее МЭК действий.

4374

"Step expected instead of transition '<name>'"

SFC POU повреждено, возможно, вследствие экспорта-импорта.

4375

"Transition expected instead of step '<name>'"

SFC POU повреждено, возможно, вследствие экспорта-импорта.

4376

"Step expected after transition '<name>'"

SFC POU повреждено, возможно, вследствие экспорта-импорта.

4377

"Transition expected after step '<name>'"

SFC POU повреждено, возможно, вследствие экспорта-импорта.

4400

Import / conversion of POU '<name>' contains errors resp. is not complete."

Данный POU нельзя преобразовать в МЭК 61131-3.

4401

"S5 time constant <number> seconds is too big (max. 9990s)."

Ошибка указания времени в BCD формате.

4402

"Direct access only allowed on I/Os."

Убедитесь, что вы используете переменные, определенные как входы или выходы.

4403

"STEP5/7 instruction invalid or not convertible to IEC 61131-3."

Некоторые STEP5/7 команды нельзя преобразовать в МЭК 61131-3, например, команды ЦПУ, такие, как MAS.

4404

"STEP5/7 operand invalid or not convertible to IEC 61131-3."

Некоторые STEP5/7 операнды нельзя преобразовать в МЭК 61131-3.

4405

"Reset of a STEP5/7 timer cannot be converted into IEC 61131-3."

Соответствующий таймер МЭК не имеет входа сброса.

4406

"STEP5/7 Counter constant out of range (max. 999)."

Ошибка указания времени в BCD формате.

4407

"STEP5 instruction not convertible to IEC 61131-3."

Некоторые инструкции STEP5/7 нельзя преобразовать в МЭК 61131-3, например, DUF.

4408

"Bit access of timer or counter words not convertible into IEC 61131-3."

Специальные таймеры/счетчики нельзя преобразовать в МЭК 61131-3.

4409

"Contents of ACCU1 or ACCU2 undefined, not convertible into IEC 61131-3."

Команда не может быть преобразована, поскольку состояние аккумулятора не определено.

4410

"Called POU not in project."

Импортируйте вызываемый POU.

4411

"Error in global variable list."

Проверьте SEQ файл.

4412

"Internal error no.11"

Обратитесь к изготовителю ПЛК.

4413

"Error in format of line in data block"

Ошибочные данные в импортируемом блоке.

4414

"FB/FX name missing."

В исходном S5D файле символьное имя POU пропущено.

4415

"Instruction after block end not allowed."

Защищенные POU не могут быть импортированы.

4416

"Invalid command"

Команда S5/S7 не может быть дизассемблирована

4417

"Comment not closed"

Завершите комментарий "*)".

4418

"FB/FX-Name too long (max. 8 characters)"

Символьное имя POU слишком длинное.

4419

"Expected format of line ""(* Name: <FB/FX-Name> *)"" "

Исправьте строку.

4420

"Name of FB/FX parameter missing"

Проверьте POU.

4421

"Type of FB/FX parameter invalid"

Проверьте POU.

4422

"Type of FB/FX parameter missing"

Проверьте POU.

4423

"Invalid FB/FX call parameter"

Проверьте интерфейс POU.

4424

"Warning: FB/FX for call either missing or parameters invalid or has '0' parameters"

Вызываемый POU еще не импортирован либо определен не верно, либо не имеет параметров (в последнем случае вы можете игнорировать сообщение).

4425

"Definition of label missing"

Метка перехода не определена.

4426

"POU does not have a valid STEP 5 block name, e.g. PB10"

Измените имя POU.

4427

"Timer type not declared"

Добавьте объявление таймера в глобальных переменных..

4428

"Maximum number of open STEP5 brackets exceeded"

Не используйте более семи открывающих скобок.

4429

"Error in name of formal parameter"

Имя параметра не должно превышать четыре символа.

4430

"Type of formal parameter not IEC-convertible"

В МЭК 61131-3 таймеры, счетчики и RPOU не могут служить формальными параметрами.

4431

"Too many 'VAR_OUTPUT' parameters for a call in STEP5 STL"

RPOU не должен иметь формальных выходных параметров.

4432

"Labels within an expression are not allowed"

В МЭК 61131-3 метки перехода можно ставить только в определенных позициях.

4434

"Too many labels"

RPOU должен иметь менее 100 меток.

