Система ввода-вывода Fastwel I/O
Контроллеры CPM711/CPM712/CPM713

Руководство программиста

ИМЕС.00300-02 33 03-1

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 CPM711, CPM712 и CPM713 серии Fastwel I/O в среде CoDeSys 2.3 фирмы 3S Smart Software Solutions (далее CoDeSys).

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

1. ИМЕС.00300-02 33 03-2. CPM711. Контроллер узла сети CANopen. Руководство по конфигурированию и программированию сетевых средств
2. ИМЕС.00300-02 33 03-3. CPM712. Контроллер узла сети MODBUS RTU/ASCII. Руководство по конфигурированию и программированию сетевых средств
3. ИМЕС.00300-02 33 03-4. CPM713. Контроллер узла сети MODBUS TCP. Руководство по конфигурированию и программированию сетевых средств
4. ИМЕС.00300-02 33 03-6. Протокол DNP3. Руководство по конфигурированию и программированию.

Информация о конфигурировании модулей ввода-вывода Fastwel I/O приведена в документе *ИМЕС.00300-02 32 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 Vortex86DX фирмы DMP Electronics, совместимого с 80486 и имеющего тактовую частоту 600 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.

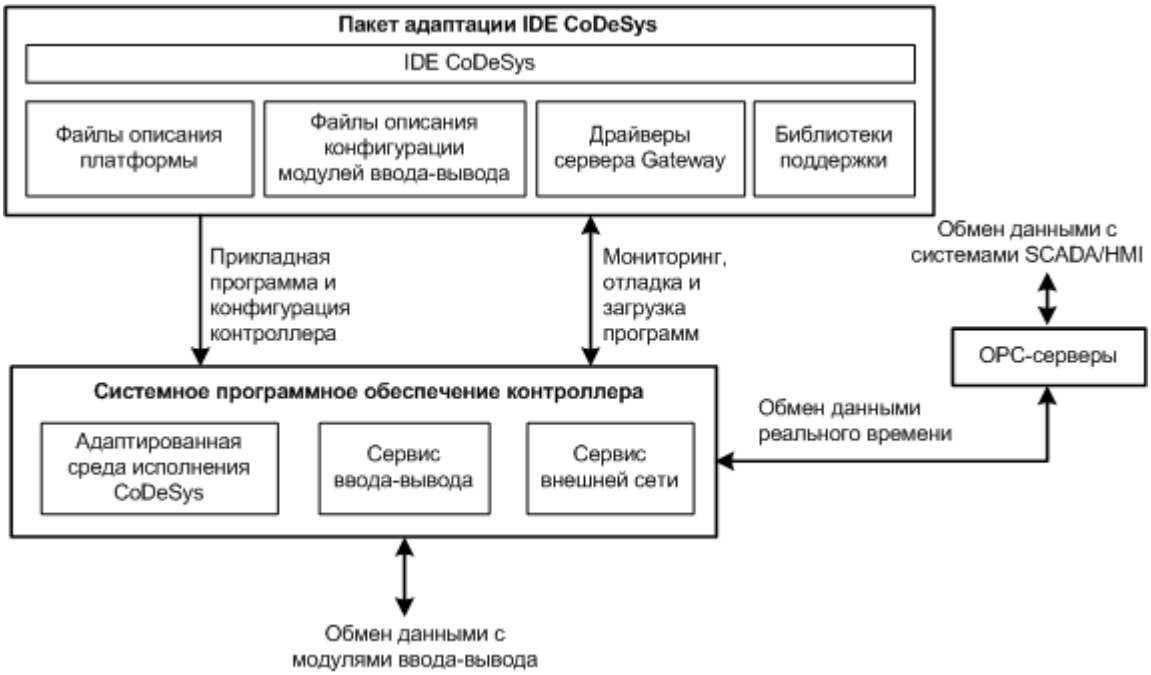


Рис. 1. Структура программного обеспечения Fastwel I/O

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 16-ти циклических и до 64-х 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.

Исполнение программы под управлением одной циклической задачи, которая реализует некоторый алгоритм сбора данных и управления, представлено на рис. 2. Исполнение одной ациклической задачи иллюстрируется рис. 3.

Для хранения данных приложения, разрабатываемого в среде разработки CoDeSys, после выключения питания среда исполнения контроллеров CPM71x имеет сегмент энергонезависимых переменных размером до 131056 байт, отображенный на статическую память с автономным питанием от батареи. Переменные, объявленные в секциях VAR RETAIN, размещаются компилятором CoDeSys в данном сегменте. Контроль исправности батареи может осуществляться в пользовательском приложении при помощи функции *SysRtcCheckBatery* внешней библиотеки SysLibRtc.lib,

подключаемой к проекту CoDeSys. Остальные функции библиотеки предназначены для получения и установки абсолютного времени встроенных часов реального времени.

Рис. 2. Исполнение циклической задачи

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		131072	
Размер области выходных переменных приложения	байт		131072	
Размер области внутренних переменных приложения	байт		2097152	
Размер области исполняемого кода приложения	байт			2097152
Размер области конфигурации прикладной программы	байт			2097152
Размер области энергонезависимых переменных	байт		131056	
Производительность				
Сложение и вычитание 2-байтовых операндов	опер/с		25800000	
Умножение 2-байтовых операндов	опер/с		20100000	
Деление 2-байтовых операндов	опер/с		9800000	
Сложение и вычитание целочисленных 4-байтовых операндов	опер/с		41000000	
Умножение целочисленных 4-байтовых операндов	опер/с		25000000	
Деление целочисленных 4-байтовых операндов	опер/с		10400000	
Сложение и вычитание операндов типа REAL	опер/с		439000	
Умножение операндов типа REAL	опер/с		430000	
Деление операндов типа REAL	опер/с		406000	
Сложение и вычитание операндов типа LREAL	опер/с		404000	
Умножение операндов типа LREAL	опер/с		406000	
Деление операндов типа LREAL	опер/с		373000	
Ядро системы исполнения				
Количество циклических задач	шт	1		16
Период циклической задачи	мс	2	10	1000
Количество уровней приоритета циклических задач				32
Размер стека циклической задачи	байт		16000	
Количество ациклических задач	ед	0		64
Количество уровней приоритета ациклических задач	ед			32
Размер стека ациклической задачи	байт		16000	
Количество единиц организации программы (POU)	шт	1		16384
Количество связей задачи с областью входных данных[1]	шт	0		8064
Количество связей задачи с областью выходных данных	шт	0		8064
Размер переменной типа STRING	байт		80	255
Количество одновременно используемых временных переменных типа STRING при работе со строковыми операциями из библиотеки STANDARD.LIB	шт			4096
Количество одновременно используемых временных переменных типа STRING при выполнении преобразований переменных примитивных типов в строки	шт			4096
Отладчик				
Количество устанавливаемых постоянных точек останова	шт			100
Количество промежуточных временных точек останова	шт			100
Глубина дерева вызовов при отладке				30
Взаимодействие со средой разработки				
Максимальный размер описаний сохраняемых (PERSISTENT) переменных[2]	кбайт			512
Размер буфера форсируемых переменных	байт			32768
Размер буфера считываемых переменных	байт			131072
Размер буфера трассировки	байт			131072
Таймаут между блоками транспортного протокола обмена со средой разработки	мс			10000

[1] 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] PERSISTENT (сохраняемые) переменные поддерживаются в системе исполнения контроллеров и пакете адаптации CoDeSys 2.3 для Fastwel I/O, начиная с версии 2.67.23949

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			150
в режиме *Group per Module* (одна группа на каждый модуль)	мкс	110		150
Период обмена данными	мс	2	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 DVD

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 вычислительных платформ *Fastwel CPM711 CANopen Programmable Controller, Fastwel CPM712 MODBUS RTU/ASCII Programmable Controller, Fastwel CPM713 MODBUS TCP Programmable Controller*.
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 вычислительных платформ Fastwel I/O для CoDeSys выполняется автоматически программой установки FastwelCoDeSysAdaptation.exe. 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 300 Мбайт;
- привод CD-ROM

Рекомендуемое разрешение монитора не менее 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 CPM711 по сети CAN компьютер должен быть оснащен адаптером сети CAN фирмы IXXAT (любым) либо адаптером PCAN-USB фирмы PEAK-Systems Technik.

Для использования адаптеров фирмы IXXAT на компьютер необходимо установить пакет программной поддержки VCI 2.20, который доступен на web-узле компании IXXAT.

Для работы с адаптером PCAN-USB фирмы PEAK-Systems Technik требуется загрузить пакет программной поддержки с web-узла компании, распаковать и запустить программу установки PeakOemDrv.exe, которая установить драйверы и библиотеки поддержки для работы с адаптером в операционных системах Windows XP/Vista/7.

Для удаленной отладки и загрузки программного обеспечения в контроллер CPM712 компьютер должен иметь в своем составе последовательный порт интерфейса RS-232C или RS-485.

Для удаленной отладки и загрузки программного обеспечения в контроллер CPM713 компьютер должен быть оснащен адаптером сети Ethernet 100 Мбит/с.

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 2.3.9.22, для чего запустите CoDeSys и выберите команду **About** меню **Help**.

В случае, если компьютер содержит установленный ранее пакет CoDeSys версии ниже 2.3.9.22, 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 (например, CoDeSys Gateway Server), 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** может пройти 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.

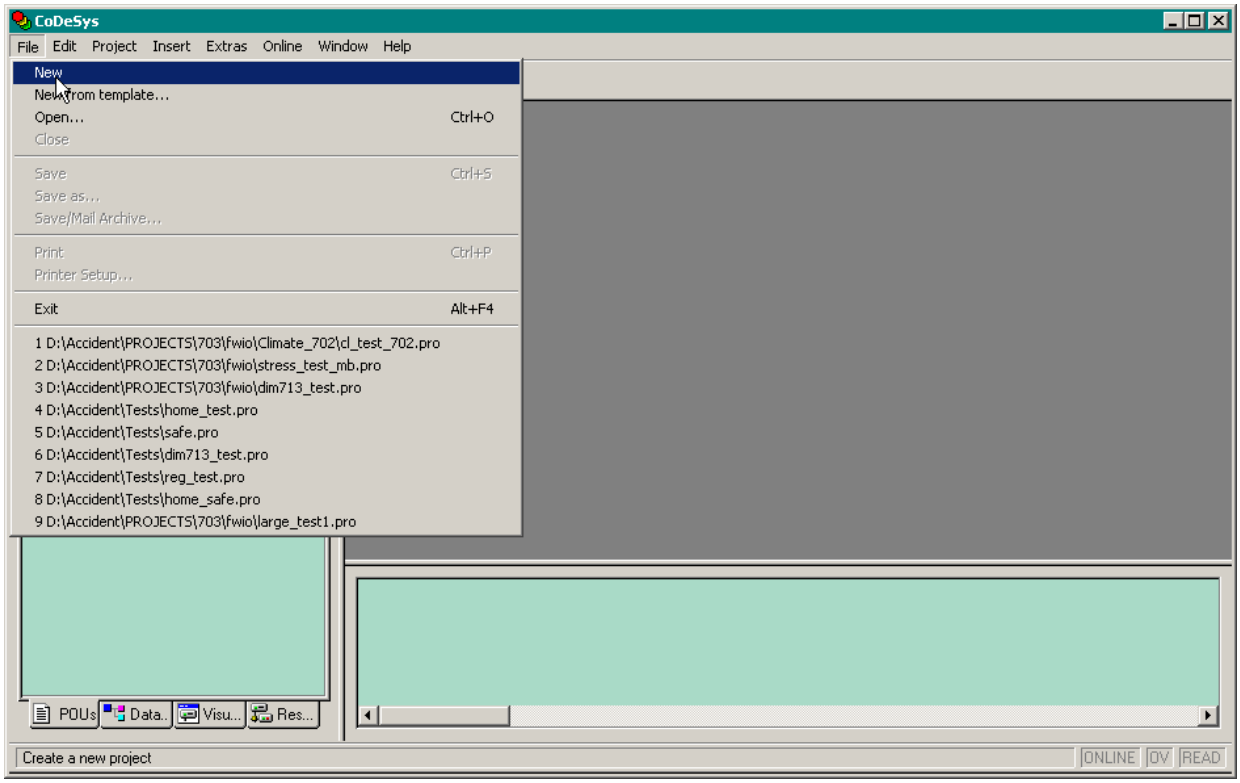


Рис. 4. Главное окно среды разработки CoDeSys

2. Выберите команду **New** в меню **File**. В появившейся диалоговой панели **Target Settings** (Параметры платформы) в выпадающем списке **Configuration** выберите опцию *Fastwel CPM71x ... Programmable Controller* (в зависимости от типа контроллера). Внешний вид диалоговой панели **Target Settings** примет вид, показанный на рис. 5.

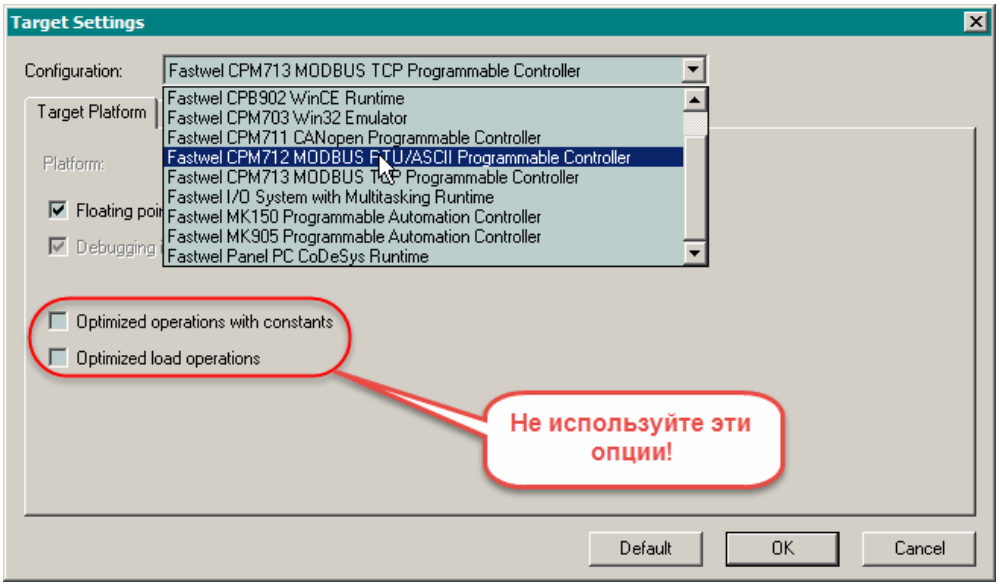


Рис. 5. Диалоговая панель Target Settings после выбора платформы

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.
4. В области ввода тела программы введите единственный символ ; и выполните команду **Project–Rebuild all**.

Если при выполнении п. 4 проект скомпилирован без ошибок, это свидетельствует об успешном завершении установки адаптации.

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 любой 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. 6.

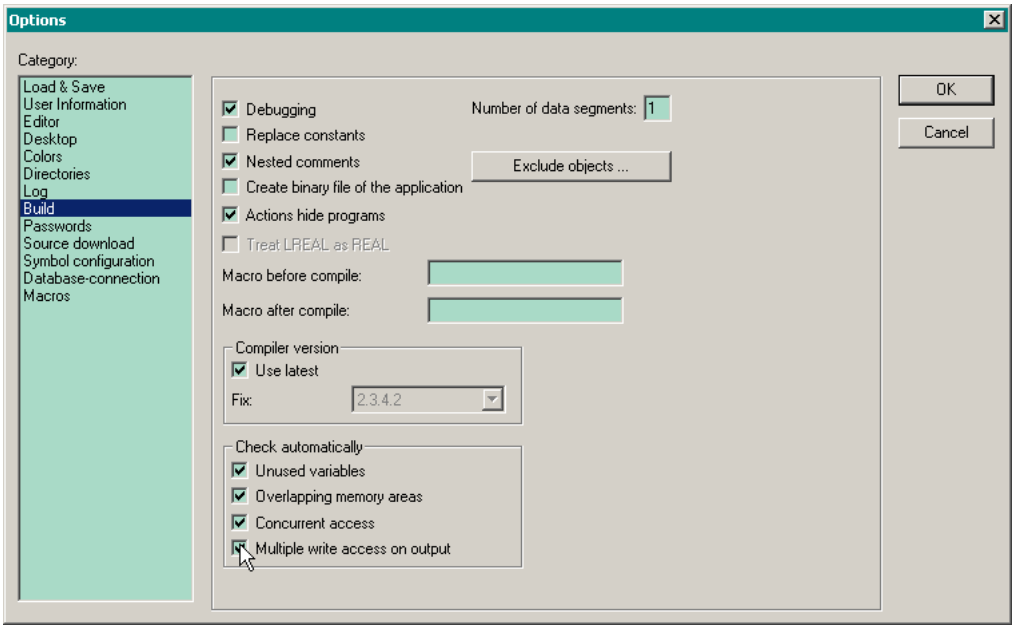


Рис. 6. Настройка параметров трансляции проекта IDE CoDeSys

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 CPM711:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM711

для контроллера CPM712:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM712

для контроллера CPM713:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Firmware/CPM713.

Для просмотра текущей версии в среде разработки CoDeSys 2.3:

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. 7.
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. 7.

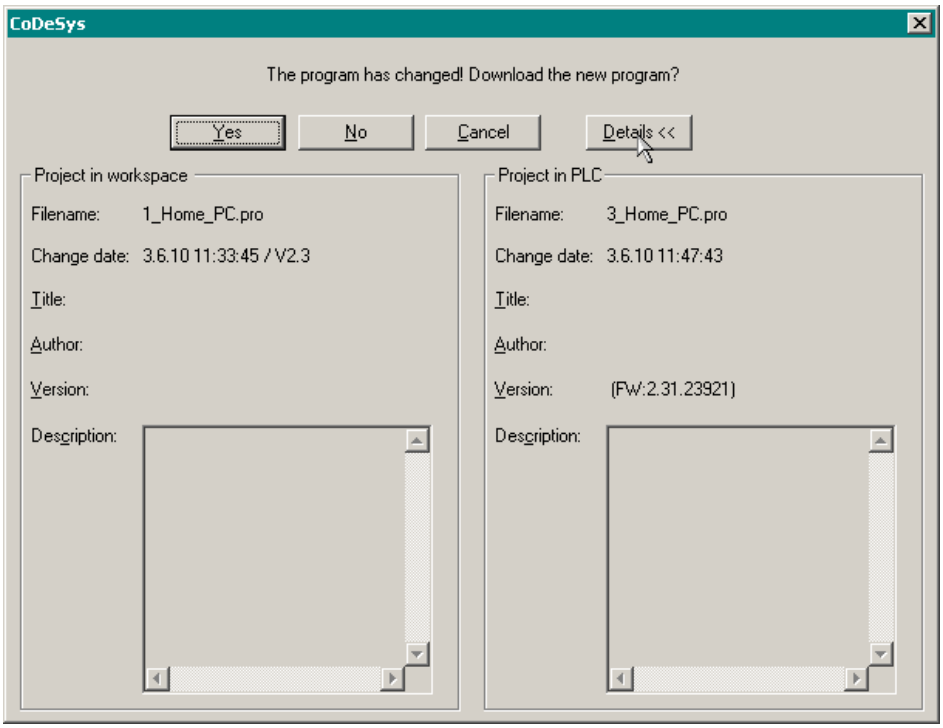


Рис. 7. Просмотр информации о версии системного ПО контроллера

3.7. Миграция проектов для контроллеров серии CPM70x на платформы CPM71x

Имеется возможность переноса проектов, созданных в среде CoDeSys для контроллеров CPM701/CPM702/CPM703 серии Fastwel I/O, на соответствующие контроллеры CPM711/CPM712/CPM713.

ВНИМАНИЕ!

Допускается перенос проектов только в следующем порядке:

с CPM701 на CPM711

с CPM702 на CPM712

с CPM703 на CPM713

После переноса проект будет компилироваться и функционировать на соответствующем контроллере CPM71x в режиме совместимости по конфигурации, ранее созданной пользователем в окне ресурса **PLC Configuration**. В режиме совместимости могут быть недоступны некоторые сетевые функциональные возможности выбранного контроллера CPM71x.

Для переноса проекта, ранее созданного для контроллера CPM701, на контроллер CPM711:

1. Откройте проект для контроллера CPM701 в среде CoDeSys
2. Щелкните на вкладке **Resources** и дважды щелкните на ресурсе *Target Settings* в дереве ресурсов проекта.
3. В появившейся диалоговой панели **Target Settings** в выпадающем списке **Configuration** выберите опцию *Fastwel CPM711 CANopen Programmable Controller*, как показано на рис. 8 и нажмите **OK**.
4. В появившейся диалоговой панели **CoDeSys: In case of different PLC configurations**... нажмите **Yes** (**Да**).
5. Выполните команды **Project-Clean all**, **Project–Rebuild all** в главном меню CoDeSys.
6. Сохраните проект командой **File–Save**.

Перенос проекта для CPM702 на CPM712 или CPM703 на CPM713 выполняется аналогично путем выбора опций *Fastwel CPM712 MODBUS RTU/ASCII Programmable Controller* или *Fastwel CPM713 MODBUS TCP Programmable Controller* соответственно при выполнении п. 3 приведенных выше указаний.

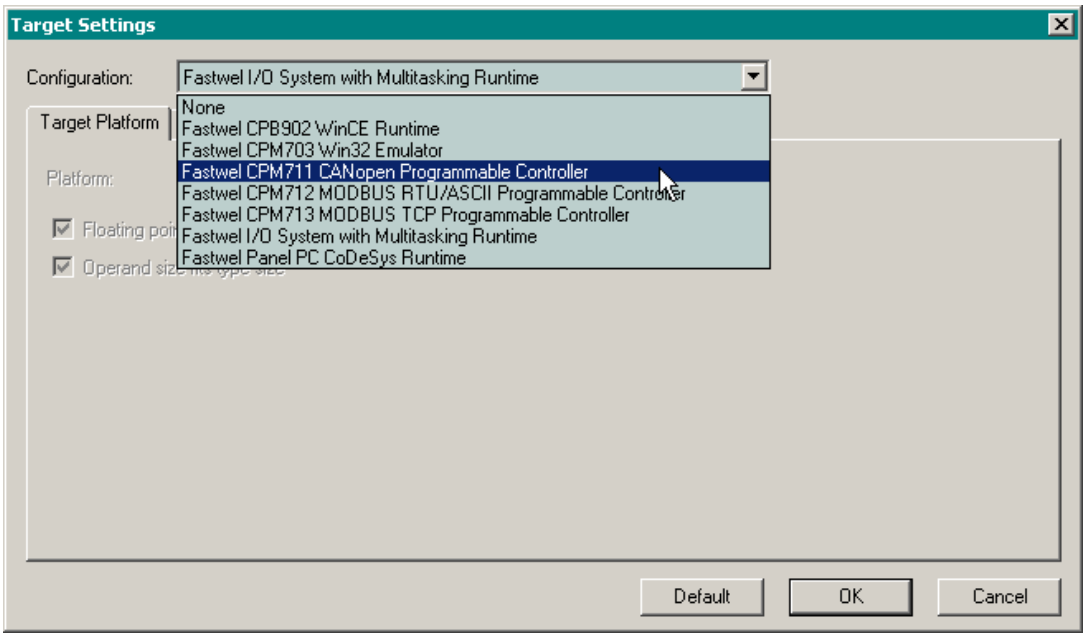


Рис. 8. Выбор платформы при переносе проекта

ВНИМАНИЕ!

При смене платформы в проекте, ранее созданном для платформы *Fastwel I/O System with Multitasking Runtime*, на соответствующую платформу для контроллера CPM711, CPM712 или CPM713 среда разработки CoDeSys 2.3 утрачивает информацию о ранее установленных обработчиках системных событий в окне ресурса **Task Configuration**.

Это не особенность пакета адаптации CoDeSys 2.3 для Fastwel I/O, а изначальное свойство среды разработки CoDeSys 2.3.

После переноса проекта, ранее разработанного для контроллера CPM701, CPM702 или CPM703, на соответствующий контроллер CPM711, CPM712 или CPM713 требуется повторно установить обработчики системных событий в окне ресурса **Task Configuration**. Более подробная информация об установке обработчиков системных событий приведена в п. 5.6 настоящего руководства.

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**.

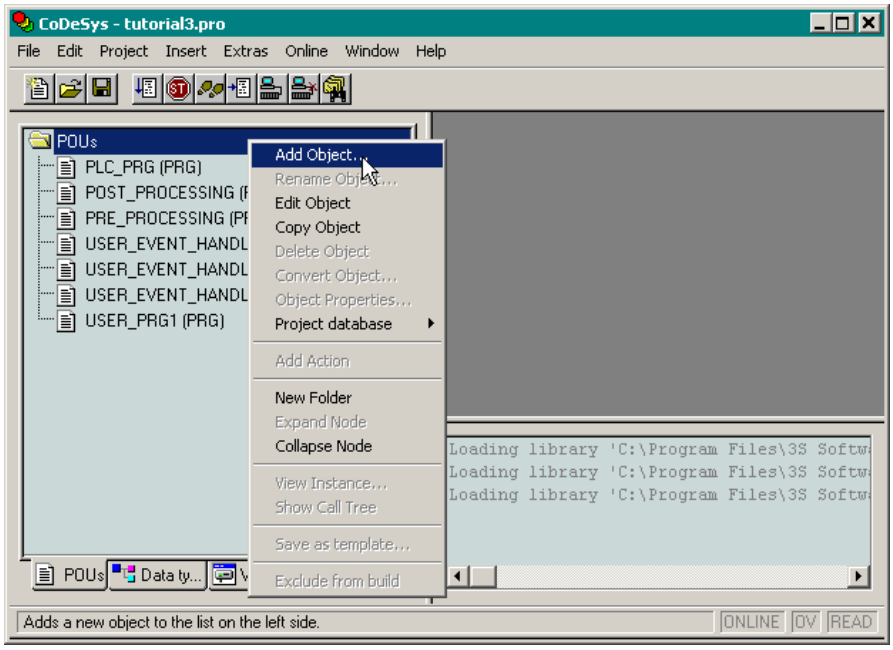


Рис. 9. Добавление программной единицы в древовидный список POU

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.

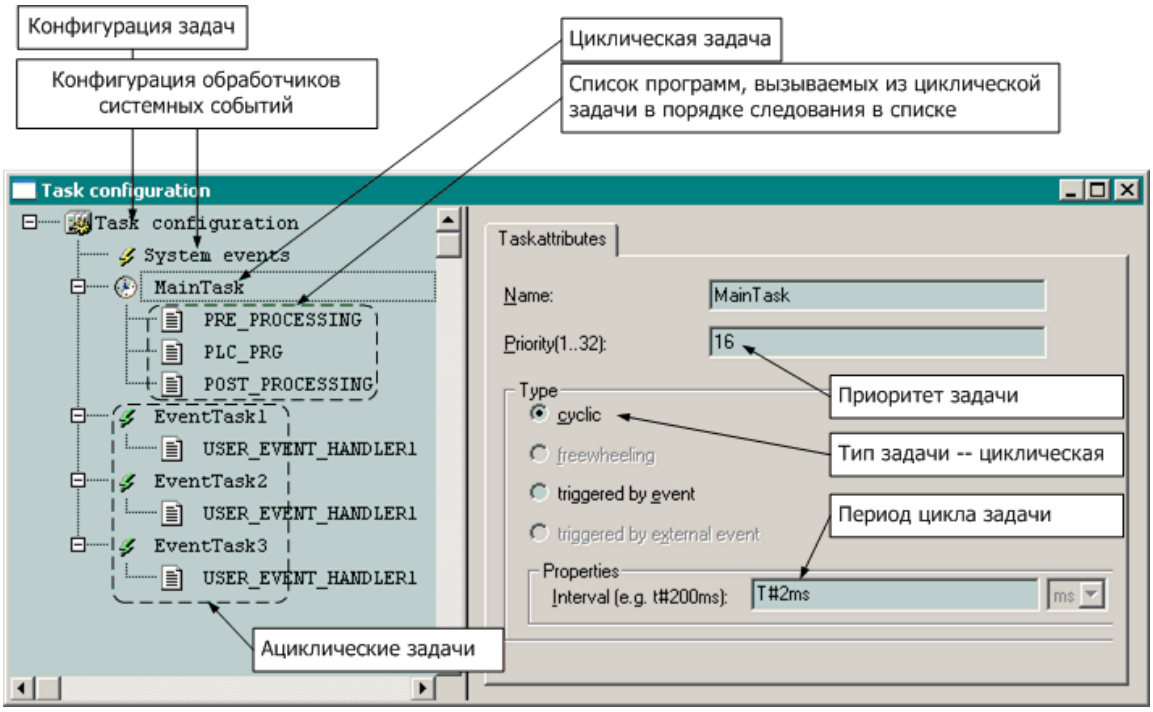


Рис. 10. Пример конфигурации задач приложения пользователя

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 сервисного 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 *CPM71x ... Programmable 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).

Примечание. Если в окно ресурса **Tasks Configuration** не добавлено ни одной задачи любого из поддерживаемых типов, среда разработки CoDeSys автоматически создаст описание циклической задачи с именем *DefaultTask* и ассоциирует с ней программу PLC_PRG (если она имеется в проекте). Период цикла данной задачи будет равен значению параметра *CPM71x ... Programmable Controller:SampleRate* (по умолчанию – 10 мс) в окне ресурса **PLC Configuration**.

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.

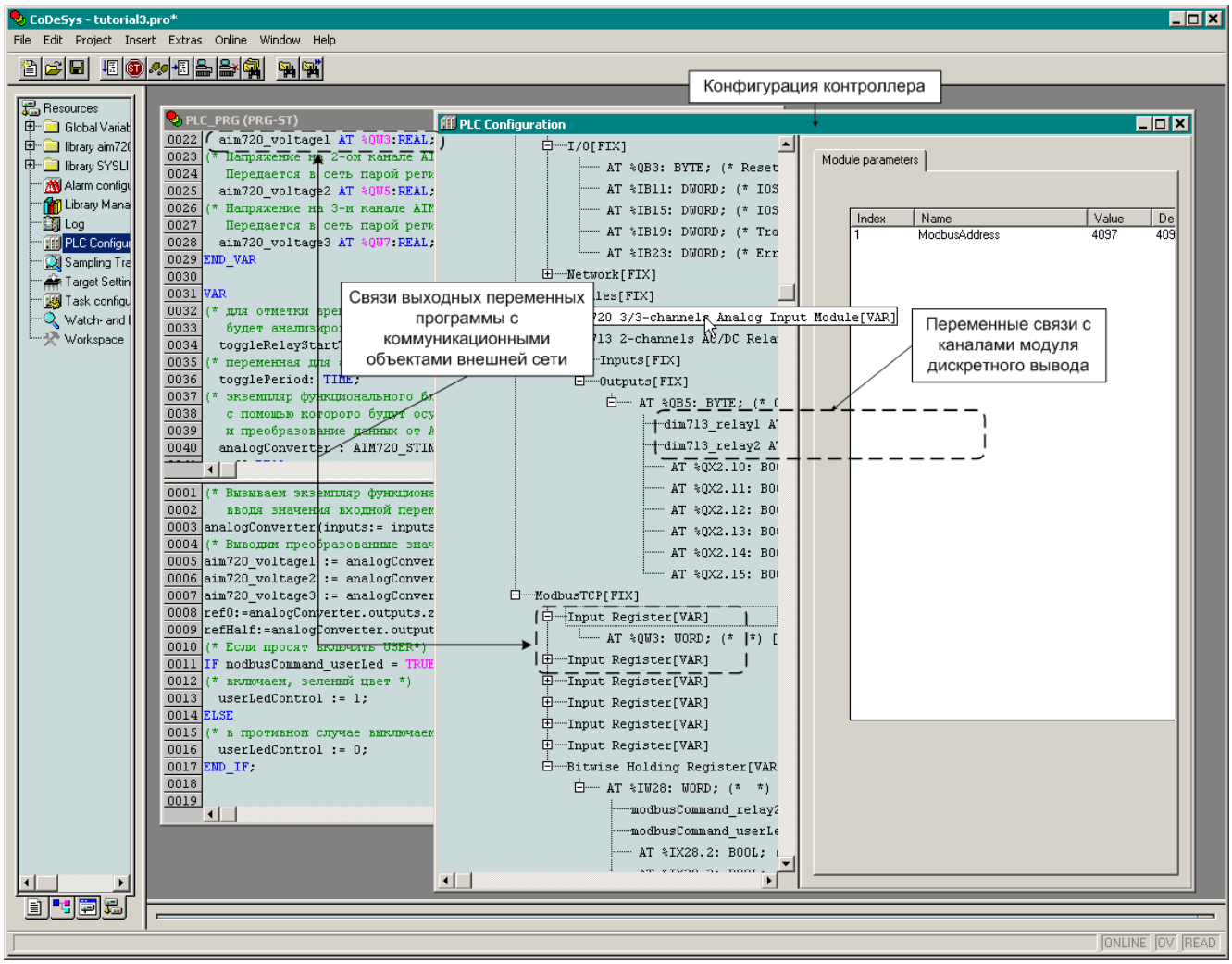


Рис. 11. Конфигурация контроллера

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 CPM711 – CANopen; CPM712 – MODBUS RTU/ASCII (подчиненный или мастер); CPM713 – MODBUS TCP (мастер и подчиненный)). Предназначен для обмена данными и командами с контроллером по сети. Описание принципов работы и способов конфигурировании перечисленных сервисов внешней сети приведено в соответствующих документах, перечисленных в 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 по данной сети, 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 не фатальных 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 при неправильном формате конфигурации.
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

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

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	резерв (исключения для операций с плавающей точкой не генерируются)
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** (**Нет**).
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 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 СРМ711, СРМ712 и СРМ713 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.

