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

Методические указания по разработке приложений

ИМЕС.00300-02 33 03-5

Версия 2.0

СОДЕРЖАНИЕ

Контактная информация

Адрес: 119313, Москва, а/я 242;

Телефон: (095) 234–0639;

Факс: (095) 232–1654;

E-mail: info@fastwel.ru;

Web: www.fastwel.ru.

Поставка и техническая поддержка

ПРОСОФТ® осуществляет поставку и техническую поддержку продукции Фаствел®.

Адрес: 119313, Москва, а/я 81;

Телефон: (095) 234–0636;

Факс: (095) 234–0640;

E-mail: info@prosoft.ru;

Web: www.prosoft.ru.

Фаствел® приветствует любые предложения и замечания по улучшению данного руководства по эксплуатации, а также объективную информацию о функционировании представленного изделия и встроенного системного программного обеспечения.

Настоящий документ содержит указания по настройке и программированию контроллеров CPM711, CPM712 и CPM713 комплекса Fastwel I/O в среде CoDeSys.

Фаствел® является официальным OEM-партнером фирмы 3S Smart Software Solutions и производителем адаптации среды CoDeSys для работы совместно с устройствами серии Fastwel I/O.

ПРОСОФТ® является официальным дистрибьютором Фаствел®.

Каталог продукции Фаствел® размещен на Web-странице:

http://www.fastwel.ru/products/catalog/index.htm.

Каталог продукции Фаствел® размещен также на файл-сервере ПРОСОФТ® по адресу:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/.

1. ВВЕДЕНИЕ

Настоящий документ содержит информацию, достаточную для ознакомления с основными возможностями адаптированной среды CoDeSys применительно к конфигурированию и программированию контроллеров CPM711, CPM712 и CPM713 серии Fastwel I/O в среде CoDeSys фирмы 3S Smart Software Solutions.

Подробная информация о принципах функционирования, указания по настройке и программированию контроллеров CPM711, CPM712 и CPM713 серии Fastwel I/O в среде CoDeSys фирмы 3S Smart Software Solutions приведена в документе:

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

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

1. *CPM711. Контроллер узла сети CANopen. Руководство по конфигурирированию и программированию сетевых средств*.
2. *CPM712. Контроллер узла сети MODBUS RTU/ASCII. Руководство по конфигурирированию и программированию сетевых средств*.
3. *CPM713. Контроллер узла сети MODBUS TCP. Руководство по конфигурирированию и программированию сетевых средств*.

Информация о конфигурировании модулей ввода-вывода Fastwel I/O приведена в документе *Модули ввода-вывода Fastwel I/O. Руководство программиста*.

Предполагается, что пользователь среды CoDeSys, адаптированной для программирования контроллеров серии Fastwel I/O, имеет навыки программирования на языках стандарта IEC 61131-3 и знаком 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 технологическими процессами. 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:

1. Контроллеры узла сети
2. Модули ввода-вывода
3. Вспомогательные модули

Контроллер узла сети является вычислительным устройством на базе микропроцессора Vortex86DX фирмы DMP Electronics, совместимого с 80486 и имеющего тактовую частоту 600 МГц.

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

Интерфейс внешней сети контроллера узла 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 узла сети, предназначены для организации связи контроллера с датчиками и исполнительными механизмами объекта управления.

2.3. Структура программного обеспечения Fastwel I/O

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

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

2.4. Назначение среды разработки 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, перечень описаний сообщений (коммуникационных объектов), поступающих из внешней сети и выдаваемых в сеть контроллером, 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 прикладных алгоритмов обработки данных и управления на языках ST, IL, LD, FBD, SFC стандарта IEC 61131-3 и трансляцию разработанного приложения в исполняемый код процессора.
4. Отладку разработанного приложения в режиме эмуляции.
5. Загрузку приложения в контроллер.
6. Удаленную отладку и управление исполнением приложения в контроллере.

2.5. 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 для контроллеров CPM71x серии Fastwel I/O, должно состоять хотя бы из одной программы (в терминах IEC 61131-3), 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.

Таким образом, задачи являются системными ресурсами контроллера, позволяющими распределить процессорное время между программами, входящими в состав приложения.

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 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.

2.6. Цель обучения

После ознакомления с настоящим документом, пользователь приобретет понимание основ программной модели контроллера, научится разрабатывать небольшие программы для контроллеров Fastwel I/O в среде CoDeSys и создавать конфигурацию модулей ввода-вывода и коммуникационных объектов внешней сети.

Более подробная информация об использовании среды CoDeSys для разработки приложения приведена в документации CoDeSys:

1. *Руководство пользователя по программированию ПЛК в CoDeSys 2.3* (CoDeSys_V23_RU.pdf)
2. *Первые шаги с CoDeSys* (First Steps with CoDeSys RU.pdf).

После установки комплекта адаптации CoDeSys для Fastwel I/O указанные руководства находятся в подкаталоге *\Doc\CoDeSys\Russian* каталога установки адаптации (по умолчанию – *C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation*).

2.7. Системные требования к рабочему месту разработчика

2.7.1. 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 узла сети CPM711 по сети CAN компьютер должен быть оснащен адаптером сети CAN фирмы IXXAT (любым) либо адаптером PCAN-USB фирмы PEAK-Systems Technik.

Использование адаптера фирмы IXXAT возможно только под управлением операционных систем Windows 2000 или Windows XP, при этом на компьютер необходимо установить пакет программной поддержки VCI 2.16 + Service Pack 2, 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 узла сети CPM712 компьютер должен иметь в своем составе последовательный порт интерфейса RS-232C или RS-485.

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

2.7.2. Требования к системному программному обеспечению

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

1. операционная система Windows 2000 Professional SP4, Windows XP SP3, Windows 7 Professional;
2. для чтения документации в формате Adobe PDF требуется установить программу Adobe Acrobat Reader версии не ниже 6.0.

Указания по установке адаптированной среды разработки CoDeSys приведены в разделе 3 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.

3. МЕТОДИКА РАЗРАБОТКИ ПРИЛОЖЕНИЙ

3.1. Общие сведения

Процесс разработки проекта приложения в среде CoDeSys состоит из следующих операций:

1. Создание проекта CoDeSys для платформы, соответствующей используемому типу контроллера.
2. Создание конфигурации модулей ввода-вывода контроллера.
3. Создание конфигурации внешней сети.
4. Разработка программных единиц (POU), реализующих требуемые прикладные алгоритмы, включая *программы*, *функциональные блоки* и *функции*.
5. Создание *циклических* и *ациклических задач*, под управлением которых будут исполняться *программы*.
6. При необходимости, создание *функций обработки системных событий*, происходящих в контроллере.
7. Трансляция прикладной программы.
8. Загрузка прикладной программы в контроллер.
9. Мониторинг переменных и отладка прикладной программы в контроллере.
10. Трассировка переменных.

В данном разделе вкратце рассматриваются перечисленные операции на примере небольшого учебного проекта.

Для выполнения приведенных ниже указаний требуется иметь следующие оборудование и принадлежности:

1. Контроллер Fastwel I/O CPM71x (CPM711/712/713).
2. Кабель соединительный ACS00019.
3. Блок питания с выходным напряжением 24 В постоянного тока.
4. Модуль аналогового ввода (AIM729 или аналогичный).
5. Модуль дискретного вывода (DIM712 или аналогичный).
6. ПК, имеющий, по крайней мере, один порт интерфейса RS-232C. На ПК должна быть установлена адаптированная среда разработки CoDeSys.
   Для интеграции с программным обеспечением верхнего уровня на ПК может быть установлен OPC-сервер промышленной сети, поддерживаемой имеющимся типом контроллера (например, Fastwel Modbus OPC Server).
   Для загрузки приложения в контроллер и для взаимодействия между средой разработки CoDeSys и контроллером по сети Ethernet ПК должен быть оснащен сетевым адаптером Ethernet, подключенным к сетевому концентратору кабелем 5-й категории.
   Для загрузки приложения в контроллер и для взаимодействия между средой разработки CoDeSys и контроллером по сети CAN к ПК должен быть подключен адаптер сети CAN фирмы IXXAT или PCAN-USB фирмы PEAK-Systems Technik.

3.2. Подготовка к работе

3.2.1. Конфигурация аппаратных средств

Схема соединений контроллера, для которого будет создаваться учебный проект, представлена на рис. 1. При использовании контроллера, отличного от CPM713, сетевой концентратор Ethernet должен быть заменен на адаптер соответствующей сети.

1. Выполните соединения в соответствии с рис. 1. Перед присоединением вилки MiniUSB кабеля ACS00019 к порту консоли контроллера узла при помощи отвертки осторожно извлеките пластмассовую крышку, закрывающую порт консоли.

2. Включите блок питания. Индикатор RUN/ERR контроллера будет светиться прерывисто, поочередно меняя цвет с зеленого на красный и прекращая свечение, что свидетельствует об отсутствии в контроллере приложения пользователя.

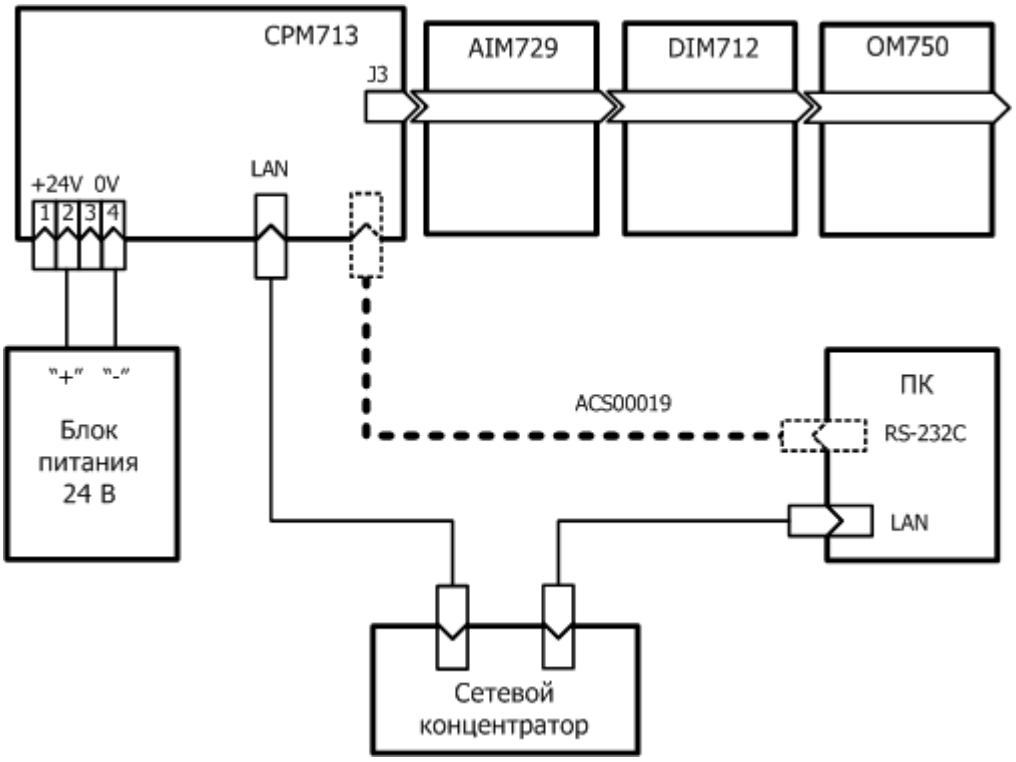

Рис. 1. Схема соединений контроллера для реализации учебного проекта

3.2.2. Цель работы

В результате работы над проектом будет создана прикладная программа для контроллера со следующими функциями:

1. Чтение каналов модуля аналогового ввода.
2. Преобразование шкалы считанных значений на каналах модулей аналогового ввода в напряжение в формате с плавающей точкой.
3. Периодическое включение и отключение первого канала модуля дискретного вывода.
4. Передача преобразованных значений на каналах модуля аналогового ввода по сети другому узлу.
5. Включение/выключение второго канала модуля дискретного вывода по командам, поступающим в контроллер по сети.
6. Включение и выключение индикатора USER на передней панели контроллера по команде, поступающей по сети.
7. Прерывистое свечение индикатора USER красным цветом при отсутствии связи по сети.

3.3. Создание проекта

3.3.1. Составные части проекта CoDeSys

Проект CoDeSys состоит из набора описаний информационных и функциональных элементов приложения (прикладной программы) пользователя, которые, после выполнения трансляции проекта по команде меню **Project–Build All**, могут быть загружены в контроллер командой **Online–Login.**

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 его 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 список **POUs**, показанный на рис. 2, по команде контекстного меню **Add Object**.

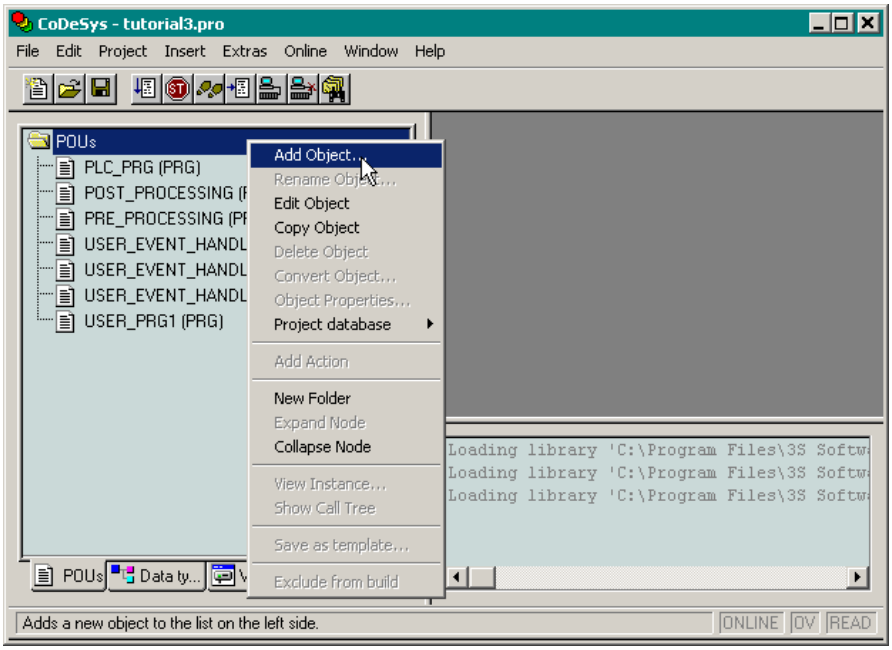

Рис. 2. Добавление программной единицы в древовидный список POU

2. Множество циклических задач. Циклической задачей является вычислительный процесс типа *Cyclic Task*, описываемый пользователем в окне ресурса проекта CoDeSys **Tasks Configuration**. 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 (см. рис. 3) с периодом задачи в порядке следования в списке POU данной задачи, как показано на рис. 4.
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. 3. Исполнение циклической задачи

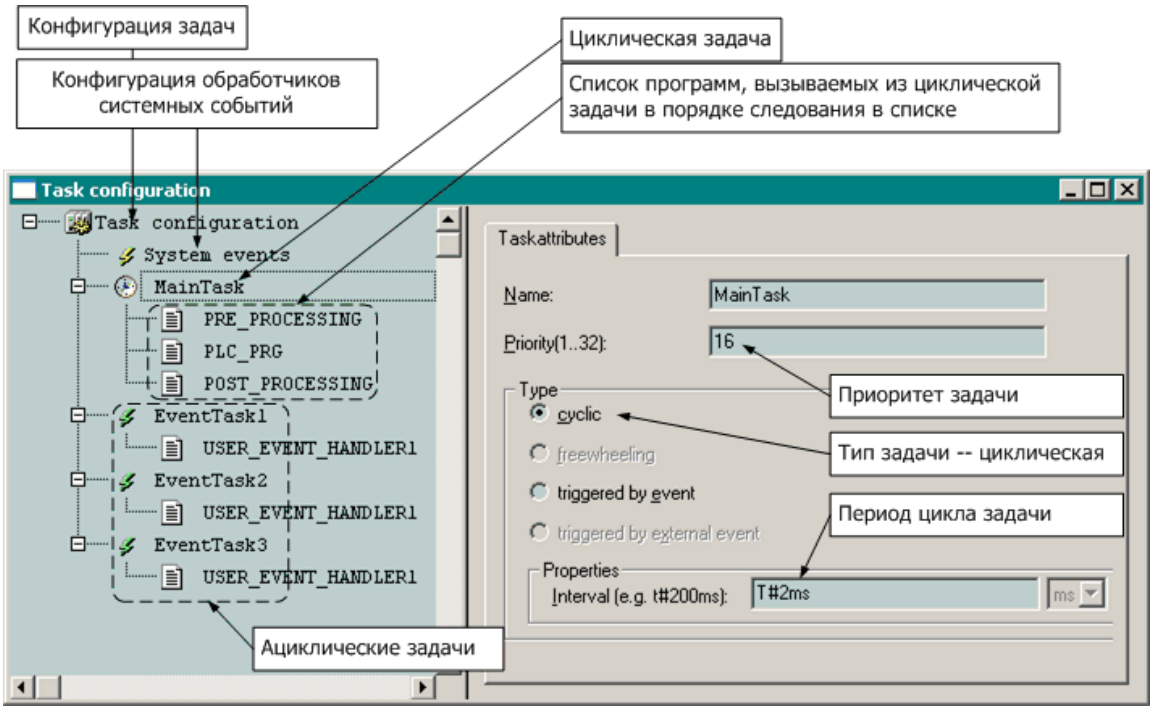

Рис. 4. Пример конфигурации задач приложения пользователя

3. Множество ациклических задач. Ациклической задачей является вычислительный процесс типа *Event Task (triggered by event)*, описываемый пользователем в окне ресурса проекта CoDeSys **Tasks Configuration**. С ациклической задачей может быть ассоциирована одна или несколько программ – программных единиц IEC 61131-3 типа *PROGRAM*.
   Программы, ассоциированные с некоторой ациклической задачей, запускаются на контексте высокоприоритетного потока исполнения операционной системы 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 высокоприоритетного сервисного потока операционной системы контроллера 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). Исполнение ациклической задачи схематически иллюстрируется рис. 5.

Примечание. Если в окно ресурса **Tasks Configuration** не добавлено ни одной задачи любого из поддерживаемых типов, среда разработки CoDeSys автоматически создаст описание одной 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 ... 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.

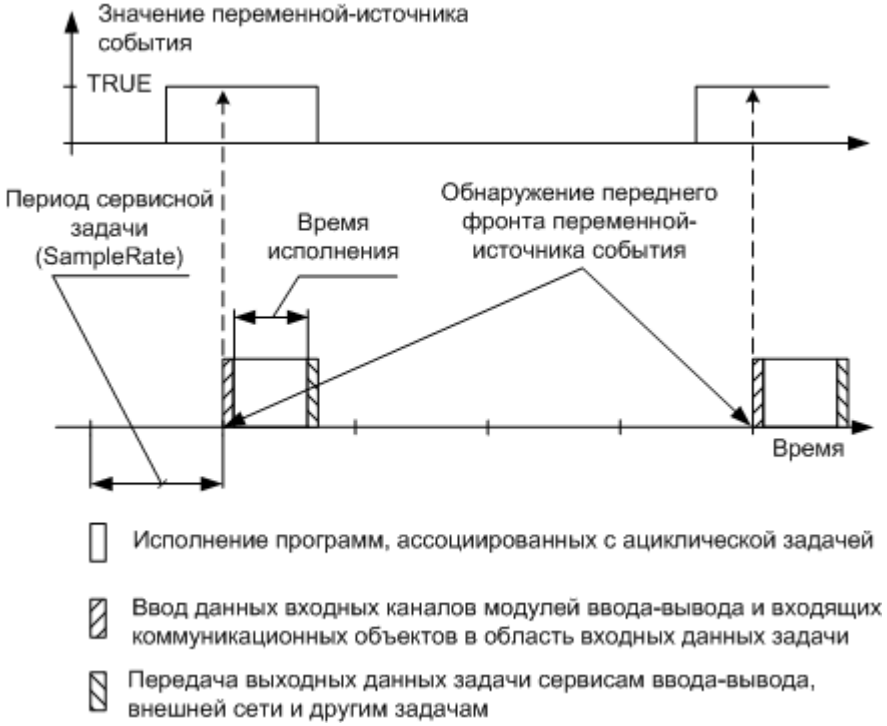

Рис. 5. Исполнение ациклической задачи

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.

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. 6. 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 из приложения.

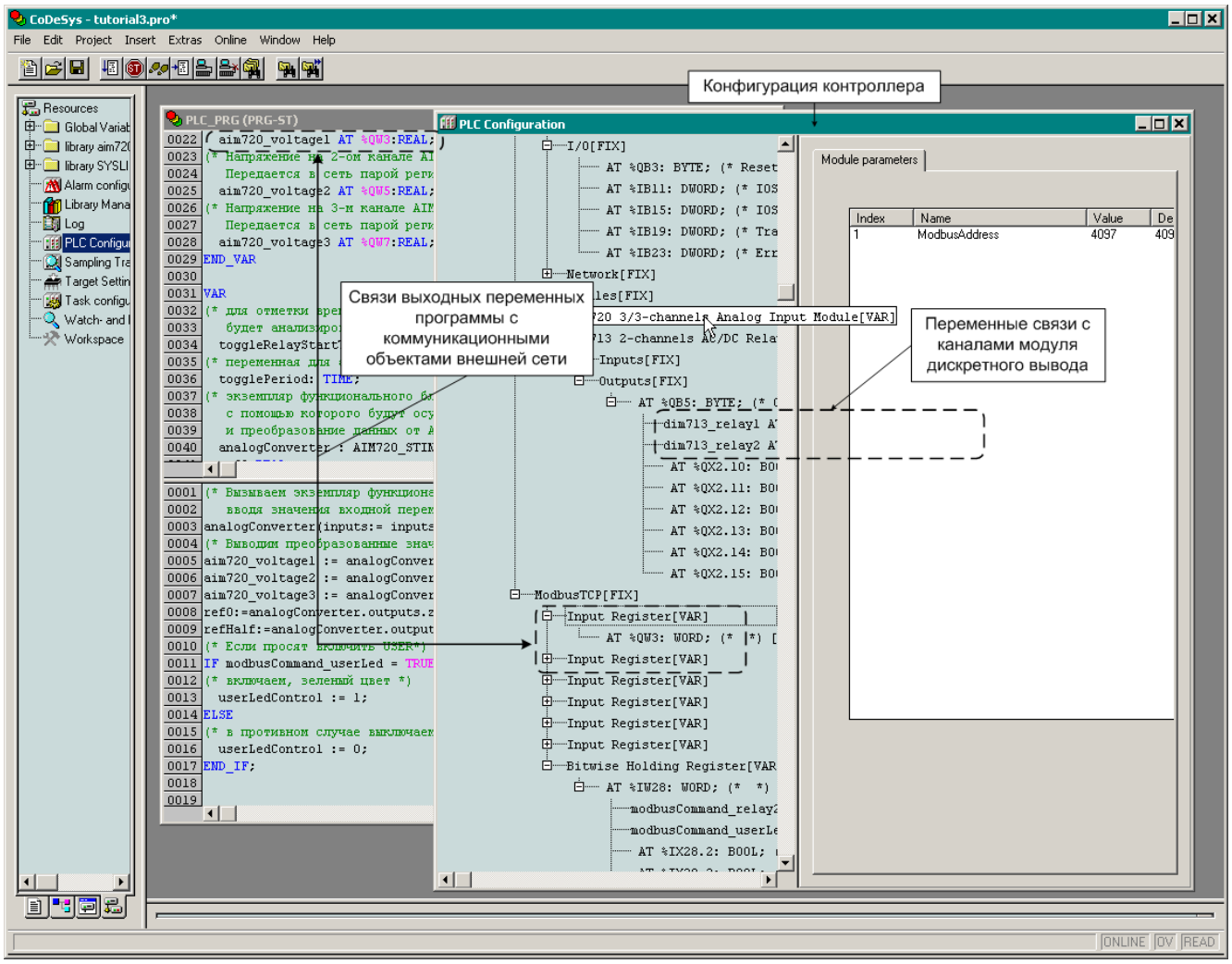

Рис. 6. Конфигурация контроллера

3.3.2. Создание и сохранение проекта для контроллеров Fastwel I/O серии CPM71x

1. Запустите среду разработки CoDeSys.
2. В меню **File** выберите команду **New** и в появившейся диалоговой панели **Target Settings** выберите опцию, соответствующую типу используемого контроллера, в выпадающем списке **Configuration**, после чего нажмите кнопку **OK**. На экран будет выведена диалоговая панель **New POU**.

   При использовании контроллеров серии CPM71x должно соблюдаться следующее соответствие между типом контроллера и выбираемой опцией:

   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.
3. Нажмите кнопку **OK** в диалоговой панели **New POU**. В меню **File** выберите команду **Save**, введите имя файла проекта *CPM71x_tutorial* в появившейся диалоговой панели **Save As** и нажмите кнопку **Save**. Окно CoDeSys примет вид, показанный на рис. 7.
4. Щелкните на вкладке **POUs** и в области редактирования кода программы введите единственный пустой оператор **";"** (точка с запятой), который позволит немедленно транслировать (построить) проект командой **Project–Rebuild all**.

Древовидный список *POUs* в левой части главного окна CoDeSys содержит перечень программных единиц, имеющихся в проекте. Если в конфигурации задач, создаваемой в ресурсе **Tasks Configuration**, не создано ни одной задачи, то программа *PLC_PRG* будет являться главной (корневой) и будет вызываться в контроллере с периодом, который определяется параметром *CPM71x ... Programmable Controller:SampleRate* (по умолчанию – 10 мс) в окне ресурса **PLC Configuration**.

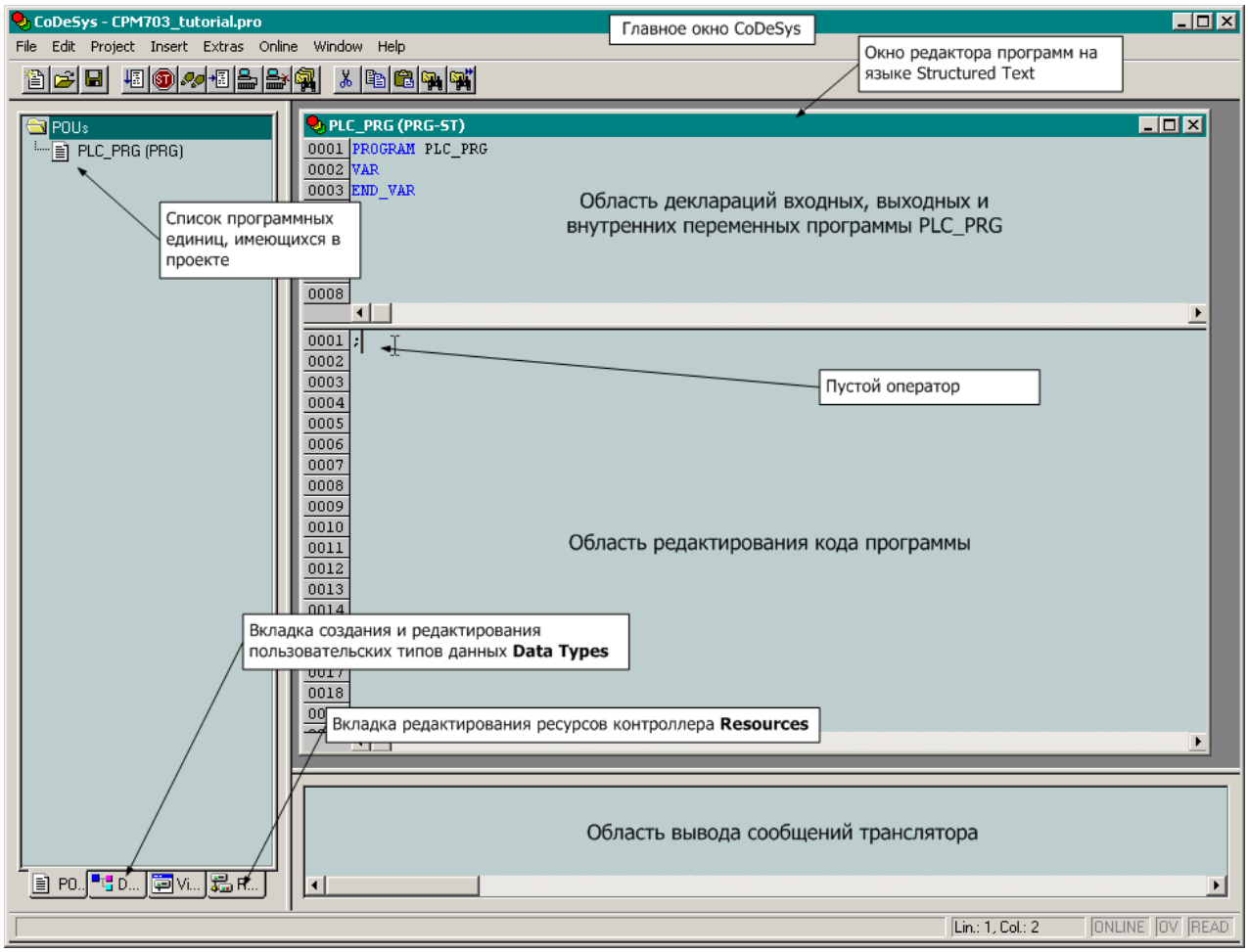

Рис. 7. Главное окно CoDeSys после создания проекта

Окно редактора программ на языке Structured Text (ST), разделенное на две области: деклараций переменных программы и кода программы, – показано на рис. 7 справа.

Таким образом, создан проект для платформы, соответствующей типу используемого контроллера, и имеется возможность приступить к созданию конфигурации задач, настройке параметров контроллера, созданию конфигурации модулей ввода-вывода и внешней сети.

3.4. Создание конфигурации задач

3.4.1. Общие сведения

Задача является ресурсом контроллера, который предоставляет *программам*, входящим в приложение, процессорное время.

Более подробные сведения о задачах приведены в п. 3.3.1 настоящего документа, а также в п. 4.2.4 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.

3.4.2. Создание и конфигурирование циклической задачи

В учебном проекте будет создана одна циклическая задача с периодом цикла 10 мс, из которой будет вызываться основная программа *PLC_PRG*. Для создания задачи:

1. Щелкните на вкладке **Resources** в главном окне CoDeSys, а затем дважды щелкните на ресурсе **Tasks Configuration** в древовидном списке ресурсов контроллера. На экране появится окно ресурса **Tasks Configuration**, показанное на рис. 8.
2. Щелкните правой кнопкой мыши над корневым элементом *Task configuration* древовидного списка задач и выберите команду **Append Task** в появившемся контекстном меню. В древовидном списке *Task configuration* появится элемент *NewTask*, а вкладка **Task Attributes** будет иметь вид, представленный на рис. 9.