4435

"After jump / call, a new expression must start"

После перехода или вызова в LD должна следовать команда Load .

4436

"Bit result undefined, not convertible to IEC 61131-3."

Команда, использованная в VKE, не может быть конвертирована, поскольку значение VKE не определено.

4437

"Type of instruction and operand are not compatible"

Битовая команда применена к операнду типа **слово** и т.п.

4438

"No data block opened (insert instruction C DB before)"

Вставьте соответствующую инструкцию.

4500

"Unrecognized variable or address"

Переменная не определена в проекте. Используйте Ассистент ввода <F2> для выбора существующей переменной.

4501

"Extra characters following valid watch expression"

Удалите лишние символы.

4520

"Error in Pragma: Flag expected before '<Name>'"

Ошибочная инструкция компилятора. Проверьте допустимость флага '<Name>'

4521

"Error in Pragma: Unexpected element '<Name>'"

Проверьте, правильно ли составлена инструкция компилятора.

4522

"flag off' pragma expected!"

Инструкция компилятора задана не полностью, добавьте флаг.

4523

"Pragma {<Pragmaname>} not allowed in interface of type '<Name>'"

Данная инструкция здесь не применима. Уточните правила определения инструкций компилятора по документации.

4550

"Index out of defined range : Variable OD "number>, Line <line number>."

Убедитесь что индекс лежит в заданном диапазоне (См. Target settings/networkfunctionality).

4551

"Subindex out of defined range : Variable OD "number>, Line <line number>."

Убедитесь, что подиндекс лежит в заданном диапазоне (См. Target settings /networkfunctionality).

4552

"Index out of defined range : Parameter OD "number", Line <line number>."

Убедитесь, что индекс лежит в заданном диапазоне (См. Target settings/networkfunctionality).

4553

"Subindex out of defined range : Parameter OD "number", Line <line number>."

Убедитесь, что подиндекс лежит в заданном диапазоне (См. Target settings/networkfunctionality).

4554

"Variablename invalid: Variable OD <number>, Line <line number>."

Введите имя переменной проекта в поле 'variable'. Используйте синтаксис <POU name>.<variable name> или .<variable name> для глобальных переменных.

4555

"Empty table-entry, input not optional: Parameter OD <number>, Line <line number>"

Ввод значения в этом поле обязателен.

4556

"Empty table-entry, input not optional: Variable OD <number>, Line <number>"

Ввод значения в этом поле обязателен.

4557

"The required parameter memory is too large"

Исчерпан максимальный размер данных, загруженных через список параметров (Parameters). Данный размер определен в установках целевой платформы. Измените размер списка параметров.

4558

"The required variable memory is too large"

Исчерпан максимальный размер данных, загруженных через список параметров (Variables). Данный размер определен в установках целевой платформы. Измените размер списка параметров.

4560

"Invalid value: Dictionary '<Name>', column '<Name>', line '<line number>'"

Проверьте правильность ввода. Она определяется допустимыми атрибутами данного поля, заданными в платформенно-зависимом XML файле Менеджера параметров или стандартными установками, или такой файл отсутствует.

4561

"Column not defined: '<Name>'"

Ввод в колонке списка параметров ссылается на другую колонку, которая еще не определена. Описания колонок содержатся в XML файле Менеджера параметров. Если XML файл отсутствует, используются стандартные установки.

4562

"Index/subindex used already: Dictionary '<Name>', line '<Line Number>'"

Комбинация Индекс/Подиндекс должна быть уникальной для всех списков параметров. Исправьте индекс.

4563

"Identifier '<Name>' used already: Dictionary '<Name>', line '<Line Number>'"

Имя должно быть уникальным для всех списков параметров, поскольку оно используется для доступа к параметру.

4564

"Index '<Name>' is out of range: Dictionary '<Name>', line '<Line Number>' "

Задайте индекс в диапазоне, определенном в установках целевой системы для соответствующего типа списка (Variables, Parameters, Mappings).

4565

"Subindex '<Name>' is out of range: Dictionary '<Name>', line '<Line Number>' "

Задайте подиндекс в диапазоне, определенном в установках целевой системы 'SubIndex range'.

4566

"An error occurred during import of the parameter manager"

Вы используете файл экспорта, содержащий ошибки в информации Менеджера Параметров. Проверьте *.ехр-файл.