Информация о возможных причинах перехода в безопасный режим приведена в разделе 8 настоящего документа.

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**–*CPM71x ... Controller–I/O Modules*);

зеленый цвет (прерывисто) – обмен с модулями ввода-вывода, подключенными к контроллеру, не может быть выполнен за время, заданное в конфигурации контроллера (**Resources–PLC Configuration**–*CPM71x ... 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 проверки 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 переменные приложения 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 (установки начальных значений) 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 *CPM71x 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 *CPM71x Controller–I/O Modules* конфигурации контроллера. Для групп устанавливается период, определяемый параметром *CPM71x 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.

В процессе запуска при включении питания система исполнения проверяет наличие ранее сохраненных в энергонезависимой памяти контроллера RETAIN-переменных приложения. Для этого проверяется статус разряда батареи, после чего из области энергонезависимой памяти, расположенной непосредственно после сегмента RETAIN-переменных, считывается маркер конца сегмента RETAIN-переменных и специальная сигнатура, уникальная для приложения и формируемая при его загрузке в контроллер из среды разработки CoDeSys.

Если батарея разряжена или маркер конца сегмента RETAIN-переменных не обнаружен, либо считанная сигнатура отличается от только вычисленной для запускаемого приложения, RETAIN-переменным присваиваются начальные значения, заданные пользователем в областях деклараций RETAIN-переменных.

Если маркер конца сегмента RETAIN-переменных обнаружен и считанная сигнатура совпадает с только что вычисленной для запускаемого приложения, сегмент RETAIN-переменных оставляется неизменным, и приложение пользователя получит доступ к значениям RETAIN-переменных, сохраненным в энергонезависимой памяти непосредственно перед отключением питания или сбросом контроллера (кроме «холодного» по **Online–Reset (cold)**).

4.2.3. Процесс загрузки или обновления приложения

4.2.3.1. Общие сведения

Адаптированная среда исполнения приложений CoDeSys 2.3 для контроллеров Fastwel I/O до версии 2.67.23949 не позволяет выполнять загрузку в контроллер нового приложения посредством команды **Online–Create boot project (Онлайн–Создание загрузочного проекта)**, возвращая среде разработки CoDeSys код ошибки 20019 (запись системного файла запрещена).

Начиная с версии 2.67.23949, адаптированная среда исполнения поддерживает механизм горячего обновления (ONLINE CHANGE) приложения, при котором в контроллер загружаются только программные единицы, изменившиеся с момента последней загрузки приложения, а также дополнительная служебная информация, включая описания PERSISTENT-переменных, информацию о проекте и описания связей задач с образом процесса. В дальнейшем под загрузкой приложения в контроллер подразумевается процесс полной загрузки, а под обновлением – процесс загрузки только изменений (ONLINE CHANGE).

ВНИМАНИЕ!

Механизм горячего обновления (ONLINE CHANGE) поддерживается в пакете адаптации CoDeSys 2.3 для Fastwel I/O, начиная с версии 2.67.23949 и в среде исполнения приложений CoDeSys для контроллеров CPM711/CPM712/CPM713, начиная с версии 2.67.23949.

Для включения механизма горячего обновления следует открыть окно ресурса **Target Settings** (**Настройка целевой платформы**), щелкнуть на вкладке **General** (**Общие**) и отметить флажок **Online Change** (**Горячее обновление**), после чего закрыть окно **Target Settings** (**Настройка целевой платформы**) нажатием кнопки **OK**.

После одного или нескольких обновлений приложения в контроллере для сохранения изменений следует установить соединение с контроллером и выполнить команду **Online–Create boot project (Онлайн–Создание загрузочного проекта)**.

Процесс полной загрузки приложения начинается в момент, когда пользователь, предварительно настроив коммуникационный канал взаимодействия с контроллером через интерфейс P2P или через интерфейс внешней сети, выполняет команду **Online–Login** (**Онлайн–Соединение**) в среде разработки CoDeSys и нажимает кнопку **Yes** в появившейся диалоговой панели *The program has changed! Download the new program?(Программа была изменена! Загрузить новую программу?)*

Указанная диалоговая панель появляется в случае, если проект, открытый в CoDeSys, отличается конфигурацией и/или именем от того, что был ранее загружен в контроллер, а также, при отсутствии указанных отличий, в случае, если в каталоге размещения текущего открытого файла проекта (с расширением *.pro) отсутствует файл, генерируемый CoDeSys и содержащий информацию о предыдущей загрузке проекта в контроллер с именем *<имя файла проекта><идентификатор платформы>r.ri*. Если проект был ранее загружен в контроллер, и в каталоге размещения текущего открытого файла проекта присутствует данный файл, то после внесения изменений в исходный текст проекта при выполнении команды **Online–Login** (**Онлайн–Соединение**) на экран монитора будет выведена диалоговая панель **Online Change** (**Горячее обновление**) с кнопками **Yes** (**Да**), **No** (**Нет**),

Cancel (**Отмена**) и **Download All** (**Загрузить все)**. При нажатии **Download All** (**Загрузить все)** будет произведена полная загрузка приложения, а при нажатии **Yes** (**Да**) будут загружены только изменения.

Обратите внимание, что горячее обновление становится невозможным, если внесено хотя бы одно изменение в конфигурацию задач в окне ресурса **Task Configuration** (**Конфигурация задач**) или в конфигурацию контроллера окне ресурса **PLC Configuration** (**Конфигурация ПЛК**).

При изложении последующего описания процессов загрузки и обновления приложения предполагается, что контроллер функционирует в нормальном режиме согласно п. 4.2.1.2.

4.2.3.2. Полная загрузка

1. После того, как пользователь, выполнив команду **Online**–**Login** (**Онлайн**–**Соединение**) в среде разработки CoDeSys нажимает кнопку **Yes** (**Да**) в диалоговой панели **CoDeSys** с сообщением *The program has changed! Download the new program? (Программа была изменена! Загрузить новую программу?)* или **Download All (Загрузить все**) в диалоговой панели **Online Change** (**Горячее обновление**), среда разработки начинает загрузку секции исполняемого кода проекта. 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 ранее установлена пользователем в текущем приложении контроллера. Во время загрузки секций нового приложения работа текущего приложения контроллера, обмен данными с модулями ввода-вывода и обмен по сети **не прекращаются**.

   Начиная с версии 2.67.23947 системного программного обеспечения контроллеров CPM711/CPM712/CPM713, в данный момент запоминаются значения *сохраняемых* (PERSISTENT) переменных. Переменная становится сохраняемой (реманентной), если она объявлена в области декларации с ключевым словом PERSISTENT. Таким образом, при загрузках и обновлениях приложения в контроллере обеспечивается возможность сохранения неизменными значений некоторых переменных.

   Обратите внимание, что переменные, объявленные как VAR RETAIN PERSISTENT, становятся *сохраняемыми* и *энергонезависимыми*.

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** (**Конфигурация ПЛК**). В диалоговой панели статуса загрузки 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 **Task Configuration** (**Конфигурация задач**). 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** (**Информация о проекте**), вызываемой по команде меню **Project–Project Info** (**Проект–Информация о проекте**). 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 переменными, ссылающимися на непосредственные адреса во входной и выходной областях образа процесса. 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 вторичном 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, приостанавливается 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.

    Начиная с версии 2.67.23947 системного программного обеспечения контроллеров CPM711/CPM712/CPM713, в данный момент восстанавливаются значения *сохраняемых* (PERSISTENT) переменных, запомненные перед началом загрузки.

    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** (**Информация о проекте**) 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* 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.

ВНИМАНИЕ!

Начиная с версии 2.67.23949 системного программного обеспечения контроллеров CPM711/CPM712/CPM713, если в настройках целевой платформы включена поддержка механизма горячего обновления (отмечен флажок **Общие–Горячее обновление**), параметр *Fastwel I/O System Configuration:HotUpdateDisabled* игнорируется системой исполнения контроллера, если приложение в контроллере хотя бы раз было обновлено методом ONLINE CHANGE.

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.

ВНИМАНИЕ!

Не рекомендуется использовать механизм, основанный на параметре *Fastwel I/O System Configuration:HotUpdateDisabled=No* в контроллерах CPM711/CPM712/CPM713, начиная с версии системного программного обеспечения 2.67.23947.

Для сохранения значений некоторых энергонезависимых переменных после загрузки и обновления приложения при их объявлении используйте ключевое слово VAR RETAIN PERSISTENT.

Начиная с версии 2.61 системного программного обеспечения контроллеров CPM711/CPM712/CPM713, при загрузке нового или обновлении существующего приложения в контроллере применяется следующий алгоритм управления RETAIN-переменными:

1. Если параметр *Fastwel I/O System Configuration:HotUpdateDisabled* (*горячее обновление отключено*) установлен пользователем в *Yes* (*Да*), то при любом изменении приложения (код, данные, конфигурация, информация о проекте в **Project–Project Info**) в среде разработки CoDeSys всем RETAIN-переменным будут присвоены начальные значения, заданные пользователем в областях деклараций RETAIN-переменных.
2. Если параметр *Fastwel I/O System Configuration:HotUpdateDisabled* (*горячее обновление отключено*) установлен пользователем в *No* (*Нет*), то система исполнения контроллера будет стремиться восстанавливать ранее сохраненные значения RETAIN-переменных, если только пользователем не выполнено одно из следующих действий: **Online–Reset (Cold)**; или изменены состав, структура, или типы RETAIN-переменных приложения; или изменено любое из полей **Project–Project Info** с последующими **Project–Clean All** и **Project–Rebuild All**.

   ВНИМАНИЕ! В случае изменений структуры и/или состава/типов RETAIN-переменных, при которых сохраняется неизменным размер сегмента RETAIN-переменных, имеется вероятность неправильного восстановления значений «старых» RETAIN-переменных и инициализации «новых». В связи с этим не используйте данную функциональную возможность при модификации приложения без предварительного останова контролируемого технологического процесса.

Обратите внимание, что перед загрузкой другого приложения в контроллер среда разработки CoDeSys всегда изменит секцию **Project Info**, поэтому невозможна ситуация восстановления значений RETAIN-переменных от «старого» приложения в «новом» приложении.

4.2.3.3. Горячее обновление (ONLINE CHANGE)

1. После нажатия **Yes** (**Да**) в диалоговой панели **Online Change** (**Горячее обновление**) среда разработки начинает передачу в контроллер секции кода обновляемого приложения, которая содержит описания сохраняемых (PERSISTENT) переменных, исполняемый код только изменившихся с момента последней успешной полной загрузки программных единиц приложения, таблицу привязки адресов переменных в коде (relocation table) и информацию о функциях внешних системных библиотек, требующихся для работы приложения. В диалоговой панели статуса загрузки изменяется значение счетчика загруженных байт.
   По окончании передачи система исполнения сохраняет полученную секцию кода во вторичном сегменте кода, предназначенном для горячего обновления.
2. После передачи секции кода обновляемого приложения среда разработки передает в контроллер секцию информации о проекте, определяемую в диалоговой панели **Проект–Информация о проекте**. В диалоговой панели статуса загрузки значение счетчика загруженных байт не изменяется.
3. Среда разработки, убедившись, что секция информации о проекте передана в контроллер успешно, приступает к передаче секции описателей связей задач обновляемого приложения с образом процесса. Содержимое секции определяется

созданными или измененными пользователем переменными, ссылающимися на непосредственные адреса во входной и выходной областях образа процесса. В диалоговой панели статуса загрузки изменение значения счетчика загруженных байт по-прежнему не происходит.

4. По окончании передачи секции описателей связей задач происходит инициализация загруженного кода измененных программных единиц приложения во вторичном сегменте кода, вызывается обработчик системного события *OnProgramChange*, после чего неизменившиеся программные единицы копируются во вторичный сегмент кода сразу после изменившихся.
   Если только что переданная секция описателей связей задач приложения с образом процесса отличается от ранее загруженной в контроллер, функционирование приложения приостанавливается, и происходит повторное связывание задач с образом процесса.
   При успешном завершении всех перечисленных операций производится перестановка первичного и вторичного сегментов кода, и функционирование приложения возобновляется.

Обратите внимание, что в случае изменения хотя бы одного начального значения хотя бы одной переменной среда разработки сгенерирует заново код скрытой программной единицы инициализации данных приложения (GlobalInit), который, при большом количестве переменных в приложении, будет передан в контроллер в течение значительного времени. При этом все ранее существующие переменные обновленного приложения сохраняет свои значения. Крайне осторожно пользуйтесь механизмом горячего обновления при обновлении приложений, алгоритм которых имеет зависимости от предыдущего состояния.

При большом количестве переменных в приложении и, особенно, при большом количестве сохраняемых (PERSISTENT) переменных передаваемая секция кода может всегда быть довольно большой, и передача секции кода в контроллер может занимать существенное время. Размер описателя каждой сохраняемой (PERSISTENT) переменной в передаваемой в контроллер секции кода составляет не менее 13 байт плюс количество символов (байт) в полном имени переменной. Например, DEST_PRG.byPRetAr[1], описывает первый элемент массива byPRetAr, объявленный в программной единице DEST_PRG. Таким образом, для ускорения процесса горячего обновления минимизируйте количество сохраняемых переменных и не используйте для них слишком длинные имена.

Если в загруженном в контроллер приложении используется адресная арифметика (указатели), то при изменении адресов и/или размеров переменных сразу после горячего обновления приложение может начать работать неправильно или, в наихудшем случае, контроллер может перейти в безопасный режим. Кроме того, механизм межзадачного взаимодействия, описанный в п. 0, и системная библиотека FastwelModbusServer.lib, описанная в п. 6.5, не предусматривают возможности продолжения правильной работы приложения после горячего обновления в случае, если в обновленном приложении изменились адреса или размеры переменных, передаваемых функциям *F_IecTasks_linkVariables* и *FwModbusServerInit*. При использовании в приложении библиотеки FastwelModbusServer.lib и/или межзадачного взаимодействия средствами библиотеки FastwelTasksExchange.lib в случае изменения размеров или адресов переменных, передаваемых функциям *F_IecTasks_linkVariables* и *FwModbusServerInit*, всегда используйте полную загрузку приложения в контроллер.

ВНИМАНИЕ!

После одного или нескольких обновлений приложения в контроллере для сохранения изменений следует выполнить команду **Online–Create boot project (Онлайн–Создание загрузочного проекта)** при наличии установленного соединения с контроллером. В противном случае при последующем перезапуске в контроллере будет запущено приложение, загруженное до обновлений.

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.

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.

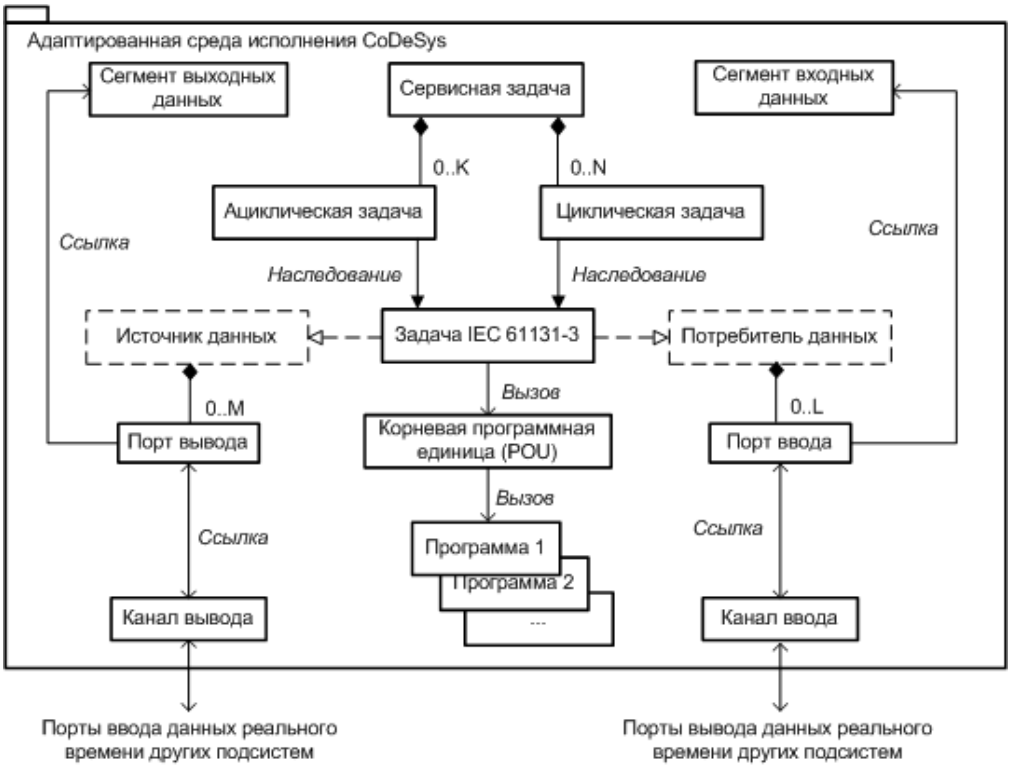


Рис. 12. Архитектура адаптированной среды исполнения CoDeSys

На сервисную задачу возлагается выполнение следующих основных функций:

1. Вызов большинства пользовательских функций обработки системных событий.
2. 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**.
3. 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 10 с.
4. 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.

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.

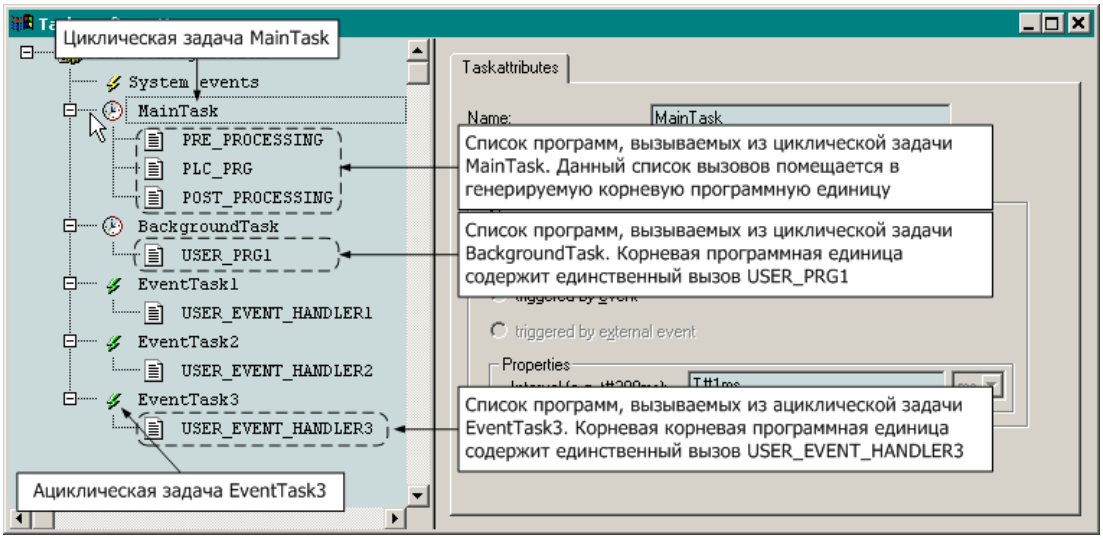


Рис. 13. Списки программных единиц, вызываемых из задач

Принцип формирования описателей ссылок иллюстрируется рис. 14. 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.

Для каждой циклической задачи приложения создается собственный сегмент входных данных, который обновляется с периодом данной задачи, что позволяет избежать проблемы перекрытия общих входных данных, используемых в алгоритмах разных циклических и ациклических задач. При запуске приложения производится релокация ссылок в программах, исполняемых в каждой циклической задаче, на собственный сегмент входных данных данной задачи.

ВНИМАНИЕ!

В POU типа FUNCTION, вызываемых из разных циклических задач, запрещено обращение к переменным, отображенным на сегмент входных данных (AT %I*).

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.

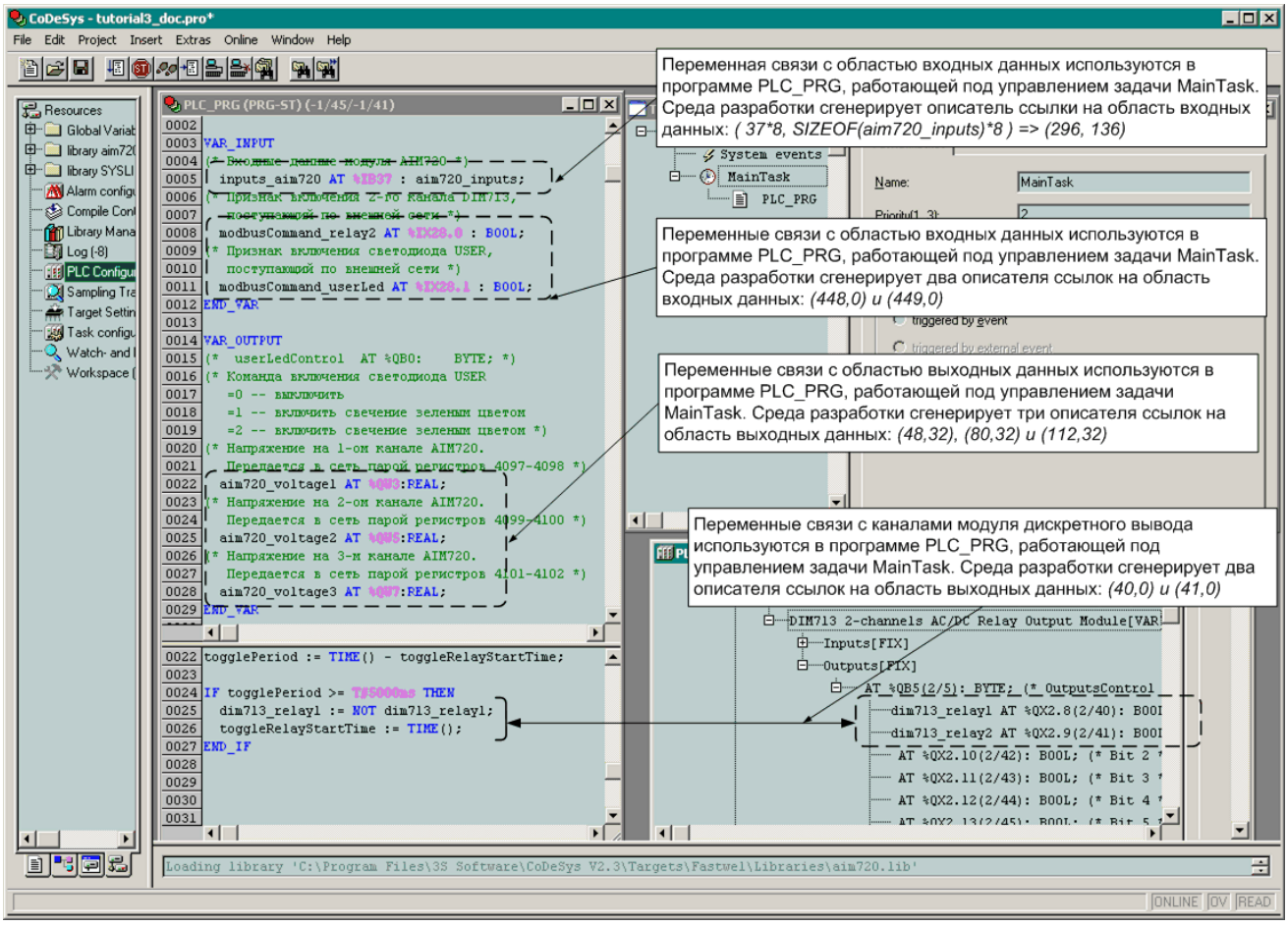


Рис. 14. Принцип формирования описателей ссылок на области входных и выходных данных приложения

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 пользователь не добавил ни одной задачи в ресурс **Task Configuration** то программа PLC_PRG будет исполняться под управлением циклической задачи *DefaultTask* с периодом, заданным в ресурсе **PLC Configuration** для параметра *CPM71x ... Programmable Controller:Sample Rate*.

Каждая циклическая задача исполняется на контексте отдельного потока операционной системы, и в процессе выполнения производятся следующие действия:

1. Выполняется ввод данных в собственный сегмент входных данных задачи.
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 30-40 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 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 на 15-20 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 *CPM71x–Diagnostics–Application–CyclesCounter* и *CPM71x–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 *CPM71x–Diagnostics–Cyclic Tasks Status–Task1_16*;

с периодом 10 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 *CPM71x...Controller:SampleRate*.
11. По прошествии времени *CPM71x...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)

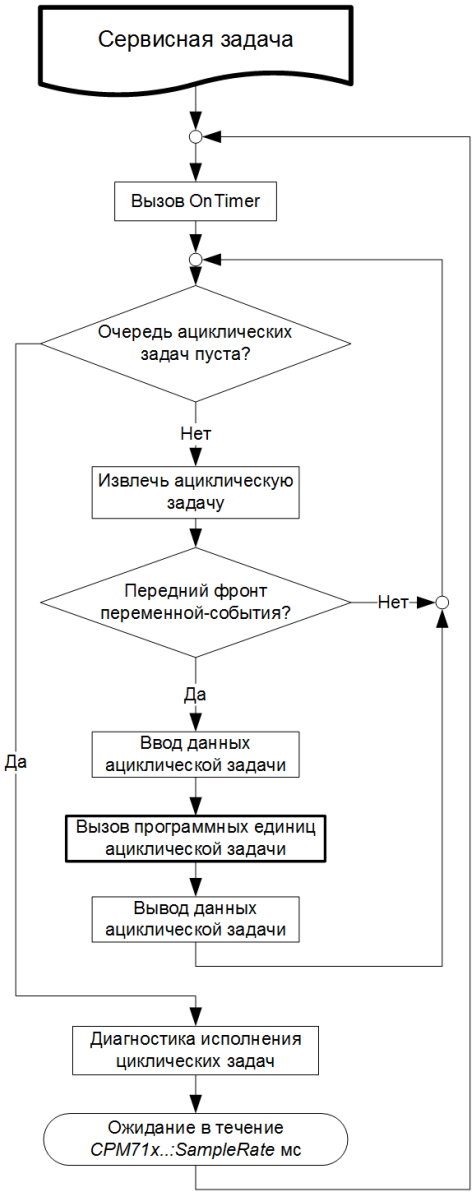


Рис. 15. Алгоритм функционирования сервисной задачи

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 или обновленной программы, описанного в п. 4.2.3.2 и п. 4.2.3.3 ▪ 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 *CPM71x...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 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:
         CloseFilesAndPorts();
```

```
    F_EVENT_ON_INIT:
          LinkTasks();
          OpenFilesAndPorts();
    F_EVENT_POWER_ON:
          PerformSomeOnPowerOnActions();
    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. 0 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 (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.

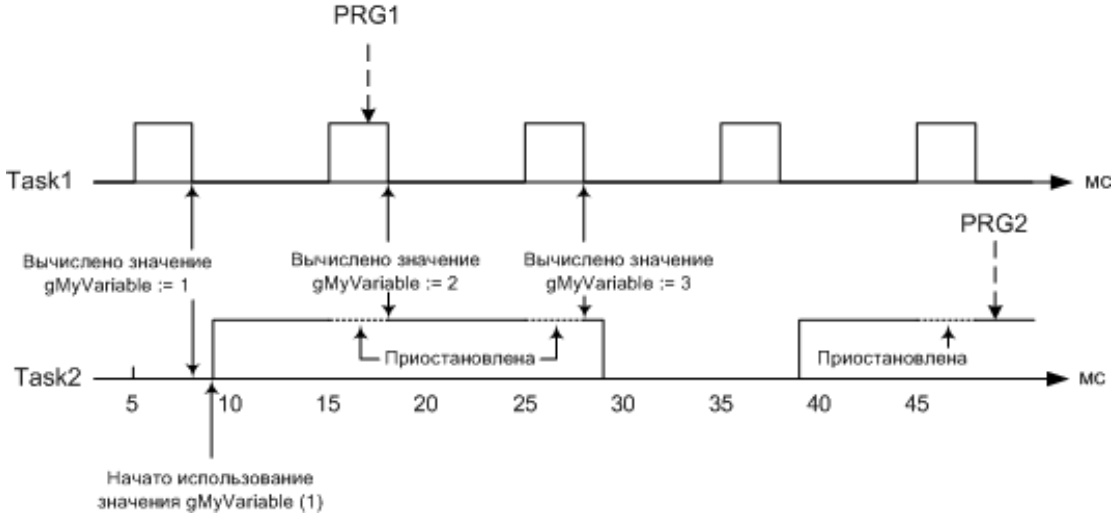


Рис. 16. Доступ к общей переменной из двух разных задач

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 CPM71x составляет 32 бита. Переменные типов LREAL, массивы и структуры, длина которых превышает 4 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, только что вычисленного "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.

Механизм межзадачного взаимодействия, реализуемый системной библиотекой FastwelTasksExchange.lib, не предусматривает возможности продолжения правильной работы приложения после горячего обновления (ONLINE CHANGE) в случае, если в обновленном приложении изменились адреса или размеры переменных, передаваемых функции *F_IecTasks_linkVariables*. В случае изменения размеров или адресов переменных, передаваемых функции *F_IecTasks_linkVariables*, всегда используйте полную загрузку приложения в контроллер.

4.2.5. Диагностика

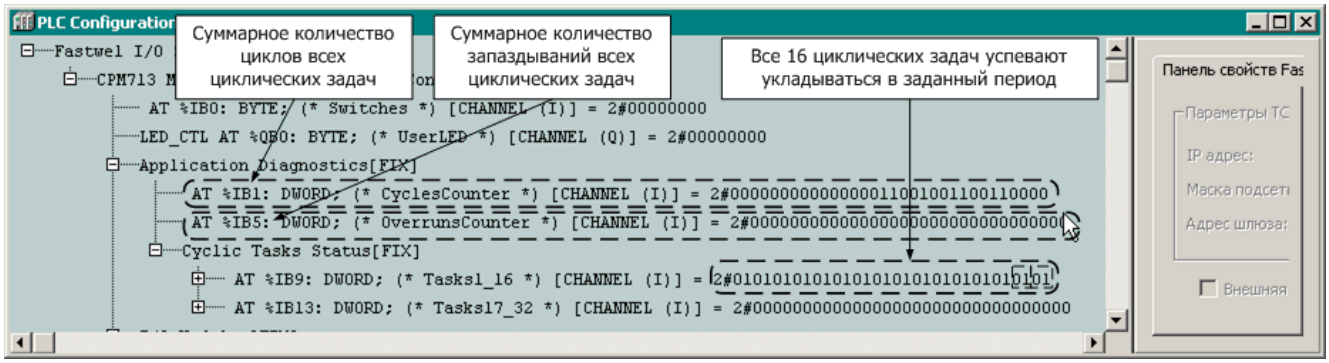


Рис. 17. Диагностические каналы среды исполнения CoDeSys

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 Application Diagnostics конфигурации контроллера узла			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
CyclesCounter	%IB1	DWORD	Общее количество циклов всех циклических задач
OverrunsCounter	%IB5	DWORD	Общее количество циклов циклических задач, во время которых они не успели завершить выполнение в течение своих заданных периодов
Cyclic Tasks Status–Task1_16	%IB9	DWORD	2-битовые статусы циклических задач

Секция *Cyclic Tasks Status* содержит канал *Task1_16* типа DWORD, пары 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.

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.