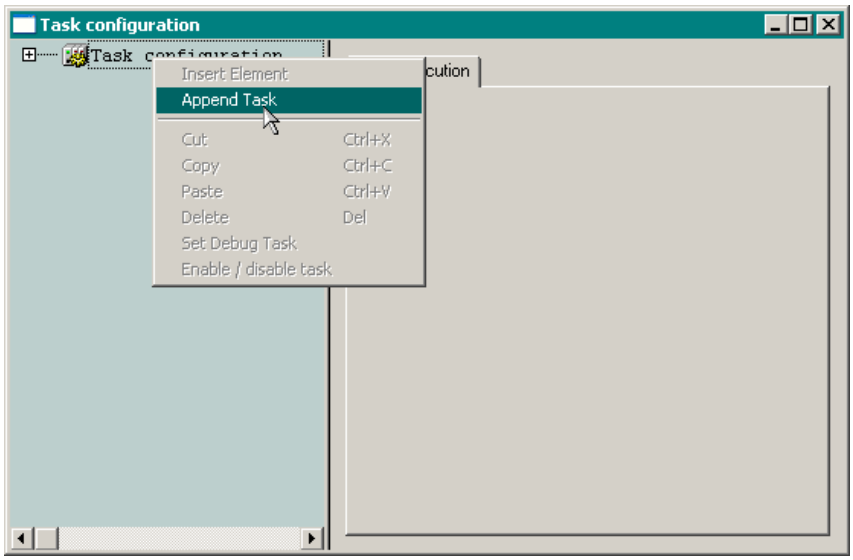

Рис. 8. Добавление задачи в окне ресурса Tasks Configuration

3. В поле **Name** вкладки **Task Attributes** введите имя задачи *MainTask*, а в поле **Properties:Interval (e.g. t#200ms)** – требуемый период цикла *T#10ms*, после чего щелкните правой кнопкой мыши над элементом *NewTask* в древовидном списке задач и выберите команду **Append Program Call** в появившемся контекстном меню. Имя элемента *NewTask* в древовидном списке изменится на *MainTask*, и под ней появится подчиненный элемент, представляющий вызов программы из данной задачи. При этом вкладка в правой части окна **Tasks Configuration** будет иметь вид, как на рис. 10.

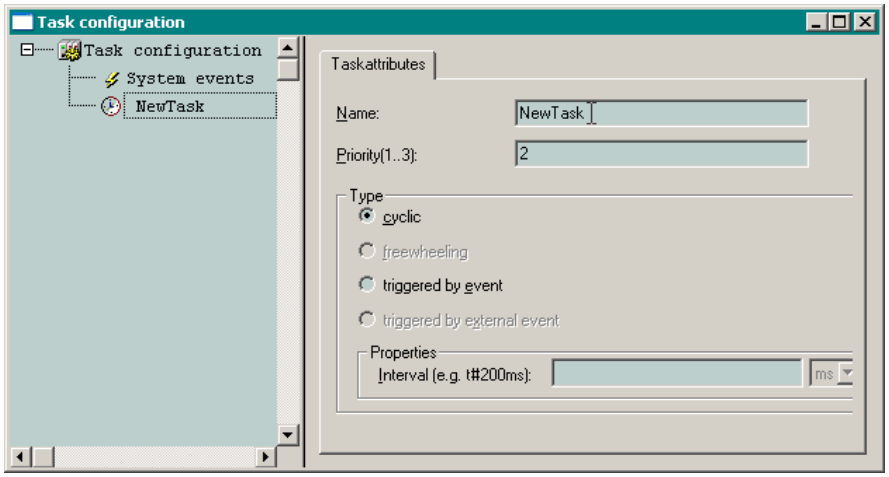

Рис. 9. Окно ресурса Tasks Configuration после добавления задачи

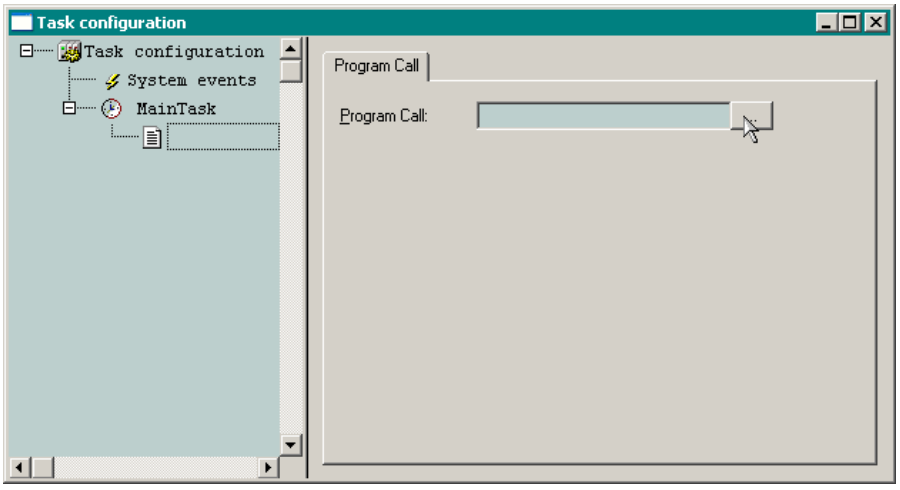

Рис. 10. Окно ресурса Tasks Configuration после выполнения команды Append Program Call в контекстном меню задачи

4. Введите имя программы *PLC_PRG* в поле **Program Call** или нажмите кнопку **...**, расположенную справа от поля **Program Call**, и выберите имя вызываемой программы в

диалоговой панели **Помощника ввода** (**Input assistant**), сняв флажки **With Arguments** (с аргументами) и **Structured** (структурированное представление), как показано на рис. 11. Имя выбранной программы появится в древовидном списке, как показано на рис. 12.

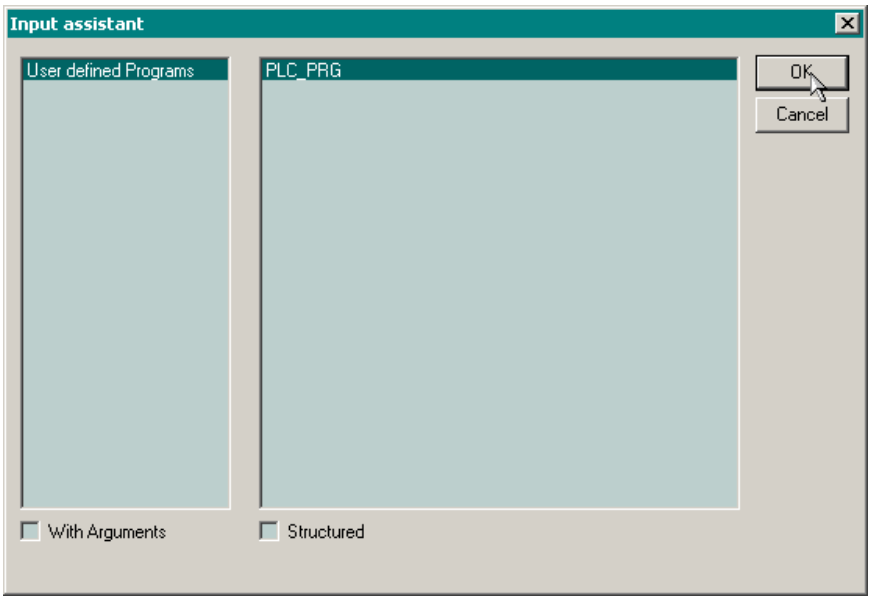

Рис. 11. Использование помощника ввода для выбора программы, вызываемой из задачи

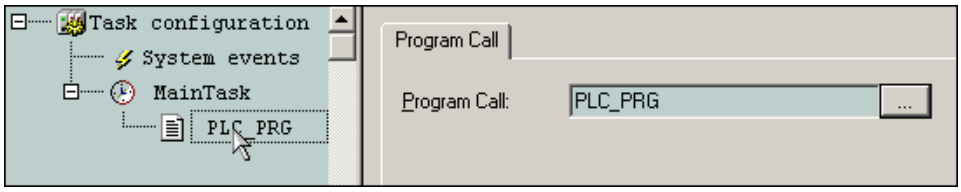

Рис. 12. Ассоциация программы с циклической задачей

Если в данный момент транслировать проект и загрузить его в контроллер, в контроллере будет циклически, с периодом 10 мс, вызываться программа *PLC_PRG*, выполняющая единственный ничего не делающий оператор "**;**".

3.5. Настройка параметров контроллера

3.5.1. Общие сведения

Настройка параметров контроллеров CPM71x серии Fastwel I/O выполняется в окне ресурса **PLC Configuration**. Контроллеры CPM71x имеют три параметра, значения которых могут быть заданы пользователем:

1. Параметр *Fastwel I/O System Configuration–Module Parameters: HotUpdateDisabled* (по умолчанию *Yes*) – служит для отключения (*Yes*) или включения (*No*) функции горячего обновления прикладной программы при отсутствии изменений в структурах данных и размерах входной и выходной областей образа процесса во вновь загруженном приложении по сравнению с текущим приложением. Более подробная информация о загрузке и обновлении приложения приведена в п. 4.2.3 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.
2. Параметр *CPM71x ... Programmable Controller: SampleRate* – позволяет задать период исполнения сервисной задачи среды исполнения контроллера (по умолчанию 10 мс), на контексте которой среда исполнения проверяет переменные-источники событий ациклических задач, вызываются ациклические задачи, обработчик системного события *OnTimer* и другие вспомогательные действия среды исполнения, включая обслуживание запросов среды разработки CoDeSys и т.д. Более подробная информация приведена в пп. 4.2.4.3–4.2.4.4 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.
3. Параметр *CPM71x ... Programmable Controller: Autostart* предназначен для отладки приложения. Если в контроллер загружено приложение, для которого параметр

CPM71x... Programmable 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

3.5.2. Установка периода сервисной задачи

Поскольку в учебном проекте отсутствуют ациклические задачи и не предполагается обрабатывать системное событие *OnTimer*, желательно увеличить период сервисной задачи таким образом, чтобы она потребляла как можно меньше процессорного времени. Однако при этом не следует делать значение периода слишком большим, чтобы не задерживать обслуживание запросов среды разработки к контроллеру, когда пользователь будет выполнять команду **Online–Login**, устанавливая связь с контроллером.

Установите значение периода сервисной задачи равным 50 мс, для чего:

1. Дважды щелкните на ресурсе *PLC Configuration* в дереве ресурсов CoDeSys. На экране появится окно ресурса **PLC Configuration**.
2. В левой области окна **PLC Configuration** выберите элемент дерева конфигурации *CPM71x ... Programmable Controller*, после чего дважды щелкните левой кнопкой мыши на ячейке *SampleRate:Value* во вкладке **Module parameters**, введите значение *50* и нажмите клавишу Enter.

3.6. Создание конфигурации модулей ввода-вывода

3.6.1. Общие сведения

Для организации обмена данными между приложением и модулями ввода-вывода требуется добавить описания модулей ввода-вывода в секцию *CPM71x... Programmable Controller–I/O Modules* ресурса **PLC Configuration** с точным соблюдением порядка физического подключения модулей к внутренней шине FBUS 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 во все выходные каналы и чтение данных всех входных каналов модулей ввода-вывода. Тем самым обеспечивается пропускная способность не хуже 165 кбайт/с. Для выбора группового режима обмена параметр *I/O Modules:ScanMode* должен иметь значение *Single Group*.

Кроме того, предусмотрен режим индивидуального обмена с модулями, при использовании которого для каждого модуля, добавленного в конфигурацию контроллера, создается отдельная группа ввода-вывода. Для выбора режима индивидуального обмена параметр должен иметь значение *Group per Module*. Режим индивидуального обмена с модулями может использоваться в ситуациях, когда неисправность отдельных модулей, подключенных к внутренней шине контроллера, не приводящая к нарушению электрической связи с остальными модулями, не должна приводить к прекращению обмена с остальными исправными модулями.

Период обмена с модулями ввода-вывода определяемую параметром *CPM71x... Programmable Controller–I/O Modules:SampleRate*. При использовании режима индивидуального обмена пропускная способность внутренней шины существенно ухудшается за счет появления пауз длительностью от 180 до 330 мкс между обменами с соседними модулями, а также за счет передачи в групповом ответе каждого модуля дополнительных пяти байт (идентификатора мастера и поля контрольной суммы). При использовании режима *Group per Module* сервис ввода-вывода потребляет существенно больше вычислительных ресурсов, чем в режиме *Single Group*. В результате производительность при исполнении программных единиц, вызываемых на контексте циклических задач может ухудшится на величину до 40-45%. В качестве эмпирического правила выбора значения периода обмена с модулями рекомендуется следующее соотношение: добавление одного модуля в конфигурацию контроллера должно сопровождаться увеличением периода обмена, как минимум, на 1 мс.

Более подробная информация о сервисе ввода-вывода встроенного программного обеспечения контроллеров приведена в п. 4.3 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.

3.6.2. Настройка параметров сервиса ввода-вывода и добавление описаний модулей ввода-вывода в конфигурацию контроллера

3.6.2.1. Цель работы

В результате описанных ниже действий:

1. Будет установлен режим индивидуального опроса модулей ввода-вывода с периодом 10 мс.
2. В конфигурацию контроллера будут добавлены описания модуля аналогового ввода AIM729 и модуля дискретного вывода DIM712.
3. Для модуля DIM712 будет задан интервал сторожевого таймера с мастером сети FBUS, равный примерно 3 с. Если с момента последнего обмена с модулем по внутренней шине пройдет 3 с, и модуль не получит ни одного сообщения по внутренней шине, – его каналы дискретного вывода будут установлены в безопасное состояние, определяемое параметрами **Состояние выходных каналов–Безопасное1/2**. В качестве безопасных будем считать включенное состояние первого канала модуля и выключенное – второго.
4. В приложении будет задано начальное состояние выходных каналов модуля DIM712, которое будет устанавливаться после включения питания контроллера и до запуска приложения. Задание начального состояния выходных каналов производится путем настройки соответствующих параметров модуля (**Состояние выходных каналов–Начальное1/2**) и установки начальных значений переменных, чьи значения будут передаваться в каналы во время работы приложения. В качестве начальных будем считать включенное состояние первого канала модуля и выключенное – второго.

3.6.2.2. Настройка параметров обмена и создание конфигурации модулей ввода-вывода

Пусть к внутренней шине контроллера подключены модуль аналогового ввода AIM729, а за ним – модуль релейной коммутации DIM712. Для настройки параметров и создания конфигурации сервиса ввода-вывода контроллера:

1. Щелкните на вкладке **Resources** в левой области главного окна CoDeSys. В появившемся древовидном списке *Resources* дважды щелкните на элементе *PLC Configuration*. В правой области главного окна CoDeSys появится окно **PLC Configuration**.
2. Раскройте корневой элемент *Fastwel I/O System Configuration* дерева в окне **PLC Configuration** и элемент *CPM71x... Programmable Controller*, после чего щелкните мышью на элементе *I/O Modules*.
3. Для установки режима индивидуального опроса модулей выберите соответствующую опцию для параметра **Режим обмена** сервиса ввода-вывода, как показано на рис. 13. Параметр **Период опроса, мс** определяет период (в миллисекундах), с которым будет выполняться информационный обмен с модулями ввода-вывода, добавленными в конфигурацию контроллера на следующих шагах.
   Поле **Время опроса, мкс** содержит значение интервала времени в микросекундах, в течение которого будет происходить информационный обмен с модулями в выбранном режиме обмена.
   Поле **Рекомендуемый период опроса, мкс** содержит рекомендуемое значение периода опроса модулей в миллисекундах в выбранном режиме обмена.

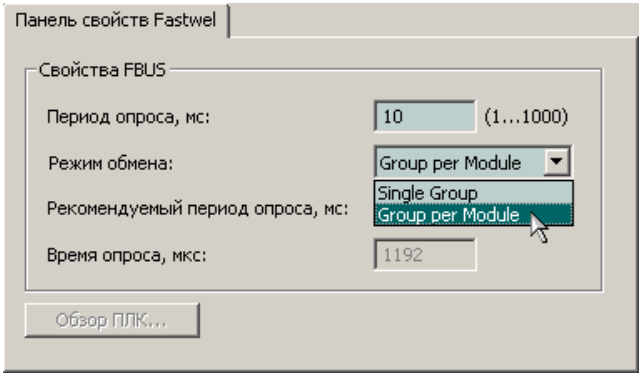

Рис. 13. Установка режима индивидуального опроса модулей ввода-вывода

4. Щелкните правой кнопкой мыши над элементом *I/O Modules* и выберите строку **Append Subelement–AIM729 2-channels +/-20V Analog Input Module** в появившемся контекстном меню. У элемента *I/O Modules* появится дочерний элемент, представляющий описание конфигурации модуля AIM729, как показано на рис. 15.

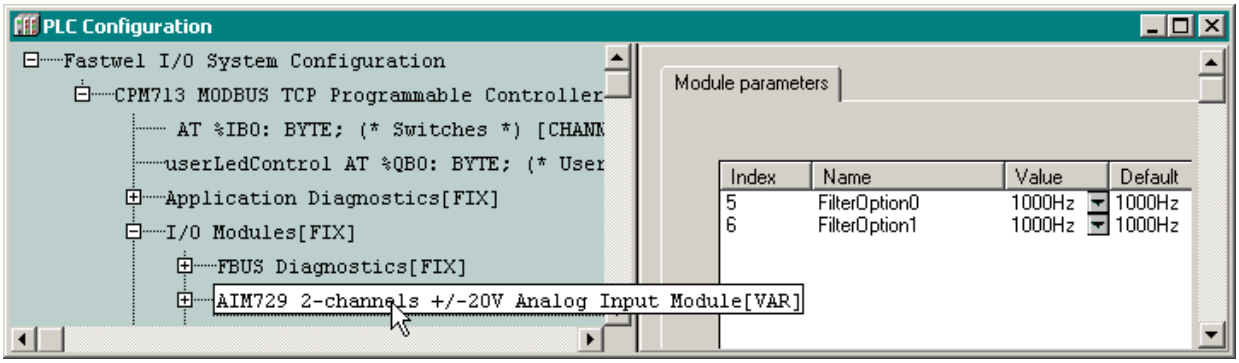

Рис. 14. Дерево конфигурации контроллера после добавления описания модуля AIM729

5. Щелкните правой кнопкой мыши над элементом *I/O Modules* и выберите строку **Append Subelement–DIM712 2-channels AC/DC SPDT Relay Output Module** в появившемся контекстном меню. У элемента *I/O Modules* появится еще один дочерний элемент, представляющий описание конфигурации модуля DIM712, как показано на рис. 15.

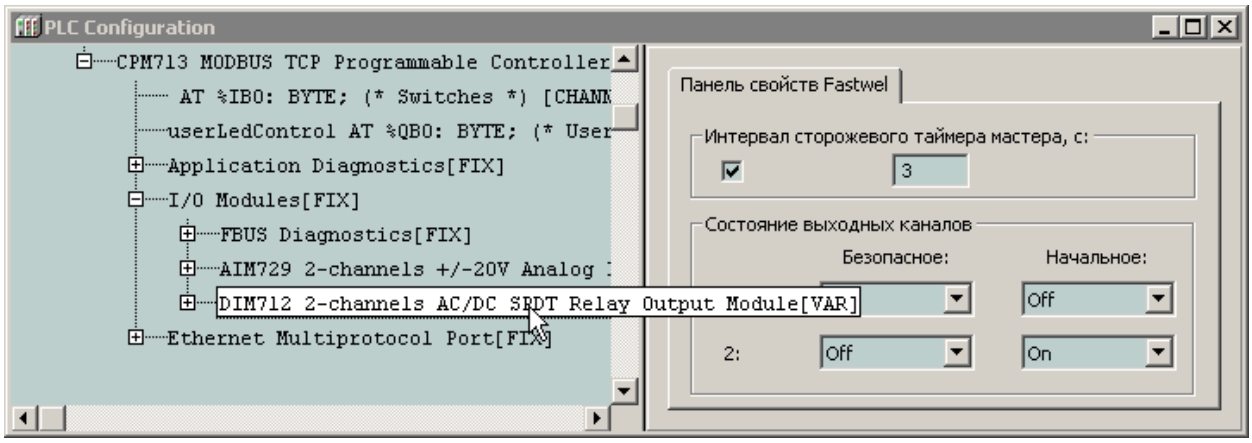

Рис. 15. Дерево конфигурации контроллера после добавления описания модуля DIM712

6. Установите значение интервала сторожевого таймера связи модуля DIM712 с мастером сети FBUS, для чего, выбрав в дереве конфигурации описание модуля DIM712, отметьте флажок в группе параметров **Интервал сторожевого таймера мастера, с** и введите значение *3* в ставшее доступным поле ввода. Таким образом, если в процессе работы модуль не получит ни одного запроса от мастера внутренней шины FBUS, каковым в данном случае является контроллер CPM71x, выходы модуля будут переведены в безопасное состояние, определяемое параметрами **Состояние выходных каналов – Безопасное:1** и **Безопасное:2**.
7. Установите безопасное состояние для первого канала модуля DIM712, для чего выберите значение *On* в выпадающем списке **Безопасное:1** диалоговой панели настройки параметров модуля DIM712. Таким же образом установите начальное состояние первого канала модуля, выбрав *On* в выпадающем списке **Начальное:1**. Диалоговая панель настройки параметров модуля DIM712 примет вид, показанный на рис. 16.

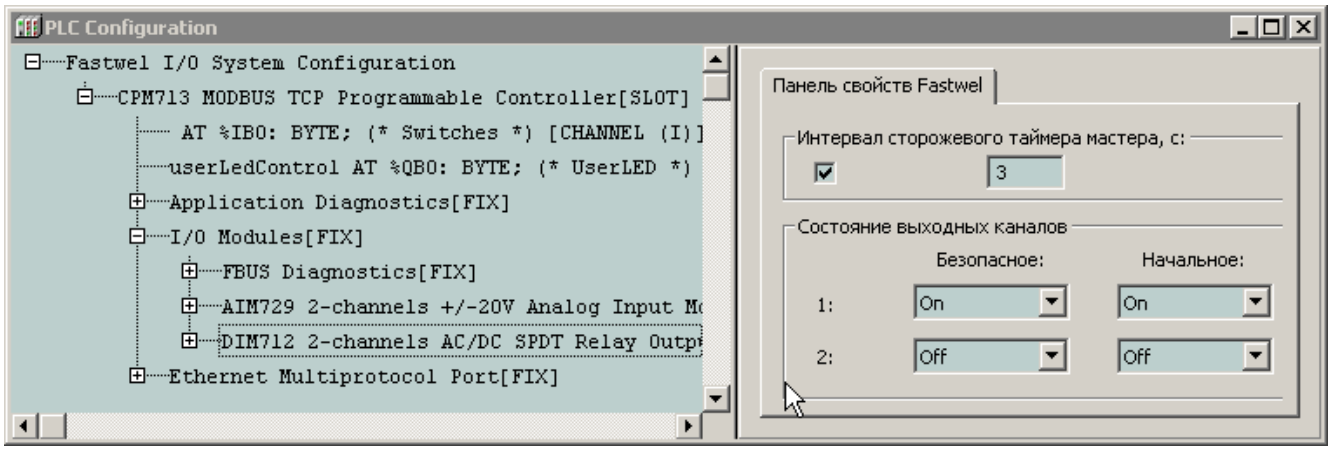

Рис. 16. Параметры модуля DIM712 после установки интервала сторожевого таймера связи с мастером FBUS и начального и безопасного состояний выходных каналов

3.6.2.3. Создание символических имен для каналов модуля DIM712

Для задания начального состояния каналов модуля DIM712 необходимо создать переменные, чьи значения будут определять состояние каждого канала во время работы приложения, и установить для них начальные значения. Указанные переменные будут связывать разрабатываемую программу с окружением.

Как указывалось в п. **Ошибка! Источник ссылки не найден.**, окружением программы являются каналы модулей ввода-вывода, входящих в состав контроллера, и коммуникационные объекты внешней сети. Данные объекты окружения представляются образом процесса.

Среда разработки CoDeSys поддерживает три способа организации связи программ с образом 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*.

В данном случае для каналов дискретного вывода модуля DIM712 будут созданы символические имена, в результате чего доступ к этим каналам из программ CoDeSys может быть получен через соответствующие глобальные переменные.

1. Раскройте элемент *DIM712 2-channels AC/DC SPDT Relay Output Module* дерева конфигурации, после чего раскройте дочерний элемент *Outputs* и канал *OutputsControl*, как показано на рис. 17.
2. Дважды щелкните левой кнопкой мыши слева от надписи *AT %QX0.8* и введите символическое имя канала *relay_out1* в появившееся поле ввода и нажмите клавишу Enter.
3. Дважды щелкните левой кнопкой мыши слева от надписи *AT %QX0.9* и введите символическое имя канала *relay_out2* в появившееся поле ввода и нажмите клавишу Enter.

Таким образом, созданы две глобальные выходные переменные, ссылающиеся на выходные каналы модуля DIM712.

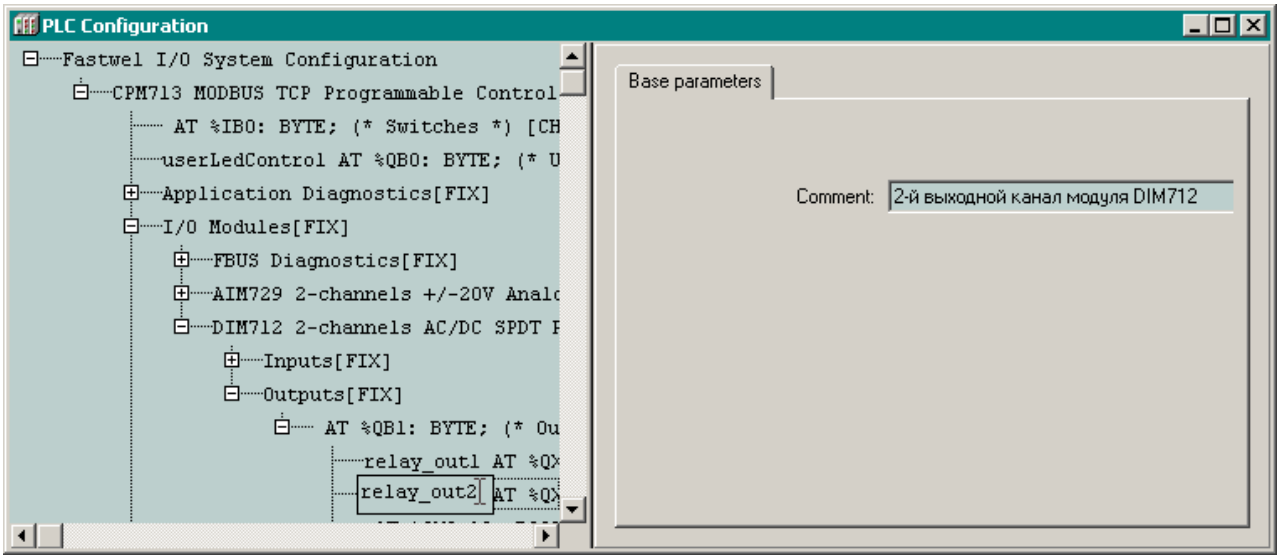

Рис. 17. Создание символических имен для каналов дискретного вывода модуля DIM712

3.6.2.4. Создание обработчика системного события OnPowerOn для фиксации начального состояния выходных каналов

После загрузки приложения в контроллер логические состояния, заданные параметрами **Состояние выходных каналов–Начальное1/2** в таблице параметров модуля DIM712, будут устанавливаться на выходных каналах модуля DIM712 сразу после включения питания контроллера, но до запуска его системного программного обеспечения.

Для того, чтобы сохранить указанные логические состояния на выходных каналах модуля, требуется задать начальные значения переменным *relay_out1* и *relay_out2*, ссылающимся на данные каналы. Установка начальных значений для глобальных выходных переменных, созданных для каналов с символическими именами, может быть осуществлена при помощи пользовательской функции обработки системного события *OnPowerOn*, которая будет вызываться однократно при включении питания контроллера (или после сброса) перед запуском приложения.

Для создания пользовательской функции обработки системного события *OnPowerOn*:

1. Щелкните на вкладке **Resources** в главном окне CoDeSys, после чего дважды щелкните на ресурсе **Tasks Configuration** в древовидном списке ресурсов контроллера. На экране появится окно ресурса **Tasks Configuration**.
2. Выберите элемент *System events* в древовидном списке *Task configuration*. Окно ресурса **Tasks Configuration** примет вид, показанный на рис. 19.

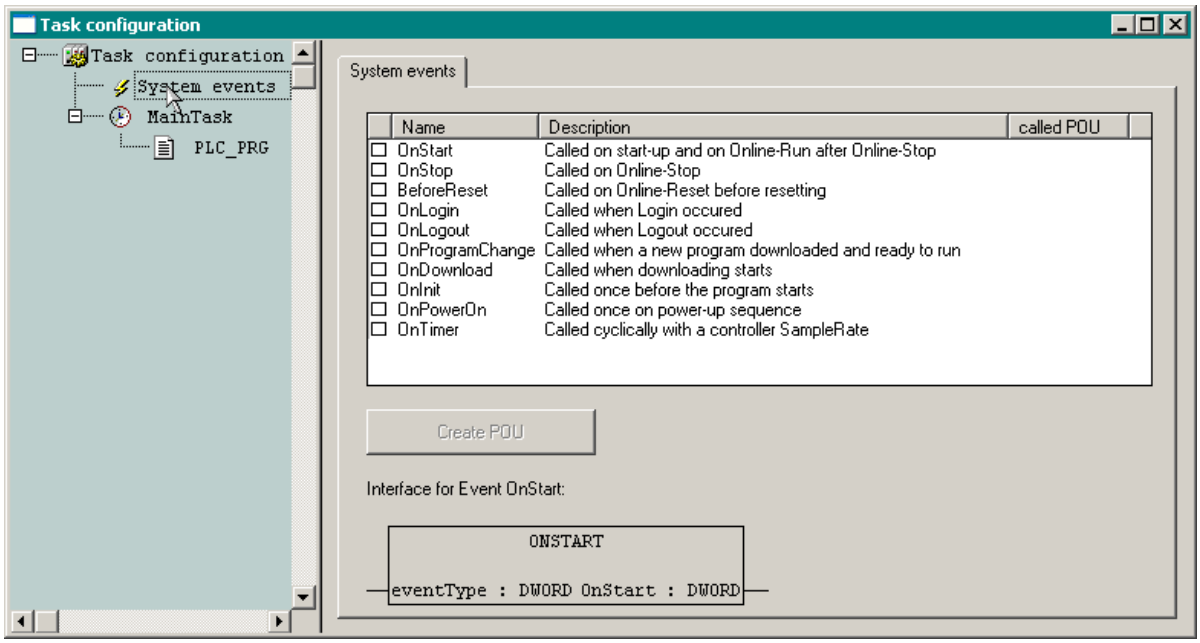

Рис. 18. Таблица системных событий

3. В таблице событий **System events** отметьте флажок слева от имени события *OnPowerOn*, после чего дважды щелкните левой кнопкой мыши в ячейке *OnPowerOn: called POU*, как показано на рис. 19, введите имя функции *InitOutputs*, которая будет вызываться по событию *OnPowerOn*, и щелкните мышью на ячейке, находящейся слева от той, куда только что было введено имя функции *InitOutputs*. Вкладка **System events** примет вид, показанный на рис. 20, при этом кнопка **Create POU ...** станет активной (доступной для нажатия).

