Система ввода-вывода Fastwel I/O
Контроллеры CPM701/CPM702/CPM703/CPM704

Руководство программиста

ИМЕС.00300-02 33 02-1

Версия 2.0

СОДЕРЖАНИЕ

Торговые марки

ДОЛОМАНТ™, ФАСТВЕЛ™, Fastwel™ – официально зарегистрированные торговые марки ЗАО «НАУЧНО-ПРОИЗВОДСТВЕННАЯ ФИРМА «ДОЛОМАНТ», Москва, Российская Федерация.

Кроме того, настоящий документ может содержать наименования, фирменные логотипы и торговые марки, являющиеся зарегистрированными торговыми марками, а следовательно, права собственности на них принадлежат их законным владельцам.

Права собственности

Настоящий документ содержит информацию, которая является собственностью ЗАО «НАУЧНО-ПРОИЗВОДСТВЕННАЯ ФИРМА «ДОЛОМАНТ». Он не может быть скопирован или передан с использованием известных средств, а также не может храниться в системах хранения и поиска информации без предварительного письменного согласия ЗАО «НАУЧНО-ПРОИЗВОДСТВЕННАЯ ФИРМА «ДОЛОМАНТ» или одного из ее уполномоченных агентов. Информация, содержащаяся в настоящем документе, насколько нам известно, не содержит ошибок, однако, ЗАО «НАУЧНО-ПРОИЗВОДСТВЕННАЯ ФИРМА «ДОЛОМАНТ» не может принять на себя ответственность за какие-либо неточности и их последствия, а также ответственность, возникающую в результате использования или применения любой схемы, продукта или примера, приведенного в настоящем документе. ЗАО «НАУЧНО-ПРОИЗВОДСТВЕННАЯ ФИРМА «ДОЛОМАНТ» оставляет за собой право изменять и усовершенствовать как настоящий документ, так и представленный в нем продукт по своему усмотрению без дополнительно извещения.

Контактная информация

Изготовитель – ЗАО «НАУЧНО-ПРОИЗВОДСТВЕННАЯ ФИРМА «ДОЛОМАНТ»:

Почтовый адрес: Россия, 117342, Москва, ул. Введенского, д.3

Телефон: +7 (495) 232-2033

Факс: +7 (495) 232-1654

Электронная почта: info@dolomant.ru

Web: http://www.dolomant.ru

Служба технической поддержки:

Телефон: +7 (495) 232-1698

Электронная почта: support@fastwel.ru

Эксклюзивный дистрибьютор компания «Прософт»

Электронная почта: info@prosoft.ru

Web: http://www.prosoft.ru/

Телефон: +7 (495) 234-0636

Факс: +7 (495) 234-0640

Авторское право

1. ВВЕДЕНИЕ

Настоящий документ содержит сведения о принципах функционирования, а также указания по настройке и программированию контроллеров CPM701, CPM702, CPM703 и CPM704 серии Fastwel I/O в среде CoDeSys фирмы 3S Smart Software Solutions.

Информация о принципах функционирования и о конфигурировании сетевых интерфейсов контроллеров приведена в соответствующих документах:

1. ИМЕС.0300-02 33 02-2. CPM701. Контроллер узла сети CANopen. Руководство по конфигурированию и программированию сетевых средств.
2. ИМЕС.0300-02 33 02-3. CPM702. Контроллер узла сети MODBUS RTU/ASCII. Руководство по конфигурированию и программированию сетевых средств.
3. ИМЕС.0300-02 33 02-4. CPM703. Контроллер узла сети MODBUS TCP. Руководство по конфигурированию и программированию сетевых средств
4. ИМЕС.0300-02 33 02-5. CPM704. Контроллер узла сети PROFIBUS DP-V1. Руководство по конфигурированию и программированию сетевых средств

Информация о конфигурировании модулей ввода-вывода Fastwel I/O приведена в документе *ИМЕС.00300-02 33 01. Модули ввода-вывода Fastwel I/O. Руководство программиста*.

При работе с настоящим документом следует также пользоваться следующими документами:

1. ФАПИ.421459.700 РЭ. FASTWEL-I/O. Распределённая система ввода–вывода. Руководство по эксплуатации
2. *User Manual for PLC Programming with CoDeSys 2.3*

Предполагается, что пользователь среды CoDeSys, адаптированной для программирования контроллеров на базе контроллеров серии Fastwel I/O, должен иметь навыки программирования на языках стандарта IEC 61131-3 и быть знакомым с операционной системой Windows на уровне, достаточном для квалифицированного использования.

2. ОБЩИЕ СВЕДЕНИЯ

2.1. Назначение Fastwel I/O

Fastwel I/O является аппаратно-программным комплексом, предназначенным для создания автоматизированных систем сбора данных и управления.

Аппаратно-программные средства Fastwel I/O могут использоваться для построения как автономных программируемых контроллеров, так и распределенных систем сбора данных и управления.

2.2. Структура аппаратных средств Fastwel I/O

В комплекс Fastwel I/O входят следующие аппаратные средства:

- Контроллеры узла сети (далее – контроллеры)
- Модули ввода-вывода
- Вспомогательные модули

Контроллер является вычислительным устройством на базе микропроцессора R1610C фирмы RDC, совместимого с 80C186 и имеющего тактовую частоту 100 МГц.

Контроллер имеет интерфейс с модулями ввода-вывода, далее называемый внутренней шиной FBUS, а также интерфейс внешней сети.

Интерфейс внешней сети контроллера предназначен для обмена данными с рабочими станциями и автоматизированными рабочими местами верхнего уровня автоматизированных систем сбора данных и управления.

Модули ввода-вывода, подключаемые к внутренней шине контроллера, предназначены для организации связи контроллера с датчиками и исполнительными механизмами объекта управления.

2.3. Структура программного обеспечения Fastwel I/O

2.3.1. Состав программного обеспечения Fastwel I/O

В комплекс Fastwel I/O входит следующее системное и инструментальное программное обеспечение:

- Пакет адаптации среды разработки прикладных программ на языках стандарта IEC 61131-3 CoDeSys (далее – пакет адаптации IDE CoDeSys);
- Адаптированная среда исполнения прикладных программ, разрабатываемых в среде CoDeSys, (далее – среда исполнения CoDeSys), поставляемая в каждом контроллере;
- Демонстрационные версии OPC-серверов для сетей CAN, Modbus RTU/ASCII и Modbus TCP.

Структура программного обеспечения Fastwel I/O показана на рис. 1.

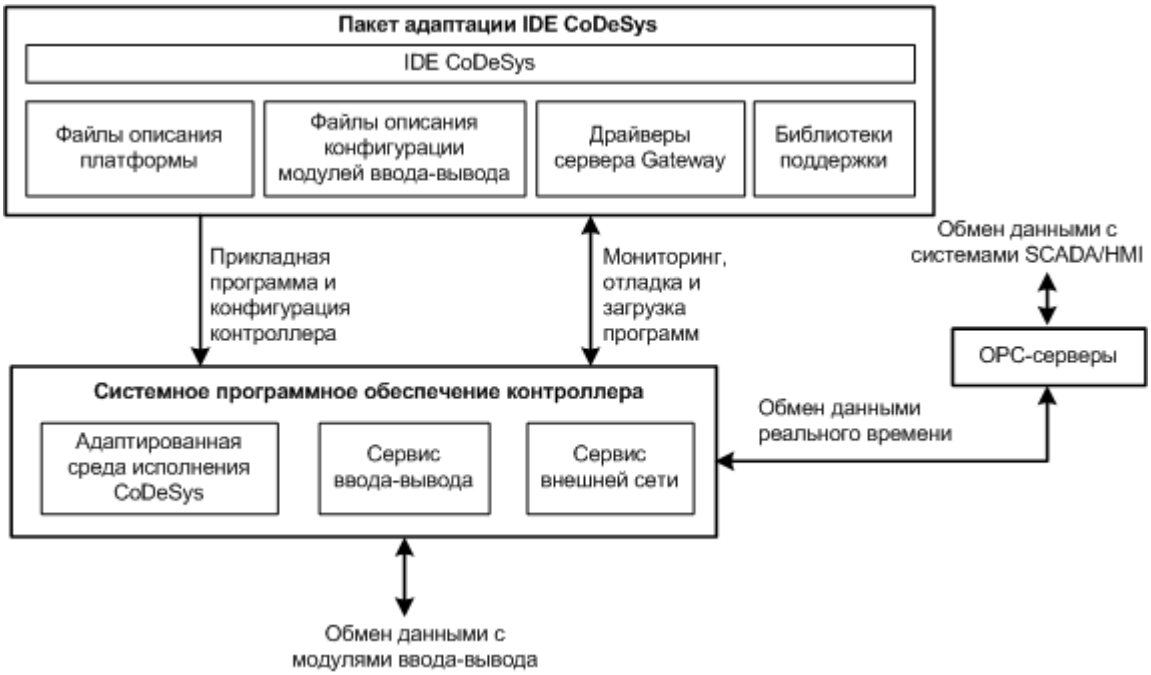


Рис. 1. Структура программного обеспечения Fastwel I/O

2.3.2. Пакет адаптации IDE CoDeSys

Пакет адаптации IDE CoDeSys поставляется в едином установочном комплекте в следующем составе:

1. Интегрированная среда разработки IDE CoDeSys фирмы 3S Smart Software Solutions
2. Файлы описания платформы Fastwel I/O, интегрируемые с IDE CoDeSys и позволяющие генерировать исполняемый код прикладных программ для контроллеров Fastwel I/O средствами IDE CoDeSys
3. Файлы описания конфигурации модулей ввода-вывода, интегрируемые с IDE CoDeSys и позволяющие генерировать конфигурационную информацию для контроллеров Fastwel I/O средствами IDE CoDeSys
4. Драйверы коммуникационного сервера CoDeSys Gateway Server, интегрируемые с CoDeSys Gateway Server и позволяющие выполнять загрузку прикладных программ в контроллер, удаленную отладку и мониторинг переменных
5. Библиотеки поддержки платформы Fastwel I/O, содержащие функциональные блоки и функции, обеспечивающие доступ к специфическим функциональным возможностям платформы Fastwel I/O из приложений пользователя, разрабатываемых в среде CoDeSys.

CoDeSys является интегрированной средой разработки прикладного программного обеспечения для автоматизированных систем сбора данных и управления на языках стандарта IEC 61131-3. CoDeSys обеспечивает выполнение следующих функций:

1. Создание конфигурации контроллера, которая включает в себя перечень описаний модулей ввода-вывода, входящих в его состав, параметры каждого модуля, параметры протокола внешней сети и перечень описаний сообщений, поступающих из внешней сети и выдаваемых в сеть, и параметры исполнения прикладной программы в контроллере.
2. Описание информационных связей между разрабатываемой прикладной программой и сообщениями, передаваемыми во внешнюю сеть и получаемыми по внешней сети, а также между прикладной программой и каналами модулей ввода-вывода.
3. Реализацию прикладного алгоритма обработки данных и управления на языках ST, IL, LD, FBD, SFC стандарта IEC 61131-3 и трансляцию разработанной программы в исполняемый код процессора
4. Отладку разработанной прикладной программы в режиме эмуляции
5. Загрузку прикладной программы в контроллер
6. Удаленную отладку и управление исполнением прикладной программы в контроллере.

2.3.3. Адаптированная среда исполнения CoDeSys

Адаптированная среда исполнения CoDeSys является одним из сервисов системного программного обеспечения контроллеров Fastwel I/O. Системное программное обеспечение контроллеров Fastwel I/O выполняет следующие функции:

1. Исполнение приложения пользователя, разработанного в среде CoDeSys, с возможностью просмотра и изменения значений переменных и удаленной загрузки измененной версии
2. Обмен данными между приложением пользователя и модулями ввода-вывода
3. Прием данных по сети и передачу их приложению
4. Передачу по сети данных приложения
5. Светодиодную индикацию режима работы среды исполнения
6. Диагностику функционирования основных подсистем среды исполнения и предоставление диагностической информации приложению пользователя
7. Управление режимами работы контроллера, обработку ошибок и нештатных ситуаций.

Приложение, разрабатываемое пользователем в среде CoDeSys, должно состоять хотя бы из одной программы, и может содержать до 3-х циклических и до 16-ти ациклических задач, а также функций обработки системных событий.

Циклической задачей далее называется множество программ (в терминах IEC 61131-3), запускаемых на исполнение с заданным периодом под управлением отдельного потока исполнения операционной системы контроллера. Помимо периода запуска, каждая циклическая задача имеет приоритет, согласно которому планировщик операционной системы выбирает, какому потоку операционной системы выделить процессорное время в тот или иной момент времени.

Ациклической задачей далее называется множество программ (в терминах IEC 61131-3), запускаемых на исполнение на контексте высокоприоритетного потока исполнения операционной системы в момент перехода некоторой булевой переменной (источника события), определенной в приложении пользователя, из состояния FALSE в состояние TRUE. Помимо переменной-источника события, каждая ациклическая задача имеет приоритет и порядковый номер, согласно которым среда исполнения выбирает очередность запуска ациклических задач.

Обработчиком системного события далее называется функция (в терминах IEC 61131-3), вызываемая средой исполнения при возникновении некоторого системного события. К системным событиям относятся моменты запуска и останова приложения, окончание подготовки приложения к запуску, начало загрузки нового приложения, момент перед заменой текущего приложения на вновь загруженное и другие.

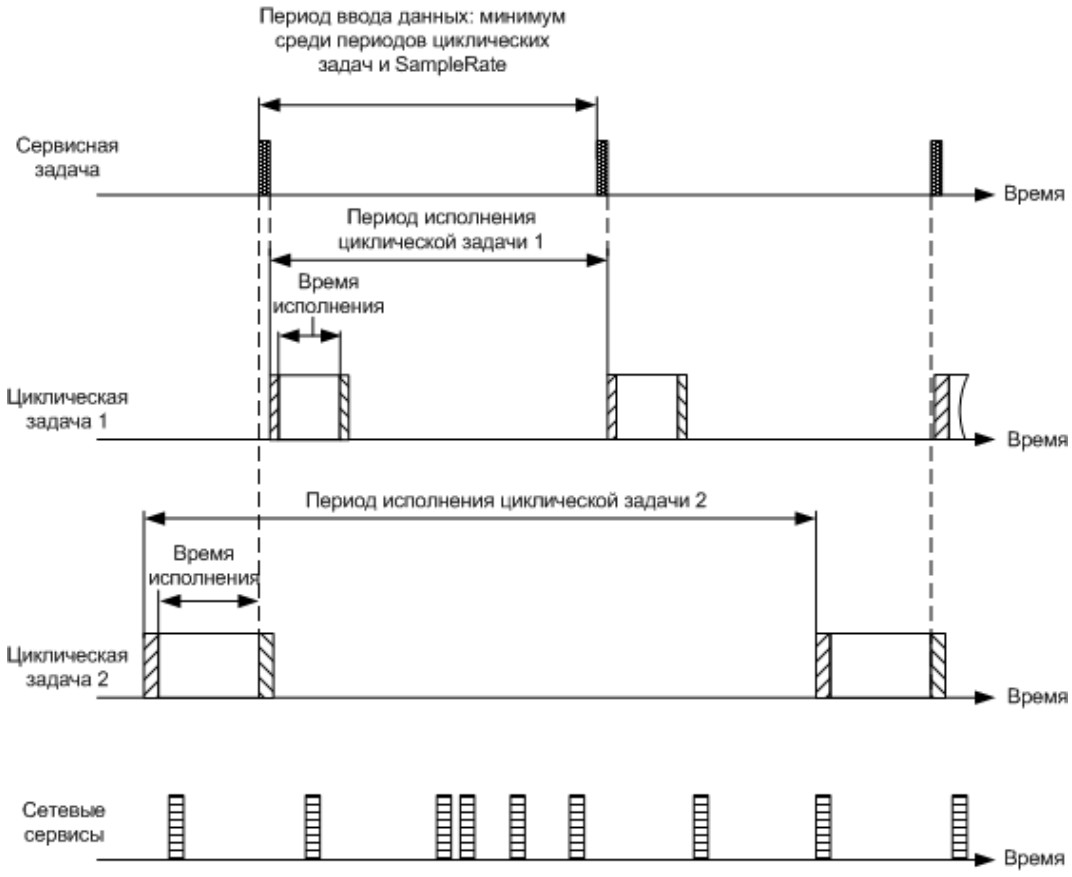
Данные *окружения* (модулей ввода-вывода, подключаемые к внутренней шине контроллера, и внешней сети), представляются так называемым *образом процесса*, состоящим из двух областей памяти с непересекающимися адресами, через которые происходит взаимодействие между циклическими и ациклическими задачами приложения и окружением.

Первая область образа процесса, называемая *областью входных данных*, предназначена для буферизации значений входных данных приложения в процессе приема информации от устройств ввода-вывода и сетевых интерфейсов..

Вторая область называется *областью выходных данных* и предназначена для буферизации значений выходных данных приложения в процессе выдачи информации устройствам ввода-вывода и в сетевые интерфейсы.

Циклограмма исполнения приложения, состоящего из двух разночастотных циклических задач, показана на рис. 2.

Исполнение одной ациклической задачи иллюстрируется рис. 3.



Исполнение программ, ассоциированных с циклической задачей

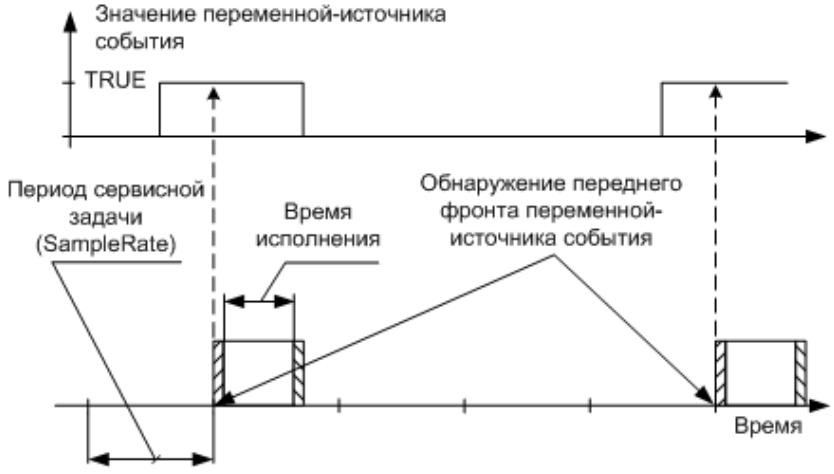
Выполнение операций, связанных с информационным обменом по внешней сети

Ввод данных входных каналов модулей ввода-вывода и входящих коммуникационных объектов

Ввод других задач через механизм межзадачного обмена с использованием FastwelTasksExchange.lib

Вывод данных данных задачи сервисам ввода-вывода, внешней сети и другим задачам

Рис. 2. Исполнение циклических задач



Исполнение программ, ассоциированных с ациклической задачей

Ввод данных входных каналов модулей ввода-вывода и входящих коммуникационных объектов в область входных данных задачи

Передача выходных данных задачи сервисам ввода-вывода, внешней сети и другим задачам

Рис. 3. Исполнение ациклической задачи

2.4. Основные характеристики контроллеров

2.4.1. Характеристики подсистемы исполнения прикладной программы пользователя

Параметр	Единица	Минимум	Номинально	Максимум
Области памяти				
Размер области входных переменных приложения	байт		8192	
Размер области выходных переменных приложения	байт		8192	
Размер области внутренних переменных приложения	байт		32768	
Размер области исполняемого кода приложения	байт			65300
Размер области конфигурации прикладной программы	байт			65300
Размер области энергонезависимых переменных	Не поддерживаются[1]			
Производительность				
Сложение и вычитание 2-байтовых операндов	опер/с	2083333	2430370	2777778
Умножение 2-байтовых операндов	опер/с	1010000	1122222	1234444
Деление 2-байтовых операндов	опер/с	793333	847037	900741
Сложение и вычитание целочисленных 4-байтовых операндов	опер/с	842593	895185	947778
Умножение целочисленных 4-байтовых операндов	опер/с	168519	174519	180556
Деление целочисленных 4-байтовых операндов	опер/с	33467	34241	35011
Сложение и вычитание операндов типа REAL	опер/с	24933	25763	26593
Умножение операндов типа REAL	опер/с	20589	21148	21707
Деление операндов типа REAL	опер/с	18996	19659	20326
Сложение и вычитание операндов типа LREAL	опер/с	23437	24219	25000
Умножение операндов типа LREAL	опер/с	19352	19881	20407
Деление операндов типа LREAL	опер/с	17856	18481	19107
Ядро системы исполнения				
Количество циклических задач	шт	0[2]	1	3
Период циклической задачи	мс	1	10	1000
Количество уровней приоритета циклических задач				3
Размер стека циклической задачи	байт		2800	
Количество ациклических задач	ед	0		16
Количество уровней приоритета ациклических задач	ед			3
Размер стека ациклической задачи	байт		2000	
Количество единиц организации программы (POU)	шт	1		1024
Количество связей задачи с областью входных данных[3]	шт	0		512
Количество связей задачи с областью выходных данных	шт	0		512
Размер переменной типа STRING	байт		80	255
Количество одновременно используемых временных переменных типа STRING при работе со строковыми операциями из библиотеки STANDARD.LIB	шт			16
Количество одновременно используемых временных переменных типа STRING при выполнении преобразований переменных примитивных типов в строки	шт			16
Отладчик				
Количество устанавливаемых постоянных точек останова	шт			10
Количество промежуточных временных точек останова	шт			100
Глубина дерева вызовов при отладке				30
Количество позиций просмотра потока данных (flow positions)	шт			1000
Взаимодействие со средой разработки				
Размер буфера форсируемых переменных	байт			1024
Размер буфера считываемых переменных	байт			4096
Размер буфера трассировки	байт			4096
Таймаут между блоками транспортного протокола обмена со средой разработки	мс			5000

[1] Механизм горячего обновления приложения пользователя во время работы реализован без использования переменных VAR PERSISTENT. Энергонезависимые переменные могут быть реализованы в приложении путем использования функций библиотеки FastwelSysLibFile.lib

[2] Если пользователь не добавил в конфигурацию задач проекта ни одной задачи любого типа, CoDeSys автоматически сгенерирует одну циклическую задачу с именем *DefaultTask*, которая будет исполняться с периодом, задаваемым параметром CPM70x...Controller:SampleRate ресурса PLC Configuration, и исполнять программу с именем PLC_PRG (режим совместимости с предыдущей версии адаптации CoDeSys для Fastwel I/O)

[3] Связью задачи с областью памяти называется описатель некоторого участка данной области, содержащий информацию о смещении и длине участка внутри области в битах, из которого задача будет вводить или выводить данные. Связь создается средой разработки для каждой непосредственно представляемой переменной, содержащей ссылку на область входных или выходных данных.

2.4.2. Характеристики сервиса ввода-вывода

Параметр	Единица	Минимум	Номинально	Максимум
Количество модулей ввода-вывода	шт	0		64
Размер области ввода данных	байт	0		2300
Размер области вывода данных	байт	0		2300
Размер поля контрольной суммы сообщения	байт		4	
Размер кадра	бит		10	
Скорость обмена	Мбит/с		2	
Общий размер передаваемой служебной информации, включая контрольную сумму	байт		5	
Пауза между обработкой предыдущего и передачей текущего запроса:				
в режиме *Single Group* (одна группа на все модули)	мкс			330
в режиме *Group per Module* (одна группа на каждый модуль)	мкс	180	220	330
Период обмена данными	мс	1	10	1000
Пропускная способность обмена данными в режиме "1 группа на все модули"	кбайт/с		165	

2.5. Состав поставляемого программного обеспечения

2.5.1. Перечень программного обеспечения на компакт-диске Fastwel I/O Product CD

Интерактивный компакт-диск Fastwel DVD имеет раздел **FASTWEL I/O**, с каталогами **Контроллеры и модули** и **Программное обеспечение**.

Каталог **Контроллеры и модули** содержит ссылки на подкаталоги с эксплуатационной документацией на компоненты аппаратно-программного комплекса Fastwel I/O.

Каталог **Программное обеспечение** содержит подкаталог **CODESYS 2.3**, в котором расположена программа установки пакета адаптации CoDeSys 2.3 для Fastwel I/O.

2.5.2. Содержимое установочного комплекта адаптированного пакета программ CoDeSys

Установочный комплект пакета программ CoDeSys фирмы 3S Smart Software Solution, адаптированного для работы с комплексом Fastwel I/O содержит оригинальный установочный комплект пакета CoDeSys и файлы, необходимые для адаптации оригинального пакета, включая:

1. Эксплуатационную документацию согласно п. 2.5.1
2. Файлы описания вычислительной платформы *Fastwel I/O System with Multitasking Runtime*.
3. Файлы описаний конфигурации аппаратных средств комплекса Fastwel I/O в формате PLC Configuration фирмы 3S Smart Software Solutions (cfg)
4. Файлы modbusDLL.dll и GDrvFastwel.dll, представляющие драйвер коммуникационного сервера CoDeSys Gateway Server, который обеспечивает возможность удаленной загрузки и отладки прикладной программы на контроллере из среды CoDeSys
5. Каталог примеров проектов приложений CoDeSys
6. Каталог библиотек поддержки комплекса Fastwel I/O для CoDeSys.

ВНИМАНИЕ!

Установка файлов поддержки вычислительной платформы Fastwel I/O для CoDeSys выполняется автоматически программой установки адаптированного пакета программ CoDeSys. Попытки самостоятельной установки поддержки платформы Fastwel I/O при помощи программы InstallTarget могут привести к некорректной работе адаптированной среды разработки CoDeSys.

2.6. Системные требования к рабочему месту разработчика

2.6.1. Требования к аппаратным средствам

Персональный компьютер, на котором предполагается использовать адаптированную среду разработки CoDeSys, должен иметь аппаратную конфигурацию не хуже:

– процессор Intel Celeron 466 МГц;

- объем установленной оперативной памяти не менее 128 Мбайт;
- размер свободного дискового пространства не менее 100 Мбайт;
- привод CD-ROM

Рекомендуемое разрешение монитора 1280×1024.

Для удаленной загрузки и отладки программного обеспечения в контроллеры через порт консоли требуется кабель соединительный ACS00019, а компьютер должен иметь, как минимум, один коммуникационный порт интерфейса RS-232C.

Для удаленной загрузки и отладки программного обеспечения в контроллер узла сети CPM701 по сети CAN компьютер должен быть оснащен адаптером сети CAN фирмы IXXAT (любым) либо адаптером PCAN-USB фирмы PEAK-Systems Technik.

Для удаленной отладки и загрузки программного обеспечения в контроллер узла сети CPM702 по сети MODBUS RTU или ASCII компьютер должен иметь в своем составе последовательный порт интерфейса RS-232C или RS-485.

Для удаленной отладки и загрузки программного обеспечения в контроллер узла сети CPM703 по сети MODBUS TCP компьютер должен быть оснащен адаптером сети Ethernet 100 Мбит/с.

Для удаленной отладки и загрузки программного обеспечения в контроллер узла сети CPM704 по сети PROFIBUS DP к компьютеру должен быть подключен адаптер Molex DRL-PFB-USB и установлено сервисное программное обеспечение BradCommunications PC Network Interfaces 4.1.

2.6.2. Требования к системному программному обеспечению

Персональный компьютер, на котором предполагается использовать адаптированную среду разработки CoDeSys, должен иметь конфигурацию программных средств не хуже:

- операционная система Windows 2000 Professional SP4, Windows XP SP3, Windows 7 не хуже Home Premium или Windows 8/8.1;
- для чтения документации в формате Adobe PDF требуется установить программу Adobe Acrobat Reader версии не ниже 6.0.

3. УСТАНОВКА И НАСТРОЙКА АДАПТИРОВАННОЙ СРЕДЫ CODESYS

3.1. Предварительные замечания

Перед началом установки убедитесь, что аппаратно-программная конфигурация компьютера, на который предполагается установить адаптированную версию CoDeSys, соответствует требованиями, приведенным в п. 2.6.

Если на компьютере уже установлен пакет CoDeSys или какие-либо его компоненты, убедитесь, что версия среды разработки не ниже 2.3.6.2, для чего запустите CoDeSys и выберите команду **About** меню **Help**.

В случае, если компьютер содержит установленный ранее пакет CoDeSys версии ниже 2.3.6.2, программа установки пакета CoDeSys, входящего в установочный комплект адаптации для Fastwel I/O, выполнит обновление автоматически.

3.2. Установка

Для установки адаптированной версии CoDeSys выполните следующие действия:

1. Запустите программу FastwelCoDeSysAdaptation.exe. Через некоторое время на экран монитора будет выведена диалоговая панель **Welcome**, в которой следует нажать кнопку **Next**
2. Если на компьютере не установлен CoDeSys версии 2.3.9.46 или выше, на экран монитора будет выведено сообщение "*Требуется установить CoDeSys 2.3.9.46 или выше. Вы хотите сделать это сейчас*?". Нажмите кнопку **Yes**
3. Если ранее осуществлялись попытки установки адаптированной среды CoDeSys, на экран монитора может быть выведена диалоговая панель с сообщением *Overwrite Protection*. Если данная диалоговая панель появилась, щелкните в ней на кнопке **Yes to All**, и установка будет продолжена
4. На экран монитора будет выведена диалоговая панель **Выбор языка**, в списке доступных языков которой выберите **Английский** и нажмите **OK**. Через некоторое время на экран монитора будет выведена диалоговая панель с сообщением *Please close all running applications before continuing installation* (Пожалуйста, завершите работу всех запущенным приложений перед продолжением установки)
5. Если в текущий момент запущены какие-либо приложения или компоненты пакета CoDeSys, следует завершить их работу. Остальные приложения завершать не обязательно. Нажмите кнопку **OK** в диалоговой панели. На экран монитора будет выведена диалоговая панель **InstallShield Wizard** с приветствием от программы установки пакета CoDeSys
6. Нажмите кнопку **Next**, после чего выберите каталог установки пакета CoDeSys в диалоговой панели **Choose Destination Location** и нажмите кнопку **Next.**
7. Ничего не трогая в появившейся диалоговой панели **InstallShield Wizard: Select Components**, нажмите кнопку **Next**, после чего в каждой из последующих двух диалоговых панелях вновь нажмите **Next**
8. Программа установки выполнит установку пакета CoDeSys
9. Незадолго до окончания установки на экран монитора будет выведено напоминание о том, что некоторые продукты, входящие в пакет CoDeSys, требуют наличия оплаченной лицензии. Нажмите **OK** в диалоговой панели данного сообщения, а затем **Finish**. После нажатия кнопки **Finish** пройдет от 5 до 10 с, в течение которых не следует предпринимать каких-либо действий
10. По истечении 5–10 с на экране появится диалоговая панель **Choose Destination Location**, в которой имеется возможность указать каталог установки файлов адаптации среды CoDeSys и документация. Нажмите **Next** в данной диалоговой панели, а затем **Finish**. Установка адаптированной среды CoDeSys на этом завершена.

Примечание. Компоненты, требующие платной лицензии, в установку адаптированной среды CoDeSys не входят.

3.3. Проверка установки

По завершении установки убедитесь, что установка выполнена успешно, для чего выполните следующие действия:

1. В группе *Programs\3S Software\CoDeSys V2.3* щелкните на пиктограмме *CoDeSys V2.3*. На экране монитора появится главное окно IDE CoDeSys, показанное на рис. 4.

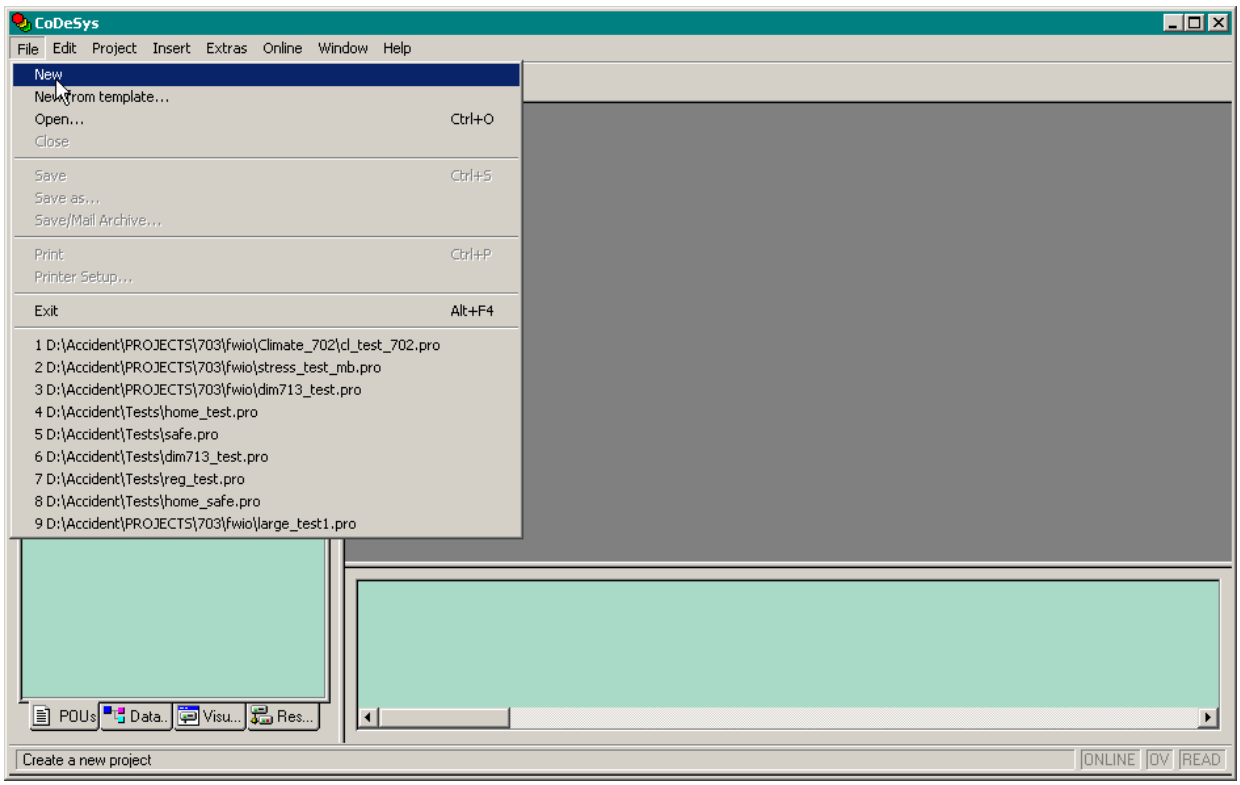


Рис. 4. Главное окно среды разработки CoDeSys

2. Выберите команду **New** в меню **File**. В появившейся диалоговой панели **Target Settings** (Параметры платформы) в выпадающем списке **Configuration** выберите опцию *Fastwel I/O System with Multitasking Runtime*. Внешний вид диалоговой панели **Target Settings** примет вид, показанный на рис. 5.

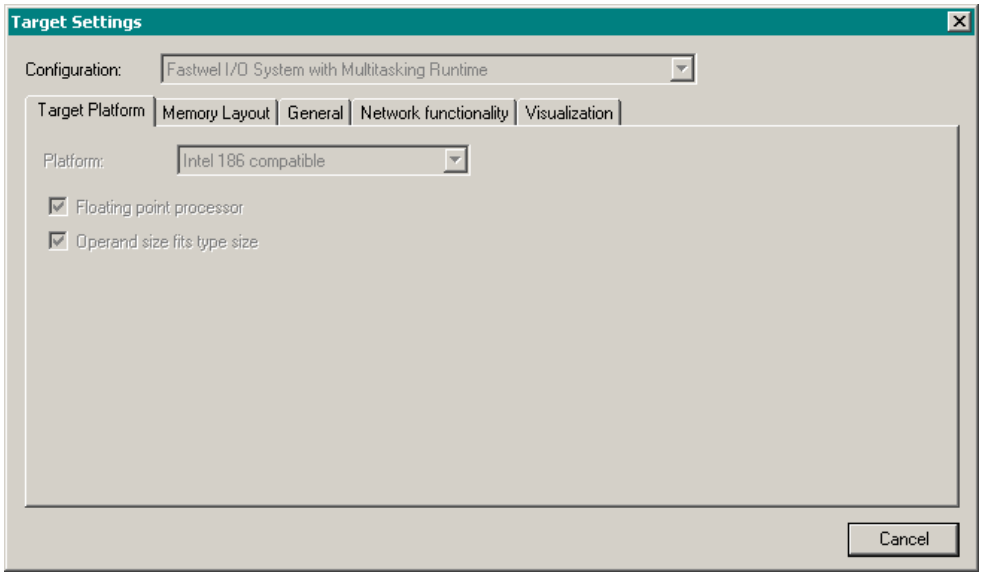


Рис. 5. Диалоговая панель Target Settings после выбора платформы Fastwel I/O System

3. Нажмите кнопку **OK** диалоговой панели. На экране монитора появится диалоговая панель **New POU** с именем первой программы, которая будет добавлена в проект.

Нажмите кнопку **OK** в диалоговой панели, после чего выберите вкладку **Resources** в левой области главного окна IDE CoDeSys и дважды щелкните на узле **PLC Configuration** в дереве **Resources**. В левой области главного окна IDE CoDeSys появится окно настройки конфигурации контроллера узла сети, показанное на рис. 6. Раскройте узел дерева *Fastwel I/O System Configuration*, щелкните правой кнопкой мыши на названии контроллера в дереве, выберите заголовок контекстного меню **Replace Element** и в появившемся подменю выберите название контроллера, для которого предполагается создать проект, как показано на рис. 6.