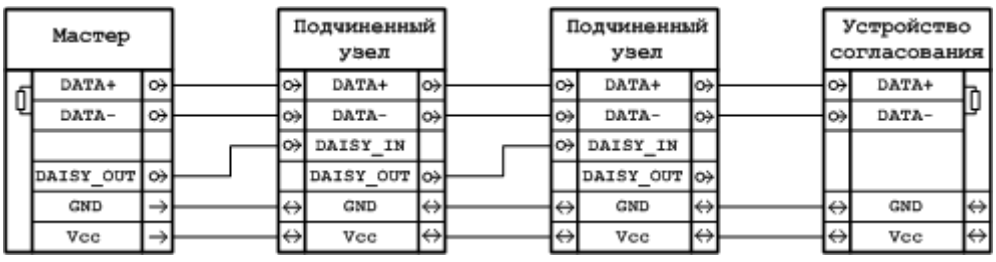


Рис. 18. Линии среды обмена данными FBUS

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 *CPM71x...Controller–I/O Modules* ресурса **PLC Configuration** (**Конфигурация ПЛК**) 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.

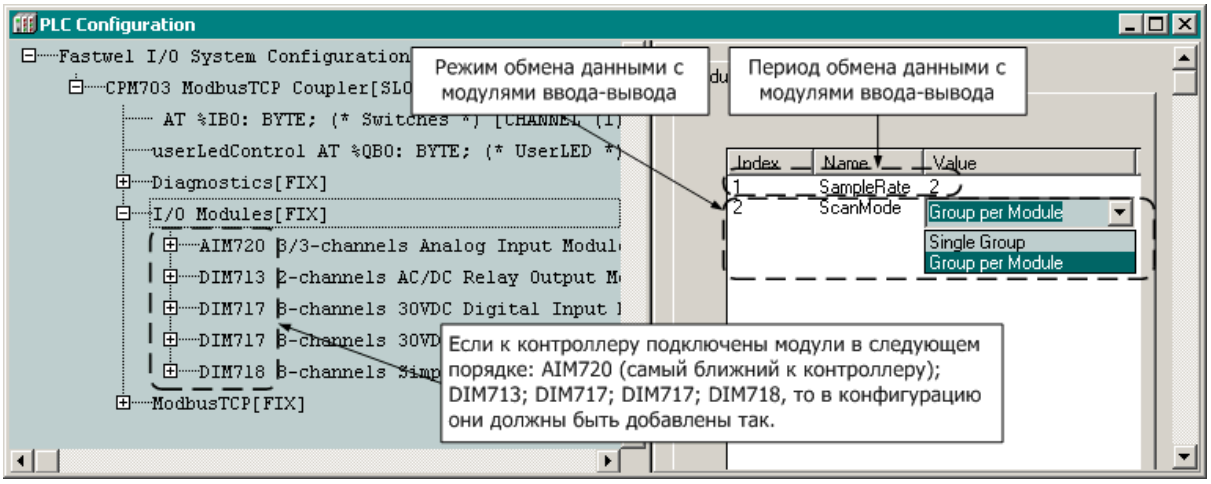


Рис. 19. Конфигурация сервиса ввода-вывода контроллера

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* (**Свойства FBUS–Режим обмена**) 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 определяется параметром *CPM71x...Controller–I/O Modules:SampleRate* (**Свойства FBUS–Период опроса, мс**). При использовании режима индивидуального обмена пропускная способность внутренней шины существенно ухудшается за счет появления пауз длительностью до 150 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 проверка сетевой конфигурации подключенных к шине модулей ввода-вывода.

В ходе проверки сервис ввода-вывода определяет сетевые идентификаторы (адреса) всех подчиненных узлов (модулей ввода-вывода), подключенных к шине FBUS, для чего выполняет короткие индивидуальные запросы по адресам от 0 до 63. Адреса, для которых получены корректные ответы, составляют множество адресов обнаруженных подчиненных узлов шины.

Если при выполнении данной операции фиксируются ответы с ошибками протокола обмена, то сетевая конфигурация подчиненных узлов на шине признается недействительной и запускается процедура назначения адресов подчиненным узлам. Адреса назначаются последовательно по возрастанию от 0 до значения, равного количеству обнаруженных подчиненных узлов, минус 1.

Этап 2:

Выполняется проверка соответствия состава и конфигурации обнаруженных подчиненных узлов составу и конфигурации модулей, определенных в списке *I/O Modules* в окне ресурса **PLC Configuration** (**Конфигурация ПЛК**) текущего приложения CoDeSys 2.3, загруженного в контроллер.

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

1. Выясняется, совпадает ли тип обнаруженного модуля, имеющего некоторый адрес, с типом модуля, имеющимся в конфигурации контроллера. Сетевой адрес модуля в конфигурации равен его порядковому номеру в списке *I/O Modules* в окне ресурса **PLC Configuration** (**Конфигурация ПЛК**), начиная с 0.
2. Если соответствие типа установлено, считывается текущий идентификатор конфигурации модуля, сохраненный в его энергонезависимой памяти, и сравнивается с идентификатором текущей конфигурации контроллера. Если идентификатор конфигурации, считанный у модуля, не совпадает с идентификатором конфигурации контроллера, новая конфигурация передается модулю по внутренней шине.

В результате инициализации шины сервис ввода-вывода определяет фактическое состояние шины как *полностью исправное, частично исправное* или *неисправное*.

При *полностью исправном* состоянии шины все модули ввода-вывода, которые имеются в конфигурации контроллера, обнаружены и правильно сконфигурированы, и обмен данными со всеми модулями выполняется без ошибок.

Допускается подключать к шине ввода-вывода дополнительные модули, которых нет в конфигурации приложения, загруженного в контроллер, но только правее последнего модуля, имеющегося в конфигурации приложения.

При *частично исправном* состоянии шины состав и конфигурация обнаруженных модулей ввода-вывода соответствуют заданным в конфигурации приложения, загруженного в контроллер, за исключением некоторых модулей, которых нет среди обнаруженных.

Обратите внимание, что при этом адреса и конфигурационная информация, считанная из всех обнаруженных модулей, полностью соответствуют определенному в загруженном приложении составу модулей ввода-вывода. Такая ситуация возможна в случае, когда все модули ввода-вывода, ранее определенные в приложении, были обнаружены и успешно сконфигурированы, но часть из них вышла

из строя или была удалена в процессе эксплуатации, и при этом приложение контроллера не изменялось.

Неисправное состояние шины устанавливается тогда, когда не может быть установлено хотя бы частично исправное состояние.

При полностью или частично исправном состоянии шины сервис ввода-вывода будет выполнять обмен данными реального времени с обнаруженными модулями в режиме, который задан параметром *I/O Modules:ScanMode* (**Свойства FBUS–Режим обмена**). При неисправном состоянии шины обмен данными реального времени с модулями ввода-вывода выполняться не будет.

При частично исправном или неисправном состоянии шины сервис ввода-вывода будет периодически (один раз в секунду) сканировать шину, выполняя проверку сетевой конфигурации. Обмен данными реального времени при частично исправном состоянии шины при этом не прекращается. В случае обнаружения изменений сетевой конфигурации повторно запускаются процедуры второго этапа инициализации, и состояние шины переопределяется по результатам их выполнения.

4.3.3. Обмен данными с модулями ввода-вывода

4.3.3.1. Групповой режим (*Single Group*)

Если инициализация внутренней шины контроллера выполнена успешно и состояние шины признано частично или полностью исправным, сервис ввода-вывода приступает к обмену данными с обнаруженными 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.

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.

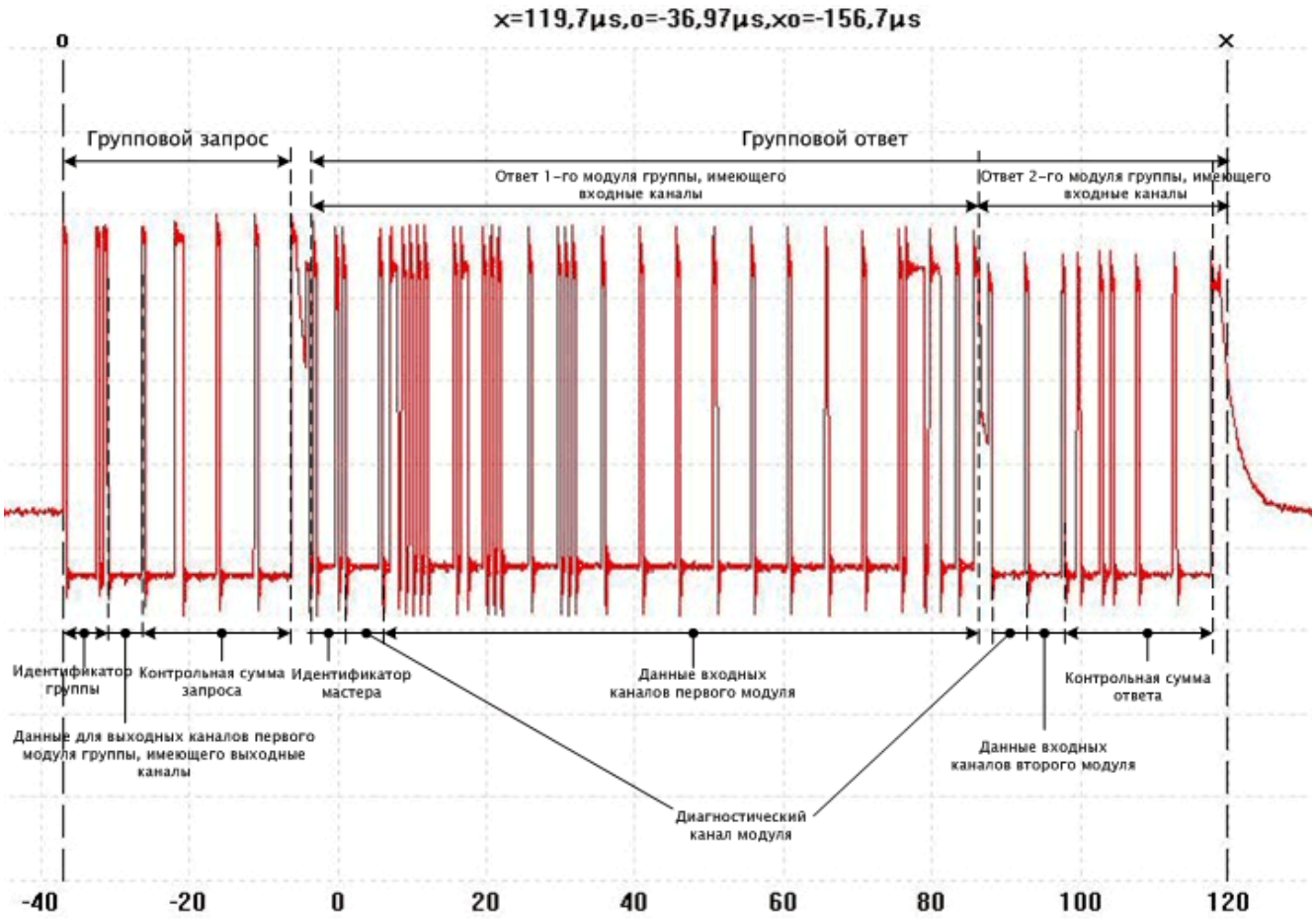


Рис. 20. Осциллограмма группового обмена с модулями AIM720 и DIM713

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

Если инициализация внутренней шины контроллера выполнена успешно и состояние шины признано частично или полностью исправным, то сервис ввода-вывода приступает к обмену данными с обнаруженными модулями. Если обнаружены не все модули, то, с периодом 1 с, выполняется процедура поиска и конфигурирования модулей согласно п. 4.3.2, пока все модули в конфигурации контроллера не будут обнаружены.

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 (*I/O Modules–FBUS 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.

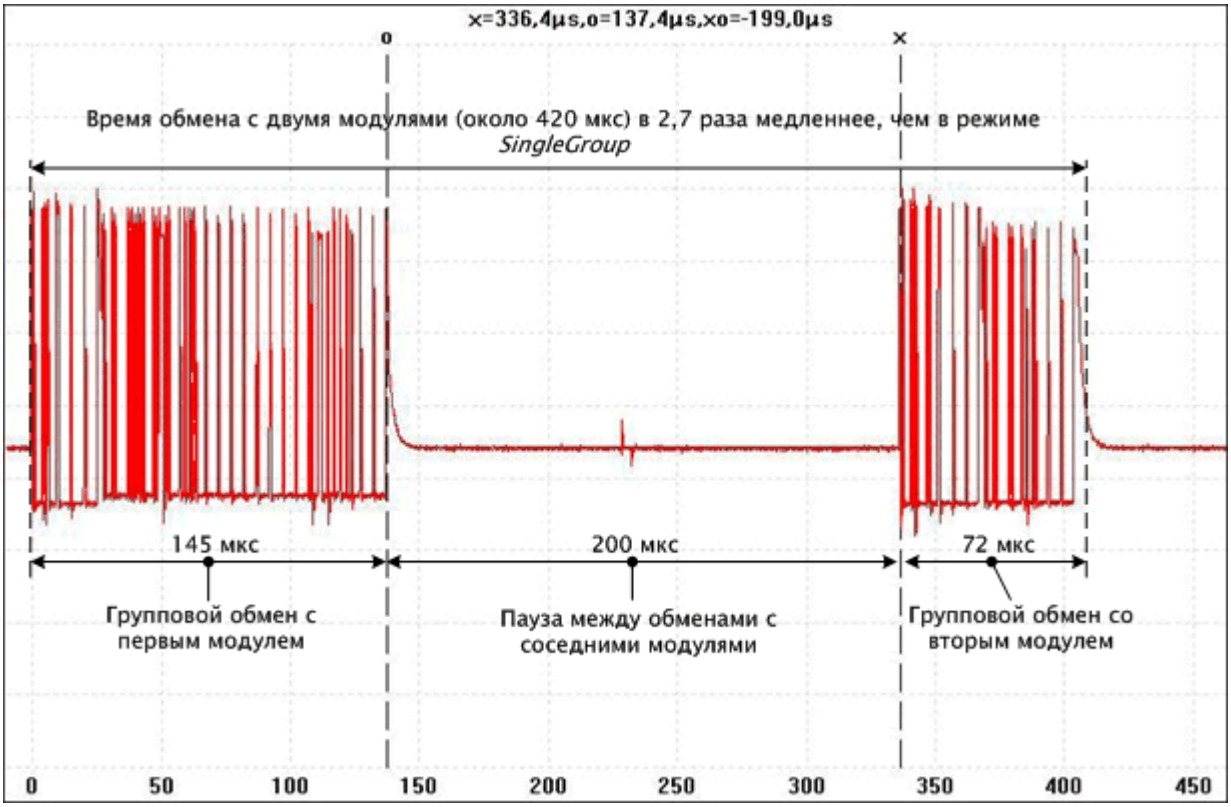


Рис. 21. Осциллограмма обмена с модулями AIM720 и DIM713 в режиме *Group per Module*

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%.

4.3.4. Обработка нештатных ситуаций

4.3.4.1. Ошибка инициализации при запуске контроллера

Если в процессе инициализации шины ее состояние определено как *полностью* или *частично исправное*, сервис ввода-вывода будет выполнять обмен данными реального времени с обнаруженными модулями в режиме, заданном в параметре *I/O Modules:ScanMode* (**Свойства FBUS–Режим обмена**). В *неисправном* состоянии обмен данными с модулями ввода-вывода выполняться не будет.

При *частично исправном* или *неисправном* состоянии шины сервис ввода-вывода будет периодически (один раз в секунду) сканировать шину выполняя процедуру проверки сетевой конфигурации. При *частично исправном* состоянии шины обмен данными реального времени с обнаруженными модулями не прекращается.

В случае обнаружения изменений сетевой конфигурации повторно запускаются процедуры второго этапа инициализации, состояние шины переопределяется по результатам их выполнения.

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

Если в процессе обмена данными с модулями ввода-вывода пять операций группового обмена подряд завершились неудачно, сервис прекращает обмен данными с модулями данной группы 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. С модулями, с которыми восстановлена связь, обмен данными реального времени немедленно возобновляется.

Процедура обнаружения и повторной инициализации модулей выполняется периодически (один раз в секунду), пока не восстановлена связь со всеми модулями ввода-вывода, описания которых имеются в конфигурации приложения, загруженного в контроллер.

ВНИМАНИЕ!

Если после разборки/сборки линейки заведомо исправных модулей состояние шины индицируется, как частично исправное, выполните следущие действия:

1. Переведите контроллер в заводской безопасный режим, для чего включите переключатель 1 и перезапустите контроллер.

2. Запустите *Сервисную утилиту FASTWEL I/O*, установите соединение с контроллером, перейдите на вкладку **Модули ввода-вывода** и нажмите **Модули–Посмотреть**.

3. Верните контроллер в нормальный режим, для чего выключите переключатель 1 и перезапустите контроллер командой **Сброс** в *Сервисной утилите FASTWEL I/O*.

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 *I/O Modules:SampleRate* (**Свойства FBUS–Период опроса, мс**).

Зеленый цвет (прерывисто) – все модули ввода-вывода, определенные в конфигурации проекта, обнаружены и успешно сконфигурированы, но обмен данными реального времени с ними не может быть выполнен за время, заданное в конфигурации контроллера *I/O Modules:SampleRate* (**Свойства FBUS–Период опроса, мс**), 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.

Зеленый и красный цвета (попеременно прерывисто) – начиная с версии 2.67 системного программного обеспечения контроллера, межмодульная шина находится в *частично исправном* состоянии. При частично исправном состоянии шины состав и конфигурация обнаруженных модулей ввода-вывода соответствуют заданным в конфигурации приложения, загруженного в контроллер, за исключением некоторых модулей, которых нет среди обнаруженных.

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.

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.

Таблица 8

Описание области I/O Modules–FBUS Diagnostics конфигурации контроллера узла			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
IOStatus0	%IB17	DWORD	Битовая маска наличия модулей ввода-вывода с 1-го по 32-й, соответствующих перечисленным в конфигурации контроллера
IOStatus1	%IB21	DWORD	Битовая маска наличия модулей ввода-вывода с 33-го по 64-й, соответствующих перечисленным в конфигурации контроллера
TransactionsCount	%IB25	DWORD	Общее количество обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера
ErrorsCount	%IB29	DWORD	Количество неудачных обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера

Логический 0 индицирует отсутствие связи с модулем. Например, если в конфигурации приложения, загруженного в контроллер, 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.

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

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

Обновление значений диагностических каналов сервиса ввода-вывода выполняется не чаще одного раза в секунду.

Обратите внимание, что при работе с модулями NIM741 и NIM742, представленными в конфигурации контроллера элементами *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module*, значения счетчиков *TransactionsCount* и *ErrorsCount* не изменяются.

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 CPM711, CPM712, CPM713 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, соответствующей используемому типу контроллера
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 CPM711 CANopen Programmable Controller для контроллера CPM711;

   Fastwel CPM712 MODBUS RTU/ASCII Programmable Controller для CPM712;

   Fastwel CPM713 MODBUS TCP Programmable Controller для CPM713

   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*, период которой будет равен значению параметра *CPM71x...Controller:SampleRate*, доступному в ресурсе PLC Configuration, как показано на рис. 22. Из корневой программной единицы данной задачи будет вызываться программа *PLC_PRG*.

   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: **;**

Параметр *Autostart* предназначен для отладки приложения. Если в контроллер загружено приложение, для которого параметр *CPM71x...Controller:Autostart* установлен в *No*, среда исполнения будет ожидать, когда пользователь выполнит команду **Online–Run** из среды разработки CoDeSys, не исполняя при этом код циклических и ациклических задач.

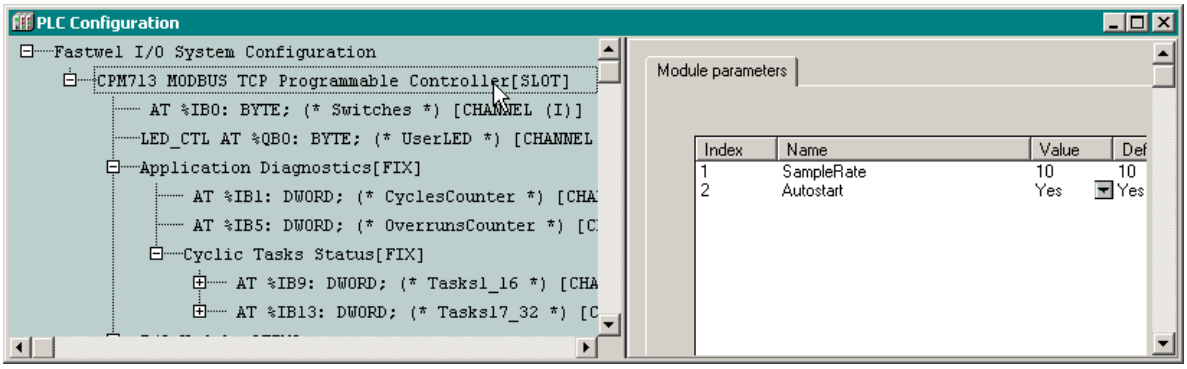


Рис. 22. Общие параметры системы исполнения контроллера

4. 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**.
5. 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.

В конфигурации всех контроллеров Fastwel I/O имеются одинаковые элементы и параметры. Основной элемент конфигурации всех контроллеров *CPM71x...Controller* (*x...* означает последнюю цифру обозначения контроллера и название интерфейса внешней сети, например *CPM711 CANopen Programmable Controller*) имеет один параметр *SampleRate*, который определяет:

1. При отсутствии в ресурсе **Tasks Configuration** задач, добавленных пользователем, – период цикла "автоматической" циклической задачи с именем *DefaultTask*, на контексте которой исполняется программа *PLC_PRG*
2. Период цикла сервисной задачи адаптированной среды исполнения CoDeSys Fastwel I/O в диапазоне от 2 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. Логическая 1 в некотором разряде данного канала свидетельствует о том, что переключатель с соответствующим номером включен.

Выходной канал *UserLED* с адресом %QB0 предназначен для управления свечением индикатора USER на передней панели контроллера 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

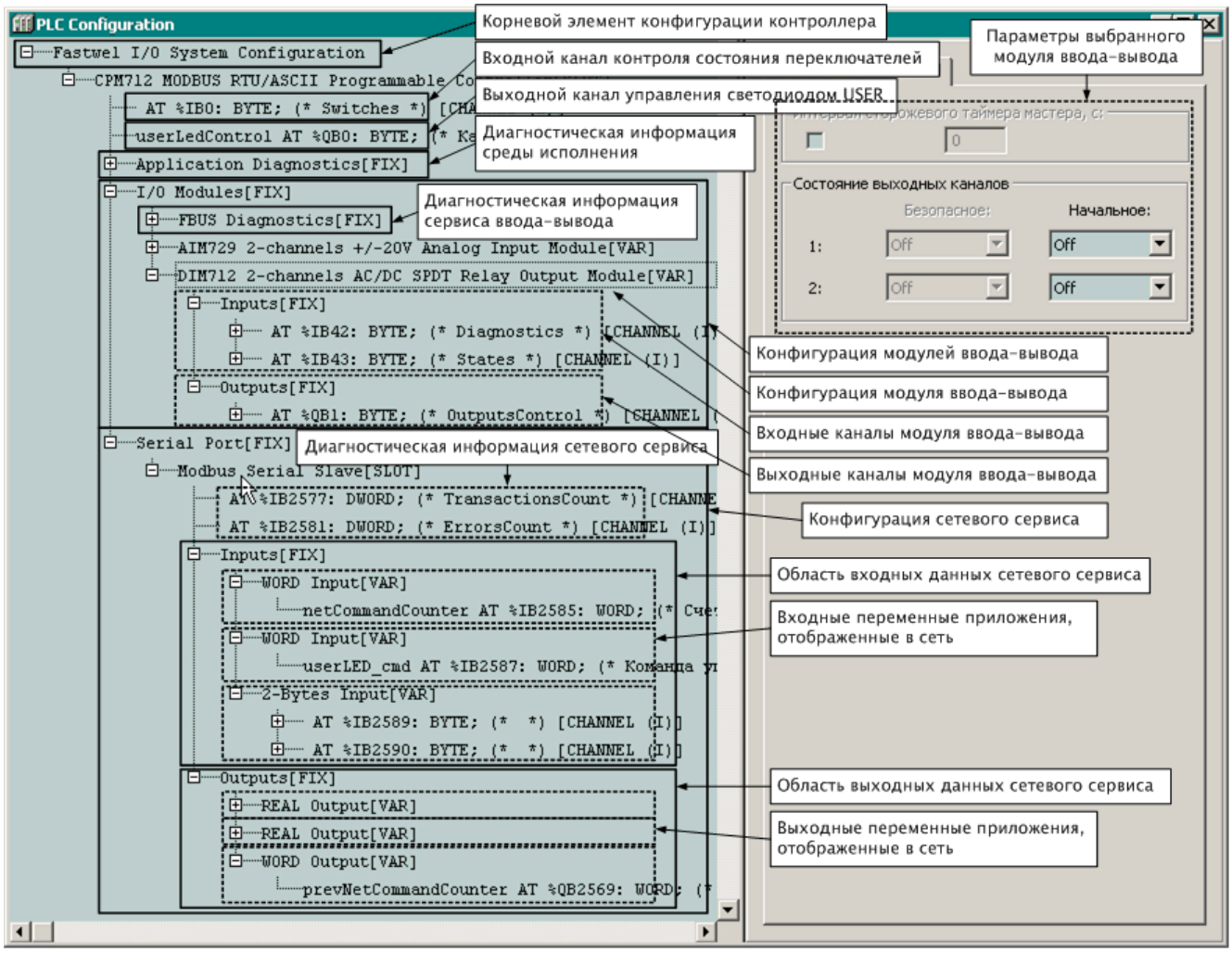


Рис. 23. Элементы конфигурации контроллера

Элемент *CPM71x... 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 *CPM71x... Controller–<Port Type> Port* предназначен для конфигурирования основного сетевого интерфейса контроллера. 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 периодически выполняйте команду **Calculate Addresses** в контекстном меню над элементом конфигурации *CPM71x... Controller*.

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 контроллеров серии CPM71x составляет 16384. 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 платформ CPM71x составляет 2 Мбайта, из которых в среднем до 3% приходится на служебную информацию. Однако при большом количестве переменных приложения и, особенно, сохраняемых (PERSISTENT) переменных, служебная информация в секции кода может составлять сотни килобайт.

Адаптированной среда исполнения CoDeSys для платформ CPM71x поддерживает до 16-ти циклических и до 64-х ациклических задач.

Если пользователь не добавил ни одной задачи в ресурс **Task Configuration**, то программа PLC_PRG будет исполняться под управлением скрытой циклической задачи *DefaultTask* с периодом, заданным в ресурсе **PLC Configuration** для параметра *CPM71x ... Programmable Controller:Sample Rate*.

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 *CPM71x...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**.

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. 4.2.3.3.

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. 4.2.3.3), 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 *I/O Modules–FBUS Diagnostics* контроллера можно использовать следующую декларацию:

```
VAR
   fbusDiagnostics AT%IB17    : FIODiagnostics;
END_VAR
```

Пусть в конфигурацию контроллера добавлены 4 модуля аналогового ввода типа AIM727 и 4 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 AIM727 и AIM728. Суммарное количество каналов составляет 4 * 4 + 4 * 4 = 32, а значит в конфигурации сети контроллера для передачи 48-ми значений типа REAL должно быть создано 32 выходные переменные типа REAL. Пусть первая из 32-х созданных переменных имеет адрес %QB2561 в области выходных данных среды исполнения.

