КОМПЬЮТЕР МОДУЛЬНЫЙ
МК905

Система разработки и исполнения приложений CoDeSys 2.3

Руководство программиста

Версия 2.0

Настоящий документ содержит информацию о принципах функционирования, а также указания по настройке и разработке прикладных программ для системы исполнения приложений CoDeSys 2.3 на компьютере модульном MK905.

Торговые марки

Логотип «Fastwel» является торговой маркой, принадлежащей ЗАО «НПФ «ДОЛОМАНТ», Москва, Российская Федерация.

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

Права собственности

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

Контактная информация

Производитель ЗАО «НПФ «ДОЛОМАНТ»:

Почтовый адрес: Российская Федерация, 117437, Москва, Профсоюзная ул., 108

Телефон: (495) 232-2033

Факс: (495) 232-1654

Электронная почта: info@fastwel.ru

Для получения информации о других продуктах, выпускаемых под торговой маркой «Fastwel», посетите наш Интернет-сайт по адресу

http://www.fastwel.ru/

Эксклюзивный дистрибьютор компания «Прософт»

Электронная почта: info@prosoft.ru

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

Телефон: (495) 234-0636

Факс: (495) 234-0640

Авторское право

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ ... 6

2. ОБЩИЕ СВЕДЕНИЯ ... 7

2.1. Назначение MK905 ... 7
2.2. Структура аппаратных средств MK905 ... 7
2.3. Структура программного обеспечения MK905 ... 8
2.3.1. Состав программного обеспечения MK905 ... 8
2.3.2. Пакет адаптации CoDeSys IDE ... 8
2.3.3. Среда исполнения CoDeSys ... 9
2.4. Основные характеристики MK905 ... 11
2.4.1. Характеристики подсистемы исполнения приложений CoDeSys ... 11
2.4.2. Характеристики сервиса ввода-вывода ... 12
2.5. Состав поставляемого программного обеспечения ... 12
2.5.1. Перечень программного обеспечения на компакт-диске Fastwel I/O Product CD ... 12
2.5.2. Содержимое программы установки Fastwel CoDeSys Adaptation ... 13
2.6. Системные требования к рабочему месту разработчика ... 13
2.6.1. Требования к аппаратным средствам ... 13
2.6.2. Требования к системному программному обеспечению ... 13

3. УСТАНОВКА И НАСТРОЙКА АДАПТИРОВАННОЙ СРЕДЫ CODESYS 2.3 ... 14

3.1. Предварительные замечания ... 14
3.2. Установка ... 14
3.3. Проверка установки ... 15
3.4. Настройка параметров трансляции проекта CoDeSys ... 16
3.5. Удаление адаптированной среды разработки CoDeSys ... 17
3.6. Обновление системного программного обеспечения MK905 ... 17
3.6.1. Общие сведения ... 17
3.6.2. Обновление из среды CoDeSys ... 17
3.7. Обновление по протоколу ftp ... 18
3.8. Режим совместимости с CPM90201 ... 18
3.8.1. Активизация режима совместимости с CPM90201 ... 18
3.8.2. Отмена режима совместимости с CPM90201 ... 18

4. ПРИНЦИП РАБОТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОНТРОЛЛЕРА ... 19

4.1. Общие сведения ... 19
4.1.1. Приложение пользователя ... 19
4.1.2. Назначение системного программного обеспечения контроллера ... 22
4.2. Принципы работы адаптированной среды исполнения CoDeSys ... 22
4.2.1. Режимы работы ... 22
4.2.1.1. Безопасный режим ... 22
4.2.1.2. Нормальный режим ... 23
4.2.2. Процесс запуска контроллера после включения питания ... 23
4.2.2.1. Запуск при первом включении питания ... 23
4.2.2.2. Запуск при наличии загруженного приложения ... 24
4.2.3. Процесс загрузки или обновления приложения ... 25
4.2.4. Исполнение приложения пользователя ... 28
4.2.4.1. Общие сведения ... 28
4.2.4.2. Исполнение циклических задач ... 31
4.2.4.3. Исполнение ациклических задач ... 33
4.2.4.4. Вызов обработчиков системных событий ... 35
4.2.4.5. Обмен данными между задачами ... 37
4.2.5. Диагностика ... 41
4.3. Принцип работы сервиса ввода-вывода ... 42
4.3.1. Общие сведения ... 42
4.3.2. Инициализация шины ... 44
4.3.3. Обмен данными с модулями ввода-вывода ... 44
4.3.3.1. Групповой режим (*Single Group*) ... 44
4.3.3.2. Режим индивидуального обмена (*Group per Module*) ... 45
4.3.4. Обработка нештатных ситуаций ... 47
4.3.4.1. Ошибка инициализации при запуске контроллера ... 47
4.3.4.2. Потеря связи с модулями ввода-вывода в процессе работы ... 47
4.3.5. Диагностика ... 47
4.3.5.1. Индикация ... 47
4.3.5.2. Диагностические каналы сервиса ввода-вывода ... 47

5. УКАЗАНИЯ ПО РАЗРАБОТКЕ ПРИЛОЖЕНИЙ 49

5.1. Общие сведения 49
5.2. Создание проекта 49
5.3. Создание и редактирование конфигурации контроллера 50
5.4. Создание программных единиц и задач 51
5.5. Связывание программ с окружением и ввод-вывод данных 51
5.5.1. Общие сведения 51
5.5.2. Ссылки на образа процесса в декларациях входных или выходных переменных 52
5.5.3. Создание символических имен каналов в ресурсе PLC Configuration 54
5.5.4. Использование ресурса VAR_CONFIG 54
5.6. Создание обработчиков системных событий 54
5.7. Трансляция приложения 55
5.8. Загрузка приложения в контроллер и отладка 56
5.8.1. Общие сведения 56
5.8.2. Login 56
5.8.3. Загрузка приложения в контроллер 56
5.8.4. Просмотр и установка значений переменных 57
5.9. Запись файлов в контроллер 59
5.10. Чтение файлов из контроллера 59
5.11. Описание кодов ошибок при взаимодействии между средой разработки и контроллером 59
5.12. Трассировка переменных 60

6. СИСТЕМНЫЕ БИБЛИОТЕКИ 61

6.1. Общие сведения 61
6.2. Библиотека FastwelSysLibFile.lib 61
6.2.1. Общие сведения 61
6.2.1.1. Особенности библиотеки FastwelSysLibFile.lib 61
6.2.1.2. Относительный путь 61
6.2.1.3. Абсолютный путь 61
6.2.1.4. Доступ к съемным USB-накопителям 62
6.2.2. Описание функций 62
6.2.2.1. FwSysFileGetSize 62
6.2.2.2. FwSysFileExists 63
6.2.2.3. FwSysFileGetTime 63
6.2.2.4. FwSysFileCopy 64
6.2.2.5. FwSysFileDelete 64
6.2.2.6. FwSysFileRename 65
6.2.2.7. FwSysFileOpen 65
6.2.2.8. FwSysFileClose 67
6.2.2.9. FwSysFileGetPos 67
6.2.2.10. FwSysFileSetPos 67
6.2.2.11. FwSysFileRead 67
6.2.2.12. FwSysFileWrite 67
6.2.2.13. FwSysFileEOF 68
6.2.2.14. FwSysDirCreate 68
6.2.2.15. FwSysDirExist 69
6.2.2.16. FwSysDirRemove 69
6.3. Библиотека FastwelTasksExchange.lib 70
6.3.1. Общие сведения 70
6.3.2. Функция F_IecTasks_getInfo 70
6.4. Библиотека FastwelSysLibCom.lib 71
6.4.1. Общие сведения 71
6.4.2. Описание функций 71
6.4.2.1. FwSysComOpen 71
6.4.2.2. FwSysComClose 71
6.4.2.3. FwSysComSetSettings 72
6.4.2.4. FwSysComRead 72
6.4.2.5. FwSysComWrite 73
6.5. Библиотека FastwelSysLibSockets.lib 73
6.5.1. Общие сведения 73
6.5.2. Описание функций 73
6.5.2.1. FwSysSockCreate 73
6.5.2.2. FwSysSockClose 74
6.5.2.3. FwSysSockListen 74
6.5.2.4. FwSysSockAccept 74
6.5.2.5. FwSysSockBind 75
6.5.2.6. FwSysSockConnect 75
6.5.2.7. FwSysSockGetOption 76
6.5.2.8. FwSysSockSetOption 76

6.5.2.9. FwSysSockIoctl ... 77
6.5.2.10. FwSysSockSelect ... 78
6.5.2.11. FwSysSockRecv ... 78
6.5.2.12. FwSysSockRecvFrom ... 79
6.5.2.13. FwSysSockSend ... 79
6.5.2.14. FwSysSockSendTo ... 80
6.5.2.15. FwSysSockShutdown ... 80
6.5.2.16. FwSysSockHtons ... 81
6.5.2.17. FwSysSockHtonl ... 81
6.5.2.18. FwSysSockNtohs ... 81
6.5.2.19. FwSysSockNtohl ... 82
6.5.2.20. FwSysSockGetLastError ... 82
6.5.2.21. FwSysSockGetHostName ... 82
6.5.2.22. FwSysSockInetAddr ... 82
6.5.2.23. FwSysSockInetNtoa ... 83
6.5.2.24. FwSysSockSetIPAddress ... 83
6.5.3. Описание типов данных ... 83
6.5.3.1. INADDR ... 83
6.5.3.2. SOCKADDRESS ... 83
6.5.3.3. SOCKET_FD_SET ... 84
6.5.4. Коды ошибок ... 84
6.6. Библиотека FastwelUtils.lib ... 85
6.6.1. FwCheckSum16 ... 85
6.6.2. FwCheckSum32 ... 85
6.6.3. FwCRC16 ... 86
6.6.4. FwCRC32 ... 86
6.6.5. FwIsPOUExist ... 87
6.6.6. FwGetPOU_CRC32 ... 87
6.6.7. FwMemCompare ... 87
6.6.8. FwMemCopy ... 88
6.7. SysLibGetAddress.lib ... 88
6.7.1. Общие сведения ... 88
6.7.2. SysLibGetAddress ... 88
6.7.3. SysLibGetSize ... 88