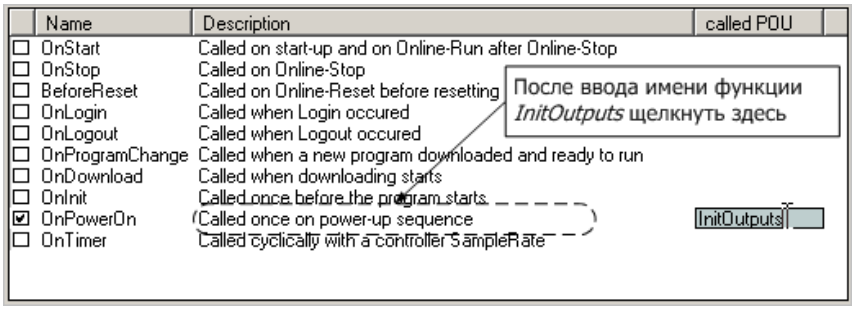

Рис. 19. Ввод имени функции, которая будет вызываться по событию *OnPowerOn* (при включении питания контроллера перед запуском приложения)

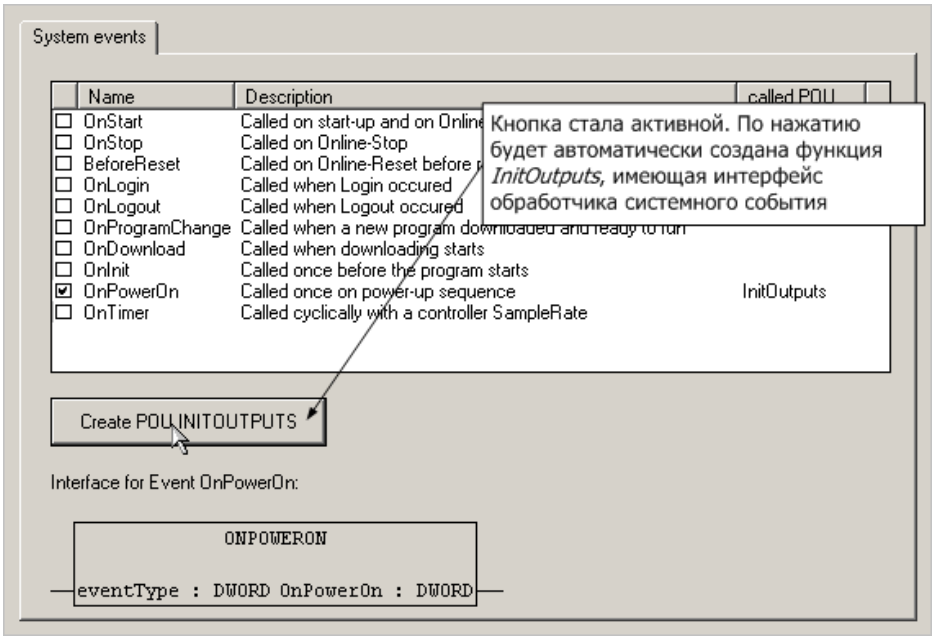

Рис. 20. Внешний вид вкладки System events после ввода имени функции и щелчка на ячейке, расположенной слева от имени функции

4. Нажмите кнопку **Create POU INITOUTPUTS**. Имя функции *InitOutputs* появится в списке программных единиц во вкладке **POUs**, как показано на рис. 21, а кнопка **Create POU** перестанет быть активной.
5. Дважды щелкните левой кнопкой мыши над названием функции *InitOutputs* в списке программных единиц. На экран будет выведено окно редактора исходного текста ST, содержащее заготовку функции *InitOutputs*. Обратите внимание на грозные предупреждения в области деклараций переменных окна редактора. Никогда (НИКОГДА!) не изменяйте содержимое области деклараций в функциях обработки системных событий. Более подробная информация о возможных последствиях нарушения этого правила приведена в п. 4.2.4.4 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.
6. В области редактирования кода функции (область реализации) введите следующие строки:

```
(* Начальное состояние 1-го канала -- включен *)
relay_out1 := TRUE;
(* Начальное состояние 2-го канала -- выключен *)
relay_out2 := FALSE;
```

7. Выберите команду **Project–Build** для трансляции проекта. При отсутствии сообщений об ошибках в области вывода сообщений транслятора можно продолжить работу. В противном случае убедитесь, что имена переменных введены правильно.

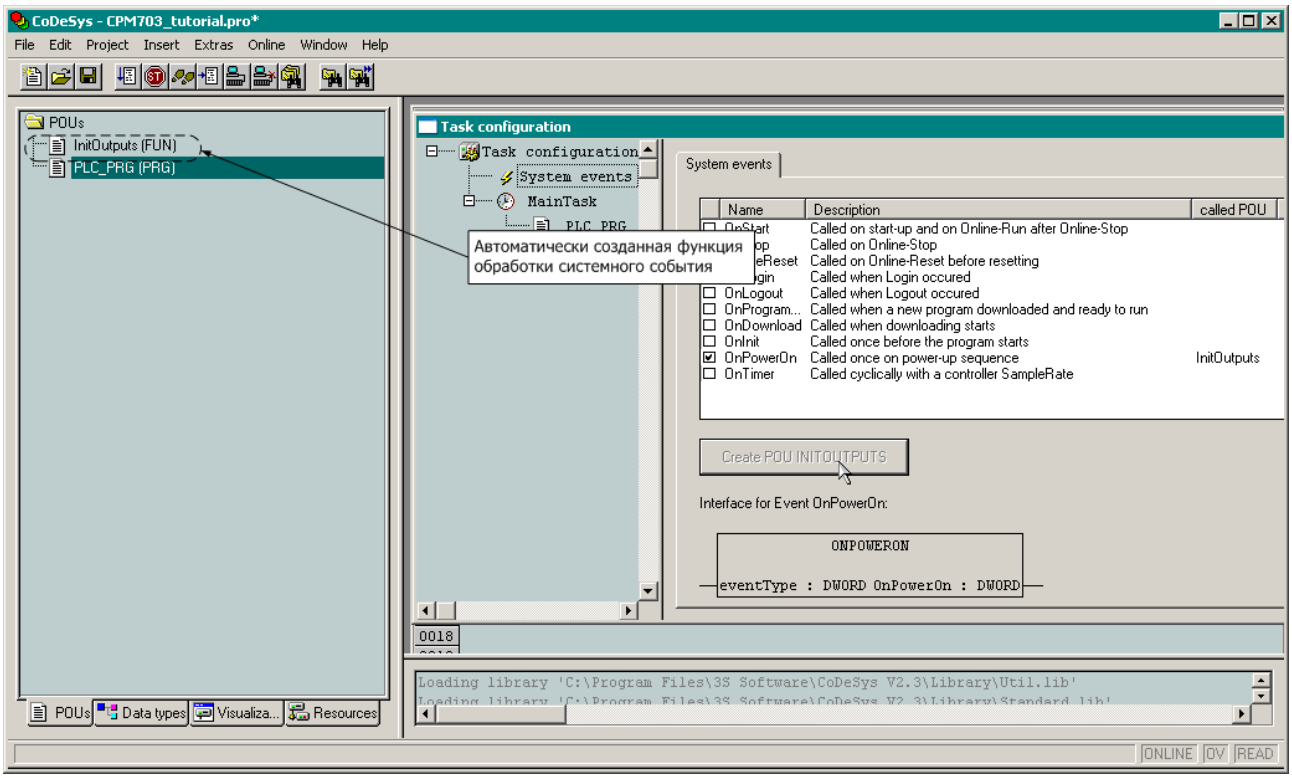

Рис. 21. Внешний вид вкладки System events после ввода имени функции и щелчка на ячейке, расположенной слева от имени функции

Для ускорения ввода имен переменных в редакторе можно воспользоваться функцией поиска символов "на лету" (аналог IntelliSense© корпорации Microsoft или CodeInsight© компании JetBrains) по нажатию сочетания клавиш Ctrl–Space (Ctrl–Пробел). После нажатия указанного сочетания клавиш при нахождении курсора в окне ввода текста непосредственно под курсором появится выпадающий список с именами объектов, доступных в разрабатываемом приложении. Начните вводить имя переменной, и во время ввода первых же букв в выпадающем списке будут предлагаться наиболее подходящие варианты имен объектов. Как только будет предложено искомое имя, нажмите клавишу Tab, и имя переменной появится в окне ввода исходного текста.

Любители щелкать левой кнопкой мыши для быстрого поиска нужной переменной могут воспользоваться диалоговой панелью **Помощник ввода** (**Input assistant**), которая выводится на экран по нажатию клавиши F2, и выглядит, как показано на рис. 22.

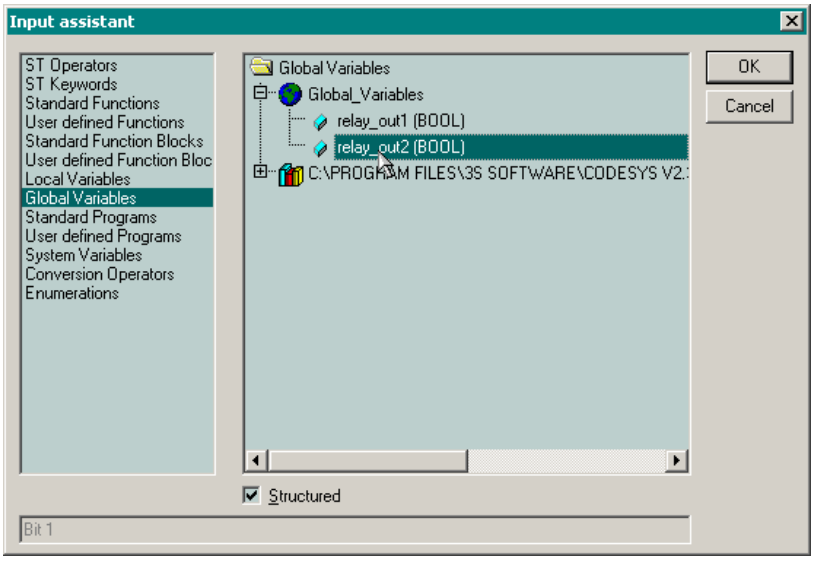

Рис. 22. Помощник ввода при поиске имени переменной

Для того, чтобы установка начального состояния выходных каналов происходила не только после включения питания контроллера, но и после загрузки обновленной версии приложения, также следует вызывать функцию *InitOutputs* по событию *OnInit*. Для получения более подробной информации об обработке системных событий обратитесь к п. 4.2.4.4 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.

3.7. Создание конфигурации внешней сети

3.7.1. Общие сведения

3.7.1.1. Типы коммуникационных объектов

Сервис внешней сети каждого контроллера реализует стек протоколов внешней сети и взаимодействует с приложением пользователя через отведенные для него области образа процесса путем использования единообразного для всех контроллеров внутреннего механизма обмена данными, описание которого приведено в п. 4.2.4 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.

Структура областей образа процесса, отводимых для сервиса внешней сети каждого контроллера, определяется каналами описаний коммуникационных объектов, добавляемых пользователем в конфигурацию контроллера в окне ресурса **PLC Configuration**. Типы коммуникационных объектов в зависимости от типа контроллера представлены в табл. 1.

Таблица 1

Контроллер	Протокол	Тип объекта	Размер	Назначение
CPM711	*CANopen (Slave)*	*RxPDO*	от 1 до 8 байт	Входящий (принимаемый) коммуникационный объект типа Rx PDO, позволяющий приложению контроллера принять от 1 до 8 байт данных по сети.
		TxPDO	от 1 до 8 байт	Исходящий коммуникационный объект типа Tx PDO, позволяющий приложению контроллера передать в сеть от 1 до 8 байт данных.
CPM712 CPM713	*MODBUS MODBUS TCP Slave*	*WORD Output*	2	Двухбайтовый выходной канал типа WORD
		DWORD Output	4	Четырехбайтовый выходной канал типа DWORD
		REAL Output	4	Четырехбайтовый выходной канал типа REAL
		LREAL Output	8	Восьмибайтовый выходной канал типа LREAL
		2-Bytes Output	2	2 однобайтовых выходных канала типа BYTE
		WORD Input	2	Двухбайтовый входной канал типа WORD
		DWORD Input	4	Четырехбайтовый входной канал типа DWORD
		REAL Input	4	Четырехбайтовый выходной канал типа REAL
		LREAL Input	8	Восьмибайтовый входной канал типа LREAL
		2-Bytes Input	2	2 однобайтовых входных канала типа BYTE
CPM712 CPM713	*MODBUS MODBUS TCP Master*	*Переменные удаленных подчиненных узлов (серверов) MODBUS*		
		Read-Write Coil	от 1 до 2000 бит	Используется для доступа по записи-чтению к данным типа Coil удаленного сервера
		Read-Write Holding Register	от 1 до 250 байт	Используется для доступа по записи-чтению к данным типа Holding Registers удаленного сервера
		Write-Only Coil	от 1 до 2000 бит	Используется для доступа по записи к данным типа Coil удаленного сервера
		Write-Only Holding Register	от 1 до 250 байт	Используется для доступа по записи к данным типа Holding Registers удаленного сервера
		Read-Only Input	от 1 до 2000 бит	Используется для доступа по чтению к данным типа Discrete Input удаленного сервера
		Read-Only Register	от 1 до 250 байт	Используется для доступа по чтению к данным типа Input Registers удаленного сервера
		Read-Only Coil	от 1 до 2000 бит	Используется для доступа по чтению к данным типа Coil удаленного сервера
		Read-Only Holding Register	от 1 до 250 байт	Используется для доступа по чтению к данным типа Holding Registers удаленного сервера

Более подробная информация о коммуникационных объектах приведена в соответствующих документах *CPM711 (CPM712/CPM/713). Руководство по конфигурированию и программированию сетевых средств*.

3.7.1.2. Сетевые функции в учебном проекте

В настоящем документе рассматривается простой учебный проект, в котором контроллер должен обеспечивать выполнение следующих функций, связанных с обменом данными реального времени по сети:

1. Прием по сети и выполнение команды включения или выключения второго канала модуля дискретного вывода.

2. Прием по сети и выполнение команды включения индикатора USER, который расположен на передней панели контроллера. При этом команда должна содержать код цвета, которым будет светиться индикатор.
3. Передачу по сети значений входных сигналов на каналах модуля аналогового ввода.
4. При отсутствии связи по сети индикатор USER контроллера должен прерывисто светиться красным цветом ("мигать").

Сетевые протоколы, реализованные в разных контроллерах Fastwel I/O, обеспечивают отличные друг от друга механизмы обмена данными реального времени между узлами сети. Как правило, передача команд или запись значений от одного узла сети (например, выполняющего функцию мастера) другому, реализуется простым (наивным) способом, при котором один из узлов, периодически или однократно, передает другому узлу коммуникационный объект, содержащий требуемое значение переменной. В первом случае, узел, которому передают значение переменной, скорее всего получит наиболее актуальное (свежее) значение независимо от момента времени, когда он подключился к сети. Во втором случае, значение переменной может быть никогда не получено принимающим узлом, если тот подключился к сети после того, как соответствующий коммуникационный объект уже был передан по сети.

В рассматриваемом учебном проекте будет представлен чуть более сложный способ передачи команд от одного узла сети другому:

1. В контроллере CPM71x объявляется специальная переменная *netCommandCounter* типа *WORD*, в которую удаленный узел (в случае использования протокола "мастер-подчиненный" выполняющий функцию мастера) записывает номер команды.
2. Приложение контроллера CPM71x сравнивает значение номера принятой команды со значением, запомненным в переменной *prevNetCommandCounter* предыдущим принятым значением, и, в случае неравенства предыдущего значения с текущим, актуализирует значения, соответствующие параметрам команды: включает или выключает второй канал модуля дискретного вывода и включает или выключает нужным цветом индикатор USER.
3. Предыдущее значение номера команды, запомненное в переменной *prevNetCommandCounter,* передается контроллером CPM71x в сеть удаленному узлу (в CPM711 – по инициативе самого контроллера, а в CPM712/713 – по запросу мастера) в соответствующем исходящем коммуникационном объекте, тем самым позволяя приложению удаленного узла (мастера) определить, принял ли контроллер наиболее актуальное значение команды.

Указанный механизм взаимодействия между приложениями на разных узлах сети, с некоторыми оговорками, позволяет гарантировать доставку и выполнение команды на удаленный узел независимо от типа используемой сети.

3.7.2. Создание конфигурации внешней сети контроллера CPM711 (CANopen)

3.7.2.1. Настройка параметров протокола CANopen

Конфигурация сети CANopen представлена элементом *CANopen Interface* в дереве конфигурации контроллера **PLC Configuration** и содержит основные параметры протокола, а также списки входящих и исходящих коммуникационных объектов, относящихся к данному узлу. Перечень параметров протокола приведен в табл. 2.

Диалоговая панель настройка параметров протокола CANopen контроллера CPM711 представлена на рис. 23.

Для установки сетевого адреса узла выберите элемент *CANopen Interface* в дереве конфигурации контроллера **PLC Configuration** и введите значение адреса (например, *12*) в поле **Адрес узла**, как показано на рис. 23.

Служебное сообщение Heartbeat с идентификатором, равным 16#700+*NodeID*, передаваемое контроллером в сеть с некоторым периодом, позволяет мастеру CANopen убедиться в том, что контроллер с сетевым адресом, заданным параметром **Адрес узла**, присутствует в сети. Установите интервал передачи служебного сообщения Heartbeat равным 500 мс, для чего введите значение *500* в поле **Период Heartbeat, мс**.

Таблица 2

Параметр	Назначение
Конфигурировать протокол из приложения	Если данный флажок установлен, сервис протокола внешней сети будет остановлен перед входом в состояние Initialisation (см. спецификацию DS-301), ожидая, когда приложение выполнит настройку параметров протокола при помощи библиотеки *FastwelCANopen.lib*.
Запуск по команде NMT-мастера	Если данный флажок установлен, сервис протокола внешней сети будет остановлен в состоянии Pre-Operational (см. спецификацию DS-301), ожидая, когда NMT-мастер выполнит настройку параметров протокола и переведет его в состояние Operational.
Адрес узла	Адрес узла в сети CANopen в диапазоне от 1 до 127. По умолчанию 1. В безопасном режиме, не связанном с перезапуском по ошибке, и при отсутствии прикладной программы: 127. Адрес необходим для обеспечения обязательной функциональности протокола CANopen на узле в части NMT и SDO.
Использовать протокол LSS Node-ID	Если данный флажок установлен, адрес узла примет значение 255, а сервис протокола внешней сети будет ожидать, пока LSS-мастер выполнит установку адреса узла.
Скорость обмена	Скорость обмена. Может принимать одно из следующих значений (кбит/с): 10, 20, 50, 125, 250, 500, 800, 1000. По умолчанию и при отсутствии прикладной программы 250.
SYNC CAN-ID	Идентификатор синхронизирующего сообщения SYNC и признак, по которому сервис внешней сети узла устанавливает, является ли он источником сообщения SYNC (SYNC-мастером). Значение по умолчанию 128 (16#080).
Узел является SYNC-мастером	Если установлен данный флажок, данный узел будет выполнять функцию SYNC-мастера, передавая в сеть синхронизирующее сообщение с идентификатором, заданным параметром **SYNC CAN-ID** и периодом, заданным параметром **Период SYNC, мс**
Период SYNC, мс	Период передачи синхронизирующего сообщения в миллисекундах, если узел является SYNC-мастером. Значение по умолчанию 100. Диапазон значений: 25…10000 (от 25 мс до 10 с).
Окно синхронизации, мс	Интервал времени в миллисекундах, в течение которого узел должен передать в сеть все синхронные исходящие коммуникационные объекты. По умолчанию 0. Диапазон значений от 0 до 10000.
Период Heartbeat, мс	Период времени в миллисекундах, с которым узел будет передавать в сеть сообщение Heartbeat. Идентификатор сообщения Heartbeat: 16#700+NodeID. Сообщение содержит байт текущего состояния узла: 4 (STOPPED), 5 (OPERATIONAL), 127 (PRE-OPERATIONAL) Значение по умолчанию равно 0, что означает отсутствие передачи Heartbeat.

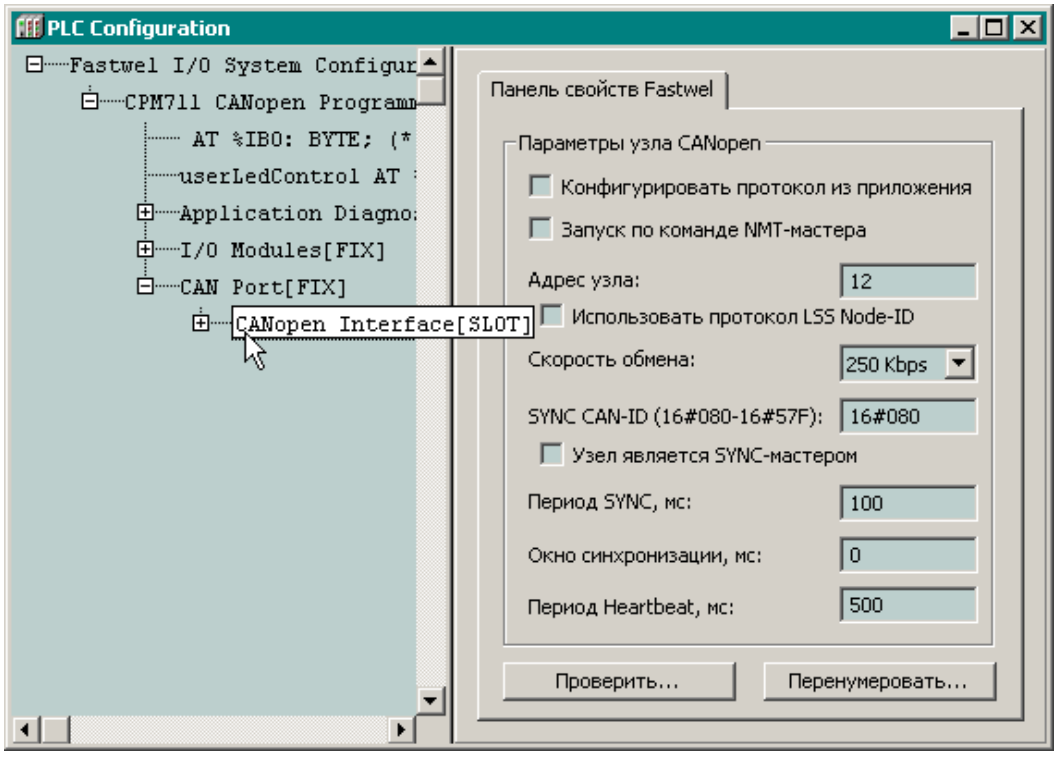

Рис. 23. Параметры протокола CANopen

Остальные параметры протокола оставьте без изменений.

3.7.2.2. Типы коммуникационных объектов протокола CANopen

Сеть CAN является широковещательной системой передачи данных, т.е., во-первых, каждый узел сети может начать передачу по собственной инициативе; во-вторых, все узлы сети "слышат" все сетевые сообщения (если в сетевых адаптерах не выполняется аппаратная фильтрация сообщений), и, в-третьих, сетевое сообщение на уровне передачи данных не содержит адресов отправителя и

получателя, поскольку сообщения различаются получателями по назначенному числовому идентификатору. Данный идентификатор (CAN-ID) служит для выполнения арбитража доступа к шине нескольких узлов в приемо-передатчиках каждого узла. Чем меньше абсолютное значение CAN-ID, тем выше приоритет сообщения с таким идентификатором, а, значит, тем раньше оно будет передано в сеть.

В семействе протоколов CAL (**C**AN **A**pplication **L**ayer), к которому относится CANopen, для идентификации сообщения также используется понятие CAN-ID (**C**ommunication **OB**ject **ID**entifier), под которым подразумевается 32-разрядное число, включающее в себя 11-ти или 29-разрядный CAN-ID и дополненное управляющими битовыми полями, которые используются только прикладным уровнем протокола.

Для сети CAN, вообще, и протокола CANopen, в частности, должно соблюдаться следующее правило – сообщение с некоторым идентификатором должно передаваться в сеть одним, и только одним, узлом. Небольшим исключением из данного правила является сообщение типа RTR (**R**emote **T**ransmission **R**equest – кадр запроса удаленной передачи), которое передается от одного узла другому с тем, чтобы узел, получивший RTR, передал в адрес отправителя RTR сообщение с данными, имеющее тот же идентификатор, что и у RTR.

В протоколе CANopen основным параметром коммуникационного объекта, наряду с CAN-ID, является тип передачи, далее называемый *Transmission Type*, который определяет алгоритм обработки или передачи коммуникационного объекта.

Для обмена данными реального времени в сервисе протокола CANopen контроллера CPM711 предусмотрены следующие типы коммуникационных объектов:

Входящий коммуникационный объект (RxPDO)

Rx PDO – сообщение, принимаемое данным узлом, и позволяющее передавать данному узлу до 8-ми байт данных от любого другого узла сети.

Идентификатор сообщения определяется параметром **CAN-ID**, значение которого должно устанавливаться пользователем из диапазона от 16#101 (257) до 16#57F (1407) и быть уникальным среди остальных входящих и исходящих коммуникационных объектов. Меньшим значениям идентификатора соответствуют более высокие приоритеты CAN-сообщений.

Настройка параметров входящих коммуникационных объектов выполняется в диалоговой панели, показанной на рис. 30, а описание параметров приведено в табл. 3.

Таблица 3

Описание параметров входящих коммуникационных объектов типа RxPDO	
Параметр	**Описание**
Неактивен	Установка данного флажка приведет к тому, что данный коммуникационный объект не будет обрабатываться сервисом протокола внешней сети до тех пор, пока не будет активизирован мастером сети или приложением, исполняющимся на данном контроллере, посредством библиотеки FastwelCANopen.lib.
CAN-ID	Идентификатор сообщения. Диапазон от 16#101 (257) до 16#57F (1407)
Получать по SYNC	Коммуникационный объект будет поступать в узел после синхронизирующего сообщения SYNC
Получать немедленно	Указывают на то, что коммуникационный объект будет поступать в контроллер вне зависимости от сообщения SYNC
Получать по RTR и SYNC	Означает, что контроллер должен передавать кадр удаленного запроса (RTR) для данного коммуникационного объекта. Кадр запроса всегда отправляется только после очередного сообщения SYNC. Если значение параметра **Период RTR, мс**, задаваемое в миллисекундах, отлично от нуля, то дополнительным условием передачи RTR является истечение интервала времени, заданного данным параметром с момента последней передачи запроса.
Получать по RTR	Означает, что контроллер должен передавать кадр удаленного запроса (RTR) для данного коммуникационного объекта. Запрос формируется, если значение параметра **Период RTR, мс** (в миллисекундах) отлично от нуля, и истекло время, заданное данным параметром с момента последней передачи запроса. Кадр запроса отправляется сразу же независимо от синхронизирующего сообщения SYNC.
Период RTR, мс	Период выдачи RTR в миллисекундах от 0 до 10000.

Исходящий коммуникационный объект (TxPDO)

Tx PDO – сообщение, позволяющее контроллеру передавать в сеть до 8-ми байт данных.

Идентификатор сообщения определяется параметром **CAN-ID**, значение которого должно устанавливаться пользователем из диапазона от 16#101 (257) до 16#57F (1407) и быть уникальным для каждого коммуникационного объекта среди остальных входящих и исходящих коммуникационных

объектов. Меньшим значениям идентификатора соответствуют более высокие приоритеты CAN-сообщений.

Настройка параметров исходящих коммуникационных объектов выполняется в диалоговой панели, показанной на рис. 25, а описание значений приведено в табл. 4.

Таблица 4

Описание параметров исходящих коммуникационных объектов типа TxPDO	
Параметр	**Описание**
Неактивен	Установка данного флажка приведет к тому, что данный коммуникационный объект не будет обрабатываться сервисом протокола внешней сети до тех пор, пока не будет активизирован мастером сети или приложением, исполняющимся на данном контроллере, посредством библиотеки FastwelCANopen.lib.
CAN-ID	Идентификатор сообщения. Диапазон от 16#101 (257) до 16#57F (1407)
Передавать по изменению данных и SYNC (=0)	Указывает на то, что коммуникационный объект будет передан в сеть после получения данным узлом синхронизирующего сообщения SYNC при условии изменения данных PDO с момента последней передачи.
Передавать по количеству _ x SYNC	Указывает на то, что коммуникационный объект будет передан в сеть после получения данным узлом **N** синхронизирующих сообщений SYNC, где $1 \le \mathbf{N} \le 240$
Передавать по RTR и SYNC (=252)	Означает, что контроллер будет передавать в сеть данный коммуникационный объект по получении кадра удаленного запроса (RTR) с идентификатором данного коммуникационного объекта, но только после получения очередного сообщения SYNC.
Передавать по RTR (=253)	Означает, что контроллер будет передавать в сеть данный коммуникационный объект после получении кадра удаленного запроса (RTR) с идентификатором данного коммуникационного объекта.
Передавать по таймеру (=254)	Означает, что контроллер будет передавать в сеть данный коммуникационный объект с периодом, задаваемым параметром **Период, мс** (в миллисекундах, от 0 до 10000), если его значение отлично от 0. Коммуникационный объект будет передаваться в сеть не чаще, чем установлено параметром **Не чаще, мс** (в миллисекундах, от 0 до 10000).
Передавать изменению данных или таймеру 255	Означает, что контроллер будет передавать в сеть данный коммуникационный объект в случае изменения его данных или по истечении интервала времени, задаваемого параметром **Период, мс** (в миллисекундах, от 0 до 10000), если его значение отлично от 0. Коммуникационный объект будет передаваться в сеть не чаще, чем установлено параметром **Не чаще, мс** (в миллисекундах, от 0 до 10000).

3.7.2.3. Добавление описаний коммуникационных объектов

По условию задачи, решаемой в процессе создания учебного проекта, контроллер должен передавать в сеть следующую информацию:

1. Значения сигналов на каналах модуля аналогового ввода.
   Для передачи преобразованных значений входных сигналов модуля AIM729, имеющего 2 канала ввода напряжения, в формате с плавающей точкой одинарной точности (тип REAL), потребуется 8 байт, которые могут быть переданы в сеть одним сообщением типа TxPDO. Будем считать, что передача данного сообщения должна происходить после получения контроллером синхронизирующего сообщения SYNC, формируемого мастером синхронизации. В этом случае для исходящего объекта должен быть установлен тип передачи **Передавать по количеству *1* x SYNC**. Пусть идентификатор данного TxPDO будет равен 301 (16#12D).
2. Предыдущее значение счетчика команды управления индикатором USER и вторым каналом модуля дискретного вывода, принятой от другого узла. Будем считать, что передача данного сообщения также должна происходить после получения контроллером синхронизирующего сообщения SYNC, формируемого мастером синхронизации. Пусть идентификатор данного TxPDO будет равен 300 (16#12C), т.е. данный коммуникационный объект будет передаваться в сеть перед коммуникационным объектом, в котором содержатся значения сигналов на каналах модуля аналогового ввода.

Для добавления описания первого исходящего коммуникационного объекта в конфигурацию контроллера:

1. Раскройте элемент *CANopen Interface* в дереве конфигурации контроллера в окне ресурса **PLC Configuration**.
2. Щелкните правой кнопкой мыши над элементом *Transmitting PDOs* и выберите команду **Append TxPDO** в контекстном меню, как показано на рис. 24. Описание соответствующего коммуникационного объекта появится в дереве конфигурации, как показано на рис. 25.