Программа, преобразующая показания 4-ти модулей аналогового ввода типа AIM727, 4-х модулей аналогового ввода типа AIM728 и выводящая результаты в MODBUS, может выглядеть следующим образом:

```
PROGRAM PLC_PRG
   VAR CONSTANT
          AIM727_ARRAY_SIZE :INT := 3;
          AIM728_ARRAY_SIZE :INT := 3;
          NETWORK_BUF_BOUND := 31;
   END_VAR
   VAR
   (* Адрес первого канала первого модуля AIM727 из 4-х – %IB33 *)
          aim727_in AT %IB33 : ARRAY [0..AIM727_ARRAY_SIZE] OF AIM727_inputs;
   (* Адрес первого канала первого модуля AIM728 из 4-х – %IB101 *)
          aim728_in AT %IB101 : ARRAY [0..AIM728_ARRAY_SIZE] OF AIM728_inputs;
   (* Адрес 1-й выходной сетевой переменной из 32-ч – %QB2561 *)
          networkBuffer AT%QB2561 : ARRAY [0.. NETWORK_BUF_BOUND] OF REAL;
   (* Массив блоков обработки показаний модулей AIM727 *)
          aim727_conv : ARRAY [0..AIM727_ARRAY_SIZE] OF AIM727_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 AIM727_ARRAY_SIZE DO
          aim727_conv[i](inputs:= aim727_in[i], diagnostics=> , outputs=> );
          networkBuffer[netBufferIndex] := aim727_conv[i].outputs.vout0;
          netBufferIndex := netBufferIndex + 1;
          networkBuffer[netBufferIndex] := aim727_conv[i].outputs.vout1;
          netBufferIndex := netBufferIndex + 1;
          networkBuffer[netBufferIndex] := aim727_conv[i].outputs.vout2;
          netBufferIndex := netBufferIndex + 1;
          networkBuffer[netBufferIndex] := aim727_conv[i].outputs.vout3;
          netBufferIndex := netBufferIndex + 1;

   END_FOR;

   FOR i := 0 TO AIM728_ARRAY_SIZE DO
          aim728_conv[i](inputs:= aim728_in[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 *AIM727_inputs* и *AIM728_inputs*. Массив aim727_inputs, состоящий из 4-х элементов типа *AIM727_inputs*, размещается, начиная с адреса первого канала первого модуля AIM727 из 4-х имеющихся в конфигурации контроллера. Массив aim728_inputs, состоящий из 4-х элементов типа *AIM728_inputs*, размещается, начиная с адреса первого канала первого модуля AIM728 из 4-х имеющихся в конфигурации контроллера. Кроме того, для вывода в MODBUS в программе объявлен массив из 32 переменных типа REAL, которые ссылаются на область выходных данных прикладной программы, начиная с адреса %QB2561, т.е. с того места, где располагается первая выходная переменная сервиса MODBUS.

Далее, в программе объявлены массивы функциональных блоков типа *AIM727_STIN* и *AIM728_STIN* соответственно, каждый из которых состоит из 4-х элементов. Вызовы блоков преобразования выполняются в двух циклах. Теперь в случае добавления каких-либо модулей перед первыми 4-мя AIM727 достаточно будет скорректировать значения адресов, на которые ссылаются переменные-массивы aim727_inputs и aim727_inputs, заглянув в секцию **PLC Configuration.** Если же какие-нибудь модули вставляются между первыми 4-мя AIM727 и группой из 4-х AIM728, нужно будет скорректировать значение адреса, на который ссылается переменная-массив aim728_inputs. Кроме того, имеется возможность считывать значения всех 32-х 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.

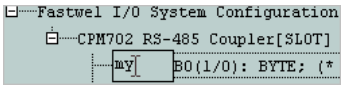
Основным недостатком данного способа является необходимость коррекции ссылок AT% в декларациях переменных при изменении структуры образа процесса, например, из-за вставки или удаления модулей ввода-вывода или коммуникационных объектов.

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:
   F_EVENT_ON_INIT:
```

```
            LinkTasks();
    F_EVENT_POWER_ON:
    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.

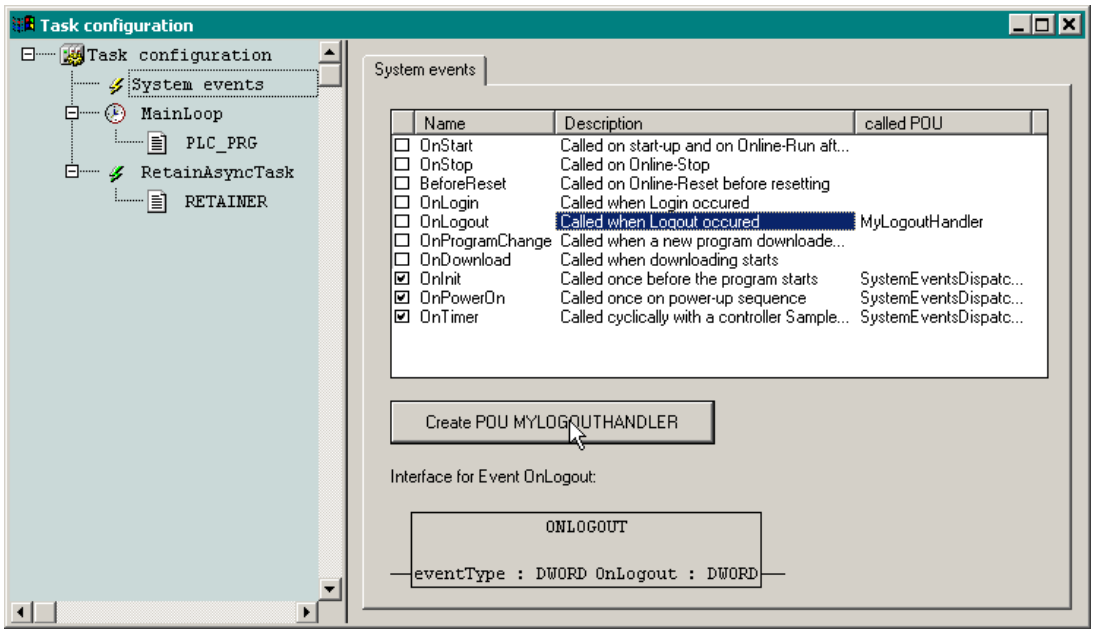


Рис. 24. Добавление обработчика системного события

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.

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 соединением среды CoDeSys с контроллером 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 CPM711, CPM712 и CPM713 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:**).

Обратите внимание, что, начиная с версии 2.65 системного программного обеспечения контроллера CPM713, пароль, заданный командой *setpwd*, может также использоваться для установления соединения с CPM713 по протоколу ftp. При этом в качестве имени пользователя должна использоваться строка *FastwelCPM713*.

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.

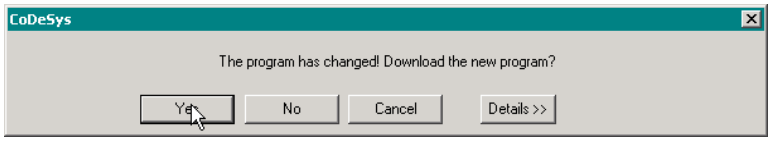


Рис. 26. Предложение загрузить программу

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.

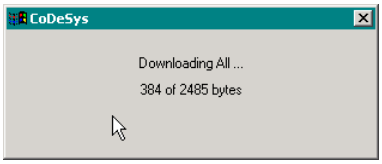


Рис. 27. Отображение хода загрузки программы

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.

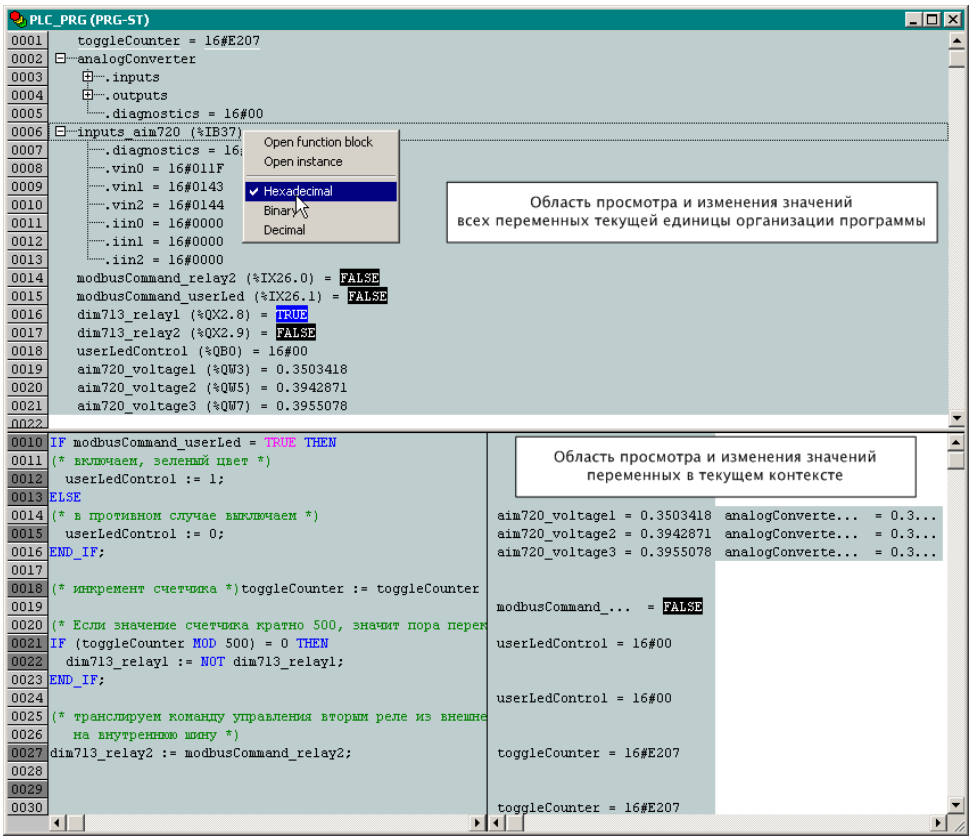


Рис. 28. Просмотр переменных

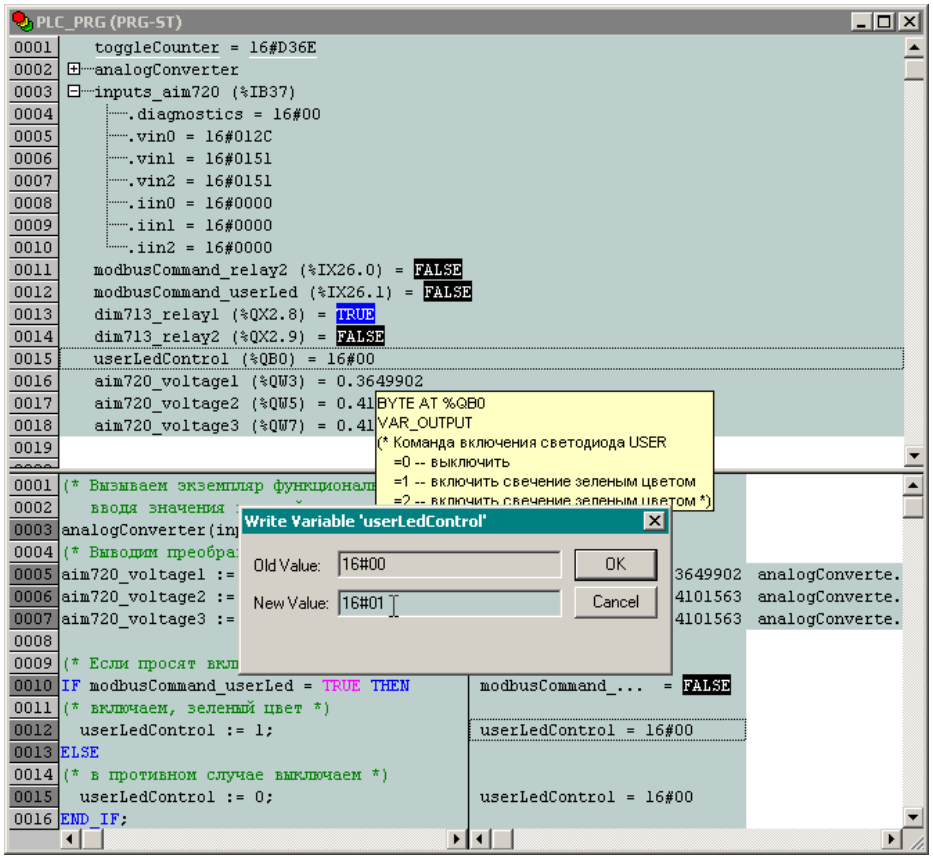


Рис. 29. Подготовка к изменению значения переменной

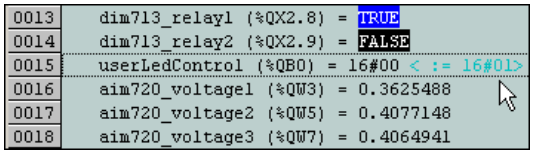


Рис. 30. Подготовленное значение переменной

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. 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 данных пользовательского приложения, сохраненных на основном дисковом накопителе, 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 9

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
20029	Команда **Online–Display Flow Control** запрещена, если в коде имеется хотя бы одна точка останова
20030	Установка точек останова запрещена во время выполнения **Online–Display Flow Control**

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 платформ CPM71x 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 для реализации собственных протоколов обмена через доступные пользовательской программе последовательные порты контроллера.
5. FastwelModbusServer.lib – реализует функциональность подчиненного узла сети MODBUS RTU/ASCII через доступные пользовательской программе последовательные порты контроллера.
6. SysLibRtc.lib – библиотека доступа к часам-календарю с автономным питанием, входящим в состав контроллеров CPM71x. Информация по использованию функций данной библиотеки приведена в документации и справочной системе на системные библиотеки CoDeSys. В текущей версии среды исполнения не реализована (всегда возвращает 1) функция *SysRtcGetHourMode()*. Функция *SysRtcCheckBattery()* возвращает 0, если разряжена батарея автономного питания статической энергонезависимой памяти и часов.
7. SysLibTime.lib – библиотека для работы с абсолютным временем. Информация по использованию функций данной библиотеки приведена в документации и справочной системе на системные библиотеки CoDeSys.
8. FastwelUtils.lib – библиотека вспомогательных функций.
9. FastwelGPS.lib – библиотека для синхронизации системных часов контроллера с часами GPS-приемника. Для связи контроллеров CPM711/CPM712/CPM713 с GPS- приемником может использоваться коммуникационный порт контроллера, расположенный на его передней панели под пластиковой крышкой, или порт интерфейсного модуля NIM742, подключенного к шине FBUS контроллера. Информация по подключению GPS приемника к модулю NIM742 и его конфигурированию для выполнения данной функции приведена в п. 3.3.17.5 документа *ИМЕС.00300-02 33 01 Модули ввода-вывода Fastwel I/O. Руководство программиста*.
10. FastwelHayesModem.lib – предназначена для обмена данными по выделенным и коммутируемым каналам связи. Библиотека содержит функции для работы с hayes-совместимыми модемами, поддерживающими AT-команды.
11. FastwelHayesModemControls.lib – написана на языке Structured Text в среде CoDeSys 2.3 и предназначена для обмена данными через модем при помощи специального функционального блока FW_HAYES_MODEM.
12. FastwelSMS.lib – предназначена для приема и отправки SMS-сообщений и содержит функции, необходимые для управления GSM-модемами с помощью AT-команд.
13. FastwelSMSControls.lib – написана на языке Structured Text в среде CoDeSys 2.3 и предназначена для приема и отправки SMS-сообщений через GSM-модем при помощи специального функционального блока FWSMS.
14. 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.
15. 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

6.2.1.1. Особенности библиотеки FastwelSysLibFile.lib

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: 0 означает FALSE, а не 0 – TRUE.

Основное отличие библиотеки FastwelSysLibFile.lib от SysLibFile.lib состоит в том, что в функциях FastwelSysLibFile.lib обеспечена возможность работы с абсолютными и относительными путями файловой системы контроллера. Для работы с относительными путями на диске контроллера отведен специальный каталог *\user*.

6.2.1.2. Относительный путь

Относительный путь начинается с имени файла или каталога без каких-либо разделителей пути, предшествующих первому компоненту пути. Например:

```
Temp1\Temp2\my_file.txt
```

указывает на файл *my_file.txt* в подкаталоге *Temp1\Temp2* специального каталога *user*.

Операции открытия файлов для чтения, которым передается относительный путь, в первую очередь обращаются в каталог запуска системного программного обеспечения контроллера, поскольку загрузка файлов командой **Online–Write file to PLC** может быть выполнена только в данный каталог. Если в каталоге запуска файл с заданным именем не найден, то выполняется поиск файла в каталоге *\user*.

6.2.1.3. Абсолютный путь

Абсолютный путь начинается с префикса ****. Например:

```
\FixedDisk1\Temp1\my_file.txt
```

указывает на файл *my_file.txt* в каталоге *Temp1* основного дискового накопителя контроллера.

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, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\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, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

Возвращаемый результат:

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

6.2.2.3. FwSysFileGetTime

```
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, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

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. Файл создается относительно каталога \user, если используются относительные пути.. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей файл создается относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

FileSource: STRING

Имя файла-оригинала. Поиск файла выполняется относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

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, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

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, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

6.2.2.7. FwSysFileOpen

Функция открывает файл для чтения, 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 и используется относительный путь к файлу, поиск его сначала выполняется в каталоге запуска системного программного обеспечения, а затем – в каталоге \user.

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

Входные параметры:

FileName: STRING

Имя открываемого файла. Поиск файла выполняется относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

При использовании абсолютных путей поиск файла выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на основном дисковом накопителе *)
fileName2 : STRING := '\FixedDisk1\subdir\MyData.dat';
```

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.

Пример:

```
(*****************************************************************
   Программа LOGGER сохраняет данные в файле FileName.

   Массив данных определяется входным параметром pData,
   при этом количество байт, подлежащих сохранению, задается
   параметром DataSize.
   Параметр AppendMode позволяет дописывать данные в конец файла.
   Выходная переменная LastResult при равенстве TRUE свидетельствует
   об успехе.
*****************************************************************)
PROGRAM LOGGER

VAR_INPUT
   FileName            : STRING := '';
   (* Имя файла *)
   pData                       : POINTER TO BYTE := 0;
```

```
    (* Указатель на данные, подлежащие сохранению *)
    DataSize            : DWORD :=0;
    (* Количество байт для сохранения *)
    AppendMode    : BOOL := FALSE;
    (* Режим сохранения: TRUE -- добавление в конец файла *)
END_VAR

VAR_OUTPUT
    LastResult : BOOL := FALSE;
    (* Результат последней операции *)
    OperCount : DWORD := 0;
    (* Кол-во вызовов *)
    ErrorCount : DWORD := 0;
    (* Кол-во вызовов, завершившихся ошибкой *)
    MinExecTime : TIME := T#64h;
    (* Минимальное время исполнения *)
    MaxExecTime : TIME := T#0ms;
    (* Максимальное время исполнения *)
    LastExecTime : TIME := T#64h;
    (* Время исполнения последнего вызова *)
END_VAR

VAR_TEMP
    start_time : TIME;
    handle : DWORD;
    data_written : DWORD;
    file_size : DINT;
END_VAR

(* Код программы начинается здесь *)
LastResult := FALSE;

(* проверяем корректность входных параметров *)
IF pData = 0 OR DataSize = 0 OR LEN(FileName) = 0 THEN
    RETURN;
END_IF

(* берем текущее время *)
start_time := TIME();

(* инкремент счетчиков операций и ошибок *)
OperCount := OperCount + 1;
ErrorCount := ErrorCount + 1;

(* определяем размер файла *)
file_size := FwSysFileGetSize(FileName);

(* открываем файл для записи, если файла нет, он будет создан *)
handle := FwSysFileOpen(FileName, 'cw');

IF handle <> 0 THEN
    IF AppendMode AND file_size > 0 THEN
            (* если нужно дописывать, встаем в конец *)
            IF FwSysFileSetPos(handle, file_size) = 0 THEN
                    FwSysFileClose(handle);
                    RETURN;
            END_IF
    END_IF

    (* записываем данные *)
    data_written := FwSysFileWrite(handle, pData, DataSize);

    IF data_written = DataSize THEN
            (* если все получилось, декремент счетчика ошибок *)
            LastResult := TRUE;
            ErrorCount := ErrorCount - 1;
    END_IF

    FwSysFileClose(handle);
END_IF

(* определяем времена исполнения *)
LastExecTime := TIME() - start_time;
MinExecTime := MIN(LastExecTime, MinExecTime);
MaxExecTime := MAX(LastExecTime, MaxExecTime);
```

```
END_PROGRAM
```

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 каталог с именем, совпадающим с последним компонентом пути, переданным в качестве параметра, а также все отсутствующие каталоги, имена которых предшествуют последнему компоненту.

Пример относительного пути к создаваемому каталогу:

```
MyDir1\Temp1\TargetDir
```

В данном случае в каталоге user сначала будет выполнен поиск каталога MyDir1. При отрицательном результате MyDir1 будет создан в каталоге user, после чего в нем будет создан каталог Temp1, а в каталоге Temp1 – TargetDir.

Пример абсолютного пути к создаваемому каталогу:

```
\FixedDisk1 \Temp1\TargetDir
```

В данном случае в корневом каталоге \ файловой системы контроллера сначала будет выполнен поиск каталога FixedDisk1. При отрицательном результате FixedDisk1 будет создан в каталоге \, после чего в нем будет создан каталог Temp1, а в каталоге Temp1 – TargetDir. Таким образом, перед созданием каталога на съемном USB-накопителе следует убедиться в наличии основного диска вызова FwSysDirExist с передачей ‘\FixedDisk1’ в качестве параметра.

Прототип функции FwSysDirCreate

```
FUNCTION FwSysDirCreate : WORD
VAR_INPUT
   DirName: STRING;
END_VAR
VAR
END_VAR
;
END_FUNCTION
```

Входные параметры:

```
DirName: STRING
```

Полный путь к создаваемому каталогу. Компоненты пути являются именем каталогов, которые будут созданы в существующих каталогах, представленных предшествующими компонентами пути.

Создание производится относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

```
(* в каталоге \user будет создан подкаталог \MyDir1 *)
dirName1 : STRING := 'MyDir1';
(* в каталоге \user будут созданы подкаталоги \SubDir и \Subdir\MyDir2 *)
dirName2 : STRING := 'Subdir\MyDir2';
```

При использовании абсолютных путей создание каталога выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* в каталоге \ будет создан подкаталог \Temp *)
dirName1 : STRING := '\Temp';
(* в каталоге \FixedDisk1 будет создан подкаталог subdir, а в нем -- mydir *)
dirName2 : STRING := '\FixedDisk1\subdir\mydir';
```

Возвращаемый результат:

Функция возвращает ненулевое значение, если целевой каталог уже существует или успешно создан. В противном случае возвращается 0.

6.2.2.15. FwSysDirExist

Возвращает ненулевое значение, если найден каталог с заданным именем.

```
FUNCTION FwSysDirExist : WORD
VAR_INPUT
   DirName: STRING;
END_VAR
VAR
END_VAR
;
END_FUNCTION
```

Входные параметры:

```
DirName: STRING
```

Полный путь к искомому каталогу. Поиск производится относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

```
(* в каталоге \user ищется подкаталог \MyDir1 *)
dirName1 : STRING := 'MyDir1';
(* в каталоге \user ищется подкаталог\Subdir\MyDir2 *)
dirName2 : STRING := 'Subdir\MyDir2';
```

При использовании абсолютных путей поиск каталога выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* в каталоге \ ищется подкаталог \Temp *)
dirName1 : STRING := '\Temp';
(* поиск в каталоге \FixedDisk1 *)
dirName2 : STRING := '\FixedDisk1\subdir\mydir';
```

6.2.2.16. FwSysDirRemove

Возвращает ненулевое значение, если был обнаружен и успешно удален подкаталог, представленный последним компонентом заданного пути.

Пример относительного пути к удаляемому каталогу:

```
MyDir1\Temp1\TargetDir
```

В данном случае в каталоге user сначала будет выполнен поиск каталога MyDir1, и, в случае успеха, в MyDir1 будет выполнен поиск Temp1, после чего в Temp1 будет сделана попытка удалить TargetDir.

Пример абсолютного пути к удаляемому каталогу:

```
\FixedDisk1\Temp1\TargetDir
```

В данном случае в корневом каталоге \ файловой системы контроллера сначала будет выполнен поиск каталога FixedDisk1, после чего в \FixedDisk1 – поиск Temp1, а затем – попытка удалить TargetDir в Temp1.

Причиной неудачного удаления может быть наличие файлов в удаляемом каталоге.

```
FUNCTION FwSysDirRemove : WORD
VAR_INPUT
   DirName: STRING;
END_VAR
VAR
END_VAR
;
END_FUNCTION
```

Входные параметры:

```
DirName: STRING
```

Полный путь к удаляемому каталогу. Поиск производится относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

```
(* в каталоге \user будет удален подкаталог \MyDir1 *)
dirName1 : STRING := 'MyDir1';
(* в каталоге \user\Subdir будет удален подкаталог MyDir2 *)
dirName2 : STRING := 'Subdir\MyDir2';
```

При использовании абсолютных путей поиск каталога выполняется относительно корневого каталога \ файловой системы контроллера. Пример имен, в которых используются абсолютные пути:

```
(* в каталоге \ будет удален подкаталог \Temp *)
dirName1 : STRING := '\Temp';
```

6.3. Библиотека FastwelTasksExchange.lib

6.3.1. Общие сведения

Библиотека содержит функцию F_IecTasks_linkVariables, предназначенную для связывания задач по данным, а также вспомогательные функции:

F_IecTasks_getCount – возвращает общее количество циклических и ациклических задач;

F_getProjectName – возвращает имя проекта, загруженного в контроллер;

F_IecTasks_getInfo – возвращает диагностическую информацию о циклической или ациклической задаче.

Описание применения функции F_IecTasks_linkVariable приведено в п. 0 настоящего руководства.

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,* из приложений пользователя.. По сути 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_cpm711.pro, который входит в пакет адаптации CoDeSys для Fastwel I/O. Открывать коммуникационный порт следует при обработке системного события *OnInit*. 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.

На контроллере CPM712 возможно использование библиотекой порта COM2 интерфейса RS-485. Для этого необходимо, чтобы в проекте для CPM712 для элемента конфигурации *CPM712 MODBUS RTU/ASCII Programmable Controller–Serial Port* была установлена опция *Not Used*.

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 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 должен производиться в коде приложения, выполняемом на контексте циклической, ациклической или сервисной задачи. Библиотека позволяет открыть одновременно не более 32 устройств.

```
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). Допустимые значения: COM1, 101–164 – для портов, организованных посредством элементов *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module* в конфигурации контроллера. Если в проекте для CPM712 для элемента конфигурации *CPM712 MODBUS RTU/ASCII Programmable Controller–Serial Port* установлена опция *Not Used*, то возможно использовать значение COM2.

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: 16#FFFFFFFF.

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 CPM711/CPM712/CPM713 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*. На контроллере CPM712 библиотека может быть использована для коммуникационного порта COM2 интерфейса RS-485. Для этого необходимо, чтобы в проекте для CPM712 для элемента конфигурации *CPM712 MODBUS RTU/ASCII Programmable Controller–Serial Port* была установлена опция *Not Used*.

Примечание. 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.

Пример использования библиотеки имеется в проекте *fmbserverlib_test_cpm713.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 CPM711 и CPM713 допустимые значения поля Port: 1(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 CPM712 допустимые значения поля Port: 1(COM1), 2, 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...), или RETAIN-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...) или RETAIN-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; – 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...), или RETAIN-сегментах; – заданный индекс программной единицы **pDiagnosticsDescriptor^.PouIndex**) не относится к множеству индексов программных единиц, вызываемых из каких-либо циклических или ациклических задач приложения.

Код	Значение
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, подлежащего 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

ВНИМАНИЕ!

Системная библиотека FastwelModbusServer.lib не предусматривает возможности продолжения правильной работы приложения после горячего обновления (ONLINE CHANGE) в случае, если в обновленном приложении изменились адреса или размеры переменных, передаваемых функции *FwModbusServerInit*. В случае изменения размеров или адресов переменных, передаваемых функции *FwModbusServerInit*, всегда используйте полную загрузку приложения в контроллер.

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 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 идентифицирует сегмент энергонезависимых переменных (VAR RETAIN).

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 идентификаторов DATAID_INPUT и DATAID_OUTPUT возвращаются указатели на сегменты входных и выходных данных соответственно. В качестве адреса сегмента входных данных возвращается адрес сегмента входных данных сервисной задачи.

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.

Обратите внимание, что каждая циклическая задача имеет собственный сегмент входных данных, доступ к которому при помощи данной функции невозможен.

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, 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 FastwelGPS.lib

6.8.1. Общие сведения

Библиотека FastwelGPS.lib предназначена для определения точного времени на основе данных GPS-приемника, а также для коррекции системных часов контроллера в соответствии с этим временем.

Под GPS-приемником понимается любое устройство, передающее телеграммы протокола NMEA 0183 по интерфейсу RS-232. Сервис FastwelGPS.lib является пассивным “слушателем” данных GPS-приемника. Первоначальное конфигурирование GPS-приемника, установка параметров передачи, конфигурация набора и периода, передаваемых им сообщений и т.п., выполняются отдельно с помощью программного обеспечения, входящего в комплект поставки используемого GPS-приемника.

Для связи контроллеров CPM711/CPM712/CPM713 с GPS-приемником может использоваться

- коммуникационный порт контроллера, расположенный на его передней панели под пластиковой крышкой (COM1)
- порт интерфейсного модуля NIM742, подключенного к шине FBUS контроллера.

  В конфигурации модулей ввода-вывода контроллера данный модуль должен быть представлен элементом *NIM742 RS-232 1xUART Stream Module*. Идентификатор COM-порта модуля формируется по правилу

  100+n

  где n – позиция модуля, начиная с 1, в секции *...I/O Modules* ресурса **PLC Configuration**.

Для функционирования предоставляемого библиотекой сервиса, GPS-приемник должен передавать (среди прочих возможных NMEA телеграмм) телеграммы типа “RMC”. Именно эти сообщения используются сервисом в качестве источника данных GPS-времени и соотношения (синхронизации) с ним текущего системного времени контроллера.

Следует отметить, что привязка времени к эталонным часам GPS-приемника (синхронизация с часами GPS-приемника) только по данным NMEA RMC телеграмм имеет относительно невысокую

точность. Со стороны GPS-приемника гарантируется, что передача пакета с меткой, соответствующей текущему индицируемому GPS-приемником времени, начнется с момента его начала и завершится до начала следующего индицируемого момента времени. Фактическое положение пакета внутри указанного интервала, в общем случае, не определено и зависит от объема, состава и скорости передаваемой GPS-приемником информации. Так, если GPS-приемник сконфигурирован на передачу RMC телеграмм с периодом 1 с, то оценка точности привязки данных времени принимаемых из сообщения GPS-приемника к его часам составит 1 с. Если период передачи RMC телеграмм равен 100 миллисекундам, то точность привязки составит 100 мс.

Точность привязки данных времени, выполняемой сервисом, может быть кардинально улучшена и достигать значений вплоть до 1 мс в системах, использующих GPS-приемник с сигналом 1PPS. Для связи с приемником в этом случае должен использоваться модуль NIM742. Выход 1PPS GPS-приемника должен быть подключен к входу CTS модуля NIM742. При такой аппаратной конфигурации секундная часть эталонного времени берется сервисом из данных RMC телеграмм, а его миллисекундная составляющая привязывается к фронту сигнала 1PPS. Для работы с сигналом 1PPS не требуется никакой дополнительной настройки конфигурации сервиса и NIM742. Учет синхросигнала 1PPS при его фиксации производится автоматически. Модуль NIM742 должен иметь версию микропрограммы 2.7 и выше.

6.8.2. Принцип работы

Работа сервиса, предоставляемого библиотекой FastwelGPS.lib (далее сервис FastwelGPS или просто сервис), начинается с вызова приложением пользователя функции FwGPS_initListener, выполняющей инициализацию “слушателя” данных GPS-приемника. Возвращенный функцией FwGPS_initListener системный идентификатор “слушателя” должен использоваться впоследствии при вызове других функций библиотеки.

Пример:

```
VAR CONSTANT
   (* Значение невалидного хэндла слушателя данных GPS-приемника *)
   INVALID_GPS_HANDLE : DWORD := 16#FFFFFFFF;
END_VAR

VAR
   (* Параметры настройки интерфейса связи с GPS-приемником
       индекс COM-порта: 101 – NIM742 (первый в списке модулей ввода/вывода
       скорость передачи: 115200 Кбит/с
       число стоповых бит в кадре: 1
   *)
   listener_settings : FW_GPS_LISTENER_SETTINGS :=
                       (Port:=101, BaudRate:=115200, StopBits:=1);

   (* Хэндл слушателя *)
   nr_listener_handle : DWORD := INVALID_GPS_HANDLE;
END_VAR

IF nr_listener_handle = INVALID_GPS_HANDLE THEN
  nr_listener_handle := FwGPS_initListener(ADR(listener_settings));
  IF nr_listener_handle = INVALID_GPS_HANDLE THEN
    (* Ошибка. Проверить параметры порта *)
    RETURN;
  ELSE
    (* Сервис FastwelGPS запущен *)
  END_IF
END_IF
```

После успешной инициализации “слушателя” сервис FastwelGPS находится в активном состоянии, в котором принимает и обрабатывает данные GPS-приемника.

После получения достоверных данных времени от GPS-приемника сервис производит их синхронизацию, то есть привязку к системному таймеру контроллера, оценивает точность выполненной операции и определяет значение расхождения системных часов контроллера с часами GPS-приемника. Данные, относящиеся к последней выполненной синхронизации хранятся в памяти сервиса до того времени, пока не будет выполнена следующая операция синхронизации, после которой они заменяются новыми значениями. Если, по каким либо причинам, новые достоверные данные времени от GPS приемника приходить не будут, то данные последней синхронизации будут действительны (использоваться сервисом) в течение 24 суток. На основе этих данных и интервала

времени, прошедшего с момента синхронизации (измеряется системным таймером), сервис производит оценку GPS времени в будущие моменты. Точность оценки GPS времени определяется точностью выполнения самой операции синхронизации и величиной временного интервала прошедшего с ее момента (из-за нестабильности системного таймера).

Сразу же, после выполнения операции синхронизации и вычисления значения расхождения системных часов контроллера с часами GPS-приемника, сервис FastwelGPS автоматически корректирует системные часы контроллера при условии, что:

1. Значение расхождения укладывается во временной интервал, определяемый минимальным и максимальным допустимыми значениями. Пороговые значения для «автокоррекции» устанавливаются (и переопределяются) пользователем в любой момент после успешной инициализации “слушателя” (функция FwGPS_configTime). **Обратите внимание, по умолчанию (сразу же после инициализации слушателя) пороговые параметры имеют значения исключающие возможность коррекции при любом значения расхождения.**
2. Автокоррекция не запрещена.

Запрет автокоррекции устанавливается, если текущее значение расхождения превысило максимальное пороговое значение, при этом операция автокоррекции запрещается и в дальнейшем, начиная с этого момента. Автоматическая коррекция вновь разрешается после выполнения пользователем принудительной коррекции системных часов контроллера функцией FwGPS_syncTime.

После успешной инициализации “слушателя” (функция FwGPS_initListener) и до его закрытия (функция FwGPS_closeListener) пользователь имеет возможность:

- в любой момент получать текущее время GPS-приемника и связанную с ним диагностическую информацию – функция FwGPS_getTime;
- в любой момент выполнять операцию программной коррекции (коррекции «вручную») системных часов контроллера к текущему времени GPS-приемника – функция FwGPS_syncTime;
- настраивать, включать/выключать функционал сервиса по автоматической коррекции системных часов контроллера – функции FwGPS_configTime и FwGPS_syncTime.

После успешного вызова функции FwGPS_closeListener сервис FastwelGPS деактивируется, используемый им коммуникационный порт освобождается.

6.8.3. Описание функций

6.8.3.1. FwGPS_initListener

Функция FwGPS_initListener выполняет инициализацию и запуск сервиса “слушателя” данных GPS-приемника:

- открывает и конфигурирует коммуникационный порт, используемый для связи с GPS-приемником,
- активизирует сервис приема и обработки данных, передаваемых GPS-приемником.

```
FUNCTION FwGPS_initListener : DWORD
VAR_INPUT
   pSettings: POINTER TO FW_GPS_LISTENER_SETTINGS;
END_VAR
;
END_FUNCTION
```

Входные параметры:

```
pSettings: POINTER TO FW_GPS_LISTENER_SETTINGS
```

Указатель на переменную типа FW_GPS_LISTENER_SETTINGS, содержащую требуемые значения параметров коммуникационного порта. Определение полей структуры и допустимые значения:

```
TYPE FW_GPS_LISTENER_SETTINGS :
STRUCT
   (* Индекс коммуникационного порта, к которому подключен GPS приемник         *)
   (* Например: 1 для COM1, 2 для COM2, …                                        *)
   (* Для NIM742 – порядковый номер модуля на шине  FBUS + 100: 101,102, …      *)
```

```
   Port: BYTE;

   (* Параметры передачи данных GPS приемником                                  *)
   BaudRate: DWORD;          (* Скорость передачи: 1200, 2400, 4800, 9600, *)
                             (* 19200, 38400, 57600, 115200 *)

   StopBits: BYTE;           (* Число стоповых бит: 1, 2 *)
   (* Всегда для NMEA 0183 устройств:                                           *)
   (* Число бит данных - 8. Бит 7 всегда 0                                      *)
   (* Четность – нет                                                            *)
END STRUCT
END TYPE
```

ВНИМАНИЕ! В контроллерах CPM711/CPM712/CPM713 коммуникационный порт, расположенный на его передней панели под пластиковой крышкой, имеет идентификатор COM1. Для того, чтобы иметь 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.

Возвращаемый результат:

Системный идентификатор, который в дальнейшем используется в вызовах других функций библиотеки. В случае ошибки возвращается значение: 16#FFFFFFFF.