7. СЕТЕВЫЕ СРЕДСТВА ... 89

7.1. Общие сведения ... 89
7.2. Характеристики сетевых средств ... 89
7.2.1. Характеристики сервера MODBUS TCP ... 89
7.2.2. Характеристики клиента MODBUS TCP ... 90
7.2.3. Характеристики сервера MODBUS SERIAL ... 90
7.2.4. Характеристики клиента MODBUS SERIAL ... 91
7.3. Принцип работы и способы конфигурирования сервера MODBUS ... 91
7.3.1. Общие сведения ... 91
7.3.2. Параметры протокола сервера MODBUS SERIAL ... 92
7.3.3. Параметры протокола сервера MODBUS TCP ... 93
7.3.4. Коммуникационные объекты сервера MODBUS ... 93
7.3.5. Обмен данными с клиентами MODBUS ... 94
7.3.6. Формат запроса/ответа на чтение расширенной идентификационной информации ... 97
7.3.7. Обслуживание сетевых запросов ... 97
7.3.8. Диагностика ... 98
7.4. Принцип работы и способы конфигурирования клиента MODBUS ... 98
7.4.1. Общие сведения ... 98
7.4.2. Параметры мастера MODBUS TCP ... 99
7.4.3. Параметры мастера MODBUS SERIAL ... 100
7.4.4. Обмен данными с подчиненными устройствами ... 101
7.4.5. Диагностика ... 103
7.5. Инициализация сервисов сети MODBUS ... 103
7.6. Доступ к полям данных коммуникационных объектов из приложения ... 103
7.7. Коммуникационные средства верхнего уровня ... 105
7.7.1. Общие сведения ... 105
7.7.2. Установка коммуникационного драйвера CoDeSys Gateway Server ... 106
7.7.3. Создание логического информационного канала между средой разработки и контроллером по протоколу MODBUS TCP ... 106
7.7.4. Создание логического информационного канала между средой разработки и контроллером по протоколу MODBUS Serial ... 107
7.7.5. Создание логического информационного канала между средой разработки и контроллером по последовательному каналу связи (P2P) ... 108

7.7.6. Дополнительные замечания ... 109

8. ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ ... 110

8.1. Общие сведения ... 110
8.2. Копирование файлов по протоколу ftp ... 110
8.3. Перенос файлов на USB-накопителе ... 111
8.3.1. Локальное копирование ... 111
8.3.2. Удаление файлов ... 111
8.3.3. Копирование без использования монитора и клавиатуры ... 112

9. ПОДСИСТЕМА ЦЕЛЕВОЙ ВИЗУАЛИЗАЦИИ ... 113

9.1. Общие сведения ... 113
9.2. Функциональные возможности и характеристики ... 114
9.2.1. Функциональные возможности целевой визуализации MK905 ... 114
9.2.2. Основные характеристики целевой визуализации MK905 ... 115
9.3. Принцип работы подсистемы целевой визуализации ... 116
9.3.1. Общие сведения ... 116
9.3.2. Функционирование в нормальном режиме ... 117
9.3.3. Функционирование в безопасном режиме ... 118
9.4. Указания по применению ... 118
9.4.1. Настройка свойств целевой визуализации ... 118
9.4.1.1. Общие сведения ... 118
9.4.1.2. Настройка разрешения видеоподсистемы контроллера ... 118
9.4.1.3. Настройка параметров целевой визуализации в проекте ... 119
9.4.1.4. Отключение загрузки вспомогательных файлов визуализации ... 119
9.4.2. Создание и редактирование форм визуализации ... 120
9.4.3. Особенности работы с растровыми изображениями ... 120
9.4.4. Особенности отображения и ввода текстовой информации ... 120
9.4.4.1. Общие сведения ... 120
9.4.4.2. Вывод статической текстовой информации символами кириллицы ... 120
9.4.4.3. Вывод динамической информации символами кириллицы ... 122
9.4.5. Указания по реализации поддержки нескольких языков интерфейса оператора ... 124
9.4.5.1. Общие сведения ... 124
9.4.5.2. Создание файла динамического текста ... 125
9.4.5.3. Подключение файла динамического текста к проекту CoDeSys ... 126
9.4.5.4. Создание программы ... 126
9.4.5.5. Создание кнопки выбора языка ... 126
9.4.5.6. Создание кнопки переключения TOGGLE_VAR ... 128
9.4.5.7. Создание статической надписи для отображаемого значения ANGLE_VAR ... 129
9.4.5.8. Создание динамического элемента отображения значения ANGLE_VAR ... 130
9.4.5.9. Загрузка приложения в контроллер ... 132
9.4.6. Особенности реализации реакции на действия оператора/пользователя ... 132
9.4.6.1. Общие сведения ... 132
9.4.6.2. Фиксированное переключение значения булевой переменной ... 133
9.4.6.3. Нефиксированное переключение значения булевой переменной ... 133
9.4.6.4. Переключение между окнами форм визуализации ... 133
9.4.6.5. Ввод числовой и текстовой информации по месту отображения ... 134
9.4.6.6. Ввод числовой информации при помощи системной диалоговой панели Numpad ... 134
9.4.6.7. Ввод информации при помощи системной диалоговой панели Keypad ... 135
9.4.6.8. Выполнение простых выражений с присвоением результата некоторой переменной ... 136
9.4.6.9. Выбор языка интерфейса оператора ... 137
9.4.6.10. Переключение языка интерфейса оператора ... 137
9.4.6.11. Запуск внешнего приложения ... 137
9.4.6.12. Изменение текущего уровня пользователя ... 137
9.4.6.13. Изменение паролей для уровней пользователей ... 138
9.4.7. Отображение графиков и архивация данных ... 139
9.4.7.1. Общие сведения ... 139
9.4.7.2. Создание приложения с поддержкой целевой визуализации ... 139
9.4.7.3. Разработка программы ... 140
9.4.7.4. Создание формы визуализации и добавление элемента Тренд ... 142
9.4.7.5. Настройка элемента Тренд ... 143
9.4.7.6. Настройка параметров архивации ... 145
9.4.7.7. Создание дополнительных элементов визуализации ... 147
9.4.7.8. Установка цвета фона формы визуализации ... 148
9.4.7.9. Загрузка приложения в контроллер ... 149
9.4.7.10. Управление элементом Тренд во время исполнения приложения ... 149

10. РЕАЛИЗАЦИЯ ПРИКЛАДНЫХ АЛГОРИТМОВ НА C/C++ И РАСШИРЕНИЕ СИСТЕМЫ ИСПОЛНЕНИЯ ПРИЛОЖЕНИЙ CODESYS ... 151

10.1. Общие сведения ... 151

10.2. Требования к рабочему месту разработчика ... 151
10.2.1. Требования к конфигурации программного обеспечения ... 151
10.2.2. Средства разработки для Windows CE 5.0 ... 151
10.3. Взаимодействие системы исполнения контроллера с DLL пользователя ... 152
10.3.1. Общие сведения ... 152
10.3.2. Функция связывания системы исполнения с DLL пользователя F_CdsUsr_Link ... 152
10.3.3. Интерфейс доступа к сегментам данных приложения CoDeSys ... 152
10.3.3.1. F_USR_CDS_IFACE ... 152
10.3.3.2. pfReadVariable/pfWriteVariable ... 153
10.3.3.3. Функция чтения статуса приложения ... 154
10.3.4. Интерфейс координации совместной работы с DLL ... 155
10.3.4.1. F_CDS_USR_IFACE ... 155
10.3.4.2. pfAccept ... 155
10.3.4.3. pfEvent ... 157
10.3.4.4. pfDoCycle ... 158
10.3.5. Вызов функций интерфейса координации совместной работы с DLL пользователя ... 158
10.3.5.1. Последовательность обращения к функциям интерфейса DLL ... 158
10.3.5.2. Обработка исключений в коде функций интерфейса DLL ... 160
10.3.6. Вызов функций интерфейса системы исполнения из DLL пользователя ... 160
10.3.6.1. Общие сведения ... 160
10.3.6.2. Вызов функции чтения сегментов приложения ... 160
10.3.6.3. Вызов функции записи в сегменты приложения ... 161
10.3.7. Получение информации о размещении переменных приложения CoDeSys ... 161
10.3.7.1. Формирование файла символической информации о проекте CoDeSys ... 161
10.3.7.2. Формат символической информации ... 162
10.4. Пример реализации DLL пользователя ... 164
10.4.1. Общие сведения ... 164
10.4.2. Проект CoDeSys ... 164
10.4.3. Проект DLL ... 164
10.4.3.1. Общие сведения ... 164
10.4.3.2. Описание исходного текста DLL ... 164

11. СООБЩЕНИЯ ОБ ОШИБКАХ ... 168

11.1. Общие сведения ... 168
11.2. Исключения ... 168
11.2.1. Типы исключений ... 168
11.2.2. Исключения при исполнении кода приложения CoDeSys ... 169
11.2.3. Исключение вне кода приложения CoDeSys ... 169
11.2.4. Исключения в коде пользовательской DLL расширения системы исполнения ... 169
11.3. Сообщения о дефиците системных ресурсов ... 170
11.4. Зацикливания в пользовательском коде ... 170
11.4.1. Зацикливание в циклической задаче ... 170
11.4.2. Зацикливание в сервисной задаче ... 171
11.5. Ошибки конфигурации приложения ... 171

ПРИЛОЖЕНИЕ 1 . ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ ... 174

1. Введение

Настоящий документ содержит сведения о принципах функционирования, а также указания по настройке и программированию компьютеров модульных MK905-01,-03\CDS в среде разработки CoDeSys 2.3 фирмы 3S Smart Software Solutions.

Компьютер модульный MK905 с предустановленной адаптированной системой исполнения прикладных программ, разрабатываемых в среде CoDeSys 2.3, будет далее называться MK905 или контроллер MK905.

При работе с настоящим документом следует также пользоваться документами:

1. Модули ввода-вывода Fastwel I/O. Руководство программиста.
2. Компьютеры модульные MK905-01,03\CDS. Руководство по эксплуатации.

и документом *User Manual for PLC Programming with CoDeSys 2.3*, входящим в состав среды разработки CoDeSys.

Предполагается, что пользователь среды разработки CoDeSys 2.3, адаптированной для создания приложений для MK905, 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 MK905

Компьютеры модульные MK905 с предустановленной системой исполнения приложений CoDeSys 2.3 являются программируемымм логическими контроллерами, предназначенными для создания автоматизированных систем сбора данных и управления. Приложения для MK905-01,-03\CDS должны разрабатываться в среде разработки CoDeSys 2.3 (далее – CoDeSys IDE) на языках ST, IL, SFC, FBD, LD стандарта IEC 61131-3.

MK905 может использоваться как автономно, так и в качестве элемента распределенной системы сбора данных и управления.

Варианты исполнения MK905 с предустановленной системой исполнения приложений CoDeSys перечислены в табл. 1.