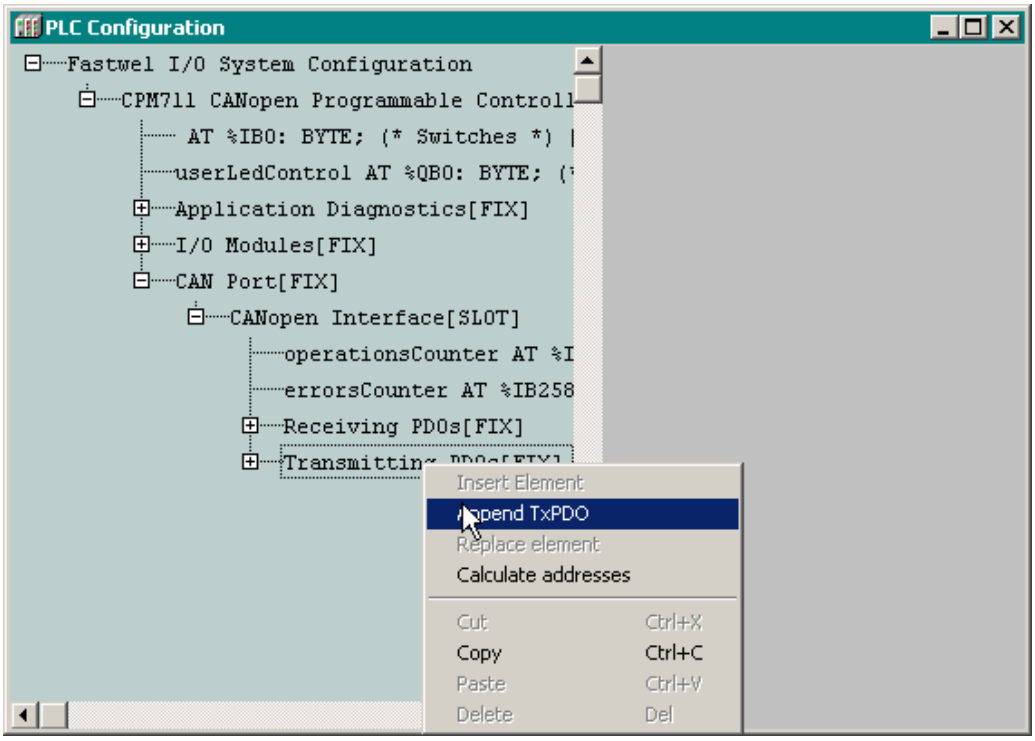

Рис. 24. Добавление описания исходящего коммуникационного объекта

3. Введите значение идентификатора, равное *16#12D*, в поле CAN-ID диалоговой панели настройки параметров коммуникационного объекта, как показано на рис. 26. Остальные параметры оставьте без изменений.

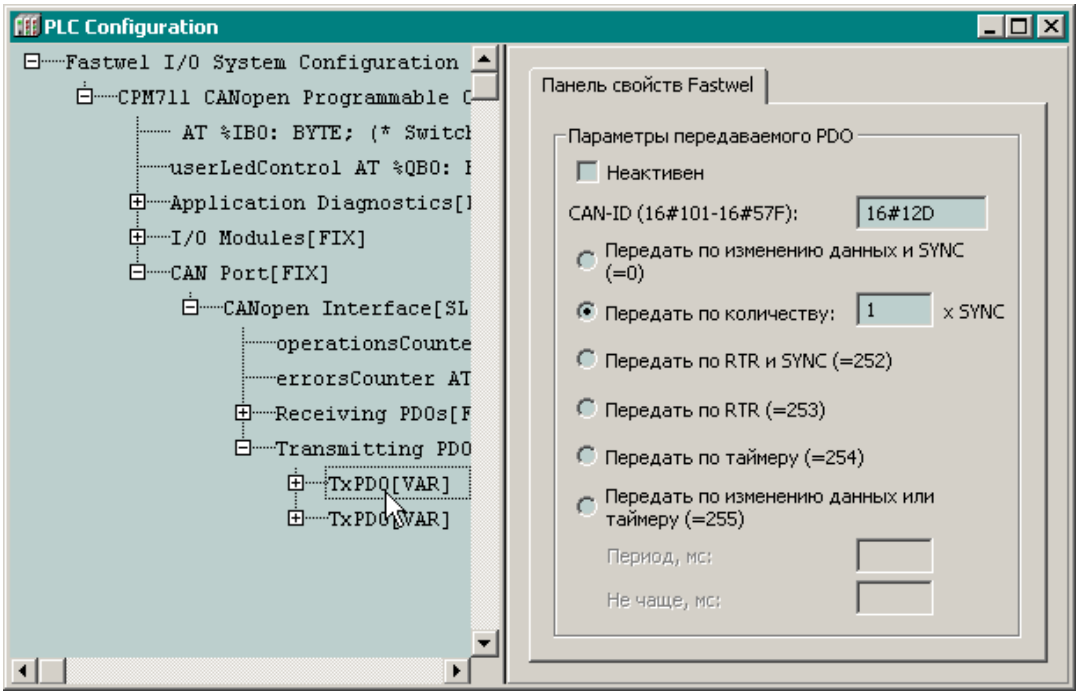

Рис. 25. Установка идентификатора коммуникационного объекта

4. Добавьте два поля передачи данных типа *REAL Output* к описанию созданного коммуникационного объекта, как показано на рис. 26. Выходные каналы указанных полей будут использоваться в приложении для передачи в сеть преобразованных значений на каналах модуля аналогового ввода.

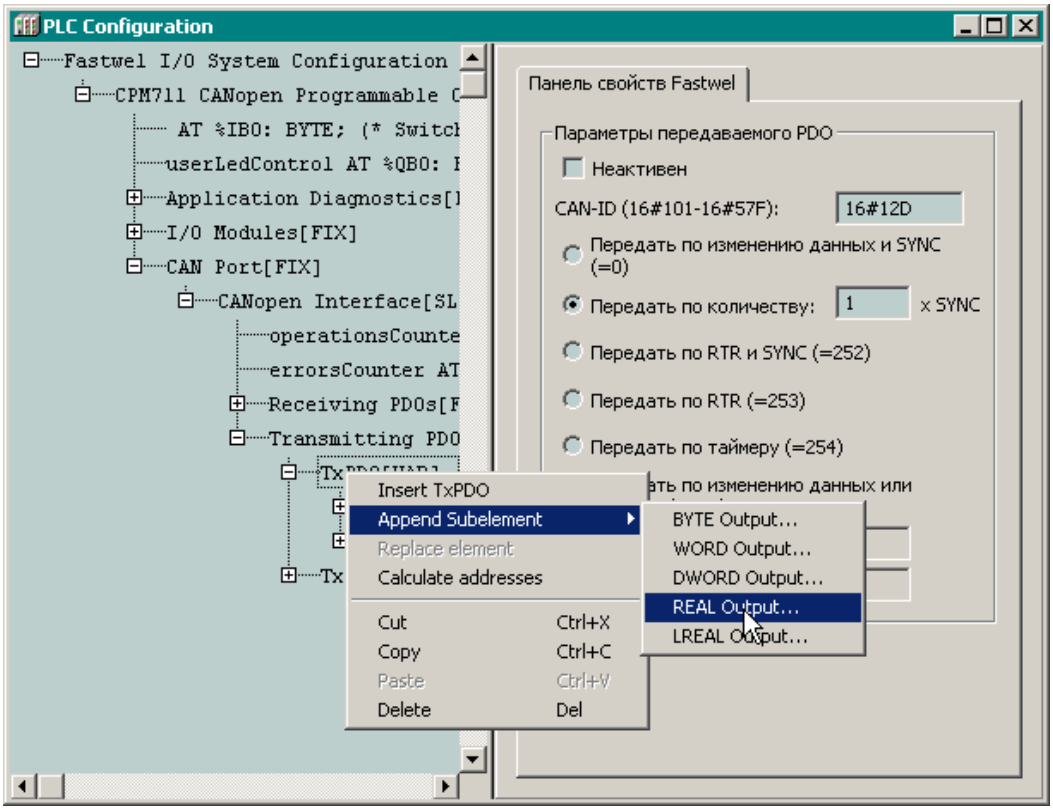

Рис. 26. Добавление одного из полей передачи данных

Для добавления в конфигурацию контроллера описания второго исходящего коммуникационного объекта:

1. Выполните действия пп. 1–3, которые были произведены при добавлении первого исходящего коммуникационного объекта. В качестве идентификатора (COB_ID) введите значение *16#12C*.
2. Добавьте к описанию только что созданного коммуникационного поле передачи данных типа *WORD Output*.
3. Раскройте добавленное поле передачи данных, как показано на рис. 27 и введите символическую ссылку для выходного канала *WORD Output – prevNetCommandCounter*. Процесс ввода символических ссылок иллюстрируется рис. 28. Созданная выходная переменная будет служить для передачи в сеть предыдущего полученного значения счетчика команд.

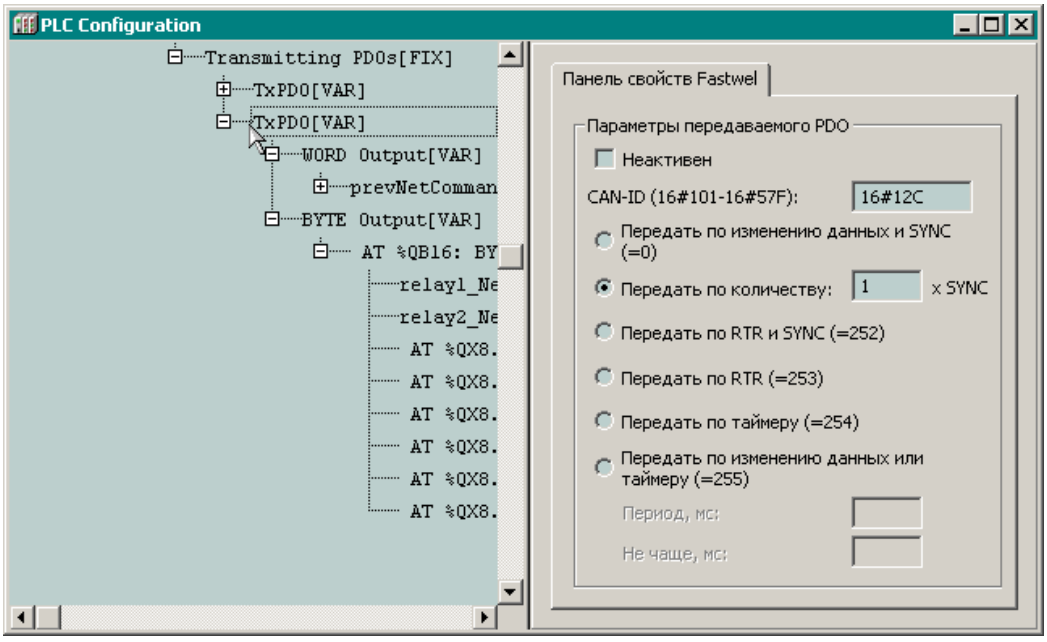

Рис. 27. Каналы полей передачи данных

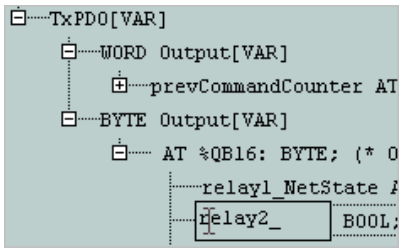

Рис. 28. Ввод символических имен для каналов полей передачи данных

Контроллер должен принимать по сети следующую информацию:

1. Счетчик команды управления индикатором USER и вторым каналом модуля дискретного вывода длиной 2 байта (типа WORD).
2. Код цвета, которым должен светиться индикатор USER, принимающий значения от 0 до 2 (0 – выключить, 1 – зеленый, 2 – красный). Код цвета может быть передан одним байтом данных, однако с целью обеспечения совместимости программного кода со всеми типами контроллеров, будем передавать код цвета двумя байтами.
3. Один бит данных, определяющий состояние второго канала модуля дискретного вывода.

Перечисленные принимаемые данные могут быть переданы контроллеру одним коммуникационным объектом, описание которого в виде RxPDO должно быть добавлено в конфигурацию контроллера.

Для добавления описания входящего коммуникационного объекта в конфигурацию контроллера:

1. Раскройте элемент *CANopen Interface* в дереве конфигурации контроллера в окне ресурса **PLC Configuration**.
2. Щелкните правой кнопкой мыши над элементом *Receiving PDOs* и выберите команду **Append RxPDO**, как показано на рис. 29. Описание соответствующего коммуникационного объекта появится в дереве конфигурации.

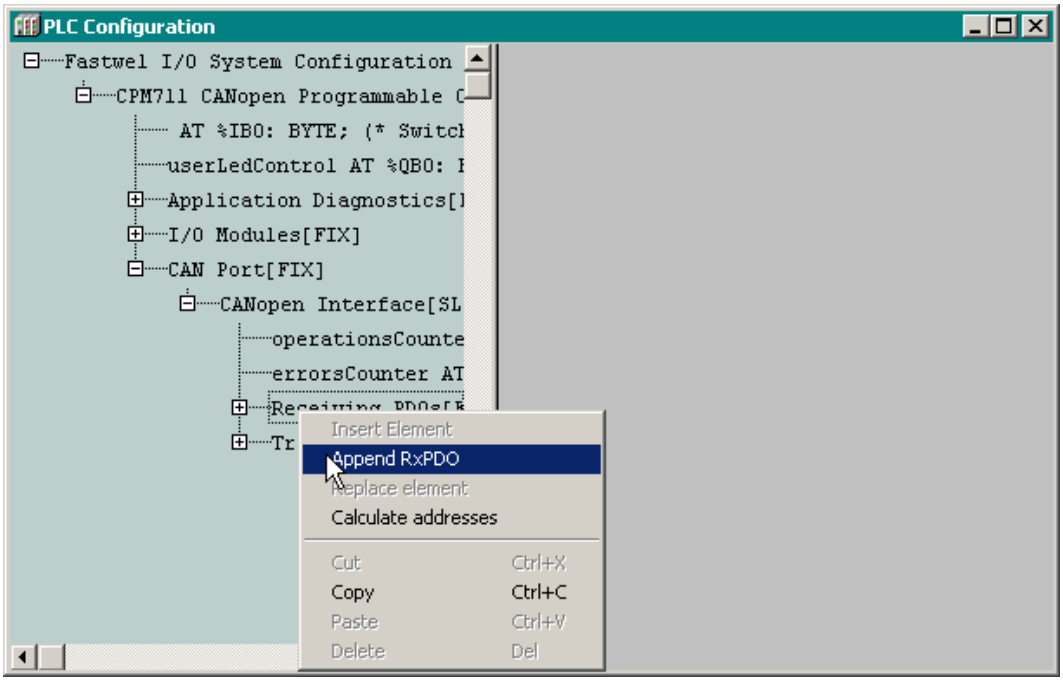

Рис. 29. Добавление описания входящего коммуникационного объекта

3. Введите значение идентификатора, равное *16#101*, в поле **CAN-ID** диалоговой панели настройки параметров коммуникационного объекта. Остальные параметры оставьте без изменений.
4. Добавьте к описанию только что созданного коммуникационного два поля приема данных типа *WORD Input* и одно – типа *BYTE Input* по команде контекстного меню, вызываемого над *RxPDO*.
5. Раскройте добавленные поля приема данных, как показано на рис. 30 и введите символические ссылки для соответствующих входных каналов: для первого *WORD*

Input – netCommandCounter, для второго – *userLED_cmd*, а в первом битовом поле *BYTE Input – relay2_cmd*. Созданные входные переменные будут служить для приема счетчика команд, команды управления индикатором USER и вторым каналом модуля дискретного вывода соответственно.

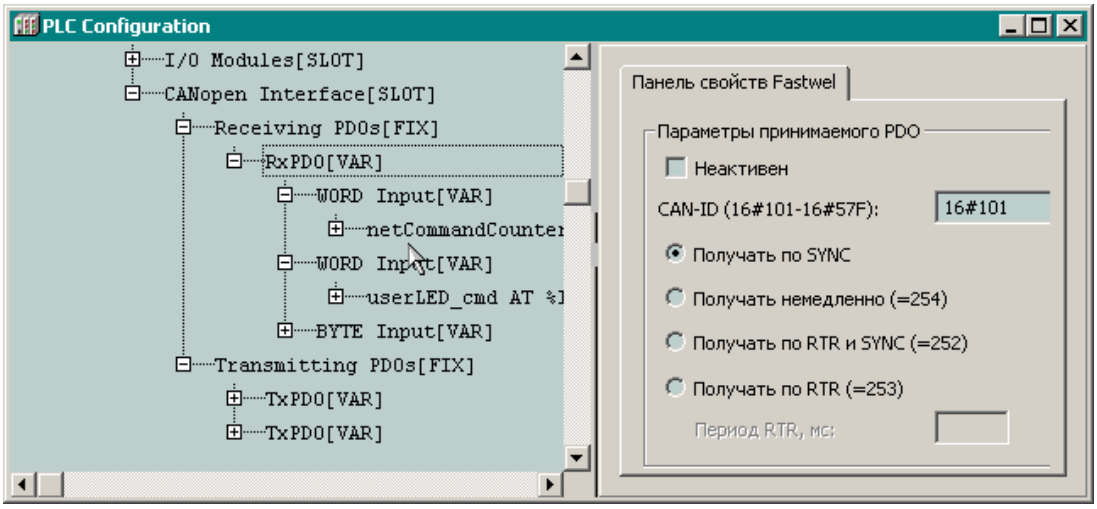

Рис. 30. Символические имена для каналов полей приема данных

На этом создание конфигурации внешней сети контроллера можно считать завершенным.

3.7.3. Создание конфигурации внешней сети контроллеров CPM712 (MODBUS RTU/ASCII) и CPM713 (MODBUS TCP)

3.7.3.1. Настройка параметров протокола MODBUS контроллера CPM712

Перед началом настройки параметров сервера протокола MODBUS в конфигурации проекта для контроллера CPM712 следует выбрать тип протокола, который будет функционировать через коммуникационный порт COM2. Для этого щелкните правой кнопкой мыши над элементом *CPM712 MODBUS RTU/ASCII Programmable Controller–Serial Port–Not Used* в дереве конфигурации и выберите опцию **Replace elemenet–Modbus Serial Slave** в появившемся контекстном меню, как показано на рис. 31.

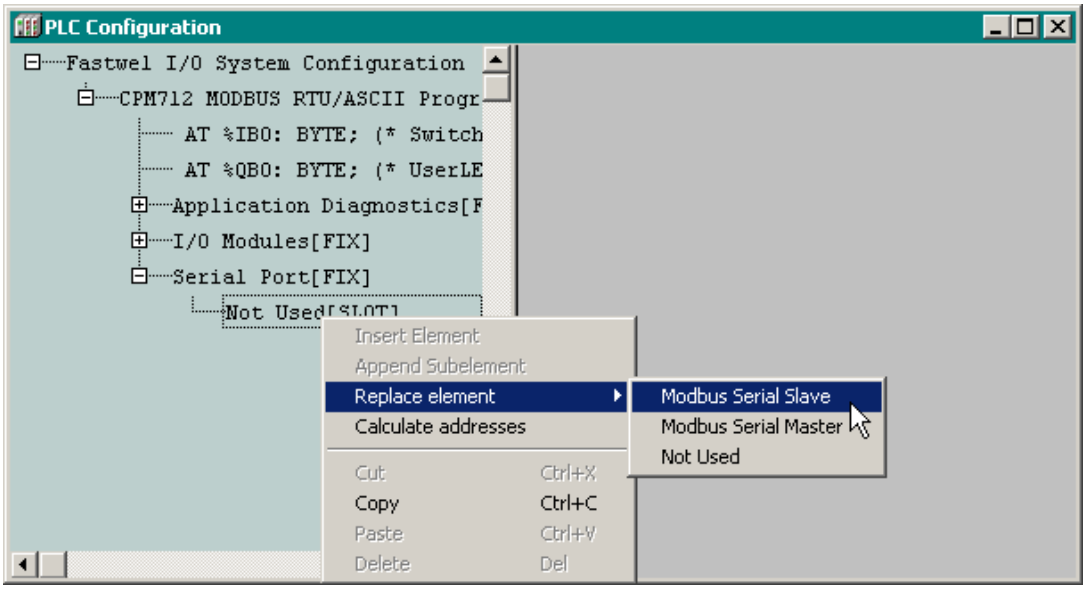

Рис. 31. Установка сервера MODBUS RTU/ASCII для порта COM2 CPM712

Содержимое окна **PLC Configuration** примет вид, показанный на рис. 32.

Настройка параметров протокола контроллера CPM712 выполняется в группе полей редактирования **Параметры сети MODBUS** вкладки **Панель свойств Fastwel**, показанной на рис. 32. Данная вкладка появляется в правой части окна **PLC Configuration** при щелчке мышью на элементе *Modbus Serial Slave* в дереве конфигурации.

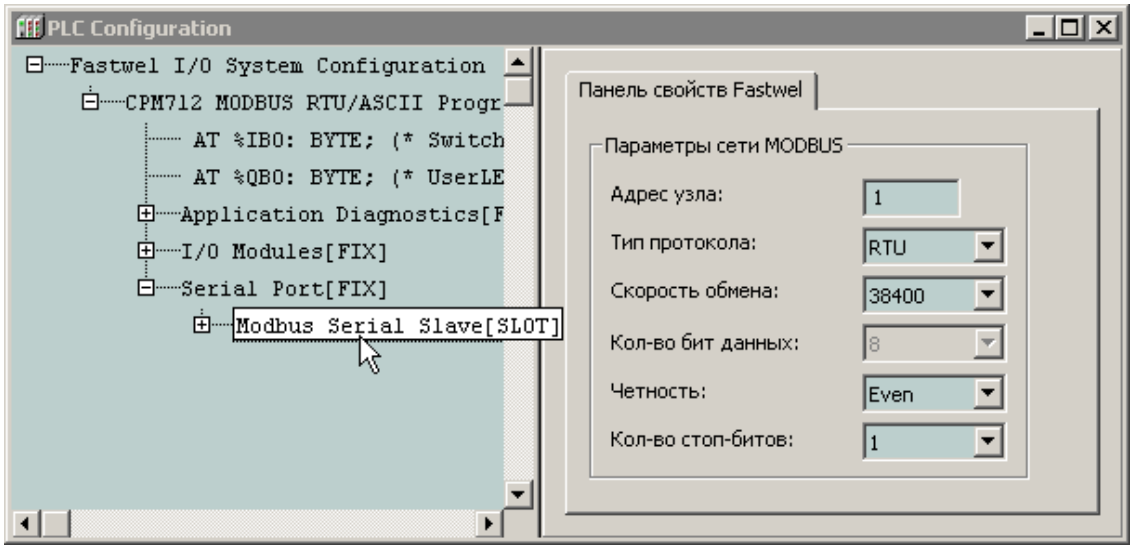

Рис. 32. Параметры протокола MODBUS RTU/ASCII

Для того, чтобы только что установленные параметры протокола были переданы в дерево конфигурации проекта CoDeSys, необходимо выполнить одно из следующих действий:

- сохранить проект (**File–Save**); или
- выполнить построение проекта (**Project–Build**); или
- выбрать в дереве проекта в ресурсе **PLC Configuration** узел, отличный от *Modbus Serial Slave* (по простому говоря, нужно щелкнуть мышкой на каком-нибудь другом элементе дерева).

В рассматриваемом учебном проекте оставьте неизменными исходные значения параметров протокола.

3.7.3.2. Настройка параметров протокола MODBUS TCP контроллера CPM713

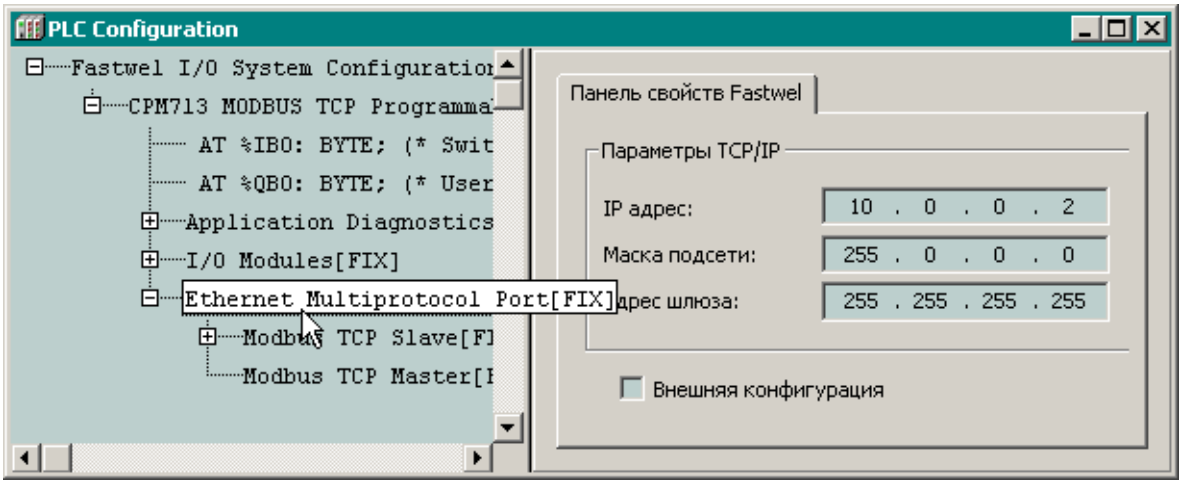

Рис. 33. Параметры протокола MODBUS TCP

Настройка параметров стека протоколов TCP/IP контроллера CPM713 выполняется в группе полей редактирования **Параметры TCP/IP** вкладки **Панель свойств Fastwel**, показанной на рис. 33. Данная вкладка появляется в правой части окна **PLC Configuration** при щелчке мышью на элементе *CPM713 MODBUS TCP Programmable Controller–Ethernet Multiprotocol Port* в дереве конфигурации.

В поле **IP адрес** установите значение 10.0.0.2, а остальные параметры оставьте неизменными.

Если параметры сегмента сети, к которой подключен CPM713, отличаются от имеющихся в группе **Параметры TCP/IP**, введите допустимые в используемом сегменте сети значения IP-адреса, маски подсети и адреса шлюза.

Для того, чтобы только что установленные параметры протокола были переданы в дерево конфигурации проекта CoDeSys, необходимо выполнить одно из следующих действий:

- сохранить проект (**File–Save**); или
- выполнить построение проекта (**Project–Build**); или

– выбрать в дереве проекта в ресурсе **PLC Configuration** узел, отличный от *Ethernet Multiprotocol Port* (по простому говоря, нужно щелкнуть мышкой на каком-нибудь другом элементе дерева).

ВНИМАНИЕ! Не устанавливайте флажок **Внешняя конфигурация**.

3.7.3.3. Создание объектов данных

Области памяти контроллера, отображаемые на множества регистров и битовых полей сервера MODBUS, описываются в секциях *Inputs* и *Outputs* в древовидном списке конфигурации сервера. Секция *Inputs*, показанная на рис. 34, содержит список объектов доступа к данным, поступающим по сети от удаленных клиентов MODBUS.

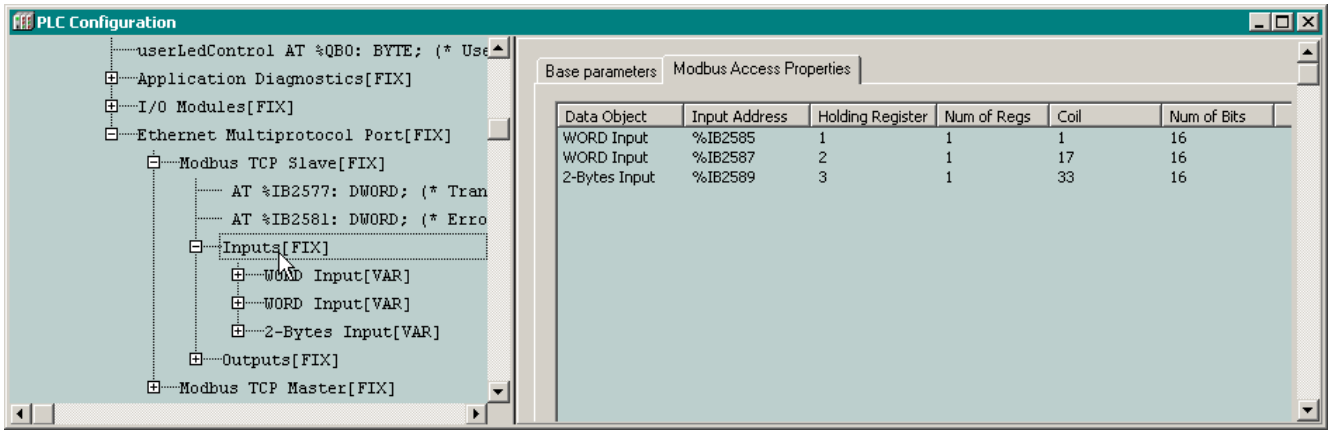

Рис. 34. Область входных данных сервера MODBUS

Секция *Outputs*, показанная на рис. 35, содержит список объектов доступа к данным для передачи в сеть.

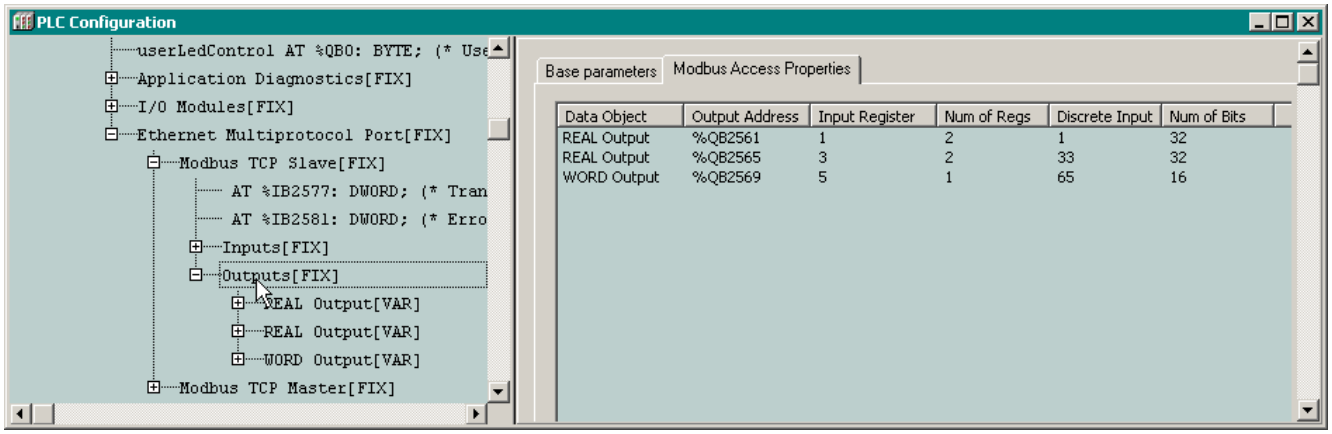

Рис. 35. Область выходных данных сервера MODBUS

Перечень типов и назначение используемых объектов данных приведены в табл. 5.

Таблица 5

Тип секции	Тип объекта	Область данных объекта
Outputs	WORD Output	Двухбайтовый выходной канал типа WORD
	DWORD Output	Четырехбайтовый выходной канал типа DWORD
	REAL Output	Четырехбайтовый выходной канал типа REAL
	LREAL Output	Восьмибайтовый выходной канал типа LREAL
	2-Bytes Output	2 однобайтовых выходных канала типа BYTE
Inputs	WORD Input	Двухбайтовый входной канал типа WORD
	DWORD Input	Четырехбайтовый входной канал типа DWORD
	REAL Input	Четырехбайтовый выходной канал типа REAL
	LREAL Input	Восьмибайтовый входной канал типа LREAL
	2-Bytes Input	2 однобайтовых входных канала типа BYTE

Доступ к каждому объекту данных по сети или к группам объектов может осуществляться при помощи сетевых запросов чтения или/и записи, имеющихся в протоколе MODBUS. Информация о типе коммуникационного объекта MODBUS (Input Register, Holding Register, Discrete Input или Coil), его начальном адресе и количестве объектов в запросе MODBUS содержится в панели свойств **Modbus Access Properties**, показанной на рис. 36.