6.8.3.2. FwGPS_closeListener

Функция завершает работу сервиса библиотеки и освобождает используемый коммуникационный порт.

```
FUNCTION FwGPS_closeListener : BOOL
VAR_INPUT
   hL: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

hL: DWORD
Системный идентификатор слушателя данных GPS-приемника.

Возвращаемый результат:

Функция возвращает TRUE при успешном завершении, иначе FALSE.

6.8.3.3. FwGPS_configTime

Функция устанавливает пороговые значения расхождения времени системных часов контроллера со временем часов GPS-приемника, в пределах которых разрешается его автоматическая коррекция. Значения параметров по умолчанию, после инициализации “слушателя”, фактически исключают возможность выполнения такой синхронизации для любого значения расхождения и должны/могут быть переопределены пользователем в любой момент, в соответствии с условиями конкретной задачи. Следует отметить, что при любых значениях пороговых параметров, остается возможность выполнения принудительной коррекции системных часов функцией FwGPS_syncTime.

Если, в процессе работы, после получения очередных достоверных данных времени GPS-приемника, сервис определяет, что текущее расхождение времен превышает установленный для него максимальный пороговый интервал, то, как указывалось, текущая коррекция не производится. Также, начиная с этого момента, запрещаются все последующие коррекции, не зависимо от значения расхождения. Работа механизма автокоррекции будет возобновлена после выполнения пользователем принудительной “ручной” синхронизации функцией FwGPS_syncTime.

```
FUNCTION FwGPS_configTime : BOOL
VAR_INPUT
   hL: DWORD;
   pSettings: POINTER TO FW_GPS_SRVTIME_SETTINGS;
END_VAR
;
END_FUNCTION
```

Входные параметры:

hL: DWORD

Системный идентификатор слушателя данных GPS-приемника.

pSettings: POINTER TO FW_GPS_SRVTIME_SETTINGS
Указатель на переменную типа FW_GPS_SRVTIME_SETTINGS, в которую записаны требуемые параметры конфигурации. Определение полей структуры и допустимые значения:

```
TYPE FW_GPS_SRVTIME_SETTINGS:
STRUCT
(*
   Временные интервалы в мс, определяющие пороговые значения расхождения локального
   времени с временем GPS приемника для разрешения выполнения автоматической коррекции
   локальных часов. Для разрешения подводки часов вперед и назад используются
   отдельные параметры. Данные настройки позволяют управлять сервисом автоматической
   кореекции от полного разрешения до полного запрещения.
*)

   (*
       1) Абсолютное значение максимального расхождения часов контроллера и
       GPS-приемника, которое допускается корректировать в автоматическом режиме.

       Задание для MaxAdjustForwardThreshold и MaxAdjustBackwardThreshold любого
       значения, меньшего 0, определяет бесконечно большую величину порога.

       Если в какой-то момент времени расхождение превышает заданный порог, то, начиная
       с него, автоматическая коррекция выключается и не выполняется до тех пор, пока
       пользователь не выполнит принудительную синхронизацию часов функцией
       FwGPS_syncTime.
   *)

   (*
       Максимальный интервал коррекции системных часов вперед (системные часы
       контроллера отстают).
   *)
   MaxAdjustForwardThreshold: DINT;

   (*
       Максимальный интервал коррекции локальных часов назад (системные часы
       контроллера спешат).
   *)
   MaxAdjustBackwardThreshold: DINT;


   (*
       2) Абсолютное значение расхождения часов контроллера и GPS-приемника, которое
       может не корректироваться сервисом (допустимая точность хода).
       Данная настройка может использоваться для исключения частых (не нужных по смыслу
       прикладной системы) корректировок системных часов контроллера.
   *)
   (*
       Минимальный интервал коррекции системных часов контроллера вперед (допустимое
       отставание).
   *)
   MinAdjustForwardThreshold: DINT;

   (*
       Минимальный интервал коррекции системных часов контроллера назад (допуситмое
       «убегание»).
   *)
   MinAdjustBackwardThreshold: DINT;
END STRUCT
END TYPE
```

Таким образом, если системные часы контроллера отстают, то их коррекция вперед выполняется автоматически, если она разрешена и выполняняется условие

MinAdjustForwardThreshold <= LagTime < MaxAdjustForwardThreshold

где LagTime – положительная разность показаний часов GPS-приемника и системных часов контроллера.

Если системные часы контроллера опережают часы GPS-приемника (спешат), то их коррекция назад выполняется автоматически, если она разрешена и при выполняется условие:

MinAdjustBackwardThreshold <= OverrunTime < MaxAdjustBackwardThreshold

где OverrunTime – положительная разность показаний системных часов контроллера и часов GPS-приемника.

Возвращаемый результат:

Функция возвращает TRUE в случае успеха, иначе FALSE.

6.8.3.4. FwGPS_syncTime

Выполняет безусловную синхронизацию системных часов контроллера с часами GPS- приемника. Если до вызова данной функции автоматическая коррекция системных часов контроллера была запрещена, то ее успешное выполнение вновь активирует данный механизм.

В отличие от автоматической коррекции, применение которой ограничено пороговыми значениями расхождения (см. описание функции FwGPS_configTime), функция FwGPS_syncTime выполняет коррекцию системного времени контроллера всегда, независимо от значения расхождения.

```
FUNCTION FwGPS_syncTime : BOOL
VAR_INPUT
   hL: DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

hL: DWORD
Системный идентификатор слушателя данных GPS-приемника.

Возвращаемый результат:

Функция возвращает TRUE при успешном завершении, иначе FALSE.

6.8.3.5. FwGPS_getTime

Возвращает оценку текущего значения времени часов GPS-приемника и диагностические параметры сервиса. При вычислении текущего значения используется значение времени, полученное от GPS- приемника в последней успешной операции синхронизации, к которому добавляется прошедший с ее момента интервал времени. Данный интервал времени измеряется при помощи системного таймера контроллера.

```
FUNCTION FwGPS_getTime : BOOL
VAR_INPUT
   hL: DWORD;
   pDT: POINTER TO SystemTimeDate;
   pDiags: POINTER TO FW_GPS_SRVTIME_DIAGNOSTICS;
END_VAR
;
END_FUNCTION
```

Входные параметры:

hL: DWORD
Системный идентификатор слушателя данных GPS-приемника.

pDT: POINTER TO SystemTimeDate
Необязательный параметр, который может содержать указатель на переменную типа SystemTimeDate для получения текущего значения времени часов GPS-приемника. Может быть равен 0. Структура SystemTimeDate определена в системной библиотеке SysLibTime.lib CoDeSys:

```
TYPE SystemTimeDate :
STRUCT
   (* Время в микросекундах                                          *)
   (* Младшая часть dwLowMsecs содержит время в секундах, начиная с 0 часов     *)
   (* 1 января 1970 года (соответствует определению типа DATE_AND_TIME).        *)
   dwLowMsecs: DWORD;

   (* Старшая часть dwHighMsec содержит дробную часть времени – число           *)
   (* микросекунд текущей секунды.                                              *)
   dwHighMsec: DWORD;

   (* Год, например, «2012»                                                     *)
   Year: UINT;
```

```
    (* Месяц, например, «12»                                                *)
    Month: UINT;

    (* День месяца, например, «32»                                          *)
    Day: UINT;

    (* Час времени суток, например, «13»                                    *)
    Hour: UINT;

    (* Минуты текущего часа, например, «43»                                 *)
    Minute: UINT;

    (* Секунды текущей минуты, например, «15»                               *)
    Second: UINT;

    (* Миллисекунды текущей секунды, например, «649»                        *)
    Milliseconds: UINT;

    (* День недели, например, «2» (Понедельник = 1)                         *)
    DayOfWeek: UINT;
END STRUCT
END TYPE
```

pDiags: POINTER TO FW_GPS_SRVTIME_DIAGNOSTICS
Необязательный параметр, который может содержать указатель на переменную типа FW_GPS_SRVTIME_DIAGNOSTICS для получения данных расширенной диагностики сервиса синхронизации времени. Определение полей структуры:

```
TYPE FW_GPS_SRVTIME_DIAGNOSTICS :
STRUCT
    (*
      Оценка точности синхронизации с часами GPS-приемника в мс.
    *)
    tiPrecision: DWORD;

    (*
      Расхождение между оценкой времени по часам GPS-приемника и временем
      по системным часам контроллера в мс.
      Системные часы контроллера отстают от часов GPS-приемника: значение > 0
      Системные часы контроллера опережают часы GPS-приемника: значение < 0
    *)
    tiLocalClockMistiming: DINT;


    (*
      Индикатор функционирования механизма автоматической коррекции
      системных часов контроллера по показаниям часов GPS-приемника.

      TRUE, если расхождение системных часов контроллера и часов GPS-приемника
      не превышало установленных порогов и автоматическая коррекция запущена.

      FALSE, если автоматическая коррекция выключена.

      Автоматическая коррекции выключается, если в результате сопоставления
      показаний системных часов контроллера и часов GPS-приемника полученное
      расхождение показаний превысило пороговое значение, установленное в конфигурации
      сервиса при вызове функции FwGPS_configTime.

      После выполнения принудительной синхронизации функцией FwGPS_syncTime механизм
      автокоррекции запускается и функционирует в пределах, установленных параметрами
      конфигурации.
    *)
    LocalClockAutoAjust: BOOL;

    (*
      Время последней выполненной коррекции системных часов контроллера.
      Совпадает со значением времени по часам GPS-приемника, записанным в системные
      часы контроллера в момент последней успешной коррекции.
    *)
    dtLocalClockAdjustment: SystemTimeDate;
END STRUCT
END TYPE
```

Возвращаемый результат:

В случае наличия достоверных данных времени от GPS-приемника возвращает TRUE. Иначе, если данных нет, FALSE. Данные от GPS-приемника, полученные более 24-х суток назад, утрачивают достоверность.

6.9. Библиотека FastwelHayesModem.lib

6.9.1. Назначение

Библиотека FastwelHayesModem.lib предназначена для обмена данными по выделенным и коммутируемым каналам связи. Библиотека содержит функции для работы с hayes-совместимыми модемами, поддерживающими AT-команды.

В контроллерах CPM711/CPM712/CPM713 комплекса Fastwel I/O данная библиотека может быть использована для установления соединения с удаленными модемами по выделенным или коммутируемым каналам связи, а также через сеть GSM. Модем может быть подключен к:

- коммуникационному порту, организованному на основе модулей NIM741/NIM742 и элементов *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module*;
- коммуникационному порту, расположенному на передней панели CPM711/712/713 под пластиковой крышкой;
- коммуникационному порту COM2 интерфейса RS-485 на контроллере CPM712 (для этого в проекте для CPM712 для элемента конфигурации *CPM712 MODBUS RTU/ASCII Programmable Controller-Serial Port* необходимо установить режим *Not Used*).

Примечание. 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.

Выбор режима работы модема и установление соединения (дозвон) производятся с помощью строк инициализации, содержащих AT-команды. Обмен данными с удаленным терминалом возможен только в режиме факс/данные (CSD для GSM-сетей).

Пример использования библиотеки имеется в проекте fw_hayes_modems_test.pro (для контроллера CPM713), который входит в пакет адаптации CoDeSys для Fastwel I/O (расположен в подкаталоге Examples\CPM71x\CPM713\Hayes_Modem каталога установки адаптации CoDeSys 2.3 для Fastwel I/O).

Общее количество модемов, которыми можно управлять с помощью библиотеки FastwelHayesModem.lib, ограничено 16.

6.9.2. Принцип работы

6.9.2.1. Общие сведения

Работа сервиса, предоставляемого библиотекой FastwelHayesModem.lib (далее “сервис модема” или просто “сервис”), начинается с вызова приложением пользователя функции FwModem_open, выполняющей запуск и инициализацию модема. Возвращенный функцией идентификатор должен впоследствии использоваться при вызове других функций библиотеки. После вызова FwModem_open сервис последовательно проходит через состояния HM_START, HM_INITIALIZATION в HM_READY. Для проверки текущего состояния сервиса служит функция FwModem_getStatus. Состояние сервиса HM_READY означает, что модем включен, успешно выполнил AT-команды из строки инициализации и готов к работе.

Из состояния HM_READY готовности к работе сервис модема можно перевести в режим установления соединения HM_CONNECTING, вызвав функцию FwModem_connect. Во входных параметрах функции содержится адрес строки с AT-командами, необходимыми для установления соединения.

При успешном установлении соединения модем переходит в режим HM_DATA_EXCHANGE, в котором с помощью функций FwModem_read, FwModem_write можно обмениваться данными с удаленным терминалом. При продолжительном отсутствии входных данных от модема сервис самостоятельно принимает решение о тестировании соединения. На время теста состояние сервиса

меняется на HM_TEST_CONNECTION. Ошибка при проверке соединения приведет к переходу в состояние HM_READY.

Для прекращения (или разрыва) соединения и выхода из режима передачи данных необходимо вызвать функцию FwModem_disconnect. При этом сервис переходит из состояния HM_DATA_EXCHANGE в состояние HM_READY через HM_DISCONNECTING.

Отключение сервиса модема и освобождение занятого им коммуникационного порта осуществляется функцией FwModem_close. Отключить сервис можно и во время передачи/приема данных без завершения соединения, однако после этого коммуникационный порт будет занят приблизительно 5 секунд, в течение которых сервис будет самостоятельно корректно закрывать соединение и коммуникационный порт.

6.9.2.2. Состояния сервиса модема

Состояние сервиса модема в любой момент можно запросить функцией FwModem_getStatus, передав ей идентификатор в качестве параметра. Ниже перечислены возможные состояния и действия в каждом из них.

HM_CLOSED:

Состояние HM_CLOSED означает, что сервис модема с запрошенным идентификатором закрыт и не работает. Для начала работы с модемом необходимо вызвать функцию FwModem_open, которая вернет идентификатор сервиса в случае его успешного запуска.

HM_START:

Состояние HM_START показывает, что сервис начинает работу. Контролировать состояние сервиса можно в любое время, передав его идентификатор в качестве параметра функции FwModem_getStatus.

HM_INITIALIZATION:

Состояние HM_INITIALIZATION показывает, что сервис занимается запуском подключенного к коммуникационному порту модема с помощью строки инициализации. Если инициализация завершается успехом, то сервис перейдет в состояние готовности к работе HM_READY.

HM_READY:

Состояние HM_READY показывает, что подключенный к коммуникационному порту модем обнаружен и запущен с помощью строки инициализации.

Если строка установления соединения уже была передана сервису при вызове функции FwModem_connect, то из состояния HM_READY производится немедленный переход в HM_CONNECTING для набора номера или ожидания звонка.

Если строка установления соединения не была передана сервису, то он ожидает дальнейших команд от приложения CoDeSys, периодически производя контроль работоспособности модема. Если по какой-либо причине (например, при аварии питания у модема) подключенный модем не отвечает на контрольный запрос, сервис переходит в состояние HM_INITIALIZATION и пытается провести процедуру инициализации подключенного модема повторно.

Находящийся в состоянии HM_READY сервис может быть закрыт командой FwModem_close или переведен в состояние установления соединения HM_CONNECTING функцией FwModem_connect. При вызове функции FwModem_connect сервису вместе с командой “установить соединение” передается строка сценария дозвона, которая сохраняется им в памяти.

HM_CONNECTING:

Состояние HM_CONNECTING показывает, что сервис выполняет сценарий установления соединения (дозвон или ожидание входящего звонка) с удаленным терминалом. В случае успеха будет произведен переход в состояние HM_DATA_EXCHANGE, при ошибке дозвона сервис повторяет сценарии инициализации модема и установления соединения, проходя состояния HM_INITIALIZATION, HM_READY, HM_CONNECTING. Вызывать функции FwModem_connect повторно не требуется - сервис модема запоминает сценарии инициализации и установления соединения и самостоятельно пытается дозвониться до удаленного терминала. Прекратить автодозвон не закрывая сервис модема функцией FwModem_close можно, вызвав функцию FwModem_disconnect.

HM_DATA_EXCHANGE:

Успешно дозвонившись или дождавшись вызова удаленного терминала, сервис переходит в режим обмена данными HM_DATA_EXCHANGE. С помощью функций FwModem_write, FwModem_read приложение CoDeSys имеет возможность обмениваться данными с удаленным модемом. Обратите внимание, что функции обмена данными возвращают реальное число записанных или считанных байт, которое может отличаться от требуемого количества. Передача или прием большого объема данных может потребовать многократного вызова этих функций. Следует также учитывать пропускную способность модемного соединения:

- скорость обмена по телефонной линии в режиме факс/данные не превышает 38400 бит/с;
- скорость обмена в режиме факс/данные (CSD) для GSM-модемов не превышает 9600 бит/с.

При долгом отсутствии входящих данных сервис может производить кратковременные проверки соединения, переходя в состояние HM_TEST_CONNECTION. При разрыве соединения сервис вновь приступает к выполнению сценариев инициализации и дозвона. Отслеживать состояние соединения сервиса можно, используя функцию FwModem_getStatus.

После окончания обмена данными установленное соединение может быть закрыто по инициативе приложения CoDeSys с помощью вызова функции FwModem_disconnect. При этом сервис удаляет хранящийся у него сценарий установления соединения (“дозвона”), меняет свое состояние на HM_DISCONNECTING и начинает выполнять специальный сценарий завершения соединения (“кладет телефонную трубку”). После завершения процедуры закрытия соединения сервис возвращается в состояние HM_READY и ожидает новых команд.

Для полного завершения работы сервиса модема и освобождения занятого им коммуникационного порта можно вызвать функцию FwModem_close(). Вызов функции FwModem_close() для модема, находящегося в режиме передачи или приема данных, также завершится успехом, но при этом сервис модемной библиотеки некоторое время (6-7 секунд) будет самостоятельно корректно завершать установленное соединение. В течение указанного выше времени коммуникационный порт будет недоступен для открытия.

После загрузки в контроллер нового приложения из CoDeSys все открытые сервисы модемов будут закрыты, если параметр *Fastwel I/O System Configuration: HotUpdateDisabled* установлен в *TRUE*. Принудительное закрытие сервиса модема, находящегося в режиме факс/данные, включает в себя выполнение сценария завершения соединения (модем “кладет трубку”). Сразу после старта нового загруженного приложения CoDeSys коммуникационный порт модема может быть занят на время выполнения сценария (6-7 секунд).

Если параметр *Fastwel I/O System Configuration: HotUpdateDisabled* установлен в *FALSE* и в процессе обновления приложения CoDeSys оказалось, что структуры данных, размеры образа процесса и связи задач с образом процесса не изменились, то автоматического закрытия сервисов модемов не произойдет. Попытка открыть на старте приложения сервис модема через уже используемый коммуникационный порт будет неудачной. Поэтому в режиме работы с “горячим обновлением приложения” наиболее правильным является вызов функции закрытия сервисов модема во время обработки события *OnProgramChange*.

6.9.2.3. Сценарии инициализации и установления соединения

Функции FwModem_open и FwModem_connect позволяют передавать сервису модема управляющие строки (сценарии) инициализации и установления соединения соответственно. Каждая строка состоит из набора выражений, разделенных пробелами, которые определяют набор команд, ответов и интервалов ожидания при взаимодействии с модемом. Для выражений используется следующий синтаксис:

Выражение	Описание
AT*cmdsymbols response*	Данное выражение определяет команду для модема и строку ответа от модема при успешном выполнении команды. В приведенном ниже примере модему передается команда *ATA* и ожидается ответ CONNECT. Пример: ***ATD+79161234567 CONNECT***
TIMEOUT *time*	Определяет интервал времени, в течение которого ожидается ответ модема на любую команду в текущем сценарии взаимодействия с модемом. Параметр *time* определяет длительность интервала в диапазоне от 1 до 999 секунд. Допускается наличие нескольких команд *TIMEOUT* в одной строке, что позволяет устанавливать разные интервалы времени для разных команд. В приведенном ниже примере ответ модема *OK* будет ожидаться в течение 3 секунд после команды *AT*, ожидание ответа *CONNECT* будет длиться в течение 60 секунд после отправки команды *ATD+79161234567*. Пример: ***TIMEOUT** 3 AT OK **TIMEOUT** 60 ATD+79161234567 CONNECT*
ABORT "*response***"**	Определяет строку от модема, получение которой интерпретируется сервисом как ошибка сценария. Строка ответа должна быть заключена в кавычки. Каждая секция сценария взаимодействия с модемом может содержать до пяти определений типа ABORT. В приведенном ниже примере сценарий будет прерван с результатом "ошибка" либо при получении от модема строк *NO CARRIER* или *BUSY*, либо при отсутствии ответа *CONNECT* на команду ATA в течение 60 секунд. Сценарий завершится успешно, если модем ответит *CONNECT* на команду *ATA*. Пример: *TIMEOUT 60 **ABORT** "NO CARRIER" **ABORT** "BUSY" ATD+79161234567 CONNECT*
PAUSE *time*	Определяет интервал времени, в течение которого сервис выдерживает паузу и не отсылает каких-либо команд модему. Параметр *time* определяет длительность интервала в диапазоне от 1 до 999 секунд. В приведенном ниже примере пауза длительностью 1 секунда будет выдержана между ответом *OK* на команду *AT* и командой *ATD+79161234567*, а после получения ответа *CONNECT* перед окончанием сценария будет выдержана пауза 2 секунды. Сценарий завершится ошибкой, если модем выдаст *"NO CARRIER"*, *"BUSY"* или *"NO DIALTONE"* в ответ на какую-либо команду или не будет получено ответов *OK*, *CONNECT* в течение 30 секунд после команд модему. Пример: *TIMEOUT 60 ABORT "NO CARRIER" ABORT "BUSY" ABORT "NO DIALTONE" AT OK **PAUSE** 1 ATD+79161234567 CONNECT **PAUSE** 2*
Response	Любая последовательность символов, не начинающаяся с AT, не располагающаяся после команды и не являющаяся служебным словом TIMEOUT, ABORT, PAUSE показывает сервису, что указанную последовательность необходимо ожидать в виде запроса от модема. В приведенном ниже примере сервис будет ожидать строки *RING* от модема в течение 60 секунд. Если запрос *RING* от модема будет получен, то сервис отправит команду ATA Пример: *TIMEOUT 60 ABORT "NO CARRIER" ABORT "BUSY" ABORT "NO DIALTONE" **RING** ATA CONNECT*

6.9.1. Управляющие последовательности модемов для различных типов линий

6.9.1.1. Общие сведения

С помощью библиотеки FastwelHayesModem.lib можно устанавливать модемные соединения через выделенные и коммутируемые линии. Модем может являться как инициатором соединения, так и отвечать на удаленный вызов, ожидая звонок. Ниже приведены сценарии действий для модемов Zyxel, Courier, Cinterion MC52, работающих в различных режимах.

6.9.1.2. Выделенная линия

Примеры строк инициализации для модема, осуществляющего вызов:

```
Zyxel: 'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=60 OK AT&K0 OK'
Courier: 'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=60 OK AT&I0 OK AT&R1 OK'
```

Примеры строк установления соединения для модема, осуществляющего вызов:

```
Zyxel/Courier: 'TIMEOUT 60 ABORT "NO CARRIER" ATX3D CONNECT PAUSE 1'
```

Примеры строк инициализации для модема, ожидающего вызов:

```
Zyxel: 'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=20 OK AT&K0 OK'
Courier: 'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=20 OK AT&I0 OK AT&R1 OK'
```

Примеры строк установления соединения для модема, ожидающего вызов:

```
Zyxel/Courier: 'TIMEOUT 20 ABORT "NO CARRIER" ATA CONNECT PAUSE 1'
```

6.9.1.3. Коммутируемая линия

Пример строки инициализации для GSM-модема Cinterion MC52, осуществляющего или ожидающего вызов:

```
'ABORT "ERROR" TIMEOUT 3 AT+CREG=0 OK ATX4 OK AT+CREG? +CREG: 0,1 OK PAUSE 1'
```

Пример строки установления соединения для GSM-модема Cinterion MC52, осуществляющего вызов:

```
'TIMEOUT 60 ABORT "NO CARRIER" ABORT "BUSY" ABORT "NO DIALTONE" ATD+79261234567 CONNECT PAUSE 1'
```

Пример строки установления соединения для GSM-модема Cinterion MC52, ожидающего вызов:

```
'TIMEOUT 60 ABORT "NO CARRIER" ABORT "BUSY" ABORT "NO DIALTONE" RING ATA CONNECT PAUSE 1'
```

6.9.2. Описание функций

6.9.2.1. FwModem_open

Функция открывает (запускает) модем через указываемый во входных параметрах функции порт.

```
FUNCTION FwModem_open : DWORD
VAR_INPUT
   iPort: INT;
   dwBaudRate: DWORD;
   StopBits: FW_HAYES_STOPBITS;
   Parity: FW_HAYES_PARITY;
   pInitStr: POINTER TO STRING(254);
END_VAR
```

Входные параметры:

```
iPort: INT
```

Идентификатор порта типа INT. Допустимые значения: 1, 101-164 – для портов, организованных посредством элемента *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module.* Если в проекте для CPM712 для элемента конфигурации *CPM712 MODBUS RTU/ASCII Programmable Controller-Serial Port* установлен режим *Not Used*, то возможно использовать значение 2.

```
dwBaudRate: DWORD
```

Скорость, на которой коммуникационный порт должен работать с модемом. Допустимо значение из набора: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

```
StopBits: FW_HAYES_STOPBITS
```

Количество используемых коммуникационным портом стоповых бит. Принимает значение из набора (ONESTOPBIT :=0, ONE5STOPBITS :=1, TWOSTOPBITS := 2).

```
Parity: FW_HAYES_PARITY
```

Количество используемых коммуникационным портом бит четности. Принимает значение из набора (NOPARITY:=0, ODDPARITY:=1, EVENPARITY:=2).

```
pInitStr: POINTER TO STRING(254)
```

Указатель на строку инициализации модема. Содержащиеся в строке AT-команды будут выполнены при запуске модема.

Возвращаемый результат:

Системный идентификатор порта, который в дальнейшем используется в вызовах других функций библиотеки. В случае ошибки (например, если коммуникационный порт занят) возвращается значение: 16#FFFFFFFF.

6.9.2.2. FwModem_close

Функция закрывает модемное соединение и освобождает коммуникационный порт.

```
FUNCTION FwModem_close : DWORD
VAR_INPUT
   dwHandle: DWORD;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема.

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успеха.

6.9.2.3. FwModem_connect

Функция запуска процедуры соединения с удаленным модемом.

```
FUNCTION FwModem_connect : DWORD
VAR_INPUT
   dwHandle: DWORD;
   pDialStr: POINTER TO STRING;
   dwLineControlPeriod: DWORD;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема.

```
pDialStr: POINTER TO STRING
```

Указатель на строку, содержащую последовательность AT-команд для установления соединения (дозвона).

```
dwLineControlPeriod: DWORD
```

Интервал времени в миллисекундах, используемый для контроля работоспособности соединения. Если в течение указанного времени модемом не было получено ни одного байта от удаленного терминала, то будет произведен тест соединения. Передача значения 0 в качестве параметра dwLineControlPeriod отключает периодический тест соединения.

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успешного установления соединения.

6.9.2.4. FwModem_disconnect

Функция для закрытия установленного соединения.

```
FUNCTION FwModem_disconnect : DWORD
VAR_INPUT
   dwHandle: DWORD;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема.

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успешного завершения соединения.

6.9.2.5. FwModem_getStatus

Функция чтения текущего состояния модема.

```
FUNCTION FwModem_getStatus : DWORD
VAR_INPUT
   dwHandle: DWORD;
END_VAR
```

Входные параметры:

dwHandle: DWORD

Идентификатор, полученный при открытии модема.

Возвращаемый результат:

Состояние сервиса модема, описанное подробно в п. 6.9.2.1.

6.9.2.6. FwModem_read

Пытается прочитать wDataLength байт из приемного буфера модема.

```
FUNCTION FwModem_read : WORD
VAR_INPUT
   dwHandle: DWORD;
   pData: POINTER TO BYTE;
   wDataLength: WORD;
END_VAR
```

Входные параметры:

dwHandle: DWORD

Идентификатор, полученный при открытии модема.

pData: POINTER TO BYTE

Указатель на буфер, в который требуется выполнить чтение.

wDataLength: WORD

Количество байт, которое требуется прочитать.

Возвращаемый результат:

Функция возвращает фактическое количество считанных байт.

6.9.2.7. FwModem_write

Пытается записать wDataLength байт в модем для передачи удаленному терминалу.

```
FUNCTION FwModem_write : WORD
VAR_INPUT
   dwHandle: DWORD;
   pData: POINTER TO BYTE;
   wDataLength: WORD;
END_VAR
```

Входные параметры:

dwHandle: DWORD

Идентификатор, полученный при открытии модема.

pData: POINTER TO BYTE

Указатель на буфер, из которого требуется выполнить запись.

wDataLength: WORD

Количество байт, которое требуется записать.

Возвращаемый результат:

Функция возвращает количество записанных байт.

6.10. Библиотека FastwelHayesModemControls.lib

6.10.1. Назначение

Библиотека FastwelHayesModemControl.lib написана на языке Structured Text в среде CoDeSys 2.3 и предназначена для обмена данными через модем. В состав библиотеки входит функциональный блок FW_HAYES_MODEM, который необходимо периодически вызывать в приложении CoDeSys. Интерфейс функционального блока FW_HAYES_MODEM приведен ниже:

```
VAR_INPUT
   blSwitchModemOn:          BOOL;
   blOpenConnection:         BOOL;

   strModemInit:             STRING(254) := '';
   strModemDial:             STRING(254) := '';

   byComPortNmb:             BYTE;
   dwBaudrate:               DWORD := 115200;
   byStopbits:               FW_HAYES_STOPBITS   := ONESTOPBIT;
   byParity:                 FW_HAYES_PARITY     := NOPARITY;

   blStartDataTransfer:      BOOL;
   pSendBuffer:              POINTER TO BYTE;
   bytesToSend:              WORD;

   blStartDataReception:     BOOL;
   pReceiveBuffer:           POINTER TO BYTE;
   bytesToReceive:           WORD;
   dwLineControlPeriod:      DWORD;
END_VAR
```

Перевод входа *blSwitchModemOn* в состояние *TRUE* и удержание его в этом состоянии позволяет запустить сервис модема, подключенного к коммуникационному порту *byComPortNmb*. Для запуска сервиса необходимо передать строку инициализации *strModemInit* и настройки *dwBaudrate*, *byStopbits*, *byParity*. Перевод входа в состояние *FALSE* закроет сервис модема, подключенного к коммуникационному порту *byComPortNmb*. После успешного запуска сервиса и инициализации модема значение выхода *blModemReady* переходит в состояние *TRUE*.

Перевод входа *blOpenConnection* в состояние *TRUE* позволяет запустить процедуру установления соединения, используя строку *strModemDial* в качестве сценария. Для завершения рабочего соединения или прекращения автодозвона достаточно перевести вход *blOpenConnection* в состояние *FALSE*. После успешной установки соединения с удаленным терминалом значение выхода *blConnectionReady* блока станет равным *TRUE*.

Вход *strModemInit* содержит строку сценария инициализации модема.

Вход *strModemDial* содержит строку сценария установления соединения модема с удаленным терминалом.

Вход *byComPortNmb* содержит номер коммуникационного порта, к которому подключен модем.

Вход *dwBaudrate* определяет скорость, на которой коммуникационный порт должен работать с модемом. Допустимо значение из набора: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

Вход *byStopbits* определяет количество используемых коммуникационным портом стоповых бит. Принимает значение из набора (ONESTOPBIT :=0, ONE5STOPBITS :=1, TWOSTOPBITS := 2).

Вход *byParity* определяет количество используемых коммуникационным портом бит четности. Принимает значение из набора (NOPARITY:=0, ODDPARITY:=1, EVENPARITY:=2).

Перевод входа *blStartDataTransfer* в состояние *TRUE* позволяет сервису в режиме HM_DATA_EXCHANGE начать передачу по модему удаленному терминалу *bytesToSend* байт из буфера *pSendBuffer*. После передачи *bytesToSend* байт выход *blIsEndOfDataTransfer* перейдет в состояние *TRUE*.

Перевод входа *blStartDataReception* в состояние *TRUE* позволяет сервису в режиме HM_DATA_EXCHANGE начать прием по модему от удаленного терминала *bytesToReceive* байт в буфер *pReceiveBuffer*. После приема bytesToReceive байт выход *blIsEndOfDataReception* перейдет в состояние *TRUE*.

Вход *dwLineControlPeriod* определяет интервал времени в миллисекундах, определяющий период контроля линии связи при длительном отсутствии входных данных. Значение с данного входа считывается при старте процедуры установления соединения.