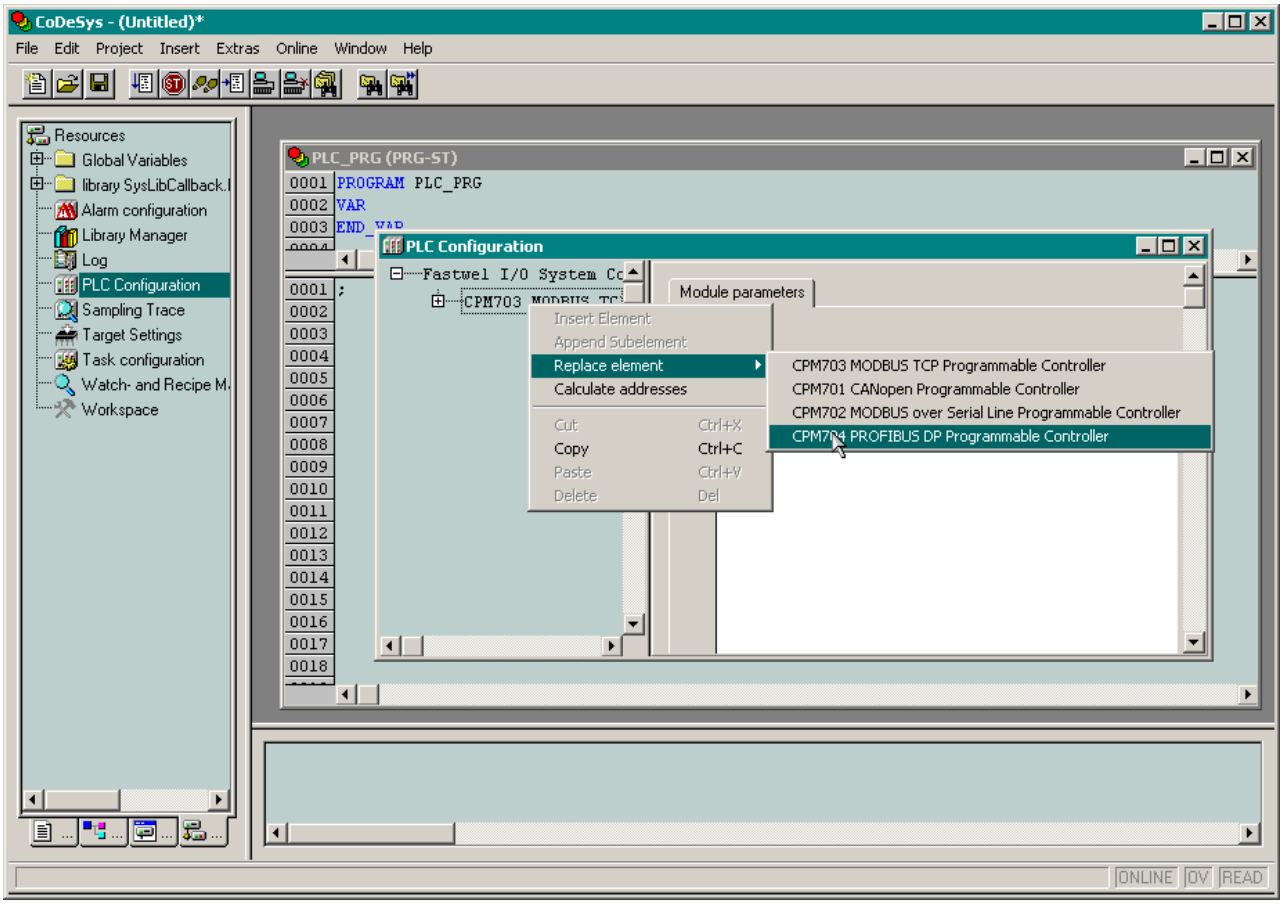


Рис. 6. Окно PLC Configuration при создании проекта для платформы Fastwel I/O System with Multitasking Runtime

Если в процессе выполнения вышеперечисленных действий содержимое окон или диалоговых панелей не соответствует приведенному описанию, то это свидетельствует о неудачном завершении установки пакета адаптации и требуется выполнить его повторную установку.

3.4. Настройка параметров трансляции проекта CoDeSys

После создания проекта для платформы Fastwel I/O в IDE CoDeSys для получения детальной диагностической информации о доступе к областям входных и выходных данных программы при трансляции проекта:

1. Выберите вкладку **Resources** в левой области главного окна IDE CoDeSys и дважды щелкните на узле **Workspace** в дереве **Resources** (или выберите команду **Options** в меню **Project**);
2. В появившейся диалоговой панели **Options** выберите опцию **Build** и отметьте флажки, входящие в группу **Check Automatically**, как показано на рис. 7.

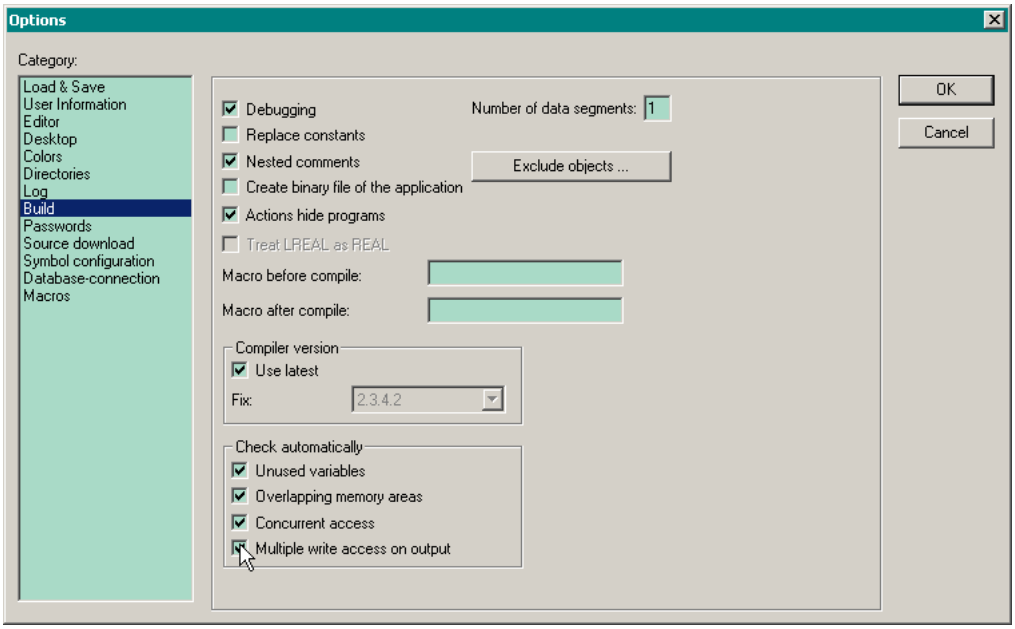


Рис. 7. Настройка параметров трансляции проекта IDE CoDeSys

Таким образом, при трансляции проекта по команде **Rebuild All** меню **Project** будут выводиться сообщения о наличии неиспользуемых переменных, о пересечении адресов разных переменных, отображаемых на области входных и выходных данных программы и о наличии в программе нескольких выходных переменных, отображаемых на перекрывающиеся участки в области выходных данных программы.

3.5. Удаление адаптированной IDE CoDeSys

Для удаления адаптированной IDE CoDeSys:

1. В панели управления Windows дважды щелкните левой кнопкой мыши на элементе **Add/Remove Programs**;
2. В появившейся диалоговой панели **Add/Remove Programs** выберите строку *Fastwel CoDeSys Adaptation* и нажмите кнопку **Change/Remove**, после чего нажмите кнопку **Yes** в появившейся диалоговой панели **Confirm File Deletion**. Произойдет удаление файлов пакета адаптации IDE CoDeSys для Fastwel I/O;
3. Для удаления среды разработки IDE CoDeSys в диалоговой панели **Add/Remove Programs** выберите строку *CoDeSys Automation Alliance* и нажмите кнопку **Change/Remove**;
4. В появившейся диалоговой панели **InstallShield Wizard** установите переключатель в положение **Remove** и нажмите кнопку **Next**, после чего следуйте указаниям программы удаления IDE CoDeSys.

3.6. Указания по обновлению системного программного обеспечения контроллера

Для просмотра текущей версии и обновления системного программного обеспечения контроллера должна использоваться **Сервисная утилита FASTWEL IO**, программа установки которой находится на ftp-узле фирмы Прософт по адресу:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/Utilities/ServiceUtility.

Наиболее актуальное обновление системного программного обеспечения контроллера может быть загружено с ftp-узла фирмы Прософт по следующим адресам:

для контроллера CPM701:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM701

для контроллера CPM702:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM702

для контроллера CPM703:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM703/

для контроллера CPM704:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM704

Для просмотра текущей версии в среде разработки CoDeSys:

1. В среде разработки CoDeSys откройте проект (файл с расширением *.pro), посредством которого была получена текущая пользовательская программа контроллера. Если в контроллере отсутствует исполняющееся приложение, создайте новый проект в соответствии с указаниями п. 5.2 настоящего руководства.
2. Если контроллер подключен к сети (CAN, MODBUS или MODBUS TCP, в зависимости от типа контроллера), установите соединение между средой разработки CoDeSys и контроллером через соответствующий коммуникационный канал согласно указаниям руководства по настройке и программированию сетевых средств на данный контроллер путем выполнения команды **Online–Login** в среде разработки CoDeSys.

   Если контроллер не подключен к сети, подключите контроллер к последовательному порту компьютера, на который установлена среда разработки CoDeSys, при помощи кабеля последовательной связи ACS00019, после чего установите соединение между средой разработки CoDeSys и контроллером через коммуникационный канал P2P.
3. Если после выполнения команды **Online-Login** на экране монитора появится диалоговая панель *The program has changed...*, нажмите в ней кнопку **Details**. Диалоговая панель CoDeSys примет вид, показанный на рис. 8.

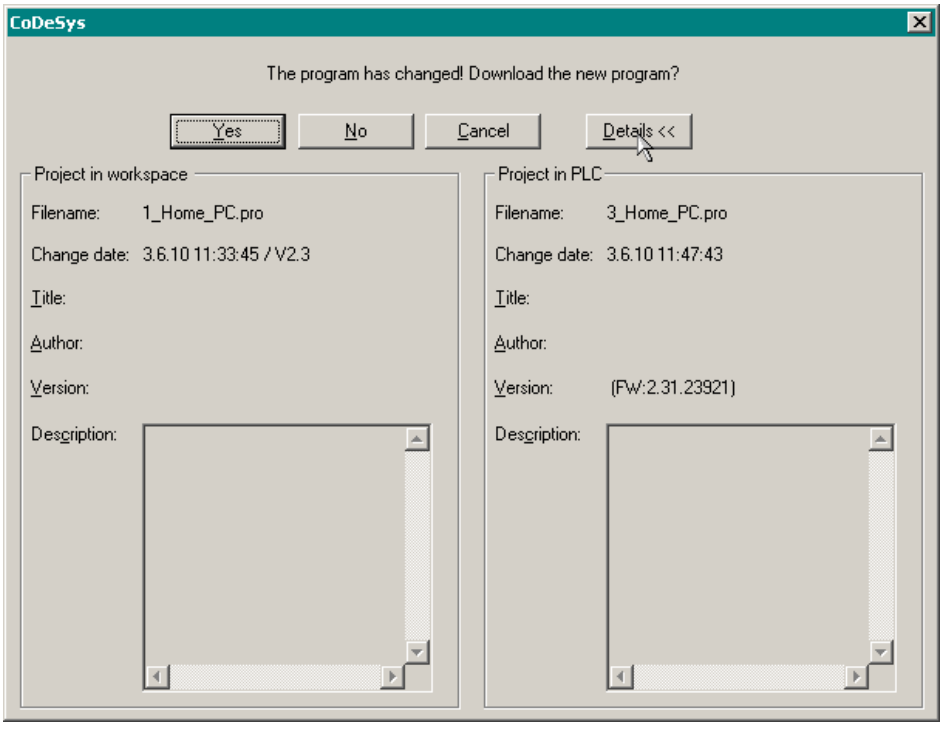


Рис. 8. Просмотр информации о версии системного ПО контроллера

4. Номер версии системного программного обеспечения, загруженного в контроллер, отображается в скобках после префикса *FW:* в поле **Project in PLC – Version**, как показано на рис. 8.

4. ПРИНЦИП РАБОТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОНТРОЛЛЕРА

4.1. Общие сведения

4.1.1. Приложение пользователя

Контроллеры серии Fastwel I/O относятся к классу программируемых логических контроллеров, т.е. для того, чтобы контроллер начал выполнять какую-либо полезную работу, пользователь должен создать приложение в адаптированной среде разработки CoDeSys и загрузить в контроллер.

Приложение, разрабатываемое пользователем в среде CoDeSys, должно включать в себя следующие элементы:

1. Как минимум, одну программу – программную единицу типа *PROGRAM*, добавляемую в древовидный список проекта **POUs**, показанный на рис. 9, по команде контекстного меню **Add Object**.

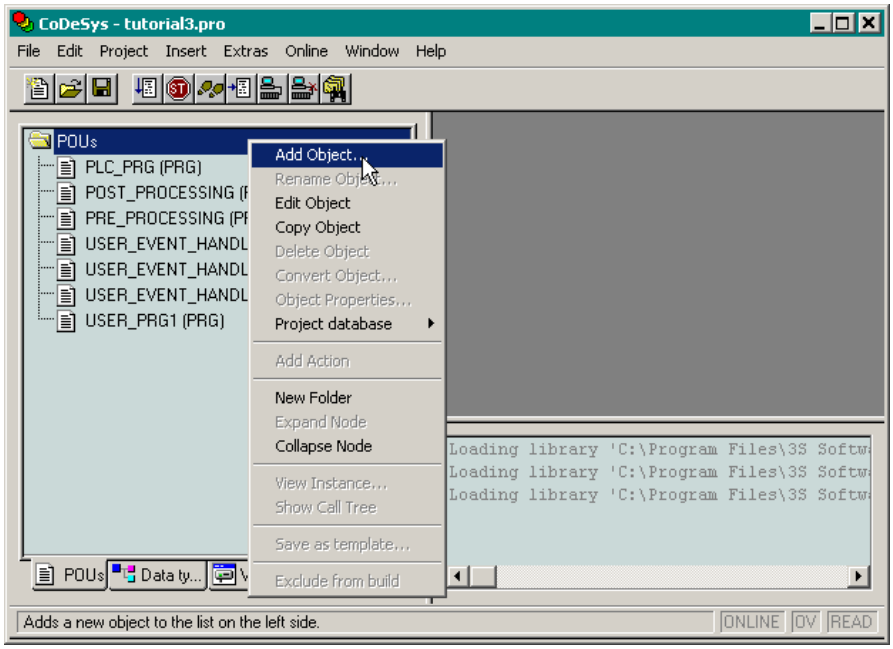


Рис. 9. Добавление программной единицы в древовидный список POU

2. Множество циклических задач. Циклической задачей является вычислительный процесс типа *Cyclic Task*, описываемый пользователем в окне ресурса **Tasks Configuration** проекта CoDeSys. С задачей может быть ассоциирована одна или несколько программ – программных единиц IEC 61131-3 типа *PROGRAM*.
   Программы, ассоциированные с некоторой циклической задачей, запускаются на исполнение под управлением отдельного потока исполнения операционной системы контроллера циклически с периодом задачи в порядке следования в списке POU данной задачи, как показано на рис. 10.
   Помимо периода запуска, каждая циклическая задача имеет приоритет, согласно которому планировщик операционной системы выбирает, какому потоку исполнения выделить процессорное время в тот или иной момент.
   Задача, имеющая большее значение приоритета, вытесняет задачу с меньшим значением приоритета – вытесненная задача прерывается на текущей выполняемой инструкции и не продолжает выполнение до тех пор, пока не завершится очередной цикл вытеснившей ее более приоритетной задачи.

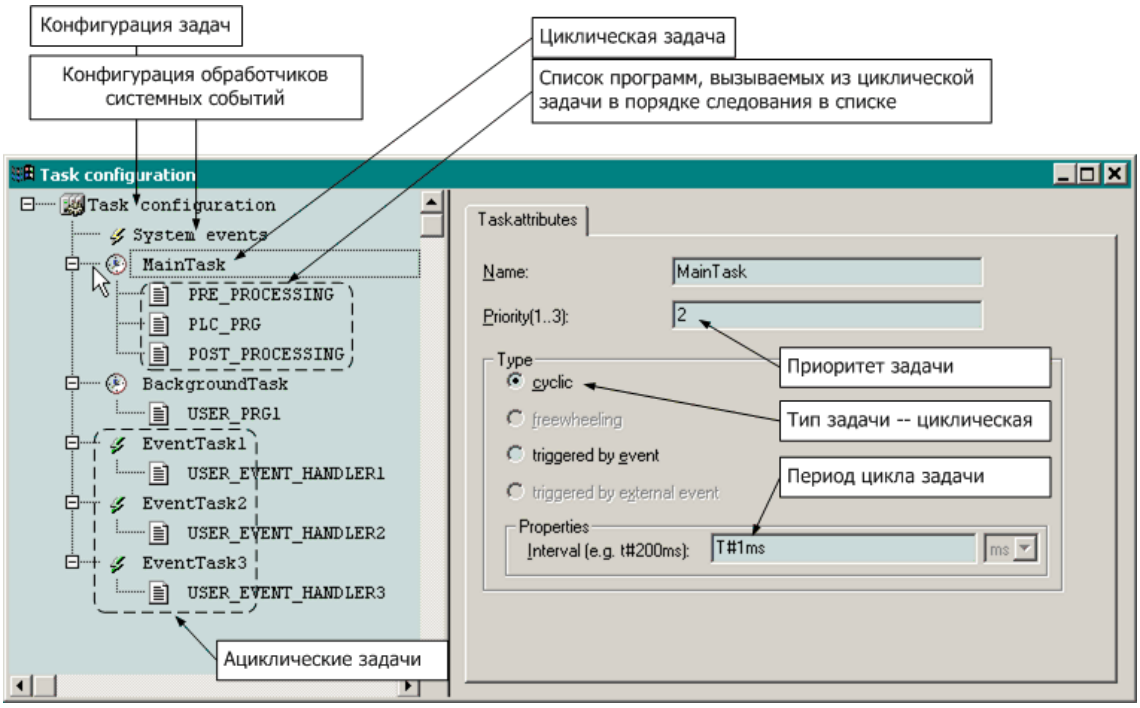


Рис. 10. Пример конфигурации задач приложения пользователя

3. Множество ациклических задач. Ациклической задачей является вычислительный процесс типа *Event Task (triggered by event)*, описываемый пользователем в окне ресурса проекта CoDeSys **Tasks Configuration**. С ациклической задачей может быть ассоциирована одна или несколько программ – программных единиц IEC 61131-3 типа *PROGRAM*.
   Программы, ассоциированные с некоторой ациклической задачей, запускаются на контексте высокоприоритетного потока исполнения операционной системы (сервисной задачи) в момент перехода из состояния FALSE в состояние TRUE некоторой булевой переменной (источника события), определенной в свойствах задачи параметром **Task Attributes–Properties–Event**.
   Проверка изменений переменных-источников событий выполняется сервисной задачей с периодом, который определен пользователем при помощи параметра *CPM70x ... Controller:SampleRate* (по умолчанию – 10 мс) в окне ресурса **PLC Configuration**.
   Помимо переменной-источника события, каждая ациклическая задача имеет приоритет и порядковый номер, согласно которым среда исполнения выбирает очередность запуска ациклических задач. При одновременном обнаружении нескольких событий (передних фронтов нескольких переменных источников событий) порядок запуска ациклических задач определяется соотношением их приоритетов (выше приоритет – раньше запуск), а, в случае равенства, – порядковым номером задачи (меньше номер – раньше запуск).
4. Множество обработчиков системных событий. Обработчиком системного события называется функция (программная единица типа *FUNCTION*), назначенная пользователем для одного или нескольких системных событий во вкладке **Tasks Configuration–System events**. К системным событиям относятся моменты запуска и останова приложения, окончание подготовки приложения к запуску, начало загрузки нового приложения, момент перед заменой текущего приложения на вновь загруженное и другие.

ВНИМАНИЕ!

В виду особенностей соглашения о вызовах функций, используемого кодогенератором CoDeSys, функции, устанавливаемые в качестве обработчиков системных событий, должны иметь следующий неизменный интерфейс: единственную входную переменную типа DWORD, возвращаемый результат типа DWORD, и ни одной локальной переменной. При необходимости использования локальных переменных в функциях обработки системных событий следуйте рекомендациям п. 4.2.4.4.

5. Конфигурация контроллера, состоящая из списков описаний модулей ввода-вывода и коммуникационных объектов внешней сети, которые должны использоваться системой исполнения контроллера во время работы приложения.
   Конфигурация контроллера определяется в окне ресурса **PLC Configuration** проекта CoDeSys, внешний вид которого представлен на рис. 11. В конфигурации, помимо описаний объектов разных подсистем контроллера, определяется структура образа процесса, а также могут быть объявлены символьные имена каналов ввода-вывода для последующего использования в приложении в качестве входных и выходных переменных.
   Кроме того, в конфигурации определяются значения различных параметров подсистем контроллера, и имеются каналы доступа к диагностической информации о работе подсистем.

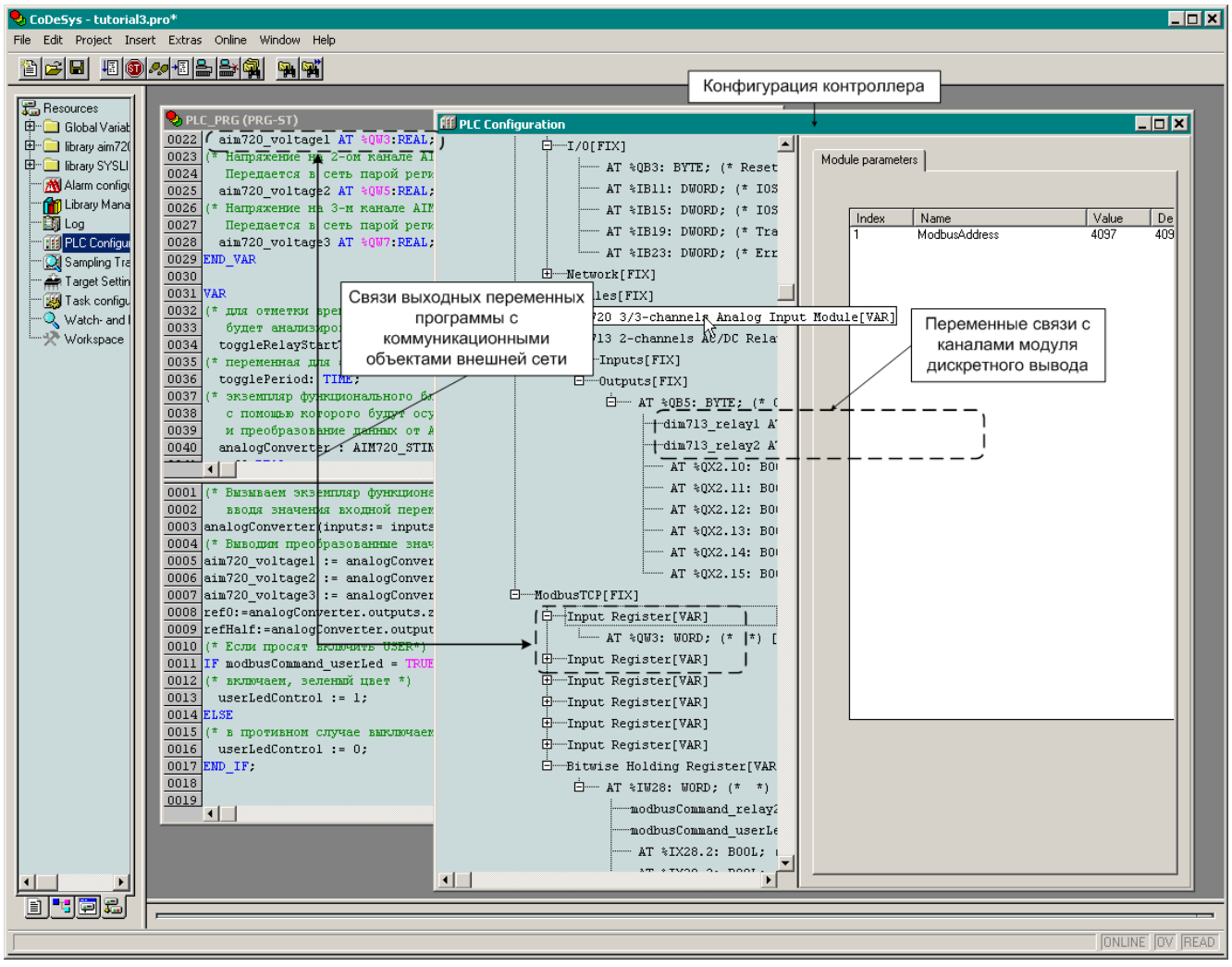


Рис. 11. Конфигурация контроллера

4.1.2. Назначение системного программного обеспечения контроллера

Приложение, разработанное пользователем в среде CoDeSys и загруженное в контроллер, исполняется под управлением адаптированной среды исполнения CoDeSys, которая интегрирована в системное программное обеспечение контроллера.

Системное программное обеспечение контроллера состоит из следующих основных сервисов:

1. Адаптированная среда исполнения CoDeSys – предназначена для исполнения кода приложения пользователя.
2. Сервис ввода-вывода – предназначен для управления, инициализации и обмена данными с модулями ввода-вывода, подключенными в внутренней шине контроллера. Обеспечивает функционирование стека протоколов внутренней шины контроллера, а также инициализацию, обнаружение и обработку ошибок обмена по внутренней шине и т.п.
3. Сервис внешней сети – состоит из стека протоколов внешней сети и сервиса протокола прикладного уровня (в контроллере CPM701 – CANopen; CPM702 – Modbus RTU/ASCII; CPM703 – Modbus TCP). Предназначен для обмена данными и командами с контроллером по сети. Описание принципов работы и способов конфигурировании перечисленных сервисов внешней сети приведено в соответствующих документах согласно п. 2.5.1.

Остальные сервисы, входящие в состав системного программного обеспечения контроллера, в настоящем документе подробно не рассматриваются.

4.2. Принципы работы адаптированной среды исполнения CoDeSys

4.2.1. Режимы работы

4.2.1.1. Безопасный режим

При поставке контроллер не содержит приложения пользователя и при включении питания запускается в так называемом безопасном режиме, о чем свидетельствует попеременное свечение индикатора RUN/ERR зеленым и красным цветами и прекращение свечения. В безопасном режиме системное программное обеспечение контроллера не обращается к модулям ввода-вывода и ожидает загрузки приложения пользователя.

В случае, когда неизвестны параметры обмена контроллера по внешней сети, и требуется загрузить приложение, контроллер может быть принудительно переведен в безопасный режим. Для этого следует включить переключатель «1» и перезапустить контроллер. В этом случае коммуникационные параметры контроллера примут значения по умолчанию (в соответствии с информацией руководства по конфигурированию и программированию сетевых средств на конкретный тип контроллера).

Контроллер может перейти в безопасный режим самостоятельно, если при запуске или во время функционирования приложения произошла какая-либо ошибка.

В безопасном режиме индикатор RUN/ERR циклически меняет свой цвет с зеленого на красный и погасает с частотой около 2 Гц.

Если контроллер не содержит приложения, загруженного пользователем из среды CoDeSys, индикаторы APP, IO и USER всегда погашены.

Если контроллер перешел в безопасный режим, причина перехода индицируется при помощи последовательности переключений индикатора APP или IO следующим образом:

1. N включений/выключений индикатора APP или IO с определенной частотой $F_{Hz} = 1/T_s$;
2. пауза длительностью $N \times T_s$, во время которой индикатор выключен;
3. возврат к шагу 1.

В случае, если ошибка, по которой произошел переход контроллера в безопасный режим, вызвана некоторой неточностью пользователя при разработке проекта, индикатор (APP или IO) во включенном состоянии имеет зеленый цвет.

Если ошибка вызвана более серьезными причинами, индикатор APP во включенном состоянии имеет красный цвет.

Кодовые последовательности индикации о нефатальных ошибках в загруженном приложении, которые формируются контроллером в безопасном режиме, представлены в табл. 1–3.

Таблица 1

Индикация безопасного режима по ошибке в прикладной программе Индикатор: **APP** Цвет: **Зеленый** Частота F_{Hz}: **2 Гц**	
N (кол-во включений)	**Причина**
2	Ошибка чтения/записи энергонезависимого хранилища проектной информации в контроллере, включая: – неправильная длина секции проектной информации в хранилище; – ошибка чтения/записи секции проектной информации в хранилище; – ошибка чтения/записи файла хранилища
3	Ошибка распределения сегментов памяти данных и кода программы из-за нехватки свободной оперативной памяти
4	Ошибка при инициализации кода программы, в том числе: – ошибка выполнения функции CodeInit, сгенерированной CoDeSys; – ошибка выполнения функции GlobalInit, сгенерированной CoDeSys; – ошибка привязки адресов переменных в процессе запуска программы (relocation error); – неправильный размер сегмента кода или данных, запрошенный программой.
5	Ошибки связывания и конфигурирования системы исполнения: – ошибка динамического связывания с библиотечными функциями (библиотечная функция, запрошенная программой, не найдена), как правило, вызванная тем, что пользователь добавил в приложение библиотеку, не поддерживаемую адаптированной средой исполнения; – ошибка связывания задачи с областью входных или выходных данных (обычно из-за неправильного отображения переменных на область входных или выходных данных), как правило, вызванная тем, что пользователь удалил из конфигурации контроллера ранее имевшиеся там модули ввода-вывода или коммуникационные объекты и не выполнил пару команд Project–Clean All и Project–Rebuild All перед загрузкой приложения в контроллер; – ошибка конфигурирования задачи, как правило, вызванная тем, что пользователю удалось добавить в конфигурацию контроллера большее количество задач, либо задать неправильные параметры задачи, а среда разработки CoDeSys этого не заметила.
6	Ошибка разбора конфигурации контроллера: неправильный формат конфигурации или неправильный тип контроллера (например, в контроллер CPM703 загружена конфигурация контроллера CPM702).
7	– бесконечный цикл в каком-либо обработчике системного события или в ациклической задаче; – сбой по цепям питания во время операции перестановки первичного и вторичного хранилищ приложения в контроллера
8	резерв

Таблица 2[1]

Индикация безопасного режима по ошибке в конфигурации сервиса внешней сети Индикатор: **APP** Цвет: **Зеленый** Частота F_{Hz}: **1 Гц**	
N (кол-во включений)	**Причина**
2	– неправильный тип протокола; – неправильный тип сети; – неподдерживаемый тип сетевого устройства; – отсутствующий номер сетевого устройства
3	неправильный адрес (идентификатор) узла
4	– в конфигурации внешней сети имеются два и более коммуникационных объектов с одинаковым идентификатором (ModbusAddress или COB_ID); – неправильное значение идентификатора коммуникационного объекта (выходящее за пределы допустимого диапазона)
5	– неправильный формат конфигурации; – неправильный тип коммуникационного объекта; – неподдерживаемый или неправильный параметр коммуникационного объекта
6	резерв

Таблица 3

[1] Информация об индикации безопасного режима контроллера CPM704 приведена в " CPM704. Контроллер узла сети PROFIBUS DP-V1. Руководство по конфигурированию и программированию сетевых средств ".

Индикация безопасного режима по ошибке в конфигурации сервиса ввода-вывода Индикатор: **IO** Цвет: **Зеленый** Частота F_{Hz}: 2 **Гц**	
N (кол-во включений)	**Причина**
3	ошибка конфигурирования сервиса ввода-вывода
4	неправильный формат конфигурации сервиса ввода-вывода
5	количество модулей в конфигурации превышает 64
6	резерв

Кодовые последовательности индикации о фатальных ошибках в загруженном приложении, которые формируются контроллером в безопасном режиме, представлены в табл. 4.

Таблица 4

Индикация безопасного режима по невосстановимой ошибке в прикладной программе Индикатор: **APP** Цвет: **Красный** Частота F_{Hz}: **2 Гц**	
N (кол-во включений)	**Причина**
бесконечное	Ошибка доступа к системной функции из-за порчи памяти системы исполнения при работе с указателями
2	Исключение при неправильном выравнивании кода или неправильном восстановлении стека вследствие ошибки кодогенератора CoDeSys (MISALIGNED CODE Exception)
3	Исключение по целочисленному делению на 0 (DIVISION BY ZERO Exception)
4	Исключение по неправильному коду операции в программе (INVALID OPCODE Exception). Ошибка является следствием: – неправильного восстановления кода возврата при завершении вызова функции прикладной программы; – неправильной упаковки входных и распаковки выходных параметров функций; – отсутствия функции, вызываемой из кода прикладной программы
5	Немаскируемое прерывание (FPU EMULATOR exception). Является следствием: – деления на 0 числа с плавающей точкой; – обращения по чтению к пустому или по записи к полному стеку эмулятора сопроцессора из кода прикладной программы
6	Исключение по выходу за границы массива (ARRAY BOUNDS exception)
7	Нехватка вычислительных ресурсов. Данная ошибка возникает в ситуации, когда контроллер в течение длительного времени не может начать исполнение загруженной программы или прекратить исполнение старой программы после загрузки новой. Причина может состоять в том, что одна из задач содержит бесконечный или почти бесконечный цикл (длительностью более 30-ти секунд).
8	резерв

Имеются два специальных режима индикации, когда индикатор APP прерывисто светится ("мигает") зеленым цветом без паузы:

1. Не удалось загрузить конфигурацию безопасного режима: непрерывное переключение индикатора APP с частотой 2 Гц.
2. Невосстановимое повреждение хранилища приложения: непрерывное переключение индикатора APP с частотой 1 Гц.

Оба случая специального режима индикации, как правило, проявляются при повреждении файловой системы. Если последующий перезапуск контроллера не позволяет устранить указанные признаки, обратитесь к поставщику для получения указаний по восстановлению работоспособности контроллера.

ВНИМАНИЕ!

Если контроллер запустился в безопасном режиме по ошибке, его коммуникационные параметры примут значения из последней успешно загруженной конфигурации.

При перезапуске контроллера, функционирующего в безопасном режиме по ошибке в ранее загруженном приложении, происходит сброс информации о последней причине перехода безопасный режим (однако в файле rstat.bin хранятся причины последних 30-ти перезагрузок). В связи с этим до установления причины перехода в безопасный режим не рекомендуется перезапускать контроллер.

Если по светодиодной индикации безопасного режима и имеющемуся проекту CoDeSys не удается самостоятельно установить и устранить причину перехода в безопасный режим, выполните следующие действия:

1. В среде разработки CoDeSys откройте проект приложения, загруженного в контроллер, и выполните команду **Online–Login** в отношении контроллера, функционирующего в безопасном режиме. На экран монитора будет выведена диалоговая панель *The program has changed...,* – нажмите в ней кнопку **No.**
2. Выполните команду **Online–Read file from PLC** в среде разработки CoDeSys и в появившейся диалоговой панели **Read file from PLC** введите имя файла *normdump.txt* и нажмите кнопку **Save**. На экране монитора появится окно, отображающее прогресс загрузки файла в контроллер.
3. Еще раз выполните команду **Online–Read file from PLC** в среде разработки и в появившейся диалоговой панели **Read file from PLC** введите имя файла *rstat.bin* и нажмите кнопку **Save**. На экране монитора появится окно, отображающее прогресс загрузки файла в контроллер.
4. Выполните команду **Online–Logout**
5. Выполните команду **File–Save/Mail Archive**
6. Отправьте выгруженные из контроллера файлы *normdump.txt, rstat.bin* и файл архива, сформированный средой разработки CoDeSys, по электронной почте на адрес soft.support@fastwel.ru.
   В сообщении, отправляемом по электронной почте, укажите:
   название своего предприятия,
   фамилию и инициалы,
   фактическую аппаратную конфигурацию контроллера (тип контроллера, номенклатуру модулей ввода-вывода в порядке подключения к контроллеру и тип используемого источника питания),
   обстоятельства, при которых контроллер перешел в безопасный режим.

Если контроллер запустился в безопасном режиме по ошибке, можно попробовать перезапустить его без выключения питания. Для этого следует изменить положение переключателя «2» (включить, если он выключен, или выключить, если включен).

Начиная с версии 2.64 системного программного обеспечения контроллеров и пакета адаптации CoDeSys 2.3 для Fastwel I/O, имеется возможность получения информации о причине перехода контроллера в безопасный режим при помощи команды *pinf* браузера ПЛК.

Для установления причины перехода в безопасный режим:

1. В среде разработки CoDeSys 2.3 откройте проект приложения, загруженного в контроллер, или создайте новый пустой проект и установите соединение с контроллером, выполнив команду **Online–Login** (**Онлайн–Подключение**). На экран монитора будет выведена диалоговая панель *The program has changed...(Программа была изменена!...),* – нажмите в ней кнопку **No** (**Нет**).
2. В CoDeSys 2.3 откройте окно ресурса **PLC Browser** (**ПЛК-Браузер**) и в поле ввода команд введите:

   ```
   pinf
   ```

   в области ответного сообщения на команду будет отображено:

   ```
   pinf
   Device: <наименование системы исполнения контроллера>
   Firmware Version: <версия системного ПО>
   Mode: SAFE
   Reason: <причина перехода в безопасный режим>
   ```

Строка с префиксом *Device:* содержит полное описание типа системы исполнения контроллера.

Строка *Firmware Version:* содержит информацию о версии системного программного обеспечения (микропрограммы) контроллера.

Строка *Mode:* содержит режим работы контроллера (*SAFE* – безопасный, *NORMAL* – нормальный).

Строка *Reason:* отображается только если контроллер функционирует в безопасном режиме и содержит описание причины перехода контроллера в безопасный режим

4.2.1.2. Нормальный режим

Если в контроллере имеется успешно загруженное приложение пользователя, при включении питания или перезапуске контроллер будет функционировать в нормальном режиме.

В нормальном режиме выполняются основные функции контроллера, включая исполнение задач и обработчиков системных событий, входящих в приложение, обмен данными с модулями ввода-вывода, обслуживание запросов, поступающих по внешней сети и т.д. При этом индикаторы контроллера будут светиться следующим образом:

RUN/ERR:

зеленый цвет

1. если в приложении имеется единственная циклическая задача, и она хотя бы иногда успевает укладываться в заданный период
2. если в приложении имеется более одной циклической задачи, и хотя бы одна из них хотя бы иногда успевает укладываться в заданный период

"Укладываться в заданный период" означает, что все программы, исполняемые под управлением данной задачи, заканчивают свою работу на очередном цикле до наступления времени начала следующего цикла и запускаются повторно строго в момент начала очередного цикла.

красный цвет – если в приложении имеется более одной циклической задачи, и ни одна из них никогда не успевает укладываться в заданный период.