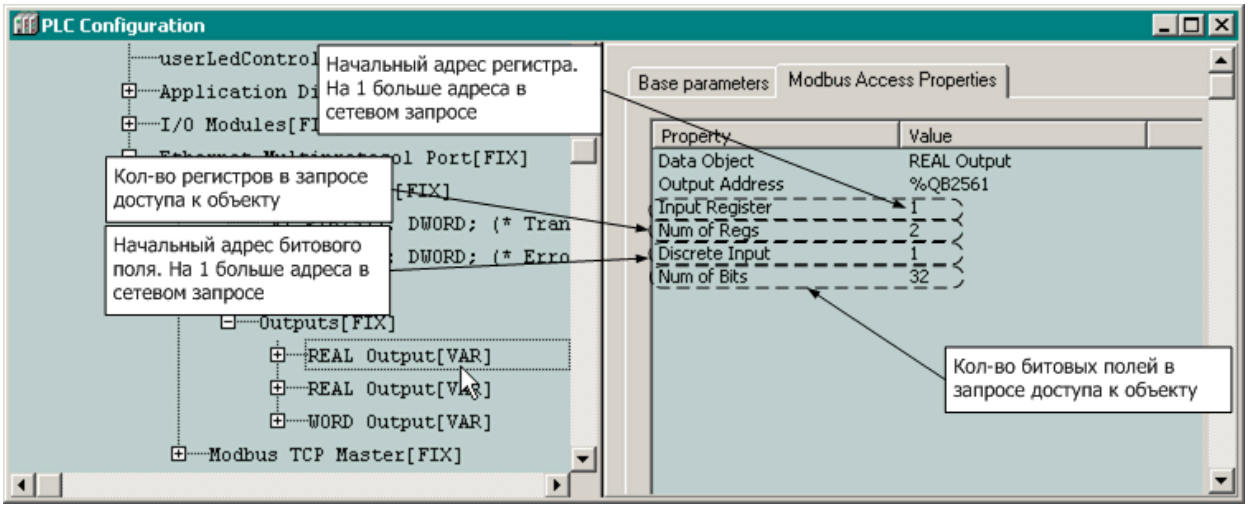

Рис. 36. Сетевые свойства объекта данных MODBUS

По условию задачи, решаемой в процессе создания учебного проекта, контроллер должен передавать в сеть следующую информацию:

1. Значения сигналов на каналах модуля аналогового ввода.
   Для передачи значений входных сигналов модуля AIM729, имеющего 2 канала ввода напряжения, в формате с плавающей точкой одинарной точности (тип REAL) потребуется в область *Outputs* добавить два объекта данных типа *REAL Output*.
2. Предыдущее значение счетчика команды управления индикатором USER и вторым каналом модуля дискретного вывода, принятой от другого узла, может передано одним объектом данных типа *WORD Output*.

Для добавления описаний объектов данных, передаваемых в сеть:

1. Щелкните правой кнопкой мыши над элементом *Serial Port–Modbus Serial Slave–Outputs* (для CPM712) или *Ethernet Multiprotocol Port–Modbus TCP Slave–Outputs* (для CPM713) в окне ресурса **PLC Configuration.** и выберите команду **Append subelement–REAL Output** в появившемся контекстном меню. Описание соответствующего объекта данных появится в дереве конфигурации.
2. Повторите действие п. 1 еще один раз, после чего раскройте последний из добавленных элементов дерева. Дерево конфигурации протокола примет вид, показанный на рис. 34.
3. Щелкните правой кнопкой мыши над элементом *Serial Port–Modbus Serial Slave–Outputs* (для CPM712) или *Ethernet Multiprotocol Port–Modbus TCP Slave–Outputs* (для CPM713) в окне ресурса **PLC Configuration**. и выберите команду **Append subelement–WORD Output** в появившемся контекстном меню. Описание соответствующего объекта данных появится в дереве конфигурации.
4. Раскройте в дереве только что добавленный элемент WORD Output, дважды щелкните левой кнопкой мыши слева от надписи *AT %Q...* и введите символическое имя *prevNetCommandCounter* для канала объекта данных, через который будет передаваться предыдущее значение счетчика команды мастеру сети, и нажмите клавишу Enter. Процесс ввода символического имени иллюстрируется рис. 37.

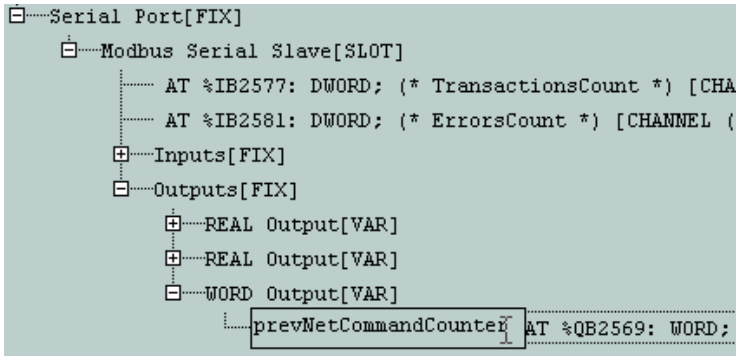

Рис. 37. Ввод символического имени для канала объекта выходных данных

Контроллер должен принимать по сети следующую информацию:

1. Счетчик команды управления индикатором USER и вторым каналом модуля дискретного вывода длиной 2 байта (типа WORD). Для приема счетчика команды может использоваться один объект данных типа *WORD Input*.
2. Код цвета, которым должен светиться индикатор USER, принимающий значения от 0 до 2 (0 – выключить, 1 – зеленый, 2 – красный). Для приема кода цвета также может использоваться один объект данных типа *WORD Input*.
3. Один бит данных, определяющий состояние второго канала модуля дискретного вывода. Для приема команды управления вторым каналом модуля дискретного вывода может использоваться первое битовое поле объекта типа *2-Bytes Input*.

Для добавления описаний объектов данных, принимаемых по сети:

1. Щелкните правой кнопкой мыши над элементом *Serial Port–Modbus Serial Slave–Inputss* (для CPM712) или *Ethernet Multiprotocol Port–Modbus TCP Slave–Inputs* (для CPM713) в окне ресурса **PLC Configuration**. и выберите команду **Append subelement–WORD Input** в появившемся контекстном меню. Описание соответствующего объекта данных появится в дереве конфигурации.
2. Повторите действие п. 1 еще один раз, после чего раскройте оба добавленных элемента дерева.
3. Щелкните правой кнопкой мыши над элементом *Serial Port–Modbus Serial Slave–Inputss* (для CPM712) или *Ethernet Multiprotocol Port–Modbus TCP Slave– Inputss* (для CPM713) в окне ресурса **PLC Configuration**. и выберите команду **Append subelement–2-Bytes Input** в появившемся контекстном меню. Описание соответствующего объекта данных появится в дереве конфигурации.
4. Раскройте только что добавленные элементы дерева типа *WORD Input* и *2-Bytes Input*, после чего добавьте символические ссылки, как показано на рис. 38.

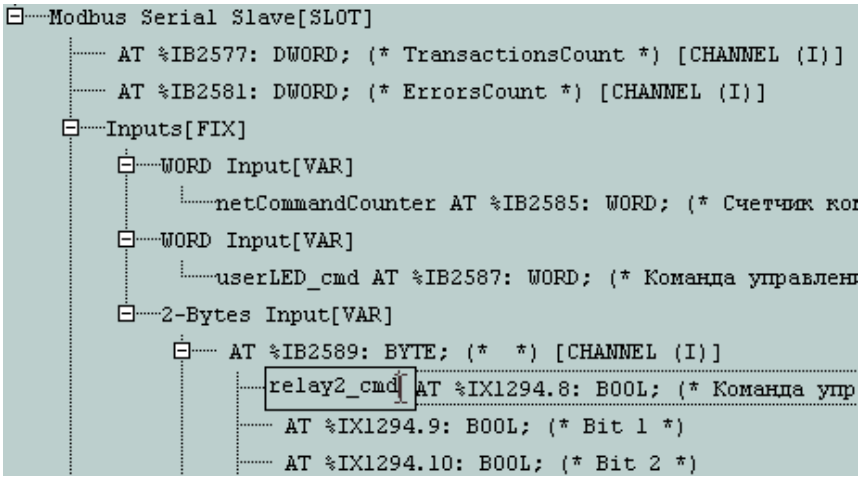

Рис. 38. Описания входных и выходных регистров MODBUS

5. Для сохранения изменений выполните команду меню **File–Save**.

3.8. Разработка программы

3.8.1. Общие сведения о программной модели контроллера

Программная модель контроллера определяет порядок выполнения прикладной программы, систему типов данных, представляющих состояние программы и внешнего окружения, а также способы взаимодействия прикладной программы с устройствами ввода-вывода и внешней сетью, которые также определяются моделью окружения программы.

3.8.2. Единицы организации программы

Приложение, разрабатываемое пользователем в среде CoDeSys для контроллеров CPM71x серии Fastwel I/O, должно состоять хотя бы из одной программы (в терминах IEC 61131-3), и может

содержать до 16-ти циклических и до 64-ти ациклических задач, а также функций обработки системных событий.

Циклической задачей 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 исполнения операционной системы выделить процессорное время в тот или иной момент времени.

Ациклической задачей 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.

Таким образом, задачи являются системными ресурсами контроллера, позволяющими распределить процессорное время между программами, входящими в состав приложения.

Обработчиком системного события 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, сброс контроллера по команде из среды CoDeSys и другие.

Прикладной алгоритм, реализуемый пользователем в приложении, представляется множеством программ (программных единиц типа *PROGRAM*), которые, в свою очередь, могут содержать множество вызовов других программ, функций (программных единиц типа *FUNCTION*) и экземпляров функциональных блоков (программных единиц типа *FUNCTION BLOCK*), реализованных на языках ST, CFC, SFC, LD, FBD, IL стандарта IEC 61131-3.

Программная единица при вызове получает входные данные, обрабатывает их, в том числе с учетом своего внутреннего состояния, и передает результаты в виде выходных данных другим программным единицам или во внешнее окружение.

Входные данные программы, функционального блока и функции представляются так называемыми входными переменными, которые объявляются в секции *VAR_INPUT* области декларации переменных редактора программ.

Выходные данные программы, функционального блока и функции представляются так называемыми выходными переменными, которые объявляются в секции *VAR_OUTPUT* области декларации переменных редактора программ.

Внутреннее состояние программной единицы представляется так называемыми внутренними переменными, которые объявляются в секции *VAR* области декларации переменных редактора программ.

Важно понимать основные отличия между программой, функциональным блоком и функцией:

1. Программа и экземпляр функционального блока, в отличие от функции, сохраняют свое состояние между вызовами. Иными словами, внутренние переменные программы и экземпляра функционального блока сохраняют свои значения после очередного вызова и до следующего. Время жизни входных, выходных и внутренних (локальных) переменных функции ограничено интервалом между вызовом и возвратом результата вызова.
2. Программа отличается от функционального блока тем, что в приложении может присутствовать единственный экземпляр программы с некоторым именем, в то время, как функциональный блок с некоторым именем может использоваться для декларации и последующего вызова множества переменных, тип которых совпадает с именем функционального блока.

Таким образом, программная единица типа *PROGRAM* с некоторым именем (например, PROG_NAME), созданная пользователем, по сути определяет пользовательский тип данных (PROG_NAME) и единственный экземпляр ("переменную") данного типа, имя которого совпадает с именем типа. Программа может вызываться автоматически из корневой программной единицы некоторой задачи, с которой она явно ассоциирована в окне ресурса **Tasks Configuration**. Кроме того,

программу можно вызвать явно по имени с передачей входных переменных и получением выходных переменных.

Программная единица типа *FUNCTION_BLOCK* с некоторым именем (например, FB_NAME) определяет пользовательский тип данных с этим именем (FB_NAME), а экземпляры данного типа создаются пользователем в виде внутренних переменных программ или других блоков. Вызов экземпляра функционального блока осуществляется по имени соответствующей переменной в теле содержащей его программы или блока, причем при вызове можно передавать входные и получать выходные переменные.

Более подробная информация о программных единицах CoDeSys приведена в документе *Руководство пользователя по программированию ПЛК в CoDeSys V2.3*.

3.8.3. Типы данных

В адаптированной среде CoDeSys поддержаны следующие примитивные типы IEC 61131-3: BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME.

К непримитивным типам (или конструкторам типов) относятся массив (ARRAY) и структура (STRUCT).

Специальными типами являются указатель (POINTER TO), перечисление, поддиапазон, ссылка и строка (STRING).

Более подробная информация о типах CoDeSys приведена в документе *Руководство пользователя по программированию ПЛК в CoDeSys V2.3*.

3.8.4. Модель окружения

Окружением приложения, разрабатываемого пользователем для программируемого контроллера, являются каналы устройств и модулей ввода-вывода, поля передачи данных входящих и исходящих коммуникационных объектов внешней сети, диагностическая информация от различных подсистем контроллера и другие сущности, которые не могут быть представлены явно в понятиях программной модели IEC 61131-3.

Моделью окружения приложения можно считать совокупность способов описания связи кода прикладной программы с данными устройств ввода-вывода и сети, а также механизмы среды исполнения контроллера, реализующие обмен данными между программными единицами приложения и устройствами ввода-вывода и сетью.

Для каждой пользовательской задачи, добавляемой в окне ресурса **Tasks Configuration**, 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. 39.

Описатель ссылки на входную или выходную область образа процесса служит для обмена данными между задачей и окружением перед вызовом (для ввода данных из окружения) и после вызова (для вывода данных в окружение) корневой программной единицы задачи и формируется средой разработки CoDeSys во время трансляции проекта для каждой входной или выходной переменной, отображаемой на образ процесса и принадлежащей программе, которая вызывается из данной задачи.

Принцип формирования описателей ссылок иллюстрируется рис. 40. 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.

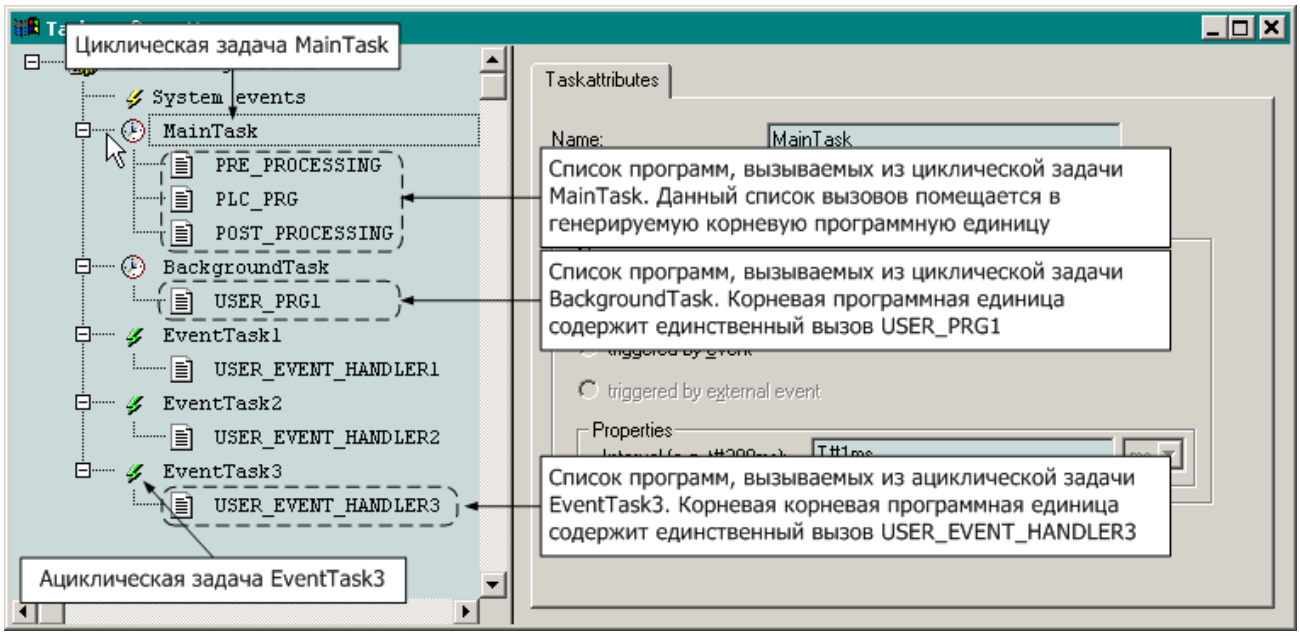

Рис. 39. Списки программных единиц, вызываемых из задач

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!

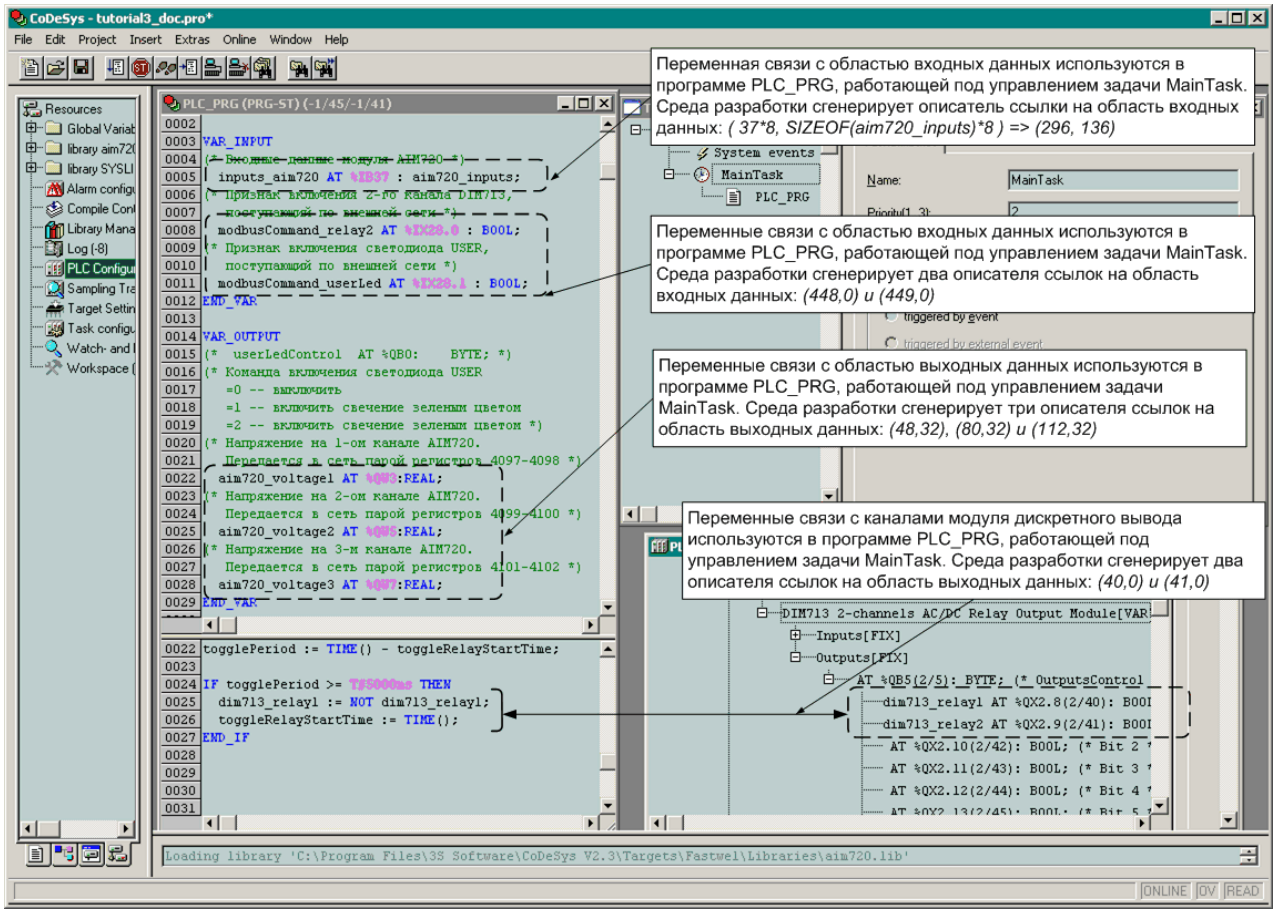

Рис. 40. Принцип формирования описателей ссылок на области входных и выходных данных приложения

Каждая ссылка на образ процесса используется средой исполнения контроллера для связывания циклических и ациклических задач с образом процесса непосредственно перед запуском приложения.

Перед началом связывания по 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, и называемых 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 (AT %I) образа процесса, размещаются компилятором во входном сегменте данных, а переменные, ссылающиеся на адреса в выходной области (AT %Q) – 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.

В контроллерах СРМ71х каждая циклическая задача имеет собственный сегмент входных данных.

Процесс связывания входных портов пользовательских задач состоит в задании канала ввода в качестве источника данных для операций чтения каждого порта, а местоположение и размер данных каждого порта в соответствующей части входного 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-получатели данных в канале.

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.

Во время функционирования приложения перед вызовом корневой программной единицы каждой пользовательской задачи выполняется последовательное чтение всех ее входных портов, в результате чего данные из участков входной части образа процесса копируются в соответствующие участки входного сегмента циклической или сервисной задачи. Во время чтения портов некоторой пользовательской задачи запись в канал ввода через связанные с ним выходные порты других источников данных запрещена. По завершении исполнения корневой программной единицы каждой пользовательской задачи выполняется последовательная запись во все ее выходные порты, в результате чего значения ее выходных переменных, находящиеся в выходном сегменте приложения, копируются в соответствующие участки выходной части образа процесса. Во время записи в выходные порты некоторой пользовательской задачи чтение канала вывода другими подсистемами запрещено.

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.

3.8.5. Способы организации ссылок на образ процесса

3.8.5.1. Общие сведения

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

1. Посредством декларации входных или выходных переменных, ссылающихся на адреса 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*.

В рассматриваемом учебном проекте будут использованы все три способа связи программы с окружением.

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

В конструкциях *VAR_INPUT* и *VAR_OUTPUT* секции декларации переменных программы возможно объявлять т.н. непосредственно представляемые переменные, ссылающиеся на адреса области входных или выходных данных образа процесса. Например:

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

В данном случае декларируется входная 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 адрес канала 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. 3.8.4), а отображение массивов типа BOOL не поддерживается.

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

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

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

```
VAR_INPUT
   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.

3.8.5.3. 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.

Примеры создания символических имен в дереве конфигурации приложения уже рассмотрены в пп. 3.6.2.3, 3.7.2.3, 3.7.3.3 настоящего документа. В качестве упражнения читателю предлагается самостоятельно создать символическое имя *userLedControl* для канала с адресом %QB0 в конфигурации контроллера.

Первый недостаток данного способа состоит в том, что символические имена исчезают при удалении объектов, которым они принадлежат, из конфигурации контроллера.

Второй недостаток заключается в том, что символические имена не могут использоваться для доступа к подобластям образа процесса, покрывающим каналы нескольких объектов конфигурации контроллера.

3.8.6. Использование ресурса VAR_CONFIG

Функциональные блоки могут включать в себя декларации входных и выходных переменных с недоопределенными ссылками на образ процесса в форме AT %I* или AT %Q*. Указанные ссылки должны быть доопределены в проекте в ресурсе *VAR_CONFIG* при использовании подобных блоков в приложении. Библиотеки поддержки платформы Fastwel I/O включают в себя функциональные блоки обработки данных от модулей ввода-вывода (с суффиксом *_DIRECT*), которые могут рассматриваться в качестве примеров использования недоопределенных ссылок на образ процесса.

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
```

3.8.7. Функциональные требования к прикладной программе контроллера

Как указывалось в п. 3.2.2, прикладная программа контроллера в данном учебном проекте должна обеспечивать выполнение следующих функций:

1. Чтение каналов модуля аналогового ввода.
2. Преобразование считанных значений в формате аналого-цифрового преобразователя в значения напряжения в формате с плавающей точкой.
3. Периодическое включение и отключение первого канала модуля дискретного вывода. Пусть период переключения будет равен 5 с.
4. Передача преобразованных значений на каналах модуля аналогового ввода по сети.
5. Включение/выключение второго канала модуля дискретного вывода по командам, поступающим в контроллер по сети.
6. Включение и выключение индикатора USER на передней панели контроллера по команде, поступающей по сети.
7. Прерывистое свечение индикатора USER красным цветом при отсутствии связи по сети.

3.8.8. Описание прикладного алгоритма

Жизненный цикл приложения контроллера в рассматриваемом учебном проекте (и не только) можно характеризировать следующими основными состояниями:

1. Первоначальный запуск после включения питания до начала циклического исполнения единственной задачи, добавленной в учебный проект при выполнении указаний п. 3.4 настоящего документа.

В данном состоянии среда исполнения автоматически присваивает начальные значения всем внутренним переменным всех программ, которые заданы пользователем в секциях деклараций переменных в каждой программе и функциональном блоке.
Кроме того, в данном состоянии требуется установить начальные значения на выходных каналах модулей ввода-вывода и исходящих коммуникационных объектов внешней сети. Установка начальных значений в учебном проекте была выполнена при следовании указаниям пп. 3.6.2.3–3.6.2.4 настоящего документа путем обработки системного события *OnPowerOn* функцией *InitOutputs()*.

2. Циклическое исполнение единственной задачи.
В учебном проекте алгоритм весьма прост: в начале каждого цикла вызывается экземпляр функционального блока, преобразующего значения на входных каналах модуля аналогового ввода, выходные переменные данного функционального блока копируются в выходные переменные, ссылающиеся на адреса каналов исходящих коммуникационных объектов внешней сети. Далее предыдущее принятое значение счетчика команд сравнивается с текущим, и, при неравенстве одного другому, значение кода цвета индикатора USER и состояние второго канала модуля дискретного вывода, принятые по сети, копируются в соответствующие глобальные выходные переменные, после чего текущее принятое значение счетчика команд копируется в предыдущее. Затем вызывается экземпляр функционального блока BLINK из библиотеки UTIL.LIB, который формирует текущее состояние на первом канале модуля дискретного вывода. Наконец, вызывается экземпляр пользовательского функционального блока, контролирующего наличие связи по сети, на основе булевого значения выходной переменной которого формируется периодическое свечение красным цветом и погасание индикатора USER при отсутствии признаков сетевого обмена.

3. Повторный запуск приложения после того, как пользователь внес в него изменения и загрузил из среды CoDeSys, и до начала циклического исполнения единственной задачи. Поведение контроллера в данном состоянии зависит от значения параметра *Fastwel I/O System Configuration:HotUpdateDisabled*. Если данный параметр имеет значение *Yes* (горячее обновление запрещено), то вновь загруженное приложение запускается так, как будто запуск происходит после включения питания контроллера. В случае, если данный параметр установлен в *No* (горячее обновление разрешено), то среда исполнения приложения предпринимает попытку безударного перехода на новое приложение с сохранением данных предыдущего: 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.

3.8.9. Реализация алгоритма

3.8.9.1. Добавление библиотек функциональных блоков

Для добавления библиотек AIM729.LIB и UTIL.LIB в конфигурацию проекта:

1. Щелкните на вкладке **Resources** в левой части главном окне CoDeSys, после чего дважды щелкните на значке ресурса **Library Manager** в списке ресурсов проекта. На экран будет выведено окно менеджера библиотек **Library Manager**, показанное на рис. 41.

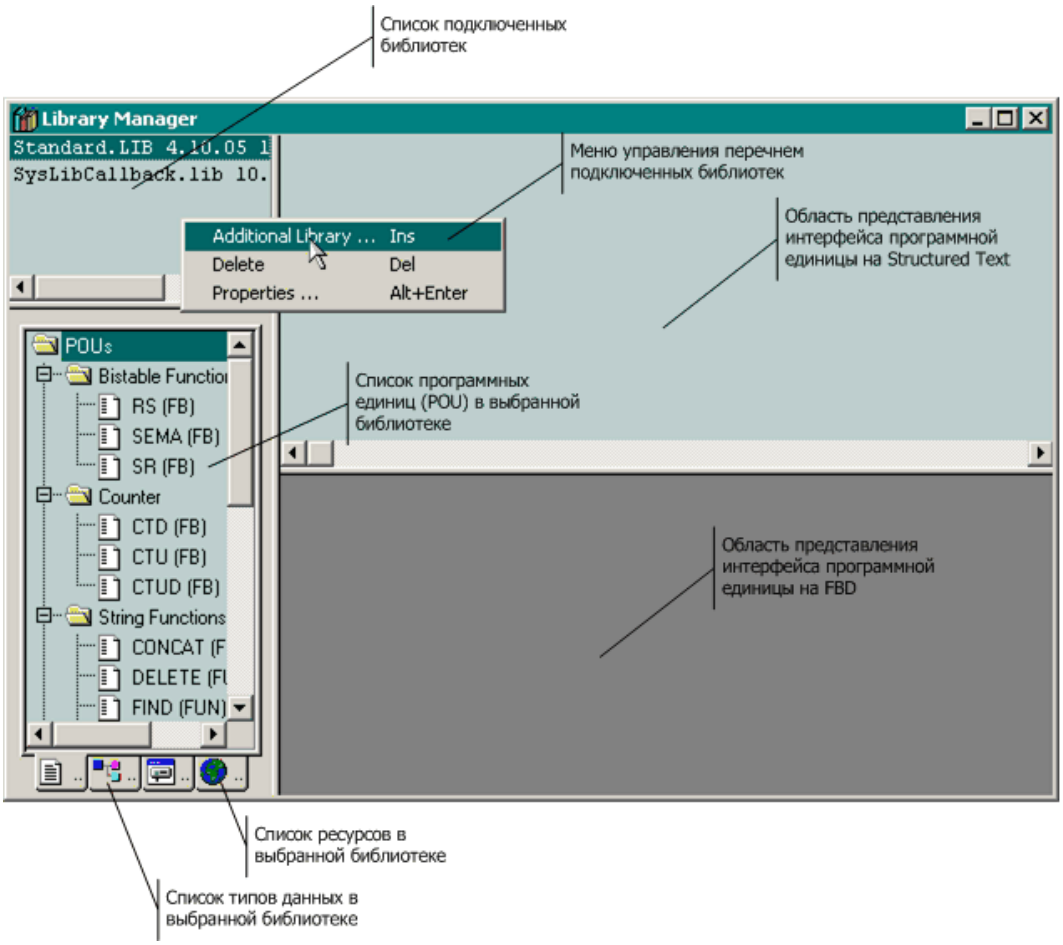

Рис. 41. Окно ресурса Library Manager

2. Щелкните правой кнопкой мыши в списке подключенных библиотек и выберите команду **Additional Library** в появившемся контекстном меню. В диалоговой панели **Open** переместитесь в подкаталог *Libraries* каталога установки файлов адаптации CoDeSys для Fastwel I/O (по умолчанию: *C:\Program Files\3S Software\CoDeSys V2.3\Targets\Fastwel*) и дважды щелкните на имени добавляемой библиотеки, относящейся к используемому модулю аналогового ввода (в данном случае – AIM729.LIB). Название библиотеки появится в списке подключенных библиотек.
3. Таким же образом добавьте библиотеку UTIL.LIB, которая находится в подкаталоге *Library* каталога установки среды CoDeSys (по умолчанию: *C:\Program Files\3S Software\CoDeSys V2.3*). Название библиотеки также появится в списке подключенных библиотек.
4. Сохраните проект (**File–Save** в главном меню) и закройте окно ресурса **Library Manager** нажатием сочетания клавиш Ctrl–F4.

3.8.9.2. Создание функционального блока контроля наличия связи по сети

Функциональный блок контроля наличия связи по сети, вызываемый из программы *PLC_PRG*, должен будет один раз в 2 секунды проверять, изменился счетчик сетевых операций с момента предыдущей проверки. Если счетчик не изменился, принимается решение об отсутствии связи по сети.

Доступ к счетчику сетевых операций в контроллерах CPM71x можно получить через следующие диагностические каналы по входному адресу %IB2577:

CPM711 – *CAN Port–CANopen Interface–OperationsCount*

CPM712 – *Serial Port–Modbus Serial Slave–TransactionsCount*

CPM713 – *Ethernet Multiprotocol Port–Modbus TCP Slave–TransactionsCount*.

1. Щелкните левой кнопкой мыши на вкладке **POUs** в левой части главного окна CoDeSys, щелкните правой кнопкой внутри списка программных единиц и выберите команду **Add object** в контекстном меню. На экран монитора будет выведена диалоговая панель **New POU**.