Таблица 1

Обозначение	Описание
MK905-01\CDS	Компьютер модульный MK905-01, CPU AMD Geode LX 800 500 МГц, 1x FBUS, 2x Ethernet, 4x RS-485 (COM5–COM8), 1x RS-232C (COM1, сервисный*), 4x USB, FixedDisk1 (NAND Flash) 1 Гбайт, FixedDisk2 (CompactFlash, 1 Гбайт), CoDeSys 2.3 Runtime HMI, Windows CE 5.0 Core
MK905-03\CDS	Компьютер модульный MK905-03, CPU AMD Geode LX 800 500 МГц, 1x FBUS, 2x Ethernet, 4x RS-485 (COM5–COM8), 1x RS-232C (COM1, сервисный*), 1x RS-232C (COM2), 4x USB, FixedDisk1 (NAND Flash) 1 Гбайт, FixedDisk2 (CompactFlash, 1 Гбайт), CoDeSys 2.3 Runtime HMI, Windows CE 5.0 Core

*= порт COM1 интерфейса RS-232C может использоваться только для загрузки и отладки приложений из CoDeSys IDE.

2.2. Структура аппаратных средств MK905

MK905 является вычислительным устройством на базе микропроцессора AMD Geode LX800, совместимого с Intel Pentium и имеющего тактовую частоту 500 МГц.

MK905 имеет следующие аппаратные интерфейсы, обслуживаемые предустановленным на него системным программным обеспечением:

1. FBUS – интерфейс с модулями ввода-вывода Fastwel I/O, далее называемый внутренней шиной. Для подключения модулей ввода-вывода Fastwel I/O к MK905 должен использоваться модуль OM796 и кабель соединительный ACS00055 (либо кабель UTP-5 длиной до 5 м).
2. Два адаптера сети Ethernet могут использоваться для обмена данными между приложением MK905 и другими устройствами по протоколу прикладного уровня MODBUS TCP в качестве мастера и подчиненного узла сети. Кроме того, данные интерфейсы могут использоваться для взаимодействия с CoDeSys IDE по протоколу MODBUS TCP (в режиме подчиненного узла).
3. COM1 – универсальный асинхронный приемо-передатчик (УАПП) интерфейса RS-232C, предназначенный для взаимодействия между CoDeSys IDE и MK905 по протоколу "точка–точка" (P2P).
4. COM2 (только на MK905-03) – УАПП интерфейса RS-232 может использоваться для обмена данными между приложением MK905 и другими устройствами по протоколу прикладного уровня MODBUS RTU или ASCII в качестве мастера или подчиненного узла сети. Данный интерфейс может использоваться для взаимодействия с CoDeSys IDE по протоколу MODBUS RTU или ASCII (в режиме подчиненного узла) либо для непосредственного обмена данными с различными устройствами из приложения через библиотеку FastwelSysLibCom.lib.
5. COM5, COM6, COM7, COM8 – УАПП интерфейсов RS-422/RS-485 могут использоваться для обмена данными между приложением MK905 и другими устройствами по протоколу прикладного уровня MODBUS RTU или ASCII в качестве мастера или подчиненного узла сети. Интерфейсы могут использоваться для

взаимодействия с CoDeSys IDE по протоколу MODBUS RTU или ASCII (в режиме подчиненного узла) либо для непосредственного обмена данными с различными устройствами из прикладной программы через библиотеку FastwelSysLibCom.lib.

6. Четыре адаптера USB могут использоваться для подключения съемных дисковых накопителей, а также клавиатуры и мыши. Доступ к съемным дисковым накопителям из приложения MK905 программы может быть осуществлен при помощи функций библиотеки FastwelSysLibFile.lib. Обратите внимание, что одновременно к MK905 может быть подключено не более трех USB-устройств и не более двух USB-накопителей.
7. Если приложение MK905 должно выполнять отображение графических мнемосхем оператора, созданных в приложении CoDeSys, порт видеоадаптера RGB (VGA) может использоваться для подключения монитора.

2.3. Структура программного обеспечения MK905

2.3.1. Состав программного обеспечения MK905

Программные средства MK905 состоят из следующего системного и инструментального программного обеспечения:

1. Пакет адаптации среды разработки прикладных программ на языках стандарта IEC 61131-3 CoDeSys 2.3 для Fastwel I/O. Программа установки текущей версии пакета адаптации CoDeSys для Fastwel I/O может быть загружена по следующему адресу: ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/ (файл FastwelCoDeSysAdaptation2xx23xxx.exe).
2. Среда исполнения приложений, разрабатываемых в CoDeSys IDE, поставляемая в каждом MK905.

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

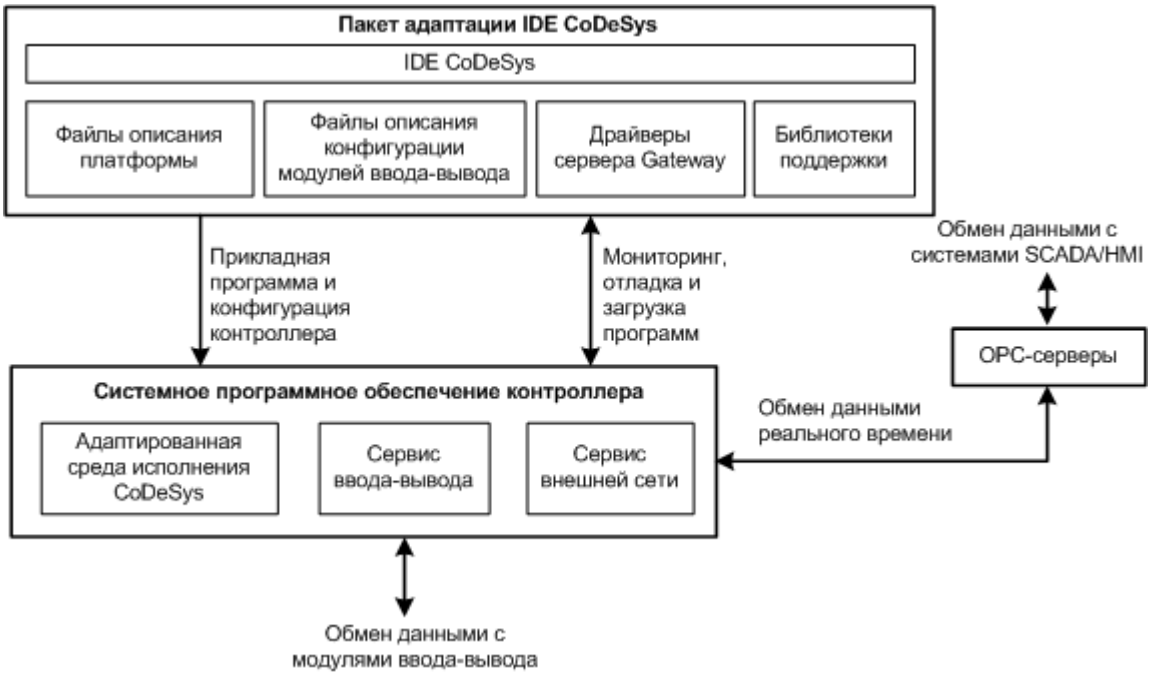


Рис. 1. Структура программного обеспечения MK905

2.3.2. Пакет адаптации CoDeSys IDE

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

1. Интегрированная среда разработки IDE CoDeSys фирмы 3S Smart Software Solutions.
2. Файлы описания платформы MK905, интегрируемые с CoDeSys IDE и позволяющие генерировать исполняемый код приложений для MK905 средствами CoDeSys IDE.

3. Файлы описания конфигурации модулей ввода-вывода, интегрируемые с CoDeSys IDE и позволяющие генерировать конфигурационную информацию для МК905 средствами CoDeSys IDE.
4. Драйверы коммуникационного сервера CoDeSys Gateway Server, интегрируемые с CoDeSys Gateway Server и позволяющие выполнять загрузку прикладных программ в контроллер, удаленную отладку и мониторинг переменных
5. Библиотеки поддержки платформы МК905, содержащие функциональные блоки и функции, обеспечивающие доступ к специфическим функциональным возможностям платформы МК905 из приложений, разрабатываемых в CoDeSys IDE.

CoDeSys IDE 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 сетевых протоколов и перечни описаний сообщений, поступающих по сетям и выдаваемых в сети контроллером, и параметры исполнения приложения в МК905.
2. Отображение входных и выходных переменных разрабатываемого приложения на сетевые сообщения и на каналы модулей ввода-вывода.
3. Реализацию прикладного алгоритма 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 контроллере.
6. Удаленную отладку и управление исполнением приложением в контроллере.

2.3.3. Среда исполнения CoDeSys

Среда исполнения CoDeSys является одним из сервисов системного программного обеспечения МК905. Системное программное обеспечение МК905 выполняет следующие функции:

1. Исполнение приложения, разработанного в CoDeSys IDE, с возможностью просмотра и изменения значений переменных и удаленной загрузки измененной версии.
2. Обмен данными между приложением 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.
7. Управление режимами работы контроллера, обработку ошибок и нештатных ситуаций.

Приложение, разрабатываемое пользователем в CoDeSys IDE, должно состоять хотя бы из одной программы, и может содержать до 32-х циклических и до 64-ти ациклических задач, а также функций обработки системных событий.

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

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

Обработчиком системного события далее называется функция (в терминах IEC 61131-3), вызываемая средой исполнения при возникновении некоторого системного события. К системным

событиям относятся моменты запуска и останова приложения, окончание подготовки приложения к запуску, начало загрузки нового приложения, момент перед заменой текущего приложения на вновь загруженное и другие.