4600

"Networkvariables: '<name>' expression is not from type bool!"

Убедитесь, что переменная, указанная в поле 'Transmit on event' диалога свойств сетевых переменных, имеет тип BOOL.

4601

"Network variables '<name>': No cyclic or freewheeling task for network variable exchange found"

В проекте нет циклической или free-wheeling задачи, где бы использовались сетевые CAN или UDP переменные, включая PLC_PRG (просто объявления переменных недостаточно!). Вы должны обеспечить использование этих переменных в соответствующей задаче или в PLC_PRG. Если вы хотите использовать их в нескольких задачах, обратите внимание, что сетевой обмен данными будет соотнесен с задачей с наивысшим приоритетом.

4602

""<name of network variables list>': The object uses UDP port '<port number>' instead of '<port number>'"

В настройках данного списка сетевых переменных указан порт, отличный от указанного для списка, найденного первым в папке глобальных переменных. Проверьте правильность указания порта!

4620

В проекте обнаружены неиспользуемые переменные. См. описание команды 'Project' 'Check' (Unused Variables).

4621

Обнаружено пересечение памяти переменных в „АТ“-объявлениях. См. описание команды 'Project' 'Check' ('Overlapping memory areas').

4622

МЭК адреса, использующие одну и ту же область памяти, используются в разных задачах. См. описание команды 'Project' 'Check' ('Concurrent Access').

4623

В проекте присутствует обращение на запись к одной и той же области памяти в нескольких местах. См. описание команды 'Project' 'Check' ('Multiple writes to output').

4650

"AxisGroup '<Name>': Task '<Name>' does not exist."

В PLC Configuration для группы осей (диалог 'Module parameters', колонка 'Value') указано имя задачи, управляющей передачей данных, отсутствующее в Конфигурации задач.

4651

"AxisGroup '<Name>': Cychetime (dwCycle) not set."

Задайте в диалоге 'Module parameters' для группы осей значение времени цикла (dwCycle).

4670

"CNC program '<Name>': Global variable '<Name>' not found."

В CNC программе используется глобальная переменная (например, \$glob_var\$), не определенная в проекте. Добавьте соответствующее определение либо используйте существующую переменную.

4671

"CNC program '<Name>': Variable '<Name>' has an incompatible type."

Инструкция присваивания в CNC программе использует переменную не допустимого типа. Используйте другую переменную или измените тип в объявлении.

4685

"CAM '<Name>': CAM table type unknown."

Проверьте тип данных, указанный в диалоге "Compile options.." САМ редактора для эквидистанции либо элемент в таблице оптимизированных точек

4686

"CAM '<Name>': CAM point exceeds datatype range."

В САМ указана точка, выходящая за определенный диапазон в САМ редакторе. См. диалог 'Compile options..' в САМ-редакторе..

4700

""<Number>' (<Name>): Watch expression '<Name>' is not a numeric variable."

В конфигурации визуализации использована переменная, не являющаяся числом, как это требуется в данном месте (например, значение XOffset или Angle и т.д.).

4701

""<Name>' (<Number>): Watch expression '<name>' is not of type BOOL."

В конфигурации визуализации используется переменная отличного от BOOL типа, как это требуется в данном месте

4702

""<Name>' (<Number>): Watch expression '<name>' is not of type STRING."

В конфигурации визуализации используется переменная отличного от STRING типа, как это требуется в данном месте

4703

""<Name>' (<Number>): Invalid watch expression '<Name>'"

Визуализация включает ошибочную переменную.

4704

""<Name>' (<Number>): Invalid initial value in watchlist '<Name>'."

В списке переменных, используемых в визуализации (команда INTERN, категория Input), задано ошибочное начальное значение. Проверьте используемый список.

4705

""<name>' (<number>): No valid alarm group assigned to alarm table."

Введите действительную группу тревоги в таблице диалога тревог (категория Alarm table).

4900

"Invalid type for conversion"

Вы используете преобразование текстов, не подержанное в выбранном генераторе кода.

4901

"Internal error: Overflow in array access!"

Размерность массива слишком велика для 32-разрядных индексов. Уменьшите размерность массива.

5100

"<Name> (<Zahl>): Expression too complex. No more registers available"

Данное выражение слишком сложное для обработки посредством доступных регистров. Уменьшите вложенность выражения, используйте промежуточные переменные.