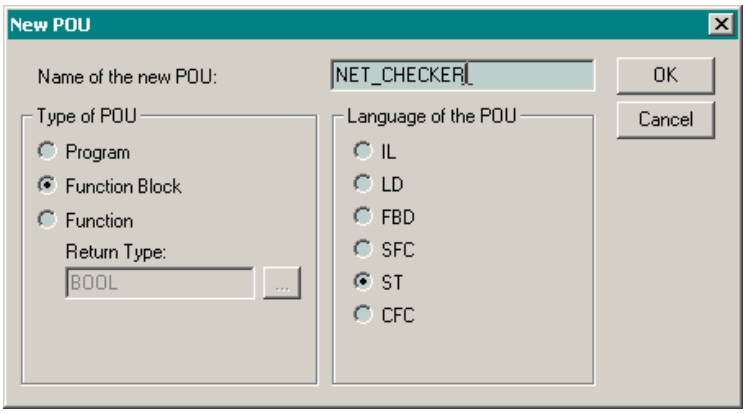

Рис. 42. Диалоговая панель New POU создания программной единицы при создании функционального блока

2. Выберите опцию **Type of POU: Function Block**, введите имя блока *NET_CHECKER*, как показано на рис. 42, и нажмите кнопку **OK**. Название созданного функционального блока будет добавлено в список программных единиц, а в области главного окна CoDeSys справа появится окно редактора исходного текста с заготовкой блока, как показано на рис. 43.

3. Для контроллеров СРМ711/712/713 в окне редактора исходного текста блока введите декларации переменных блока и реализацию алгоритма блока, как показано на рис. 44. Входная переменная блока *operationsCounter* представляет недоопределенную ссылку на область входных данных образа процесса, которая должна быть доопределена для каждого экземпляра данного функционального блока в ресурсе *VAR_CONFIG* следующим образом:

```
VAR_CONFIG
(*************************************************************
   Входная переменная operationsCounter экземпляра блока netChecker,
   который принадлежит программе PLC_PRG, ссылается на адрес %IB2577 образа
   процесса
*************************************************************)
   PLC_PRG.netChecker.operationsCounter AT %IB2577 : DWORD;
END_VAR
```

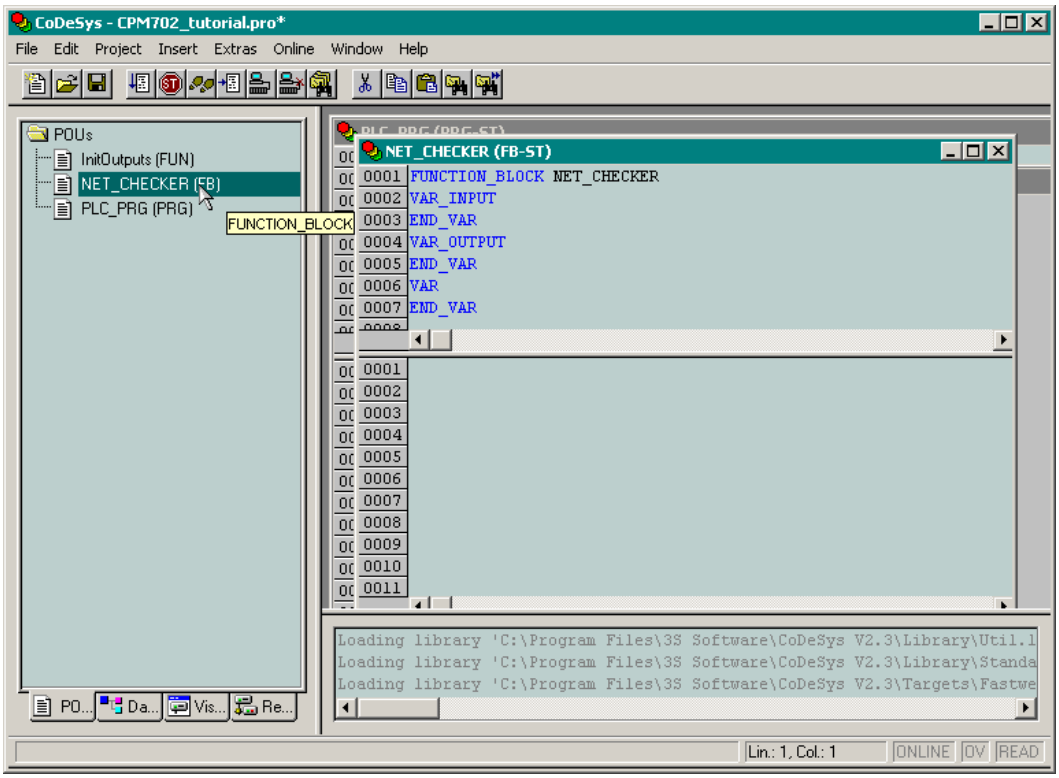

Рис. 43. Главное окно CoDeSys после создания функционального блока

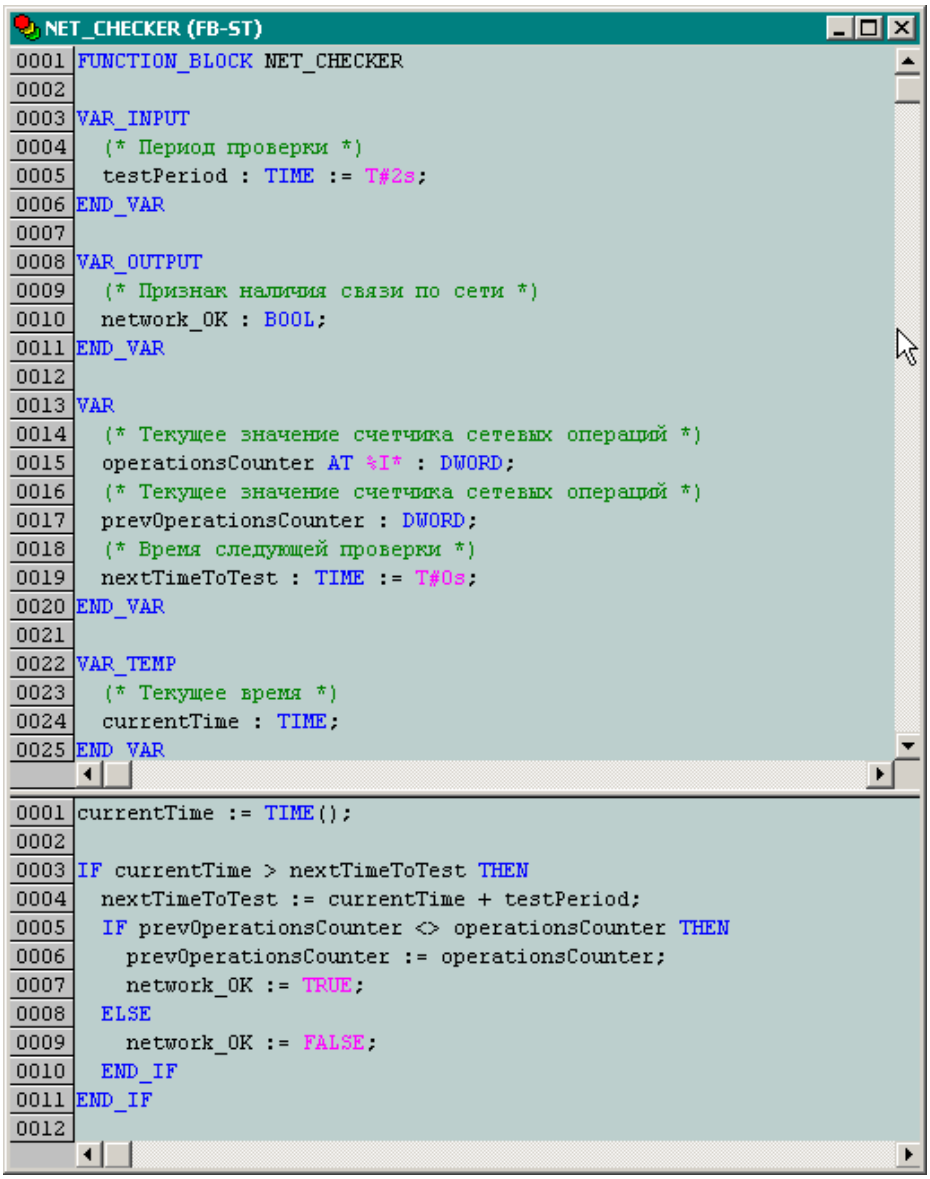

Рис. 44. Исходный текст функционального блока контроля наличия связи

Исходный текст блока *NET_CHECKER* для контроллеров СРМ711/702/703 выглядит следующим образом:

```
FUNCTION_BLOCK NET_CHECKER
(* Проверка наличия связи по сети для контроллеров CPM711/702/703 *)
VAR_INPUT
   (* Период проверки *)
   testPeriod : TIME := T#2s;
END_VAR

VAR_OUTPUT
   (* Признак наличия связи по сети *)
   network_OK : BOOL;
END_VAR

VAR
   (* Текущее значение счетчика сетевых операций *)
   operationsCounter AT %I* : DWORD;
   (* Текущее значение счетчика сетевых операций *)
   prevOperationsCounter : DWORD;
   (* Время следующей проверки *)
   nextTimeToTest : TIME := T#0s;
END_VAR

VAR_TEMP
   (* Текущее время *)
   currentTime : TIME;
END_VAR
```

```
(************ Реализация начинается здесь ************)
(* Берем текущее время *)
currentTime := TIME();
IF currentTime >= nextTimeToTest THEN
   (* Если пора проверять наличие связи *)
   (* Вычисляем следующий момент времени проверки *)
   nextTimeToTest := currentTime + testPeriod;
   IF prevOperationsCounter <> operationsCounter THEN
          (* Если кол-во сетевых операций изменилось с момента последней
             проверки, считаем, что связь есть    *)
          prevOperationsCounter := operationsCounter;
          network_OK := TRUE;
   ELSE
          (* В противном случае -- связи нет *)
          network_OK := FALSE;
   END_IF
END_IF
END_FUNCTION_BLOCK
```

3.8.9.3. Разработка программы PLC_PRG

Для начала добавим перечислимый тип, посредством которого кодируются состояния светодиодного индикатора USER:

1. Щелкните на вкладке **Data types** в главном окне CoDeSys, щелкните правой кнопкой мыши в пустом списке пользовательских типов, выберите команду **Add object** в контекстном меню и введите имя создаваемого типа *USER_LED_COLOR* в поле **Name of the new data type** появившейся диалоговой панели **New data type**, после чего нажмите кнопку **OK**. Имя типа появится в списке пользовательских типов, а в области главного окна CoDeSys справа появится окно редактора исходных текстов с заготовкой декларации типа *USER_LED_COLOR*.
2. Определите создаваемый тип следующим образом:

```
(* Коды цвета индикатора USER *)
TYPE USER_LED_COLOR :
(
   (* Индикатор выключен *)
   LED_OFF           := 0,
   (* Индикатор светится зеленым цветом *)
   LED_GREEN         := 1,
   (* Индикатор светится красным цветом *)
   LED_RED           := 2
);
END_TYPE
```

Окно редактирования типа представлено на рис. 45.

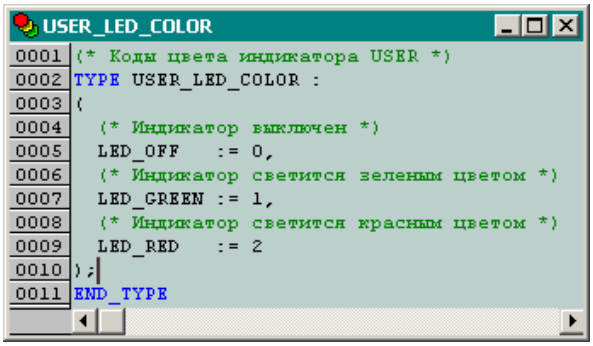

Рис. 45. Исходный текст декларации типа USER_LED_COLOR

Откройте окно исходного текста программы *PLC_PRG* и введите следующий исходный текст:

```
PROGRAM PLC_PRG
(* Основная программа учебного проекта для контроллеров CPM711-713 *)
(* Внутренние переменные PLC_PRG, сохраняющие значения между вызовами *)
VAR
   (* Первый экземпляр функционального блока BLINK из библиотеки util.lib,
      посредством которого осуществляется переключение 1-го канала модуля DIM712 *)
   relay1_toggler : BLINK;
   (* Экземпляр функционального блока NET_CHECKER, который проверяет наличие связи
   по сети. Адрес входной переменной operationsCounter доопределен в
   ресурсе VAR_CONFIG *)
```

```
    netChecker : NET_CHECKER;
    (* Второй экземпляр функционального блока BLINK из библиотеки util.lib,
       посредством которого осуществляется прерывистое свечение индикатора USER
       при отсутствии связи *)
    netFailureBlinker : BLINK;
    (* Экземпляр функционального блока AIM729_DIRECT, который преобразует значения АЦП
       модуля AIM729 в значения напряжения в формате REAL. Адрес входной переменной
       доопределен в ресурсе VAR_CONFIG *)
    analogConverter : AIM729_DIRECT;
    (* Последнее значение кода цвета, полученное по сети, используемое для включения
       правильного цвета после пропадания и восстановления связи *)
    lastUserLedColor : USER_LED_COLOR;
END_VAR
VAR_OUTPUT
    (* Значения напряжения, выводимые в сеть в контроллерах CPM711/702/703 *)
    netOut_voltages AT %QW3 : ARRAY [0..1] OF REAL;
    (* Значения напряжения, выводимые в сеть в контроллере  *)
    (* netOut_voltages AT %QB2305 : ARRAY [0..1] OF REAL; *)
END_VAR

(************ Реализация начинается здесь ************)
(* Переключаем 1-й канал модуля DIM712 с периодом 1 с*)
relay1_toggler(ENABLE:= TRUE, TIMELOW:= T#2500ms, TIMEHIGH:= T#2500ms, OUT=>
relay_out1);
(* Вызываем преобразователь показаний в формате АЦП *)
analogConverter;
(* Выводим значения напряжения в сеть *)
netOut_voltages[0] := analogConverter.outputs.vout0;
netOut_voltages[1] := analogConverter.outputs.vout1;
 (* Проверяем наличие связи по сети*)
netChecker(operationsCounter:= , testPeriod:= T#2000ms, network_OK=> );
(* вызов для
   netChecker;  *)
IF NOT netChecker.network_OK THEN
    (* Если связи нет, "мигаем" красным цветом путем формирования меандра
    с периодом 200 мс *)
    netFailureBlinker(ENABLE:= TRUE, TIMELOW:= T#100ms, TIMEHIGH:= T#100ms, OUT=> );
    IF netFailureBlinker.OUT THEN
        userLedControl := INT_TO_BYTE(LED_RED);
    ELSE
        userLedControl := INT_TO_BYTE(LED_OFF);
    END_IF
ELSE
    IF prevNetCommandCounter <> netCommandCounter THEN
            (* Если принятый счетчик команды изменился, выполняем команду, запомнив
               принятое значение счетчика *)
            prevNetCommandCounter := netCommandCounter;
            (* Запоминаем код цвета *)
            lastUserLedColor := (userLed_cmd AND 3);
            (* Выводим команду управления реле *)
            relay_out2 := relay2_cmd;
    END_IF
    (* Зажигаем USER последним запомненным цветом или выключаем вовсе *)
    userLedControl := INT_TO_BYTE(lastUserLedColor);
END_IF
END_PROGRAM
```

Окно редактирования исходного текста программы *PLC_PRG* представлено на рис. 46.

Если в настоящий момент выполнить трансляцию проекта, то в области вывода сообщений транслятора появится следующее сообщение об ошибке:

```
Error 3500: No 'VAR_CONFIG' for 'PLC_PRG.analogConverter.inputs'
```

Данное сообщение свидетельствует о том, что вход *inputs* экземпляра *analogConverter* блока *AIM729_DIRECT*, принадлежащего программе *PLC_PRG*, ссылается неопределенный адрес.

Доопределить адрес необходимо в окне ресурса *VAR_CONFIG*:

1. Щелкните на вкладке **Resources** и в списке ресурсов раскройте группу ресурсов *Global Variables*, после чего щелкните на имени ресурса *Variable_Configuration*. На экране монитора появится окно редактирования ссылок недоопределенных входных и выходных переменных.

2. Введите в окно редактирования ссылку на адрес первого канала описания модуля AIM729, как показано на рис. 53.

PLC_PRG (PRG-ST)

```
0001 PROGRAM PLC_PRG
0002 (* Основная программа учебного проекта для контроллеров CPM711-713 *)
0003 (* Внутренние переменные PLC_PRG, сохраняющие значения между вызовами *)
0004 VAR
0005   (* Первый экземпляр функционального блока BLINK из библиотеки util.lib,
0006      посредством которого осуществляется переключение 1-го канала модуля DIM712 *)
0007   relay1_toggler : BLINK;
0008   (* Экземпляр функционального блока NET_CHECKER, который проверяет наличие связи по сети.
0009      Адрес входной переменной operationsCounter доопределен в ресурсе VAR_CONFIG  *)
0010   netChecker : NET_CHECKER;
0011   (* Второй экземпляр функционального блока BLINK из библиотеки util.lib,
0012      посредством которого осуществляется прерывистое свечение индикатора USER при отсутствии связи *)
0013   netFailureBlinker : BLINK;
0014   (* Экземпляр функционального блока AIM729_DIRECT, который преобразует значения АЦП
0015      модуля AIM729 в значения напряжения в формате REAL.
0016      Адрес входной переменной доопределен в ресурсе VAR_CONFIG  *)
0017   analogConverter : AIM729_DIRECT;
0018   (* Последнее значение кода цвета, полученное по сети, используемое для включения
0019      правильного цвета после пропадания и восстановления связи *)
0020   lastUserLedColor : USER_LED_COLOR;
0021 END_VAR
0022 VAR_OUTPUT
0023   (* Значения напряжения, выводимые в сеть в контроллерах CPM711/712/713 *)
0024   netOut_voltages AT %QB2561 : ARRAY [0..1] OF REAL;
0025 END_VAR
```

```
0001 (* Переключаем 1-й канал модуля DIM712 с периодом 1 с*)
0002 relay1_toggler(ENABLE:= TRUE, TIMELOW:= T#2500ms, TIMEHIGH:= T#2500ms, OUT=> relay_out1);
0003 (* Вызываем преобразователь показаний в формате АЦП *)
0004 analogConverter;
0005 (* Выводим значения напряжения в сеть *)
0006 netOut_voltages[0] := analogConverter.outputs.vout0;
0007 netOut_voltages[1] := analogConverter.outputs.vout1;
0008 (* Проверяем наличие связи по сети*)
0009 netChecker(testPeriod:= T#2000ms, network_OK=> );
0010
0011 IF NOT netChecker.network_OK THEN
0012   (* Если связи нет, "мигаем" красным цветом путем формирования меандра с периодом 200 мс *)
0013   netFailureBlinker(ENABLE:= TRUE, TIMELOW:= T#100ms, TIMEHIGH:= T#100ms, OUT=> );
0014   IF netFailureBlinker.OUT THEN
0015     userLedControl := INT_TO_BYTE(LED_RED);
0016   ELSE
0017     userLedControl := INT_TO_BYTE(LED_OFF);
0018   END_IF
0019 ELSE
0020   IF prevNetCommandCounter <> netCommandCounter THEN
0021     (* Если принятый счетчик команды изменился, выполняем команду, запомнив принятое значение счетчика *)
0022     prevNetCommandCounter := netCommandCounter;
0023     (* Запоминаем код цвета *)
0024     lastUserLedColor := (userLed_cmd AND 3);
0025     (* Выводим команду управления реле *)
0026     relay_out2 := relay2_cmd;
0027   END_IF
0028   (* Зажигаем USER последним запомненным цветом или выключаем вовсе *)
0029   userLedControl := INT_TO_BYTE(lastUserLedColor);
0030 END_IF
0031
```

Рис. 46. Исходный текст программы PLC_PRG

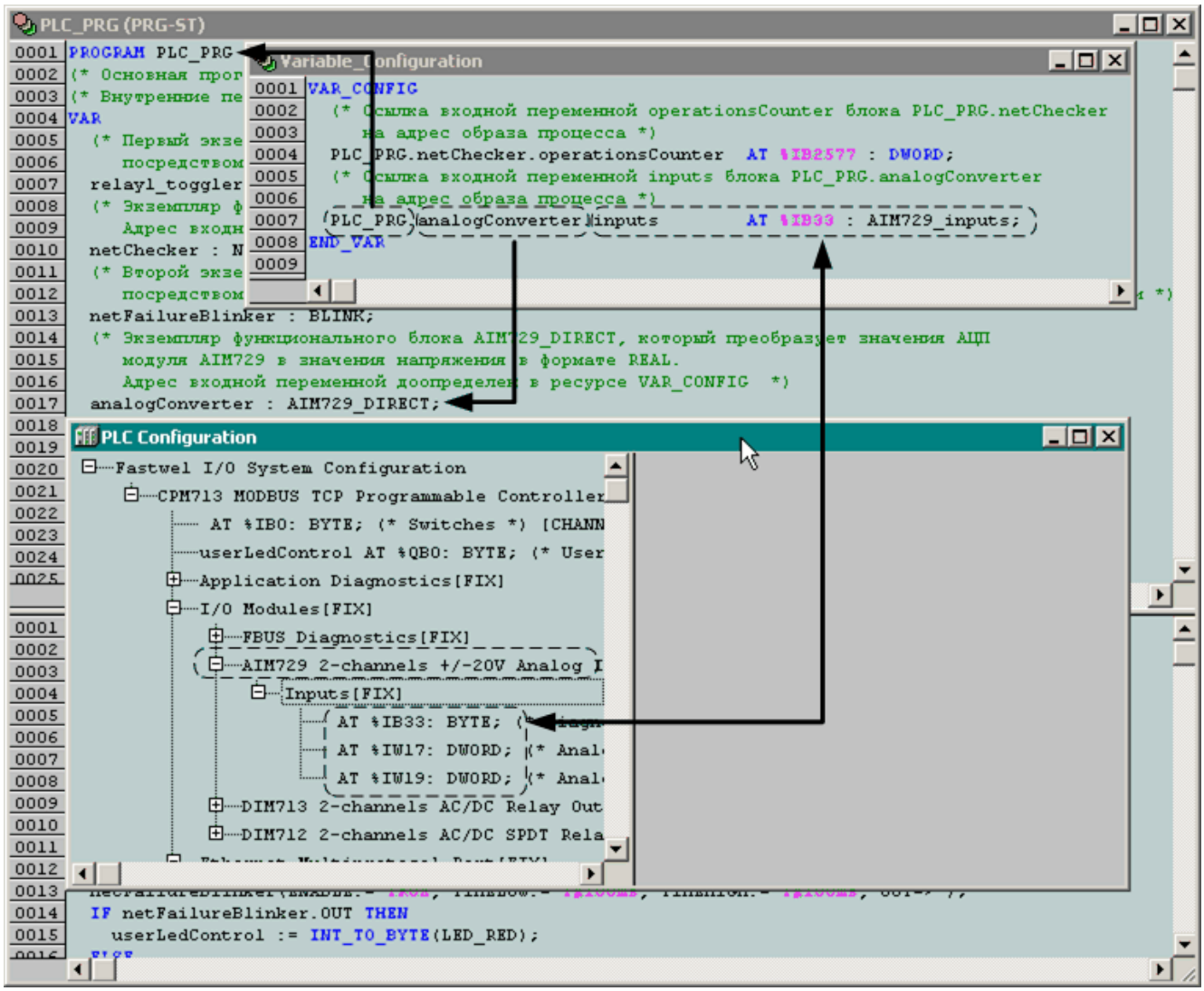

Рис. 47. Ссылка входной переменной непримитивного типа экземпляра функционального блока на три канала в описании модуля аналогового ввода

Выполните трансляцию проекта командой **Project–Rebuild all** в главном окне CoDeSys. Если при трансляции будут выведены какие-либо сообщения об ошибках (выделяются красным цветом), найдите и устраните ошибки.

При рассмотрении исходного текста программы *PLC_PRG* следует обратить внимание на следующие моменты:

1. Глобальная выходная переменная *userLedControl*, определяющая состояние индикатора USER, по сути имеет два источника данных: приложение, исполняющееся на удаленном узле сети, и внутренний алгоритм контроля наличия связи по сети. В связи с этим в программу *PLC_PRG* добавлена переменная *lastUserLedColor*, в которой сохраняется последнее значение цвета индикатора USER, поступившее по сети. При пропадании связи по сети значение выходной переменной *userLedControl* формируется непосредственно внутренним алгоритмом приложения, а при восстановлении связи и до прихода новой команды управления индикатором состояние индикатора будет восстановлено из переменной *lastUserLedColor*.
2. Глобальная выходная переменная *userLedControl* имеет тип *BYTE*, а перечислимый тип *USER_LED_COLOR* по длине эквивалентен типу *INT*. В связи с этим непосредственное управление переменной *userLedControl* при отсутствии связи по сети осуществляется с дополнительным преобразованием типов: *INT_TO_BYTE(LED_RED)*.
3. Значения напряжения на двух каналах модуля аналогового ввода выводятся в сеть путем использования выходной переменной в виде массива из двух элементов типа *REAL*. При этом ссылка на область выходных данных образа процесса организуется посредством указания непосредственного адреса в декларации переменной, как показано на рис. 48.

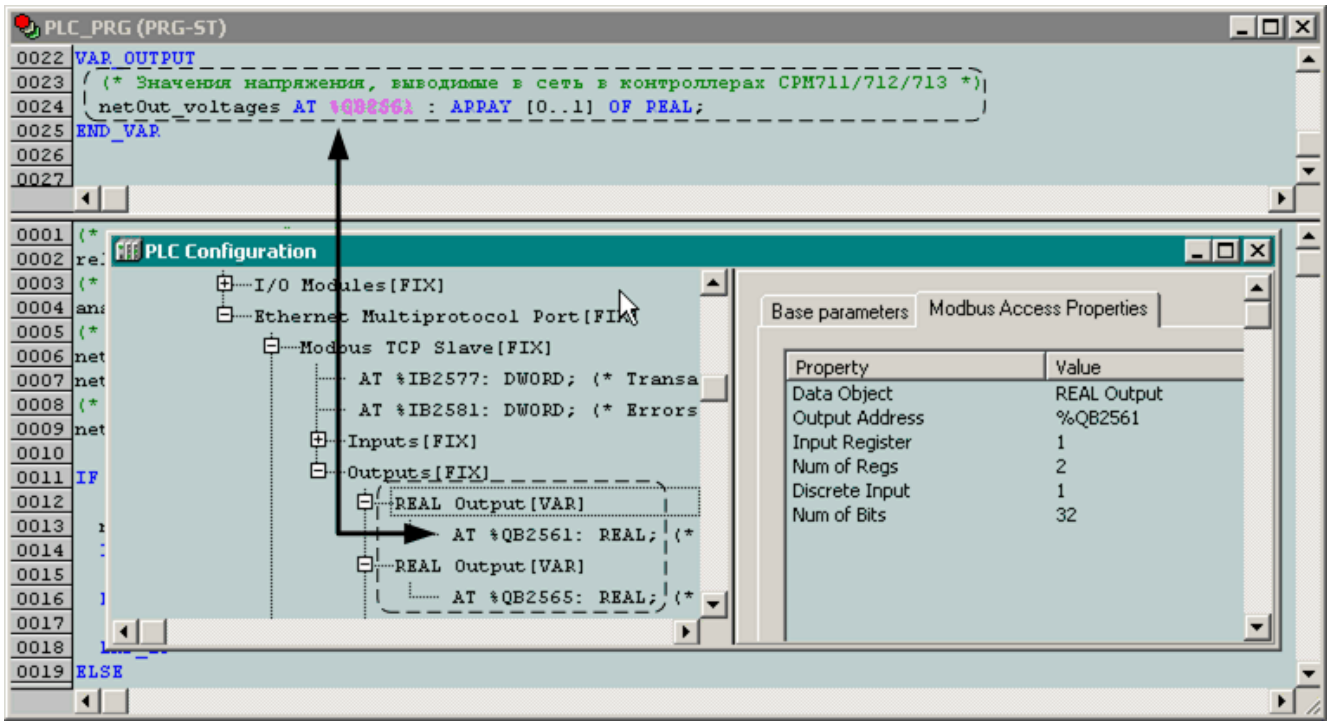

Рис. 48. Ссылка выходной переменной непримитивного типа на адрес в области выходных данных образа процесса

3.9. Загрузка приложения в контроллер

3.9.1. 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 удаленным контроллером путем выполнения команды **Online–Login**.

В настоящем документе будет рассмотрены приемы взаимодействия между средой разработки CoDeSys и контроллерами Fastwel I/O через интерфейс прямого соединения P2P.

3.9.2. Создание логического информационного канала между средой разработки и контроллером через P2P

Для создания информационного канала выполните следующие действия:

1. Соедините последовательный порт ПК с портом консоли контроллера, расположенным под пластмассовой защитной крышкой на передней панели контроллера, при помощи кабеля соединительного ACS00019.

2. Запустите среду разработки CoDeSys.
3. Выберите команду меню **Online–Communication Parameters**… На экран будет выведена диалоговая панель **Communication Parameters**.
4. Для создания логического информационного канала через последовательный порт нажмите кнопку **New** и в появившейся диалоговой панели введите имя создаваемого канала, а в списке **Device** выберите строку *P2P: Fastwel P2P driver*, как показано на рис. 49, введите имя канала в поле **Name** и закройте диалоговую панель нажатием кнопки **OK**. В древовидном списке **Channels** диалоговой панели **Communication Parameters** появится элемент, соответствующий созданному каналу, а в таблице параметров канала справа – параметры созданного канала, как показано на рис. 50.

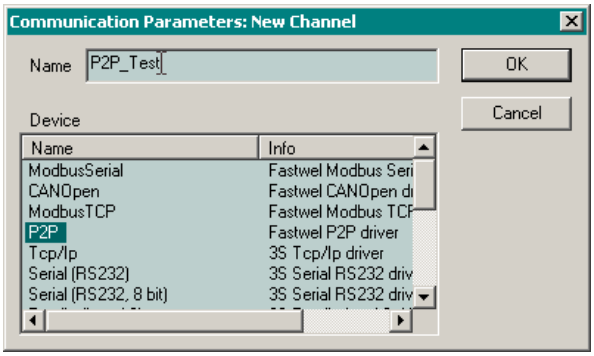

Рис. 49. Создание канала с использованием драйвера Fastwel P2P driver

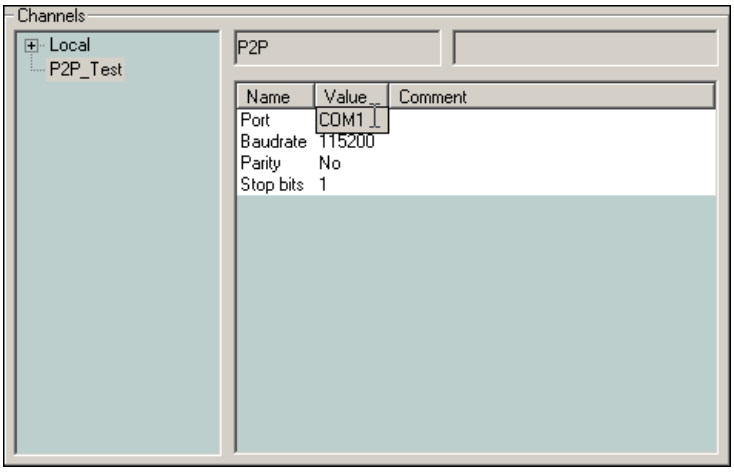

Рис. 50. Настройка параметров канала

5. Если для связи с контроллером используется последовательный порт, отличный от COM1, дважды щелкните левой кнопкой мыши над именем *COM1* в таблице параметров и клавишами ↑ ("стрелка вверх") или ↓ ("стрелка вниз") выберите требуемый последовательный порт компьютера, через который будет осуществляться взаимодействие с контроллером в режиме "точка-точка", и нажмите клавишу Enter.
6. Закройте диалоговую панель **Communication Parameters** нажатием кнопки **OK**.