APP:

отсутствие свечения – приложение содержит только ациклические задачи;

зеленый цвет – все циклические задачи всегда успевают укладываться в заданный период

красный цвет (непрерывно) – все циклические задачи никогда не успевают укладываться в заданный период

красный цвет (прерывисто) – хотя бы одна циклическая задача хотя бы иногда успевает укладываться в заданный период;

зеленый цвет (прерывисто) – одна циклическая задача из нескольких иногда не успевает укладываться в заданный период.

IO:

зеленый цвет (непрерывно) – все модули ввода-вывода, определенные в конфигурации проекта, обнаружены и успешно сконфигурированы. Обмен с модулями ввода-вывода, подключенными к контроллеру, успевает завершиться за время, равное значению параметра *SampleRate* в конфигурации сервиса ввода-вывода контроллера (**Resources–PLC Configuration**–*CPM70x ... Controller–I/O Modules*). В случае ускоренного преобразования проекта до новой версии значение данного параметра в проекте будет равным 0, поэтому в качестве истинного значения периода сервис ввода-вывода будет использовать период, определенный параметром *SampleRate* в конфигурации контроллера (**Resources–PLC Configuration**–*CPM70x ... Controller*);

зеленый цвет (прерывисто) – обмен с модулями ввода-вывода, подключенными к контроллеру, не может быть выполнен за время, заданное в конфигурации контроллера (**Resources–PLC Configuration**–*CPM70x ... Controller–I/O Modules:SampleRate*), в связи с чем используется расчетное минимально достижимое время обмена данными со всеми модулями при загрузке внутренней шины на 50%;

красный цвет – конфигурация модулей ввода-вывода, определенная в проекте, не совпадает с аппаратной конфигурацией модулей, подключенных к контроллеру. Кроме того, свечение красным сразу после загрузки нового приложения свидетельствует о выполнении конфигурирования сервиса ввода-вывода;

отсутствие свечения – в конфигурации загруженного приложения отсутствуют описания модулей ввода-вывода.

4.2.2. Процесс запуска контроллера после включения питания

4.2.2.1. Запуск при первом включении питания

При поставке контроллер не содержит пользовательского приложения и при включении питания запускается в безопасном режиме, о чем свидетельствует попеременное свечение индикатора RUN/ERR зеленым и красным цветами, и прекращение свечения. В безопасном режиме системное программное обеспечение контроллера не обращается к модулям ввода-вывода и ожидает загрузки в контроллер приложения, разработанного в среде CoDeSys. Более подробная информация об условиях запуска контроллера в безопасном режиме и об индикации безопасного режима приведена в п. 4.2.1.1 настоящего руководства.

4.2.2.2. Запуск при наличии загруженного приложения

Если перед последним выключением питания в контроллер было загружено приложение пользователя, после включения питания (или перезагрузки) выполняются следующие действия:

1. Проверяется причина последней перезагрузки, сохраненная в специальном файле. Если данный файл содержит информацию о какой-либо ошибке, происходит запуск контроллера в безопасном режиме. В противном случае предпринимается попытка запуска в нормальном режиме.
2. Открывается основное системное хранилище приложения пользователя, содержащее пять секций проектной информации: код приложения; конфигурацию сервиса ввода-вывода и внешней сети; конфигурацию задач; конфигурацию связей задач с образом процесса (каналами модулей ввода-вывода и коммуникационными объектами внешней сети) и секцию информации о проекте, включая имя проекта, дату создания, версию и пр., заданную пользователем при работе над проектом по команде **Project–Project Info** в среде разработки CoDeSys. Если открыть хранилище не удалось, контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией.
3. После успешного открытия хранилища приложения, выполняется проверка целостности данных во всех пяти секциях проектной информации путем последовательной проверки циклических контрольных сумм длин секций, а затем проверка контрольной суммы данных каждой секции. Если какая-либо секция повреждена, хранилище помечается, как пустое, после чего контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией причины перезагрузки.
4. После успешной проверки целостности информации в секциях приложения из хранилища, выполняется последовательная загрузка секций проектной информации и конфигурирование сервисов системного программного обеспечения контроллера.
5. В первую очередь загружается секция конфигурации устройств ввода-вывода и сервиса внешней сети, получаемая на основе информации, которая определена пользователем в окне ресурса **PLC Configuration** среды CoDeSys. После загрузки секции строится дерево конфигурации контроллера, структура которого в точности совпадает с той, что представлена в окне **PLC Configuration** для данного проекта. Если в процессе загрузки или построения дерева конфигурации обнаруживаются какие-либо ошибки, контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией.
6. Конфигурируется адаптированная среда исполнения CoDeSys. Сначала выполняется проверка контрольной суммы сегмента кода, сгенерированного средой разработки CoDeSys, который загружен из хранилища, после чего выполняется проверка достаточности размеров сегментов области данных приложения (фактические размеры приведены в п. 2.4.1 настоящего руководства). Если размер какого-либо сегмента оказывается недостаточным, предпринимается попытка его увеличения. При отрицательном результате контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией.
7. В загруженном сегменте кода вызывается функция инициализации кода, местоположение которой известно сразу после загрузки из заголовка сегмента кода. Далее в сегменте кода выполняется релокация: преобразование всех относительных ссылок на данные в сегментах области данных в абсолютные, полученные на основе фактического значения адреса сегмента данных. По окончании проверяется целостность ранее выделенной памяти системного программного обеспечения контроллера

(проверка целостности "кучи"). При отрицательном результате контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией.

8. Вызывается функция инициализация данных всех программных единиц (POU) приложения, также находящаяся в сегмента кода, сгенерированного средой CoDeSys, с последующей проверкой целостности ранее выделенной памяти и переходом в безопасный режим в случае отрицательного результата проверки. Кроме того, в этот момент происходит установка пользовательских обработчиков системных событий, заданных в окне ресурса **Tasks Configuration–Systems Events** загруженного проекта.
9. Выполняется загрузка секции конфигурации задач, а затем создание и конфигурирование циклических и ациклических задач, добавленных пользователем в окне **Tasks Configuration** загруженного проекта. Помимо этого выполняется анализ исполняемого кода с целью определения подмножеств программных единиц, вызываемых из каждой задачи. Последнее необходимо для отладчика среды исполнения, а также для последующего формирования диагностики в случае наличия фатальных ошибок в сгенерированном коде.
10. Загружается секция описателей связей задач с образом процесса, после чего сортировка описателей в порядке возрастания смещений, а далее – объединение описателей связей со смежными участками указанных областей.
11. Загружается секция информации о проекте.
12. На основе дерева конфигурации контроллера, построенного в п. 5, выполняется конфигурирование сервиса внешней сети, в процессе чего создаются входящие и исходящие коммуникационные объекты, а также устанавливаются параметры протокола. Параметры протокола (скорость обмена, адрес и т.п.), успешно извлеченные из дерева конфигурации контроллера, запоминаются в энергонезависимой памяти контроллера, чтобы в случае перезапуска в безопасный режим по ошибке использовать их для работы сервиса внешней сети.
13. На основе дерева конфигурации контроллера, построенного в п. 5, выполняется конфигурирование сервиса ввода-вывода, в процессе чего создается список описателей модулей ввода-вывода, которые должны быть подключены к внутренней шине контроллера, затем выполняется поиск модулей, фактически подключенных к контроллеру, а далее – конфигурирование обнаруженных модулей, чьи типы и местоположение на шине соответствуют ожидаемым в проекте и чьи параметры отличаются от полученных из запускаемого приложения.
14. В зависимости от режима, заданного пользователем с помощью параметра *CPM70x Controller–I/O Modules:ScanMode*, создается одна группа обмена с модулями ввода-вывода или множество групп, количество которых совпадает с количеством модулей ввода-вывода в поддереве *CPM70x Controller–I/O Modules* конфигурации контроллера. Для групп устанавливается период, определяемый параметром *CPM70x Controller–I/O Modules:SampleRate*. Если процесс конфигурирования сервиса ввода-вывода завершился не вполне успешно, контроллер продолжит работу в нормальном режиме с соответствующей индикацией.
15. Выполняется связывание с образом процесса всех объектов системы, нуждающихся в обмене данными (задач CoDeSys, коммуникационных объектов и групп обмена с модулями ввода-вывода) во время работы приложения. Если в процессе связывания обнаружены какие-либо ошибки, контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией. По окончании связывания вызываются пользовательские функции-обработчики системных событий *OnPowerOn*, а затем – *OnInit* (если они были установлены пользователем в приложении).
16. Происходит запуск обмена данными с модулями ввода-вывода, запускается сервис внешней сети, а также все задачи адаптированной среды исполнения CoDeSys. Непосредственно перед запуском вызывается пользовательская функция-обработчик системного события *OnStart* (если она была установлена пользователем в приложении).
17. Системное программное обеспечение контроллера функционирует в нормальном режиме, исполняя алгоритмы приложения, обмениваясь данными с модулями ввода-вывода и по внешней сети, а также выполняя диагностику всех подсистем.

4.2.3. Процесс загрузки или обновления приложения

Адаптированная среда исполнения CoDeSys для контроллеров Fastwel не позволяет выполнять загрузку в контроллер нового приложения посредством команды **Online–Create boot project**, возвращая среде разработки CoDeSys код ошибки 20019 (запись системного файла запрещена).

Процесс загрузки или обновления приложения начинается, когда пользователь, предварительно настроив коммуникационный канал взаимодействия с контроллером через интерфейс P2P или через интерфейс внешней сети, выполняет команду **Online–Login** в среде разработки CoDeSys и нажимает кнопку **Yes** в появившейся диалоговой панели *The program has changed! Download the new program?* Указанная диалоговая панель появляется в случае, если проект, открытый в среде CoDeSys, отличается от того, что находится контроллер. При изложении последующего описания процесса загрузки и обновления приложения предполагается, что контроллер функционирует в нормальном режиме согласно п. 4.2.4. Загрузка приложения при работе контроллера в безопасном режиме выполняется аналогично.

1. После того, как пользователь в среде разработки CoDeSys нажимает кнопку **Yes** в диалоговой панели с сообщением *The program has changed! Download the new program?*, среда разработки начинает загрузку секции исполняемого кода текущего загружаемого приложения. В диалоговой панели статуса загрузки изменяется значение счетчика загруженных байт. В момент начала загрузки секции кода вызывается пользовательская функция-обработчик системного события *OnDownload* (если она была установлена пользователем в приложении). Во время загрузки секций нового приложения работа текущего приложения контроллера, обмен данными с модулями ввода-вывода и обмен по сети, в отличие от предыдущей версии, **не прекращаются**.
2. По окончании загрузки секции кода происходит запись секции во вторичное хранилище приложения в энергонезависимой памяти контроллера (хранилище для загружаемого нового приложения). В случае ошибки записи секции кода среде разработки передается код ошибки 20014 (ошибка загрузки секции кода). После успешного сохранения секции кода адаптированная среда исполнения CoDeSys контроллера ожидает секцию конфигурации загружаемого приложения.
3. Среда разработки CoDeSys, убедившись, что секция кода загружена успешно, приступает к загрузке секции конфигурации приложения, определяемой в окне **PLC Configuration**. В диалоговой панели статуса загрузки по причине, известной только разработчикам среды CoDeSys, прекращается изменение значения счетчика загруженных байт.
4. По окончании загрузки секции конфигурации контроллера происходит запись секции во вторичное хранилище. В случае ошибки записи секции конфигурации среде разработки передается код ошибки 20015 (ошибка загрузки секции конфигурации). После успешного сохранения секции конфигурации контроллера адаптированная среда исполнения CoDeSys контроллера ожидает секцию конфигурации задач загружаемого приложения.
5. Среда разработки CoDeSys, убедившись, что секция конфигурации загружена успешно, приступает к загрузке секции конфигурации задач, определяемой в окне **Tasks Configuration**. В диалоговой панели статуса загрузки изменение значения счетчика загруженных байт по-прежнему не происходит.
6. По окончании загрузки секции конфигурации задач происходит запись секции во вторичное хранилище приложения в энергонезависимой памяти контроллера. В случае ошибки записи секции конфигурации задач среде разработки передается код ошибки 20016 (ошибка загрузки секции конфигурации задач). После успешного сохранения адаптированная среда исполнения CoDeSys контроллера ожидает секцию информации о проекте загружаемого приложения.
7. Среда разработки CoDeSys, убедившись, что секция конфигурации задач загружена успешно, приступает к загрузке секции информации о проекте загружаемого приложения, определяемой пользователем в диалоговой панели **Project Information**, вызываемой по команде меню **Project–Project Info.** В диалоговой панели статуса загрузки изменение значения счетчика загруженных байт не происходит.
8. По окончании загрузки секции информации о проекте загружаемого приложения выполняется запись секции во вторичное хранилище приложения в энергонезависимой

памяти контроллера. В случае ошибки записи секции среде разработки передается код ошибки 20018 (ошибка загрузки секции информации о проекте). После успешного сохранения секции адаптированная среда исполнения CoDeSys контроллера ожидает секцию описателей связей задач загружаемого приложения.

9. Среда разработки CoDeSys, убедившись, что секция информации о проекте загружена успешно, приступает к загрузке секции описателей связей задач загружаемого приложения с образом процесса. Содержимое секции определяется созданными пользователем входными и выходными переменными программ, ссылающимися на непосредственные адреса в областях входных и выходных данных. В диалоговой панели статуса загрузки изменение значения счетчика загруженных байт по-прежнему не происходит.
10. По окончании загрузки секции описателей связей задач происходит сохранение данной секции во вторичное хранилище приложения в энергонезависимой памяти контроллера. В случае ошибки записи секции среде разработки передается код ошибки 20017 (ошибка загрузки проекта).
11. После успешной загрузки и сохранения последней секции загружаемого приложения (связей задач), системное программное обеспечение контроллера выполняет проверку целостности секций, загруженных во вторичное хранилище в соответствии с шагом 3 п. 4.2.2.2, после чего, при равенстве длин, выполняется побайтовое сравнение только что загруженных секций во вторичном хранилище с содержимым однотипных секций в первичном хранилище, откуда был осуществлен успешный запуск текущего приложения. Если никаких отличий не обнаружено, контроллер продолжает функционировать в нормальном режиме, исполняя ранее загруженное приложение.
12. Если в какой-либо секции вторичного хранилища обнаружены какие-либо отличия от содержимого однотипной секции первичного хранилища, контроллер приостанавливает исполнение приложения и обмен данными по сети и с модулями ввода-вывода по внутренней шине и, в зависимости от типов секций, где обнаружены изменения, выполняется выборочное или полное конфигурирование сервисов контроллера согласно последовательности шагов 5–16 п. 4.2.2.2. Перед выполнением указанных действий вызывается пользовательская функция-обработчик системного события *OnProgramChange* (если она была установлена пользователем в приложении).
Полный перечень реконфигурирующих действий по указанным шагам выполняется в случае, если изменилось содержимое всех секций приложения или если в секции информации о проекте изменились название проекта, имя файла проекта, информация о версии, информация об авторе проекта или краткое описание проекта (см. диалоговую панель **Project Information** в среде CoDeSys).
Если какая-либо секция во вновь загруженном приложении оказалась неизменной, соответствующие реконфигурирующие действия не выполняются.
13. Если секция кода изменилась, но параметр *Fastwel I/O System Configuration:HotUpdateDisabled (горячее обновление отключено)* установлен пользователем в *No (Нет)*, выполняются дополнительные действия, цель которых – по возможности сохранить неизменными значения всех внутренних, входных и выходных переменных предыдущего приложения, которое подвергается обновлению.
Значения внутренних, входных и выходных переменных предыдущего приложения сохраняются неизменными, если:

остались неизменными все секции объявлений переменных во всех программных единицах проекта, включая количество и типы переменных, а также заданные для них инициализирующие значения;
остались неизменными связи задач с образом процесса;
в конфигурации контроллера не изменились размеры областей входных и выходных данных.

14. По окончании этапа связывания объектов, нуждающихся в обмене данными, с образом процесса пользовательская функция-обработчик системного событий *OnPowerOn* не вызывается, но последовательно вызываются *OnInit* и *OnStart*.
15. Производится перестановка первичного и вторичного хранилищ приложения: первичное хранилище, в котором находилось старое приложение, помечается, как

пустое, после чего делается вторичным. Вторичное хранилище, в которое было загружено новое приложение, делается первичным.

4.2.4. Исполнение приложения пользователя

4.2.4.1. Общие сведения

Описание состава приложения пользователя приведено в п. 4.1.1. Сокращенная архитектура адаптированной среды исполнения CoDeSys в упрощенной нотации UML представлена на рис. 12.

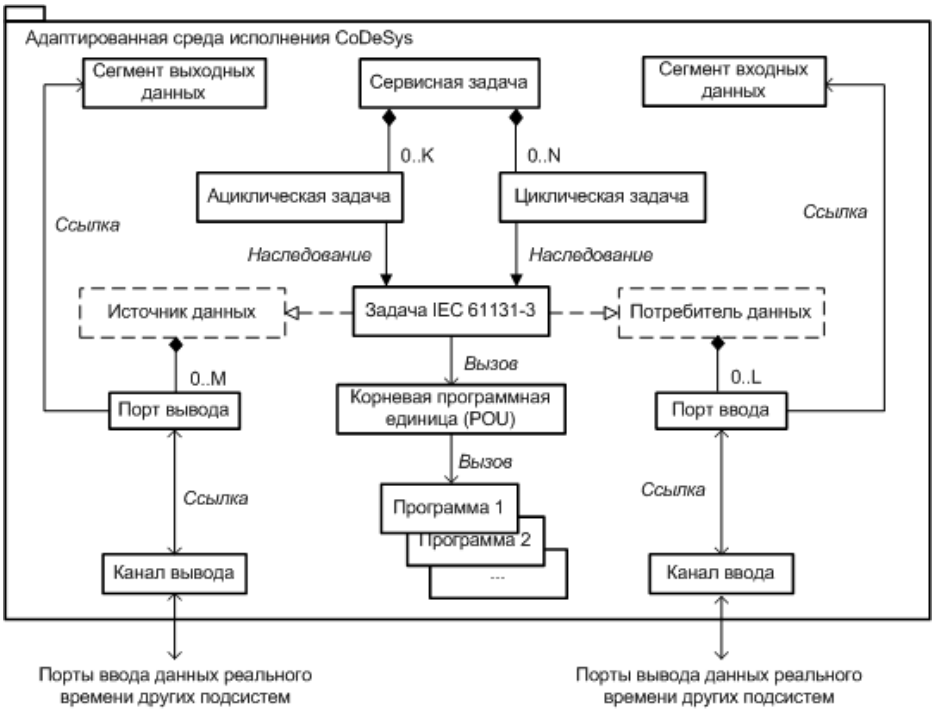


Рис. 12. Архитектура адаптированной среды исполнения CoDeSys

Одним из основных активных элементов архитектуры адаптированной среды исполнения CoDeSys является сервисная задача. Для сервисной задачи создан отдельный высокоприоритетный поток исполнения операционной системы. Сервисной задаче принадлежат множества объектов, представляющих циклические и ациклические задачи, описания которых формируются средой разработки CoDeSys, а также глобальная область данных, полученная у операционной системы на основе информации, сгенерированной средой разработки.

На сервисную задачу возлагается выполнение следующих основных функций:

1. Ввод данных из образа процесса в сегмент входных данных приложения.
2. Прием и обслуживание запросов среды разработки CoDeSys, поступающих по сети или через интерфейс прямого соединения P2P, включая чтение/запись переменных, загрузку приложения, чтение/запись файлов, вставку и снятие точек останова во время отладки и т.п.
3. Вызов большинства пользовательских функций обработки системных событий.
4. Управление ациклическими задачами, включая проверку переменных-событий ациклических задач на наличие перехода их значений с FALSE к TRUE, обмен (ввод-вывод) данными ациклических задач с образом процесса и вызов корневых программных единиц ациклических задач. Корневая программная единица некоторой задачи является функцией, которая сгенерирована компилятором среды разработки CoDeSys, и содержит вызовы (запуски) программ, ассоциированных пользователем с данной задачей, в порядке, установленном для программ задачи в ресурсе **Tasks Configuration**.
5. Исполнение программных единиц единственной циклической задачи, добавленной в ресурс **Tasks Configuration**.
6. Диагностику работы циклических задач, включая анализ значений счетчиков циклов и запаздываний каждой циклической задачи, обновление соответствующей информации в подобласти диагностики системы образа процесса, а также перевод контроллера в

безопасный режим в случае, если какая-либо из циклических задач не выполнила ни одного цикла в течение более чем 30 с.

7. Взаимодействие со сторожевым таймером с целью предотвращения зависания контроллера при вызовах пользовательских обработчиков системных событий или ациклических задач.

Циклические и ациклические задачи, которые пользователь добавляет в конфигурацию задач приложения в среде разработки CoDeSys, представляются соответствующими классами в архитектуре адаптированной среды исполнения, унаследованными от общего базового класса "Задача IEC 61131-3", далее называемого *пользовательская задача*. Для каждой пользовательской задачи, помимо ее специфических параметров, среда разработки передает в контроллер *индекс корневой программной единицы* и два множества *описателей ссылок задачи на входную и выходную области образа процесса*.

Корневой программной единицей является скрытая от пользователя программа, в которую вставлены вызовы ассоциированных с пользовательской задачей программ в порядке их перечисления в окне ресурса **Tasks Configuration**, как показано на рис. 13.

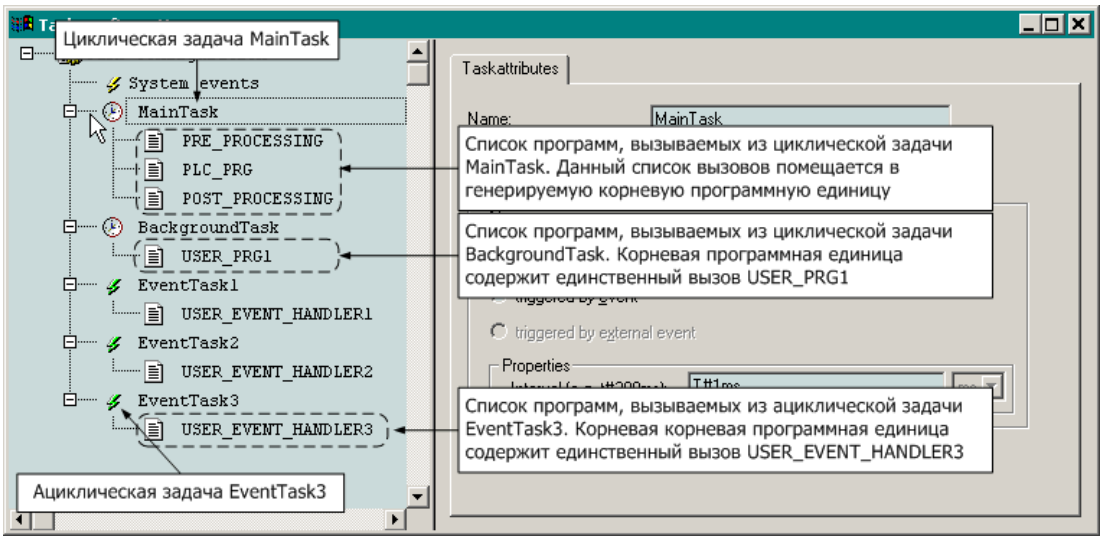


Рис. 13. Списки программных единиц, вызываемых из задач

Описатель ссылки на входную или выходную область образа процесса формируется средой разработки CoDeSys во время трансляции проекта для каждой непосредственно представляемой (directly represented) переменной, принадлежащей программе, которая вызывается из данной задачи.

Принцип формирования описателей ссылок иллюстрируется рис. 17. Согласно данному принципу, если какой-либо адрес в области входных или выходных данных явно или косвенно, через соответствующую непосредственно представляемую переменную, используется в теле программы в качестве операнда или результата выражения, то для задачи, из которой явно, или через цепочку вызовов, вызывается данная программа, при компиляции проекта будет сгенерирована ссылка на соответствующую область в формате (*смещение*, *длина*), где *смещение* – смещение в битах в соответствующей области образа процесса, а *длина* – длина ссылки в битах. Для отдельных (скалярных) переменных типа BOOL, ссылающихся на области входных и выходных данных, длина ссылки равна 0.

ВНИМАНИЕ!

Для полей типа BOOL переменных непримитивного типа (STRUCT или ARRAY), ссылающихся на область входных или выходных данных образа процесса, длина в описателях ссылок будет равна 8!

Множества описателей ссылок каждой задачи на образ процесса передаются контроллеру во время загрузки приложения, при этом среда разработки не выполняет сортировку элементов множеств по возрастанию смещений и не объединяет ссылки на смежные участки памяти в образе процесса. В связи с этим для оптимизации операций ввода-вывода перед связыванием задач с образом процесса система выполняет сортировку и объединение ссылок на смежные участки.

Перед началом связывания по оптимизированным множествам описателей ссылок задач на образ процесса для каждой пользовательской задачи сначала создаются специальные объекты сервиса обмена данными реального времени контроллера, называемые входными и выходными портами, через которые осуществляется чтение и запись образа процесса. Кроме того, перед связыванием создаются

два объекта связи, представляющих образ процесса для всех источников и потребителей данных реального времени системы, называемые каналами ввода и вывода.

Глобальная область памяти, выделяемая адаптированной средой исполнения на основе соответствующей информации в загруженном приложении пользователя, разделена на следующие основные сегменты: сегмент входных данных, сегмент выходных данных и сегмент внутренних переменных и экземпляров функциональных блоков (экземпляр – объект-переменная некоторого типа, каковым является функциональный блок). Переменные, ссылающиеся на адреса во входной области образа процесса, размещаются компилятором во входном сегменте данных, а аналогичные переменные, ссылающиеся на адреса в выходного области образа процесса – в выходном сегменте. Каждый входной порт задачи содержит смещение одиночной входной переменной или группы входных переменных, занимающих во входном сегменте непрерывный участок памяти, во входном сегменте приложения, а также длину переменной или группы переменных. То же самое касается каждого выходного порта по отношению к выходному сегменту приложения.

Образ процесса, о котором до сих говорилось в настоящем документе, буквально является парой буферов, размеры которых в точности совпадают с размерами входного и выходного сегментов приложения. Указанные буфера ассоциируются с каналами ввода и вывода при создании или повторном конфигурировании последних.

Процесс связывания выходных портов пользовательских задач состоит в задании подобласти буфера канала вывода в качестве приемника данных для операций записи в каждый порт, а местоположение и размер данных каждого порта в соответствующей части выходного сегмента приложения определяется смещением и длиной, полученными из описателя ссылки, на основе которого был создан данный порт. Кроме того, при связывании каждый порт добавляется в соответствующее множество ссылок на порты-источники данных в канале.

Начиная с версии 2.31 системного программного обеспечения контроллера, ввод данных модулей ввода-вывода и входящих коммуникационных объектов внешней сети в сегмент входных данных приложения всегда осуществляется сервисной задачей. При этом сегмент входных данных обновляется целиком с периодом, минимальным среди всех циклических задач, и только тогда, когда все циклические задачи завершили очередной цикл. Это позволяет избежать проблемы перекрытия общих входных данных, используемых в алгоритмах разных циклических и ациклических задач.

По завершении исполнения корневой программной единицы каждой пользовательской задачи выполняется последовательная запись во все ее выходные порты, в результате чего значения ее переменных, отображенных на выходной сегмент приложения, копируются в соответствующие участки выходной части образа процесса. Во время записи в выходные порты некоторой пользовательской задачи чтение канала вывода запрещено.

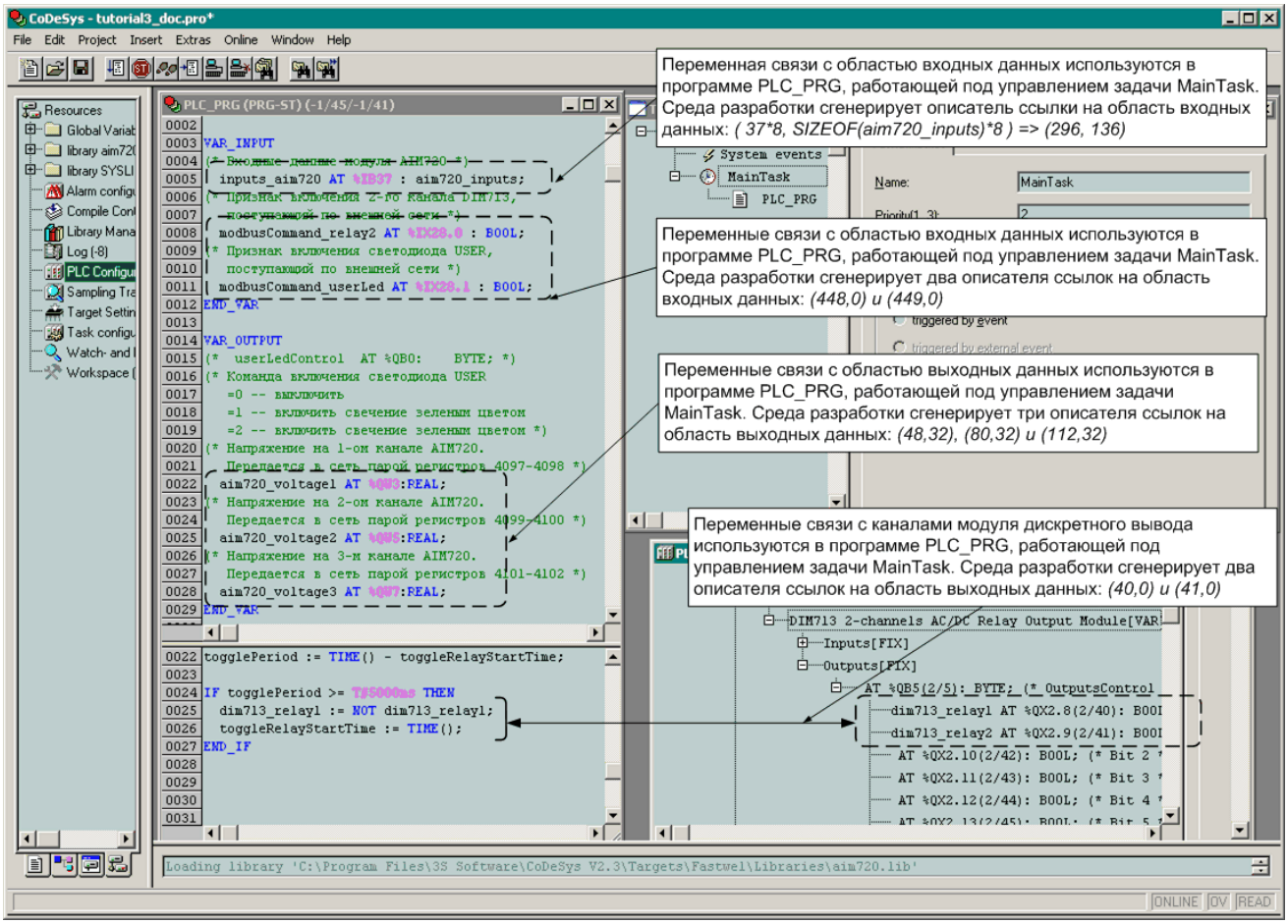


Рис. 14. Принцип формирования описателей ссылок на области входных и выходных данных приложения

Кажущаяся сложность описанного механизма обусловлена тем, что пользовательские задачи запускаются с разными частотами асинхронно друг относительно друга и относительно сервисов ввода-вывода и внешней сети. Указанный механизм позволяет реализовать требование когерентности входных данных каждой пользовательской задачи, устанавливаемое стандартом IEC 61131-3.

4.2.4.2. Исполнение циклических задач

Если в проекте не определено ни одной или присутствует только одна, и при том циклическая, задача, исполнение приложения осуществляется на контексте высокоприоритетной сервисной задачи.

Если пользователь не добавил ни одной задачи в ресурс **Task Configuration** то программа PLC_PRG будет исполняться под управлением высокоприоритетной сервисной задачи с периодом, заданным в ресурсе **PLC Configuration** для параметра *CPM70x ... Programmable Controller:Sample Rate*.

Если пользователь добавил в ресурс **Task Configuration** только одну циклическую задачу и ни одной ациклической, то программа, ассоциированная с единственной задачей, будет исполнятся под управлением высокоприоритетной сервисной задачи с периодом, заданным для задачи параметром **Properties – Interval** в свойствах задачи,.

При выполнении единственной циклической задачи под управлением высокоприоритетной сервисной задачи производятся следующие действия:

1. Ввод данных из области входных данных образа процесса в сегмент входных данных.
2. Вызов корневой программной единицы задачи.

   Если в пользовательском коде имеется бесконечный (или почти бесконечный) цикл, сервисная задача зависнет и система примерно на 40 с утратит работоспособность, после чего контроллер будет переведен в безопасный режим по зависанию сервисной задачи (7 "миганий" индикатора APP зеленым цветом).

Если в пользовательском коде имеется деление на 0 целочисленных операндов, возникнет исключение DIVISION BY ZERO, и контроллер перейдет в безопасный режим с соответствующей индикацией (3 "мигания" индикатора APP красным цветом).

Если в пользовательском коде имеется деление на 0 операндов с плавающей точкой, возникнет исключение FPU EMULATOR, и контроллер перейдет в безопасный режим с соответствующей индикацией (5 "миганий" индикатора APP красным цветом).

3. При успешном завершении работы корневой программной единицы задачи выполняется запись во все выходные порты задачи. Во время записи в порты канал вывода блокирован (объектом синхронизации типа "критическая секция") для чтения со стороны связанных с ним входных портов других участников обмена данными.
4. Вычисляется суммарное время ввода-вывода и исполнения текущего цикла в микросекундах.
5. Вычисляется время в миллисекундах, в течение которого данная задача должна находится в состоянии пассивного ожидания ("спать"), отдав процессор другим, в том числе системным, задачам.
6. Увеличивается счетчик циклов данной циклической задачи. Если время исполнения и ввода-вывода на текущем цикле превысило заданный период цикла, увеличивается счетчик запаздываний данной задачи.

   ВНИМАНИЕ! Если оказывается, что время исполнения и ввода-вывода на текущем цикле превысило 65% периода единственной циклической задачи, то во избежание перегрузки системы период исполнения задачи увеличивается в два раза. Например, если изначально был задан период 10 мс, а на очередном цикле приложение выполнялось дольше 6500 мкс (скажем, 6750 мкс), то измеренное время исполнения (6750 мкс) будет умножено на 2, округлено до ближайшей верхней миллисекунды и присвоено периоду цикла, что в данном случае даст 6750 * 2 / 1000 → 14 мс.

   После того, как период скорректирован, система исполнения считает, что контроллер никогда не успевает уложиться в заданный период, и поэтому оба верхних индикатора (RUN/ERR и APP) светятся красным цветом.
7. Задача переводится в состояние пассивного ожидания на время, вычисленное при выполнении шага 6.
8. Задача "просыпается" и переходит к шагу 1 данного алгоритма.

Если в проекте имеется более одной задачи (например, хотя бы одна циклическая и хотя бы одна ациклическая), каждая циклическая задача исполняется на контексте отдельного потока операционной системы, и в процессе выполнения производятся следующие действия:

1. Ожидается завершение ввода данных сервисной задачей.
2. Выполняется вызов корневой программной единицы задачи.

   Если в пользовательском коде имеется бесконечный (или почти бесконечный) цикл, задача зависнет, однако система не утратит работоспособность – все циклические задачи более высокого приоритета, чем текущая, обработчик системного события *OnTimer* и все ациклические задачи продолжат работу до тех пор, пока сервисная задача по истечении примерно 30 с не обнаружит, что счетчик циклов зависшей задачи не изменился, и не переведет систему в безопасный режим по нехватке вычислительных ресурсов (7 "миганий" индикатора APP красным цветом).

   Если в пользовательском коде имеется деление на 0 целочисленных операндов, возникнет исключение DIVISION BY ZERO, и контроллер перейдет в безопасный режим с соответствующей индикацией (3 "мигания" индикатора APP красным цветом).

   Если в пользовательском коде имеется деление на 0 операндов с плавающей точкой, возникнет исключение FPU EMULATOR, и контроллер перейдет в безопасный режим с соответствующей индикацией (5 "миганий" индикатора APP красным цветом).

   Если в пользовательском коде имеется ошибка кодогенератора CoDeSys, с высокой вероятностью возникнет исключение MISALIGNED CODE или INVALID OPCODE, и контроллер перейдет в безопасный режим с соответствующей индикацией (2 "мигания" индикатора APP красным цветом в первом случае, и 4 "мигания" – во втором).

3. При успешном завершении работы корневой программной единицы задачи выполняется запись во все выходные порты задачи. Во время записи в порты канал вывода блокирован (объектом синхронизации типа "критическая секция") для чтения со стороны связанных с ним входных портов других участников обмена данными.
4. Вычисляется суммарное время ввода-вывода и исполнения текущего цикла.
5. Вычисляется время в миллисекундах, в течение которого данная задача должна находится в состоянии пассивного ожидания ("спать"), отдав процессор другим задачам. Если оказывается, что время исполнения и ввода-вывода на текущем цикле превысило период цикла, задача будет "спать" до начала ближайшего следующего цикла.
6. Увеличивается счетчик циклов данной циклической задачи. Если время исполнения и ввода-вывода на текущем цикле превысило заданный период цикла, увеличивается счетчик запаздываний данной задачи.
7. Задача переводится в состояние пассивного ожидания на время, вычисленное при выполнении шага 4.
8. Задача "просыпается" и переходит к шагу 1 данного алгоритма.

Если имеется несколько циклических задач с разными приоритетами, управление в первую очередь получает задача с наибольшим по значению приоритетом. Как только она переводится в состояние пассивного ожидания (шаг 6), управление получает задача с приоритетом ниже данной на 1 и т.д.

Если имеются несколько циклических задач с одинаковыми значениями приоритетов, задачи будут выстроены операционной системой в цепочку: сначала будет полностью выполнен цикл задачи, которая находится выше остальных в списке задач в окне ресурса **Tasks Configuration**, а затем остальные задачи. Порядок задач в цепочке будет меняться во время работы контроллера в соответствии с различиями в их периодах и продолжительности выполнения ассоциированных с ними программ.