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

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

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

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

Рис. 2. Исполнение циклической задачи

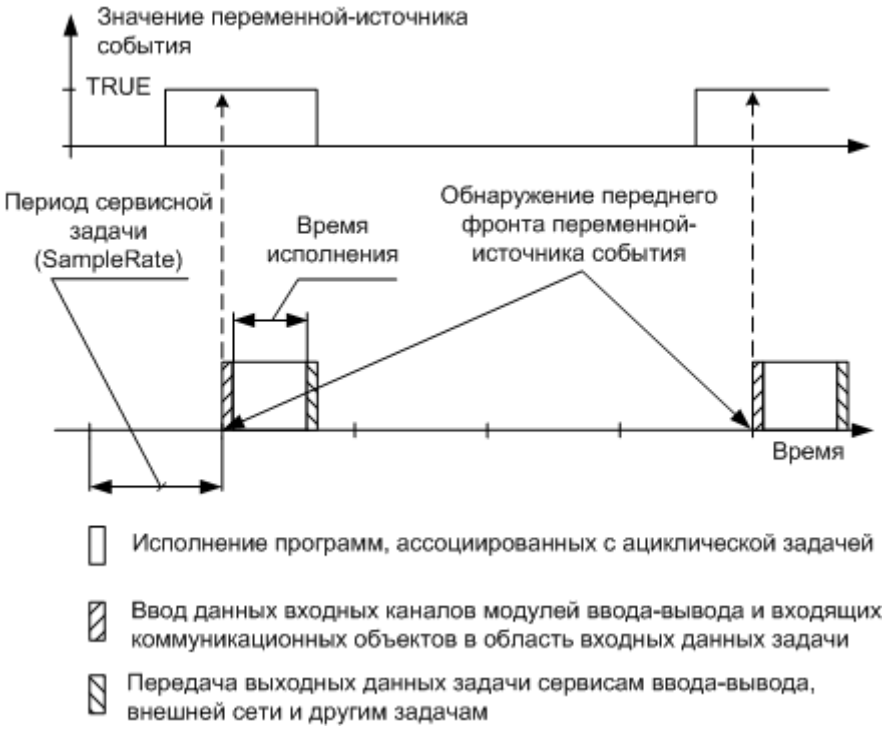


Рис. 3. Исполнение ациклической задачи

2.4. Основные характеристики MK905

2.4.1. Характеристики подсистемы исполнения приложений CoDeSys

Параметр	Единица	Минимум	Номинально	Максимум
Области памяти				
Размер области входных переменных приложения	байт		262144	
Размер области выходных переменных приложения	байт		262144	
Размер области внутренних переменных приложения	байт		2097152	
Размер области исполняемого кода приложения	байт			2097152
Размер области конфигурации приложения	байт			2097152
Размер области энергонезависимых переменных	байт		8192	
Производительность				
Сложение и вычитание 2-байтовых операндов	опер/с		54600000	
Умножение 2-байтовых операндов	опер/с		32800000	
Деление 2-байтовых операндов	опер/с		9457000	
Сложение и вычитание целочисленных 4-байтовых операндов	опер/с		54500000	
Умножение целочисленных 4-байтовых операндов	опер/с		32700000	
Деление целочисленных 4-байтовых операндов	опер/с		9420000	
Сложение и вычитание операндов типа REAL	опер/с		25700000	
Умножение операндов типа REAL	опер/с		21000000	
Деление операндов типа REAL	опер/с		8500000	
Сложение и вычитание операндов типа LREAL	опер/с		20000000	
Умножение операндов типа LREAL	опер/с		17400000	
Деление операндов типа LREAL	опер/с		7840000	
Ядро системы исполнения				
Количество циклических задач	шт	0[1]	1	32
Период циклической задачи	мс	1	10	1000
Количество уровней приоритета циклических задач				32
Размер стека циклической задачи	байт		16000	
Количество ациклических задач	ед	0		64
Количество уровней приоритета ациклических задач	ед			32
Размер стека ациклической задачи	байт		15600	
Количество программных единиц (POU)	шт	1		16384
Количество связей задачи с областью входных данных[2]	шт	0		1024
Количество связей задачи с областью выходных данных	шт	0		1024
Размер переменной типа STRING	байт		80	255
Отладчик				
Количество устанавливаемых постоянных точек останова	шт			100
Количество промежуточных временных точек останова	шт			100
Глубина дерева вызовов при отладке				30
Количество позиций просмотра потока данных (flow positions)	шт			1000
Взаимодействие со средой разработки				
Размер буфера форсируемых переменных	байт			32768
Размер буфера считываемых переменных	байт			131072
Размер буфера трассировки	байт			131072
Таймаут между блоками транспортного протокола обмена со средой разработки	мс			1000

[1] Если пользователь не добавил в конфигурацию задач проекта ни одной задачи любого типа, CoDeSys автоматически сгенерирует одну циклическую задачу с именем *DefaultTask*, которая будет исполняться с периодом, задаваемым параметром MK905 Programmable Automation Controller:EventsRate ресурса PLC Configuration, и исполнять программу с именем PLC_PRG

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

2.4.2. Характеристики сервиса ввода-вывода

Параметр	Единица	Минимум	Номинально	Максимум
Количество модулей ввода-вывода	шт	0		64
Размер области ввода данных	байт	0		1680
Размер области вывода данных	байт	0		1680
Размер поля контрольной суммы сообщения	байт		4	
Размер символа	бит		10	
Скорость обмена	Мбит/с		2	
Общий размер передаваемой служебной информации, включая контрольную сумму	байт		5	
Пауза между обработкой предыдущего и передачей текущего запроса:				
в режиме *Single Group* (одна группа на все модули)	мкс			330
в режиме *Group per Module* (одна группа на каждый модуль)	мкс	330	400	510
Период обмена данными	мс	5	10	1000
Пропускная способность обмена данными в режиме "1 группа на все модули"	кбайт/с		165	

2.5. Состав поставляемого программного обеспечения

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

Компакт-диск Fastwel I/O Product CD содержит следующие файлы и каталоги:

1. \CoDeSysAdaptation\FastwelCoDeSysAdaptationXXXXXXX.exe – программа установки пакета программ CoDeSys фирмы 3S Smart Software Solution, адаптированного для работы с комплексом Fastwel I/O и MK905
2. \OPC Servers – каталог с программами установки демонстрационных версий OPC-серверов для сетей MODBUS и CAN
3. \Doc – каталог эксплуатационной документации на комплекс Fastwel I/O, включая:
   - *MK905_CoDeSys_Adaptation_UM.pdf* – руководство программиста на MK905.
   - *FIO_CPM71x_CoDeSys_Adaptation_UM.pdf* – руководство программиста на контроллеры CPM71x серии Fastwel I/O
   - *FIO_CPM711_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM711 с внешним интерфейсом CAN и протоколом CANopen
   - *FIO_CPM712_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM712 с внешним интерфейсом MODBUS RTU/ASCII
   - *FIO_CPM713_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM713 с внешним интерфейсом MODBUS/TCP
   - *FIO_CPM70x_CoDeSys_Adaptation_UM.pdf* – руководство программиста на контроллеры CPM70x серии Fastwel I/O
   - *FIO_CPM701_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM701 с внешним интерфейсом CAN и протоколом CANopen
   - *FIO_CPM702_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM702 с внешним интерфейсом MODBUS RTU/ASCII
   - *FIO_CPM703_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM703 с внешним интерфейсом MODBUS/TCP
   - *FIO_CPM704_CoDeSys_Adaptation_UM.pdf* – руководство по конфигурированию и программированию сетевых средств контроллера CPM704 с внешним интерфейсом PROFIBUS DP
   - *FIO_Modules_CoDeSys_Adaptation_UM.pdf* – руководство программиста на модули ввода-вывода Fastwel I/O

– *FIO_UM.pdf* – руководство по эксплуатации аппаратных средств комплекса Fastwel I/O.

4. \Lib – текущие версии библиотек поддержки комплекса Fastwel I/O для CoDeSys
5. \Acrobat – каталог программы просмотра и печати документов в формате Adobe PDF.

2.5.2. Содержимое программы установки Fastwel CoDeSys Adaptation

Программа установки адаптированной среды разработки CoDeSys 2.3 фирмы 3S Smart Software Solution содержит оригинальный установочный комплект пакета CoDeSys 2.3 и файлы, необходимые для адаптации оригинального пакета, включая:

1. Эксплуатационную документацию согласно п. 2.5.1
2. Файлы описания вычислительной платформы *Fastwel MK905 Programmable Automation Controller*.
3. Файлы описаний конфигурации аппаратных средств комплекса Fastwel I/O и MK905 в формате PLC Configuration фирмы 3S Smart Software Solutions (cfg)
4. Файлы modbusDLL.dll и GDrvFastwel.dll, представляющие драйвер коммуникационного сервера CoDeSys Gateway Server, который обеспечивает возможность удаленной загрузки и отладки прикладной программы на контроллере из среды CoDeSys

ВНИМАНИЕ!

Установка файлов поддержки вычислительной платформы MK905 для CoDeSys выполняется автоматически программой установки адаптированного пакета программ CoDeSys. Попытки самостоятельной установки поддержки платформы MK905 при помощи программы InstallTarget могут привести к некорректной работе адаптированной среды разработки CoDeSys.

2.6. Системные требования к рабочему месту разработчика

2.6.1. Требования к аппаратным средствам

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

– процессор Intel Celeron 466 МГц;
– объем установленной оперативной памяти не менее 128 Мбайт;
– размер свободного дискового пространства не менее 512 Мбайт;
– привод CD-ROM

Рекомендуемое разрешение монитора 1280×1024.

Для удаленной загрузки и отладки программного обеспечения в контроллеры через порт COM1 требуется кабель соединительный ФАПИ.685611.012-02, а компьютер должен иметь, как минимум, один коммуникационный порт интерфейса RS-232C.

Для удаленной отладки и загрузки программного обеспечения в контроллер MK905 по протоколу MODBUS RTU или ASCII компьютер должен иметь в своем составе последовательный порт интерфейса RS-232C или RS-485.

Для удаленной отладки и загрузки программного обеспечения в контроллер MK905 по протоколу MODBUS TCP компьютер должен быть оснащен адаптером сети Ethernet 100 Мбит/с.

2.6.2. Требования к системному программному обеспечению

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

– операционная система Windows XP SP3, Windows Vista, Windows 7 не хуже Home Premium;
– для чтения документации в формате Adobe PDF требуется установить программу Adobe Acrobat Reader версии не ниже 6.0.

3. Установка и настройка адаптированной среды CoDeSys 2.3

3.1. Предварительные замечания

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

Если на компьютере уже установлен пакет CoDeSys или какие-либо его компоненты, убедитесь, что версия среды разработки не ниже 2.3.9.36, для чего запустите CoDeSys и выберите команду **About** меню **Help**.

В случае, если компьютер содержит установленный ранее пакет CoDeSys версии ниже 2.3.9.36, программа установки пакета CoDeSys, входящего в установочный комплект адаптации для Fastwel I/O и MK905, выполнит обновление автоматически.