3.9.3. Login

Соединение среды разработки CoDeSys с контроллером выполняется по команде меню **Online–Login**. При успешном выполнении **Login** строка состояния главного окна среды CoDeSys принимает вид, аналогичный приведенному на рис. 51.

Рис. 51. 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. 52.

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 при выполнении команды **Online–Login** не удается установить соединение с контроллером через интерфейс внешней сети, 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.

3.9.4. 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. 52.

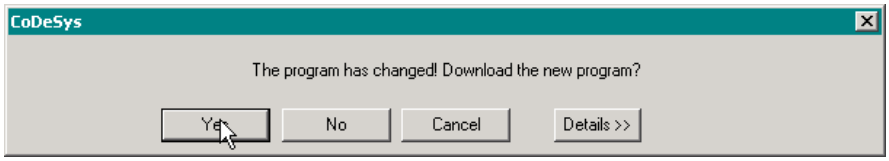

Рис. 52. Предложение загрузить программу

2. Убедитесь, что переключатель "1" контроллера выключен.
3. Нажмите кнопку **Yes**. На экран будет выведено окно, отображающее ход загрузки программы, показанное на рис. 53. 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.

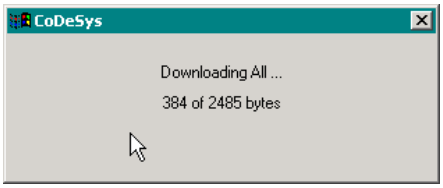

Рис. 53. Отображение хода загрузки программы

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

3.10. Функционирование приложения в контроллере

3.10.1. 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.

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) – 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*). В случае ускоренного преобразования проекта до новой версии значение данного параметра в проекте будет равным 0, поэтому в качестве истинного значения периода сервис ввода-вывода будет использовать период, определенный параметром *SampleRate* в конфигурации контроллера (**Resources–PLC Configuration**–*CPM71x ... Controller*);

зеленый цвет (прерывисто) – обмен с модулями ввода-вывода, подключенными к контроллеру, не может быть выполнен за время, заданное в конфигурации контроллера (**Resources–PLC Configuration**–*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.

3.10.2. Безопасный режим

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

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

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

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

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

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

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

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

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

Кодовые последовательности индикации об ошибках в загруженном приложении, которые формируются контроллером в безопасном режиме, представлены в табл. 1–4 п. 4.2.1.1 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713. Руководство программиста*.

Если не удается определить причину по кодовой последовательности индикации, можно выгрузить из контроллера файл normdump.txt, содержащий текстовую информацию (в кодировке Unicode) о причине последнего перехода в безопасный режим. Для выгрузки файла:

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 **Cancel**.
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. Обратитесь к разделу 8 документа *Система ввода-вывода Fastwel I/O. Контроллеры CPM711/CPM712/CPM713 Руководство программиста*. для установления причины.

3.11. Проверка работоспособности сетевых функций контроллера CPM711 (CANopen)

3.11.1. Общие сведения

Работоспособность сетевых функций контроллера CPM711 на ПК может быть проверена при помощи программного обеспечения мониторинга сети CAN, поставляемого с применяемым адаптером сети CAN, или, при использовании адаптеров фирмы PEAK-Systems Technik или IXXAT, с помощью демонстрационной версии OPC-сервера Fastwel CAN OPC Server. Далее предполагается, что в качестве адаптера сети CAN используется изделие PCAN-USB фирмы PEAK-Systems Technik и программный монитор PCANView, поставляемый в комплекте с адаптером.

3.11.2. PCANView USB

Предполагается, что драйвер адаптера PCAN-USB и программа PCANView USB были успешно установлены на ПК, а контроллер CPM711 с загруженным учебным проектом включен и исполняет прикладную программу, причем индикатор USER из-за отсутствия связи по сети светится прерывисто красным цветом.

1. Организуйте физическое соединение ПК с контроллером при помощи адаптера PCAN-USB, для чего подключите адаптер к соединителю USB ПК и к соединителю интерфейса CAN контроллера.

2. Запустите программу PCANView USB из программной группы *PCAN USB*. На экран монитора будет выведена диалоговая панель выбора адаптера и настройки параметров обмена **Connect to CAN hardware**, показанная на рис. 54.

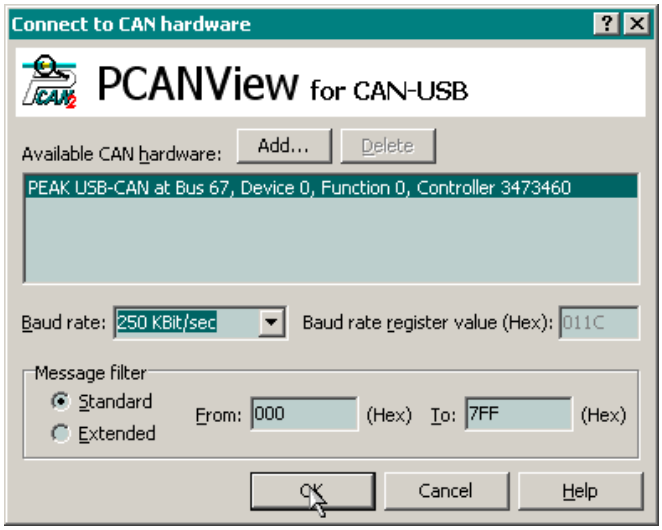

Рис. 54. Диалоговая панель выбора CAN-адаптера

3. Выберите название адаптера *PEAK USB-CAN at...* в списке **Available CAN hardware**, в поле **Baud rate** установите значение *250 KBit/sec* и нажмите кнопку **OK**. На экране появится главное окно программы PCANView USB, показанное на рис. 55.

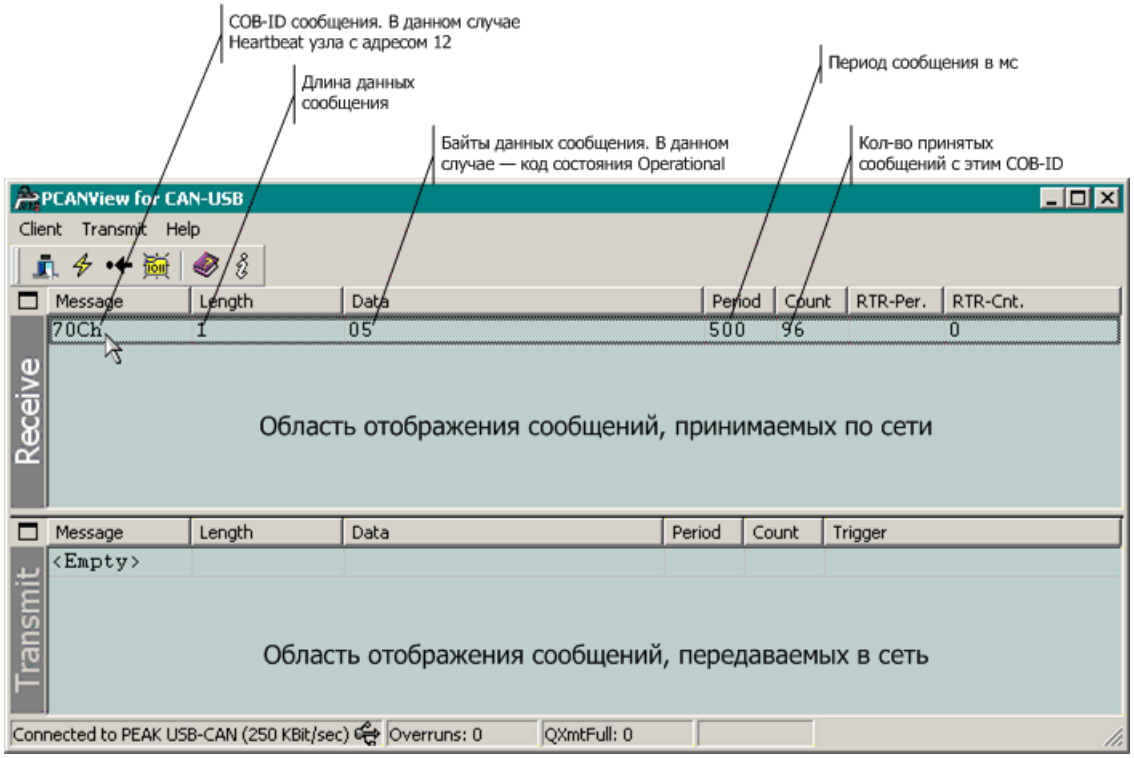

Рис. 55. Главное окно монитора сети PCANView for CAN-USB

Если физическое соединение между ПК и контроллером установлено правильно, в области отображения сообщений, принимаемых по сети (*Receive*), появится сообщение с идентификатором 70Ch, принимаемое с периодом 500 мс, и содержащее код состояния *Operational (5)* стека CANopen контроллера.

4. Контроллер не передает в сеть ни одного исходящего коммуникационного объекта, поскольку в сети отсутствует мастер синхронизации. Добавьте сообщение SYNC в область отображения сообщений, передаваемых в сеть, для чего выполните команду меню **Transmit–New** и в появившейся диалоговой панели **New transmit message** введите значения параметров сообщения, как показано на рис. 56, и нажмите кнопку **OK**, закрыв диалоговую панель **New transmit message**. Содержимое главного окна PCANView USB будет выглядеть, как показано на рис. 57, а индикатор USER контроллера прекратит "мигать" красным цветом.

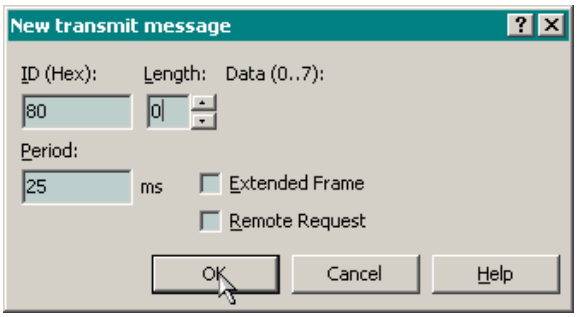

Рис. 56. Создание сообщения SYNC с периодом 25 мс

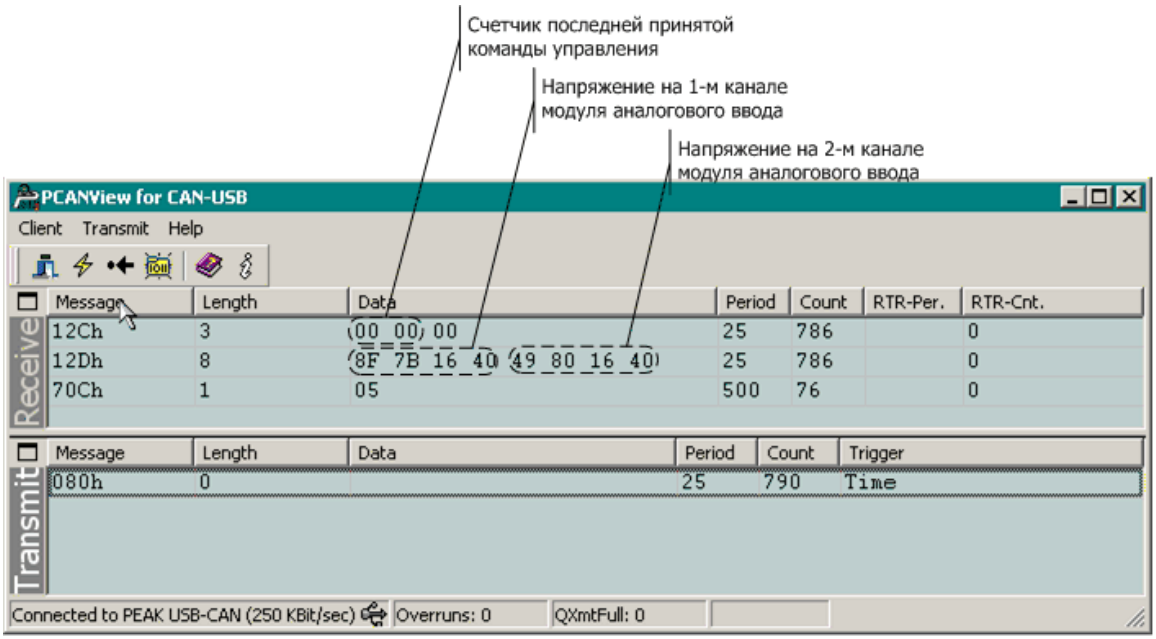

Рис. 57. Сообщения, принимаемые от контроллера

5. Для имитации команд управления, поступающих контроллеру от другого узла сети, добавьте сообщение с идентификатором 101H. Для этого выполните команду меню **Transmit–New** и в появившейся диалоговой панели **New transmit message** введите значения параметров сообщения, как показано на рис. 58, и нажмите кнопку OK. Второй канал модуля дискретного вывода немедленно включится, индикатор USER на передней панели контроллера будет светиться зеленым цветом, а в первых двух байтах сообщения 12CH, передаваемого контроллером, появится значение счетчика последней выполненной команды.

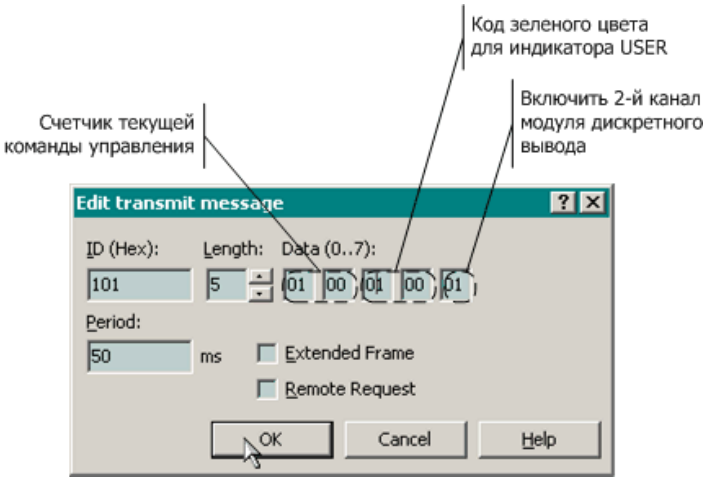

Рис. 58. Создание сообщения с CAN-ID=101H для передачи команд управления контроллеру

3.11.3. Fastwel CAN OPC Server

1. Завершите работу PVANView USB, после чего запустите Fastwel CAN OPC Server из соответствующей программной группы.
2. Выполните команду **Add–New CAN segment** в главном меню OPC-сервера. На экране монитора появится диалоговая панель настройки параметров сегмента сети CAN, показанная на рис. 59.

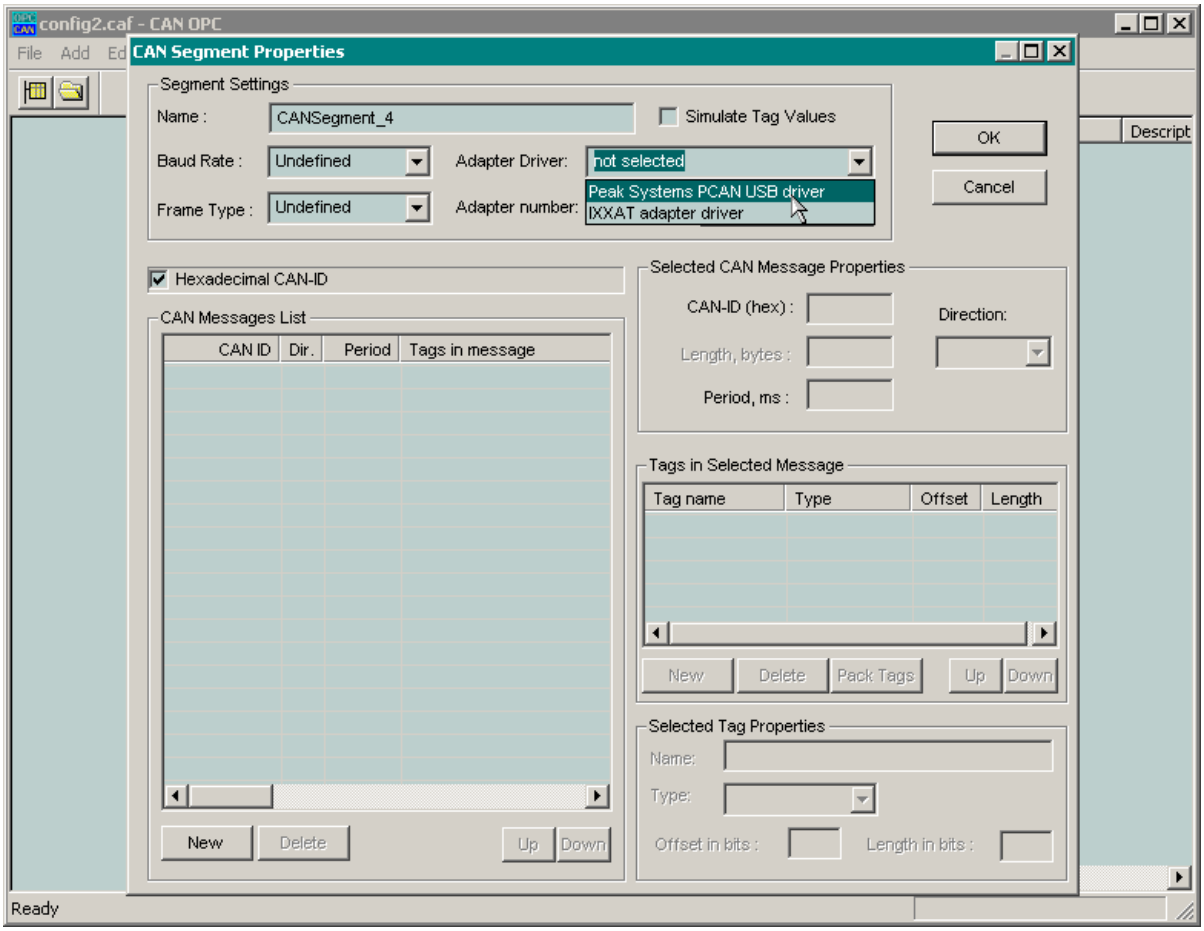

Рис. 59. Создание конфигурации сегмента сети CAN

3. Настройте параметры сегмента сети CAN, как показано на рис. 60, после чего для проверки наличия связи нажмите кнопку **CAN Monitor**. В диалоговой панели встроенного монитора сети CAN, появившейся на экране, отметьте флажок **CANopen SYNC Options–Transmit** и убедитесь в том, что контроллер передает в сеть исходящие коммуникационные объекты с идентификаторами 12CH и 12DH, как показано на рис. 61.

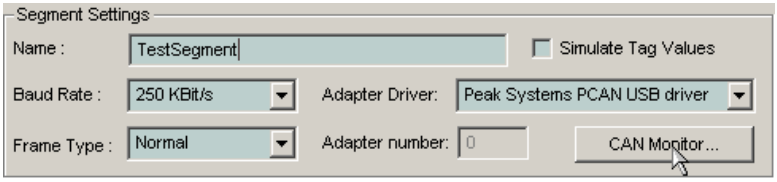

Рис. 60. Настройка параметров сегмента сети CAN

4. В диалоговой панели монитора выберите сообщения с идентификаторами 12CH и 12DH, для чего щелкните на соответствующих строках в списке принимаемых сообщений, удерживая нажатой клавишу Ctrl, после чего нажмите кнопку **Add Selected** и закройте диалоговую панель встроенного монитора нажатием кнопки **Close**. Добавленные сообщения появятся в списке **CAN Messages List**, как показано на рис. 62.

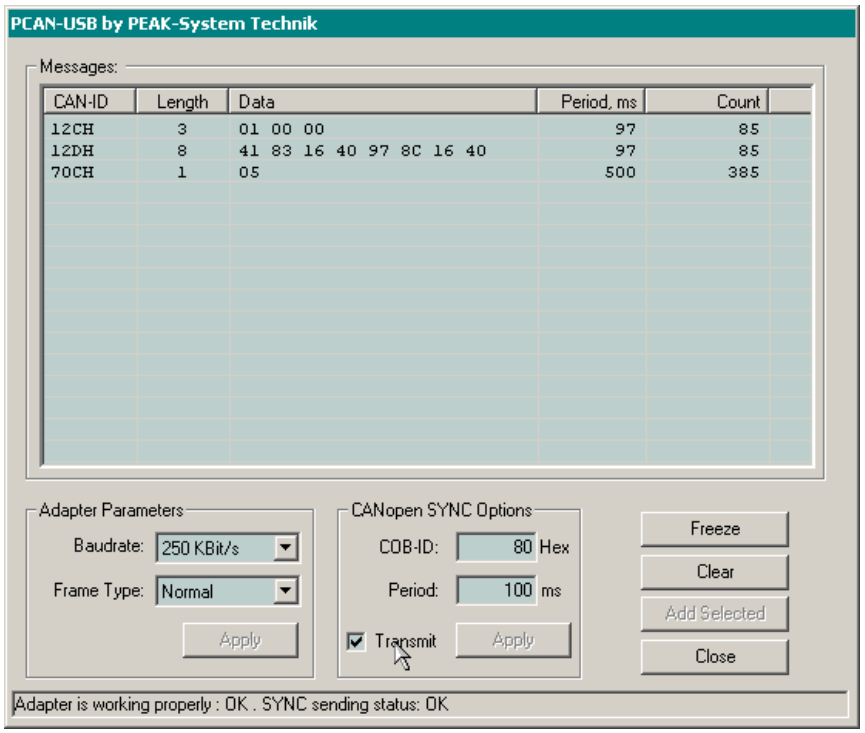

Рис. 61. Окно монитора сети CAN

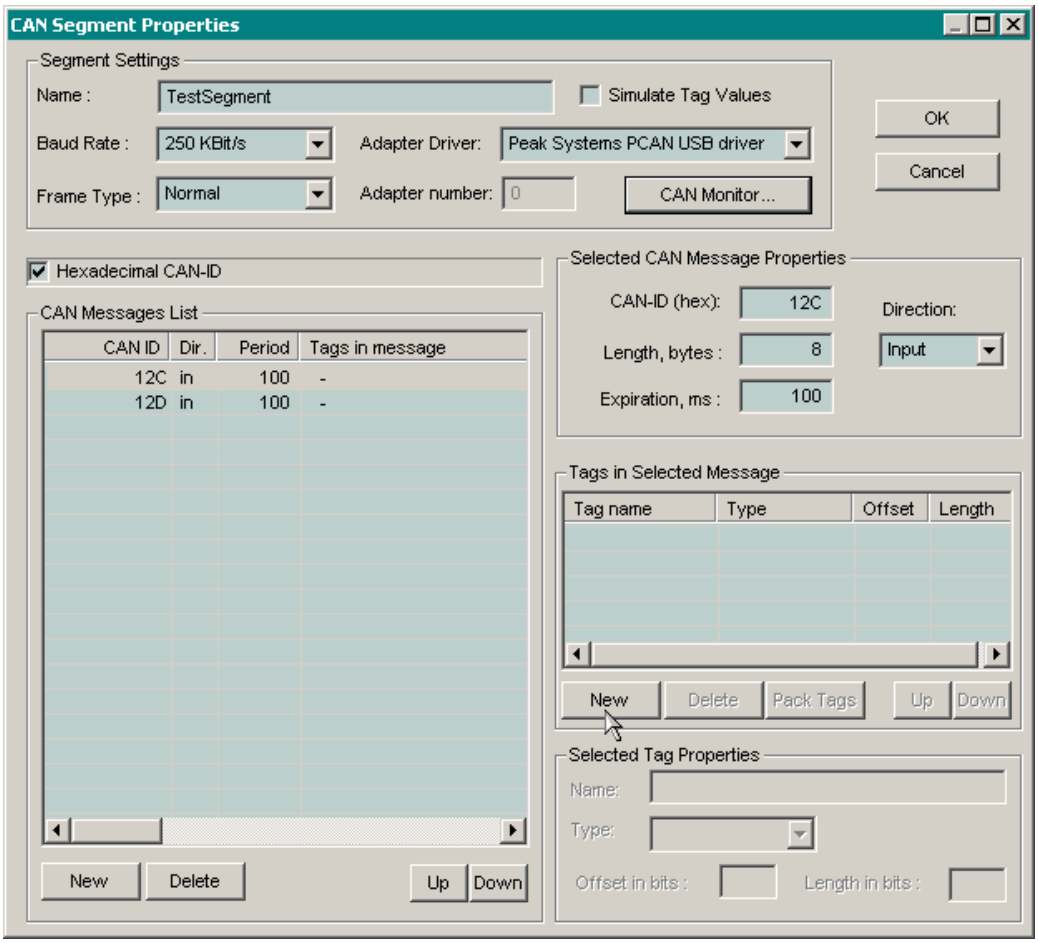

Рис. 62. Настройка параметров сегмента сети CAN

5. Выберите сообщение с CAN ID *12C* в списке **CAN Messages List**, нажмите кнопку **Tags in Selected Message–New**, а затем введите имя тега *prevNetCommandCounter* в поле **Selected Tag Properties–Name** и нажмите кнопку **Apply**. Содержимое диалоговой панели свойств сегмента сети CAN примет вид, показанный на рис. 63

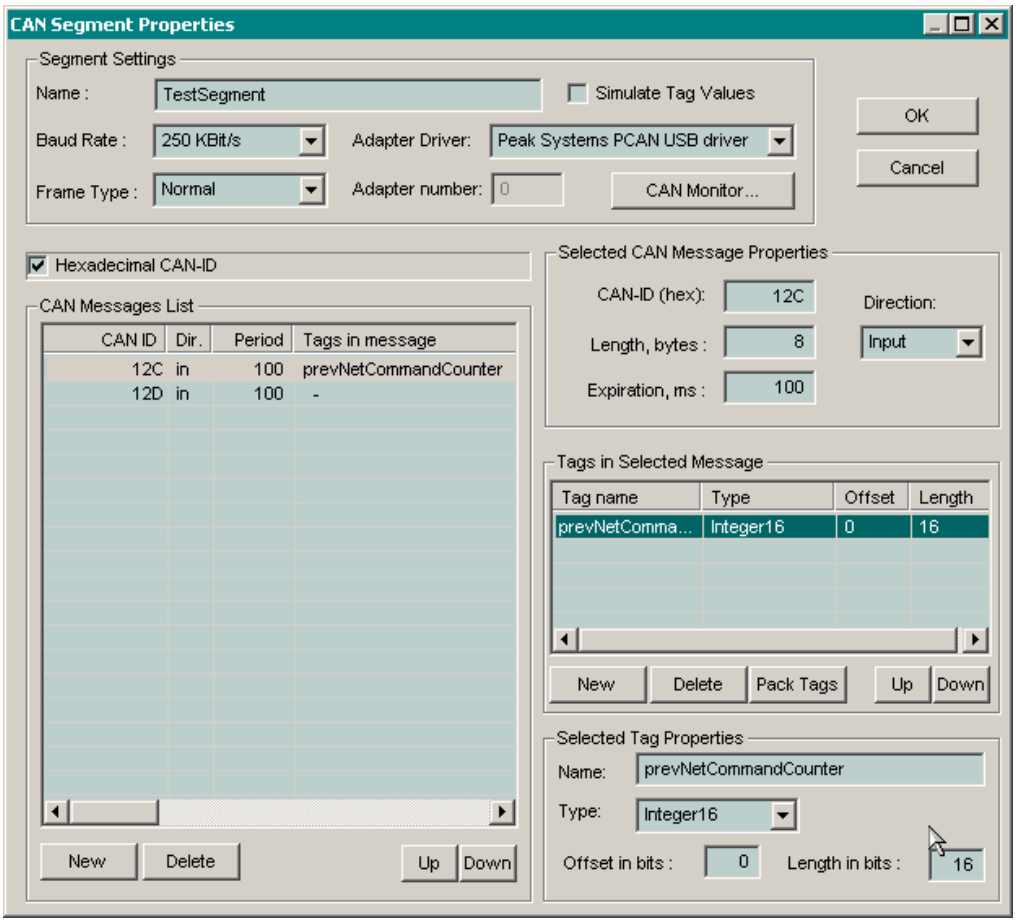

Рис. 63. Добавление тега prevNetCommandCounter

6. Для сообщения с идентификатором 12DH добавьте два тега типа *Real* с именами *netOut_voltages0* и *netOut_voltages1*, первый из которых имеет смещение 0, а второй – смещение 32 в поле данных сообщения 12DH, как показано на рис. 64.

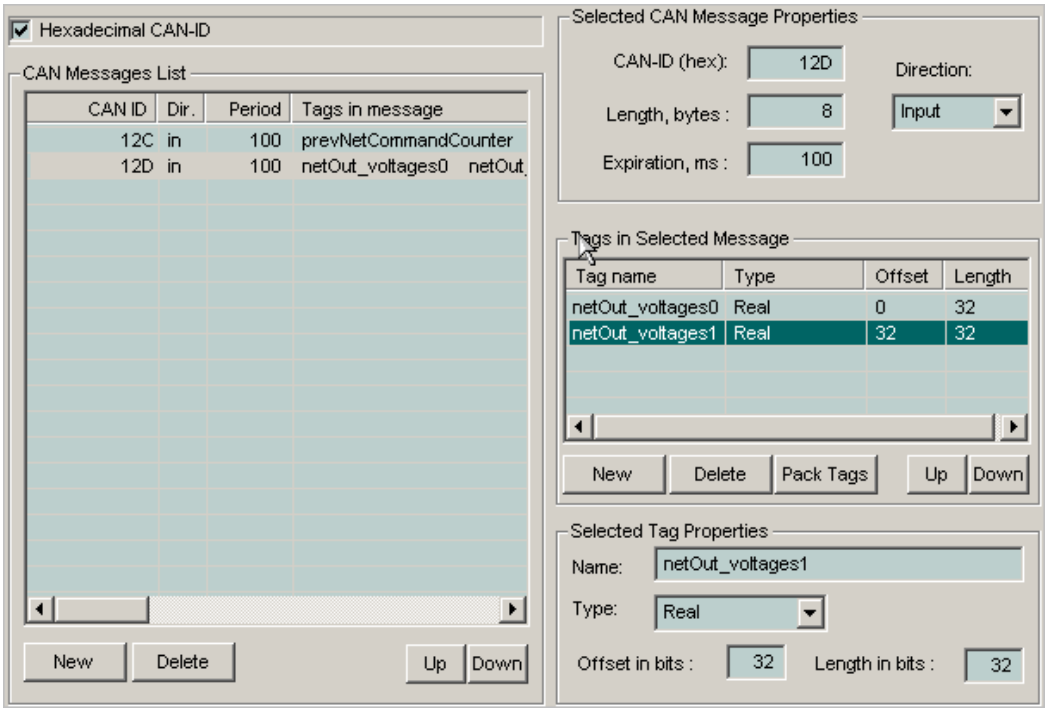

Рис. 64. Добавление тегов для значений каналов модуля аналогового ввода

7. Поскольку в сети отсутствует мастер синхронизации, добавьте передаваемое сообщение SYNC с идентификатором 80H в список **CAN Messages List**. Для этого нажмите кнопку **CAN Messages List–New**, выберите опцию *Output* в выпадающем списке **Selected CAN Messages Preoperties–Direction**, введите идентификатор *80* в поле **CAN-ID (hex)**, длину *0* в поле **Length, bytes** и период передачи *50* в поле **Send Period, ms**, как показано на

рис. 65, и нажмите кнопку **Apply**. Информация о сообщении с идентификатором 80H появится в списке **CAN Messages List**.