```
VAR_OUTPUT
   blModemReady:              BOOL;
   blConnectionReady:         BOOL;
   byModemError:              MODEM_ERRORS;
   blIsEndOfDataTransfer:     BOOL;
   blIsEndOfDataReception:    BOOL;
   eModemStatus:              DWORD;
END_VAR
```

Выход *blModemReady*, равный *TRUE*, показывает, что сервис модема успешно запустил модем и выполнил его инициализацию. Состояние выхода в состоянии *FALSE* показывает, что модем либо не подключен, либо пока не инициализирован.

Выход *blConnectionReady*, равный *TRUE*, показывает, что соединение с удаленным терминалом успешно установлено.

Выход *byModemError* индицирует результат последней проведенной сервисом операции. В случае нормальной работы блока принимает нулевое значение.

Выход *blIsEndOfDataTransfer*, равный *TRUE*, сигнализирует об окончании передачи данных удаленному терминалу.

Выход *blIsEndOfDataReception*, равный *TRUE*, сигнализирует об окончании приема требуемого количества байт от удаленного терминала.

Выход *eModemStatus* показывает текущее состояние модема. Принимает значение из набора: HM_CLOSED, HM_START, HM_INITIALIZATION, HM_READY, HM_CONNECTING, HM_DATA_EXCHANGE, HM_TEST_CONNECTION, HM_DISCONNECTING.

6.10.2. Применение функционального блока FW_HAYES_MODEM

6.10.2.1. Общие сведения

Экземпляром функционального блока называется переменная, тип которой совпадает с типом функционального блока. Экземпляр функционального блока должен быть описан в секции деклараций как обычная переменная.и

6.10.2.2. Начало работы

Перед началом работы с модемом необходимо объявить экземпляр функционального блока FW_HAYES_MODEM.

6.10.2.3. Запуск модема

Для запуска модема достаточно установить вход *blSwitchModemOn* в *TRUE*, *blOpenConnection* в *FALSE*, ввести сценарий инициализации модема в *strModemInit*, установить номер коммуникационного порта *byComPortNmb*, скорость обмена *dwBaudrate*, количество стоповых бит *byStopbits*, четность *byParity*, остальные входы обратить в 0 или *FALSE*, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. Сценарий инициализации модема для модема Zyxel может иметь следующий вид:

```
'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=60 OK AT&K0 OK'
```

После каждого вызова необходимо проверять состояние выхода *blModemReady* – его переход в состояние *TRUE* сигнализирует об обнаружении модема, его готовности к дозвону или ответу на вызов.

6.10.2.4. Установление соединения

Для установления соединения с удаленным терминалом достаточно установить вход *blSwitchModemOn* в *TRUE*, *blOpenConnection* в *TRUE*, ввести в качестве сценария инициализации пустую строку, ввести сценарий дозвона в *strModemDial*, остальные входы обратить в 0 или *FALSE*, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. Сценарий дозвона для модема Zyxel, работающего через выделенную линию, может иметь вид:

```
'TIMEOUT 60 ABORT "NO CARRIER" ATX3D CONNECT PAUSE 1'
```

После каждого вызова экземпляра функционального блока необходимо проверять состояние выхода *blOpenConnection* – его переход в состояние *TRUE* сигнализирует о переходе в режим факс/данные и возможности обмениваться данными с удаленным терминалом.

6.10.2.5. Режим факс/данные

Передача данных:

Для начала передачи данных удаленному терминалу достаточно установить вход *blSwitchModemOn* в *TRUE, blOpenConnection* в *TRUE, blStartDataTransfer* в *TRUE*, ввести в качестве сценариев инициализации и дозвона пустые строки, адрес буфера отправки записать в *pSendBuffer*, количество байт для отправки записать в *bytesToSend*, остальные входы обратить в 0 или *FALSE*, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. После каждого вызова необходимо проверять значение выхода *blIsEndOfDataTransfer* – его переход в состояние *TRUE* сигнализирует об окончании передачи данных.

Прием данных:

Для начала приема данных от удаленного терминала достаточно установить вход *blSwitchModemOn* в *TRUE, blOpenConnection* в *TRUE, blStartDataReception* в *TRUE*, ввести в качестве сценариев инициализации и дозвона пустые строки, адрес буфера приема записать в *pReceiveBuffer*, количество байт для приема записать в *bytesToReceive*, остальные входы обратить в 0 или *FALSE*, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. После каждого вызова необходимо проверять значение выхода *blIsEndOfDataReception* – его переход в состояние *TRUE* сигнализирует об окончании приема данных.

Допускаются одновременные прием и передача данных через модем. Для этого необходимо установить входы, определяющие параметры передачи и приема, в желаемое состояние, а при вызове экземпляра блока - анализировать и выход *blIsEndOfDataTransfer*, и выход *blIsEndOfDataReception*.

6.10.2.6. Завершение соединения

Для завершения соединения достаточно установить вход *blSwitchModemOn* в *TRUE*, *blOpenConnection* в *FALSE*, ввести в качестве сценариев инициализации и дозвона пустые строки, остальные входы обратить в 0 или *FALSE*, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. После каждого вызова экземпляра функционального блока необходимо проверять состояние выхода *blOpenConnection* – его переход в состояние *FALSE* сигнализирует о закрытии соединения с удаленным терминалом.

6.10.2.7. Закрытие сервиса модема

Для закрытия сервиса модема достаточно установить вход *blSwitchModemOn* в *FALSE*, ввести в качестве сценариев инициализации и дозвона пустые строки, остальные входы обратить в 0 или *FALSE*, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. После каждого вызова экземпляра функционального блока необходимо проверять состояние выхода *blModemReady* – его переход в состояние *FALSE* сигнализирует о закрытии сервиса модема и коммуникационного порта.

6.11. Библиотека FastwelSms.lib

6.11.1. Назначение

Библиотека FastwelSms.lib предназначена для приема и отправки SMS-сообщений и содержит функции, необходимые для управления GSM-модемами с помощью AT-команд.

В контроллерах CPM711/CPM712/CPM713 комплекса Fastwel I/O данная библиотека может быть использована для рассылки сообщений и приема управляющих SMS. Функции библиотеки FastwelSms.lib позволяют контролировать состояние памяти SIM-карты GSM-модема, количество денег на счету, определять факт доставки сообщений до абонентов.

GSM-модем (например, Cinterion MC52) может быть подключен к:

- коммуникационному порту, организованному на основе модулей NIM741/NIM742 и элементов *NIM741 RS-485 1xUART Stream Module* и *NIM742 RS-232 1xUART Stream Module*.

- коммуникационному порту, расположенному на передней панели CPM711/712/713 под пластиковой крышкой;

- коммуникационному порту COM2 интерфейса RS-485 на контроллере CPM712 (для этого в проекте для CPM712 для элемента конфигурации *CPM712 MODBUS RTU/ASCII Programmable Controller-Serial Port* необходимо установить режим *Not Used*).

Примечание. 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.

Пример использования библиотеки имеется в проекте fw_hayes_sms_test.pro (для контроллера CPM713), который входит в пакет адаптации CoDeSys для Fastwel I/O (расположен в подкаталоге Examples\CPM71x\CPM713\Hayes_Modem каталога установки адаптации CoDeSys 2.3 для Fastwel I/O).

Общее количество GSM-модемов, которыми можно управлять с помощью библиотеки FastwelSms.lib, ограничено 16.

6.11.2. Принцип работы

6.11.2.1. Общие сведения

Для того чтобы начать отправлять и принимать SMS-сообщения, необходимо открыть сервис GSM-модема, вызвав функцию FwModem_open библиотеки FastwelHayesModem.lib (см п.6.9). В качестве одного из параметров ей необходимо передать строку инициализации (сценарий запуска модема) вида

```
'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=20 OK AT+CUSD=1 OK AT+CNMI=2,1,2,2,1 OK
AT+CREG=0 OK ATX4 OK AT+CREG? +CREG: 0,1 OK PAUSE 1'
```

(синтаксис выражений в сценариях модемов подробно описан в п.6.9.2.2.).

Возвращенный функцией идентификатор должен впоследствии использоваться при вызове других функций библиотек FastwelHayesModem.lib и FastwelSms.lib.

Открытый сервис модема последовательно проходит через состояния HM_START, HM_INITIALIZATION в HM_READY. Для проверки текущего состояния сервиса служит функция FwModem_getStatus библиотеки FastwelHayesModem.lib. Состояние сервиса HM_READY после выполнения приведенного выше сценария показывает, что модем включен, зарегистрирован в сотовой сети и готов к работе.

В состоянии HM_READY возможны следующие действия:

- чтение и передача сообщений с помощью функций FwModem_readSms, FwModem_sendSms;
- запрос состояния одного из отправленных сообщений функцией FwModem_getSmsStatus;
- передача USSD-запросов и чтение ответов на них с помощью FwModem_writeUssd, FwModem_readUssd;
- проверка состояния встроенного буфера сообщений модема (памяти модема) функцией FwModem_getMemInfo.

USSD-запросы удобно использовать для определения количества средств на счету SIM-карты модема.

В состоянии HM_READY сервис периодически проверяет работоспособность модема и объем его свободной памяти. В том случае, если модем по каким-либо причинам (например, из-за аварии питания) перестал отвечать, сервис возвращается в состояние HM_INITIALIZATION, в котором пытается осуществить перезапуск модема и выполнить сценарий инициализации.

Для полного завершения работы сервиса модема и освобождения занятого им коммуникационного порта необходимо вызвать функцию FwModem_close.

После загрузки в контроллер нового приложения из CoDeSys все открытые сервисы модемов будут закрыты, если параметр *Fastwel I/O System Configuration: HotUpdateDisabled* установлен в *TRUE*.

Если параметр *Fastwel I/O System Configuration: HotUpdateDisabled* установлен в *FALSE* и в процессе обновления приложения CoDeSys оказалось, что структуры данных, размеры образа процесса

и связи задач с образом процесса не изменились, то автоматического закрытия сервисов модемов не произойдет. Попытка открыть на старте приложения сервис модема через уже используемый коммуникационный порт будет неудачной. Поэтому в режиме работы с “горячим обновлением приложения” наиболее правильным является вызов функции закрытия сервисов модема во время обработки события *OnProgramChange*.

Для отправки сообщений в памяти установленной в модем SIM-карты должен быть записан номер SMS-центра (SMSC). В настоящее время большинство операторов, выпуская SIM-карту, записывают в нее правильный номер SMS-центра. Если по какой-то причине он отсутствует в памяти SIM-карты, то его необходимо записать в нее либо вручную с помощью любого мобильного телефона, либо добавив в сценарий инициализации запись вида

```
'TIMEOUT 5 ABORT "ERROR" AT+CSCA="+79161234567" OK'
```

6.11.2.2. Отправка SMS-сообщений

Отправка SMS-сообщений возможна в режиме работы HM_READY сервиса модема. Для отправки сообщения необходимо вызвать функцию FwModem_sendSms, передав в качестве параметров идентификатор модема, телефонный номер абонента в международном формате, текст сообщения на русском или английском языке, время жизни сообщения в сотовой сети и признак требования подтверждения доставки.

Номер абонента должен быть передан в виде строки формата

```
'+79161234567'
```

Длина текста сообщения на русском языке не должна превышать 70 символов. Сообщение на английском языке не должно быть длиннее 160 символов. При передаче функции FwModem_sendSms недопустимо длинного сообщения абонент получит укороченное SMS.

Время жизни сообщения в сотовой сети задается в относительном формате по стандарту 3GPP TS 23.040 и может принимать значения от 5 минут до 63 недель.

Сервис модема позволяет контролировать состояние отправленного сообщения в сети GSM и определять дату, время и факт доставки сообщения абонентам. Для этого сервис принимает и анализирует полученные от сотовой сети SMS-уведомления об отправленных сообщениях. Одновременно сервис способен контролировать состояния 10-ти последних отправленных сообщений с требованием подтверждения.

Функция FwModem_sendSms возвращает отрицательное значение в том случае, если сообщение отправить не удалось. При отправке с требованием подтверждения возвращаемый функцией положительный идентификатор от 0 до 255 может быть использован для проверки состояния отправленного сообщения. При отправке сообщения без требования подтверждения функция возвратит значение SNDSMS_REPORT_NOT_REQUIRED.

6.11.2.3. Запрос состояния SMS-сообщений

Для контроля состояния отправленного в сеть SMS-сообщения необходимо при отправке установить параметр reportRequired функции FwModem_sendSms в *TRUE* и запомнить возвращаемый функцией идентификатор. После попытки отправки сообщения его статус можно запросить функцией FwModem_getSmsStatus, передав ей полученный идентификатор в качестве параметра.

6.11.2.4. Чтение SMS-сообщений

На протяжении всей работы сервис периодически (раз в 15 секунд) опрашивает память сообщений GSM-модема и определяет типы сообщений в памяти. Сохраненные отправленные, длинные сообщения, а также SMS-уведомления удаляются из памяти сразу. При наличии в памяти модема нескольких входящих сообщений содержание наиболее раннего из них будет считано в память сервиса и подготовлено для выдачи приложению CoDeSys. Обработанные таким образом входящие сообщения можно считывать по одному функцией FwModem_readSms. Входящее сообщение удаляется из памяти модема только после его передачи приложению.

6.11.2.5. Запрос баланса

Для определения состояния счета SIM-карты GSM-модема в библиотеке FastwelSms.lib предусмотрены функции для обработки USSD-запросов. Для запроса баланса достаточно вызвать

функцию FwModem_sendUssd, передав ей в качестве параметра команду запроса баланса в виде строки, например

```
'*100#'
```

Прочитать ответ на USSD-запрос можно с помощью функции FwModem_readUssd, которая возвращает текст сообщения, пришедшего в ответ.

Обработка оператором сотовой связи USSD-запроса и ответ на него занимает достаточно долгое время и может составить от 3 до 30 секунд. В течение указанного времени после отправки запроса сервис модема ожидает ответа и не готов отсылать другие Ussd-запросы. Во время ожидания ответа на USSD-запрос опрос памяти модема сервис также не производит.

6.11.2.6. Контроль состояния памяти модема

Во время работы память сообщений модема может постепенно заполняться, т.к. сервис самостоятельно не удаляет входящие, не прочитанные приложением CoDeSys сообщения. Если в памяти модема не остается места совсем, то отправить сообщение и получить подтверждение становится невозможно.

Состояние памяти сообщений может быть запрошено с помощью функции FwModem_getMemInfo. Функция возвратит положительное значение в случае, если память модема заполнена сообщениями более чем на 75%. Освободить память модема можно с помощью функции FwModem_readSms, прочитав по одному все входящие сообщения. После передачи в приложение CoDeSys очередного сообщения сервис удаляет его из памяти модема.

6.11.3. Описание функций

6.11.3.1. FwModem_sendSms

Функция для отправки SMS-сообщения абоненту.

```
FUNCTION FwModem_sendSms : INT
VAR_INPUT
   dwHandle: DWORD;
   deliveryNumber: STRING(12);
   text: STRING(160);
   byValidityPeriod: BYTE;
   blReportRequired: BOOL;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема функцией FwModem_open.

```
deliveryNumber: STRING(12)
```

Строка, содержащая телефонный номер абонента – получателя SMS-сообщения. Номер абонента должен быть записан в формате ‘+79161234567’.

```
text: STRING(160)
```

Текст исходящего сообщения. Длина сообщения, содержащего буквы русского алфавита, не должна превышать 70 символов. Длина сообщения, содержащего только буквы латинского алфавита, не должна превышать 160 символов. При попытке отправить слишком длинное сообщение абонент получит “обрезанный” текст.

```
byValidityPeriod: BYTE
```

Время жизни сообщения в относительном формате (по стандарту 3GPP TS 23.040):

0 .. 143 соответствует (*byValidityPeriod* + 1) * 5 минут – максимально можно задать 12 часов с шагом 5 минут;

144 .. 167 соответствует 12 часов + (*byValidityPeriod* - 143) * 30 минут – максимально можно задать 24 часа;

168 .. 196 соответствует (*byValidityPeriod* - 166) * 1 день = максимально можно задать 30 дней;

197 .. 255 соответствует (*byValidityPeriod* - 192) * 1 неделю = максимально можно задать 63 недели.

```
blReportRequired: BOOL
```

Признак требования подтверждения о доставке сообщения. Если значение параметра *reportRequired* равно *TRUE*, то состояние сообщения будет отслеживаться сервисом модема после его отправки. Одновременно может контролироваться состояние 10-ти отправленных сообщений.

Возвращаемый результат:

–1: в случае ошибки идентификатора модема;

SNDSMS_MODEM_IS_BUSY: в случае неготовности модема к отправке;

SNDSMS_WRONG_DELIVERY_NMB: если телефонный номер абонента записан в неправильном формате;

идентификатор отправленного сообщения (0 .. 255): в случае успешной передачи модему сообщения на отправку, если параметр *reportRequired* был установлен в *TRUE*;

SNDSMS_REPORT_NOT_REQUIRED: в случае успешной передачи модему сообщения на отправку, если параметр *reportRequired* был установлен в *FALSE*.

6.11.3.2. FwModem_readSms

Функция для чтения текста входящего SMS-сообщения.

```
FUNCTION FwModem_readSms : DWORD
VAR_INPUT
   dwHandle: DWORD;
   pSenderNmbStr: POINTER TO STRING(12);
   pTextStr: POINTER TO STRING(160);
   pDepartureDateTime: POINTER TO DATE_AND_TIME;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема.

```
pSenderNmbStr: POINTER TO STRING(12)
```

Адрес строки, в которую при получении нового входящего сообщения будет записан номер абонента – отправителя. Номер отправителя будет представлен в виде ‘+79161234567’.

```
pTextStr: POINTER TO STRING(160)
```

Адрес строки, в которую при получении нового входящего сообщения будет записан его текст. Максимально допустимая длина сообщений с русскими буквами – 70 символов, только с латинскими – 160.

```
pDepartureDateTime:    POINTER TO DATE_AND_TIME;
```

Адрес памяти, в которую при получении нового входящего сообщения будут записаны его дата и время отправки.

Возвращаемый результат:

–1: в случае ошибки идентификатора модема;

0: если новых сообщений не было получено;

длина сообщения: если новое входящее сообщение было успешно прочитано.

6.11.3.3. FwModem_getSmsStatus

Функция для чтения состояния отправленного сообщения.

```
FUNCTION FwModem_connect : DWORD
VAR_INPUT
   dwHandle: DWORD;
   iMsgId: INT;
   pArrivalTime: POINTER TO DATE_AND_TIME;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема.

iMsgId: INT

Идентификатор, полученный при отправке сообщения.

pArrivalTime: POINTER TO DATE_AND_TIME

Адрес области памяти, в которое будет записано время получения сообщения адресатом.

Возвращаемый результат:

–1: в случае ошибки идентификатора модема;

FW_SMS_STATUS_MODEM_MEM_OVERFLOW: если отправка сообщения невозможна из-за отсутствия у модема свободных ячеек в памяти сообщений;

FW_SMS_STATUS_WRONG_REPORT_ID: в случае ошибки идентификатора сообщения;

FW_SMS_STATUS_REJECTED_BY_MODEM: если GSM-модему не удалось отправить сообщение SMS-центру (например, из-за отсутствия средств на счету SIM-карты);

FW_SMS_STATUS_NO_REPORT: если для отправленного сообщения ни одного SMS-уведомления пока не было получено;

значение в диапазоне от 0 до 31: показывает, что сообщение было отправлено (поле TP-Status согласно 3GPP TS 23.040);

значение в диапазоне от 32 до 63: показывает, что SMS-центр все еще пытается доставить сообщение абоненту (поле TP-Status согласно 3GPP TS 23.040);

значение в диапазоне от 64 до 127: показывает, что сообщение не доставлено, и SMS-центр прекратил попытки доставки (поле TP-Status согласно 3GPP TS 23.040).

6.11.3.4. FwModem_sendUssd

Функция для отправки USSD-запроса оператору сотовой сети.

```
FUNCTION FwModem_sendUssd : DWORD
VAR_INPUT
   dwHandle: DWORD;
   ussdText: STRING(31);
END_VAR
```

Входные параметры:

dwHandle: DWORD

Идентификатор, полученный при открытии модема.

ussdText: STRING(31)

Текст USSD-запроса, например, ‘*100#’ или ‘*102#’ для запроса баланса SIM-карты у операторов сети.

Возвращаемый результат:

Функция возвращает ненулевое значение, если сервис приступил к отправке запроса.

6.11.3.5. FwModem_readUssd

Функция чтения текста ответа на USSD-запрос.

```
FUNCTION FwModem_readUssd : DWORD
VAR_INPUT
   dwHandle: DWORD;
   pUssdText: POINTER TO STRING(160);
END_VAR
```

Входные параметры:

dwHandle: DWORD

Идентификатор, полученный при открытии модема.

pUssdText: POINTER TO STRING(160)

Адрес строки, в которую будет записан ответ на USSD-запрос.

Возвращаемый результат:

–1: в случае ошибки идентификатора модема;

0: если ответа на последний отправленный USSD-запрос пока не получено;

1: если получен новый ответ на USSD-запрос.

6.11.3.6. FwModem_getMemInfo

Функция для определения емкости памяти сообщений модема.

```
FUNCTION FwModem_getMemInfo : INT
VAR_INPUT
   dwHandle: DWORD;
END_VAR
```

Входные параметры:

```
dwHandle: DWORD
```

Идентификатор, полученный при открытии модема.

Возвращаемый результат:

–1: в случае ошибки идентификатора модема;

0: если сообщениями занято менее 75% памяти модема;

1: если сообщениями занято более 75% памяти модема.

6.12. Библиотека FastwelSmsControls.lib

6.12.1. Назначение

Библиотека FastwelSmsControls.lib написана на языке Structured Text в среде CoDeSys 2.3 и предназначена для приема и отправки SMS-сообщений через GSM-модем. В состав библиотеки входит функциональный блок FWSMS, экземпляр которого необходимо периодически вызывать в приложении CoDeSys. Интерфейс функционального блока FWSMS приведен ниже:

```
VAR_INPUT
   blSwitchModemOn:          BOOL;
   strModemInit:             STRING(254);
   byComPortNmb:             BYTE;
   dwBaudrate:               DWORD;
   byStopbits:               FW_HAYES_STOPBITS;
   byParity:                 FW_HAYES_PARITY;
   command:                  FWSMS_COMMAND;
   strDeliveryNmb:           STRING(12);
   strText:                  STRING(160);
   byValidityPer:            BYTE;
   blReportRequired:         BOOL;
   iMessageId:               INT;
END_VAR
```

Вход *blSwitchModemOn* предназначен для запуска и останова модема.

Вход *strModemInit* предназначен для передачи сценария запуска модема сервису.

Вход *byComPortNmb* содержит номер коммуникационного порта, к которому подключен модем.

Вход *dwBaudrate* определяет скорость, на которой коммуникационный порт должен работать с модемом. Допустимо значение из набора: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

Вход *byStopbits* определяет количество используемых коммуникационным портом стоповых бит. Принимает значение из набора (ONESTOPBIT :=0, ONE5STOPBITS :=1, TWOSTOPBITS := 2).

Вход *byParity* определяет количество используемых коммуникационным портом бит четности. Принимает значение из набора (NOPARITY:=0, ODDPARITY:=1, EVENPARITY:=2).

Вход *command* определяет режим работы функционального блока и принимает значение из набора FWSMS_CMD_IDLE, FWSMS_CMD_SEND_SMS, FWSMS_CMD_READ_SMS, FWSMS_CMD_GET_SMS_STATUS, FWSMS_CMD_SEND_USSD, FWSMS_CMD_READ_USSD.

Строка *strDeliveryNmb* в режиме отправки FWSMS_CMD_SEND_SMS содержит номер абонента – получателя SMS-сообщения.

Строка *strText* содержит текст для отправки SMS-сообщения или USSD-запроса.

Вход *byValidityPeriod* в режиме отправки FWSMS_CMD_SEND_SMS указывает время жизни сообщения в сети.

Вход *blReportRequired* в режиме отправки FWSMS_CMD_SEND_SMS указывает сервису модема, что состояние отправленного сообщения необходимо отслеживать.

Вход *iMessageId* содержит в себе идентификатор отправленного сообщения, статус которого запрашивается в режиме FWSMS_CMD_GET_SMS_STATUS.

```
VAR_OUTPUT
   byErrCode:                FWSMS_ERR_CODE;
   iMessageSentId:           INT;
   strSenderNmb:             STRING(12);
   strIncomingText:          STRING(160);
   dtDateTime:               DATE_AND_TIME;
   bySmsStatus:              BYTE;
   eModemStatus:             DWORD;
   blModemMemOverflow:       BOOL;
END_VAR
```

Выход *byErrCode* показывает результат последней выполненной операции и принимает значение из набора FWSMS_OK, FWSMS_ERROR, FWSMS_NEW_MESSAGE_NOT_FOUND, FWSMS_USSD_RESPONSE_NOT_FOUND.

Выход *iMessageSentId* возвращает идентификатор отправленного сообщения.

Выход *strSenderNmb* возвращает телефонный номер отправителя SMS-сообщения в виде строки.

Выход *strIncomingText* возвращает текст входящего SMS-сообщения или ответа на USSD-запрос.

Выход *dtDateTime* возвращает дату и время отправки входящего SMS-сообщения или доставки абоненту отправленного SMS-сообщения.

Выход *bySmsStatus* показывает статус отправленного SMS-сообщения.

Выход *eModemStatus* показывает текущее состояние модема.

Выход *blModemMemOverflow* показывает текущее состояние памяти сообщений модема. Значение *TRUE* выхода блока сигнализирует о том, что более 75% памяти сообщений модема занято. Требуется считать непрочитанные входящие сообщения командой FWSMS_CMD_READ_SMS.

Пример использования библиотеки имеется в проекте fw_hayes_sms_test.pro, который входит в пакет адаптации CoDeSys для Fastwel I/O.

6.12.2. Режимы работы функционального блока FWSMS

Режим работы функционального блока FWSMS выбирается с помощью входа *command*.

6.12.2.1. Режим FWSMS_CMD_IDLE

Функциональный блок в режиме FWSMS_CMD_IDLE может использоваться при инициализации модема или проверки его состояния. На работу блока в данном режиме влияют входы *blSwitchModemOn*, *strModemInit*, *byComPortNmb*, *dwBaudrate*, *byStopbits*, *byParity*. Результат работы блока можно контролировать по выходам *byErrCode*, *eModemStatus*, *blModemMemOverflow*.

6.12.2.2. Режим FWSMS_CMD_SEND_SMS

Режим FWSMS_CMD_SEND_SMS функционального блока FWSMS предназначен для отправки SMS-сообщений. На работу блока в данном режиме влияют входы *blSwitchModemOn, strDeliveryNmb, strText, byValidityPeriod, blReportRequired*. Результат работы блока можно контролировать по выходам *byErrCode, iMessageSentId, eModemStatus, blModemMemOverflow*.

6.12.2.3. Режим FWSMS_CMD_READ_SMS

Режим FWSMS_CMD_READ_SMS функционального блока FWSMS предназначен для чтения входящих SMS-сообщений. На работу блока в данном режиме влияет вход *blSwitchModemOn*.

Результат работы блока можно контролировать по выходам *byErrCode*, *strSenderNmb*, *strIncomingText*, *dtDateTime*, *eModemStatus*, *blModemMemOverflow*.

6.12.2.4. Режим FWSMS_CMD_GET_SMS_STATUS

Режим FWSMS_CMD_GET_SMS_STATUS функционального блока FWSMS предназначен для чтения состояния одного из отправленных сообщений. На работу блока в данном режиме влияют входы *blSwitchModemOn*, *iMessageId*. Результат работы блока можно контролировать по выходам *byErrCode*, *dtDateTime*, *bySmsStatus*, *eModemStatus*, *blModemMemOverflow*.

6.12.2.5. Режим FWSMS_CMD_SEND_USSD

Режим FWSMS_CMD_SEND_USSD функционального блока FWSMS предназначен для отправки USSD-запроса оператору сотовой сети. На работу блока в данном режиме влияют входы *blSwitchModemOn*, *strText*. Результат работы блока можно контролировать по выходам *byErrCode*, *eModemStatus*, *blModemMemOverflow*.

6.12.2.6. Режим FWSMS_CMD_READ_USSD

Режим FWSMS_CMD_READ_USSD функционального блока FWSMS предназначен для чтения ответа на USSD-запрос. На работу блока в данном режиме влияет вход *blSwitchModemOn*. Результат работы можно контролировать по выходам *byErrCode*, *strIncomingText*, *eModemStatus*, *blModemMemOverflow*.

6.12.3. Применение функционального блока FWSMS

6.12.3.1. Начало работы

Перед началом работы с модемом необходимо объявить экземпляр функционального блока FWSMS.

6.12.3.2. Запуск модема

Для запуска модема достаточно установить вход *blSwitchModemOn* в *TRUE*, ввести сценарий инициализации модема в *strModemInit*, установить номер коммуникационного порта *byComPortNmb*, скорость обмена *dwBaudrate*, количество стоповых бит *byStopbits*, четность *byParity*, перевести режим работы *command* блока в FWSMS_CMD_IDLE, на остальные входы подать *0* или пустую строку, а затем периодически вызывать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. Сценарий инициализации модема для GSM-модема Cinterion MC52 может иметь следующий вид:

```
'TIMEOUT 3 ABORT "ERROR" ATE0V1Q0L0S7=20 OK AT+CUSD=1 OK AT+CNMI=2,1,2,2,1 OK
AT+CREG=0 OK ATX4 OK AT+CREG? +CREG: 0,1 OK PAUSE 1'
```

В случае успешной инициализации модема сервис последовательно пройдет через состояния HM_START, HM_INITIALIZATION в HM_READY. Отслеживать состояние сервиса можно по выходу *eModemStatus* функционального блока.

После перехода сервиса модема в режим готовности HM_READY можно приступать к отправке и приему SMS-сообщений и USSD-запросов.

6.12.3.3. Отправка SMS-сообщения

Для отправки SMS-сообщения достаточно перевести вход *command* в состояние FWSMS_CMD_SEND_SMS, установить вход *blSwitchModemOn* в *TRUE*, записать на входы *strDeliveryNmb* и *strText* строки с номером абонента и текстом сообщения, выбрать время жизни сообщения в сети с помощью входа *byValidityPer*, установить вход *blReportRequired* в состояние *TRUE*, если требуется контролировать состояние отправленного сообщения, на остальные входы подать 0 или пустую строку, а затем однократно вызвать экземпляр функционального блока из приложения CoDeSys. После вызова необходимо проверить результат завершения операции путем сравнения кода ошибки *byErrCode* на выходе блока со значением FWSMS_OK. При успешном завершении операции для дальнейшего контроля состояния отправленного сообщения необходимо сохранить его идентификатор *iMessageSentId* в переменной секции VAR приложения CoDeSys.

6.12.3.4. Чтение состояния отправленного сообщения

Для проверки состояния отправленного SMS-сообщения достаточно перевести вход *command* в состояние FWSMS_CMD_GET_SMS_STATUS, установить вход *blSwitchModemOn* в *TRUE,* записать на вход *iMessageId* идентификатор отправленного сообщения, на остальные входы подать 0 или пустую строку, а затем однократно вызвать экземпляр функционального блока из приложения CoDeSys. В случае успешного завершения операции выход *byErrCode* примет значение FWSMS_OK, состояние отправленного сообщения показывает выход *bySmsStatus*, дату и время доставки индицирует выход *dtDateTime* блока. Подробно о возможных состояниях отправленных сообщений изложено в п.6.11.

6.12.3.5. Чтение входящего SMS-сообщения

Для чтения сообщения достаточно перевести вход *command* в состояние FWSMS_CMD_READ_SMS, установить вход *blSwitchModemOn* в *TRUE,* на остальные входы подать 0 или пустую строку, а затем однократно вызвать экземпляр функционального блока из приложения CoDeSys. В случае успешного завершения операции выход *byErrCode* примет значение, отличное от FWSMS_ERROR. Значение FWSMS_OK выхода *byErrCode* означает, что текст нового входящего сообщения можно считать с выхода *strIncomingText,* номер отправителя доступен на *strSenderNmb*, а дату и время отправки сообщения показывает выход *dtDateTime* функционального блока. Значение FWSMS_NEW_MESSAGE_NOT_FOUND выхода *byErrCode* сигнализирует о том, что сервисом модема новых входящих сообщений пока не получено.

6.12.3.6. Отправка USSD-запроса

Для отправки USSD-запроса оператору сотовой сети достаточно перевести вход *command* в состояние FWSMS_CMD_SEND_USSD, установить вход *blSwitchModemOn* в *TRUE,* записать текст USSD-запроса в *strText,* на остальные входы подать 0 или пустую строку, а затем однократно вызвать экземпляр функционального блока из приложения CoDeSys. В случае успешного начала обработки запроса выход *byErrCode* примет значение FWSMS_OK.