3.2. Установка

Для установки адаптированной версии CoDeSys выполните следующие действия:

1. Запустите программу \CoDeSysAdaptation\setup.exe. Через некоторое время на экран монитора будет выведена диалоговая панель **Welcome**, в которой следует нажать кнопку **Next**
2. Если на компьютере не установлен CoDeSys версии 2.3.9.36 или выше, на экран монитора будет выведено сообщение "*Требуется установить CoDeSys 2.3.9.36 или выше. Вы хотите сделать это сейчас*?". Нажмите кнопку **Yes**
3. Если ранее осуществлялись попытки установки адаптированной среды CoDeSys, на экран монитора может быть выведена диалоговая панель с сообщением *Overwrite Protection*. Если данная диалоговая панель появилась, щелкните в ней на кнопке **Yes to All**, и установка будет продолжена
4. На экран монитора будет выведена диалоговая панель **Выбор языка**, в списке доступных языков которой выберите **Английский** и нажмите **OK**. Через некоторое время на экран монитора будет выведена диалоговая панель с сообщением *Please close all running applications before continuing installation* (Пожалуйста, завершите работу всех запущенным приложений перед продолжением установки)
5. Если в текущий момент запущены какие-либо приложения или компоненты пакета CoDeSys, следует завершить их работу. Остальные приложения завершать не обязательно. Нажмите кнопку **OK** в диалоговой панели. На экран монитора будет выведена диалоговая панель **InstallShield Wizard** с приветствием от программы установки пакета CoDeSys
6. Нажмите кнопку **Next**, после чего выберите каталог установки пакета CoDeSys в диалоговой панели **Choose Destination Location** и нажмите кнопку **Next.**
7. Если на компьютере не установлена предыдущая версия пакета CoDeSys, выберите требуемые дополнительные компоненты пакета CoDeSys в появившейся диалоговой панели **InstallShield Wizard: Select Components**, нажмите кнопку **Next**, после чего в каждой из последующих двух диалоговых панелях вновь нажмите **Next**
8. Программа установки выполнит установку пакета CoDeSys
9. Незадолго до окончания установки на экран монитора будет выведено напоминание о том, что некоторые продукты, входящие в пакет CoDeSys, требуют наличия оплаченной лицензии. Нажмите **OK** в диалоговой панели данного сообщения, а затем **Finish.**
10. На экране появится диалоговая панель **Choose Destination Location**, в которой имеется возможность указать каталог установки файлов адаптации среды CoDeSys и документация. Нажмите **Next** в данной диалоговой панели, а затем **Finish**. Установка адаптированной среды CoDeSys на этом завершена.

3.3. Проверка установки

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

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

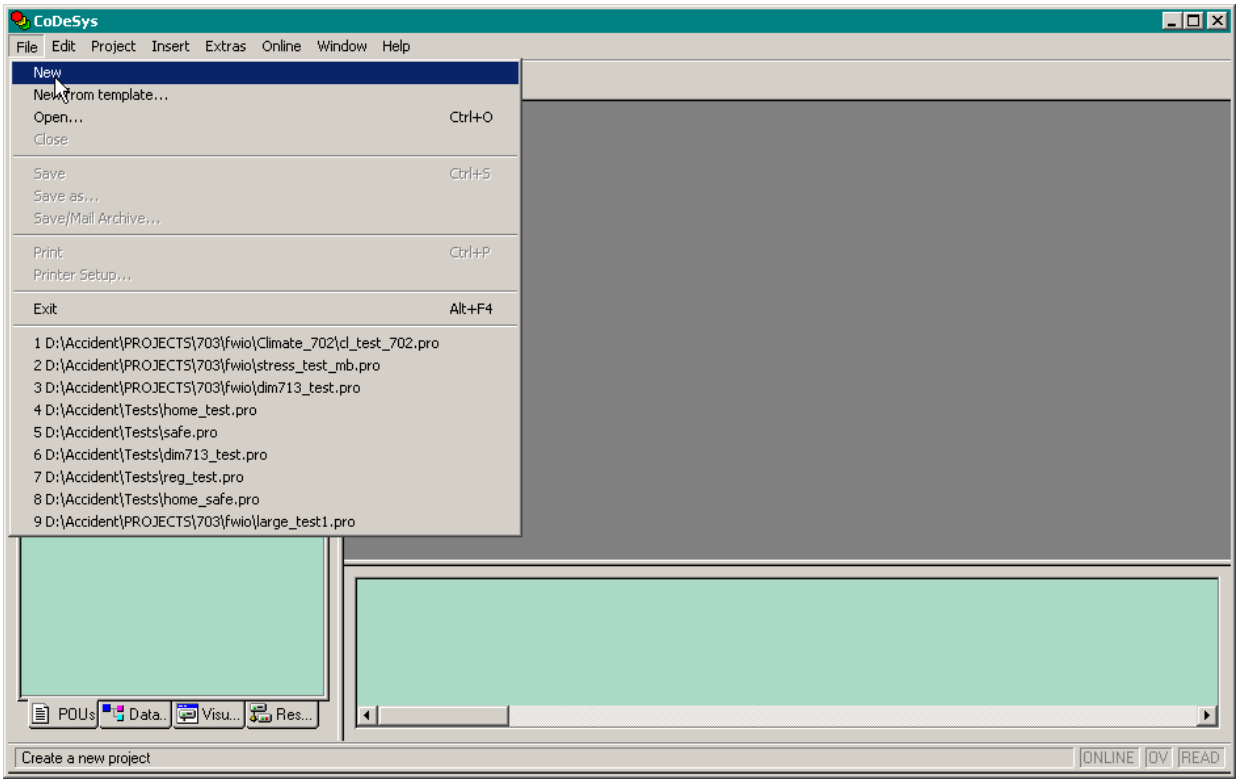


Рис. 4. Главное окно среды разработки CoDeSys

2. Выберите команду **New** в меню **File**. В появившейся диалоговой панели **Target Settings** (Параметры платформы) в выпадающем списке **Configuration** выберите опцию *Fastwel MK905 Programmable Automation Controller*. Диалоговая панель **Target Settings** примет вид, показанный на рис. 5.

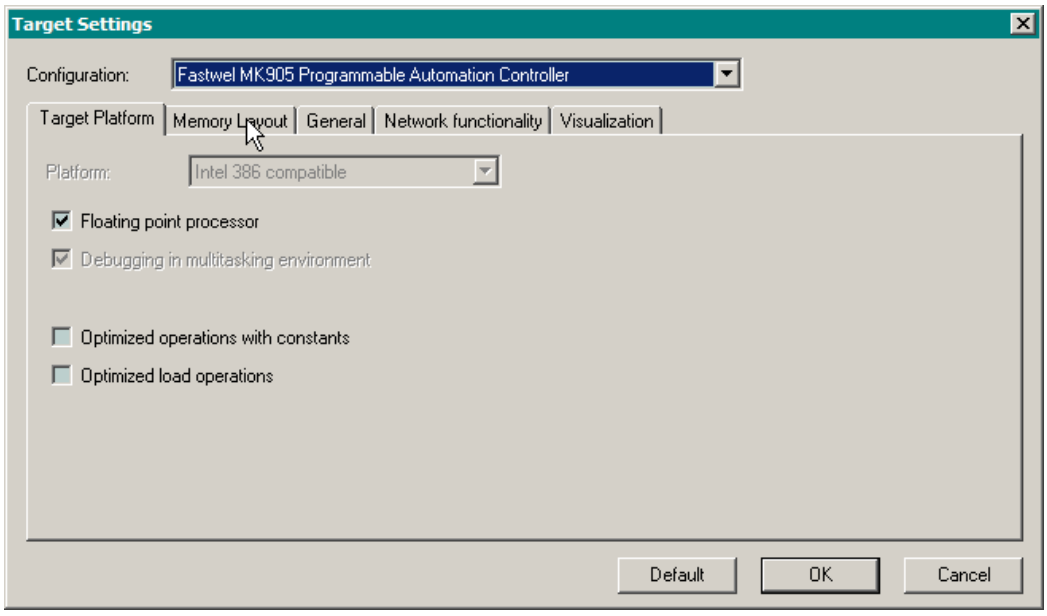


Рис. 5. Диалоговая панель Target Settings после выбора платформы Fastwel MK905 Programmable Automation Controller

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

левой области главного окна IDE CoDeSys и дважды щелкните на узле **PLC Configuration** в дереве **Resources**. В левой области главного окна IDE CoDeSys появится окно настройки конфигурации контроллера, показанное на рис. 6.

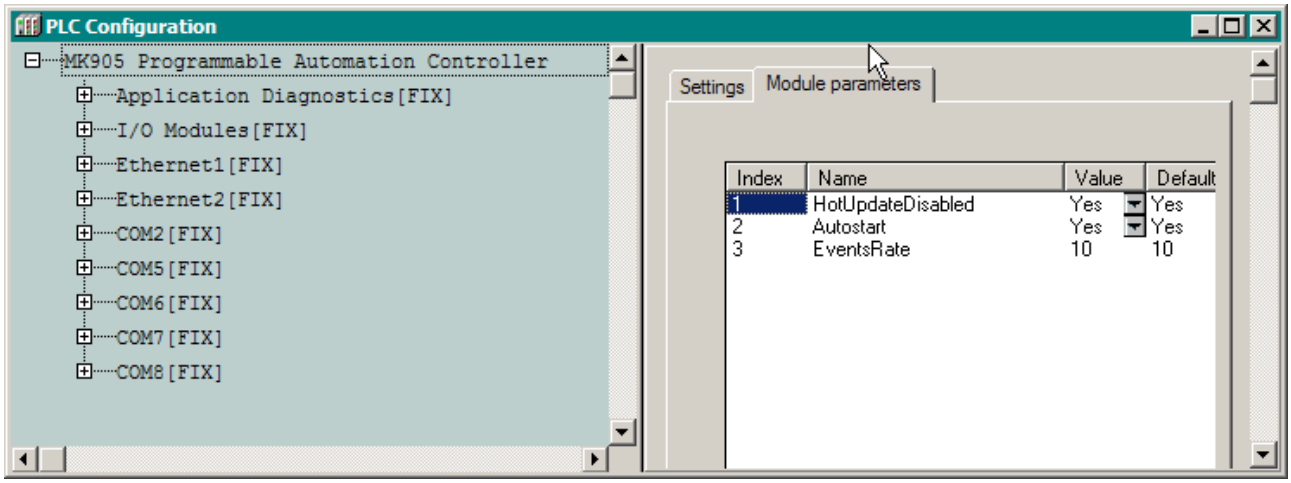


Рис. 6. Окно PLC Configuration при создании проекта для платформы Fastwel MK905 Programmable Automation Controller

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

3.4. Настройка параметров трансляции проекта CoDeSys

После создания проекта для платформы MK905 в IDE CoDeSys для получения детальной диагностической информации о доступе к областям входных и выходных данных программы при трансляции проекта:

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

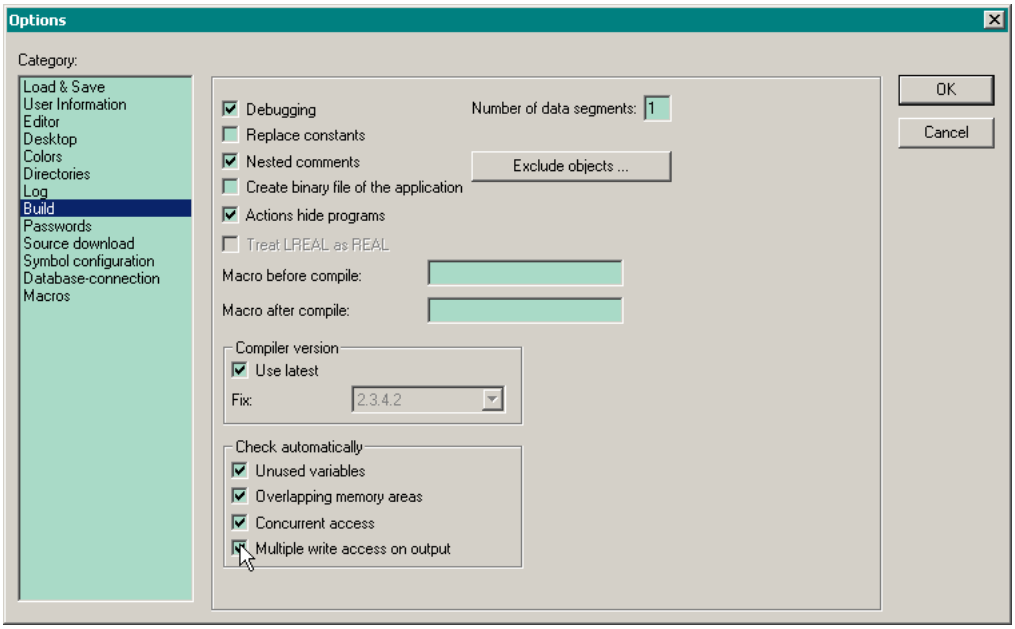


Рис. 7. Настройка параметров трансляции проекта IDE CoDeSys

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

нескольких выходных переменных, отображаемых на перекрывающиеся участки в области выходных данных программы.

3.5. Удаление адаптированной среды разработки CoDeSys

Для удаления адаптированной среды разработки CoDeSys:

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

3.6. Обновление системного программного обеспечения MK905

3.6.1. Общие сведения

Для просмотра текущей версии системы исполнения MK905 воспользуйтесь указаниями п. 9.1 настоящего документа.

Обновление системного программного обеспечения контроллера MK905 состоит в загрузке в контроллер файла *norm.dnl*, предварительно загруженного с ftp-узла фирмы Прософт по адресу:

ftp://ftp.prosoft.ru/pub/hardware/Fastwel/Fastwel_IO/Version2/Firmware/MK905/

Если загрузка файла в контроллер выполнялась из CoDeSys IDE, после успешной загрузки файла *norm.dnl* произойдет перезапуск системы исполнения контроллера.

3.6.2. Обновление из среды CoDeSys

1. Если в контроллере не имеется загруженной программы или отсутствует файла проекта CoDeSys, из которого была получена текущая функционирующая программа, откройте в среде разработки проект default.pro, находящийся в подкаталоге \MK905\SAFE каталога установки файлов адаптации CoDeSys для Fastwel I/O (по умолчанию – C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Examples).
   Если в контроллере исполняется приложение, откройте в CoDeSys IDE проект приложения.
2. Если контроллер подключен к сети (MODBUS или MODBUS TCP) в качестве подчиненного узла, установите соединение между средой разработки CoDeSys и контроллером через соответствующий коммуникационный канал согласно указаниям руководства программиста на контроллер путем выполнения команды **Online–Login** в среде разработки CoDeSys.
3. Если контроллер не подключен к сети, соедините порт COM1 с последовательным портом компьютера, на котором установлена среда разработки CoDeSys, при помощи нуль-модемного кабеля ASC00042, после чего установите соединение между средой разработки CoDeSys и контроллером через коммуникационный канал P2P.
4. Если после выполнения команды **Online-Login** на экране монитора появится диалоговая панель *The program has changed...*, нажмите в ней кнопку **Cancel**.
5. Выполните команду **Online–Write file to PLC** в среде разработки CoDeSys и в появившейся диалоговой панели **Write file to PLC** выберите файл norm.dnl и нажмите кнопку **Open**. На экране монитора появится окно, отображающее прогресс загрузки файла в контроллер.

6. По окончании загрузки файла контроллер автоматически перезапустит системное программное обеспечение и возобновит исполнение прикладной программы. При этом произойдет разрыв соединения со средой разработки CoDeSys.

3.7. Обновление по протоколу ftp

Загрузите файл norm.dnl по протоколу ftp в соответствии с указаниями п. 8.2 настоящего руководства, после чего выключите и повторно включите питание МК905.

3.8. Режим совместимости с CPM90201

3.8.1. Активизация режима совместимости с CPM90201

Среда исполнения приложений CoDeSys MK905 обеспечивает возможность функционирования в режиме совместимости с CPM90201. При работе в режиме совместимости в контроллер могут быть загружены приложения, ранее разработанные в среде CoDeSys IDE для платформы *CPB902 WinCE Runtime*.

Для активизации режима совместимости загрузите в МК905 текстовый файл с именем *target.txt*, содержащий единственную строку:

```
CPM902
```

Файл может быть загружен из среды CoDeSys IDE в соответствии с указаниями п. 5.9, либо по протоколу ftp в соответствии с указаниями п. 8.2, либо путем использования USB-накопителя, подключенного к одному из гнезд USB, в соответствии с указаниями п. 8.3 настоящего руководства.

Файл должен находиться в корневом каталоге NAND Flash-накопителя с именем FixedDisk1.

После загрузки файла *target.txt* необходимо перезапустить МК905.

3.8.2. Отмена режима совместимости с CPM90201

Для прекращения работы МК905 в режиме совместимости с CPM90201 следует либо удалить файл *target.txt* из корневого каталога FixedDisk1, либо удалить/изменить находящуюся в нем строку *CPM902*.

Удалить файл можно либо с использованием ftp-клиента (например, Total Commander) путем подключения к МК905 в соответствии с указаниями п. 8.2, либо непосредственно, подключив к МК905 монитор, клавиатуру и мышь, как указано в п. 8.3.2 настоящего руководства.

4. Принцип работы программного обеспечения контроллера

4.1. Общие сведения

4.1.1. Приложение пользователя

Контроллеры серии Fastwel I/O и MK905 относятся к классу программируемых логических контроллеров, т.е. для того, чтобы контроллер начал выполнять какую-либо полезную работу, пользователь должен создать приложение в адаптированной среде разработки CoDeSys и загрузить в контроллер.

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

1. Как минимум, одну программу – программную единицу типа *PROGRAM*, добавляемую в древовидный список проекта **POUs**, показанный на рис. 8, по команде контекстного меню **Add Object**.

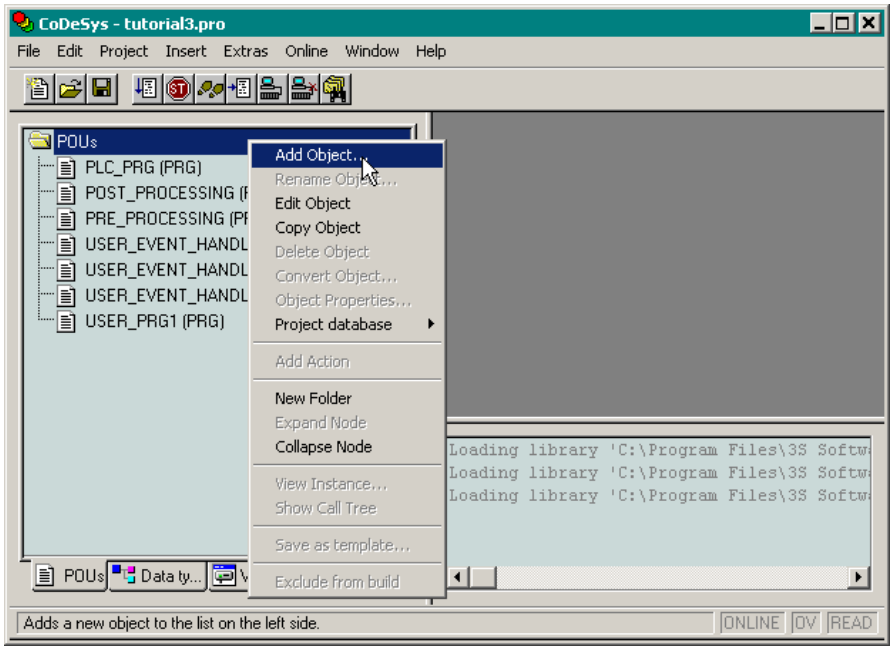


Рис. 8. Добавление программной единицы в древовидный список POU

2. Множество циклических задач. Циклической задачей является вычислительный процесс типа *Cyclic Task*, описываемый пользователем в окне ресурсе 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 с периодом задачи в порядке следования в списке POU данной задачи, как показано на рис. 9.
   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.

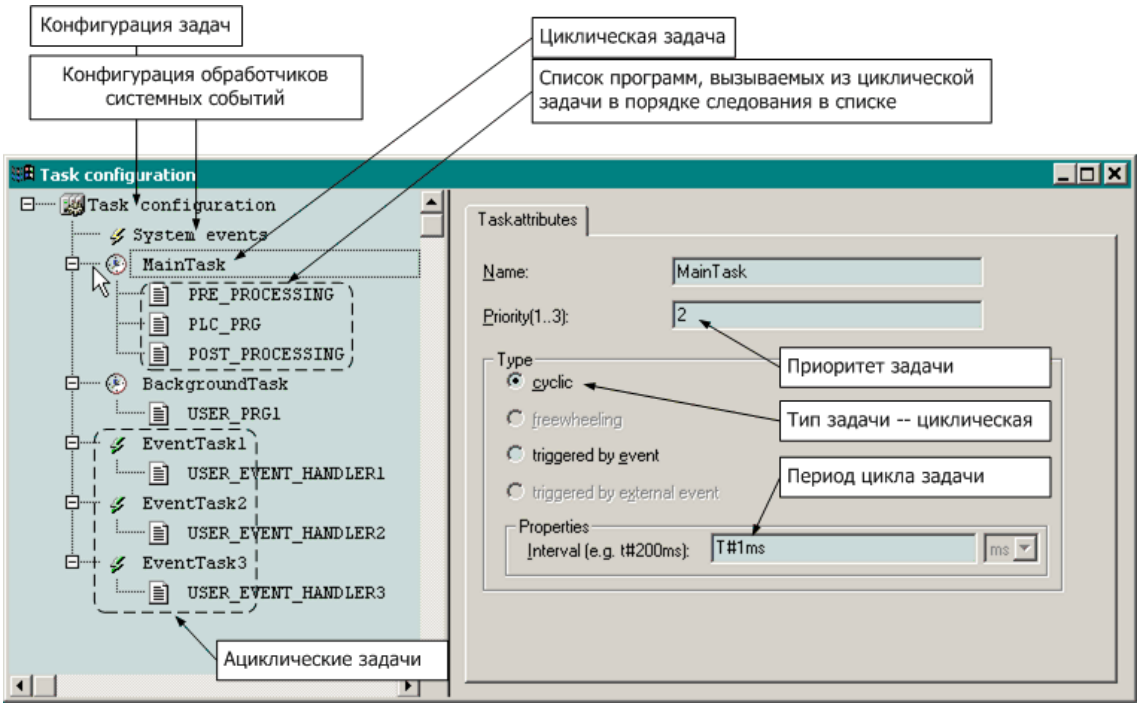


Рис. 9. Пример конфигурации задач приложения пользователя

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 или переменной чувствительности), определенной в свойствах задачи параметром **Task Attributes–Properties–Event**.
   Проверка изменений переменных чувствительности выполняется на контексте высокоприоритетного сервисного потока операционной системы контроллера с периодом, который определен пользователем при помощи параметра *MK905 Programmable Automation Controller:EventsRate* (по умолчанию – 10 мс) в окне ресурса **PLC Configuration**.
   Помимо переменной чувствительности, 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 чувствительности) порядок запуска ациклических задач определяется соотношением их приоритетов (выше приоритет – раньше запуск), а, в случае равенства, – порядковым номером задачи (меньше номер – раньше запуск).