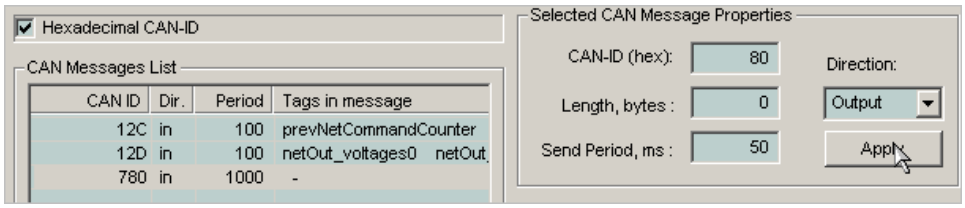

Рис. 65. Добавление синхронизирующего сообщения SYNC с периодом передачи 50 мс

8. Добавьте в список **CAN Messages List** передаваемое сообщение с идентификатором 101H и создайте для него теги *netCommandCounter*, *userLED_cmd* и *relay2_cmd*, как показано на рис. 66, после чего закройте диалоговую панель **CAN Segment Properties** нажатием кнопки OK.

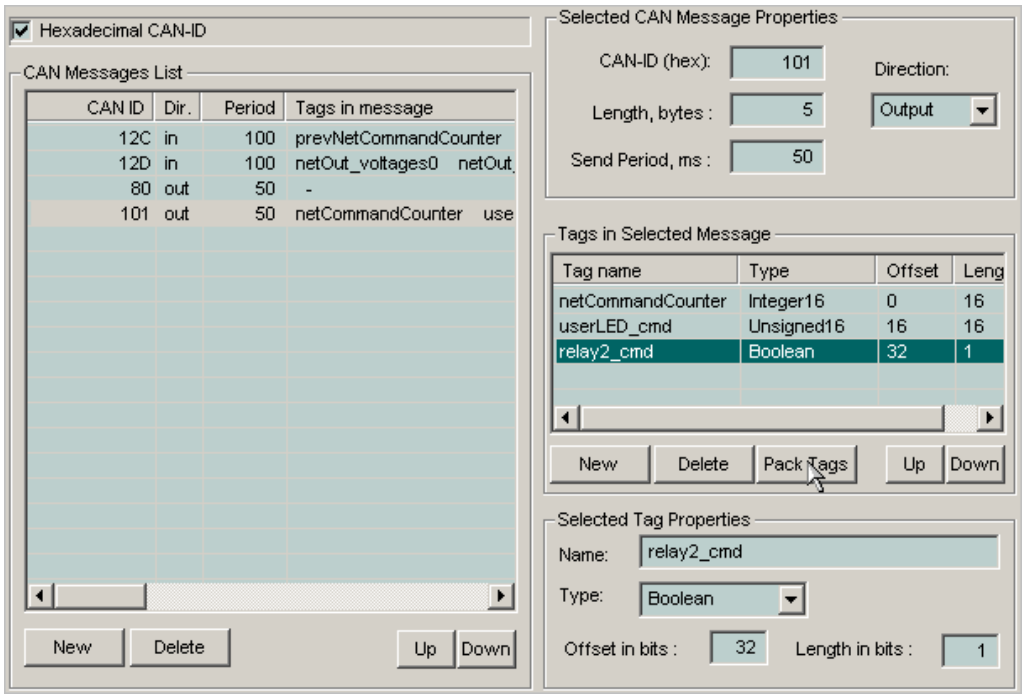

Рис. 66. Добавление тегов передачи команд управления

9. Запустите тестового клиента OPC (ярлык *OPC Client* в программной группе *Fastwel CAN OPC Server*), выполните команду **OPC–Connect** в его главном меню, выберите *Fastwel.canopc* в диалоговой панели **Select OPC Server** и нажмите кнопку **OK**.
10. Выполняйте команду **OPC–Add Item** в главном меню клиента OPC последовательно для всех только что созданных тегов OPC-сервера, в результате чего главное окно клиента будет выглядеть, как показано на рис. 67.

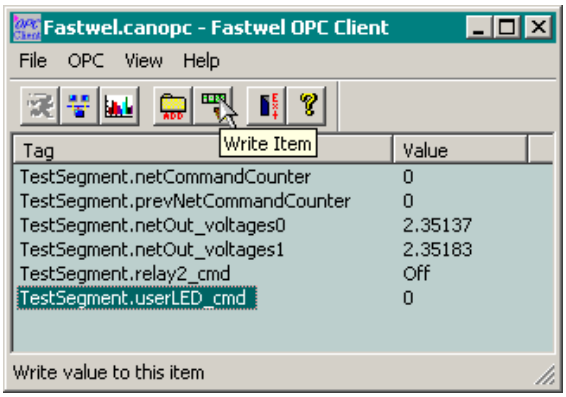

Рис. 67. Главное окно тестового OPC-клиента после соединения с тегами Fastwel CAN OPC Server

11. Измените значение тега *TestSegment.userLED_cmd* с *0* на *1*, для чего выберите тег *TestSegment.userLED_cmd*, нажмите кнопку **Write Item** в панели инструментов OPC-клиента, введите *1* в поле ввода диалоговой панели **Write Item Value** и нажмите кнопку

OK. В ячейке *TestSegment.userLED_cmd:Value* появится значение 1, однако индикатор USER на передней панели контроллера останется выключенным.

12. Измените значение тега *TestSegment.relay2_cmd* с *0* на *1*, для чего выберите тег *TestSegment.relay2_cmd*, нажмите кнопку **Write Item** в панели инструментов OPC-клиента, введите *1* в поле ввода диалоговой панели **Write Item Value** и нажмите кнопку **OK**. В ячейке *TestSegment. relay2_cmd:Value* появится значение *On*, однако второй канал модуля дискретного вывода останется выключенным.
13. Измените значение тега *TestSegment.netCommandCounter* с *0* на *1*, для чего выберите тег *TestSegment.netCommandCounter*, нажмите кнопку **Write Item** в панели инструментов OPC-клиента, введите *1* (или любое другое значение, отличное от 0) в поле ввода диалоговой панели **Write Item Value** и нажмите кнопку **OK**. В ячейке *TestSegment. netCommandCounter:Value* появится введенное значение, включится второй канал модуля дискретного вывода, а индикатор USER начнет светиться зеленым цветом.
14. Закройте окно OPC-клиента, выбрав команду **File–Exit**, после чего закройте OPC-сервер (аналогичной командой). Демонстрационная версия OPC-сервера не поддерживает сохранение конфигурации тегов, поэтому в появившемся сообщении о необходимости сохранить конфигурацию сервера нажмите **No**. Индикатор USER начнет прерывисто светиться красным цветом, а второй канал модуля дискретного вывода останется включенным.

3.12. Проверка работоспособности сетевых функций контроллеров CPM712 (MODBUS RTU) и CPM713 (MODBUS TCP)

Работоспособность сетевых функций контроллера CPM712 и CPM713 на ПК может быть проверена при помощи демонстрационной версии OPC-сервера Fastwel Modbus OPC Server, который может одновременно обмениваться данными с узлами MODBUS RTU/ASCII и MODBUS TCP. В дальнейшем предполагается, что демонстрационная версия Fastwel Modbus OPC Server успешно установлена на ПК, а контроллер подключен к одному из последовательных портов ПК, оснащенных преобразователем интерфейса RS-232C в RS-485.

1. Запустите OPC-сервер из программной группы *Fastwel Modbus OPC Server*, нажмите кнопку в панели инструментов главного окна сервера и введите имя добавляемого узла сети *TestNode* в поле **Имя** появившейся диалоговой панели **Свойства Modbus-устройства**.

   Для контроллера CPM712 оставьте поле **Тип устройства** без изменений, как показано на рис. 68, выберите имя коммуникационного порта ПК, через который установлена связь между ПК и контроллером по сети MODBUS, при необходимости, настройте параметры обмена, нажав кнопку **Свойства порта** и введите сетевой адрес **Modbus-устройства** в поле **Адрес** (в учебном проекте это не требуется), после чего нажмите кнопку **Ok**.

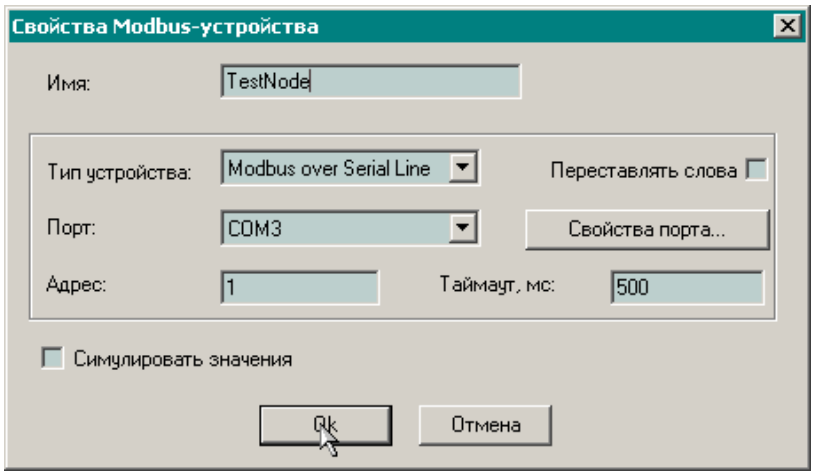

Рис. 68. Настройка параметров MODBUS-устройства для протокола MODBUS RTU

Для контроллера CPM713 измените значение в поле **Тип устройства** на *Modbus TCP*, как показано на рис. 69, и введите IP-адрес устройства в соответствии с настройками контроллера в учебном проекте (например, *10.0.0.2*), после чего нажмите кнопку **Ok**.

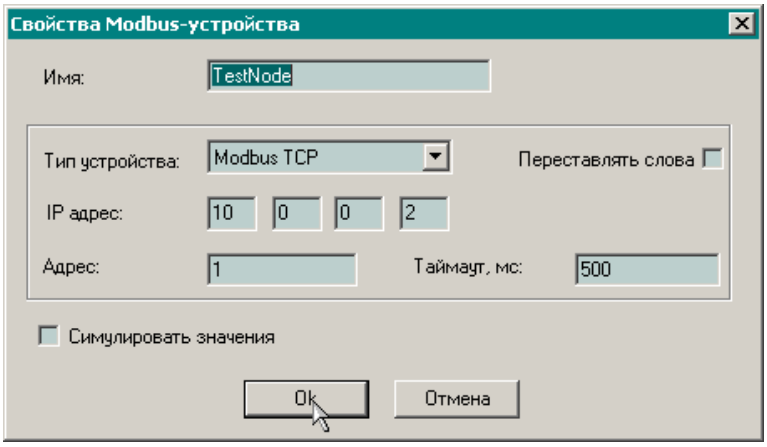

Рис. 69. Настройка параметров MODBUS-устройства для протокола MODBUS TCP

Название устройства появится в списке устройств, расположенном в левой части главного окна OPC-сервера, как показано на рис. 70.

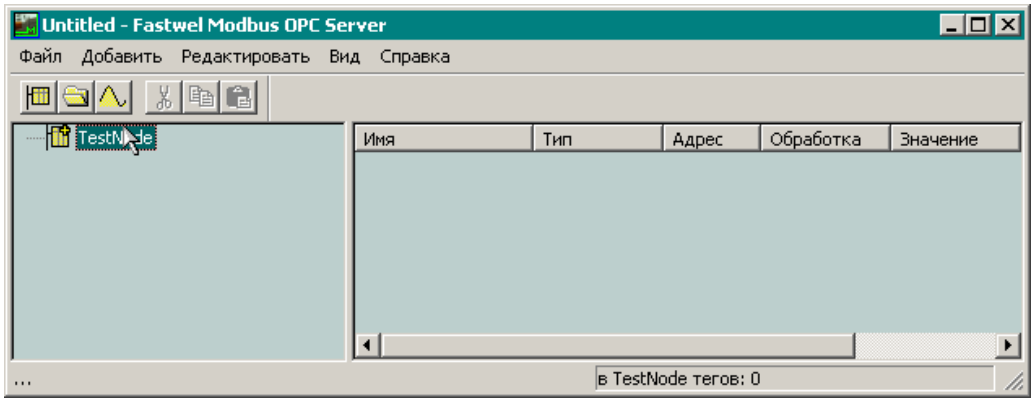

Рис. 70. Главное окно Fastwel Modbus OPC Server после добавления устройства

2. Для добавления тегов, через которые будут приниматься значения на каналах модуля аналогового ввода, нажмите кнопку  или выполните команду меню **Добавить–Новый тег**. На экране появится диалоговая панель **Свойства тега**, показанная на рис. 71.

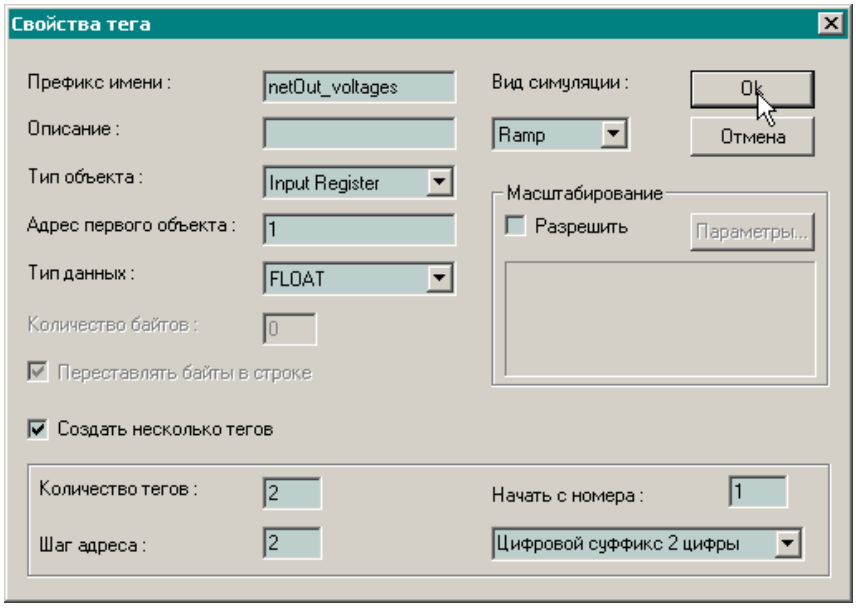

Рис. 71. Диалоговая панель Свойства тега при создании тегов, соответствующих значениям на каналах модуля аналогового ввода

3. Отметьте флажок **Несколько тегов**, в поле **Префикс имени** введите *netOut_voltages*, в поле **Адрес первого объекта** – *1*, выберите тип данных *FLOAT* для добавляемых объектов в поле **Тип данных**, а в полях **Количество тегов** и **Шаг адреса** введите значения *2*, как показано на рис. 71. Это означает, что сервер автоматически создаст два тега действительного типа одинарной точности, которые будут получаться у Modbus-устройства командой группового чтения четырех регистров, начиная с адреса 5000. Закройте диалоговую панель нажатием кнопки **Ok**. Имена тегов *netOut_voltages01* и *netOut_voltages02* появятся в правой области главного окна OPC-сервера, как показано на рис. 72.

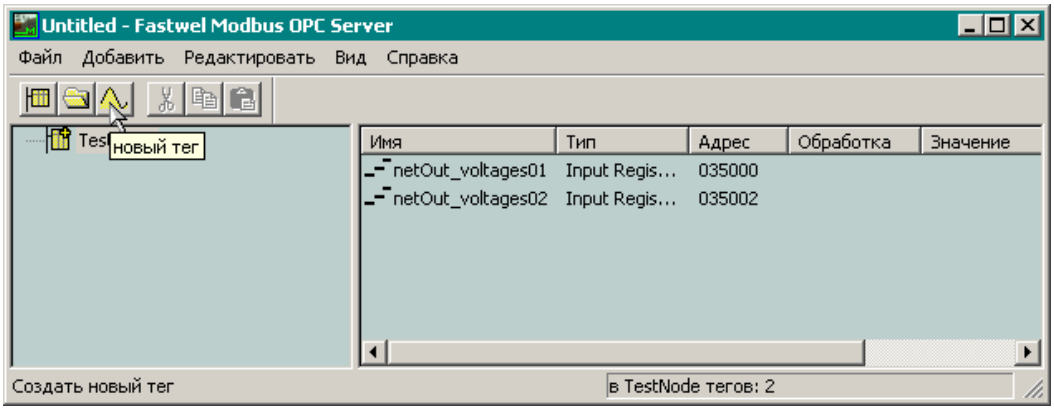

Рис. 72. Главное окно OPC-сервер после создания тегов, соответствующих значениям на каналах модуля аналогового ввода

4. Для добавления тега, который будет служить для приема последнего значения счетчика сетевых команд, принятого контроллером, нажмите кнопку или выполните команду меню **Добавить–Новый тег** и в появившейся диалоговой панели **Свойства тега** введите параметры тега *prevNetCommandCounter*, как показано на рис. 73, и нажмите кнопку **Ok**.

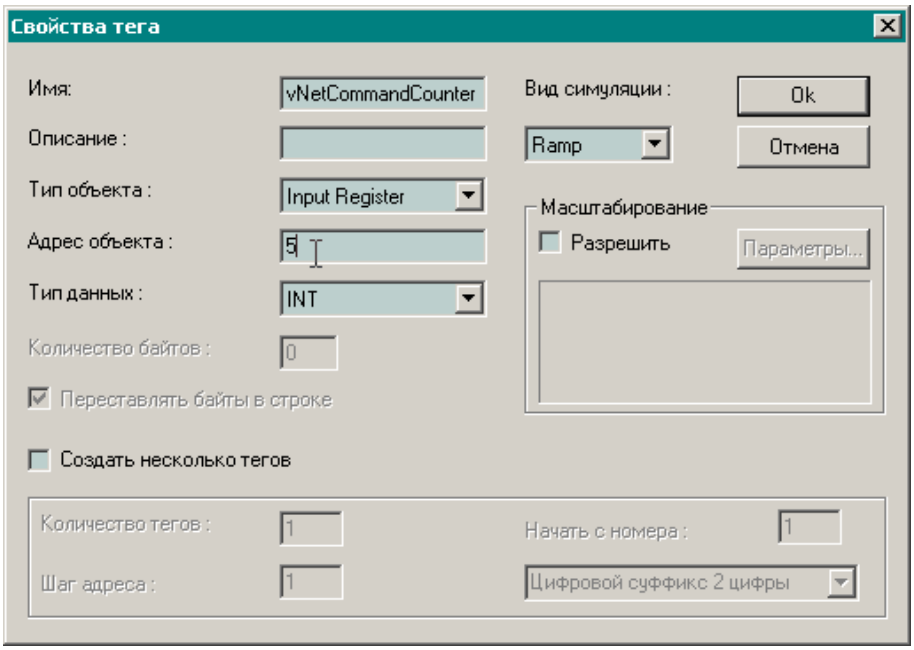

Рис. 73. Диалоговая панель Свойства тега при создании тега, соответствующего последнему значению счетчика, принятому контроллером

5. Для добавления тега, который будет использоваться для передачи счетчика сетевых команд контроллеру, нажмите кнопку или выполните команду меню **Добавить–Новый тег**, в появившейся диалоговой панели **Добавить–Новый тег** введите параметры тега *netCommandCounter*, как показано на рис. 74, и нажмите кнопку **Ok**. Обратите внимание, что в поле **Тип объекта** выбрана опция *Holding Register*.

6. Для добавления тега *userLED_cmd*, который будет использоваться для передачи кода цвета индикатора USER, нажмите кнопку или выполните команду меню **Добавить–Новый тег**, в появившейся диалоговой панели **Свойства тега** введите параметры тега,

как показано на рис. 75, и нажмите кнопку **Ok**. Обратите внимание, что в поле **Тип объекта** также выбрана опция *Holding Register*, значение в поле **Адрес** соответствует адресу регистра в контроллере, а поле **Тип данных** содержит опцию *WORD*.

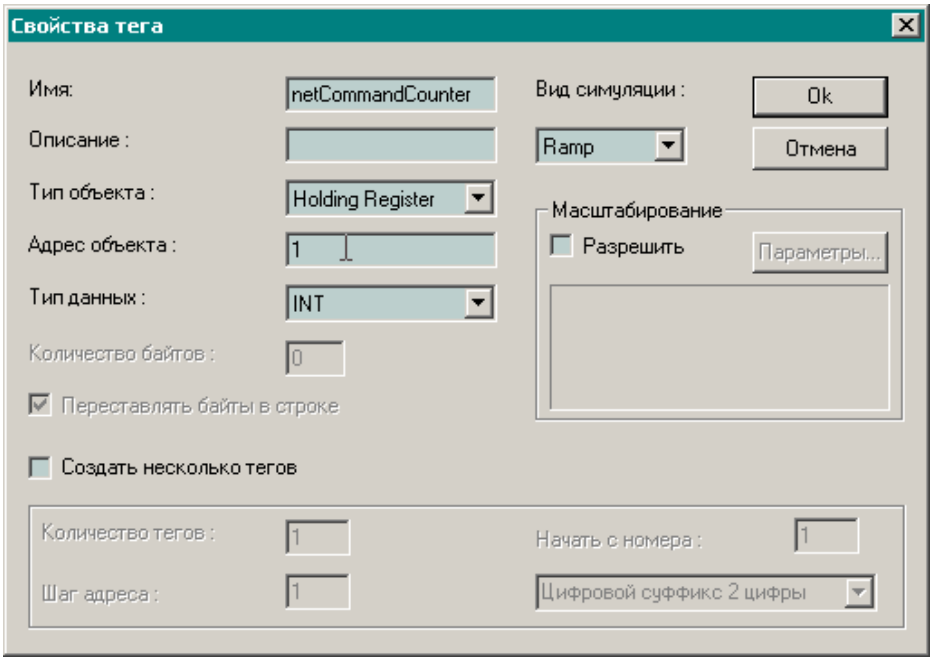

Рис. 74. Диалоговая панель Свойства тега при создании тега для передачи счетчика команд

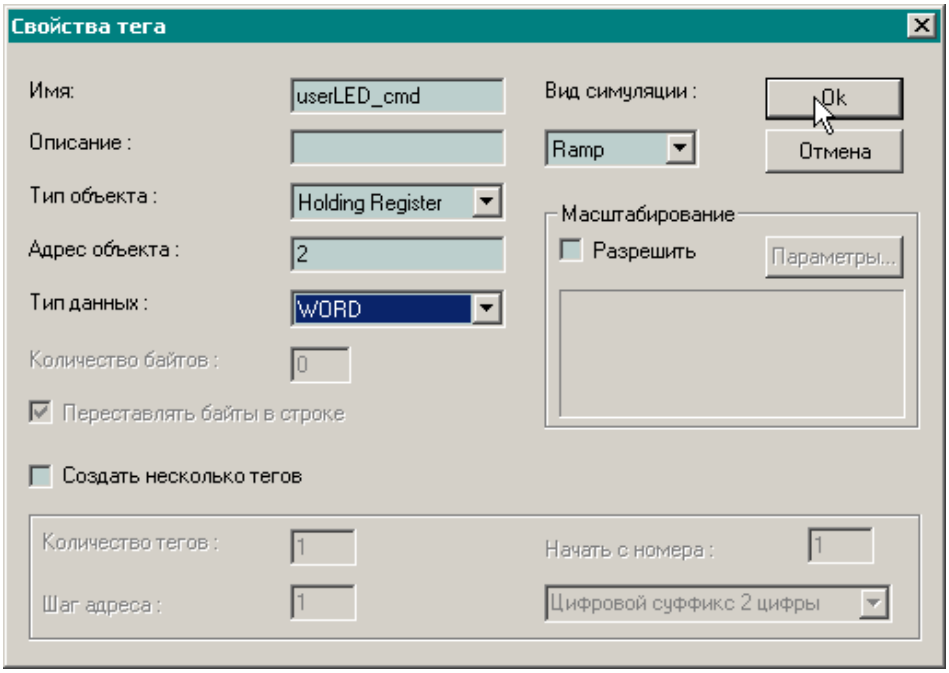

Рис. 75. Диалоговая панель Свойства тега при создании тега для передачи кода цвета индикатора USER

7. Для добавления тега *relay2_cmd*, который будет использоваться для передачи команды управления вторым каналом модуля дискретного вывода, нажмите кнопку  или выполните команду меню **Добавить–Новый тег**, в появившейся диалоговой панели **Свойства тега** введите параметры тега, как показано на рис. 76, и нажмите кнопку **Ok**. Обратите внимание, что в поле **Тип объекта** выбрана опция *Coil*, значение в поле **Адрес** соответствует адресу выходного битового поля в **Панели свойств Fastwel** в окне ресурса **PLC Configuration** среды CoDeSys, а поле **Тип данных** содержит опцию *BIT*.

   Полный список созданных тегов в правой части окна OPC-сервера показан на рис. 77.

8. Запустите тестового клиента OPC (ярлык *OPC Client* в программной группе *Fastwel Modbus OPC Server*), выполните команду **OPC–Connect** в его главном меню, выберите *Fastwel.Modbusopc* в диалоговой панели **Select OPC Server** и нажмите кнопку **OK**.

9. Выполняйте команду **OPC–Add Item** в главном меню клиента OPC последовательно для всех только что созданных тегов OPC-сервера, в результате чего главное окно клиента будет выглядеть, как показано на рис. 78.

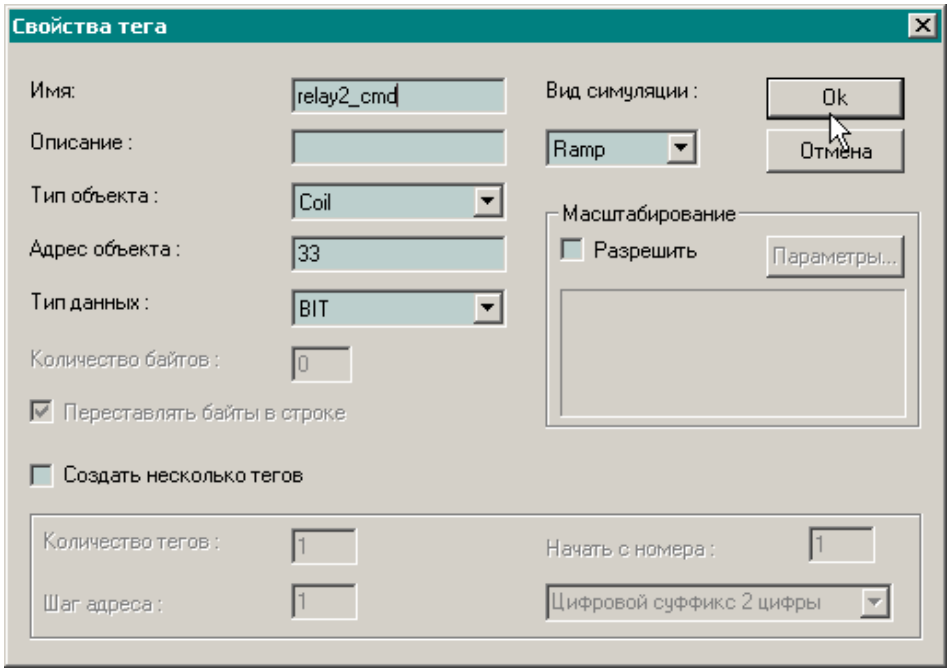

Рис. 76. Диалоговая панель Свойства тега при создании тега для передачи команды управления вторым каналом модуля дискретного вывода

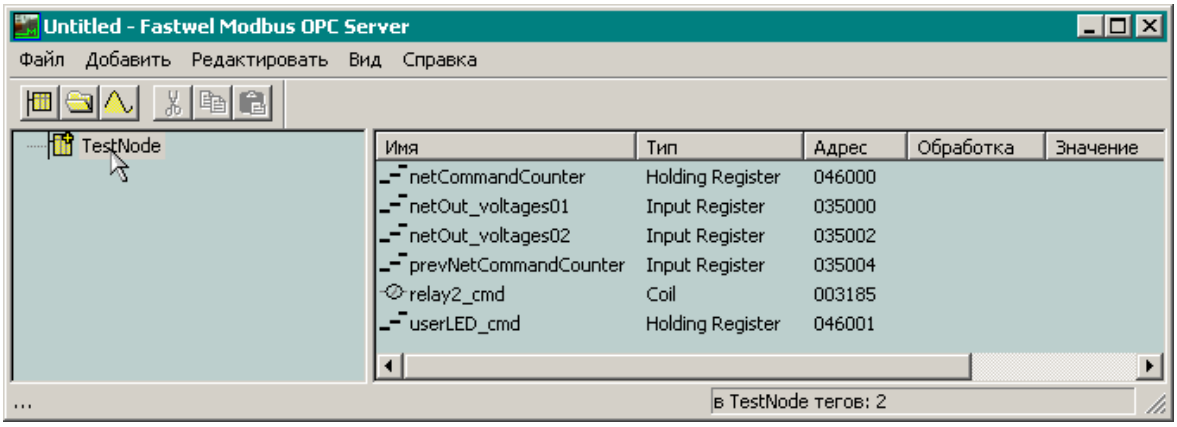

Рис. 77. Полный список тегов OPC-сервера

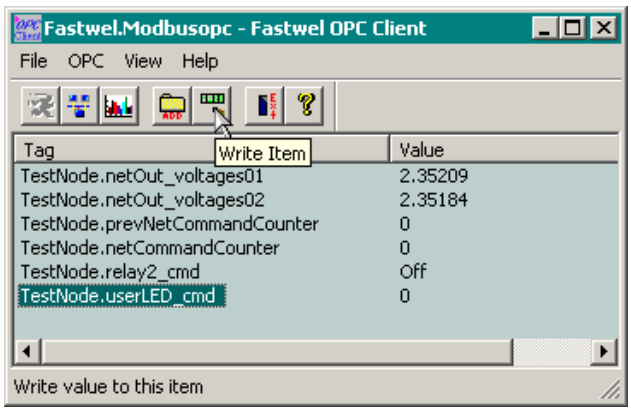

Рис. 78. Главное окно тестового OPC-клиента после соединения с тегами Fastwel Modbus OPC Server

10. Измените значение тега *TestNode.userLED_cmd* с *0* на *1*, для чего выберите тег *TestNode.userLED_cmd*, нажмите кнопку **Write Item** в панели инструментов OPC-клиента, введите *1* в поле ввода диалоговой панели **Write Item Value** и нажмите кнопку **OK**. В ячейке *TestNode.userLED_cmd:Value* появится значение 1, однако индикатор USER на передней панели контроллера останется выключенным.

11. Измените значение тега *TestNode.relay2_cmd* с *0* на *1*. В ячейке *TestNode.relay2_cmd:Value* появится значение *On*, однако второй канал модуля дискретного вывода останется выключенным.
12. Измените значение тега *TestNode.netCommandCounter* с 0 на 1. В ячейке *TestNode.netCommandCounter:Value* появится введенное значение, включится второй канал модуля дискретного вывода, а индикатор USER начнет светиться зеленым цветом.
13. Закройте окно OPC-клиента, выбрав команду **File–Exit**. Индикатор USER начнет прерывисто светиться красным цветом, а второй канал модуля дискретного вывода останется включенным.

ПРИЛОЖЕНИЕ А. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Версия	**Дата**	**Ссылка**	**Статус**	**Примечания**
2.52.23926	18.03.2011	Документ	создан	
2.61.23940	25.06.2013	Документ	изменен	Устранены дефекты оформления