4.2.4.3. Исполнение ациклических задач

Ациклические задачи исполняется на контексте потока сервисной задачи, когда назначенные для них переменные-события меняют свои значения с FALSE на TRUE. Приоритет сервисной задачи выше приоритетов всех циклических задач, поэтому ациклические задачи всегда вытесняют циклические. Алгоритм функционирования сервисной задачи, которая управляет ациклическими задачами, представлен на рис. 15.

Во время конфигурирования ациклические задачи помещаются в приоритетную очередь – задача с наибольшим приоритетом располагается ближе остальных к началу очереди, а при равенстве приоритетов, ближе к началу очереди располагается задача с меньшим порядковым номером. Перед началом функционирования сразу после конфигурирования и связывания сервисная задача считывает начальные значения переменных-событий всех ациклических задач. Управление ациклическими задачами осуществляется по следующему алгоритму:

1. Вызывается функция-обработчик системного события *OnTimer*, если она была установлена пользователем в окне ресурса **Tasks Configuration–System events** приложения.
2. Если очередь ациклических задач пуста, просмотр очереди завершается и осуществляется переход к предпоследнему шагу данного алгоритма. Если очередь не пуста, из ее "головы" извлекается находящаяся там ациклическая задача.
3. Считывается значение переменной-события извлеченной задачи. Если переменная-событие находится в сегменте входных данных, ее значение считывается по соответствующему адресу из образа процесса. Если же переменная-событие расположена сегменте выходных данных или в сегменте внутренних переменных приложения, то значение переменной считывается непосредственно из того сегмента приложения. Считанное значение запоминается в объекте-задаче.
4. Если значение переменной-события, считанное на предыдущем цикле просмотра очереди ациклических задач, было равным FALSE (0), а считанное на текущем цикле просмотра – TRUE (1), то считается, что произошло событие, по которому должна быть вызвана корневая программная единица данной задачи. В противном случае осуществляется переход к шагу 2 настоящего алгоритма.

5. Выполняется последовательное чтение всех входных портов задачи. Во время чтения портов канал ввода блокирован (объектом синхронизации типа "критическая секция") для записи со стороны связанных с ним выходных портов других участников обмена данными.
6. Выполняется вызов корневой программной единицы задачи.

   Если в пользовательском коде имеется бесконечный (или почти бесконечный) цикл, задача зависнет и система примерно на 40 с утратит работоспособность, после чего контроллер будет переведен в безопасный режим по зависанию сервисной задачи (7 "миганий" индикатора APP зеленым цветом).

   Если в пользовательском коде имеется деление на 0 целочисленных операндов, возникнет исключение DIVISION BY ZERO, и контроллер перейдет в безопасный режим с соответствующей индикацией (3 "мигания" индикатора APP красным цветом).

   Если в пользовательском коде имеется деление на 0 операндов с плавающей точкой, возникнет исключение FPU EMULATOR, и контроллер перейдет в безопасный режим с соответствующей индикацией (5 "миганий" индикатора APP красным цветом).

   Если в пользовательском коде имеется ошибка кодогенератора CoDeSys, с высокой вероятностью возникнет исключение MISALIGNED CODE или INVALID OPCODE, и контроллер перейдет в безопасный режим с соответствующей индикацией (2 "мигания" индикатора APP красным цветом в первом случае, и 4 "мигания" – во втором).
7. При успешном завершении работы корневой программной единицы задачи выполняется последовательная запись во все выходные порты задачи. Во время записи в порты канал вывода блокирован (объектом синхронизации типа "критическая секция") для чтения со стороны связанных с ним входных портов других участников обмена данными.
8. Осуществляется переход к шагу 2 настоящего алгоритма
9. С периодом 1 с выполняется диагностика исполнения циклических задач, в том числе:

   суммируются общие количества циклов и запаздываний всех циклических задач, полученные числа выводятся в диагностические каналы *CPM70x–Diagnostics–Application–CyclesCounter* и *CPM70x–Diagnostics–Application–OverrunsCounter*;

   формируется маска состояния циклических задач (0 – неактивна; 1 – активна и успевает; 2 – активна и иногда не успевает; 3 – активна и никогда не успевает) и выводится в диагностический канал *CPM70x–Diagnostics–SystemStatus*;

   с периодом 30 с принимается решение о переводе контроллера в безопасный режим, если хотя бы одна из циклических задач не увеличила свой счетчик циклов;

   в зависимости от выявленного состояния циклических задач формируется индикация светодиодов RUN/ERR и APP.
10. Сервисная задача переводится в состояние пассивного ожидания на время, заданное параметром *CPM70x...Controller:SampleRate*.
11. По прошествии времени *CPM70x...Controller:SampleRate* сервисная задача "просыпается" и переходит к шагу 1 настоящего алгоритма.

ВНИМАНИЕ!

Если в приложении имеется хотя бы одна циклическая задача, не возлагайте длительную вычислительную работу на функцию-обработчик системного события *OnTimer* и на программные единицы, вызываемые из ациклических задач. В противном случае циклическим задачам может не хватить вычислительных ресурсов, и контроллер перейдет в безопасный режим (7 "миганий" красным)

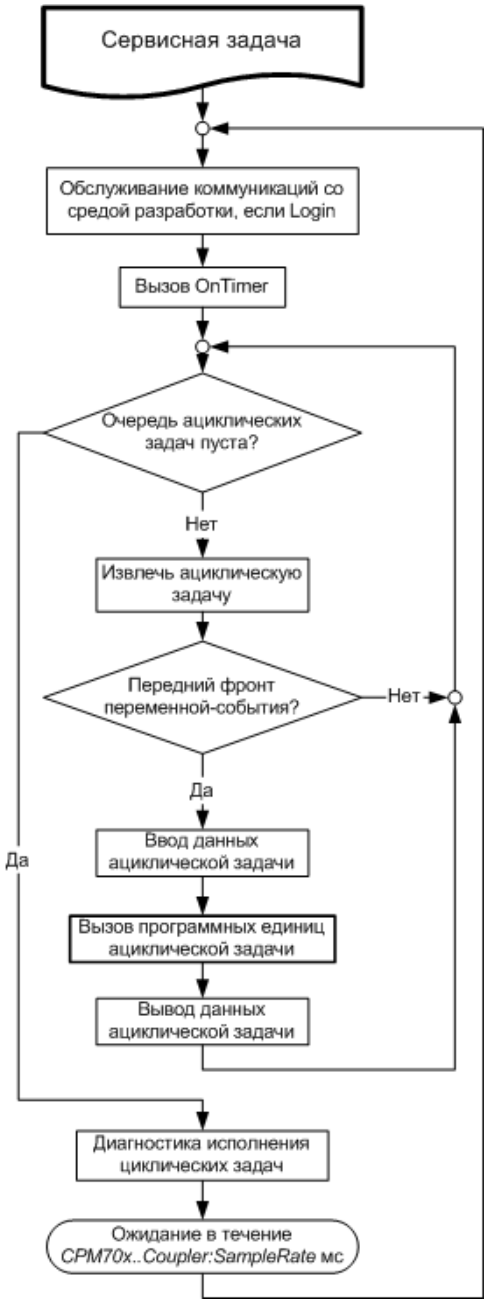


Рис. 15. Алгоритм функционирования сервисной задачи

4.2.4.4. Вызов обработчиков системных событий

Перечень системных событий, поддерживаемых адаптированной средой исполнения CoDeSys, приведен в табл. 5.

Обработчиком системного события является функция (программная единица типа *FUNCTION*) со следующей сигнатурой:

```
FUNCTION SystemEventFunctionName : DWORD
   VAR_INPUT
         eventType : DWORD;
   END_VAR
   (* операции *)
END_FUNCTION
```

Таблица 5

Поддерживаемые системные события Tasks Configuration–System events		
Тип события	**Значение входного параметра**	**Вызов**
OnStart	**F_EVENT_START (1)**	▪ Перед первым циклом сервисной задачи после события OnInit; ▪ По команде **Online–Run**, выполняемой после команды **Online–Stop**
OnStop	**F_EVENT_STOP (2)**	По команде **Online–Stop**
BeforeReset	**F_EVENT_BEFORE_RESET (3)**	Непосредственно перед перезапуском контроллера: ▪ по командам **Online–Reset**; **Online–Reset (cold)**; **Online–Reset (original)**; ▪ по окончании загрузки новой версии системного ПО контроллера после события OnProgramChange
OnLogin	**F_EVENT_LOGIN (501)**	▪ по команде **Online–Login**; ▪ перед запуском вновь загруженного приложения перед событием OnInit
OnLogout	**F_EVENT_LOGOUT (1503)**	▪ по команде **Online–Logout**, выполняемой после команды **Online–Login**; ▪ перед началом процесса конфигурирования вновь загруженной программы, описанного в п. 4.2.3, до вызова обработчика OnProgramChange; ▪ при разрыве связи со средой разработки, находившейся в состоянии Login на данном контроллере;
OnProgramChange	**F_EVENT_ONLINE_CHANGE (33)**	▪ перед началом процесса конфигурирования вновь загруженной программы, описанного в п. 4.2.3 ▪ по окончании загрузки новой версии системного ПО контроллера перед перезапуском контроллера
OnDownload	**F_EVENT_BEFORE_DOWNLOAD (34)**	Перед началом загрузки из среды разработки секции кода нового приложения
OnInit	**F_EVENT_ON_INIT (1501)**	Перед запуском приложения. **Вызов функции F_IecTasks_linkVariables() из внешней библиотеки FastwelTasksExchange.lib допускается делать только при обработке данного события!**
OnPowerOn	**F_EVENT_POWER_ON (1502)**	Перед запуском приложения перед OnInit в случае, если контроллер стартует по включению питания
OnTimer	**F_EVENT_TIMER (30)**	Каждый цикл сервисной задачи с периодом *CPM70x...Controller–SampleRate*

ВНИМАНИЕ!

Категорически запрещается изменять интерфейс функции обработки системных событий – добавлять локальные переменные, изменять тип возвращаемого значения или тип и количество входных параметров!

При нарушении приведенного интерфейса обработчика системного события в лучшем случае, функция просто не будет делать то, что запланировал ее автор.

В хорошем случае – контроллер сразу перейдет в безопасный режим по фатальной ошибке (MISALIGNED CODE: 2 "мигания" красным или INVALID OPCODE: 4 "мигания" красным).

В плохом случае – некоторое (заранее неизвестное) время контроллер проработает без заметных отклонений от нормы, а потом неожиданно перейдет в безопасный режим по фатальной ошибке.

В худшем случае – произойдет то же, что и в плохом случае, но код функции обработки системного события с нарушенным интерфейсом будет портить стек среды исполнения (сервисной задачи), что может привести к неправильной работе ациклических задач и, в конце концов, к перезапуску в безопасном режиме по фатальной ошибке.

В наихудшем случае – произойдет то же, что и в худшем, но контроллер никогда не перейдет в безопасный режим по фатальной ошибке, однако иногда, в трудноуловимые моменты, будут неправильно работать ациклические задачи

Это НЕ особенность адаптации среды исполнения CoDeSys! Это следствие соглашения о вызовах функций, используемого кодогенератором CoDeSys!

При разработке приложений с обработкой системных событий настоятельно рекомендуется в качестве обработчиков системных событий использовать одну и ту же функцию с приведенным фиксированным интерфейсом, которая анализирует значение входного параметра и, в зависимости от типа события, вызывает другие функции, выполняющие требуемую работу по обработке системных событий. Интерфейс функций, вызываемых из функций-обработчиков системных событий, может быть любым.

Пример диспетчеризации системных событий:

```
FUNCTION SystemEventsDispatcher : DWORD
   VAR_INPUT
         eventType : DWORD;
   END_VAR
   CASE eventType OF
   F_EVENT_TIMER:
         ActualEventTimerHandler();
   F_EVENT_ONLINE_CHANGE:
         SaveMyPersistentVariables();
   F_EVENT_ON_INIT:
         LinkTasks();
         LoadMyPersistentVeariables();
   F_EVENT_POWER_ON:
         LoadMyRetainVariables();
   ELSE
         (* ничего не делаем *);
   END_CASE;
END_FUNCTION
```

Обработчики событий *BeforeReset* и *OnProgramChange* вызываются в момент, когда все задачи приложения закончили выполнения своих корневых программных единиц. После вызова обработчика *BeforeReset* произойдет аппаратный перезапуск контроллера, до которого ни одна задача приложения ни разу не вызовет свою корневую программную единицу.

После последовательного вызова пары обработчиков системных событий *OnLogout* и *OnProgramChange* непосредственно перед началом процесса конфигурирования вновь загруженного приложения ни одна задача обновляемого/заменяемого приложения ни разу не вызовет свою корневую программную единицу.

После последовательного вызова пары обработчиков системных событий *OnProgramChange* и *BeforeReset* по завершении загрузки в контроллер новой версии системного программного обеспечения ни одна задача текущего приложения ни разу не вызовет свою корневую программную единицу, после чего произойдет аппаратный перезапуск контроллера.

Событие *OnProgramChange* может использоваться для сохранения значений некоторых переменных, которые должны использоваться вновь загруженной программой. Кроме того, настоятельно рекомендуется использовать событие *OnProgramChange* для закрытия всех открытых файлов и коммуникационного порта COM1 (если они были открыты старым приложением).

Событие *OnInit* может служить для реализации в пользовательском приложении однократных инициализирующих действия над каким-либо переменными приложения, для открытия файлов и т.п., а также для связывания задач приложения по переменным в соответствии с п. 4.2.4.5 перед запуском задач приложения. Если активизирована функция горячего обновления приложения (параметр *Fastwel I/O System Configuration:HotUpdateDisabled* имеет значение *No*), необходимо соблюдать особую осторожность при открытии файлов (при помощи функции *FwSysFileOpen()* из библиотеки *FastwelSysLibFile.lib*) и коммуникационного порта COM1 (при помощи функции *FwSysComOpen()* из библиотеки *FastwelSysLibCom.lib*) – если в процессе обновления окажется, что структуры данных, размеры образа процесса и связи задач с образом процесса не изменились, то может произойти повторное открытие одних и тех же файлов и неудачное открытие ранее открытого порта COM1. Поэтому наиболее правильным способом борьбы с такой ситуацией является закрытие всех файлов и порта во время обработки события *OnProgramChange*.

Пример:

```
(* диспетчер системных событий *)
FUNCTION SystemEventsDispatcher : DWORD
   VAR_INPUT
         eventType : DWORD;
   END_VAR
   CASE eventType OF
   F_EVENT_ONLINE_CHANGE:
         (* сохраняем какие-нибудь переменные *)
         SaveMyPersistentVariables();
         (* закрываем все открытые хэндлы *)
         CloseAllHandles();
   F_EVENT_ON_INIT:
         (* загружаем сохраненные переменные *)
         LoadMyPersistentVeariables();
         (* связываем задачи *)
```

```
        LinkTasks();
    F_EVENT_POWER_ON:
        (* загружаем какие-нибудь энергонезависимые переменные *)
        LoadMyRetainVariables();
    ELSE
        (* ничего не делаем *);
    END_CASE;
END_FUNCTION
```

4.2.4.5. Обмен данными между задачами

Традиционно при разработке приложений для программируемых логических контроллеров применяется подход, при котором все входные переменные (их значения получаются приложением от входных каналов модулей ввода-вывода и входящих коммуникационных объектов внешней сети) и большинство внутренних переменных алгоритма, реализуемого приложением, делаются глобальными, т.е. доступными любой программной единице приложения.

Если система исполнения контроллера выполняет все программные единицы последовательно, то указанный подход вполне приемлем, особенно для небольших приложений.

При разработке приложения для контроллера с многозадачной системой исполнения использование глобальных переменных, чтение и запись которых может осуществляться в разных, параллельно исполняющихся задачах, может привести к серьезным и, в ряде случаев, трудновоспроизводимым ошибкам.

Пусть, например, некоторая переменная gMyVariable объявлена в ресурсе **Global Variables** (VAR_GLOBAL), и ее значение вычисляется в программе PRG1, исполняющейся под управлением циклической задачи Task1, для которой заданы период исполнения 10 мс и приоритет 3. Пусть значение gMyVariable используется в алгоритме, выполняемом другой программой PRG2, функционирующей под управлением другой циклической задачи Task2, имеющей период исполнения 30 мс и приоритет 2, т.е. более низкий, чем Task1. Циклограмма приложения представлена на рис. 16.

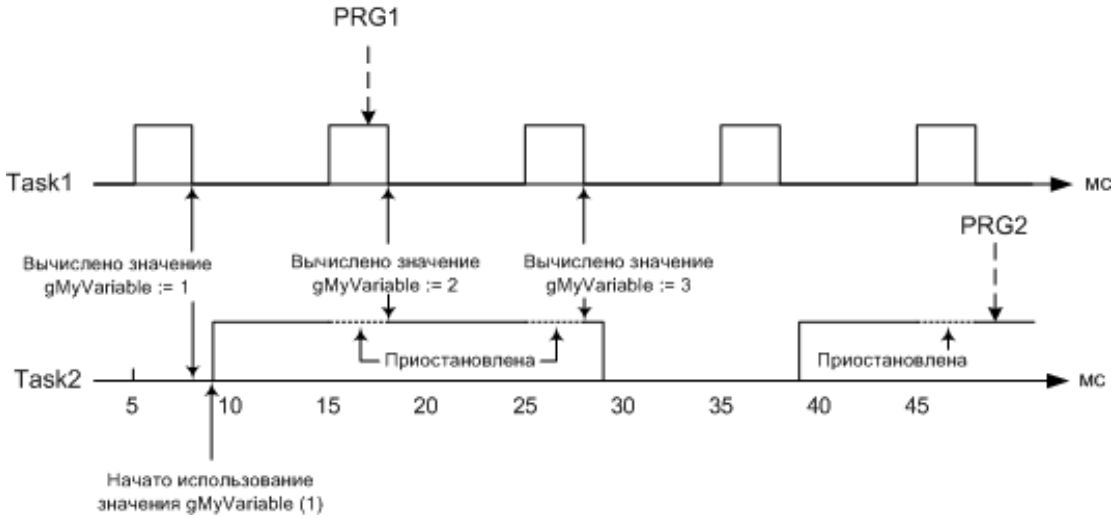


Рис. 16. Доступ к общей переменной из двух разных задач

В цикле Task1, начавшемся в момент (5 мс), PRG1 вычисляет значение gMyVariable, которое, к примеру, становится равным 1. Далее в момент, близкий к (10 мс), начинается исполнение задачи Task2, которая запускает PRG2. PRG2 использует gMyVariable в реализуемом алгоритме в течение промежутка времени между моментами (10 мс) и (30 мс), полагая, что ее значение равно 1. При наступлении момента (15 мс) более приоритетная Task1 вытесняет Task2 (приостанавливает Task2) и вызывает PRG1, которая вновь вычисляет gMyVariable, после чего управление возвращается в ту точку кода PRG2, где была вытеснена Task2. Как видно, значение gMyVariable теперь не равно 1, каковым оно было при вызове PRG2 в начале цикла Task2, а это значит, что алгоритм PRG2 скорее всего будет работать неправильно.

Если же PRG2 в процессе работы изменяет gMyVariable, то ее изменения "отменяются" всякий раз, когда задача Task1 вытесняет Task2, и PRG1 изменяет gMyVariable.

Решение данной проблемы путем копирования gMyVariable в какую-нибудь внутреннюю или входную переменную PRG2 в начале каждого цикла Task2 является корректным только для переменных длиной не более разрядности процессора, которая в контроллерах Fastwel I/O серии

CPM70x составляет 16 бит (2 байта). Переменные типов REAL, LREAL, DWORD, LINT, массивы и структуры, длина которых превышает 2 байта, невозможно скопировать атомарно – т.е. исключив возможность вытеснения во время копирования другой, более приоритетной, задачей, которая изменит копируемое значение, в результате чего во внутреннюю переменную PRG2 будет скопирована часть старого значения и часть нового, перевычисленного "вклинившейся" более приоритетной задачей.

Например, в лучшем случае при копировании переменной типа REAL или LREAL, это сразу приведет к переходу контроллера в безопасный режим, скажем, по переполнению эмулятора сопроцессора. В худшем же случае алгоритм PRG2 получит вполне корректное, с его точки зрения, значение, которое, однако, не имеет ничего общего с текущим состоянием контролируемого процесса.

Для того, чтобы пользователь был способен избежать указанных явлений при разработке многозадачных приложений, в комплект адаптации CoDeSys для Fastwel I/O входит библиотека поддержки *FastwelTasksExchange.lib*.

Функция *F_IecTasks_linkVariables()*, имеющаяся в данной библиотеке предназначена для связи переменных одинакового размера, принадлежащих программам, вызываемым из разных задач. При этом механизм межзадачного обмена данными в точности совпадает с описанным в п. 4.2.4.1:

1. Для переменной связываемой задачи, которая будет выступать в качестве источника данных, создается выходной порт.
2. Для переменной другой связываемой задачи, которая будет выступать в качестве получателя данных, создается входной порт.
3. Создается канал обмена с отдельным буфером, размер которого равен размеру связываемых переменных.
4. Выходной порт связывается с каналом в качестве источника, а входной порт – в качестве приемника данных.

В процессе работы задача, чья переменная связана с переменной другой задачи в качестве приемника данных, перед вызовом корневой программной единицы последовательно читает все свои входные порты, среди которых также оказываются и созданные при вызове *F_IecTasks_linkVariables()*. При чтении порта, когда доступ по записи к связанному с ним каналу блокирован, значение, находящееся в буфере канала, копируется в переменную-приемник данных. Задача, чья переменная связана с переменной другой задачи в качестве источника данных, после вызова корневой программной единицы последовательно пишет во все свои выходные порты, среди которых оказываются и созданные при вызове *F_IecTasks_linkVariables()*. При записи в выходной порт, когда доступ по чтению к связанному с ним каналу блокирован, значение переменной-источника данных копируется в буфер канала.

Примечание. Организовывать обмен данными описанным способом между ациклическими задачами не требуется, поскольку они исполняются синхронно относительно друг друга. Указанный способ должен применяться для связи по переменным большого размера между циклических задачами, а также между циклическими и ациклическими задачами.

Функция F_IecTasks_linkVariables() имеет следующий прототип (в синтаксисе IEC 61131-3):

```
FUNCTION F_IecTasks_linkVariables : F_LINK_RESULT
VAR_INPUT
   pSourceDescriptor : POINTER TO F_LINK_DESCRIPTOR;
   pDestinationDescriptor : POINTER TO F_LINK_DESCRIPTOR;
END_VAR
;
END_FUNCTION
```

Входные параметры:

pSourceDescriptor : POINTER TO F_LINK_DESCRIPTOR
Указатель на описатель переменной, которая будет использоваться в межзадачном обмене в качестве источника данных.
pDestinationDescriptor : POINTER TO F_LINK_DESCRIPTOR
Указатель на описатель переменной, которая будет использоваться в межзадачном обмене в качестве получателя данных.
Тип **F_LINK_DESCRIPTOR** является структурой, описывающей связываемую переменную:

```
TYPE F_LINK_DESCRIPTOR :
   STRUCT
    (*     Указатель на переменную, которую требуется связать с другой переменной.
           Пример: descr.variableAddress := ADR(SomeProgram.SomeVariable); *)
     variableAddress : DWORD;
```

```
    (*     Размер переменной (в байтах)
           Пример: descr.variableSize := SIZEOF(SomeProgram.SomeVariable); *)
      variableSize : INT;
    (*     Индекс программной единицы (POU), содержащей описываемую переменную.
           Пример: descr.pouIndex := INDEXOF(SomeProgram);       *)
      pouIndex : INT;
   END_STRUCT
END_TYPE
```

Если переменная данного типа объявляется в секции VAR функции, ее поля в момент вызова будут содержать следующие инициализирующие значения:

```
(variableAddress := 0, variableSize:= 0, pouIndex := 16#FFFF)
```

Возвращаемый результат:

F_LINK_UNCERTAIN := 0

Зарезервированное значение, которое присваивается переменной типа F_LINK_RESULT по умолчанию пока еще не выполнено никаких действий по связыванию.

F_LINK_OK := 1

Связывание выполнено успешно.

F_LINK_INVALID_SOURCE := 2

Неправильный описатель переменной-источника данных. Ситуации:

pSourceDescriptor равен 0;

адрес переменной (**pSourceDescriptor^.variableAddress**) равен 0;

переменная находится во входном (INPUT (AT%I...)) или выходном OUTPUT (AT%Q...), или флаговом FLAGS MEMORY (AT%M...) или RETAIN-сегментах;

заданный индекс программной единицы (**pSourceDescriptor^.pouIndex**) не относится к множеству индексов программных единиц, вызываемых из каких-либо циклических или ациклических задач приложения.

F_LINK_INVALID_DESTINATION := 3

Неправильный описатель переменной-получателя данных. Ситуации:

pDestinationDescriptor равен 0;

адрес переменной (**pDestinationDescriptor^.variableAddress**) равен 0;

переменная находится во входном (INPUT (AT%I...)) или выходном OUTPUT (AT%Q...), или флаговом (AT%M...) или RETAIN-сегментах;

заданный индекс программной единицы (**pDestinationDescriptor^.pouIndex**) не относится к множеству индексов программных единиц, вызываемых из каких-либо циклических или ациклических задач приложения.

F_LINK_INVALID_SOURCE_LEN := 4

Неправильная длина переменной-источника данных (0 или более размера глобальной области данных)

F_LINK_INVALID_DESTINATION_LEN := 5

Неправильная длина переменной-приемника данных (0 или более размера глобальной области данных)

F_LINK_SOURCE_DESTINATION_LEN_NOT_EQUAL := 6

Отличие длин переменных-источника и приемника данных. Они должны быть равными друг другу и не равными нулю.

F_LINK_ONE_TASK_CONNECTION_NOT_ALLOWED := 7

Попытка связать переменные, принадлежащие POU, которые вызываются из одной и той же задачи.

F_LINK_CONNECTION_ALLOWED_ON_INIT_INTARGET_ONLY := 8

Попытка вызов данной функции в месте, отличном от обработчика события OnInit, или в режиме симуляции (**Online–Simulation Mode**).

F_LINK_NOT_ENOUGH_RESOURCES := 9

Не хватило системных ресурсов для связывания.

F_LINK_INPORT_EXIST_FOR_DESTINATION := 10

Переменная, заданная в качестве получателя данных вторым параметром, уже связана с другой (или этой же) переменной-источником данных.

Пример:

```
VAR_GLOBAL
(* глобальная переменная для хранения результата связывания *)
   globalLinkResult : F_LINK_RESULT;
END_VAR

FUNCTION LinkTasks: DWORD
(*****************************************************************
Функция, вызываемая из пользовательского диспетчера системных событий по событию
OnInit.
НИКОГДА не ставьте ее непосредственно в качестве обработчика OnInit в диалоге Tasks
Configuration–System events среды разработки CoDeSys, иначе контроллер гарантированно
упадет после вызова!
*****************************************************************)
VAR_INPUT
   dwEventType : DWORD;
END_VAR
```

```
VAR
   linkResult : F_LINK_RESULT;
   sourceDesc : F_LINK_DESCRIPTOR;
   destDesc : F_LINK_DESCRIPTOR;
END_VAR
(* Тело функции *)
IF dwEventType = F_EVENT_ON_INIT THEN
(* Устанавливаем связь по переменным, хранящим уставки температуры резисторов,
между POU "PLC_PRG" и "RESISTORS_CONTROL" *)
(* Описатель источника данных *)
(* Индекс POU PLC_PRG*)
   sourceDesc.pouIndex := INDEXOF(PLC_PRG);
(* Адрес переменной-источника данных *)
   sourceDesc.variableAddress := ADR(PLC_PRG.arResistorsValue[2]);
(* Длина переменной-источника данных, два элемента массива (2-й и 3-й) *)
   sourceDesc.variableSize := 2 * SIZEOF(PLC_PRG.arResistorsValue[2]);
(* Описатель получателя данных *)
(* Индекс POU RESISTORS_CONTROL *)
   destDesc.pouIndex := INDEXOF(RESISTORS_CONTROL);
(* Адрес переменной-получателя данных *)
   destDesc.variableAddress := ADR(RESISTORS_CONTROL.arResistorsValue[2]);
(* Длина переменной-получателя данных, два элемента массива (2-й и 3-й) *)
   destDesc.variableSize := 2 * SIZEOF(PLC_PRG.arResistorsValue[2]);
   linkResult := F_IecTasks_linkVariables(ADR(sourceDesc), ADR(destDesc));

   globalLinkResult := linkResult;

   IF linkResult <> F_LINK_OK THEN
         LinkTasks := 0;
         RETURN;
   END_IF
(* Устанавливаем связь по переменным текущей температуры резисторов между POU
   "PLC_PRG" и "RESISTORS_CONTROL" *)
(* Описатель источника данных *)
(* Индекс POU RESISTORS_CONTROL *)
   sourceDesc.pouIndex := INDEXOF(RESISTORS_CONTROL);
(* Адрес переменной-источника данных *)
   sourceDesc.variableAddress := ADR(RESISTORS_CONTROL.arResistorsValue[0]);
(* Длина переменной-источника данных, два элемента массива (0-й и 1-й) *)
   sourceDesc.variableSize := 2 * SIZEOF(RESISTORS_CONTROL.arResistorsValue[0]);
(* Описатель получателя данных *)
(* Индекс POU PLC_PRG*)
   destDesc.pouIndex := INDEXOF(PLC_PRG);
(* Адрес переменной-получателя данных *)
   destDesc.variableAddress := ADR(PLC_PRG.arResistorsValue[0]);
(* Длина переменной-получателя данных, два элемента массива (0-й и 1-й) *)
   destDesc.variableSize := 2 * SIZEOF(RESISTORS_CONTROL.arResistorsValue[0]);
   linkResult := F_IecTasks_linkVariables(ADR(sourceDesc), ADR(destDesc));
(* Сохраняем результат в глобальную переменную *)
   globalLinkResult := linkResult;
   LinkTasks := 1;
ELSE
   LinkTasks := 0;
END_IF
END_FUNCTION
```

ВНИМАНИЕ!

Если по какой-либо причине невозможно или затруднительно использовать функцию *F_IecTasks_linkVariables*, не добавляйте в конфигурацию задач приложения более одной задачи или не добавляйте задач вовсе.

Если в **Task Configuration** не добавлено ни одной задачи, программа с именем *PLC_PRG* будет исполняться под управлением сервисной задачи с периодом, равным значению параметра *CPM70x ... Programmable Controller:SampleRate*, заданному в ресурсе **PLC Configuration**.

Если в **Task Configuration** добавлена только одна циклическая задача и ни одной ациклической, то программа, ассоциированная с единственной задачей, будет исполнятся под управлением высокоприоритетной сервисной задачи с периодом, заданным для задачи параметром **Properties – Interval** в свойствах задачи.

4.2.5. Диагностика

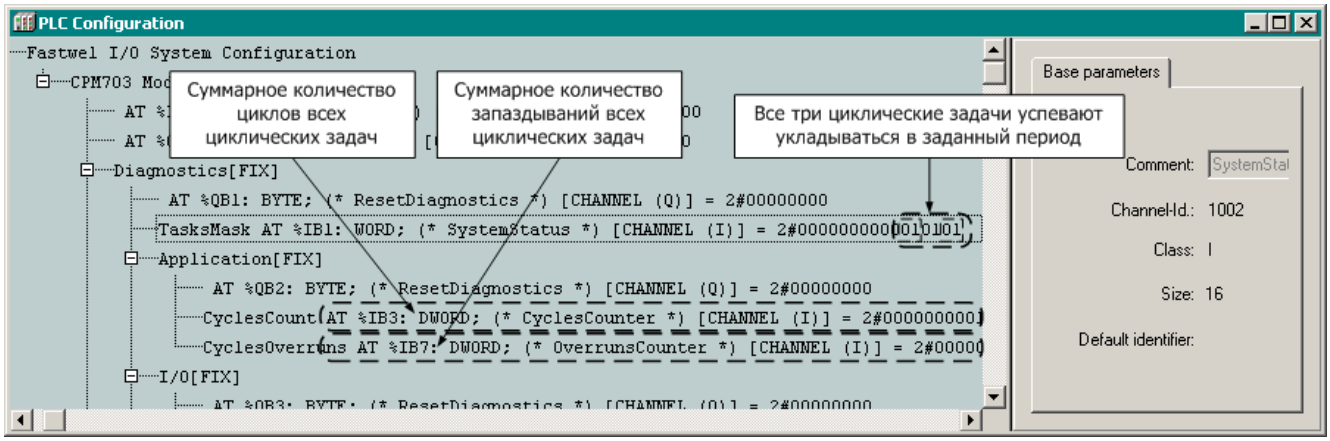


Рис. 17. Диагностические каналы среды исполнения CoDeSys

В конфигурации контроллера имеется секция *Diagnostics–Application*, в которой определены два входных канала, позволяющих приложению во время выполнения получить общее количество циклов всех циклических задач и общее количество циклов, во время которых циклические задачи не успели завершить исполнение в течение заданных периодов. Назначение каналов представлено в табл. 6.

Таблица 6

Описание секции Diagnostics–Application конфигурации контроллера узла			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
ResetDiagnostics	%QB2	BYTE	Не используется в текущей версии
CyclesCounter	%IB3	DWORD	Общее количество циклов всех циклических задач
OverrunsCounter	%IB7	DWORD	Общее количество циклов циклических задач, во время которых они не успели завершить выполнение в течение своих заданных периодов

Кроме того, секция *Diagnostics* содержит канал *SystemStatus* типа WORD, младшие 6 бит которого содержат текущий статус циклических задач в порядке их следования в списке ресурса **Tasks Configuration**: младшие два бита соответствуют задаче с наименьшим номером. Статус задачи может принимать следующие значения:

0: неактивная задача;

1: задача активна и успевает укладываться в заданный период;

2: задача активна и не всегда успевает укладываться в заданный период;

3: задача активна и никогда не успевает укладываться в заданный период.

Обновление значений перечисленных каналов в сегменте входных данных приложения происходит только в том случае, если на них ссылаются переменные приложения, которые используются в правой части хотя бы одного выражения в коде приложения. Для получения более подробной информации о принципе формирования описателей ссылок задач на образ процесса средой разработки CoDeSys обратитесь к п. 4.2.4.1.

4.3. Принцип работы сервиса ввода-вывода

4.3.1. Общие сведения

Модули ввода-вывода подключаются к внутренней шине контроллера, выполненной в соответствии со спецификацией FBUS 2.0. Сервис ввода-вывода адаптированной среды исполнения CoDeSys реализует стек протоколов FBUS 2.0.

Шина FBUS является системой последовательной передачи данных, которая предназначена для организации обмена данными реального времени и конфигурационной информацией между вычислительным устройством (контроллером узла) и устройствами (модулями) ввода-вывода в программируемых логических контроллерах и системах распределенного ввода-вывода.

Физический уровень сети представлен на рис. 18.

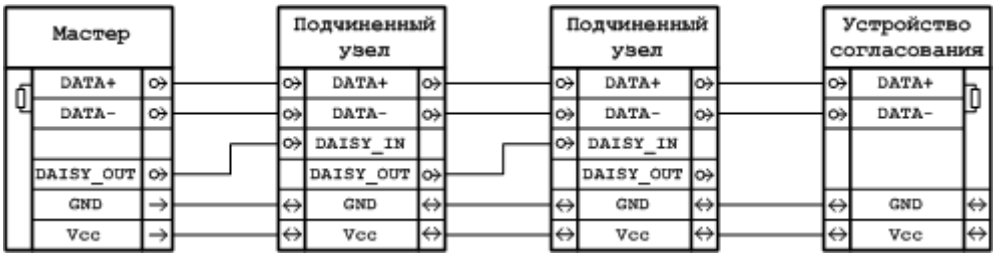


Рис. 18. Линии среды обмена данными FBUS

Таблица 7

Линия	Направление	Назначение
DATA+	Вход/выход	Дифференциальная симметричная пара, по которой происходит обмен данными между мастером и подчиненными узлами.
DATA–		
DAISY_IN	Вход	Вход подчиненного узла. Электрические параметры соответствуют ТТЛ. Активный уровень – высокий. При наличии активного уровня на данном входе подчиненный узел отвечает на запросы, адресуемые мастером узлу с неназначенным сетевым идентификатором (ID = 7Dh)
DAISY_OUT	Выход	Выход мастера и подчиненного узла. Электрические параметры соответствуют ТТЛ. Активный уровень – высокий. Предназначен для установки/сброса текущим узлом активного уровня на входе DAISY_IN следующего узла.
GND		Общий провод источника питания узлов
Vcc		Потенциальный провод источника питания узлов. При использовании соединителя типа 1 напряжение питания составляет 5 В

Обмен данными между мастером и подчиненными узлами выполняется со скоростью 2 Мбит/с. Подробная информация об остальных уровнях протокола FBUS выходит за рамки настоящего документа.

ВНИМАНИЕ!

Для организации обмена данными между приложением и модулями ввода-вывода требуется добавить описания модулей ввода-вывода в секцию *CPM70x...Controller–I/O Modules* ресурса **PLC Configuration** с точным соблюдением порядка физического подключения к внутренней шине контроллера. Например, если к контроллеру подключены модули в следующем порядке: AIM720 (самый ближний к контроллеру); DIM713; DIM717; DIM717; DIM718, то конфигурация сервиса ввода-вывода должна выглядеть, как показано на рис. 19.