6.12.3.7. Чтение ответа на USSD-запрос

Для чтения ответа на USSD-запрос достаточно перевести вход *command* в состояние FWSMS_CMD_READ_USSD, установить вход *blSwitchModemOn* в *TRUE,* на остальные входы подать 0 или пустую строку, а затем однократно вызвать экземпляр функционального блока из приложения CoDeSys. В случае успешного завершения операции чтения выход *byErrCode* примет значение, отличное от FWSMS_ERROR. Значение FWSMS_OK выхода *byErrCode* означает, что текст ответа на USSD-запрос доступен на выходе *strIncomingText* функционального блока. Значение FWSMS_USSD_RESPONSE_NOT_FOUND выхода *byErrCode* сигнализирует о том, что сервисом модема ответа на USSD-запрос пока не получено.

6.12.3.8. Закрытие модема

Для закрытия модема достаточно установить вход *blSwitchModemOn* в *FALSE,* на остальные входы подать *0* или пустую строку, а затем однократно вызвать экземпляр функционального блока с указанными входными параметрами из приложения CoDeSys. В случае успешного завершения операции выход *byErrCode* примет значение FWSMS_OK.

6.13. Библиотека FastwelModbusRTUClientSerial.lib

6.13.1. 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.

6.13.2. Принцип работы

6.13.2.1. 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.

6.13.2.2. 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*.

6.13.2.3. 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.

6.14. Библиотека FastwelPlatformControl.lib

6.14.1. 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*).

6.14.2. Описание функций

6.14.2.1. FwPlatformGetTemperature

Функция не поддерживается на контроллерах CPM711, CPM712, CPM713 и возвращает значение F_PLCTL_NOT_SUPPORTED.

6.14.2.2. 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.

6.14.2.3. 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.

6.14.2.4. 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, ЗА ИСКЛЮЧЕНИЕМ ПЕРЕМЕННЫХ, ОБЪЯВЛЕННЫХ В БЛОКАХ RETAIN. Аналогичные действия выполняются при исполнении команды **Online–Reset** из меню главного окна IDE CoDeSys.
F_RESET_COLD	Выполняется перезапуск программы контроллера. После перезапуска производится инициализация всех переменных программы, ВКЛЮЧАЯ ПЕРЕМЕННЫЕ, ОБЪЯВЛЕННЫЕ В БЛОКАХ RETAIN. 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.

6.14.2.5. 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
```

7. РЕАЛИЗАЦИЯ ПРИКЛАДНЫХ АЛГОРИТМОВ НА C/C++ И РАСШИРЕНИЕ СИСТЕМЫ ИСПОЛНЕНИЯ ПРИЛОЖЕНИЙ CODESYS

7.1. Общие сведения

Настоящий раздел содержит указания по расширению системы исполнения контроллера пользовательским кодом, разрабатываемым на языках общего применения C/C++.

Система исполнения контроллера обеспечивает возможность выполнения пользовательских алгоритмов, реализованных на языках общего применения C/C++ и оформленных в виде специальной библиотеки динамической компоновки (далее – DLL), загруженной в контроллер.

DLL создается при помощи средств разработки Microsoft семейства Visual Studio или Microsoft eMbedded Visual C++ 4.0 SP4 с использованием SDK для образа операционной системы Windows CE 5.0, загруженной в контроллер.

DLL размещается в корневом каталоге основного дискового накопителя контроллера путем загрузки из среды CoDeSys (с предварительной упаковкой в zip-архив и переименованием файла архива в norm.dnl).

Система исполнения пытается загрузить DLL либо при переходе из безопасного режима в нормальный, либо при включении питания контроллера. В случае успешной загрузки DLL выполняется динамическое связывание, после чего система исполнения вызывает пользовательские функции интерфейса DLL и обрабатывает вызовы, совершаемые пользовательским кодом DLL в отношении системы исполнения.

Программист, предполагающий расширять систему исполнения при помощи DLL, должен иметь квалификацию, достаточную для создания системного и прикладного ПО для операционной системы Windows/Windows CE на языках C или C++.

7.2. Требования к рабочему месту разработчика

7.2.1. Требования к конфигурации программного обеспечения

На инструментальном ПК должна быть установлена одна из следующих операционных систем:

1. Windows XP Home или Professional SP3, или
2. Windows Vista (не ниже Home Premium) SP2, или
3. Windows 7 (не ниже Home Premium).

7.2.2. Средства разработки для Windows CE 5.0

Для построения поставляемого примера DLL и создания собственных DLL расширения системы исполнения контроллера для ОС Windows CE 5.0 на инструментальном ПК должна быть установлена одна из следующих сред разработки:

1. Microsoft eMbedded Visual C++ 4.0 SP4, или
2. Microsoft Visual C++ 2008, или
3. Microsoft Visual C++ 2005, или
4. Microsoft Visual C++ .NET 2003, или
5. Microsoft Visual C++ .NET 2002

Утилиты доступа к удаленному устройству с установленной ОС Windows CE 5.0 и менеджер платформы входят в состав Microsoft eMbedded Visual C++ 4.0.

Среда разработка Microsoft eMbedded Visual C++ 4.0 может быть загружена с Web-узла компании Microsoft по следующему адресу:

http://www.microsoft.com/downloads/details.aspx?FamilyId=1DACDB3D-50D1-41B2-A107-FA75AE960856&displaylang=en

Указания по установке, включая информацию о ключе продукта, приведены по следующему адресу:

http://www.microsoft.com/downloads/details.aspx?FamilyId=1DACDB3D-50D1-41B2-A107-FA75AE960856&displaylang=en#Instructions

Пакет обновлений Microsoft eMbedded Visual C++ 4.0 SP4 может быть загружен с Web-узла компании Microsoft по следующему адресу:

http://www.microsoft.com/downloads/details.aspx?FamilyID=4a4ed1f4-91d3-4dbe-986e-a812984318e5&displaylang=en

ВНИМАНИЕ! Адреса на Web-узле Microsoft регулярно меняются. Если какая-либо из указанных ссылок окажется неработоспособной, воспользуйтесь поисковой системой на http://www.microsoft.com.

Кроме того, на инструментальном ПК должен быть установлен пакет программной поддержки используемого контроллера для Windows CE 5.0 (Windows CE 5.0 SDK):

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/SDK/CPM711

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/SDK/CPM712

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/SDK/CPM713

После установки SDK следует создать переменную окружения **`CESDK`**, значение которой должно содержать путь к каталогу установленного Windows CE 5.0 SDK. Например:

"C:\Program Files\Windows CE Tools\wce500\CPM713n"

7.3. Взаимодействие системы исполнения контроллера с DLL пользователя

7.3.1. Общие сведения

Расширение системы исполнения контроллера осуществляется путем размещения DLL с именем FwCdsUsr.dll в корневом каталоге основного дискового накопителя контроллера (абсолютный путь \FixedDisk1). DLL должна быть построена для целевой операционной системы Windows CE 5.0 с использованием SDK для используемого типа контроллера.

Файл FwCdsUsr.dll может быть загружен в контроллер следующим образом:

1. FwCdsUsr.dll упаковывается на инструментальном ПК в zip-архив
2. В архив помещается пустой текстовый файл, имя которого содержит тип контроллера, суффикс *UP* и расширение .log: CPM71*X*UP.log. Например, для контроллера CPM713 файл должен иметь имя CPM713UP.log.
3. Файл архива переименовывается в norm.dnl,
4. Полученный файл norm.dnl загружается в контроллер из среды разработки CoDeSys командой **Online–Write file to PLC**. По завершении загрузки файла произойдет автоматический перезапуск системы исполнения контроллера и распаковка загруженного архива, после чего система исполнения попытается загрузить DLL и выполнить связывание.

FwCdsUsr.dll, помимо точки входа (**`DllMain`**), должна содержать единственную экспортируемую функцию **`F_CdsUsr_link`**, описание которой приведено в п. 7.3.2 настоящего руководства.

7.3.2. Функция связывания системы исполнения с DLL пользователя F_CdsUsr_Link

DLL в обязательном порядке должна содержать экспортируемую функцию:

```
PF_CDS_USR_IFACE F_CdsUsr_link(PF_USR_CDS_IFACE pRuntimeIface);
```

Загрузка DLL осуществляется при запуске системы исполнения или после перехода из безопасного в нормальный режим.

После успешной загрузки DLL система исполнения выполняет поиск данной функции в загруженной DLL. Если функция не найдена, DLL выгружается, а при успешном обнаружении – производится вызов данной функции. В качестве параметра функции система исполнения передает указатель на интерфейс доступа к сегментам данных приложения CoDeSys (см. п. 7.3.3.1). В качестве

результата функция должна вернуть системе исполнения указатель на интерфейс, предоставляемый DLL для координации совместной работы с системой исполнения (см. п. 7.3.4.1).

7.3.3. Интерфейс доступа к сегментам данных приложения CoDeSys

7.3.3.1. F_USR_CDS_IFACE

Система исполнения предоставляет пользовательской DLL интерфейс доступа к сегментам данных приложения CoDeSys, загруженного в контроллер. Таким образом, приложение CoDeSys может содержать только объявления переменных и ни одной строки кода на языках IEC 61131-3, за исключением ‘;’ в программе PLC_PRG.

Интерфейс представлен структурой **F_USR_CDS_IFACE**, которая объявлена в заголовочном файле fcds_usr.h следующим образом:

```
struct F_USR_CDS_IFACE
{
  // Контрольное поле. Должно содержать размер структуры в байтах
  size_t                    this_size;

  // Функция чтения участка сегмента данных приложения CoDeSys
  PF_USR_CDS_READ_VAR       pfReadVariable;

  // Функция записи участка сегмента данных приложения CoDeSys
  PF_USR_CDS_WRITE_VAR      pfWriteVariable;

  // Функция получения текущего статуса системы исполнения
  PF_USR_CDS_GET_STATUS     pfGetStatus;

  // Функция получения текущего режима системы исполнения
  PF_USR_CDS_GET_MODE       pfGetMode;

};
```

Поле **this_size** инициализировано размером структуры **F_USR_CDS_IFACE**, и позволяет убедиться в совпадении версий структуры, используемой DLL и системой исполнения.

Поля с префиксом **pf** содержат указатели на функции, предоставляемые пользовательской DLL системой исполнения.

7.3.3.2. pfReadVariable/pfWriteVariable

Поля **pfReadVariable** и **pfWriteVariable** содержат указатели на функции чтения и записи участка одного из сегментов приложения CoDeSys, загруженного в контроллер. Типы данных функций декларированы в заголовочном файле fcds_usr.h следующим образом:

```
// Чтение переменной приложения
typedef F_USR_RESULT (*PF_USR_CDS_READ_VAR)(const PF_CDS_USR_VAR pVar,
                                            void* pUserDst);
// Запись в переменную приложения
typedef F_USR_RESULT (*PF_USR_CDS_WRITE_VAR)(const PF_CDS_USR_VAR pVar,
                                             void* pUserSrc);
```

В качестве первого параметра каждая из функций принимает указатель на описатель переменной (непрерывного участка) в одном из сегментов данных приложения CoDeSys:

```
struct F_CDS_USR_VAR
{
  // Идентификатор сегмента данных приложения
  F_CDS_RT_SEGMENT  segment_id;

  // Смещение в байтах в выбранном сегменте данных приложения
  size_t            offset;

  // Длина (в байтах) участка в выбранном сегменте, подлежащего чтению/записи
  size_t            len;

};
```

Идентификатор сегмента представлен типом **F_CDS_RT_SEGMENT**:

```
// Тип сегмента данных приложения
typedef size_t F_CDS_RT_SEGMENT;
```

и может принимать следующие значения:

```
// Сегмент флагов и переменных типа %M
#define CDS_SEG_DATAID_MEMORY     0
// Сегмент входных данных
#define CDS_SEG_DATAID_INPUT      1

// Сегмент выходных данных
#define CDS_SEG_DATAID_OUTPUT     2

// Сегмент энергонезависимых переменных
#define CDS_SEG_DATAID_RETAIN     3

// Сегмент глобальных внутренних переменных блоков и программ
#define CDS_SEG_DATAID_GLOBVARS   4
```

Обе функции в качестве результата возвращают статус операции:

```
// Результат обращения DLL к системе исполнения
typedef enum F_USR_RESULT
{
  // Успех
  F_UC_OK,

  // Один или оба параметра равны NULL.
  F_UC_BAD_PARAM,

  // Вызов функции чтения или записи произведен в момент, когда
  // система исполнения не готова к его выполнению.
  // Например, чтение/запись переменных не разрешаются во время
  // обновления приложения после загрузки из среды CoDeSys.
  F_UC_INVALID_STATE,

  // Первый параметр описывает несуществующий сегмент, или
  // участок, определенный заданными смещением и длиной
  // находится за пределами выбранного сегмента данных приложения.
  F_UC_BAD_REQUEST

}F_USR_RESULT;
```

Функция чтения **pfReadVariable** в качестве второго параметра принимает указатель на переменную DLL (буфер), в которую требуется прочитать содержимое участка сегмента данных приложения, описанного первым параметром.

Функция записи **pfWriteVariable** в качестве второго параметра принимает указатель на переменную DLL, содержимое которой требуется записать в участок сегмента данных приложения, описанный первым параметром.

Для обращения к функциям **pfReadVariable** и **pfWriteVariable** из DLL в заголовочном файле fcds_usr.h имеются два макроопределения:

```
F_USR_RESULT F_CdsUsr_readVariable(PF_USR_CDS_IFACE pif,
                                   const PF_CDS_USR_VAR pVar,
                                   void* pDst);

F_USR_RESULT F_CdsUsr_writeVariable(PF_USR_CDS_IFACE pif,
                                    const PF_CDS_USR_VAR pVar,
                                    void* pSrc);
```

В качестве первого параметра каждому макроопределению должен быть передан указатель на интерфейс **F_USR_CDS_IFACE**, ранее полученный DLL от системы исполнения при вызове функции **F_CdsUsr_link**, экспортируемой DLL.

Более подробное описание работы функций чтения и записи сегментов данных приложения приведено в п. 7.3.6 настоящего руководства.

7.3.3.3. Функция чтения статуса приложения

Для получения текущего статуса системы исполнения может быть использовано макроопределение:

```
F_CDS_RT_STATUS F_CdsUsr_getStatus(PF_USR_CDS_IFACE pif);
```

которое возвращает одно из следующих значений:

```
// Статус контроллера (Run/Stop/Breakpoint/Unknown)
typedef size_t F_CDS_RT_STATUS;

// Неопределенный (система пока не готова к работе)
#define CDSS_UNDEFINED  0

// Приложение CoDeSys функционирует нормально
#define CDSS_RUN        1

// Приложение CoDeSys остановлено
#define CDSS_STOP       2

// Приложение CoDeSys приостановлено на точке останова
#define CDSS_BREAKPOINT 3
```

В качестве первого параметра макроопределение принимает указатель на интерфейс **F_USR_CDS_IFACE**, ранее переданный системой исполнения при вызове функции **F_CdsUsr_link**, экспортируемой DLL.

Макроопределение **F_CdsUsr_getMode** возвращает текущий режим работы системы исполнения. В текущей версии система исполнения взаимодействует с DLL только в нормальном режиме, поэтому результат всегда будет равен 1.

7.3.4. Интерфейс координации совместной работы с DLL

7.3.4.1. F_CDS_USR_IFACE

В качестве результата вызова функции **F_CdsUsr_link** пользовательская DLL должна вернуть системе исполнения указатель на интерфейс координации совместной работы DLL с системой исполнения. Интерфейс представлен структурой **F_CDS_USR_IFACE** в заголовочном файле fcds_usr.h:

```
struct F_CDS_USR_IFACE
{
  // Контрольное поле, должно содержать размер структуры в байтах.
  size_t                    this_size;

  // Функция проверки допустимости совместной работы
  // загруженной DLL с текущим приложением CoDeSys, загруженным
  // в контроллер.
  PF_CDS_USR_ACCEPT_PROJECT pfAccept;

  // Функция обработки событий в системе исполнения.
  PF_CDS_USR_HANDLE_EVENT   pfEvent;

  // Функция, вызываемая системой исполнения на контексте
  // сервисной задачи с периодом, заданным в конфигурации
  // приложения параметром EventsRate.
  PF_CDS_USR_DO_CYCLE       pfDoCycle;

};
```

Поле **this_size** инициализировано размером структуры **F_CDS_USR_IFACE**, и позволяет убедиться, что версии структуры, используемой DLL и системой исполнения, совпадают.

Поля с префиксом **pf** должны содержать указатели на функции, передаваемые пользовательской DLL системе исполнения.

7.3.4.2. pfAccept

Функция **pfAccept** вызывается системой исполнения непосредственно перед запуском приложения CoDeSys, загруженного в контроллер, и предназначена для проверки допустимости совместной работы загруженной DLL с текущим приложением CoDeSys. Тип функции декларирован в заголовочном файле fcds_usr.h следующим образом:

```
typedef F_USR_CODE_RESULT
(*PF_CDS_USR_ACCEPT_PROJECT)(const PF_CDS_PROJECT_INFO pProjectInfo);
```

Функция принимает указатель на структуру **F_CDS_PROJECT_INFO**, содержащую информацию о текущем приложении CoDeSys, и возвращает значение типа **F_USR_CODE_RESULT**.

Тип **F_USR_CODE_RESULT** декларирован в заголовочном файле fcds_usr.h следующим образом:

```
typedef size_t F_USR_CODE_RESULT;
```

```
// Допускается совместная работа DLL с текущим приложением.
#define F_UCR_OK  0

// Совместная работа DLL с текущим приложением не допускается.
// Система исполнения не будет вызывать другие функции интерфейса DLL.
#define F_UCR_FAILED 1

// Совместная работа DLL с текущим приложением не допускается.
// Система исполнения должна перевести контроллер в безопасный режим.
#define F_UCR_ABORT 2
```

Тип **F_CDS_PROJECT_INFO** декларирован в заголовочном файле fcds_usr.h следующим образом:

```
// Максимальный размер строки в полях F_CDS_PROJECT_INFO
#define CDS_TEXT_SIZE 80

// Информация о сегментах данных текущего приложения CoDeSys
struct F_CDS_SEGS_INFO
{
  // Размер сегмента флагов и памяти %M
  size_t  memory_size;

  // Размер сегмента входных данных
  size_t  inputs_size;

  // Размер сегмента выходных данных
  size_t  outputs_size;

  // Размер сегмента энергонезависимых данных
  size_t  retains_size;

  // Размер сегмента глобальных данных
  size_t  global_size;

};

// Информация о текущем приложении CoDeSys
struct F_CDS_PROJECT_INFO
{
  // Контрольное поле, должно содержать размер структуры в байтах.
  size_t           this_size;

  // Время и дата трансляции проекта в формате
  // DATA_AND_TIME IEC 61131-3 (UNIX time или
  // время в секундах, начиная с 1 января 1970 года.
  unsigned long    date;

  // Имя файла проекта с расширением pro.
  char             project[CDS_TEXT_SIZE];

  // Заголовок проекта
  char             title[CDS_TEXT_SIZE];

  // Информация о версии проекта, включая версию
  // системы исполнения контроллера
  char             version[CDS_TEXT_SIZE];

  // Информация об авторе проекта
  char             author[CDS_TEXT_SIZE];

  // Дополнительная описательная информация о проекте
  char             description[CDS_TEXT_SIZE];

  // Информация о размерах сегментов данных приложения
  F_CDS_SEGS_INFO  segments_info;

  // Период сервисной задачи в мс.
  // Содержит значению, заданное параметром EventsRate.
  size_t          cycle_period;

  // Зарезервировано
  void*           pReserved;
};
```

Поле **date** содержит время трансляции приложения в секундах, начиная с 1 января 1970 г (см. тип DATE_AND_TIME CoDeSys или описание функции time() стандартной библиотеки C).

Поля **project**, **title**, **version**, **author** и **description** содержат строковую информацию о текущем загруженном приложении CoDeSys. Данная информация может быть определена пользователем в диалоговой панели **Project Information**, выводимой на экран по команде **Project–Project Info** главного меню среды разработки CoDeSys. Обратите внимание, что данные поля являются массивами типа **char**, т.е. для отображения соответствующих строк на экране необходимо преобразование данных строк в **wchar_t*** при помощи функции **MultiByteToWideChar** (см. Windows API).

Поле **segments_info** содержит размеры сегментов данных загруженного приложения, а **cycle_period** – период сервисной задачи (в мс), с которым будет вызываться функция **pfDoCycle** интерфейса DLL.

Функция **pfAccept**, реализуемая в DLL, может проверить информацию о проекте, переданную в качестве параметра, и, в зависимости от результата проверки:

1. Разрешить системе исполнения вызывать функции **pfEvent** и/или **pfDoCycle**, вернув значение **F_UCR_OK**.
2. Запретить системе исполнения вызывать какие-либо функции интерфейса **F_CDS_USR_IFACE**, вернув значение **F_UCR_FAILED**.
3. Потребовать перехода контроллера в безопасный режим, вернув значение **F_UCR_ABORT**.

Например, если пользовательской DLL безразлично, с каким приложением CoDeSys ей предстоит совместно работать, она может всегда возвращать **F_UCR_OK**.

7.3.4.3. pfEvent

Функция **pfEvent** вызывается системой исполнения при возникновении системных событий, перечисленных в п. 4.2.4.4, исключая событие F_EVENT_TIMER. Тип функции декларирован в заголовочном файле fcds_usr.h следующим образом:

```
typedef void (*PF_CDS_USR_HANDLE_EVENT)(F_CDS_EVENT_TYPE type);
```

Параметр типа **F_CDS_EVENT_TYPE** содержит код события:

```
// При выполнении Online-Start и непосредственно перед началом работы приложения
#define CDS_EVENT_START           1

// При выполнении Online-Stop
#define CDS_EVENT_STOP            2

// При выполнением Online-Reset непосредственно перед перезапуском
#define CDS_EVENT_BEFORE_RESET    3

// В момент готовности к запуску вновь загруженной программы.
// Перед данным событием вызывается функция pfAccept.
#define CDS_EVENT_ONLINE_CHANGE   33

// В момент начала сетевой загрузки нового приложения CoDeSys
#define CDS_EVENT_BEFORE_DOWNLOAD 34

// При выполнении Online-Login
#define CDS_EVENT_LOGIN           501

// В момент перед запуском программы до CDS_EVENT_START
#define CDS_EVENT_ON_INIT         1501

// При включении питания перед запуском программы до CDS_EVENT_INIT
#define CDS_EVENT_POWER_ON        1502

// При выполнении Online-Logout
#define CDS_EVENT_LOGOUT          1503
```

Пользовательская DLL не обязана реализовывать данную функцию, – в этом случае достаточно обнулить поле **pfEvent** структуры **F_CDS_USR_IFACE**. Если же данная функция определена, категорически запрещается выполнять в ней длительные и/или блокирующие операции. При

нарушении данного правила контроллер перейдет в безопасный режим либо по сторожевому таймеру, либо по подозрению в зацикливании пользовательского кода.

7.3.4.4. pfDoCycle

Функция **pfDoCycle** вызывается системой исполнения на контексте высокоприоритетной сервисной задачи с периодом, установленным пользователем в конфигурации приложения CoDeSys параметром *SampleRate* (см. п. 4.1.1).

Тип функции декларирован в заголовочном файле fcds_usr.h следующим образом:

```
typedef
F_USR_CODE_RESULT
(*PF_CDS_USR_DO_CYCLE)( const PF_CDS_DATA_SEGMENT pInputSegment,
                        F_CDS_RT_MODE mode,
                        F_CDS_RT_STATUS status);
```

Первый параметр является указателем на структуру:

```
struct F_CDS_DATA_SEGMENT
{
  // Указатель на буфер сегмента данных
  void* pData;

  // Размер сегмента
  size_t size;

};
```

и содержит "мгновенный снимок" области входных данных приложения CoDeSys, "снятый" непосредственно перед вызовом функции **pfDoCycle.**

Второй и третий параметры являются кодами режима и статуса системы исполнения перед вызовом **pfDoCycle**. Код режима в текущей версии системы исполнения всегда равен 1, что соответствует нормальному режиму. Описание значений кода статуса приведено в п. 7.3.3.3 настоящего руководства.

Описание типа возвращаемого значения приведено в п. 7.3.4.2. Таким образом, пользовательская DLL может в любой момент перевести контроллер в безопасный режим либо прекратить дальнейшие вызовы своего кода, реализуемого функциями **pfEvent** и **pfDoCycle.**

Пользовательская DLL не обязана реализовывать данную функцию. В этом случае достаточно обнулить поле **pfDoCycle** структуры **F_CDS_USR_IFACE.** В случае, если данная функция определена, категорически запрещается выполнять в ней длительные и/или блокирующие операции. При нарушении данного правила контроллер перейдет в безопасный режим либо по сторожевому таймеру, либо по подозрению в зацикливании пользовательского кода.

7.3.5. Вызов функций интерфейса координации совместной работы с DLL пользователя

7.3.5.1. Последовательность обращения к функциям интерфейса DLL

Взаимодействие между системой исполнения и пользовательской DLL осуществляется следующим образом:

1. После включения питания контроллера или сразу после загрузки приложения в контроллер, который находился в безопасном режиме, система исполнения пытается загрузить динамическую библиотеку с именем FwCdsUsr.dll.

   Если загрузка DLL не удалась, система исполнения продолжает работу в нормальном режиме.
2. Если загрузка DLL произведена успешно, система исполнения выполняет поиск в DLL функции **F_CdsUsr_link.** Если функция не найдена, DLL выгружается, и система исполнения продолжает работу в нормальном режиме.
3. Если функция **F_CdsUsr_link** найдена, система исполнения вызывает ее, передав в качестве параметра указатель на интерфейс доступа к сегментам данных приложения (см. п. 7.3.3).

   Если при вызове функции **F_CdsUsr_link** происходит исключение, система

исполнения переходит в безопасный режим, сохранив в файле normdump.txt диагностическую строку:
"*Error <код исключения> occurred while linking to FwCdsUsr.dll*".

После успешного вызова функции **F_CdsUsr_link** проверяется, вернула ли она указатель на интерфейс координации совместной работы с DLL (см. п. 7.3.4).

Если возвращен NULL, система исполнения продолжает работу в нормальном режиме, считая, что DLL не предполагает координировать исполнение своего кода с системой исполнения.

Если возвращен указатель на интерфейс, то проверяется значение его поля **this_size** на равенство с размером структуры **F_CDS_USR_IFACE**. Если размеры не совпадают, система исполнения переходит безопасный режим, сохранив в файле normdump.txt диагностическую строку:
"*User Dll interface size mismatch in FwCdsUsr.dll*".

4. Если проверка возвращенного указателя на интерфейс DLL завершена успешно, система исполнения проверяет в нем на NULL указатель на функцию **pfAccept**.

   В случае равенства NULL **pfAccept**, система исполнения продолжает работу в нормальном режиме, однако функции **pfEvent** и **pfDoCycle** интерфейса DLL вызываться не будут.

   Если функция **pfAccept** реализована в интерфейсе DLL, система исполнения вызывает ее, передав в качестве параметра указатель на структуру **F_CDS_PROJECT_INFO**, содержащую информацию о текущем приложении, загруженном в контроллер (см. п. 7.3.4.2).

   Если при вызове функции **pfAccept** происходит исключение, система исполнения переходит в безопасный режим, сохранив в файле normdump.txt диагностическую строку:
   "*Error <код исключения> occurred while calling FwCdsUsr.dll.accept()*".

   Если функция **pfAccept** возвращает код **F_UCR_ABORT**, система исполнения переходит в безопасный режим, сохранив в файле normdump.txt диагностическую строку:
   "*Safe mode requested while calling FwCdsUsr.dll.accept()*".

   Если функция **pfAccept** возвращает код **F_UCR_FAILED**, система исполнения продолжает работу в нормальном режиме, однако ни одна из функций интерфейса DLL более вызываться не будет.

   Если функция **pfAccept** возвращает код **F_UCR_OK**, система исполнения продолжает работу в нормальном режиме, будучи готовой вызывать другие функции DLL.

5. Если система исполнения стартует после включения питания контроллера, **pfAccept** вернула код **F_UCR_OK**, а в интерфейсе DLL имеется функция **pfEvent**, реализованная пользователем, производится вызов функции **pfEvent** с кодом события **CDS_EVENT_POWER_ON**.

   Если старт происходит после загрузки приложения из среды разработки CoDeSys, вызов **pfEvent** производится с кодом события **CDS_EVENT_LOGIN**.

   Далее выполняются вызовы функции **pfEvent** с кодами событий **CDS_EVENT_ON_INIT** и **CDS_EVENT_START**.

   В случае, если при вызове **pfEvent** происходит исключение, система исполнения переходит в безопасный режим, сохранив в файле normdump.txt диагностическую строку:
   "*Error <код исключения> occurred while calling FwCdsUsr.dll.event(<код события>)*"

6. Если функция **pfAccept** ранее вернула код **F_UCR_OK**, а в интерфейсе DLL определена функци **pfDoCycle**, система исполнения вызывает данную функцию с периодом, заданным параметром EventsRate в конфигурации приложения.
   В качестве первого параметра функции передается указатель на буфер, размер которого равен размеру области входных данных приложения и содержит данные всей области входных данных приложения, находившиеся в ней непосредственно перед вызовом **pfDoCycle**.

   Если при вызове **pfDoCycle** происходит исключение, система исполнения переходит в безопасный режим, сохранив в файле normdump.txt диагностическую строку: "*Error <код исключения> occurred while calling FwCdsUsr.dll.cycle()*".

   Если **pfDoCycle** возвращает код **F_UCR_ABORT**, система исполнения переходит в безопасный режим, сохранив в файле normdump.txt диагностическую строку: "*Safe mode requested while calling FwCdsUsr.dll.cycle()*".

   Если **pfDoCycle** возвращает код **F_UCR_FAILED**, система исполнения продолжает работу в нормальном режиме, однако ни одна из функций интерфейса DLL более вызываться не будет.

   При возврате **F_UCR_OK** система исполнения продолжает циклически вызывать **pfDoCycle**.
7. В процессе функционирования вызывается функция **pfEvent**, если происходят события **CDS_EVENT_STOP**, **CDS_EVENT_START**, **CDS_EVENT_BEFORE_RESET**, **CDS_EVENT_LOGIN**, **CDS_EVENT_LOGOUT** и **CDS_EVENT_BEFORE_DOWNLOAD**.

7.3.5.2. Обработка исключений в коде функций интерфейса DLL

При вызове любой функции интерфейса DLL система исполнения обрабатывает любое необработанное исключение, возникающее в процессе вызова. Если предполагается самостоятельная обработка некоторых или всех исключений в пользовательском коде, в DLL должен быть использован соответствующий механизм (например, Structured Exception Handling – структурированная обработка исключения).

7.3.6. Вызов функций интерфейса системы исполнения из DLL пользователя

7.3.6.1. Общие сведения

Функции **pfReadVariable** и **pfWriteVariable** интерфейса, предоставляемого системой исполнения пользовательской DLL, могут вызываться в любой момент после события **CDS_EVENT_ON_INIT** и до **CDS_EVENT_BEFORE_DOWNLOAD**.

В промежутке времени между событиями **CDS_EVENT_BEFORE_DOWNLOAD** и **CDS_EVENT_ONLINE_CHANGE**, а также после **CDS_EVENT_BEFORE_RESET**, вызовы данных функций могут завершиться с кодом возврата **F_UC_INVALID_STATE** либо **F_UC_BAD_REQUEST**.

7.3.6.2. Вызов функции чтения сегментов приложения

Если в качестве идентификатора сегмента в структуре **F_CDS_USR_VAR**, указатель на которую передается функции **pfReadVariable** первым параметром, используются **CDS_SEG_DATAID_MEMORY**, **CDS_SEG_DATAID_RETAIN** или **CDS_SEG_DATAID_GLOBVARS**, то при правильных параметрах вызова будет возвращено содержимое участка соответствующего сегмента данных приложения. Доступ из DLL к данным сегментам не взаимоисключается с доступом к их содержимому со стороны приложения CoDeSys и подсистемы целевой визуализации.

Если в качестве идентификатора сегмента используется **CDS_SEG_DATAID_INPUT**, то при правильных параметрах вызова будет возвращено содержимое участка не собственно сегмента входных данных приложения, а промежуточного буфера области входных данных, используемого системой исполнения для связи приложения с окружением.

Если в качестве идентификатора сегмента используется **CDS_SEG_DATAID_OUTPUT**, то при правильных параметрах вызова будет возвращено содержимое участка не собственно сегмента

выходных данных приложения, а промежуточного буфера области выходных данных, используемого системой исполнения для связи приложения с окружением.

Операции чтения сегментов **CDS_SEG_DATAID_INPUT** и **CDS_SEG_DATAID_OUTPUT** взаимоисключены с обращениями к соответствующим областям входных и выходных данных из других задач и сервисов системы исполнения.

7.3.6.3. Вызов функции записи в сегменты приложения

Если в качестве идентификатора сегмента в структуре **F_CDS_USR_VAR**, указатель на которую передается функции **pfWriteVariable** первым параметром, используются **CDS_SEG_DATAID_MEMORY**, **CDS_SEG_DATAID_RETAIN** или **CDS_SEG_DATAID_GLOBVARS**, то при правильных параметрах вызова произойдет запись в участок соответствующего сегмента данных приложения. Доступ из DLL к данным сегментам не взаимоисключается с доступом к их содержимому со стороны приложения CoDeSys и подсистемы целевой визуализации.

Если в качестве идентификатора сегмента используется **CDS_SEG_DATAID_INPUT**, то попытка записи будет отклонена, и функция **pfWriteVariable** вернет код **F_UC_BAD_REQUEST.**

Если в качестве идентификатора сегмента используется **CDS_SEG_DATAID_OUTPUT**, то при правильных параметрах вызова запись в соответствующий участок области выходных данных и сегмента выходных данных приложения будет произведена в том, и только в том, случае, если на данный участок не ссылается ни одна программная единица приложения CoDeSys, загруженного в контроллер. Это позволяет избежать неопределенности формируемых значений выходных переменных, выводимых в окружение.

Операция записи в сегмент **CDS_SEG_DATAID_OUTPUT** взаимоисключена с обращениями по чтению к области выходных данных со стороны других задач и сервисов системы исполнения.

7.3.7. Получение информации о размещении переменных приложения CoDeSys

7.3.7.1. Формирование файла символической информации о проекте CoDeSys

Для получения информации о размещении переменных в сегментах данных приложения CoDeSys, совместно с которым предполагается исполнять пользовательский код DLL, требуется в среде разработки CoDeSys сформировать файл символической информации для проекта приложения.

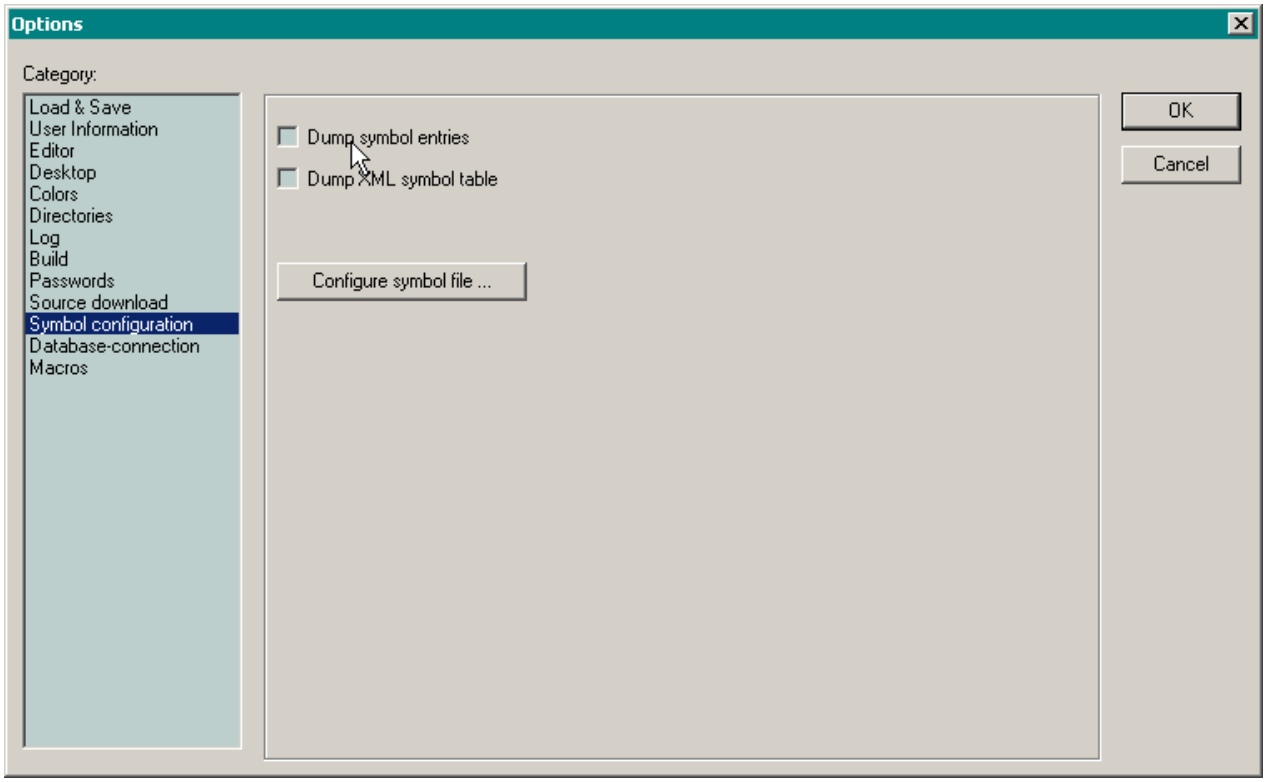


Рис. 31. Конфигурация символической информации в диалоговой панели Options

Для активизации формирования файла символической информации при каждом построении проекта командой **Project–Rebuild All** выполните следующие действия:

1. Выполните команду **Project–Options** в главном меню CoDeSys и в появившейся диалоговой панели **Options** выберите категорию опций **Symbol Configuration**, как показано на рис. 31.
2. Отметьте флажок **Dump symbol entries** и нажмите кнопку **Configure symbol file...**. На экран монитора будет выведена диалоговая панель **Set object attributes**, показанная на рис. 32.

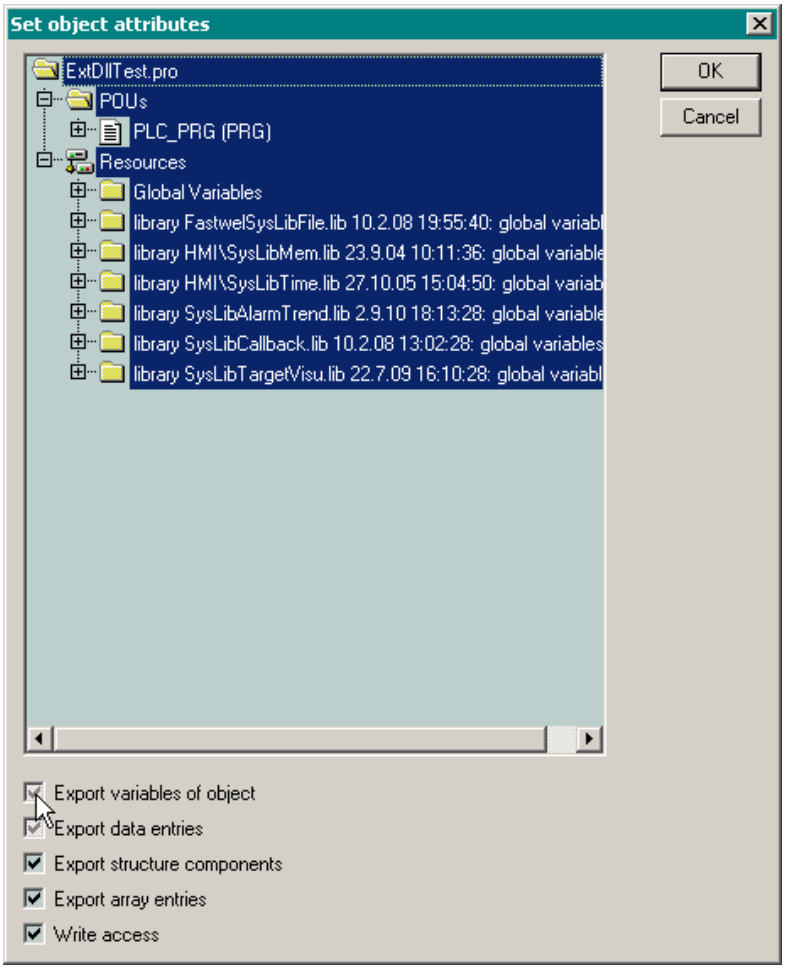


Рис. 32. Внешний вид диалоговой панели Set object attributes

3. Дважды щелкните левой кнопкой мыши на флажком **Export variables of object** и отметьте флажок **Export data entries**, так, чтобы цвет отметки над данными флажками был черным, после чего закройте диалоговые панели **Set object attributes** и **Options** нажатием кнопок **OK**.

После выполнения указанных действий при каждом построении проекта командной **Project–Rebuild All** в каталоге расположения файла проекта будет генерироваться текстовый файл с расширением *.sym, содержащий информацию о размещении переменных приложения в сегментах данных. Для того, чтобы иметь стопроцентную уверенность в актуальности сгенерированной информации, перед выполнением команду **Project–Rebuild All** выполняйте команду **Project Clean all**.

7.3.7.2. Формат символической информации

Файл символической информации с расширением *.sym содержит записи о размещении переменных приложения в сегментах данных в следующем формате:

```
[POU].<var_name>([.<var_subname>:<iec_type>]|[:DATA|<iec_type>]):<seg_id>:<seg_off>:<len>
```

где:

POU – имя программной единицы, содержащей переменную;

var_name – имя переменной или элемента массива;

var_subname – имя поля структурной переменной;

iec_type – имя типа переменной IEC 61131-3;

DATA – обозначение области данных, отведенных для переменной непримитивного типа целиком;

seg_id – идентификатор сегмента данных (1 – входной; 2 – выходной; 3 – энергонезависимых переменных; 4 – глобальных переменных; 0 – флаговый);

seg_off – байтовое смещение переменной в сегменте;

len – длина переменной в байтах.

Например:

Массив

```
.DummyArray:ARRAY [0..5] OF BYTE:4:363:6:r:16#02000040
```

описывает размещение массива с именем DummyArray, состоящего из шести элементов типа BYTE, в сегменте глобальных данных (4) по смещению 363 общей длиной 6 байт.

Элемент массива

```
.DummyArray[4]:BYTE:4:367:1:r:16#02000040
```

описывает размещение пятого элемента массива DummyArray в сегменте глобальных данных (4) по смещению 367 длиной 1 байт.

Структура

```
.IO_Information:DATA:4:339:16:r:16#12000040
```

описывает размещение структурной переменной с именем IO_Information в сегменте глобальных данных (4) по смещению 339 длиной 16 байт.

Поле структуры

```
.IO_Information.ErrorsCount:DWORD:4:351:4:r:16#00000003
```

описывает размещение поля ErrorsCount структурной переменной с именем IO_Information в сегменте глобальных данных (4) по смещению 351 длиной 4 байта.

Переменная примитивного типа

```
.ExternalRamp:LREAL:4:355:8
```

описывает размещение переменной с именем ExternalRamp типа LREAL в сегменте глобальных данных по смещению 355 длиной 8 байт.

Переменные типа BOOL, отображенные на битовые адреса в области входных или выходных данных, представляются несколько иначе:

```
[POU].<var_name>:BOOL:<seg_id>:<seg_bit_off>:0
```

где:

POU – имя программной единицы, содержащей переменную;

var_name – имя переменной;

BOOL – обозначение булева типа;

seg_id – идентификатор сегмента данных (1 – входной; 2 – выходной);

seg_bit_off – битовое смещение переменной в сегменте;

0 – длина переменной в байтах.

Например:

```
.bOut:BOOL:2:8:0
```

описывает размещение булевой переменной bOut, отображенной на 8-й, начиная с 0, бит в области выходных данных приложения.

Более подробные сведения о файлах символической информации приведены в документации на среду разработки CoDeSys.

8. СООБЩЕНИЯ ОБ ОШИБКАХ

8.1. Общие сведения

Настоящий раздел содержит описание информационных сообщений, сохраняемых в файле *normdump.txt* и отображаемых в поле **Cause** окна консоли системы исполнения в безопасном режиме, а также о соответствующих возможных причинах перехода системы исполнения в безопасный режим.

Существуют следующие классы причин перехода системы исполнения в безопасный режим:

1. Исключение – причины данного класса связаны с проявлением дефектов в пользовательском коде приложения или в системном программном обеспечении, при наличии которых система исполнения не может обеспечить предсказуемое выполнение пользовательских алгоритмов и функционирование периферийных устройств, подключенных к контроллеру.
2. Дефицит системных ресурсов – причины данного класса связаны с нехваткой памяти или других ресурсов, возникающей после загрузки приложения или в процессе его работы.
3. Зацикливание в пользовательском коде – причины данного класса связаны с наличием бесконечных (или почти бесконечных) циклов в пользовательском коде либо при остановке пользовательского кода в функциях интерфейса, возвращаемого системе исполнения из DLL расширения. Контроль дефектов данного класса выполняется системой исполнения при помощи программного сторожевого таймера с интервалом от 10 до 20 с.
4. Ошибка конфигурации приложения – возникают при запуске приложения и могут быть связаны с неправильными значениями параметров **PLC Configuration**, превышением допустимых значений параметров системы исполнения и другими причинами, делающими невозможным дальнейшее функционирования системы исполнения в нормальном режиме.

8.2. Исключения

8.2.1. Типы исключений

Коды типов исключений приведены в табл. 10.

Таблица 10

Обозначение	Причина
EXCEPTION_ACCESS_VIOLATION	Ошибка доступа к памяти по чтению или записи
EXCEPTION_DATATYPE_MISALIGNMENT	Попытка доступа к памяти с нарушением выравнивания
EXCEPTION_ARRAY_BOUNDS_EXCEEDED	Попытка доступа за пределы массива при наличии аппаратного контроля
EXCEPTION_INT_DIVIDE_BY_ZERO	Попытка целочисленного деления на 0
EXCEPTION_INT_OVERFLOW	Переполнение при выполнении операции с целочисленными операндами
EXCEPTION_PRIV_INSTRUCTION	Попытка исполнения привилегированной инструкции
EXCEPTION_NONCONTINUABLE_EXCEPTION	Попытка продолжения исполнения после исключения, не предусматривающего возможность продолжения исполнения
EXCEPTION_ILLEGAL_INSTRUCTION	Попытка исполнения неизвестной инструкции
EXCEPTION_STACK_OVERFLOW	Переполнение стека

8.2.2. Исключения при исполнении кода приложения CoDeSys

Информация об исключении, возникшем при исполнении кода программной единицы, представляется следующей строкой:

<EXCPT_TYPE> exception at <addr>. Task:<task_name> POU:<pou_num> Offset:<pou_offset>

где:

EXCPT_TYPE – тип исключения в соответствии с табл. 10;

addr – абсолютный адрес в сегменте кода;

task_name – имя задачи приложения, в которой произошло исключение;

pou_num – номер программной единицы (программы, экземпляра блока или функции);

pou_offset – смещение (в байтах) внутри кода программной единицы.

Имеется возможность определить точное место в исходном тексте приложения CoDeSys, в котором произошло исключение, если приложение реализовано на текстовых языках стандарта IEC 61131-3. Для этого следует:

1. Запустить среду разработки CoDeSys из командной строки с ключом **/debug**.
2. Открыть проект, при исполнении которого произошло исключение в некоторой программной единице.
3. Идентифицировать программную единицу по номеру POU, выведенному в строке об исключении, в древовидном списке POU. Каждая программная единица в дереве отображена в формате **POU_NAME(POU_TYPE)(-1/pou_num/-1/debug_id date_time)**, где **POU_NAME** – имя программной единицы; **POU_TYPE** – тип программной единицы; **pou_num** – номер POU.
4. Выполнить команды **Project–Clean all** и **Project–Rebuild all**.
5. Перейти в подкаталог *Compile* каталога установки среды разработки CoDeSys (по умолчанию – *C:\Program Files\3S Software\CoDeSys V2.3* на 32-разрядных Windows, *C:\ProgramData\CoDeSys V2.3* на 64-разрядных Windows) и открыть в нем файл с расширением asm и именем, совпадающим с именем только что скомпилированного проекта.
6. В файле текстовым поиском по имени найти начало кода программной единицы, в которой произошло исключение, после чего по смещению **pou_offset** в сообщении об исключении определить точное место, где произошло исключение.

8.2.3. Исключение вне кода приложения CoDeSys

Если исключение произошло вне кода, сгенерированного CoDeSys, сообщение об исключении имеет один из следующих форматов:

```
<EXCPT_TYPE> exception at <addr>. Task:<task_name>
<EXCPT_TYPE> exception at <addr> outside user code
```

где:

EXCPT_TYPE – тип исключения в соответствии с табл. 10;

addr – абсолютный адрес в сегменте кода;

task_name – имя задачи приложения или системы исполнения, в которой произошло исключение.

При возникновении такого исключения следует переслать информацию о данном инциденте по адресу службы технической поддержки Fastwel (см. п. 4.2.1.1).

8.2.4. Исключения в коде пользовательской DLL расширения системы исполнения

Информация об исключениях в коде DLL расширения системы исполнения приведена в п. 7.3.4 настоящего руководства.

8.3. Сообщения о дефиците системных ресурсов

Перечень сообщений о дефиците системных ресурсов приведен в табл. 11

Таблица 11

Обозначение	Причина
Failed to allocate RPC input buffer	Недостаточное количество ресурсов операционной системы при первоначальном запуске системы исполнения. Вероятно, наряду с системой исполнения на данном вычислительном устройстве запущено некоторое приложение, потребляющее чрезмерное количество системных ресурсов. Рассмотрите возможность функционирования системы без дополнительных приложений либо устраните в нем утечку памяти или других ресурсов.
Failed to allocate RPC output buffer	
Failed to init temporary buffer	
Failed to init PLC:	
Failed to init DXS2	
Failed to init Tasks Manager	
Failed to init User DLL Manager	
Commons initialization failure for modbus masters	
Modbus Servers Commons initialization failure	
Failed to initialize ModbusServer	
Code initialization failed:	Возникла ошибка при инициализации сегмента кода приложения CoDeSys. Справа от двоеточия отображается символьный код уточняющей информации. Перечень символьных кодов приведен в табл. 12.
Tasks configuring failed:	Возникла ошибка при конфигурировании задач приложения CoDeSys. Справа от двоеточия отображается символьный код уточняющей информации. Перечень символьных кодов приведен в табл. 12.
Tasks linking failed	Ошибка связывания задач приложения CoDeSys с образом процесса по одной из следующих причин: 1. Среда разработки CoDeSys сформировала для некоторой задачи ссылку на несуществующий участок образа процесса. 2. Для ациклической задачи задана несуществующая переменная чувствительности. В среде разработки CoDeSys откройте окно ресурса **PLC Configuration** и выполните команду меню **Extras–Calculate Addresses**, затем **Project–Clean all** и **Project Rebuild all**, после чего вновь загрузите приложение в контроллер.
Failed to update the call set for task <task_num>	Пользовательская задача с номером *task_num* (порядковый номер в списке задач в окне ресурса **Task Configuration**) содержит POU, в котором имеется слишком много вложенных вызовов других POU. Попробуйте упростить структуру вызовов либо обратитесь в службу поддержки Fastwel.
Failed to exchange storages	Не удалось выполнить перестановку вторичного хранилища приложения с первичным. Вероятно, повреждена файловая система основного дискового накопителя контроллера. Обратитесь в службу поддержки Fastwel.
Failed to initialize FBUS	Не удалось выполнить инициализацию сервиса ввода-вывода. По всей видимости, поврежден или отсутствует файл драйвера шины FBUS *ce_fbus.dll*. Восстановите его, скопировав на диск контроллера файл обновления системного ПО *norm.dnl* и повторно запустив контроллер.
Failed to link groups	Недостаточное количество системных ресурсов для связывания коммуникационных объектов сервиса FBUS с образом процесса. Рассмотрите возможность функционирования системы без дополнительных приложений либо устраните в нем утечку памяти или других ресурсов.

8.4. Зацикливания в пользовательском коде

8.4.1. Зацикливание в циклической задаче

Информация о зацикливании в циклической задаче представляется сообщением:

```
System overrun: <task_name>
```

где **task_name** – имя циклической задачи, в которой случилось зацикливание пользовательского кода.

8.4.2. Зацикливание в сервисной задаче

После зацикливания пользовательского кода в обработчике системного события, ациклической задаче или в функции интерфейса DLL расширения системы исполнения файл *normdump.txt* будет содержать строку:

```
CoDeSys2.SERVICE_TASK_STALLED
```

8.5. Ошибки конфигурации приложения

Символьные коды ошибок подсистемы исполнения кода CoDeSys перечислены в табл. 12. Каждый символьный код ошибки подсистемы исполнения кода CoDeSys предваряется префиксом **`CoDeSys2.`**. Некоторые из перечисленных кодов являются уточняющими для сообщения об ошибке дефицита системных ресурсов.

Таблица 12

Обозначение	Причина
Ошибки конфигурирования задач	
FAILED_TO_CREATE_TASK	Не удалось создать поток операционной системы. См. сообщение Failed to init PLC в табл. 11.
FAILED_TO_CONFIGURE_TASK	Не удалось сконфигурировать задачу приложения. Среда разработки ассоциировала с задачей несуществующий корневой программной единицей либо причина аналогична описанной для сообщения Failed to update the call set for task <task_num> в табл. 11. В первом случае обратитесь в службу поддержки Fastwel.
FAILED_TO_LINK_TASK	См. сообщение Tasks linking failed в табл. 11.
REINIT_TASKS_FAILURE	Одна из задач содержит слишком много ссылок на образ процесса.
Ошибки управления хранилищем проектной информации приложения	
CODE_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию кода. Попытайтесь повторно загрузить приложение.
CONFIG_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию конфигурации. Попытайтесь повторно загрузить приложение.
TASKS_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию конфигурации задач. Попытайтесь повторно загрузить приложение.
PRJINFO_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию информации о проекте. Попытайтесь повторно загрузить приложение.
IODESC_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию информации о ссылках задач на образ процесса. Попытайтесь повторно загрузить приложение.
STORAGE_FAILURE	После загрузки нового приложения в одной из секций обнаружена ошибка контрольной суммы. Попытайтесь повторно загрузить приложение.
REPLACE_STORAGES_FAILURE	Не удалось выполнить перестановку вторичного хранилища приложения с первичным. Попытайтесь повторно загрузить приложение. В случае повторной ошибки обратитесь в службу поддержки Fastwel.
LOAD_PRJINFO_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции информации о проекте. Попытайтесь повторно загрузить приложение.
LOAD_PRJINFO_FAILURE	Ошибка контрольной суммы при чтении секции информации о проекте. Попытайтесь повторно загрузить приложение.
LOAD_CODE_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции кода. Попытайтесь повторно загрузить приложение.
LOAD_CODE_FAILURE	Ошибка контрольной суммы при чтении секции кода. Попытайтесь повторно загрузить приложение.
NOT_ENOUGH_MEMORY_FOR_CODE	Размер секции кода превышает предельно допустимое значение для данной системы исполнения. Обратитесь в службу поддержки Fastwel.
CODE_REINIT_FAILURE	Ошибка инициализации сегмента кода. Обратитесь в службу поддержки Fastwel.
LOAD_CONFIG_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции конфигурации приложения. Попытайтесь повторно загрузить приложение.
LOAD_CONFIG_FAILURE	Ошибка контрольной суммы при чтении секции конфигурации приложения. Попытайтесь повторно загрузить приложение.
LOAD_TASKS_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции конфигурации задач. Попытайтесь повторно загрузить приложение.
LOAD_TASKS_FAILURE	Ошибка контрольной суммы при чтении секции конфигурации задач. Попытайтесь повторно загрузить приложение.
LOAD_IODESC_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции информации о ссылках задач на образ процесса. Попытайтесь повторно загрузить приложение.
LOAD_IODESC_FAILURE	Ошибка контрольной суммы при чтении секции информации о ссылках задач на образ процесса. Попытайтесь повторно загрузить приложение.
REINIT_IODESC_FAILURE	Одна из задач содержит слишком много ссылок на образ процесса.
PARSE_CONFIG_FAILURE	Не хватило памяти для загрузки дерева конфигурации приложения, либо структура дерева конфигурации не соответствует ожидаемой.
INVALID_CONFIG_FORMAT	В конфигурации приложения не найден требуемый элемент либо обнаружен элемент, не поддерживаемый данной системой исполнения.

Обозначение	Причина
INVALID_CONTROLLER_ID	В контроллер загружена конфигурация приложения, не соответствующая данной системе исполнения (типу контроллера).
Ошибки управления образом процесса	
PROCESS_IMAGE_REINIT_FAILURE	При загрузке нового приложения не удалось увеличить размер образа процесса. Попытайтесь повторно загрузить приложение.
FAILED_TO_INIT_DATA_MANAGER	См. сообщение Failed to init PLC: в табл. 11.
Ошибки инициализации буферов, сегментов кода и данных приложения	
BUFFERS_INITIALIZATION_FAILURE	Не хватило памяти для служебных буферов Online-сервисов взаимодействия со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
INVALID_RETAIN_DATA_SIZE	Не хватило системных ресурсов для управления энергонезависимыми переменными. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_RECEIVE_BUFFER	Не хватило памяти для буфера приема, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_RETAIN_SEGMENT	Не хватило памяти для создания сегмента энергонезависимых переменных. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_SEND_BUFFER	Не хватило памяти для буфера передачи, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_VARLIST_BUFFER	Не хватило памяти для буфера чтения/записи списков переменных, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_FORCELIST_BUFFER	Не хватило памяти для буфера форсирования значений переменных, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_TEMP_BUFFER	Не хватило памяти для вспомогательного буфера, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_RPC_BUFFER	Не хватило памяти для буфера сервиса вызова удаленных процедур, реализующего взаимодействие со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_RESOURCES_FOR_RPC	Не хватило системных ресурсов для сервиса вызова удаленных процедур, реализующего взаимодействие со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_PRIMARY_CODESEG	Не хватило памяти для сегмента кода. См. сообщение Failed to init PLC: в табл. 11.
NOT_ENOUGH_MEMORY_DATASEG	Не хватило памяти для глобального сегмента данных. См. сообщение Failed to init PLC: в табл. 11.
Ошибки загрузки и связывания с библиотечными функциями	
CODE_INIT_FAILURE	Не хватило системных ресурсов для управления библиотекой Standard.lib.
GLOBAL_INIT_RESOLUTION_FAILURE	Не удалось найти функцию инициализации глобальных данных в сегменте кода. Обратитесь в службу поддержки Fastwel.
GLOBAL_INIT_FAILED	После вызова функции инициализации глобальных, обнаруженной в сегменте кода, обнаружено повреждение памяти. Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
INVALID_DATA_SIZE	Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
INVALID_CODE_SIZE	Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
RETAIN_SIZE_EXCEEDED	Загружено приложение, у которого превышен допустимый размер сегмента энергонезависимых переменных. Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
INVALID_PROGRAM_CHECKSUM	Неправильная контрольная сумма загруженного приложения. В среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
DATA_RELOCATION_FAILURE	Не удалось выполнить релокацию ссылок на данные в коде приложения. В среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
DYNAMIC_LINKAGE_FAILURE	Не удалось найти функцию внешней библиотеки. В коде приложения имеется вызов библиотечной функции, которая не поддерживается системой исполнения. Информация о поддерживаемых внешних библиотеках приведена в разделе 6 настоящего руководства.

ПРИЛОЖЕНИЕ 1. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Версия	Дата	Ссылка	Статус	Примечания
2.52.23926	21.03.2011	Документ	создан	
2.59.23937	06.12.2012	п. 3.6–3.7	изменен	Добавлена важная информация о процессе обновления системного ПО контроллеров.
		п. 4.3.5	изменен	Исключена информация о диагностическом канале модулей ввода-вывода.
2.60.23938	20.03.2013	п. 3.6–3.7	изменен	Уточнена процедура обновления системного ПО контроллеров из среды разработки CoDeSys 2.3.
		п. 4.2.4.1	изменен	Исключена фраза «Исполнение программных единиц единственной циклической задачи, добавленной в ресурс Tasks Configuration» из перечня функций, выполняемых сервисной задачей.
2.61.23940	21.06.2013	п. 2.6.1	изменен	Добавлена информация об использовании CAN-адаптеров IXXAT при работе с контроллером CPM711
		п. 4.2.2.2	изменен	Добавлена информация об алгоритме управления энергонезависимыми (RETAIN) переменными при запуске контроллера после включения питания или сброса.
		п. 4.2.3	изменен	Добавлена информация об алгоритме управления энергонезависимыми (RETAIN) переменными при загрузке и обновлении приложения в контроллере из среды разработки CoDeSys.
2.62.23943	23.12.2013	п. 6.1	изменен	Добавлены общие сведения о системной библиотеке FastwelGPS.lib
		п. 6.8	создан	Добавлено описание системной библиотеки FastwelGPS.lib
2.63.23944	07.08.2014	Раздел 1	изменен	Скорректированы ссылки на документы
		п. 0	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
		п. 6.8.1	изменен	Исправлено название входа NIM742 для ввода сигнала 1PPS GPS-приемника
2.64.23946	27.01.2015	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. 3.7	изменен	Добавлена информации о необходимости повторной установки обработчиков системных событий при переносе проектов с платформы CPM70x на CPM71x.

Версия	Дата	Ссылка	Статус	Примечания
		п. 6.1	изменен	Добавлены общие сведения о системных библиотеках FastwelHayesModem.lib, FastwelHayesModemControls.lib, FastwelSMS.lib, FastwelSMSControls.lib, FastwelModbusRTUClientSerial.lib, FastwelPlatformControl.lib.
		п. 6.9	создан	Добавлено описание системной библиотеки FastwelHayesModem.lib.
		п. 6.10	создан	Добавлено описание системной библиотеки FastwelHayesModemControls.lib.
		п. 6.11	создан	Добавлено описание системной библиотеки FastwelSMS.lib.
		п. 6.12	создан	Добавлено описание системной библиотеки FastwelSMSControls.lib.
		п. 6.13	создан	Добавлено описание системной библиотеки FastwelModbusRTUClientSerial.lib.
		п. 6.14	создан	Добавлено описание системной библиотеки FastwelPlatformControl.lib.
		п. 7.3.1	изменен	Уточнен процесс подготовки архива с DLL к загрузке в контроллер
2.65.23947	25.06.20015	п. 4.2.1.1	изменен	Добавлена информация об установлении причины перехода контроллера в безопасный режим при помощи браузера ПЛК.
		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	14.10.2015	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	
		п. 4.3.2	изменен	Исправлена ошибка в описании алгоритма сервиса ввода-вывода при запуске, если на шине обнаружены не все ожидаемые модули ввода-вывода.
		п. 4.3.4.1	изменен	
		п. 4.3.5.2	изменен	Уточнен период обновления диагностических каналов сервиса ввода-вывода.
		п. 6.4.2.1	изменен	Отражено изменение функциональных возможностей библиотеки FastwelSysLibCom. Максимальное число одновременно открытых устройств увеличено до 32.
		документ	изменен	Скорректирована информация об изготовителе
2.67.23949	20.01.2016	п. 2.4.1	изменен	Добавлена информация о поддержке сохраняемых (PERSISTENT) переменных.
		п. 4.2.3	изменен	Разделен на пп. 4.2.3.1, 4.2.3.2, 4.2.3.3. Добавлена информация о поддержке механизма горячего обновления (ONLINE CHANGE) приложения в контроллере.
		п. 4.2.4.1	изменен	Из описания функций, возлагаемых на сервисную задачу, исключено облуживание коммуникаций со средой разработки CoDeSys 2.3 и другими клиентскими приложениями CoDeSys 2.3 Gateway Server.
		п. 4.2.4.3	изменен	Скорректирован рис. 15, исключено обслуживание коммуникаций со средой разработки CoDeSys 2.3 и другими клиентскими приложениями CoDeSys 2.3 Gateway Server.
		п. 4.2.4.4	изменен	В описание обработчика системного события OnProgramChange добавлена информация о его вызове при горячем обновлении приложения.
		п. 4.2.4.5	изменен	Добавлена информация об особенностях применения библиотеки FastwelTasksExchange.lib для организации межзадачного обмена при использовании механизма горячего обновления (ONLINE CHANGE) приложения в контроллере.
		п. 4.3	изменен	Переработано описание процедур инициализации шины FBUS, обнаружения отказов и диагностики в связи с изменениями в системного ПО контроллера, начиная с версии 2.67.
		п. 6.5.3.2	изменен	Добавлена информация об особенностях применения библиотеки FastwelModbusServer.lib при использовании механизма горячего обновления (ONLINE CHANGE) приложения в контроллере.

Версия	Дата	Ссылка	Статус	Примечания
2.68.23950	30.05.2016	п. 4.3.4.2	изменен	Добавлены указания по восстановлению работоспособности линейки заведомо исправных модулей после сборки и разборки, если контроллер индицирует частично исправное состояние межмодульной шины.
2.71.23953	31.08.2017	п. 8.2.2	изменен	Уточнена информация об идентификации программной единицы, в которой произошло исключение.