Примечание. Если в окно ресурса **Tasks Configuration** не добавлено ни одной задачи любого из поддерживаемых типов и не активизированы функции целевой визуализации, среда разработки CoDeSys автоматически создаст описание циклической задачи с именем *DefaultTask* и ассоциирует с ней программу PLC_PRG (если она имеется в проекте). Период цикла данной задачи будет равен значению параметра *MK905 Programmable Automation Controller:EventsRate* (по умолчанию – 10 мс) в окне ресурса **PLC Configuration**.

4. Множество обработчиков системных событий. Обработчиком системного события называется функция (программная единица типа *FUNCTION*), назначенная пользователем для одного или нескольких системных событий во вкладке **Tasks Configuration–System events**. К системным событиям относятся моменты запуска и останова приложения, окончание подготовки приложения к запуску, начало загрузки нового приложения, момент перед заменой текущего приложения на вновь загруженное и другие.

ВНИМАНИЕ!

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

5. Конфигурация контроллера, состоящая из множества 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. 10. В конфигурации, помимо описаний объектов разных подсистем контроллера, определяется структура образа процесса, а также могут быть объявлены символьные адреса каналов ввода-вывода для последующего использования в приложении.
Кроме того, в конфигурации определяются значения различных параметров подсистем контроллера, и имеются каналы доступа к диагностической информации о работе подсистем.

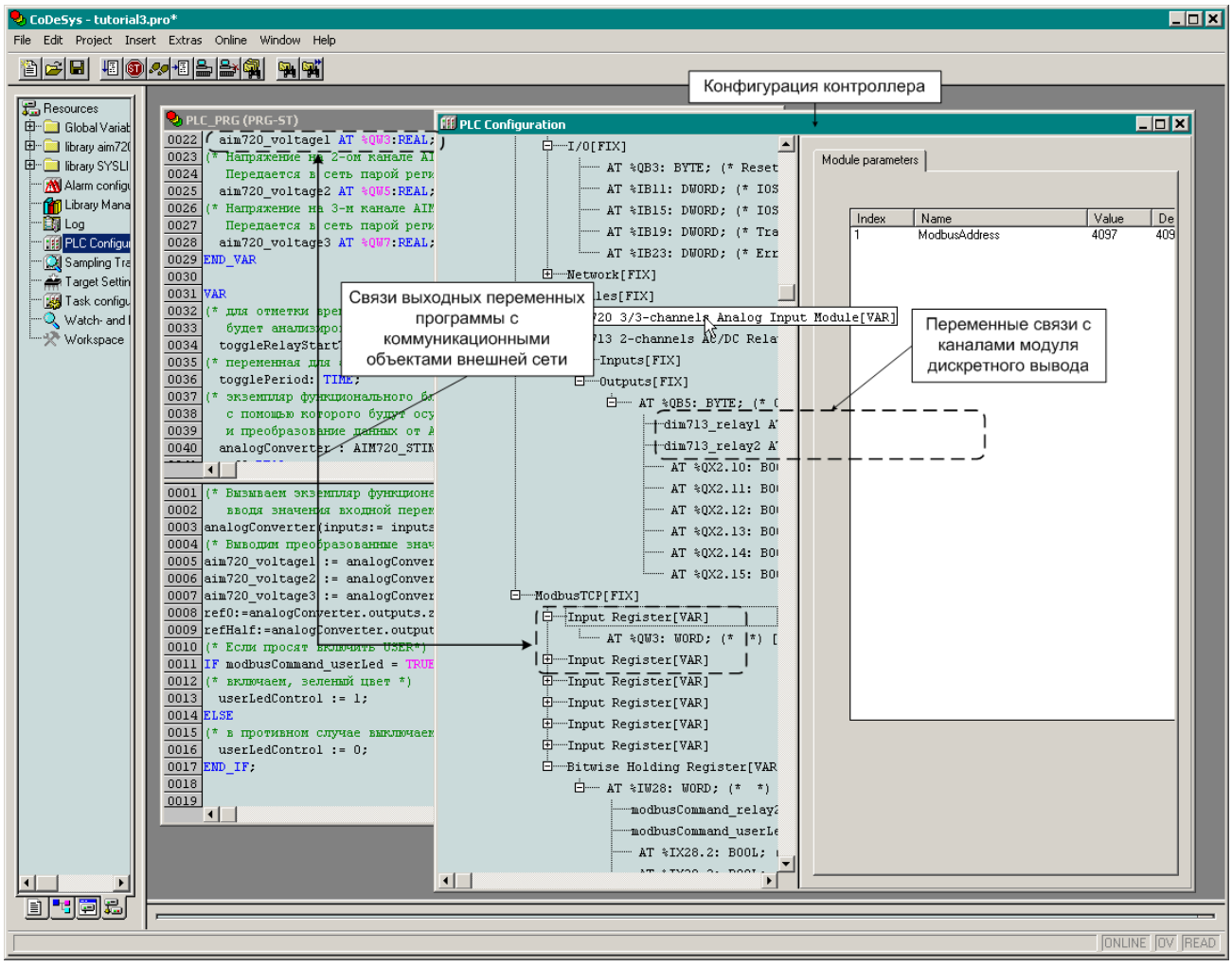


Рис. 10. Конфигурация контроллера

6. Вспомогательные файлы подсистемы целевой визуализации – файлы изображений, языковой поддержки и другие, требуемые для функционирования подсистемы целевой визуализации

4.1.2. Назначение системного программного обеспечения контроллера

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

Системное программное обеспечение контроллера состоит из следующих основных сервисов:

1. Адаптированная среда исполнения CoDeSys – предназначена для исполнения кода приложения пользователя.
2. Сервис ввода-вывода – предназначен для управления, инициализации и обмена данными с модулями ввода-вывода, подключенными в внутренней шине контроллера. Обеспечивает функционирование стека протоколов внутренней шины контроллера, а также инициализацию, обнаружение и обработку ошибок обмена по внутренней шине и т.п.
3. Сервис внешней сети – состоит из стека протоколов внешней сети и сервисов протоколов прикладного уровня MODBUS RTU/ASCII и MODBUS TCP. Предназначен для обмена данными и командами с контроллером по сети. Описание принципов работы и способов конфигурировании перечисленных сервисов внешней сети приведено в разделе 7 настоящего руководства.

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

4.2. Принципы работы адаптированной среды исполнения CoDeSys

4.2.1. Режимы работы

4.2.1.1. Безопасный режим

При поставке контроллер не содержит приложения пользователя и при включении питания запускается в так называемом безопасном режиме, о чем свидетельствует прерывистое свечение индикатора LED1 с частотой около 5 Гц. В безопасном режиме системное программное обеспечение контроллера не обращается к модулям ввода-вывода и ожидает загрузки приложения.

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

ВНИМАНИЕ!

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

При перезапуске контроллера, функционирующего в безопасном режиме по ошибке в ранее загруженном приложении, происходит сброс информации о последней причине перехода безопасный режим (однако в файле rstat.bin хранятся причины последних 32-х перезагрузок). В связи с этим до установления причины перехода в безопасный режим не рекомендуется перезапускать контроллер.

Для установления причины перехода в безопасный режим:

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. Выполните команду **Online–Logout**
4. Откройте выгруженный файл *normdump.txt* текстовым редактором, поддерживающим кодировку Unicode, и прочитайте причину перехода в безопасный режим.

Если не удается самостоятельно установить и устранить причину перехода в безопасный режим, выполните следующие действия:

1. В среде разработки откройте проект, загрузка которого в контроллер привела к переходу в безопасный режим.
2. Выполните команду **File–Save/Mail Archive**
3. Отправьте выгруженные из контроллера файлы *normdump.txt* и *rstat.bin,* а также файл архива, сформированный средой разработки CoDeSys, по электронной почте на адрес soft.support@fastwel.ru.
   В сообщении, отправляемом по электронной почте, укажите:
   название своего предприятия,
   фамилию и инициалы,
   фактическую аппаратную конфигурацию контроллера (тип контроллера, номенклатуру модулей ввода-вывода в порядке подключения к контроллеру и тип используемого источника питания),
   обстоятельства, при которых контроллер перешел в безопасный режим.

4.2.1.2. Нормальный режим

Если в контроллере имеется успешно загруженное приложение пользователя, при включении питания или после перезапуска 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 и формирование запросов внешней сети и т.д. При этом индикаторы контроллера будут светиться следующим образом:

LED1:

1. Прерывистое свечение с частотой около 0,5 Гц – все циклические задачи укладываются в заданный период
2. Прерывистое свечение с частотой около 1 Гц – хотя бы одна циклическая задача не 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.

LED2:

1. Непрерывное свечение – 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 **Период опроса, мс** в конфигурации сервиса ввода-вывода контроллера (**Resources–PLC Configuration**–*MK905 Programmable Automation Controller–I/O Modules*).
2. Прерывистое свечение – обмен с модулями ввода-вывода, подключенными к контроллеру, не может быть выполнен за время, заданное параметром **Период опроса, мс** в конфигурации контроллера (**Resources–PLC Configuration**– *MK905 Programmable Automation Controller –I/O Modules*), в связи с чем используется расчетное минимально достижимое время обмена данными со всеми модулями при загрузке внутренней шины на 50%.
3. Отсутствие свечения – в конфигурации загруженного приложения отсутствуют описания модулей ввода-вывода или конфигурация модулей ввода-вывода, определенная в проекте, не совпадает с аппаратной конфигурацией модулей, подключенных к контроллеру.