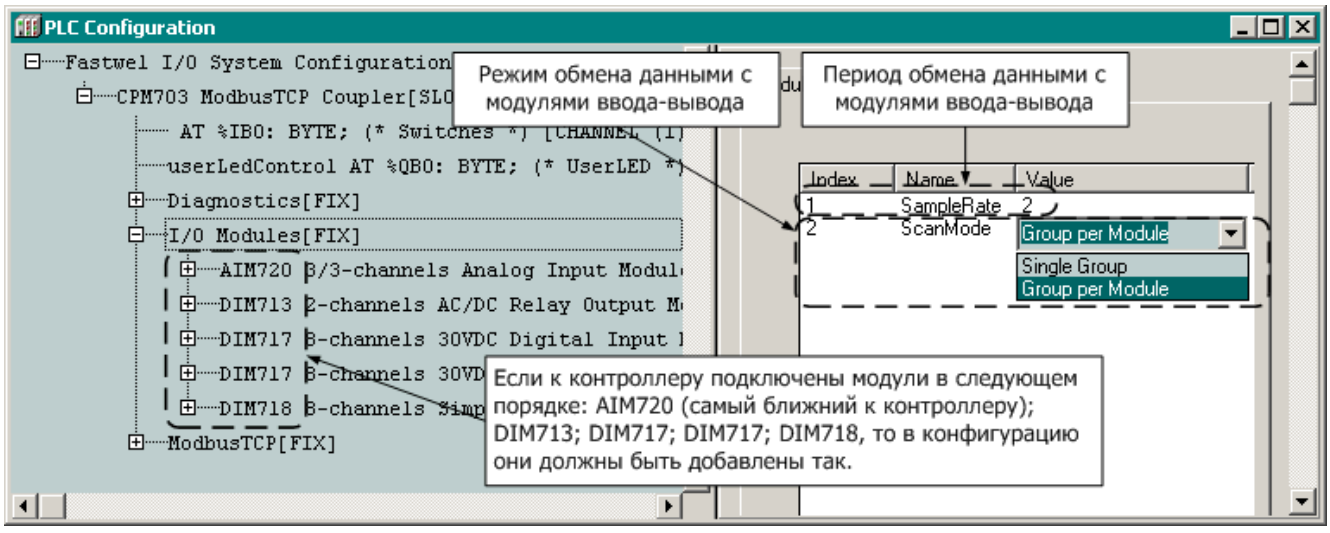


Рис. 19. Конфигурация сервиса ввода-вывода контроллера

В контроллерах Fastwel I/O обмен данными между контроллером и модулями ввода-вывода по умолчанию осуществляется в так называемом групповом режиме, при котором за одну сетевую транзакцию одновременно осуществляется запись данных для все выходные каналы и чтение данных всех входных каналов модулей ввода-вывода. Тем самым обеспечивается пропускная способность не хуже 165 кбайт/с. Для выбора группового режима обмена параметр *I/O Modules:ScanMode* должен иметь значение *Single Group*.

Кроме того, предусмотрен режим индивидуального обмена с модулями, при использовании которого для каждого модуля, добавленного в конфигурацию контроллера, создается отдельная группа ввода-вывода. Для выбора режима индивидуального обмена параметр должен иметь значение *Group per Module*.

Период обмена с модулями ввода-вывода определяемую параметром *CPM70x...Controller–I/O Modules:SampleRate*. При использовании режима индивидуального обмена пропускная способность внутренней шины существенно ухудшается за счет появления пауз длительностью от 180 до 330 мкс между обменами с соседними модулями, а также за счет передачи в групповом ответе каждого модуля дополнительных пяти байт (идентификатора мастера и поля контрольной суммы).

Сервис ввода-вывода реализует функции стека протоколов FBUS и обеспечивает выполнение следующих функций:

1. Инициализацию шины и конфигурирование модулей ввода-вывода
2. Обмен данными реального времени с модулями ввода-вывода
3. Обработку нештатных ситуаций, связанных с потерей связи с одним или несколькими модулями ввода-вывода
4. Обновление диагностической информации, доступной прикладной программе, и светодиодную индикацию.

4.3.2. Инициализация шины

Инициализация шины выполняется сервисом ввода-вывода в несколько этапов:

Этап 1:

Выполняется обнаружение всех модулей ввода-вывода, подключенных к контроллеру. Модулям последовательно назначаются сетевые идентификаторы (адреса).

Этап 2:

Для всех модулей, описания которых имеются в конфигурации контроллера, поочередно выполняются следующие действия:

1. Выясняется, совпадает ли тип обнаруженного модуля, имеющего некоторый сетевой идентификатор, с типом модуля, имеющимся в конфигурации контроллера. Сетевой идентификатор модуля в конфигурации равен его порядковому номеру в списке *I/O Modules* в окне ресурса **PLC Configuration**.
2. Если соответствие типа установлено, считывается текущий идентификатор конфигурации модуля, сохраненный в энергонезависимой памяти модуля, и сравнивается с идентификатором текущей конфигурации контроллера. Если текущий идентификатор конфигурации, считанный у модуля, не совпадает с идентификатором конфигурации контроллера, новая конфигурация передается модулю по внутренней шине.

Если при инициализации обнаружены не все модули ввода-вывода, которые имеются в конфигурации контроллера, сервис ввода-вывода, настроенный на работу в режиме *Single Group*, не будет выполнять обмен данными реального времени с модулями и со средой исполнения CoDeSys до тех пор, пока не обнаружены и не сконфигурированы все ожидаемые модули.

Если сервис ввода-вывода настроен на работу в режиме индивидуального обмена с модулями (*Group per Module*), то в случае, если при инициализации обнаружена хотя бы какая-то часть модулей в точном соответствии с порядком следования в списке *CPM703...Controller–I/O Modules*, добавленных пользователем в конфигурацию контроллера, то сервис ввода-вывода будет выполнять обмен данными только с ними, а поиск остальных выполняться не будет.

Если при инициализации не обнаружено ни одного модуля, тип которого совпадает с имеющимися в конфигурации, будет происходить периодический поиск требуемых модулей.

4.3.3. Обмен данными с модулями ввода-вывода

4.3.3.1. Групповой режим (*Single Group*)

Если инициализация внутренней шины контроллера выполнена успешно, сервис ввода-вывода приступает к обмену данными с модулями.

В групповом режиме (*Single Group*) сервис ввода-вывода передает в шину запрос, содержащий идентификатор группы, данные для выходных каналов модулей, входящих в группу, и контрольную сумму. Все модули, входящие в группу, воспринимают запрос, проверяют контрольную сумму, при необходимости подготавливают данные для выходных каналов к выдаче и начинают последовательно формировать на шине групповой ответ, причем каждый модуль группы выполняет расчет контрольной суммы группового ответа. Если у какого-либо модуля вычисленная контрольная сумма не совпала с переданной в запросе, данный модуль не участвует в формировании группового ответа и операция обмена аннулируется сервисом ввода-вывода контроллера узла.

Первый модуль в группе передает в шину идентификатор мастера шины (41h) и данные своих входных каналов. Второй модуль передает в шину данные своих входных каналов. Последний модуль, входящий в группу, передает в шину данные своих входных каналов и контрольную сумму полного группового ответа на групповой запрос. Все остальные модули, входящие в группу, проверяют собственные результаты вычисления контрольной суммы с контрольной суммой, переданной в шину последним модулем группы. Если какой-либо модуль, входящий в группу, установил несоответствие значения контрольной суммы, переданного по шине, со вычисленным значением, данный модуль передает в шину признак ошибки обмена. Сервис ввода-вывода контроллера, получив признак ошибки обмена, аннулирует результат обмена, увеличивая внутренний счетчик ошибок.

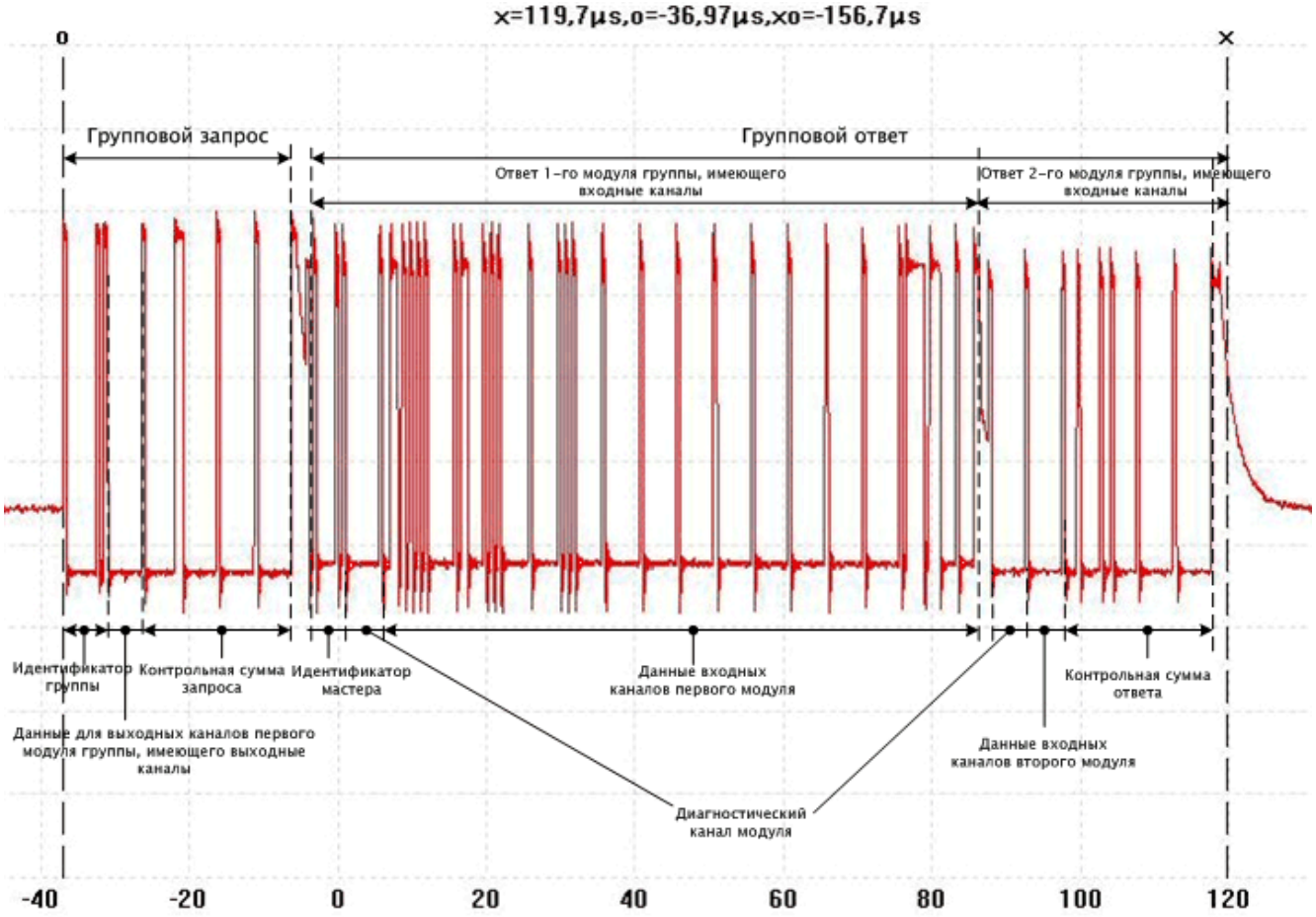


Рис. 20. Осциллограмма группового обмена с модулями AIM720 и DIM713

Осциллограмма одной групповой операции обмена данными с двумя модулями ввода-вывода показана на рис. 20.

При завершении очередной операции обмена данными с модулями ввода-вывода сервис ввода-вывода выполняет обмен данными с образом процесса.

4.3.3.2. Режим индивидуального обмена (*Group per Module*)

Если во время инициализации внутренней шины контроллера обнаружен хотя бы один из модулей ввода-вывода из имеющихся в конфигурации приложения в точном соответствии с его положением на шине, определяемом его порядковым номером в конфигурации, сервис ввода-вывода приступает к обмену данными с модулями. Если ни один модуль не обнаружен, с периодом 1 с выполняется процедура поиска и конфигурирования модулей согласно п. 4.3.2, пока хотя бы один модуль не будет обнаружен.

В режиме индивидуального обмена (*Group per Module*) для каждого модуля, имеющегося в конфигурации приложения, создается отдельная группа обмена.

В начале каждого цикла обмена сервис ввода-вывода передает в шину запрос, содержащий идентификатор первой группы (80h+номер модуля, начиная с 0), которая создана для первого модуля, данные для выходных каналов, считанные из соответствующего участка образа процесса, и контрольную сумму.

Первый модуль, приняв запрос, проверяет контрольную сумму, при необходимости подготавливает данные для выходных каналов к выдаче и передает в шину ответное сообщение, содержащее идентификатор мастера FBUS (41h), данные всех своих входных каналов и контрольную суммы длиной четыре байта.

Сервис ввода-вывода, получив ответ первого модуля (первой группы), проверяет контрольную сумму и, в случае ее правильности, сбрасывает счетчик ошибок данной группы, записывает данные в соответствующий участок входной части образа процесса. Далее таким же образом выполняется обмен с остальными модулями.

Если какой-либо модуль не ответил в течение примерно 11 мкс или ответил с ошибкой контрольной суммы, сервис ввода-вывода контроллера формирует признак ошибки и увеличивает счетчик ошибок соответствующей группы, после чего передает групповой запрос следующему модулю. Если значение счетчика ошибок какой-либо группы достигло 5 (пять неудачных обменов подряд), связь с модулем считается утраченной, в результате чего сбрасывается бит в паре диагностических каналов (*Diagnostics–I/O–IOStatus0,IOStatus1*), соответствующий номеру модуля на шине, а в участок входной части образа процесса, соответствующий виртуальному диагностическому каналу модуля, записывается значение 255 (FFh).

Осциллограмма обмена данными с двумя модулями ввода-вывода в режиме индивидуального обмена показана на рис. 21.

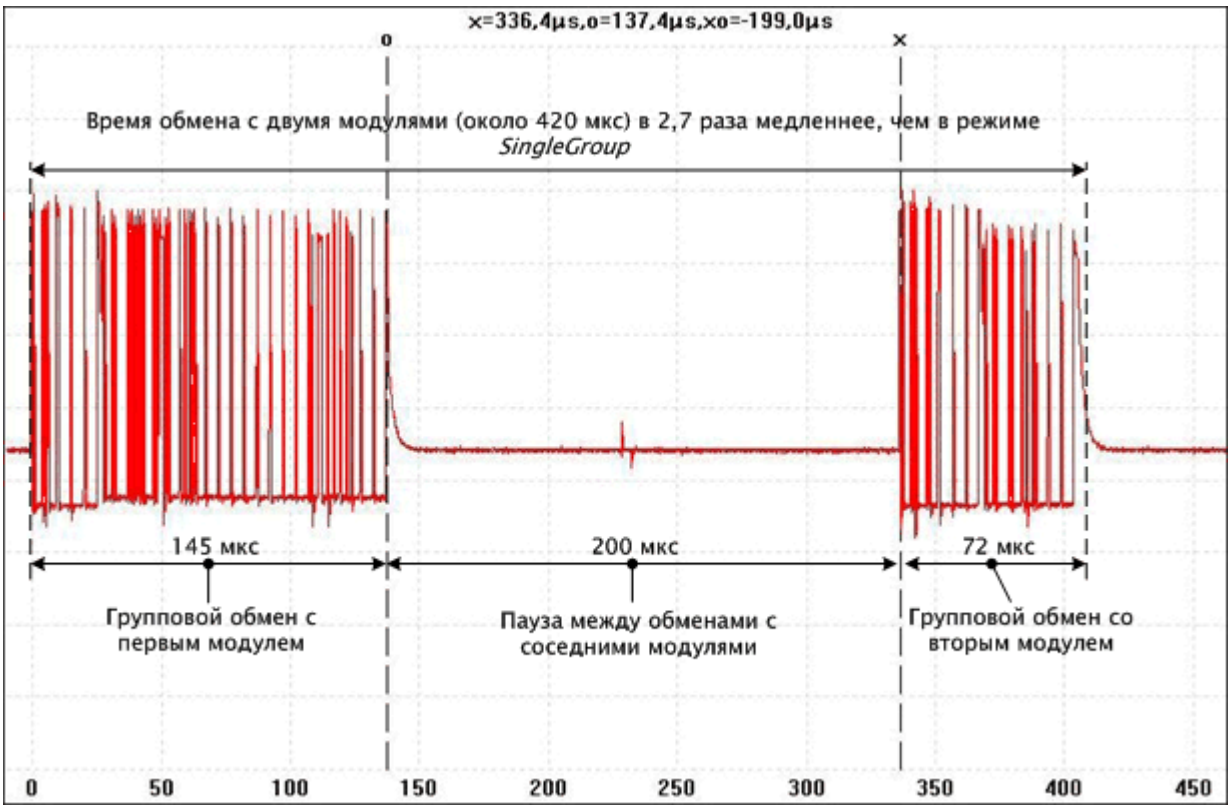


Рис. 21. Осциллограмма обмена с модулями AIM720 и DIM713 в режиме *Group per Module*

При завершении очередной операции обмена данными с модулями ввода-вывода сервис ввода-вывода выполняет обмен данными с образом процесса.

При использовании режима *Group per Module* сервис ввода-вывода потребляет существенно больше вычислительных ресурсов, чем в режиме *Single Group*. В результате производительность при исполнении программных единиц, вызываемых на контексте циклических задач может ухудшится на величину до 40-45%. В качестве эмпирического правила выбора значения периода обмена с модулями

рекомендуется следующее соотношение: добавление одного модуля в конфигурацию контроллера должно сопровождаться увеличением периода обмена, как минимум, на 1 мс.

4.3.4. Обработка нештатных ситуаций

4.3.4.1. Ошибка инициализации при запуске контроллера

Если в процессе инициализации шины сервис ввода-вывода, настроенный на работу в режиме *Single Group*, обнаружил и сконфигурировал не все модули ввода-вывода, описанные в конфигурации контроллера, обмен данными по внутренней шине не выполняется, а осуществляется поиск отсутствующих модулей.

4.3.4.2. Потеря связи с модулями ввода-вывода в процессе работы

Режим *I/O Modules– ScanMode:Single Group*

Если в процессе обмена данными с модулями ввода-вывода пять операций обмена подряд завершились неудачно, сервис, настроенный на работу в режиме *Single Group*, прекращает обмен данными и запускает специальную процедуру обнаружения и повторной инициализации модулей.

Данная процедура выполняет поиск модулей, с которыми утрачена связь. При обнаружении очередного модуля, с которым была утрачена связь, выполняется проверка совпадения идентификатора конфигурации модуля с текущим идентификатором конфигурации, имеющимся у сервиса ввода-вывода. При несовпадении принимается решение о том, что произошла замена модуля без выключения питания контроллера, и в модуль загружаются параметры конфигурации.

Обмен данными реального времени по внутренней шине контроллера возобновляется тогда, и только тогда, когда восстановлена связь со всеми модулями ввода-вывода, описания которых имеются в конфигурации прикладной программы контроллера.

Режим *I/O Modules–ScanMode: Group per Module*

При потере связи с каким-либо модулем обмен данными с остальными отвечающими модулями продолжается. Процедура обнаружения и повторной инициализации модулей не выполняется.

4.3.5. Диагностика

4.3.5.1. Индикация

Индикация функционирования сервиса ввода-вывода осуществляется при помощи светодиодного индикатора I/O (третий сверху) следующим образом:

зеленый цвет (непрерывно) – все модули ввода-вывода, определенные в конфигурации проекта, обнаружены и успешно сконфигурированы. Обмен с модулями ввода-вывода, подключенными к контроллеру, успевает завершиться за время, равное значению параметра *SampleRate* в конфигурации сервиса ввода-вывода контроллера (**Resources–PLC Configuration**–*CPM70x ... Controller–I/O Modules*). В случае ускоренного преобразования проекта к новой версии значение данного параметра в проекте будет равным 0, поэтому в качестве истинного значения периода сервис ввода-вывода будет использовать период, определенный параметром *SampleRate* в конфигурации контроллера (**Resources–PLC Configuration**–*CPM70x ... Controller*);

зеленый цвет (прерывисто) – обмен с модулями ввода-вывода, подключенными к контроллеру, не может быть выполнен за время, заданное в конфигурации контроллера (**Resources–PLC Configuration**–*CPM70x ... Controller–I/O Modules:SampleRate*), в связи с чем в режиме *Single Group* используется расчетное минимально достижимое время обмена данными со всеми модулями при загрузке внутренней шины около 50%. В режиме *Group per Module* в качестве расчетного принимается время, значение которого равно количеству модулей, переведенному в миллисекунды;

красный цвет

в момент повторного конфигурирования модулей после загрузки нового приложения

во время функционирования приложения – конфигурация модулей ввода-вывода, определенная в проекте, не совпадает с аппаратной конфигурацией модулей, подключенных к контроллеру;

отсутствие свечения – в конфигурации загруженного приложения отсутствуют описания модулей ввода-вывода.

4.3.5.2. Диагностические каналы сервиса ввода-вывода

Диагностические канала сервиса ввода-вывода предназначены для реализации контроля состояния внутренней шины в прикладной программе. Описания каналов приведены в табл. 8.

Таблица 8

Описание области Diagnostics\I/O Modules конфигурации контроллера узла			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
ResetDiagnostics	%QB3	BYTE	Не используется в текущей версии
IOStatus0	%IB11	DWORD	Битовая маска наличия модулей ввода-вывода с 1-го по 32-й, соответствующих перечисленным в конфигурации контроллера
IOStatus1	%IB15	DWORD	Битовая маска наличия модулей ввода-вывода с 33-го по 64-й, соответствующих перечисленным в конфигурации контроллера
TransactionsCount	%IB19	DWORD	Общее количество обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера
ErrorsCount	%IB23	DWORD	Количество неудачных обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера

Канал *IOStatus0* во время работы контроллера содержит битовую маску состояния первых 32-х модулей ввода-вывода, подключенных к внутренней шине контроллера.

Логическая единица в некотором бите данного канала свидетельствует о том, что модуль ввода-вывода, номер которого совпадает с номером бита (начиная с 0), описание которого имеется в конфигурации контроллера, обнаружен и сконфигурирован. Логический 0 индицирует отсутствие связи с модулем. Например, если в конфигурации программы имеются описания модулей AIM720 и DIM713, то если данные модули обнаружены и сконфигурированы сервисом ввода-вывода, значение диагностического канала *IOStatus0* будет равно 3. Если модуль AIM720 не обнаружен, то значение диагностического канала будет равно 2. Если модуль AIM720 обнаружен, а DIM713 не обнаружен, значение диагностического канала будет равно 1. Если не обнаружено ни одного модуля, значение диагностического канала будет равно 0.

Канал *IOStatus1* во время работы контроллера содержит битовую маску состояния модулей ввода-вывода с 33-го по 64-й.

В случае выхода из строя у какого-либо модуля входа Daisy In все модули, расположенные на шине правее данного модуля, не смогут быть найдены сервисом, и соответствующие битовые поля *IOStatus* будут сброшены. Однако обмен данными в режиме *Group per Module* с указанными модулями может быть продолжен с использованием предыдущих параметров конфигурации модулей.

Канал *TransactionsCount* содержит общее количество операций обмена, выполненных сервисом ввода-вывода по внутренней шине контроллера.

Значение *ErrorsCount* содержит количество неудачных операций обмена по внутренней шине.

4.3.6. Получение информации о подключенных модулях ввода-вывода

Информация о модулях ввода-вывода, подключенных к контроллеру, может быть получена при помощи приложения **Сервисная утилита FASTWEL IO**, программа установки которой находится на ftp-узле фирмы Прософт по адресу:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/Utilities/ServiceUtility

Кроме того, начиная с версии 2.64 системного программного обеспечения контроллеров и пакета адаптации CoDeSys 2.3 для Fastwel I/O, для получения информации о типах, серийных номерах и версиях микропрограмм модулей ввода-вывода можно воспользоваться командой *fiolist* браузера ПЛК:

1. В среде разработки CoDeSys 2.3 откройте проект приложения, загруженного в контроллер, или создайте новый пустой проект, после чего установите соединение с контроллером, выполнив команду **Online–Login** (**Онлайн–Подключение**). Если на экран монитора будет выведена диалоговая панель *The program has changed...(Программ была изменена!...)*, – нажмите в ней кнопку **No** (**Нет**).
2. В CoDeSys 2.3 откройте окно ресурса **PLC Browser** (**ПЛК-Браузер**) и в поле ввода команд введите:

   ```
   fiolist
   ```

   в области ответного сообщения на команду будет отображен список обнаруженных модулей ввода-вывода, подключенных к контроллеру:

   ```
   fiolist
   ```

```
FBUS modules detected: <кол-во обнаруженных модулей>
 1: DIM713      SerN(713.1933 2013/01) Firmware version(2.8)  Code(2016547633)
 2: DIM713      SerN(713.0130 2006/05) Firmware version(2.8)  Code(2016547633)
...
15: AIM727      SerN(727.0471 2011/04) Firmware version(2.12) Code(1930660215)
16: DIM714      SerN(714.9990 2014/11) Firmware version(2.7)  Code(2016547634)
```

Строка *FBUS modules detected:* содержит количество обнаруженных модулей ввода-вывода, после которой выводятся строки, описывающие обнаруженные модули, следующего формата:

***Номер*: Тип SerN(*серийный номер год/месяц*) Firmware version(*версия микропрограммы*)**

5. УКАЗАНИЯ ПО РАЗРАБОТКЕ ПРИЛОЖЕНИЙ

5.1. Общие сведения

В данном разделе вкратце рассматриваются основные операции процесса разработки проекта в среде разработки CoDeSys, включая:

1. Создание проекта для платформы Fastwel I/O с выбором требуемого типа контроллера в ресурсе **PLC Configuration**
2. Создание программ и конфигурации задач
3. Создание обработчиков системных событий
4. Трансляция приложения
5. Загрузка приложения в контроллер
6. Отладка приложения
7. Мониторинг переменных
8. Трассировка переменных
9. Обновление приложения в контроллере
10. Запись файлов в контроллер
11. Чтение файлов из контроллера
12. Обновление системного программного обеспечения контроллера

Модель взаимодействия между программами и внешним окружением реализована таким образом, что в приложении не известен тип сети, к которой подключен контроллер. В связи с этим указания по конфигурированию и использованию сервиса внешней сети разных контроллеров Fastwel I/O приведены в соответствующих руководствах, перечисленных в разделе 1 настоящего документа.

5.2. Создание проекта

Для создания проекта:

1. Запустите среду разработки CoDeSys и выберите команду **File–New**
2. В появившейся диалоговой панели **Target Settings** установите опцию **Configuration :** *Fastwel I/O System with Multitasking Runtime* и нажмите кнопку **OK**
3. В появившейся диалоговой панели **New POU** будет предложено создать программу с именем *PLC_PRG*. Нажмите кнопку **OK**, если намереваетесь иметь в проекте программу с таким именем. В противном случае измените имя создаваемой программы либо вообще откажитесь от создания программы на данном этапе проектирования.

   Если в проекте имеется программа с именем *PLC_PRG*, содержащая в своем теле хотя бы один пустой оператор (**;**) или выражение, а в ресурсе **Tasks Configuration** не создано ни одного описания циклической или ациклической задачи, то при трансляции проекта командой **Project–Build All** будет создано приложение с одной, скрытой от пользователя, циклической задачей с именем *DefaultTask*. Программа *PLC_PRG* при этом будет исполняться на контексте высокоприоритетной сервисной задачи с периодом, который определяется параметром *CPM70x...Controller:SampleRate*.

   Если в окне ресурса **Tasks Configuration** пользователем создано хотя бы одно описание циклической или ациклической задачи, то с ней потребуется явно ассоциировать программы из древовидного списка **POUs** при помощи команды контекстного меню **Append Program Call**, вызываемого правым щелчком мыши над описаниями задач в окне ресурса **Tasks Configuration**.

   Если программа *PLC_PRG* (или любая программная единица с любым именем) создана, однако на данном этапе предполагается транслировать проект без написания кода данной программной единицы, в ее теле введите единственный оператор: **;**

4. Щелкните на вкладке **Resources**, расположенной снизу в левой области главного окна среды разработки CoDeSys, после чего дважды щелкните на названии ресурса **PLC Configuration** в списке ресурсов проекта. На экране появится окно ресурса **PLC Configuration**, которое предназначено для создания и редактирования конфигурации контроллера.
5. Раскройте корневой элемент древовидного списка *Fastwel I/O System Configuration*. При создании проекта для платформы *Fastwel I/O System with Multitasking Runtime* в качестве описателя контроллера устанавливается *CPM703 MODBUS TCP Programmable Controller*.
6. Если приложение разрабатывается для другого типа контроллера, щелкните правой кнопкой мыши над описателем контроллера CPM703, выберите заголовок **Replace Element** в появившемся контекстном меню, после чего выберите строку подменю, соответствующую типу контроллера, для которого создается приложение, как показано на рис. 22.

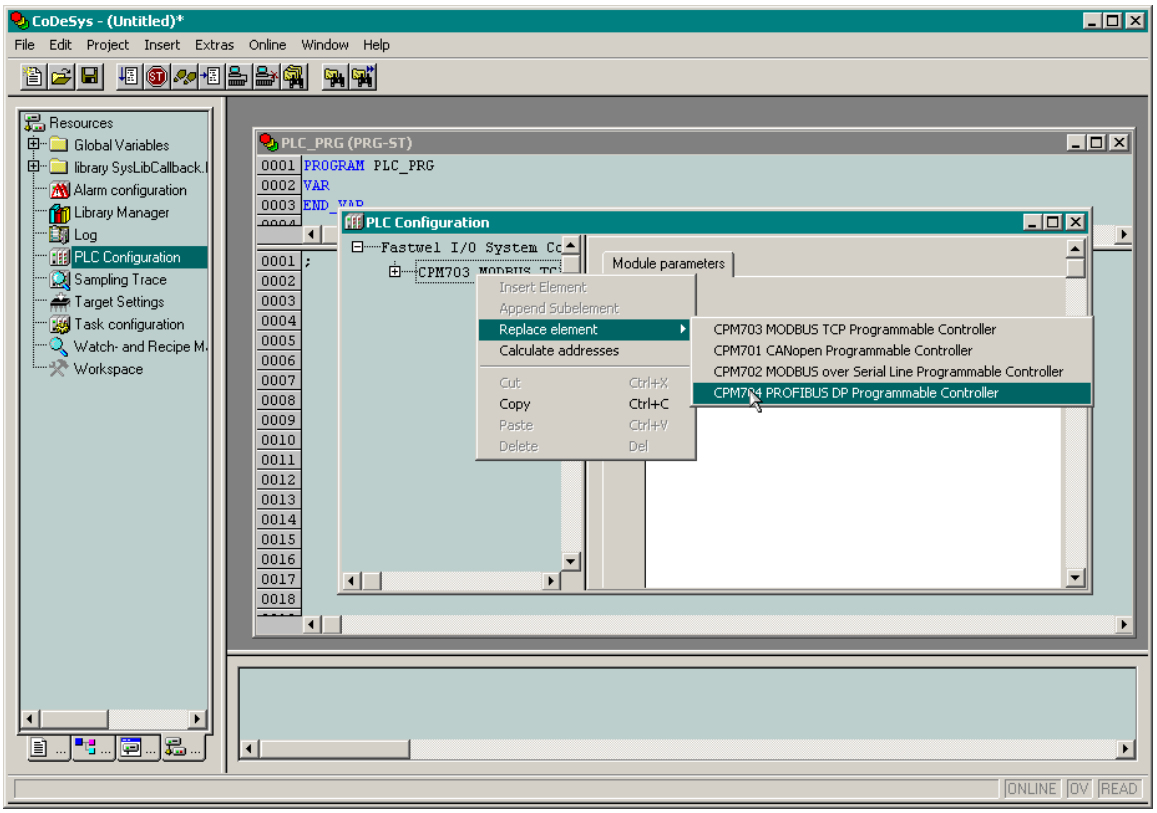


Рис. 22. Смена типа контроллера

7. Выберите **File–Save** и сохраните создаваемый проект в файле. При необходимости можно ввести информацию о проекте, включая заголовок, имя автора и т.п., выбрав команду **Project–Project Info**.
8. Для пробной трансляции проекта выберите команду **Project–Rebuild All**. Успешная трансляция будет завершена без диагностических сообщений красного цвета в панели вывода среды CoDeSys. Информация о сообщениях, выводимых средой разработки CoDeSys в панели вывода, может быть получена во встроенной справочной системе или документации на среду разработки CoDeSys.

5.3. Создание и редактирование конфигурации контроллера

Основными элементами конфигурации контроллера являются описания физических устройств и/или их подсистем, параметры физических устройств/подсистем и каналы ввода-вывода, как показано на рис. 23.

Следует обратить внимание на тот факт, что среда исполнения контроллера имеет ограничение на максимальный размер загружаемой конфигурации программы, которое составляет 65300 байт. В документе «Система ввода-вывода Fastwel I/O. Модули ввода-вывода. Руководство программиста» приведена информация о размерах отдельных элементов конфигурации контроллеров Fastwel I/O.

В конфигурации всех контроллеров Fastwel I/O имеются одинаковые элементы и параметры. Основной элемент конфигурации всех контроллеров *CPM70x...Controller* (*x...* означает последнюю цифру обозначения контроллера и название интерфейса внешней сети, например *CPM701 CANopen Programmable Controller*) имеет один параметр *SampleRate*, который определяет период цикла сервисной задачи адаптированной среды исполнения CoDeSys Fastwel I/O в диапазоне от 1 до 1000 мс (по умолчанию 10 мс). Более подробная информация сервисной задаче приведена в п. 4.2.4.1 настоящего руководства.

Входной канал *Switches* с адресом %IB0 представляет состояние переключателей контроллера узла в момент включения питания контроллера или при перезапуске. Логическая 1 в некотором разряде данного канала свидетельствует о том, что переключатель с соответствующим номером был включен при включении питания или перезапуске контроллера.

Выходной канал *UserLED* с адресом %QB0 предназначен для управления свечением индикатора USER на передней панели контроллера узла из прикладной программы. Запись 0 в данный канал приводит к прекращению свечения индикатора USER. Запись 1 в данный канал приводит к свечению индикатора USER зеленым цветом. Запись 2 в данный канал приводит к свечению индикатора USER красным цветом

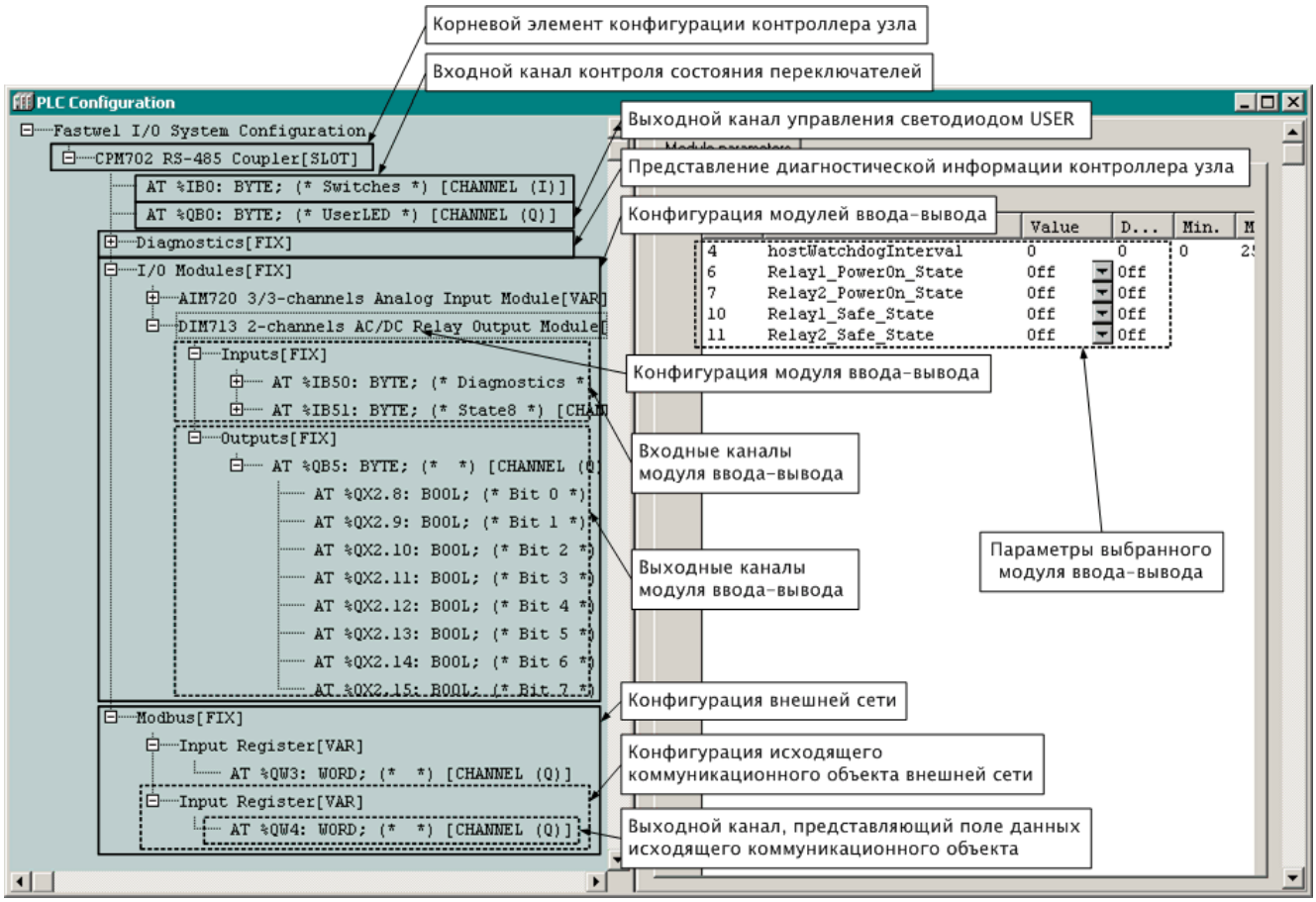


Рис. 23. Элементы конфигурации контроллера

Описание остальных дочерних элементов элемента *CPM70x... Controller–Diagnostics* приведено в табл. 9.

Элемент *CPM70x... Controller –I/O Modules* является списком, который предназначен для добавления в конфигурацию описаний модулей ввода-вывода. Модули добавляются по командам контекстного меню, вызываемым как над самим элементом *I/O Modules* (**Append Subelement**), так и над описаниями самих модулей (**Insert Subelement**).

Элемент *CPM70x... Controller–<network type>* является списком, который предназначен для добавления в конфигурацию описаний коммуникационных объектов внешней сети. Более подробная информация о конфигурации внешней сети приведена в руководствах по конфигурированию и программированию сетевых средств (см. раздел 1).

Обратите внимание:

1. При добавлении и удалении описаний модулей ввода-вывода и коммуникационных объектов внешней сети изменяется структура адресного пространства образа процесса, в результате чего может потребоваться отредактировать сдвинувшиеся адреса непосредственно представляемых переменных приложения, ссылающиеся на образ процесса или заданные в ресурсе **Variables Configuration**. Однако для отдельных каналов возможно задание символических имен, что позволяет частично сгладить проблему.
2. После удаления описания модуля ввода-вывода или коммуникационного объекта среда разработки CoDeSys не всегда пересчитывает адреса каналов и размеры образа процесса, что может привести к переходу контроллера в безопасный режим после загрузки программы. В связи с этим после удаления какого-либо объекта перед трансляцией и загрузкой программы в контроллер выполняйте команду **Calculate Addresses** в контекстном меню над элементом конфигурации *CPM70x... Controller*.

Таблица 9

Описание области Diagnostics конфигурации контроллера узла			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
ResetDiagnostics	%QB1	BYTE	Не используется в текущей версии
SystemStatus	%IB1	WORD	см. п. 4.2.5
Application	Диагностическая информация об исполнении прикладной программы		
ResetDiagnostics	%QB2	BYTE	Не используется в текущей версии
CyclesCounter	%IB3	DWORD	см. п. 4.2.5
OverrunsCounter	%IB7	DWORD	см. п. 4.2.5
I/O	Диагностическая информация подсистемы ввода-вывода		
ResetDiagnostics	%QB3	BYTE	Не используется в текущей версии
IOStatus0	%IB11	DWORD	Битовая маска наличия модулей ввода-вывода с 1-го по 32-й, соответствующих перечисленным в конфигурации контроллера
IOStatus1	%IB15	DWORD	Битовая маска наличия модулей ввода-вывода с 33-го по 64-й, соответствующих перечисленным в конфигурации контроллера
TransactionsCount	%IB19	DWORD	Общее количество обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера
ErrorsCount	%IB23	DWORD	Количество неудачных обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера
Network[1]			
ResetDiagnostics	%QB4	BYTE	Не используется в текущей версии
HostSilenceTimer	%IB27	WORD	Не используется в текущей версии
OperationsCount	%IB29	DWORD	Количество принятых входящих сообщений
ErrorsCount	%IB33	DWORD	Количество входящих сообщений, при приеме которых обнаружены ошибки

5.4. Создание программных единиц и задач

Подробная информация об архитектуре и принципах работы адаптированной среды исполнения CoDeSys приведена в п. 4.2 настоящего руководства.

Создание и редактирование программных единиц IEC 61131-3 выполняется в соответствии с указаниями эксплуатационной документации на среду разработки CoDeSys.

Максимальное количество программных единиц, поддерживаемых адаптированной средой исполнения CoDeSys для Fastwel I/O составляет 1024. Обратите внимание, что для каждой задачи, которая добавлена в проект, и к которой "прикреплены" одна и более программ, создается одна скрытая корневая программная единица, из которой прикрепленные программы вызываются друг за другом в порядке следования в списке принадлежности к задаче.

Максимальный размер секции кода, генерируемого CoDeSys, для платформы Fastwel I/O составляет 65300 байт, из которых от 5 до 10% приходится на служебную информацию. Если после трансляции программы оказалось, что максимальный размер секции кода превышен:

1. Выберите команду меню **Project–Options**, щелкните на элементе *Build* в списке диалоговой панели **Options**, после чего снимите флажок с опции **Debugging**. В результате после трансляции проекта пошаговая отладка станет недоступной, однако размер сгенерированного кода может уменьшиться на 5-10% (иногда более).

[1] Описание структуры области диагностики сети контроллера CPM704 приведено в документе " CPM704. Контроллер узла сети PROFIBUS DP-V1. Руководство по конфигурированию и программированию сетевых средств ".

2. В ресурсе **Library Manager** удалите библиотеки, функции которых не используются в проекте. Не удаляйте библиотеку SysLibCallback.lib, она автоматически подключается средой разработки CoDeSys для интеграции с подсистемой генерации системных событий программного обеспечения контроллера.

Адаптированной среда исполнения CoDeSys для Fastwel I/O поддерживает до 3-х циклических и до 16-ти ациклических задач.

Если в проекте не определено ни одной или присутствует только одна, и при том циклическая, задача, исполнение приложения осуществляется на контексте высокоприоритетной сервисной задачи.

Если пользователь не добавил ни одной задачи в ресурс **Task Configuration**, то программа PLC_PRG будет исполняться под управлением высокоприоритетной сервисной задачи с периодом, заданным в ресурсе **PLC Configuration** для параметра *CPM70x ... Programmable Controller:Sample Rate*.

Если пользователь добавил в ресурс **Task Configuration** только одну циклическую задачу и ни одной ациклической, то программа, ассоциированная с единственной задачей, будет исполнятся под управлением высокоприоритетной сервисной задачи с периодом, заданным для задачи параметром **Properties – Interval** в свойствах задачи.

При создании нескольких циклических задач назначение им приоритетов должно подчиняться следующему правилу: "короткие" по времени цикла задачи должны иметь больший приоритет, чем "длинные".

При создании ациклических задач следует помнить, что длительные операции и циклы под их управлением недопустимы, поскольку имеющимся в приложении циклическим задачам может не хватить процессорного времени. Однако допустимо создание приложения, которое состоит только из ациклических задач, управляемых обработчиком системного события *OnTimer*. В этом случае функция-обработчик *OnTimer*, вызываемая с периодом *CPM70x...Controller:SampleRate* (в конфигурации контроллера) может включать (ставить в TRUE) одну или несколько глобальных переменных типа BOOL, а программы, выполняемые на контексте ациклических задач, для которых эти переменные являются источником события, могут в конце своих циклов сбрасывать их в FALSE. В итоге все программные единицы будут исполняться на контексте одного высокоприоритетного потока операционной системы не вытесняя друг друга в порядке следования своих задач в списке ресурса **Tasks Configuration**.

ВНИМАНИЕ!

Если суммарное время выполнения обработчика системного события *OnTimer* и/или всех ациклических задач превышает 50% периода сервисной задачи, заданного параметром *CPM70x...Controller:SampleRate* в ресурсе **PLC Configuration**, система исполнения автоматически удваивает период сервисной задачи.

При наличии в приложении единственной циклической задачи, добавленной в ресурсе **Tasks Configuration**, и исполняемой под управлением сервисной задачи, в случае, если время выполнения программных единиц единственной циклической задачи превышает 50% ее периода, период единственной циклической задачи автоматически удваивается.

5.5. Связывание программ с окружением и ввод-вывод данных

5.5.1. Общие сведения

В настоящем подразделе описаны некоторые особенности среды разработки CoDeSys, касающиеся связывания разрабатываемых в ней программ с внешним окружением.

Как указывалось в п. 2.3.3 и в разделе 4, окружением программы являются каналы модулей ввода-вывода, входящих в состав контроллера, и коммуникационные объекты внешней сети. Указанные объекты окружения представляются образом процесса. Данный подраздел содержит рекомендации по связыванию пользовательских программ с окружением: с каналами модулей ввода-вывода и с коммуникационными объектами внешней сети.

Подробная информация о принципах формирования связей программных единиц приложения с образом процесса приведена в п. 4.2.4.

Ввод данных из участков входной области образа процесса, с которыми связаны входные переменные некоторой программы, производится перед очередным циклом задачи, под управлением которой исполняется данная программа, а вывод данных – по завершению очередного цикла задачи.

Среда разработки CoDeSys поддерживает три способа организации ссылок на образ процесса:

1. Посредством декларации входных или выходных переменных, ссылающихся на адреса в соответствующей части образа процесса, непосредственно в секции переменных программы.
2. Посредством создания символических имен для каналов ввода-вывода в **PLC Configuration**. Заданные символические имена доступны в программах, как обычные переменные.
3. Путем использования конфигурируемых переменных в ресурсе **Global Variables**–**Variable_Configuration** в секции *VAR_CONFIG*.

5.5.2. Ссылки на адреса образа процесса в декларациях входных или выходных переменных

В секции деклараций переменных программы возможно объявлять т.н. непосредственно представляемые переменные, ссылающиеся на адреса области входных или выходных данных образа процесса. Например:

```
VAR
   myIntInput AT%IB37  : INT;
   myBitInput AT%IX27.0: BOOL;
END_VAR
```

В данном случае декларируется переменная *myIntInput* типа INT, ссылающаяся на участок во входной области образа процесса со смещением 37 и длиной 2 байта (2 – размер типа INT), а также входная переменная *myBitInput* типа BOOL, которая ссылается на участок во входной области образа процесса со смещением 432 и длиной 1 бит.

При этом запись *%IB37* означает 37-й байт в области входных данных. Если среде разработки CoDeSys удается выровнять канал модуля ввода-вывода или коммуникационного объекта на слово, то его адрес будет представляться словным смещением: %IW10. Запись *%IX27.0* означает нулевой бит в 27-м слове области входных данных.

Указанный способ обеспечивает возможность отображения на образ процесса переменных непримитивных типов (STRUCT и ARRAY). Однако при этом следует помнить, что для членов структур типа BOOL будут создаваться ссылки размером не 1 бит, а 1 байт (см. п. 4.2.4), а отображение массивов типа BOOL не поддерживается.

Пусть, например, в проекте имеется структура, представляющая диагностические каналы сервиса ввода-вывода:

```
TYPE FIODiagnostics :
STRUCT
   nodes_0_31 : DWORD;
   nodes_32_63 : DWORD;
   transactionsCount : DWORD;
   errorsCount : DWORD;
END_STRUCT
END_TYPE
```

Для отображения входной переменной данного типа на область *Diagnostics–I/O* контроллера можно использовать следующую декларацию:

```
VAR
   fbusDiagnostics AT%IB11    : FIODiagnostics;
END_VAR
```

Пусть в конфигурацию контроллера добавлены 6 модулей аналогового ввода типа AIM726 и 9 модулей аналогового ввода типа AIM728, причем однотипные модули располагаются в конфигурации друг за другом. Также пусть требуется выводить в сеть MODBUS значения напряжения на каналах модулей AIM726 и AIM728. Суммарное количество каналов составляет 2 * 6 + 4 * 9 = 48, а значит в конфигурации сети контроллера для передачи 48-ми значений типа REAL должно быть создано не менее 96-ти входных регистров со смежными идентификаторами (адресами). Пусть первый из 96-ти созданных регистров имеет адрес %QB7 в области выходных данных среды исполнения.

Программа, преобразующая показания 6-ти модулей аналогового ввода типа AIM726, 9-ти модулей аналогового ввода типа AIM728 и выводящая результаты в MODBUS, может выглядеть следующим образом:

```
PROGRAM PLC_PRG
   VAR CONSTANT
         AIM726_ARRAY_SIZE :INT := 5;
         AIM728_ARRAY_SIZE :INT := 8;
         NETWORK_BUF_BOUND := 47;
   END_VAR
   VAR
   (* Адрес первого канала первого модуля AIM726 из шести – %IB37 *)
         aim726_inputs AT%IB37 : ARRAY [0..AIM726_ARRAY_SIZE] OF AIM726_inputs;
   (* Адрес первого канала первого модуля AIM728 из девяти – %IB91 *)
         aim728_inputs AT%IB91 : ARRAY [0..AIM728_ARRAY_SIZE] OF AIM728_inputs;
   (* Адрес канала первого регистра из 96-ти – %QB7 *)
         networkBuffer AT%QB7 : ARRAY [0.. NETWORK_BUF_BOUND] OF REAL;
   (* Массив блоков обработки показаний модулей AIM726 *)
         aim726_conv : ARRAY [0..AIM726_ARRAY_SIZE] OF AIM726_STIN;
   (* Массив блоков обработки показаний модулей AIM728 *)
         aim728_conv : ARRAY [0..AIM728_ARRAY_SIZE] OF AIM728_STIN;
         i : INT;
         netBufferIndex : INT;
   END_VAR
   (* Исполняемый код начинается здесь *)
   netBufferIndex := 0;
   FOR i := 0 TO AIM726_ARRAY_SIZE DO
         aim726_conv[i](inputs:= aim726_inputs[i], diagnostics=> , outputs=> );
         networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout0;
         netBufferIndex := netBufferIndex + 1;
         networkBuffer[netBufferIndex] := aim726_conv[i].outputs.vout1;
         netBufferIndex := netBufferIndex + 1;
   END_FOR;
   FOR i := 0 TO AIM728_ARRAY_SIZE DO
         aim728_conv[i](inputs:= aim728_inputs[i], diagnostics=> , outputs=> );
         networkBuffer[netBufferIndex] := aim728_conv[i].outputs.vout0;
         netBufferIndex := netBufferIndex + 1;
         networkBuffer[netBufferIndex] := aim728_conv[i].outputs.vout1;
         netBufferIndex := netBufferIndex + 1;
         networkBuffer[netBufferIndex] := aim728_conv[i].outputs.vout2;
         netBufferIndex := netBufferIndex + 1;
         networkBuffer[netBufferIndex] := aim728_conv[i].outputs.vout3;
         netBufferIndex := netBufferIndex + 1;
   END_FOR;
END_PROGRAM;
```

Как видно из приведенного исходного текста, в программе объявлены два массива непосредственно представляемых переменных типа *AIM726_inputs* и *AIM728_inputs*. Массив aim726_inputs, состоящий из шести элементов типа *AIM726_inputs*, размещается, начиная с адреса первого канала первого модуля AIM726 из шести имеющихся в конфигурации контроллера. Массив aim728_inputs, состоящий из девяти элементов типа *AIM728_inputs*, размещается, начиная с адреса первого канала первого модуля AIM728 из девяти имеющихся в конфигурации контроллера. Кроме того, для вывода в Modbus в программе объявлен массив из 48 переменных типа REAL, которые ссылаются на область выходных данных прикладной программы, начиная с адреса %QB7, т.е. с того места, где располагается выходной канал первого из 96-ти смежных регистров.

Далее, в программе объявлены два массива функциональных блоков типа *AIM726_STIN* и *AIM728_STIN*, содержащих 6 и 9 элементов соответственно. Вызовы блоков преобразования выполняются в двух циклах. Теперь в случае добавления каких-либо модулей перед первыми шестью AIM726 достаточно будет скорректировать значения адресов, на которые ссылаются переменные-массивы aim726_inputs и aim728_inputs, заглянув в секцию **PLC Configuration**. Если же какие-нибудь модули вставляются между первыми шестью AIM726 и группой из девяти AIM728, нужно будет скорректировать значение адреса, на который ссылается переменная-массив aim728_inputs. Кроме того, имеется возможность считывать значения всех 48-ми аналоговых каналов за один запрос чтения группы регистров, передаваемый мастером MODBUS контроллеру.

При использовании подобных приемов следует учитывать, что они работают только тогда, когда однотипные объекты окружения (модули ввода-вывода или коммуникационные объекты) располагаются в конфигурации контроллера друг за другом.

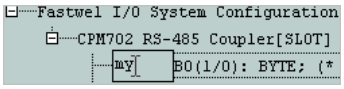
Основным недостатком данного способа является необходимость коррекции ссылок AT% в декларациях переменных при изменении структуры образа процесса, например, из-за вставки или удаления модулей ввода-вывода или коммуникационных объектов.

5.5.3. Создание символических имен каналов в ресурсе PLC Configuration

Данный способ позволяет избежать необходимости коррекции существующих ссылок AT% в декларациях переменных при изменении структуры образа процесса, однако не позволяет выполнять отображение структур и массивов.

Для создания символического имени следует:

1. Выбрать канал модуля ввода-вывода или коммуникационного объекта в дереве PLC Configuration
2. Дважды щелкнуть левой кнопкой мыши слева от надписи "AT%..." и ввести имя создаваемой переменной



3. Нажать клавишу Enter.

5.5.4. Использование ресурса VAR_CONFIG

Библиотеки поддержки платформы Fastwel I/O включают в себя функциональные блоки обработки данных от модулей ввода-вывода (с суффиксом *_DIRECT*), декларации входных и выходных переменных которых содержат недоопределенные ссылки на образ процесса в форме AT %I* или AT%Q*. Указанные ссылки должны быть доопределены в проекте в ресурсе *VAR_CONFIG*.

Пусть, например, программа *PLC_PRG* содержит объявление переменной типа AIM726_DIRECT:

```
VAR
   aim726_module1 : AIM726_DIRECT;
END_VAR
```

Если первый канал модуля AIM726, с которым ассоциируется данная переменная, расположен по адресу %IB37 во входной области образа процесса, то для доопределения ссылки блока на образ процесса ресурс *VAR CONFIG* должен содержать декларацию связи следующего вида:

```
VAR
   PLC_PRG.aim726_module1.inputs AT%IB37 : AIM726_inputs;
END_VAR
```

5.6. Создание обработчиков системных событий

Подробная информация об обработчиках системных событий приведена в п. 4.2.4.4 настоящего руководства. Однако при разработке приложений с обработкой системных событий еще раз настоятельно рекомендуется в качестве обработчиков системных событий использовать одну и ту же функцию с фиксированным интерфейсом, которая анализирует значение входного параметра и, в зависимости от передаваемого в нем типа события, вызывает другие функции, выполняющие требуемые действия по обработке системных событий. Интерфейс функций, вызываемых из функций-обработчиков системных событий, может быть любым.

Пример диспетчеризации системных событий:

```
FUNCTION SystemEventsDispatcher : DWORD
   VAR_INPUT
         eventType : DWORD;
   END_VAR
   CASE eventType OF
   F_EVENT_TIMER:
         ActualEventTimerHandler();
   F_EVENT_ONLINE_CHANGE:
         SaveMyPersistentVariables();
   F_EVENT_ON_INIT:
         LinkTasks();
         LoadMyPersistentVeariables();
   F_EVENT_POWER_ON:
```

```
            LoadMyRetainVariables();
    ELSE
            (* ничего не делаем *);
    END_CASE;
END_FUNCTION
```

Для добавления обработчика системного события:

1. Откройте окно ресурса **Tasks Configuration** и щелкните на элементе древовидного списка *System events*. В правой панели окна появится вкладка **System events**, показанная на рис. 24.
2. Дважды щелкните в ячейке таблицы *called POU* напротив названия системного события, для которого необходимо установить функцию-обработчик, и введите имя существующей функции (осторожно! у функции должен быть один входной параметр типа DWORD и возвращаемый результат типа DWORD и никаких внутренних переменных!) либо функции, которую собираетесь добавить сейчас же, после чего щелкните мышью на ячейке, расположенной слева от имени создаваемой функции, как показано на рис. 24.
3. Если для обработки события создается новая функция, станет доступной для нажатия кнопка **Create POU <*имя функции*>**. Нажмите ее, и в список **POUs** главного окна среды разработки будет добавлена функция с введенным именем.

Для удаления ранее установленного обработчика системного события щелкните на его имени в соответствующей ячейке *called POU*, после чего удалите имя функции нажатием кнопки Delete на клавиатуре компьютера.

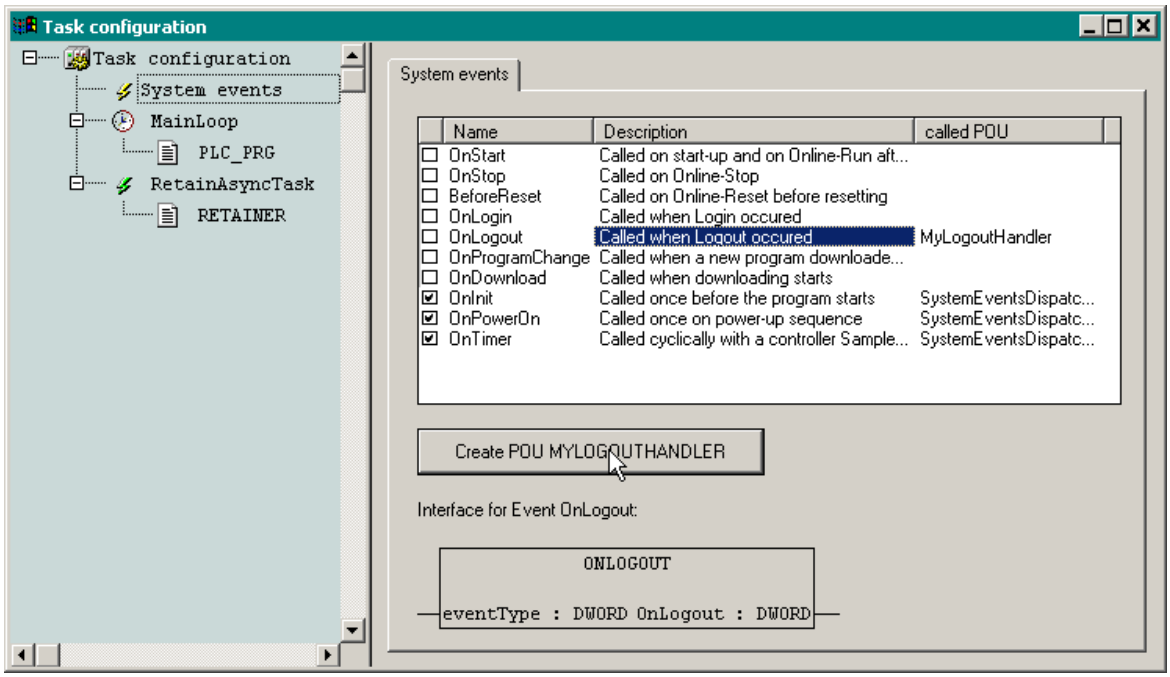


Рис. 24. Добавление обработчика системного события

5.7. Трансляция приложения

Для трансляции приложения перед загрузкой в контроллер или для проверки правильности выполненных операций по редактированию элементов проектной информации выберите команду **Project–Rebuild All.** Во избежание возможных аномалий (как в среде разработки, так и после загрузки приложения в контроллер) почаще выполняйте пару операций **Project–Clean All** и **Project–Rebuild All**.

Иногда после редактирования каких-нибудь параметров в окне ресурса **PLC Configuration**, при выполнении команды **Project–Build** в панели вывода диагностических сообщений транслятора CoDeSys появляются сообщения об ошибках в функции *_global_init*, хотя пользовательский код приложения не изменялся с момента последней успешной трансляции. Выполнение пары команд **Project–Clean All** и **Project–Rebuild All** позволит решить проблему.

В ряде случаев при отображении входных или выходных переменных на образ процесса среда разработки не распознает ситуацию, когда размер отображаемых данных превышает размер

соответствующей области образа процесса. В таком случае после загрузки программы контроллер переходит в безопасный режим с индикацией "Ошибка связывания" (см. табл. 1 п. 4.2.1.1).

5.8. Загрузка приложения в контроллер и отладка

5.8.1. Общие сведения

Среда разработки CoDeSys обеспечивает возможность выполнения следующих операций с контроллером по внешней сети или через соединение P2P:

1. Загрузку прикладной программы
2. Просмотр и изменение значений переменных прикладной программы
3. Перезапуск контроллера
4. Пошаговую отладку прикладной программы контроллера
5. Трассировку переменных
6. Просмотр потоков данных (**Online–Display flow control**) в программах на графических языках
7. и т.п.

Более подробная информация о перечисленных операциях приведена в эксплуатационной документации на среду разработки CoDeSys 2.3.

Перед началом выполнения любых операций с удаленным контроллером должна быть выполнена настройка параметров драйвера коммуникационного сервера CoDeSys Gateway Server. Настройка выполняется в соответствии с указаниями раздела 4 руководства по конфигурированию и программированию сетевых средств на контроллер определенного типа.

Выполнение операций с удаленным контроллером предваряется регистрацией среды CoDeSys на удаленном контроллере путем выполнения команды **Online–Login**.

5.8.2. Login

5.8.2.1. Общие сведения

Для соединения среды CoDeSys с контроллером настройте параметры CoDeSys Gateway Server таким образом, чтобы соединение производилось через логический информационный канал, параметры протокола которого соответствуют текущим установленным для внешней сети контроллера. **Login** через интерфейс прямого соединения P2P выполняется для всех контроллеров одинаково, независимо от типа.

Соединение устанавливается по команде меню **Online–Login**. После выполнения данной команды, если параметры CoDeSys Gateway Server настроены правильно, индикатор COMM на передней панели контроллера начинает светиться зеленым цветом. При соединении с контроллером через P2P индикатор COMM, в отсутствие трафика по внешней сети, светиться не будет.

При успешном выполнении **Login** строка состояния главного окна среды CoDeSys принимает вид, аналогичный приведенному на рис. 25.

Рис. 25. Внешний вид строки состояния CoDeSys при успешном соединении с удаленным контроллером

Если проект, открытый в среде CoDeSys в момент **Login**, содержит приложение, хотя бы в какой-то части отличающееся от имеющегося в контроллере, на экран монитора будет выведена диалоговая панель с предложением загрузить новую программу, показанная на рис. 26.

Если параметры CoDeSys Gateway Server отличаются от текущих параметров сервиса внешней сети контроллера, либо если отсутствует физическое соединение ПК с контроллером – на экран монитора будет выведено сообщение *Communication Error (#0). Logout Performed*.

Если в момент выполнения команды **Online–Login** через интерфейс внешней сети не светится индикатор COMM на передней панели контроллера, то это может быть связано с одной из следующих причин:

1. Выключено питание контроллера.
2. Отсутствует физическое сетевое соединение компьютера с контроллером.
3. Неправильно настроены параметры соединения сетевого адаптера компьютера.
4. Включен переключатель "4".
5. Неправильно настроены параметры информационного канала.

ПРИМЕЧАНИЕ. При соединении с контроллером по интерфейсу P2P индикатор COMM не светится.

5.8.2.2. Защита контроллера от несанкционированного соединения со средой разработки

Начиная с версии 2.64 системного программного обеспечения контроллеров и пакета адаптации CoDeSys 2.3 для Fastwel I/O, для защиты контроллера от несанкционированного соединения со средой разработки CoDeSys 2.3 следует воспользоваться командой *setpwd* браузера ПЛК:

1. В среде разработки CoDeSys 2.3 откройте проект приложения, загруженного в контроллер, или создайте новый пустой проект, после чего установите соединение с контроллером, выполнив команду **Online–Login** (**Онлайн–Подключение**). Если на экран монитора будет выведена диалоговая панель *The program has changed...*, – нажмите в ней кнопку **No** (**Нет**).
2. В среде разработки CoDeSys 2.3 откройте окно ресурса **PLC Browser** (**ПЛК-Браузер**) и в поле ввода команд введите:

   ```
   setpwd <пароль>
   ```

   при успешном приеме команды контроллером в области ответного сообщения браузера ПЛК будет отображено сообщение:

   ```
   Password has been set successfully
   ```

После перезапуска среды разработки CoDeSys 2.3 и при последующих попытках установления соединения с контроллером команду **Online–Login** (**Онлайн–Подключение**) на экран монитора будет в диалоговая панель **PLC password prompt** (**Запрос пароля ПЛК**), и для успешного соединения с контроллером потребуется ввести правильный (ранее заданный) пароль в поле **The PLC is password protected. Please enter the password** (**ПЛК требует пароль. Введите пароль:**).

Для сброса пароля, ранее установленного командой *setpwd*:

1. В среде разработки CoDeSys 2.3 откройте проект приложения, загруженного в контроллер, или создайте новый пустой проект, после чего установите соединение с контроллером, выполнив команду **Online–Login** и введя пароль в диалоговой панели **PLC password prompt** (**Запрос пароля ПЛК**). Если на экран монитора будет выведена диалоговая панель *The program has changed...*, – нажмите в ней кнопку **No (Нет).**
2. В среде разработки CoDeSys 2.3 откройте окно ресурса **PLC Browser** (**ПЛК-Браузер**) и в поле ввода команд введите:

   ```
   delpwd
   ```

   в области ответного сообщения браузера ПЛК будет отображено сообщение:

   ```
   Password has been deleted successfully
   ```

5.8.3. Загрузка приложения в контроллер

Для загрузки в контроллер:

1. Выберите команду **Online–Login**. При успешном соединении на экран будет выведена диалоговая панель, показанная на рис. 27.

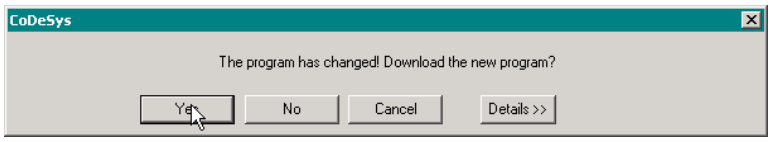


Рис. 26. Предложение загрузить программу

2. Убедитесь, что переключатель «1» контроллера выключен.
3. Нажмите кнопку **Yes**. На экран будет выведено окно, отображающее ход загрузки программы, показанное на рис. 27.

Следует обратить внимание на тот факт, что в данном окне отображается ход загрузки только исполняемого кода программы. Когда исполняемый код программы загружен, начинается загрузка секции конфигурации контроллера, а затем других секций, однако счетчик в окне показывает, будто бы загрузка остановилась. Указанная ситуация особенно заметна в больших проектах, когда конфигурация контроллера содержит большое количество модулей ввода-вывода и/или регистров.

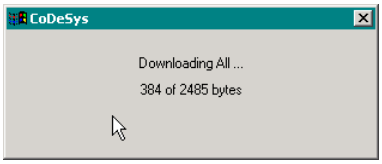


Рис. 27. Отображение хода загрузки программы

4. По завершении загрузки программы и конфигурации контроллер, при необходимости, выполняет конфигурирование в соответствии с содержимым секций загруженного приложения и переключается на новое приложение.

Более подробная информация о загрузке и горячем обновлении приложения приведена в п. 4.2.3 настоящего руководства.

5.8.4. Просмотр и установка значений переменных

Предполагается, что прикладная программа загружена в контроллер, а в среде CoDeSys открыт проект, в котором разрабатывалась данная программа.

Для просмотра значений переменных прикладной программы, исполняющейся в контроллере, выберите команду **Online–Login**. При успешном соединении среды CoDeSys с контроллером текущее активное окно редактора программ примет вид, показанный на рис. 28.

Для подготовки к записи нового значения в переменную дважды щелкните над переменной в верхней или правой области просмотра переменных и в появившейся диалоговой панели введите новое значение, как показано на рис. 29, и нажмите кнопку **OK**. Новое значение появится справа от текущего в треугольных скобках, как показано на рис. 30.

Для однократной записи нового значения нажмите сочетание клавиш Ctrl–F7 или выберите команду меню **Online–Write Values**. Новые значения всех подготовленных к изменению переменных будут однократно записаны в переменные.

Для записи и удержания нового значения нажмите клавишу F7 или выберите команду меню **Online–Force Values**. Новые значения всех подготовленных к изменению переменных будут записаны в переменные и сохраняться в них до тех пор, пока не будет выполнена команда меню **Online–Release Force** (сочетание клавиш Shift–F7).

Обратите внимание, что невозможны однократная запись и форсирование переменных, ссылающихся на область выходных данных, если эти переменные используются хотя бы в одной задаче.

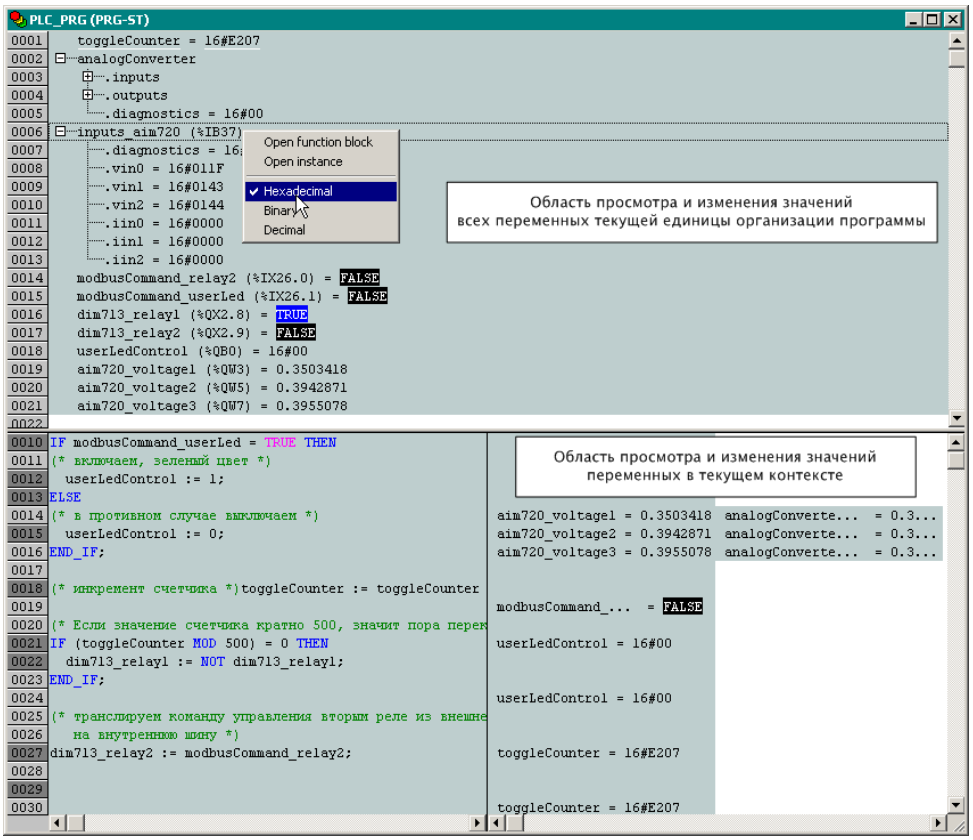


Рис. 28. Просмотр переменных

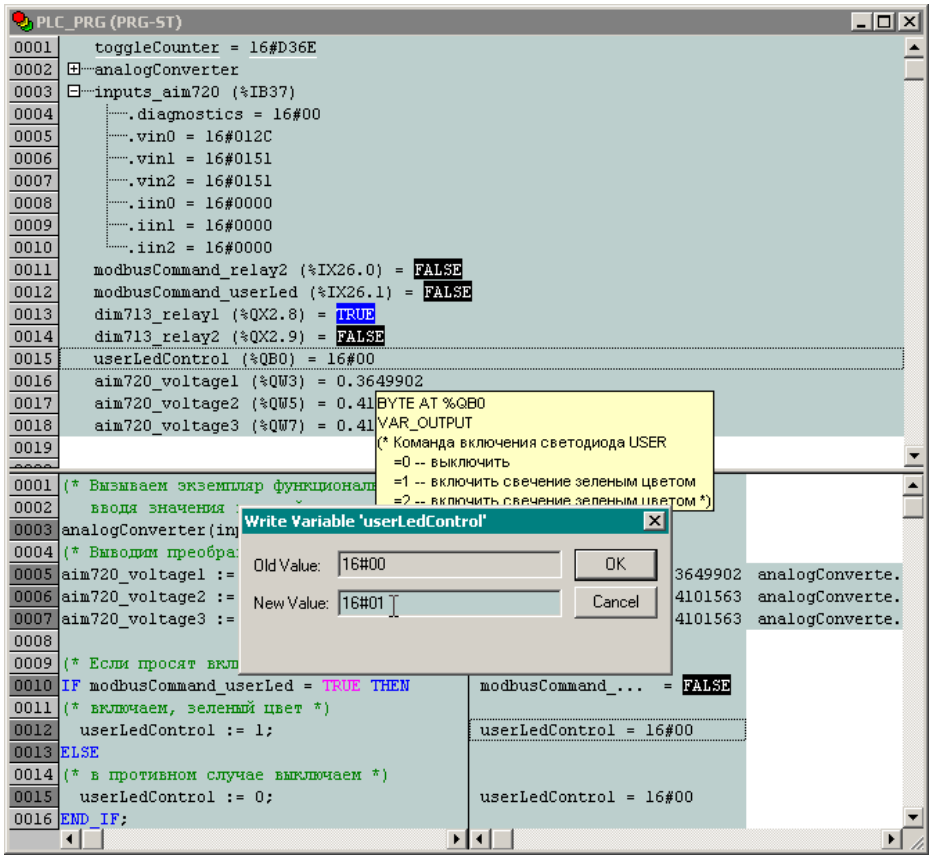


Рис. 29. Подготовка к изменению значения переменной

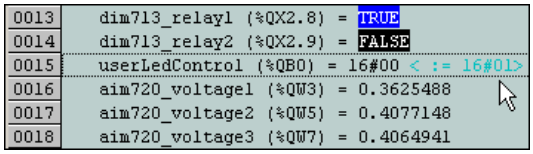


Рис. 30. Подготовленное значение переменной

5.9. Запись файлов в контроллер

Запись файлов в контроллер может быть необходима для загрузки энергонезависимых настроечных данных пользовательского приложения, а также используется для обновления системного программного обеспечения контроллера.

Для записи файла в контроллер выполните команду **Online–Login** в среде CoDeSys, после чего **Online–Write file to PLC**, а затем в появившейся диалоговой панели выберите файл, подлежащий загрузке, и нажмите **Open**. Если файл с запрашиваемым именем имеется в контроллере и не совпадает ни с одним из системных файлов, на экране появится диалоговая панель статуса загрузки файла. В противном случае на экране появится сообщение с кодом ошибки (см. п. 5.11).

Если выполняется обновление системного программного обеспечения контроллера (загрузка файла *norm.dnl*), то по окончании загрузки произойдет автоматический перезапуск контроллера.

При загрузке файлов в контроллер обратите внимание на следующие моменты:

1. Имя и расширение файла должны содержать только ASCII-символы латинского алфавита.
2. Длина имени файла не должна превышать 8 символов, расширение – 3
3. В контроллер можно загрузить не более 10-12 файлов общим размером не более 1,5 Мбайт
4. Операции записи и закрытия файлов могут потребовать от 20 до 200 мс в зависимости от размера, в течение которых работа приложения может быть приостановлена.

5.10. Чтение файлов из контроллера

Чтение файлов из контроллера может быть необходимо для выгрузки энергонезависимых данных пользовательского приложения, а также может быть использовано для получения дополнительной диагностической информации о причине перехода контроллера в безопасный режим (см. п. 4.2.1.1).

Для чтения файла из контроллера выполните команду **Online–Login** в среде CoDeSys, после чего **Online–Read file from PLC**, а затем в появившейся диалоговой панели введите имя файла, запрашиваемого у контроллера, и нажмите **Save**. Если файл с запрашиваемым именем имеется в пользовательском или корневом каталоге контроллера, и его имя не совпадает ни с одним из системных файлов, на экране появится диалоговая панель статуса выгрузки файла. В противном случае на экране появится сообщение с кодом ошибки (см. п. 5.11).

5.11. Описание кодов ошибок при взаимодействии между средой разработки и контроллером

При взаимодействии среды разработки CoDeSys с контроллерами Fastwel I/O на экране могут появляться сообщения с числовыми кодами ошибок последней операции взаимодействия. Перечень кодов представлен в табл. 10. К сожалению, среда разработки CoDeSys никак не реагирует на эти сообщения и не позволяет задать для них строки, удобные для восприятия.

Таблица 10

Код	Описание
50	Сервис не поддерживается данной платформой
104	При трассировке переменных запрошен слишком большой размер трассы
20001	Неправильная длина запроса на установление отладочной задачи по команде **Extras–Set Debug Task**
20002	В запросе на установление отладочной задачи по команде **Extras–Set Debug Task** поступил номер отсутствующей задачи.
20003	При включении **Online–Display Flow Control** оказалось, что текущая просматриваемая программная единица находится не в отладочной задаче. Для просмотра потоков данных необходимо сначала установить в качестве отладочной задачу, которая содержит требуемую программную единицу, а затем включить опцию **Online–Display Flow Control**.
20004	При включении **Online–Display Flow Control** оказалось, что текущая просматриваемая программная единица вызывается из нескольких задач. В таком случае просмотр потоков данных в текущей программной единице невозможен.
20005	Среда разработки при включении **Online–Display Flow Control** прислала позицию просмотра, которая не найдена в коде исполняющегося приложения или не содержит код команды временной точки останова.
20006	Среда разработки при включении **Online–Display Flow Control** прислала слишком много позиций для просмотра, которые не могут быть обработаны средой исполнения
20007	Просмотр стека вызовов по команде **Online–Show Call Stack** возможен только когда контроллер остановлен на установленной пользователем точке останова
20008	При включении **Online–Display Flow Control** оказалось, что текущая просматриваемая программная единица вызывается из ациклической задачи.
20009	Попытка записи или чтения файла с нулевой длиной имени
20010	Попытка чтения отсутствующего файла
20011	Возникла ошибка чтения файла
20012	Ошибка чтения информации о файле
20013	резерв
20014	Ошибка загрузки секции кода приложения в контроллер по команде **Online–Login**
20015	Ошибка загрузки секции конфигурации приложения
20016	Ошибка загрузки секции конфигурации задач
20017	Ошибка загрузки нового приложения
20018	Ошибка загрузки секции информации о проекте
20019	Запись системного файла запрещена
20020	Попытка перезаписи файла, открытого приложением или системным программным обеспечение контроллера
20021	При попытке записи по команде **Online–Write file to PLC** не удалось создать файл с заданным именем
20022	Возникла ошибка записи в файл
20023	Ошибка чтения информации о загруженном проекте
20024	Запрошенный список переменных не укладывается в буфер
20025	Не удалось начать загрузку нового приложения
20026	Попытка начать загрузку нового приложения во время другой незаконченной сессии загрузки
20027	Чтение системного файла запрещено
20028	резерв

5.12. Трассировка переменных

При трассировке переменных в ресурсе **Resources–Sampling Trace** следует учесть, что запись трассируемых значений выполняется в задаче, для которой установлен признак DEBUG в окне **Task Configuration** (по команде **Extras–Set Debug Task**).

6. СИСТЕМНЫЕ БИБЛИОТЕКИ

6.1. Общие сведения

К системным относятся библиотеки, функции и блоки которых реализованы не в среде разработки CoDeSys, а являются частью исполняемого кода программного обеспечения контроллера.

Среда исполнения CoDeSys для Fastwel I/O поддерживает следующие системные библиотеки:

1. Standard.lib – стандартная библиотека функций базовых преобразований и блоков
2. FastwelSysLibFile.lib – библиотека доступа к файловой системе
3. FastwelTasksExchange.lib – библиотека для организации межзадачного обмена
4. FastwelSysLibCom.lib – библиотека доступа к порту COM1.
5. FastwelModbusServer.lib – реализует функциональность подчиненного узла сети MODBUS RTU/ASCII через доступные пользовательской программе последовательные порты контроллера. В устройстве CPM701/CPM702/CPM703/CPM704 комплекса Fastwel I/O данная библиотека может быть использована для организации дополнительного сервиса внешней сети (MODBUS RTU/ASCII сервер) через коммуникационный порт, расположенный на его передней панели под пластиковой крышкой
6. FastwelUtils.lib – библиотека вспомогательных функций.
7. FastwelModbusRTUClientSerial.lib – написана на языке Structured Text в среде CoDeSys 2.3, и позволяет приложению контроллера реализовывать функции клиента (мастера) сети MODBUS RTU через любой коммуникационный порт.
8. FastwelPlatformControl.lib – содержит сервисные системные функции, включая функции записи и чтения пользовательского серийного номера, а также функцию сброса контроллера из приложения

6.2. Библиотека FastwelSysLibFile.lib

6.2.1. Общие сведения

Функции данной библиотеки предназначены для доступа к файловой системе контроллера. По сути данная библиотека является расширенной копией исходной библиотеки SysLibFile.lib, входящей в стандартную среду исполнения CoDeSys, однако из-за ряда особенностей компилятора среды CoDeSys в FastwelSysLibFile.lib все возвращаемые результаты типа BOOL заменены на тип WORD. При этом сохранена семантика возвращаемого результата: нулевое значение (0) означает FALSE, а ненулевое – TRUE.

Доступ к файловой системе контроллера сопряжен с риском сделать контроллер полностью неработоспособным, поскольку в его состав входит только один диск на основе флэш-памяти, размер свободного пространства которого изначально составляет от 2,0 до 2,5 Мбайт, а скорость записи минимального блока данных – до 15-20 мс. Во время записи и закрытия файлов возникают моменты, когда запрещены прерывания или блокирован планировщик операционной системы, а сами операции записи потребляют много вычислительных ресурсов, что может повлиять на стабильность поддержания фиксированного времени цикла приложения.

При циклическом выполнении операций записи в файлы, например, в случае использования библиотеки FastwelRetainsSupport.lib для сохранения значений переменных в энергонезависимой памяти контроллера, может потребоваться оценить ориентировочное время до износа флэш-диска. Оценка времени до износа, учитывающая используемый при работе с флэш-диском встроенный алгоритм выравнивания износа, может быть произведена следующим образом:

1. Определить T_{min} – минимальный период записи данных в секундах. Например, минимальный период сохранения данных при использовании функционального блока RETAIN_VAR_ENGINE из библиотеки FastwelRetainsSupport.lib определяется параметром InputMinimumPeriod.
2. Определить N_{Dmax} – максимальное количество циклов записи данных:

$N_{Dmax} = N_{max} \times S_{max} / (S_{page} \times N_{pages})$,

где: N_{max} – максимальное количество циклов перезаписи, гарантируемое производителем, и равное 100000;

S_{max} – максимальный размер дискового пространства, доступный пользователю, и равный около 2 Мбайт;

S_{page} – минимальный размер перезаписываемого блока данных флэш-памяти, равный 512 байт. Запись любого количества байт, меньшего, либо равного, 512 байт, приводит к перезаписи одного блока. Кроме того, как минимум, один дополнительный блок модифицируется алгоритмом выравнивания износа.

N_{pages} – количество перезаписываемых блоков флэш-памяти, необходимое для записи некоторого количества байт пользовательских данных и определяемое по формуле:

$N_{pages} = 1 + (S_{wr} / S_{page}) \Rightarrow 1 + (S_{wr} / 512)$, где S_{wr} – количество записываемых байт пользовательских данныых.

Пусть, например, приложению требуется периодически записывать на флэш-диск блок данных размером 1075 байт. Тогда максимальное количество циклов записи составит:

$N_{Dmax} = 100000 \times 2 \times 1024 \times 1024 / (512 \times (1 + (1075 / 512)) \approx 136500000$ (циклов).

3. Исходя из минимального периода записи данных и максимального количества циклов записи, вычислить оценку времени до износа в сутках по формуле:

   $T_{wear} = (T_{min} \times N_{Dmax}) / 86400$

   Пусть, например, период записи Tmin равен 1 с, а N_{Dmax} составляет 136500000 циклов. Тогда оценка времени до износа:

   $T_{wear} = 1 \times 136500000 / 86400 \approx 1579$ суток $\approx 4{,}3$ года.

Для функций данной библиотеки на диске контроллера отведен специальный каталог *\user*, в который перенаправлены большинство операций с файлами и каталогами. Каталог *\user* является корневым каталогом для функций из библиотеки FastwelSysLibFile.lib, поэтому не требуется указывать его явно в именах файлов, передаваемых функциям библиотеки. Однако операции открытия файлов для чтения в первую очередь обращаются в корневой каталог диска контроллера, поскольку загрузка файлов командой **Online–Write file to PLC** может быть выполнена только в корневой каталог. Если в корневом каталоге файл с заданным именем не найден, то осуществляется попытка обратиться к файлу в каталоге *\user*.

Примеры использования функций библиотеки имеются в проектах, входящих в пакет адаптации CoDeSys для Fastwel I/O.

6.2.2. Описание функций

6.2.2.1. FwSysFileGetSize

Возвращает размер файла, имя которого передано в качестве параметра.

```
FUNCTION FwSysFileGetSize : DINT
VAR_INPUT
   FileName: STRING;
END_VAR
;
END_FUNCTION
```

Входные параметры:

FileName: STRING
Имя файла. Поиск файла выполняется относительно каталога \user. Пример имен:

```
fileName1 : STRING := 'MyData.dat';
fileName2 : STRING := 'subdir\MyData.dat';
```

Возвращаемый результат:

размер файла с заданным именем;
если отсутствует либо в качестве параметра передан нулевой указатель: -1.

6.2.2.2. FwSysFileExists

```
FUNCTION FwSysFileExists : WORD
VAR_INPUT
   FileName: STRING;
END_VAR
;
END_FUNCTION
```

Входные параметры:

FileName: STRING
Имя файла. Поиск файла выполняется относительно каталога \user.

Возвращаемый результат:

возвращает ненулевое значение, если файл имеется в контроллере, и 0 – в противном случае

6.2.2.3. FwSysFileGetTime

Для файлов, которые созданы или модифицированы в самом контроллере, вызов данной функции смысла не имеет, поскольку в составе контроллера нет часов/календаря

```
FUNCTION FwSysFileGetTime : WORD
VAR_INPUT
   FileName: STRING;
   ftFileTime: POINTER TO FILETIME;
END_VAR
;
END_FUNCTION
```

Входные параметры:

FileName: STRING
Имя файла. Поиск файла выполняется относительно каталога \user.
ftFileTime: POINTER TO FILETIME;
Указатель на структуру, содержащую информацию о времени создания, последнего доступа и последней модификации файла.
Структура имеет следующий вид:

```
TYPE FILETIME :
  STRUCT
     (* время создания *)
     dtCreation:DT;
     (* время последнего доступа *)
     dtLastAccess:DT;
     (* время последней модификации *)
     dtLastModification:DT;
  END_STRUCT
END_TYPE
```

Возвращаемый результат:

возвращает ненулевое значение, если структура заполнена информацией о файле, и 0 – в противном случае

6.2.2.4. FwSysFileCopy

Создает копию файла, имя которого указано в качестве второго параметра

```
FUNCTION FwSysFileCopy : DWORD
VAR_INPUT
   FileDest: STRING;
   FileSource: STRING;
END_VAR
;
END_FUNCTION
```

Входные параметры:

FileDest: STRING
Имя файла-копии. Файла создается относительно каталога \user.
FileSource: STRING
Имя файла-оригинала. Файла создается относительно каталога \user.

Возвращаемый результат:

возвращает количество скопированных байт.

6.2.2.5. FwSysFileDelete

Функция удаляет файл с заданным именем, возвращая ненулевое значение в случае успеха. Поиск файла выполняется относительно каталога *user*.

6.2.2.6. FwSysFileRename

Функция изменяет имя файла, заданное в качестве первого параметра, на имя, заданное в качестве второго параметра, возвращая ненулевое значение в случае успеха. Поиск файла выполняется относительно каталога *user*.

6.2.2.7. FwSysFileOpen

Функция открывает файл для чтения, только записи, чтения и записи или чтения/записи с созданием в случае отсутствия.

Максимальное количество одновременно открытых файлов ограничено 16-ю.

По завершении работы с файлом его необходимо закрыть, иначе операции записи могут быть не завершены.

При загрузке приложения пользователя непосредственно перед началом переконфигурирования контроллера система автоматически закрывает все незакрытые пользователем файлы.

Если разрешен режим горячего обновления, и выясняется, что структуры данных программы, размеры образа процесса и связи задач с образом процесса не изменились, то закрытие файлов, незакрытых пользователем до обновления, не выполняется.

Если файл открывается только для чтения, поиск его сначала ведется относительно корневого каталога файловой системы контроллера, а затем – относительно каталога \user.

```
FUNCTION FwSysFileOpen : DWORD
VAR_INPUT
   FileName: STRING;
   Mode: STRING [20];
END_VAR
;
END_FUNCTION
```

Входные параметры:

FileName: STRING
Имя открываемого файла.
Mode: STRING
Режим открытия:
'r' – открыть только для чтения;
'w' – открыть только для записи;
'rw' – открыть для чтения и записи;
'cw' – открыть для чтения и записи, если файл не найден – создать;

Возвращаемый результат:

Системный идентификатор файла (хэндл). Ненулевое значение возвращается, если операция завершена успешно. Данное значение требуется сохранить для использования в последующих операциях чтения, записи и для закрытия.

6.2.2.8. FwSysFileClose

Закрывает файл, системный идентификатор которого передан в качестве параметра. Возвращает ненулевое значение в случае успеха.

6.2.2.9. FwSysFileGetPos

Возвращает текущую позицию в потоке ввода-вывода файла, системный идентификатор которого передан в качестве параметра. Возвращает значение типа DINT, большее либо равное нулю, в случае успеха.

6.2.2.10. FwSysFileSetPos

Устанавливает текущую позицию, переданную в качестве второго параметра, в потоке ввода-вывода файла, системный идентификатор которого задан в качестве первого параметра. Возвращает ненулевое значение в случае успеха.

6.2.2.11. FwSysFileRead

Пытается прочитать *Size* байт из файла, заданного первым параметром, в буфер *Buffer*, начиная с текущей позиции потока ввода-вывода данного файла.

```
FUNCTION FwSysFileRead : DWORD
VAR_INPUT
   File: DWORD;
   Buffer: DWORD;
   Size: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

File: DWORD
Системный идентификатор файла.
Buffer: DWORD
Указатель на буфер, в который требуется выполнить чтение
Size: DWORD;
Количество байт, которое требуется прочитать

Возвращаемый результат:

Количество прочитанных байт.

6.2.2.12. FwSysFileWrite

Пытается записать *Size* байт в файл, заданный первым параметром, из буфера *Buffer*, начиная с текущей позиции потока ввода-вывода данного файла.

```
FUNCTION FwSysFileWrite : DWORD
VAR_INPUT
   File: DWORD;
   Buffer: DWORD;
   Size: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

File: DWORD
Системный идентификатор файла.
Buffer: DWORD
Указатель на буфер, из которого требуется выполнить запись
Size: DWORD;
Количество байт, которое требуется записать

Возвращаемый результат:

Количество записанных байт.

6.2.2.13. FwSysFileEOF

Возвращает ненулевое значение, если текущая позиция в потоке ввода-вывода файла совпадает с концом файла.

6.2.2.14. FwSysDirCreate

Создает подкаталог с заданным именем в каталоге *\user*. Поддерживается однократный уровень вложенности, т.е. передавать возможно имена без букв диска и без символа-разделителя пути. Например, вот это правильно: '*subdir*'. А это – '*sub1\sub2*' – нет.

6.2.2.15. FwSysDirExist

Возвращает ненулевое значение, если в каталоге *user* был обнаружен и успешно удален подкаталог с заданным именем. Причиной неудачного удаления может быть наличие файлов в удаляемом каталоге.

6.2.2.16. FwSysDirRemove

Возвращает ненулевое значение, если в каталоге \user имеется подкаталог с заданным именем.

6.3. Библиотека FastwelTasksExchange.lib

6.3.1. Общие сведения

Библиотека содержит функцию F_IecTasks_linkVariables (см. п. 4.2.4.5), предназначенную для связывания задач по данным, а также вспомогательные функции:

F_IecTasks_getCount – возвращает общее количество циклических и ациклических задач в исполняющемся приложении;

F_getProjectName – возвращает имя проекта, загруженного в контроллер;

F_IecTasks_getInfo – возвращает диагностическую информацию о циклической или ациклической задаче.

6.3.2. Функция F_IecTasks_getInfo

Данная функция принимает указатель на переменную типа F_TASK_INFO в качестве первого параметра и возвращает диагностическую информацию о задаче, номер которой передан вторым параметром. Если задача с заданным номером отсутствует в системе, функция возвращает 0.

Структура F_TASK_INFO определена следующим образом:

```
TYPE F_TASK_INFO :
STRUCT
   (* для циклической задачи – период выполнения в микросекундах *)
   (* для ациклической задачи – 16#FFFFFFFF *)
   period_us : DWORD;
   (* кол-во циклов, выполненных задачей *)
   cyclesCount : DWORD;
   (* для циклической задачи – кол-во циклов, на которых задача *)
   (* не уложилась в заданный период исполнения *)
   (* для ациклической задачи – 0 *)
   overrunsCount : DWORD;
   (* минимальное время исполнения, мкс *)
   minExecutionTime_us : DWORD;
   (* максимальное время исполнения, мкс *)
   maxExecutionTime_us : DWORD;
   (* имя задачи *)
   name : STRING(31);
END_STRUCT
END_TYPE
```

Номер задачи, передаваемый в качестве второго параметра, является индексом задачи (начиная с 0) в древовидном списке ресурса **Task Configuration** среды разработки CoDeSys.

При вызове F_IecTasks_getInfo на контексте какой-либо циклической задачи в качестве номера может использоваться значение 16#FFFF. В этом случае функция вернет статистику для текущей циклической задачи.

6.4. Библиотека FastwelSysLibCom.lib

6.4.1. Общие сведения

Функции данной библиотеки предназначены для доступа к коммуникационному порту контроллера COM1, расположенному на его передней панели под пластиковой крышкой, и к портам, реализованным на базе интерфейсных модулей NIM741/NIM742, для которых в конфигурацию

контроллера добавлены элементы *NIM74x RS-xxx 1xUART Stream Module*. Данная библиотека является полной копией исходной библиотеки SysLibCom.lib, входящей в стандартную среду исполнения CoDeSys, однако из-за ряда особенностей компилятора среды CoDeSys в FastwelSysLibCom.lib все возвращаемые результаты типа BOOL заменены на тип WORD. При этом сохранена семантика возвращаемого результата: 0 означает FALSE, а не 0 – TRUE.

Следует обратить внимание на тот факт, что в реализации системы исполнения CoDeSys фирмы Фаствел функции *FwSysComRead()* и *FwSysComWrite()* не блокируют выполнение вызывающей программы, как это сказано в документации на библиотеку SysLibCom.lib CoDeSys.

Пример использования функций библиотеки имеется в проекте fsyslibcom_test_cpm701.pro, который входит в пакет адаптации CoDeSys для Fastwel I/O. Открывать коммуникационный порт COM1 следует при обработке системного события *OnInit*. Открывать порты 101–164, соответствующие модулям NIM741/NIM742 необходимо в коде приложения. При обработке системного события *OnProgramChange* порт обязательно необходимо закрыть, иначе новая версия загруженной программы не получит к нему доступ.

Для того, чтобы функции библиотеки получили доступ к коммуникационному порту COM1, необходимо до включения питания контроллера включить переключатель "4" блока переключателей. При этом после перезапуска контроллера утрачивается возможность взаимодействия между средой разработки CoDeSys и контроллером через коммуникационный канал P2P.

Идентификатор COM-порта, передаваемый функции *FwSysComOpen* библиотеки FastwelSysLibCom.lib для получения доступа к порту на основе элемента *NIM74x RS-xxx 1xUART Stream Module*, формируется по правилу

```
100+n
```

где n – позиция модуля, начиная с 1, на внутренней шине контроллера. При этом из нумерации должны быть исключены все вспомогательные модули, не участвующие в обмене по шине FBUS: OM752, OM754, OM755, OM756, OM757, OM759, OM796 и модуль оконечный OM750. Таким образом, n – это позиция модуля в секции ...*I/O Modules* ресурса **PLC Configuration**, начиная с 1.

Пример

Пусть к контроллеру узла сети подключены следующие модули ввода-вывода и вспомогательные модули: AIM721, AIM730, OM751, DIM717, DIM717, NIM741, OM752, AIM722, AIM722, AIM72503, NIM741, NIM742, NIM742, OM750, и приложению требуется получить программный доступ ко всем последовательным портам на основе модулей NIM741 и NIM742 с использованием системной библиотеки FastwelSysLibCom.lib.

Тогда в конфигурацию сервиса ввода-вывода приложения должны быть добавлены следующие элементы:

1. *AIM721 4-channels 0-20mA Analog Input Module*
2. *AIM730 2-channels Current Output Module*
3. *OM751 24VDC Power Supply Module*
4. *DIM717 8-channels 30VDC Digital Input Module*
5. *DIM717 8-channels 30VDC Digital Input Module*
6. *NIM741 RS-485 1xUART Stream Module*
7. *AIM722 2-channels 0-20mA Analog Input Module*
8. *AIM722 2-channels 0-20mA Analog Input Module*
9. *AIM72503 RTD Inputs Module (GOST 6651-2009)*
10. *NIM741 RS-485 1xUART Stream Module*
11. *NIM742 RS-232 1xUART Stream Module*
12. *NIM742 RS-232 1xUART Stream Module*

В конфигурации присутствуют по два элемента *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module*, обеспечивая возможность организации двух портов интерфейса RS-485 и двух портов интерфейса RS-232C соответственно. Данные элементы имеют порядковые номера 6, 10, 11, 12. Согласно приведенному выше правилу формирования идентификаторов портов, функции *FwSysComOpen* должны быть переданы следующие идентификаторы:

106 (первый NIM741)

110 (второй NIM741)

111 (первый NIM742)

112 (второй NIM742)

6.4.2. Описание функций

6.4.2.1. FwSysComOpen

Функция открывает коммуникационный порт контроллера для использования. Вызов данной функции для портов с номерами от 101 до 164 (но не более 16-ти) должен производиться в коде приложения, выполняемом на контексте циклической, ациклической или сервисной задачи.

```
FUNCTION FwSysComOpen : DWORD
VAR_INPUT
   Port: PORTS;
END_VAR
;
END_FUNCTION
```

Входные параметры:

Port: PORTS
Идентификатор порта перечислимого типа PORTS: (COM1:=1, COM2, COM3, COM4, COM5, COM6, COM7, COM8). Для контроллеров CPM701, CPM702 и CPM703 допустимые значение: COM1, 101–164 – для портов, организованных посредством элементов *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module* в конфигурации контроллера, но не более 16-ти портов. Для контроллера CPM704: COM2, 101–164 – но не более 16-ти.

Возвращаемый результат:

Системный идентификатор порта (хэндл), который в дальнейшем используется в вызовах других функций библиотеки. В случае ошибки (если порт не может быть открыт) возвращается значение: 0xFFFFFFFF.

6.4.2.2. FwSysComClose

Функция освобождает коммуникационный порт.

```
FUNCTION FwSysComClose : WORD
VAR_INPUT
   dwHandle: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

dwHandle: DWORD
Системный идентифтикатор порта (хэндл), полученный при открытии порта.

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успеха.

6.4.2.3. FwSysComSetSettings

Устанавливает параметры коммуникационного порта: скорость и параметры передачи, размер коммуникационного буфера.

```
FUNCTION FwSysComSetSettings : WORD
VAR_INPUT
   dwHandle: DWORD;
   ComSettings: POINTER TO COMMSETTINGS;
END_VAR
;
END_FUNCTION
```

Входные параметры:

dwHandle: DWORD
Системный идентифтикатор порта (хэндл), полученный при открытии порта.
ComSettings: POINTER TO COMMSETTINGS
Указатель на переменную структурного типа COMMSETTINGS, в которую записаны требуемые параметры коммуникационного порта. Определение полей структуры и допустимые значения:

```
TYPE COMSETTINGS :
STRUCT
   Port: PORTS;              Системное имя порта перечисляемого типа PORTS: (COM1)
   dwBaudRate: DWORD;        4800, 9600, 19200, 38400, 57600, 115200
   byStopBits: BYTE;         0 = ONESTOPBIT, 1=ONE5STOPBITS, 2=TWOSTOPBITS
   byParity: BYTE;           0 = NOPARITY, 1 = ODDPARITY, 2 = EVENPARITY
   dwTimeout: DWORD;         не используется,
   dwBufferSize: DWORD;      размер буфера устройства, должен быть > 0
   dwScan: DWORD;            не используется
END STRUCT
END TYPE
```

Размер приемного и передающего буферов будут равны dwBufferSize.

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успеха.

6.4.2.4. FwSysComRead

Пытается прочитать dwBytesToRead байт, начиная с текущей позиции, из приемного буфера устройства.

```
FUNCTION FwSysComRead : DWORD
VAR_INPUT
   dwHandle: DWORD;
   Buffer: DWORD;
   dwBytesToRead: DWORD;
   dwTimeout: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

dwHandle: DWORD
Системный идентифтикатор порта (хэндл), полученный при открытии порта.
Buffer: DWORD
Указатель на буфер, в который требуется выполнить чтение
dwBytesToRead: DWORD;
Количество байт, которое требуется прочитать
dwTimeout: DWORD;
Параметр не используется.

Возвращаемый результат:

Функция возвращает фактический размер считанных данных.

6.4.2.5. FwSysComWrite

Пытается записать dwBytesToWrite байт в передающий буфер устройства, начиная с текущей позиции.

```
FUNCTION FwSysComRead : DWORD
VAR_INPUT
   dwHandle: DWORD;
   Buffer: DWORD;
   dwBytesToRead: DWORD;
   dwTimeout: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

dwHandle: DWORD
Системный идентифтикатор порта (хэндл), полученный при открытии порта.
Buffer: DWORD
Указатель на буфер, из которого требуется выполнить запись
dwBytesToRead: DWORD;
Количество байт, которое требуется записать
dwTimeout: DWORD;
Параметр не используется.

Возвращаемый результат:

Количество записанных байт.

6.5. Библиотека FastwelModbusServer.lib

6.5.1. Общие сведения

Библиотека FastwelModbusServer.lib реализует функциональность подчиненного узла сети MODBUS RTU/ASCII через доступные пользовательской программе последовательные порты контроллера. В устройстве CPM701/CPM702/CPM703/CPM704 комплекса Fastwel I/O данная библиотека может быть использована для организации дополнительного сервиса внешней сети (MODBUS RTU/ASCII сервер) через коммуникационный порт, расположенный на его передней панели под пластиковой крышкой, или через порты, организованные на основе модулей NIM741/NIM742 и элементов *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module*.

Примечание. Для того, чтобы функции библиотеки получили доступ к данному коммуникационному порту, необходимо до включения питания контроллера включить переключатель "4" блока переключателей. При этом после перезапуска контроллера утрачивается возможность взаимодействия между средой разработки CoDeSys и контроллером через коммуникационный канал P2P.

Пример использования библиотеки имеется в проекте *fmbserverlib_test_cpm703.pro*, который входит в пакет адаптации CoDeSys для Fastwel I/O. Наиболее актуальную спецификацию протокола MODBUS over Serial Line можно загрузить с Web-узла http://www.modbus.org.

Общее количество серверов MODBUS, организуемых средствами библиотеки FastwelModbusServer.lib, ограничено 4.

6.5.2. Описание функций

FwModbusServerInit() является единственной функцией библиотеки и предназначена для инициализации и конфигурирования сервера. При вызове данной функции пользователь задаёт коммуникационные параметры узла сети и описывает области данных, которые будут отображаться на пространство адресов сервера MODBUS. Для каждого последовательного порта, поддерживаемого библиотекой, актуальная конфигурация сервера MODBUS определяется последним вызовом *FwModbusServerInit()*. Таким образом, в приложении, как правило, не имеет смысла использовать более одного вызова данной функции для некоторого последовательного порта.

Примечание. Инициализация сервера допускается только из обработчика системного события *OnInit*. Подробная информация об обработчиках системных событий приведена в п. 4.2.4.4, рекомендации и правила создания обработчика приведены в п. 5.6 настоящего руководства.

Функция *FwModbusServerInit()* имеет следующий прототип (в синтаксисе IEC 61131-3):

```
FUNCTION FwModbusServerInit: INT
VAR_INPUT
   pModbusSettings : POINTER TO F_MODBUS_SERVER_SETTINGS;
   pInputDescriptor : POINTER TO F_VAR_DESCRIPTOR;
   pOutputDescriptor : POINTER TO F_VAR_DESCRIPTOR;
   pDiagnosticsDescriptor : POINTER TO F_VAR_DESCRIPTOR;
END_VAR
;
END_FUNCTION
```

Входные параметры:

pModbusSettings : POINTER TO F_MODBUS_SERVER_SETTINGS
Указатель на переменную структурного типа **F_MODBUS_SERVER_SETTINGS,** задающую параметры сетевой конфигурации сервера:

```
TYPE F_MODBUS_SERVER_SETTINGS :
STRUCT
   Port:BYTE;           (* Порт: 1(COM1), 2(COM2), 101, 102,..., 164*)
   BaudRate:DWORD;      (* Скорость передачи: 9600, 19200, 38400, 57600, 115200 *)
   StopBits:BYTE;       (* Число стоп-битов: 1, 2 *)
   Parity:BYTE;         (* Контроль четности: 0(нет), 1(нечётность), 2(чётность) *)
   ByteSize:BYTE;       (* Число битов данных: 8(RTU), 7(ASCII) *)
   NodeAddress:BYTE;    (* Адрес узла в сети MODBUS (1 – 247) *)
END_STRUCT
END_TYPE
```

Для контроллеров CPM701, CPM702 и CPM703 допустимые значение: COM1, 101–164 (для портов, организованных посредством элементов *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module* в конфигурации контроллера). Для контроллера CPM704: COM2, 101–164.

pInputDescriptor : POINTER TO F_VAR_DESCRIPTOR
Указатель на описатель переменной, которая будет использоваться в качестве приемника данных для входящих (принимаемых по сети) коммуникационных объектов сервера, сгруппированных в набор доступных по записи и чтению битовых полей типа Coil и регистров хранения (Holding Registers). Допустимо передавать нулевое значение указателя, что означает отсутствие у сервера входящих коммуникационных объектов.

pOutputDescriptor: POINTER TO F_VAR_DESCRIPTOR
Указатель на описатель переменной, которая будет использоваться в качестве источника данных для исходящих (передаваемых в сеть) коммуникационных объектов сервера, сгруппированных в набор доступных только по чтению битовых полей типа Discrete Input и входных регистров (Input Registers). Допустимо передавать нулевое значение указателя, что означает отсутствие у сервера исходящих коммуникационных объектов.

pDiagnosticsDescriptor: POINTER TO F_VAR_DESCRIPTOR
Указатель на описатель переменной структурного типа **F_MODBUS_SERVER_DIAGNOSTICS**, которая будет использоваться в качестве приемника диагностической информации сервера. Допустимо передавать нулевое значение указателя, что означает невостребованность диагностической информации в приложении пользователя.
Описатели связываемых переменных являются переменными типа STRUCT **F_VAR_DESCRIPTOR**, определенного следующим образом:

```
TYPE F_VAR_DESCRIPTOR :
STRUCT
(* Указатель на переменную.
   Пример: descr.Address := ADR(SomeProgram.SomeVariable); *)
   Address : DWORD;
(* Размер переменной (в байтах).
   Пример: descr.Size := SIZEOF(SomeProgram.SomeVariable); *)
   Size : INT;
(* Индекс программной единицы (POU), содержащую описываемую переменную.
   Пример: descr.PouIndex := INDEXOF(SomeProgram); *)
   PouIndex : INT;
END_STRUCT
END_TYPE
```

Возвращаемый результат:

Код	Значение
MBSRV_INIT_OK (0)	Инициализация сервера выполнена успешно.
MBSRV_INIT_INVALID_STATE(1)	Попытка вызова данной функции в месте, отличном от обработчика события OnInit, или в режиме симуляции (**Online–Simulation Mode**).
MBSRV_INIT_INVALID_NODE_CFG(2)	Ошибка применения сетевой конфигурации. Ситуации: неправильные значения одного или нескольких параметров в структуре **F_MODBUS_SERVER_SETTING** или не удалось открыть коммуникационный порт.
MBSRV_INIT_INVALID_INPUTS(3)	Неправильный описатель переменной - источника выходных данных. Ситуации: – адрес переменной (**pInputDescriptor^.Address**) равен 0; – неправильная длина переменной (**pInputDescriptor^.Size**) равна 0 или более размера глобальной области данных; – переменная находится во входном (INPUT (AT%I...)) или выходном OUTPUT (AT%Q...), или флаговом (AT%M...) сегментах; – заданный индекс программной единицы (**pInputDescriptor^.PouIndex**) не относится к множеству индексов программных единиц, вызываемых из каких-либо циклических или ациклических задач приложения.
MBSRV_INIT_INVALID_OUTPUTS(4)	Неправильный описатель переменной - приемника входных данных. Ситуации: – адрес переменной (**pOutputDescriptor^.Address**) равен 0; – длина переменной (**pOutputDescriptor^.Size**) равна 0 или более размера глобальной области данных; – переменная находится во входном (INPUT (AT%I...)) или выходном OUTPUT (AT%Q...), или флаговом (AT%M...) сегментах;; – заданный индекс программной единицы (**pOutputDescriptor^.PouIndex**) не относится к множеству индексов программных единиц, вызываемых из каких-либо циклических или ациклических задач приложения.
MBSRV_INIT_INVALID_DIAGNOSTICS (5)	Неправильный описатель переменной - приемника диагностических данных сервера. Ситуации: – адрес переменной (**pDiagnosticsDescriptor^.Address**) равен 0;

Код	Значение
	– длина переменной (**pDiagnosticsDescriptor^.Size**) не равна размеру структуры **F_MODBUS_SERVER_DIAGNOSTICS**; – переменная находится во входном (INPUT (AT%I...)) или выходном OUTPUT (AT%Q...), или флаговом (AT%M...) сегментах; – заданный индекс программной единицы **pDiagnosticsDescriptor^.PouIndex**) не относится к множеству индексов программных единиц, вызываемых из каких-либо циклических или ациклических задач приложения.
MBSRV_INIT_NO_RESOURCES (6)	Недостаточно памяти или других системных ресурсов.
MBSRV_INIT_LINK_FAILURE (7)	Не удалось выполнить отображение переменной, описанной в **pInputDescriptor** или **pOutputDescriptor**, на коммуникационные объекты MODBUS. Возможная причина: перекрытие областей памяти, описанных **pInputDescriptor** и **pOutputDescriptor**

6.5.3. Принцип работы

6.5.3.1. Обмен данными с клиентами Modbus

Сервер MODBUS, реализуемый библиотекой *FastwelModbusServer.lib*, поддерживает следующие стандартные сетевые операции протокола:

	Тип	*Описание*
Операции протокола Modbus	01	Выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для записи (Coils)
	02	Выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для чтения (Discretes Input)
	03	Выдача за один запрос от 1 до 125 смежных регистров, доступных для записи (Holding Registers)
	04	Выдача за один запрос от 1 до 125 смежных регистров, доступных для чтения (Input Registers)
	05	Прием значения одного битового поля, доступного для записи
	06	Прием значения одного регистра, доступного для записи
	15	Прием за один запрос значений до 1968 смежных битовых полей, доступных для записи
	16	Прием за один запрос значений до 123 смежных регистров, доступных для записи
	22	Изменение содержимого заданного регистра, доступного для записи с использованием комбинации масок И, ИЛИ с текущим содержимым регистра для индивидуального сброса или установки бит регистра
	23	Прием и выдача за один запрос значений до 125 (выдача) и до 121 (прием) смежных регистров, доступных для записи

Области памяти, отображаемые на множества регистров и битовых полей сервера MODBUS, определяются пользователем библиотеки посредством объектов типа *F_VAR_DESCRIPTOR*, указатели на которые затем передаются функции *FwModbusServerInit()* в параметрах *pInputDescriptor* (для описателя входной, доступной по сети для чтения и записи области) и *pOutputDescriptor* (для описателя выходной, доступной по сети только для чтения области).

Регистровый адрес и количество коммуникационных объектов в сетевом запросе чтения или записи некоторого участка области памяти, определенного описателями *pOutputDescriptor* или *pInputDescriptor*, вычисляются по следующим формулам:

```
Address = ByteOffset/2
RegistersCount = BytesCount/2,
```

где:

Address – начальный регистровый адрес в запросе чтения или записи со стороны клиента;

RegistersCount – количество регистров в запросе чтения или записи со стороны клиента;

ByteOffset – смещение в области памяти, определенной *pInputDescriptor* или *pOutputDescriptor* в байтах, начиная с 0, с которого предполагается выполнить чтение или запись данных по сети;

BytesCount – количество байт, подлежащее чтению или записи по сети.

Например, для того, чтобы прочитать по сети переменную типа LREAL, которая расположена в области памяти, определенной *pOutputDescriptor*, по смещению 16 байт, требуется передать контроллеру следующий запрос:

```
0x04 0x00 0x08 0x00 0x04
```

Адрес битового поля в сетевом запросе чтения или записи некоторого участка области памяти, определенного описателями *pOutputDescriptor* или *pInputDescriptor* с точностью до бита, вычисляется по следующей формуле:

```
DiscreteInputOrCoilAddress = BitOffset
```

где:

DiscreteInputOrCoilAddress – начальный адрес битового поля в запросе чтения или записи со стороны клиента;

BitOffset – битовое смещение начала участка, поддежащего чтению или записи.

Количество битовых полей в сетевом запросе должно совпадать с количеством бит, подлежащих чтению или записи.

Например, для того, чтобы прочитать по сети два бита, расположенных в области памяти, определенная *pOutputDescriptor*, по смещению 3 бита, требуется передать контроллеру следующий запрос:

```
0x02 0x00 0x03 0x00 0x02
```

Обратите внимание на то, что в соответствии со спецификацией протокола, значение адреса регистра, вводимое в конфигурации некоторых клиентов MODBUS, на 1 больше значения адреса, передаваемого в сетевом запросе.

6.5.3.2. Обмен данными между задачами и сервером

Механизм обмена данными с сервером MODBUS, реализуемый библиотекой *FastwelModbusServer.lib*, в точности совпадает с механизмом межзадачного обмена, описанным в п. 4.2.4.1. При инициализации сервера функцией *FwModbusServerInit()*для переменной, описываемой параметров *pOutputDescriptor*, которая будет выступать в качестве источника данных для сервера:

1. Для связанной с переменной задачи (определяется по индексу программной единицы POU, содержащей описываемую переменную) создается выходной порт.
2. Для сервера MODBUS, который будет выступать в качестве получателя данных задачи, создается входной порт.
3. Создается канал обмена с отдельным буфером, размер которого равен размеру связываемой переменной.
4. Выходной порт задачи связывается с каналом в качестве источника, а входной порт сервера – в качестве приемника данных.

Аналогично, для связываемой переменной, которая будет выступать в качестве приёмника данных от сервера:

1. Для связанной с переменной задачи (определяется по индексу программной единицы POU, содержащей описываемую переменную) создается входной порт.
2. Для сервера MODBUS, который будет выступать в качестве источника данных для задачи, создается выходной порт.
3. Создается канал обмена с отдельным буфером, размер которого равен размеру связываемой переменной.
4. Входной порт задачи связывается с каналом в качестве источника, а выходной порт сервера – в качестве приемника данных.

В процессе работы задача, чья переменная связана с сервером MODBUS в качестве приемника данных, перед вызовом корневой программной единицы последовательно читает все свои входные порты, среди которых также оказывается и порт, созданный при вызове *FwModbusServerInit()*. При чтении порта, когда доступ по записи к связанному с ним каналу блокирован, значение, находящееся в буфере канала, копируется в переменную-приемник данных.

Задача, чья переменная связана с сервером MODBUS в качестве источника данных, после вызова корневой программной единицы последовательно пишет во все свои выходные порты, среди которых оказывается и созданный при вызове *FwModbusServerInit()*. При записи в выходной порт, когда доступ по чтению к связанному с ним каналу блокирован, значение переменной-источника данных копируется в буфер канала.

6.5.3.3. Обслуживание сетевых запросов

Сервис сервера MODBUS активизируется при возникновении прерывания от коммуникационного порта по приему запроса от мастера сети.

Запрос чтения одного или нескольких регистров (битовых полей) приводит к тому, что буферизованные значения, ранее выведенные из области памяти связанной с сервером переменной в буфер канала обмена, упаковываются в ответное сообщение и передаются по сети мастеру.

Запрос записи одного или нескольких регистров (битовых полей) приводит к тому, что поступившие значения буферизируются в канале обмена. Копирование в переменную-приемник данных из буфера канала будет произведено перед последующим вызовом связанной с ней задачи.

6.5.3.4. Диагностика

Механизм получения диагностической информации от сервера в точности совпадает с механизмом обмена, описанным в п. 6.5.3.2. Переменная структурного типа *F_MODBUS_SERVER_DIAGNOSTICS* - приемник данных диагностики, определяется пользователем библиотеки посредством описателя – объекта типа *F_VAR_DESCRIPTOR*, указатель на который передается затем функции *FwModbusServerInit()* на параметре *pDiagnosticsDescriptor*.

Структура *F_MODBUS_SERVER_DIAGNOSTICS* определена следующим образом:

```
TYPE F_MODBUS_SERVER_DIAGNOSTICS :
STRUCT
   (* Идентификатор состояния сервера *)
   Status : F_MODBUS_SERVER_STATUS;
   (* Код последней ошибки. *)
   LastErrorCode : WORD;
   (* Счетчик транзакций. *)
   TransactionsCount : DWORD;
   (* Счетчик коммуникационных ошибок. *)
   CommunicationErrorsCount : DWORD;
   (* Счетчик исключений MODBUS. *)
   ExceptionErrorCount : DWORD;
END_STRUCT
END_TYPE
```

F_MODBUS_SERVER_STATUS перечислимый тип, принимающий значения:

Код	Значение
MB_SERVER_STATUS_UNDEFINED	Зарезервированное значение, которое имеет вновь созданный объект пока с ним не выполнено никаких действий
MB_SERVER_STATUS_INITIALIZED	Объект сервера инициализирован.
MB_SERVER_STATUS_READY	Сервер сконфигурирован.
MB_SERVER_STATUS_STARTED	Сервер запущен.

При поступлении по сети корректного запроса протокола Modbus значение *TransactionsCount* увеличивается на 1. При обнаружении коммуникационной ошибки (четность, фрагментация, контрольная сумма и т.п.) значение *CommunicationErrorsCount* увеличивается на 1. При получении запроса, вызвавшего исключение MODBUS (неверный адрес, неподдерживаемый тип функции, недопустимое значение параметра и т.п.) значение *ExceptionErrorCount* увеличивается на 1.

6.6. Библиотека FastwelUtils.lib

6.6.1. FwCheckSum16

Функция предназначена для вычисления 16-разрядной контрольной суммы содержимого участка памяти, начальный адрес которого и размер переданы в качестве первого и второго параметров соответственно. Обратите внимание, что размер участка ограничен максимальным значением типа WORD: 16#FFFF.

```
FUNCTION FwCheckSum16 : WORD
VAR_INPUT
   pBuffer : POINTER TO BYTE;
   bufferLength : WORD;
```

```
   startChecksum : WORD;
END_VAR
   ;
END_FUNCTION
```

Входные параметры:

pBuffer : POINTER TO BYTE
Указатель на участок памяти, для содержимого которого требуется вычислить контрольную сумму.

bufferLength : WORD
Размер участка.

startChecksum : WORD
Начальное значение контрольной суммы. Данный параметр может использоваться при вычислении общей контрольной суммы нескольких несмежных участков памяти.

Возвращаемый результат:

16-разрядная сумма всех байт заданного участка памяти плюс startChecksum.

6.6.2. FwCheckSum32

Функция предназначена для вычисления 32-разрядной контрольной суммы содержимого участка памяти, начальный адрес которого и размер переданы в качестве первого и второго параметров соответственно.

```
FUNCTION FwCheckSum32 : DWORD
VAR_INPUT
   pBuffer : POINTER TO BYTE;
   bufferLength : DWORD;
   startChecksum : DWORD;
END_VAR
   ;
END_FUNCTION
```

Входные параметры:

pBuffer : POINTER TO BYTE
Указатель на участок памяти, для содержимого которого требуется вычислить контрольную сумму.

bufferLength : DWORD
Размер участка.

startChecksum : DWORD
Начальное значение контрольной суммы. Данный параметр может использоваться при вычислении общей контрольной суммы нескольких несмежных участков памяти.

Возвращаемый результат:

32-разрядная сумма всех байт заданного участка памяти плюс startChecksum.

6.6.3. FwCRC16

Функция предназначена для вычисления 16-разрядной циклической контрольной суммы содержимого участка памяти, начальный адрес которого и размер переданы в качестве первого и второго параметров соответственно. Обратите внимание, что размер участка ограничен максимальным значением типа WORD: 16#FFFF.

Циклическая контрольная сумма вычисляется в соответствии с п. 6.2.2 спецификации *MODBUS over Serial Line. Specification and Implementation Guide. V1.02*.

```
FUNCTION FwCRC16 : WORD
VAR_INPUT
   pBuffer : POINTER TO BYTE;
   bufferLength : WORD;
END_VAR
   ;
END_FUNCTION
```

Входные параметры:

pBuffer : POINTER TO BYTE
Указатель на участок памяти, для содержимого которого требуется вычислить CRC16.

bufferLength : WORD
Размер участка.

Возвращаемый результат:

16-разрядная CRC всех байт заданного участка памяти.

6.6.4. FwCRC32

Функция предназначена для вычисления 32-разрядной циклической контрольной суммы содержимого участка памяти, начальный адрес которого и размер переданы в качестве первого и второго параметров соответственно.

Циклическая контрольная сумма вычисляется по алгоритму, описанному в следующем разделе Wikipedia: http://ru.wikipedia.org/wiki/CRC#CRC-32.

```
FUNCTION FwCRC32 : DWORD
VAR_INPUT
   pBuffer : POINTER TO BYTE;
   bufferLength : DWORD;
   startCRC : DWORD;
END_VAR
   ;
END_FUNCTION
```

Входные параметры:

pBuffer : POINTER TO BYTE
Указатель на участок памяти, для содержимого которого требуется вычислить CRC16.

bufferLength : DWORD
Размер участка.

startCRC : DWORD
Начальное значение CRC32.

Возвращаемый результат:

32-разрядная CRC всех байт заданного участка памяти, вычисленная по формуле: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

6.6.5. FwIsPOUExist

Функция возвращает ненулевое значение, если POU (программа, функция, функциональный блок) с индексом, заданным параметром POUIndex, имеется в текущем проекте, исполняемом в контроллере.

```
FUNCTION FwIsPOUExist : WORD
VAR_INPUT
   POUIndex : WORD;
END_VAR
   ;
END_FUNCTION
```

Входные параметры:

POUIndex : WORD
Индекс программной единицы, который может быть получен при помощи операции INDEXOF().

Возвращаемый результат:

Ненулевое значение, если программная единица с заданным индексом имеется в проекте, загруженном в контроллер.

6.6.6. FwGetPOU_CRC32

Функция возвращает 32-разрядную циклическую контрольную сумму POU (программы, функции, функционального блока) с индексом, заданным параметром POUIndex.

```
FUNCTION FwGetPOU_CRC32 : DWORD
VAR_INPUT
   POUIndex : WORD;
```

```
END_VAR
   ;
END_FUNCTION
```

Входные параметры:

POUIndex : WORD
Индекс программной единицы, который может быть получен при помощи операции INDEXOF().

Возвращаемый результат:

32-разрядная CRC всех байт программной единицы с индексом POUIndex, вычисленная по формуле:
$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

6.6.7. FwMemCompare

Функция побайтно сравнивает содержимое двух участков памяти размером, переданным в качестве третьего параметра, и возвращает 0, если их содержимое совпадает.

Если содержимое участков отличается, возвращает ненулевое значение.

Если хотя бы один из передаваемых указателей или длина равны 0, возвращает 16#FFFF.

6.6.8. FwMemCopy

Функция копирует содержимое участка памяти размером size, определяемого указателем pSource, в участок pDestination. Оба участка должны быть расположены в области внутренних переменных приложения CoDeSys.

Функция возвращает TRUE тогда, и только тогда, когда:

1. ни один из параметров не равен нулю;
2. оба участка принадлежат сегменту внутренних (глобальных) переменных приложения CoDeSys.

6.7. SysLibGetAddress.lib

6.7.1. Общие сведения

Библиотека поддерживается в системном программном обеспечении контроллера, начиная с версии 2.44.23922, и содержит две функции, позволяющие получить адреса и размеры сегментов данных приложения. Для идентификации сегментов используется библиотечный перечислимый тип ADDRESS_SEGMENTS:

```
TYPE ADDRESS_SEGMENTS :
(
   DATAID_MEMORY,
   DATAID_INPUT,
   DATAID_OUTPUT,
   DATAID_RETAIN,
   DATAID_GLOBVARS
);
END_TYPE
```

Значение DATAID_MEMORY идентифицирует сегмент флагов и данных, прямо адресуемых при помощи конструкции %M.

Значения DATAID_INPUT и DATAID_OUTPUT идентифицируют сегменты входных и выходных данных соответственно.

Значение DATAID_RETAIN в контроллерах серии CPM70x не используется.

Значение DATAID_GLOBVARS идентифицирует сегмент глобальных данных приложения, включая переменные VAR_GLOBAL и все внутренние переменные программ и функциональных блоков, отличные от VAR_INPUT, VAR_OUTPUT и VAR TEMP.

6.7.2. SysLibGetAddress

Функция возвращает указатель на начало сегмента данных приложения, идентификатор которого передан функции в качестве параметра. Для идентификатора DATAID_RETAIN всегда возвращается 0.

Для идентификаторов DATAID_INPUT и DATAID_OUTPUT возвращаются указатели на сегменты входных и выходных данных соответственно. Эти сегменты непосредственно используются кодом приложения и частично связаны с областями входных и выходных данных образа процесса, если в коде приложения имеются ссылки на соответствующие участки. Более подробная информация приведена в п. 4.2.4.1 настоящего руководства.

6.7.3. SysLibGetSize

Функция возвращает размер (в байтах) сегмента данных приложения, идентификатор которого передан функции в качестве параметра. Для идентификатора DATAID_RETAIN всегда возвращается 0.

Для идентификаторов DATAID_MEMORY и DATAID_GLOBVARS возвращаются **полные** размеры сегментов флагов и глобальных данных (2048 и 32768 соответственно), независимо от того, каковы реальные суммарные размеры размещенных в них переменных приложения.

Для идентификаторов DATAID_INPUT и DATAID_OUTPUT возвращаются размеры входного и выходного сегментов, равные размерам входной и выходной областей образа процесса, которые определены на приложения в ресурсе PLC Configuration.

6.8. Библиотека FastwelModbusRTUClientSerial.lib

6.8.1. Назначение

Библиотека FastwelModbusRTUClientSerial.lib написана на языке Structured Text в среде CoDeSys 2.3 и позволяет работать в режиме клиента (мастера) сети MODBUS RTU через любой коммуникационный порт, доступный функциям библиотеки FastwelSysLibCom.lib.

В состав библиотеки входят следующие функциональные блоки:

1. MODBUS_CLIENT_SERIAL – реализует основные функции мастера протокола MODBUS RTU.
2. MODBUS_PI_SERIAL – реализует интерфейс между приложением CoDeSys 2.3 и основным блоком библиотеки MODBUS_CLIENT_SERIAL.
3. SERIAL_COMM_FB – выполняет формирование, отправку, прием и разбор принятых MODBUS-запросов через коммуникационный порт с помощью функций библиотеки FastwelSysLibCom.lib.

Для работы в сети MODBUS RTU клиенту требуется регулярно вызывать функциональный блок MODBUS_PI_SERIAL из приложения CoDeSys.

Пример использования библиотеки имеется в проекте fw_modbus_client_test.pro для контроллера CPM702 и находится в подкаталоге Examples каталога установки адаптации CoDeSys 2.3 для Fastwel I/O (по умолчанию c:\Program Files\Fastwel\Fastwel CoDeSys Adaptation).

Наиболее актуальную спецификацию протокола MODBUS over Serial Line можно загрузить с Web-узла http://www.modbus.org.

6.8.2. Принцип работы

6.8.2.1. Переменные приложения и объекты сети MODBUS

Данные, которыми клиент сети MODBUS может обмениваться с сервером, могут быть отнесены к одной из 4-х областей: область MB_INPUT_REGISTER 16-разрядных регистров для чтения, область MB_HOLDING_REGISTER 16-разрядных регистров для чтения и записи, область MB_DISCRETE_INPUT битовых регистров для чтения, область MB_COIL битовых регистров для чтения и записи. В каждой такой области могут быть доступны до 65535 регистров. Каждый узел сети содержит набор индивидуальных таблиц для отображения своих переменных на области регистров

MODBUS. Для чтения интересующего сетевого объекта у удаленного подчиненного узла необходимо в теле запроса указать тип области, смещение и количество регистров.

Для связывания переменных приложения с коммуникационными объектами различных подчиненных узлов в библиотеке FastwelModbusRTUClientSerial.lib используются структуры типа MODBUS_ITEM_DESCRIPTOR:

```
TYPE MODBUS_ITEM_DESCRIPTOR:
STRUCT
   RemoteAddr:          BYTE;
   LocationType:        MB_LOCATION_TYPE;
   LocationAddr:        WORD;
   ItemsCount:          WORD;
   fWrite:              BOOL;
   pabyData:            POINTER TO BYTE;
   status:              MODBUS_CLIENT_STATUS;
END_STRUCT
END_TYPE
```

Поле *RemoteAddr* определяет адрес сервера-подчиненного узла сети MODBUS, которому должен быть направлен запрос.

Поле *LocationType* принимает одно значение из набора MB_INPUT_REGISTER, MB_DISCRETE_INPUT, MB_HOLDING_REGISTER, MB_COIL и определяет тип коммуникационного объекта MODBUS.

Поле *LocationAddr* определяет адрес коммуникационного объекта в выбранной области *LocationType*, которому будет поставлена в соответствие переменная или группа переменных приложения по адресу *pabyData*.

Поле *ItemsCount* определяет количество коммуникационных объектов (регистров MODBUS), которые соответствуют переменной или группе переменных приложения по адресу *pabyData*.

Поле *fWrite* определяет тип операции, которую необходимо провести над переменной или группой переменных приложения. Установка поля *fWrite* в *TRUE* показывает, что *ItemsCount* регистров, располагающихся в приложении по адресу *pabyData*, должны быть переданы по протоколу MODBUS подчиненному узлу *RemoteAddr* и записаны в его область *LocationType* со смещением *LocationAddr*. Установка поля *fWrite* в *FALSE* показывает, что *ItemsCount* регистров должны быть считаны по протоколу MODBUS с подчиненного узла *RemoteAddr* из области *LocationType* со смещением *LocationAddr* и скопированы в память клиента по адресу *pabyData*.

Поле *pabyData* содержит адрес переменной приложения, значение которой передается или принимается по сети. При чтении регистров MODBUS подчиненного узла с адресом *RemoteAddr* их содержимое будет размещено в памяти приложения по адресу *pabyData*. При записи содержимое памяти приложения по адресу *pabyData* передается подчиненному узлу.

Поле *status* показывает результат выполнения последней операции над объектом.

Каждая структура типа MODBUS_ITEM_DESCRIPTOR определяет однозначное соответствие между переменной приложения CoDeSys 2.3 и некоторой областью регистров в памяти удаленного узла MODBUS.

При инициализации в библиотеку загружается адрес массива, состоящего из структур MODBUS_ITEM_DESCRIPTOR. Последовательность таких структур позволяет связать локальные переменные приложения клиента с выбранными областями MODBUS-регистров удаленных подчиненных узлов и определить последовательность запросов, которую клиент будет циклически отправлять в сеть.

Запросы, определяемые с помощью массива дескрипторов, будут поочередно отсылаться в коммуникационный порт в порядке следования дескрипторов в массиве. По завершении последнего по счету запроса функциональный блок MODBUS_PI_SERIAL библиотеки переходит к первому элементу массива дескрипторов и вновь начинает его отправку.

6.8.2.2. Взаимодействие с библиотекой

Функциональный блок MODBUS_PI_SERIAL обеспечивает взаимодействие библиотеки FastwelModbusRTUClientSerial.lib с приложением CoDeSys и осуществляет обход массива дескрипторов с передачей соответствующих запросов блоку MODBUS_CLIENT_SERIAL для обработки.

```
VAR_INPUT
   fInit:                    BOOL;
   fSegmented:               BOOL  := FALSE;
   timeToWait:               TIME  := T#200ms;
   intervalTimeout:          TIME  := T#10ms;
   pItemDescriptorLists:     POINTER TO MODBUS_ITEM_DESCRIPTOR;
   DescriptorsCount:         BYTE;
   comNmb:                   INT   := 1;
   dwBaudrate:               DWORD := 115200;
   byStopbits:               BYTE  := 0;
   byParity:                 BYTE  := 0;
END_VAR
```

Вход *fInit*, находящийся в состоянии *TRUE*, является сигналом для однократной инициализации внутренних структур данных блока. При этом в памяти функционального блока сохраняются значения на всех управляющих входах. Все дальнейшие вызовы блока следует производить со входом *fInit*, находящимся в состоянии *FALSE*.

Вход *fSegmented* в состоянии TRUE предписывает разбивать сетевые запросы записи общим размером более 32-х байт на отдельные запросы длиной не более 32-х байт.

Вход *timeToWait* определяет интервал времени, в течение которого ожидается ответ от каждого подчиненного узла.

Вход *intervalTimeout* определяет интервал времени, используемый для определения границы ответа на запрос. Отсутствие приема данных через коммуникационный порт в течение времени *intervalTimeout* после предшествующего приема нескольких байт будет воспринято функциональным блоком MODBUS_CLIENT_SERIAL как конец ответа на запрос.

Вход *pItemDescriptorLists* содержит начальный адрес массива структур типа MODBUS_ITEM_DESCRIPTOR. Массив описывает последовательность запросов и связывает переменные приложения клиента с коммуникационными объектами подчиненного узла MODBUS.

Вход *DescriptorsCount* содержит размер массива *pItemDescriptorLists*.

Вход *comNmb* содержит номер коммуникационного порта, через который должны отсылаться запросы. Допустимые значения данного параметра перечислены в п. 6.4.2.1 настоящего руководства.

Вход *dwBaudrate* определяет скорость работы через коммуникацонный порт и принимает значение из набора 4800, 9600, 19200, 38400, 57600, 115200.

Вход *byStopbits* определяет количество стоповых бит при работе через коммуникационный порт и принимает значение из набора 0 (1 стоповый бит), 1 (1.5 стоповых бита), 2 (2 стоповых бита).

Вход *byParity* определяет количество бит четности при работе через коммуникационный порт и принимает значение из набора 0 (нет бит четности), 1 (дополнение до нечетного), 2 (дополнение до четного).

Вызов экземпляра блока MODBUS_PI_SERIAL необходимо произвести однократно со входом *fInit* в состоянии *TRUE*, записав корректные значения на остальные входы. Далее блок необходимо вызывать периодически, подавая *FALSE* на вход *fInit*.

6.8.2.3. Инициализация библиотеки

Для начала работы в качестве клиента (мастера) сети MODBUS требуется объявить в памяти приложения CoDeSys массив описателей запросов.

Например, пусть приложению требуется передать какую-либо переменную *data_to_server* подчиненному узлу 5 сети MODBUS, а затем прочитать значение обратно с сервера в переменную *data_from_server*. Тогда массив дескрипторов будет содержать запросы записи и чтения:

```
VAR
  Modbus_Item_Array: ARRAY [0..1] OF MODBUS_ITEM_DESCRIPTOR;
END_VAR
```

Затем необходимо инициализировать массив

```
Modbus_Item_Array[0].RemoteAddr      := 5;
Modbus_Item_Array[0].LocationType    := MB_HOLDING_REGISTER;
Modbus_Item_Array[0].LocationAddr    := 0;
Modbus_Item_Array[0].ItemsCount      := SIZEOF(data_to_server)/2;
Modbus_Item_Array[0].fWrite          := TRUE;
Modbus_Item_Array[0].pabyData        := ADR(data_to_server);
```

```
Modbus_Item_Array[1].RemoteAddr     := 5;
Modbus_Item_Array[1].LocationType   := MB_HOLDING_REGISTER;
Modbus_Item_Array[1].LocationAddr   := 0;
Modbus_Item_Array[1].ItemsCount     := SIZEOF(data_from_server)/2;
Modbus_Item_Array[1].fWrite         := FALSE;
Modbus_Item_Array[1].pabyData       := ADR(data_from_server);
```

и вызвать однократно (например, по событию *OnInit*) экземпляр *ModbusPI* функционального блока MODBUS_PI_SERIAL с корректными параметрами:

```
ModbusPI
(
  fInit                := TRUE,
  pItemDescriptorLists := ADR(Modbus_Item_Array[0]),
  DescriptorsCount     := SIZEOF(ModbusItemArray) / SIZEOF(ModbusItemArray[0]),
  comNmb               := 102,
  dwBaudrate           := 115200,
  byStopbits           := 0,
  byParity             := 0
);
```

В циклически исполняемой задаче CoDeSys требуется обеспечить вызов экземпляра *ModbusPI* функционального блока MODBUS_PI_SERIAL, установив вход *fInit* в состояние *FALSE*:

```
…
ModbusPI
(
  fInit:= FALSE
);
…
```

Результатом работы приложения CoDeSys в приведенном примере станет:

1. Запись клиентом сети в удаленный подчиненный узел с адресом 5 регистров MODBUS в количестве *SIZEOF(data_to_server)/2* в область MB_HOLDING_REGISTER по смещению 0.
2. Чтение клиентом у удаленного узла с адресом 5 регистров MODBUS в количестве *SIZEOF(data_from_server)/2* из области MB_HOLDING_REGISTER по смещению 0. Прочитанные данные будут записаны в переменную *data_from_server* приложения CoDeSys.

6.9. Библиотека FastwelPlatformControl.lib

6.9.1. Общие сведения

Данная библиотека содержит сервисные системные функции, включая функции записи и чтения пользовательского серийного номера контроллера (*FwPlatformSetSerialNumber* и *FwPlatformGetSerialNumber*), а также функцию сброса контроллера из приложения (*FwPlatformReset*).

6.9.2. Описание функций

6.9.2.1. FwPlatformGetTemperature

Функция не поддерживается на контроллерах CPM701, CPM702, CPM703, CPM704 и возвращает значение F_PLCTL_NOT_SUPPORTED.

6.9.2.2. FwPlatformSetSerialNumber

Функция позволяет устанавливать серийный номер контроллера с целью его дальнейшей идентификации. Значение серийного номера в заводской поставке и после сброса контроллера в “заводской режим” равно 0.

```
FUNCTION FwPlatformSetSerialNumber : F_PLCTL_RESULT
VAR_INPUT
   NewValue :   DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

```
NewValue :      DWORD;
```

Значение серийного номера, устанавливаемое для контроллера.

Возвращаемый результат:

Значение перечислимого типа F_PLCTL_RESULT:

Значение	Описание
F_PLCTL_OK	Успешное завершение.
F_PLCTL_NOT_SUPPORTED	Операция не поддерживается
F_PLCTL_GENERIC	Системная ошибка.

6.9.2.3. FwPlatformGetSerialNumber

Функция считывает значение серийного номера контроллера. Значение серийного номера в заводской поставке и после сброса контроллера в “заводской режим” равно 0.

```
FUNCTION FwPlatformGetSerialNumber : F_PLCTL_RESULT
VAR_INPUT
   pValue : POINTER TO DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

```
pValue : POINTER TO DWORD;
```

Адрес переменной, в которую будет записано значение серийного номера контроллера.

Возвращаемый результат:

Значение перечислимого типа F_PLCTL_RESULT:

Значение	Описание
F_PLCTL_OK	Успешное завершение.
F_PLCTL_NOT_SUPPORTED	Операция не поддерживается
F_PLCTL_INCORRECT_PARAM	Недопустимое значение параметра (pValue = 0)
F_PLCTL_GENERIC	Системная ошибка.

6.9.2.4. FwPlatformReset

Функция позволяет производить сброс (перезапуск) программы контроллера.

```
FUNCTION FwPlatformReset : F_PLCTL_RESULT
VAR_INPUT
   ResetMode : F_RESET_MODE;
   pMsg : POINTER TO STRING;
END_VAR
;
END_FUNCTION
```

Входные параметры:

ResetMode : F_RESET_MODE;

Значение перечислимого типа, определяющее режим работы контроллера после сброса.

Значение	Описание
F_RESET_WARM	Выполняется перезапуск программы контроллера. После перезапуска производится инициализация переменных программы. Аналогичные действия выполняются при исполнении команды **Online–Reset** из меню главного окна IDE CoDeSys.
F_RESET_COLD	Действие идентично выполняемому при вызове данной функции с параметром F_RESET_WARM. Аналогичные действия выполняются при исполнении команды **Online–Reset(cold)** из меню главного окна IDE CoDeSys.
F_RESET_SAFE	Выполняется перезапуск программы контроллера с переходом в безопасный режим.
F_RESET_ORIGINAL	Выполняется сброс программы и всех настроек контроллера в заводской режим и затем перезапуск контроллера. Аналогичные действия выполняются при исполнении команды **Online–Reset(original)** из меню главного окна IDE CoDeSys.

pMsg : POINTER TO STRING;

Дополнительный параметр, который используется только в случае перевода программы контроллера в “Безопасный режим” – адрес строки с краткой текстовой информацией о причине перехода в “Безопасный режим”, которая будет записана в файл *normdump.txt*. Допускается нулевое значение.

Возвращаемый результат:

Значение перечислимого типа F_PLCTL_RESULT:

Значение	Описание
F_PLCTL_OK	Успешное завершение.
F_PLCTL_NOT_SUPPORTED	Операция не поддерживается
F_PLCTL_INCORRECT_PARAM	Недопустимое значение параметра (ResetMode)
F_PLCTL_GENERIC	Системная ошибка.

6.9.2.5. Пример

```
(*****************************************************************
Примеры использования функций библиотеки FastwelPlatformControl.lib
*****************************************************************)
PROGRAM PLC_PRG

VAR
   (* Переменая для записи результата вызова функций библиотеки
   FastwelPlatformControl.lib *)
   plResult : FW_PLCTL_RESULT;
   (* Значение температурного датчика контроллера *)
   plTemperature : WORD;
   (* Значение серийного номера контроллера *)
   plSerialNumber : DWORD;
   (* Кол-во вызовов, завершившихся ошибкой *)
   ErrorCount : DWORD := 0;
END_VAR

VAR CONSTANT
   SERIAL_NUMBER : DWORD := 7777;
   MSG_DEBUG : STRING := 'Debug Info';
END_VAR
```

```
(* Код программы начинается здесь *)
(*. . .*)
(* Установка серийного номера *)
plResult := FwPlatformSetSerialNumber(7777);
(* проверяем результат операции *)
IF plResult <> F_PLCTL_OK THEN
   ErrorCount := ErrorCount + 1;
END_IF
(* Чтение серийного номера *)
plResult := FwPlatformGetSerialNumber(ADR(plSerialNumber));
(* проверяем результат операции *)
IF plResult <> F_PLCTL_OK THEN
   ErrorCount := ErrorCount + 1;
END_IF
(* Перевод контроллера в безопасный режим *)
FwPlatformReset(F_RESET_SAFE, ADR(MSG_DEBUG));
(* проверяем результат операции *)
IF plResult <> F_PLCTL_OK THEN
   ErrorCount := ErrorCount + 1;
END_IF
END_PROGRAM
```

ПРИЛОЖЕНИЕ 1. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Версия	Дата	Ссылка	Статус	Примечания
2.05.2394	07.04.2008	Документ	создан	Реструктуризация версии 1.22 и выпуск версии 2.0
2.06.2395	08.05.2008	п. 4.2.1.1	изменен	Добавлена информация об индикации ошибки, связанной с порчей памяти системных областей среды исполнения
		п. 4.3.5.2	изменен	Добавлена информация о состоянии полей *IOStatus*
		п. 5.7	изменен	Добавлена информация о возможной причине перехода в безопасный режим с диагностикой "Ошибка связывания"
2.07.2396	23.07.2008	п. 5.5	создан	Добавлена дополнительная информация о связи программ с окружением
2.09.23910	10.12.2008	Раздел 1	изменен	Добавлена информация о контроллере CPM704
		п. 2.4.1	изменен	Скорректированы характеристики подсистемы исполнения прикладных программ пользователя
		п. 3.6.1	изменен	Добавлена информация о необходимости обновления ПО контроллеров (кроме CPM704) до версии не ниже 2.09.2399 при использовании типа *I/O Modules (constant size)* поддерева **PLC Configuration**
		п. 4.3.2	изменен	Уточнено условие запуска обмена с модулями ввода-вывода в режиме *Group per Module* при обнаружении только части модулей из имеющихся в конфигурации
		п. 4.3.3.2	изменен	то же, что и в п. 4.3.2
		п. 4.3.5.1	изменен	Скорректированное значение периода опроса модулей в режиме *Group per Module* равное количеству модулей, умноженному на 1 мс
		п. 6.4.2.1	изменен	Для CPM704 при открытии коммуникационного порта из библиотеки FastwelSysLibCom.lib должен использоваться параметр COM2.
2.14.23911	16.04.2009	п. 6.5	создан	Добавлено описание библиотеки FastwelModbusServer.lib
2.20.23918	23.12.2009	п. 3.7	изменены	Уточнено описание процесса обновления системного ПО контроллеров.
2.31.23922	10.06.2010	п. 3.6	изменен	Удалена информация по обновлению системного ПО контроллера с версии 1.22 до версии 2 Описан способ выяснения номера версии системного ПО контроллеров
		п. 4.2.4.1	изменен	Удалена информация о необходимости ссылок на переменные во входной части образа процесса в приложении для мониторинга их значений в состоянии **Online–Login**. Добавлена информация об механизма ввода данных входной части образа процесса во входной сегмент приложения на контексте сервисной задачи.
		п. 4.2.4.2	изменен	Приведено описание однозадачного режима, когда пользователь создает единственную циклическую задачу в ресурсе **Task Configuration** или не создает ни одной задачи.
		п. 4.2.4.5	изменен	Приведена более подробная информация о небезопасном использовании глобальных переменных в многозадачном приложении.
		п. 5.3	изменен	Добавлено описание однозадачного режима.
		п. 5.4	изменен	Добавлена информация об автоматической коррекции периода при перегрузке системы по пользовательским вычислениям, выполняемым на контексте сервисной задачи.
		п. 5.8.4	изменен	Удалена информация о невозможности форсирования переменных, отображенных на выходную часть образа процесса.
		п. 6.3	изменен	Добавлена информация о функциях F_IecTasks_getCount и F_IecTasks_getInfo библиотеки FastwelTasksExchange.lib
		п. 6.6	создан	Добавлено описание библиотеки FastwelUtils.lib
2.40.23922	09.09.2010	п. 5.8.4	изменен	Добавлено замечание по форсированию или однократной записи переменных, отображенных на область выходных данных.
		п. 5.12	создан	Добавлено замечание о том, что трассировка значений переменных выполняется на контексте циклической или ациклической задачи, для которой установлен признак DEBUG.

Версия	Дата	Ссылка	Статус	Примечания
2.44.23922	19.10.2010	п. 4.2.4.2	изменен	Уточнено значение и алгоритм функционирования в однозадачном режиме, когда в конфигурацию приложения ни добавлено ни одной задачи или добавлена одна циклическая задача
		п. 4.2.4.5	изменен	
		п. 5.4	изменен	
		п. 6.7	создан	Добавлено описание системной библиотеки SysLibGetAddress.lib
2.52.23926	18.03.2011	п. 2.3.3	изменен	Изменен рис. 2, иллюстрирующий циклограмму функционирования циклических задач
		п. 4.2.4.1	изменен	Для циклических задач уточнено понятие «укладываться в заданный период» Расширен перечень функций, выполняемых сервисной задачей.
		п. 5.5.2	изменен	Исправлены ошибки в примере
		п. 6.5.2	изменен	В таблице возвращаемых результатов изменено описание для кодов 6 и 7
2.59.23937	06.12.2012	п. 4.3.5	изменен	Удалена информация о диагностическом виртуальном канале модулей ввода-вывода
2.60.23938	19.03.2013	п. 2.6.1	изменен	Добавлена информация об адаптере PROFIBUS DP Molex DRL-PFB-USB, как необходимом средстве для удаленной загрузки и отладки приложений CoDeSys 2.3 по сети PROFIBUS DP
		п. 6.2.1	изменен	Добавлена информация об оценке времени износа флэш-диска
2.63.23944	07.08.2014	п. 4.2.4.5	изменен	Скорректировано описание параметров функции F_IecTasks_linkVariables
		п. 6.4.1	изменен	Добавлена информация об элементах конфигурации NIM741 RS-485 1xUART Stream Module и NIM742 RS-232 1xUART Stream Module и возможности доступа к коммуникационным портам 101–164 средствами библиотеки FastwelSysLibCom.lib.
		п. 6.4.2.1	изменен	Добавлена информация о значениях параметра функции FwSysComOpen, которые должны использоваться для доступа к портам, организованным через модули NIM741/NIM742 посредством элементов NIM741 RS-485 1xUART Stream Module и NIM742 RS-232 1xUART Stream Module в конфигурации контроллера.
		п. 6.5.1	изменен	Добавлена информация о возможности использования модулей NIM741/NIM742 в качестве коммуникационных портов сервера MODBUS.
		п. 6.5.2	изменен	Добавлена информация о значениях параметра функции FwModbusServerInit для использования модулей NIM741/NIM742 в качестве коммуникационных портов сервера MODBUS. Скорректировано описание аргументов и возвращаемых значений для FwModbusServerInit.
		п. 6.5.3.1	изменен	Скорректированы характеристики сервиса MODBUS, реализуемого библиотекой FastwelModbusServer.lib в части количества коммуникационных объектов в запросах 15, 16, 23
2.64.23946	22.12.2014	Документ	изменен	Обновлена контактная информация
		п. 2.5.1	изменен	Добавлена информация о компакт-диске Fastwel DVD.
		п. 2.6.2	изменен	Расширен перечень поддерживаемых операционных систем для рабочего места разработчика.
		п. 3.2	изменен	Скорректировано описание процесса установки пакета адаптации CoDeSys 2.3 для Fastwel I/O
		п. 3.6	изменен	Добавлена информация о программе **Сервисная утилита FASTWEL IO**.
		п. 6.1	изменен	Добавлены общие сведения о системных библиотеках FastwelModbusRTUClientSerial.lib и FastwelPlatformControl.lib.
		п. 6.8	создан	Добавлено описание системной библиотеки FastwelModbusRTUClientSerial.lib.
		п. 6.9	создан	Добавлено описание системной библиотеки FastwelPlatformControl.lib.
2.65.23947	25.06.2015	п. 4.2.1.1	изменен	Добавлена информация об установлении причины перехода контроллера в безопасный режим при помощи браузера ПЛК.

Версия	Дата	Ссылка	Статус	Примечания
		п. 4.3.6	создан	Добавлена информация о получении информации об обнаруженных подключенных модулях ввода-вывода при помощи браузера ПЛК.
		п. 5.8.2	изменен	Разделен на пп. 5.8.2.1 и 5.8.2.2. п. 5.8.2.2 содержит указания по защите контроллера от несанкционированного соединения со средой разработки CoDeSys 2.3
2.66.23947	22.10.2015	п. 4.2.1.1	изменен	Исправлена ошибка в последовательности подключения к контроллеру из среды разработки CoDeSys 2.3 в случае предложения загрузить изменившееся приложение. Вместо кнопки **Cancel (Отмена)** предлагается нажать **No (Нет)**.
		п. 4.3.6	изменен	
		п. 5.8.2.2	изменен	
		Документ	изменен	Скорректирована информация об изготовителе