4.2.2. Процесс запуска контроллера после включения питания

4.2.2.1. Запуск при первом включении питания

При поставке контроллер не содержит пользовательского приложения и при включении питания запускается в безопасном режиме, о чем свидетельствует прерывистое свечение индикатора LED1 с частотой около 5 Гц. В безопасном режиме системное программное обеспечение контроллера не

обращается к модулям ввода-вывода и ожидает загрузки в контроллер приложения, разработанного в среде CoDeSys. Более подробная информация об условиях запуска контроллера в безопасном режиме и об индикации безопасного режима приведена в п. 4.2.1.1 настоящего руководства.

4.2.2.2. Запуск при наличии загруженного приложения

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

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

10. Выполняется загрузка секции конфигурации задач, а затем создание и конфигурирование циклических и ациклических задач, добавленных пользователем в окне **Tasks Configuration** загруженного проекта. Помимо этого выполняется анализ исполняемого кода с целью определения подмножеств программных единиц, вызываемых из каждой задачи. Последнее необходимо для отладчика среды исполнения, а также для последующего формирования диагностики в случае наличия фатальных ошибок в сгенерированном коде.
11. Загружается секция описателей связей задач с образом процесса, после чего сортировка описателей в порядке возрастания смещений, а далее – объединение описателей связей со смежными участками указанных областей.
12. Загружается секция информации о проекте.
13. На основе дерева конфигурации контроллера, построенного в п. 5, выполняется конфигурирование сервиса внешней сети, в процессе чего создаются входящие и исходящие коммуникационные объекты, а также устанавливаются параметры протокола. Параметры протокола (скорость обмена, адрес и т.п.), успешно извлеченные из дерева конфигурации контроллера, запоминаются в энергонезависимой памяти контроллера, чтобы в случае перезапуска в безопасный режим по ошибке использовать их для работы сервиса внешней сети.
14. На основе дерева конфигурации контроллера, построенного в п. 5, выполняется конфигурирование сервиса ввода-вывода, в процессе чего создается список описателей модулей ввода-вывода, которые должны быть подключены к внутренней шине контроллера, затем выполняется поиск модулей, фактически подключенных к контроллеру, а далее – конфигурирование обнаруженных модулей, типы и местоположение которых на шине соответствуют ожидаемым в проекте при условии, что их текущие значения параметров отличаются от полученных из запускаемого приложения.
15. В зависимости от режима, заданного пользователем с помощью параметра *MK905 Programmable Automation Controller –I/O Modules:ScanMode*, создается одна группа обмена с модулями ввода-вывода или множество групп, количество которых совпадает с количеством модулей ввода-вывода в поддереве *MK905 Programmable Automation Controller –I/O Modules* конфигурации контроллера. Для групп устанавливается период, определяемый параметром *MK905 Programmable Automation Controller –I/O Modules:SampleRate*. Если процесс конфигурирования сервиса ввода-вывода завершился не вполне успешно, контроллер продолжит работу в нормальном режиме с соответствующей индикацией.
16. Выполняется связывание с образом процесса всех объектов системы, нуждающихся в обмене данными (задач CoDeSys, коммуникационных объектов и групп обмена с модулями ввода-вывода) во время работы приложения. Если в процессе связывания обнаружены какие-либо ошибки, контроллер перезапускается и стартует в безопасном режиме с соответствующей индикацией. По окончании связывания вызываются пользовательские функции-обработчики системных событий *OnPowerOn*, а затем – *OnInit* (если они были установлены пользователем в приложении).
17. Происходит запуск обмена данными с модулями ввода-вывода, запускается сервис внешней сети, а также все задачи адаптированной среды исполнения CoDeSys. Непосредственно перед запуском вызывается пользовательская функция-обработчик системного события *OnStart* (если она была установлена пользователем в приложении).
18. Системное программное обеспечение контроллера функционирует в нормальном режиме, исполняя алгоритмы приложения, обмениваясь данными с модулями ввода-вывода и по внешней сети, а также выполняя диагностику всех подсистем.

4.2.3. Процесс загрузки или обновления приложения

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

Процесс загрузки или обновления приложения начинается, когда пользователь, предварительно настроив коммуникационный канал взаимодействия с контроллером через интерфейс P2P или через интерфейс внешней сети, выполняет команду **Online–Login** в среде разработки CoDeSys и нажимает

кнопку **Yes** в появившейся диалоговой панели *The program has changed! Download the new program?* Указанная диалоговая панель появляется в случае, если проект, открытый в среде CoDeSys, отличается от исполняющегося в контроллере хотя бы датой и временем компиляции. При изложении последующего описания процесса загрузки и обновления приложения предполагается, что контроллер функционирует в нормальном режиме согласно п. 4.2.4. Загрузка приложения при работе контроллера в безопасном режиме выполняется аналогично.

1. После того, как пользователь в среде разработки CoDeSys нажимает кнопку **Yes** в диалоговой панели с сообщением *The program has changed! Download the new program?*, среда разработки начинает загрузку секции исполняемого кода текущего загружаемого приложения. В диалоговой панели статуса загрузки изменяется значение счетчика загруженных байт. В момент начала загрузки секции кода вызывается пользовательская функция-обработчик системного события *OnDownload* (если она была установлена пользователем в приложении). Во время загрузки секций нового приложения работа текущего приложения контроллера, обмен данными с модулями ввода-вывода и обмен по сети, не прекращаются.
2. По окончании загрузки секции кода происходит запись секции во вторичное хранилище приложения в энергонезависимой памяти контроллера (хранилище для загружаемого нового приложения). В случае ошибки записи секции кода среде разработки передается код ошибки 20014 (ошибка загрузки секции кода). После успешного сохранения секции кода адаптированная среда исполнения CoDeSys контроллера ожидает секцию конфигурации загружаемого приложения.
3. Среда разработки CoDeSys, убедившись, что секция кода загружена успешно, приступает к загрузке секции конфигурации приложения, определяемой в окне **PLC Configuration**. В диалоговой панели статуса загрузки прекращается изменение значения счетчика загруженных байт.
4. По окончании загрузки секция конфигурации записывается во вторичное хранилище. В случае ошибки записи среде разработки передается код ошибки 20015 (ошибка загрузки секции конфигурации). После успешного сохранения секции конфигурации контроллера адаптированная среда исполнения CoDeSys контроллера ожидает секцию конфигурации задач загружаемого приложения.
5. Среда разработки CoDeSys, убедившись, что секция конфигурации загружена успешно, приступает к загрузке секции конфигурации задач, определяемой в окне **Tasks Configuration**. В диалоговой панели статуса загрузки изменение значения счетчика загруженных байт по-прежнему не происходит.
6. По окончании загрузки секции конфигурации задач происходит запись секции во вторичное хранилище приложения в энергонезависимой памяти контроллера. В случае ошибки записи секции конфигурации задач среде разработки передается код ошибки 20016 (ошибка загрузки секции конфигурации задач). После успешного сохранения адаптированная среда исполнения CoDeSys контроллера ожидает секцию информации о проекте загружаемого приложения.
7. Среда разработки CoDeSys, убедившись, что секция конфигурации задач загружена успешно, приступает к загрузке секции информации о проекте загружаемого приложения, определяемой пользователем в диалоговой панели **Project Information**, вызываемой по команде меню **Project–Project Info.** В диалоговой панели статуса загрузки изменение значения счетчика загруженных байт не происходит.
8. По окончании загрузки секции информации о проекте загружаемого приложения выполняется запись секции во вторичное хранилище приложения в энергонезависимой памяти контроллера. В случае ошибки записи секции среде разработки передается код ошибки 20018 (ошибка загрузки секции информации о проекте). После успешного сохранения секции адаптированная среда исполнения CoDeSys контроллера ожидает секцию описателей связей задач загружаемого приложения (ссылок на образ процесса).
9. Среда разработки CoDeSys, убедившись, что секция информации о проекте загружена успешно, приступает к загрузке секции ссылок задач загружаемого приложения на образ процесса. Содержимое секции определяется созданными пользователем переменными программ, ссылающимися на непосредственные адреса в областях входных и выходных данных. В диалоговой панели статуса загрузки значение счетчика загруженных байт по-прежнему не изменяется.

10. По окончании загрузки секции ссылок задач на образ процесса происходит сохранение данной секции во вторичное хранилище приложения в энергонезависимой памяти контроллера. В случае ошибки записи секции среде разработки передается код ошибки 20017 (ошибка загрузки проекта).
11. Система исполнения в течение 2,5 с ожидает начала загрузки вспомогательных файлов подсистемы целевой визуализации. Загрузка файлов сопровождается отображением информации о загружаемых файлах в диалоговой панели статуса загрузки среды разработки CoDeSys.
12. После успешной загрузки и сохранения последней секции загружаемого приложения (ссылок задач на образ процесса), системное программное обеспечение контроллера выполняет проверку целостности секций, загруженных во вторичное хранилище в соответствии с шагом 3 п. 4.2.2.2, после чего, при равенстве длин, выполняется побайтовое сравнение только что загруженных секций во вторичном хранилище с содержимым однотипных секций в первичном хранилище, откуда был осуществлен успешный запуск текущего приложения. Если никаких отличий не обнаружено, контроллер продолжает функционировать в нормальном режиме, исполняя ранее загруженное приложение.
13. Если в какой-либо секции вторичного хранилища обнаружены какие-либо отличия от содержимого однотипной секции первичного хранилища, контроллер приостанавливает исполнение приложения и обмен данными по сети и с модулями ввода-вывода по внутренней шине и, в зависимости от типов секций, где обнаружены изменения, выполняется выборочное или полное конфигурирование сервисов контроллера согласно последовательности шагов 5–17 п. 4.2.2.2. Перед выполнением указанных действий вызывается пользовательская функция-обработчик системного события *OnProgramChange* (если она была установлена пользователем в приложении).
Полный перечень реконфигурирующих действий по указанным шагам выполняется в случае, если изменилось содержимое всех секций приложения или если в секции информации о проекте изменились название проекта, имя файла проекта, информация о версии, информация об авторе проекта или краткое описание проекта (см. диалоговую панель **Project Information** в среде CoDeSys).
Если какая-либо секция во вновь загруженном приложении оказалась неизменной, соответствующие реконфигурирующие действия не выполняются.
14. Если секция кода изменилась, но параметр *MK905 Programmable Automation Controller:HotUpdateDisabled (горячее обновление отключено)* установлен пользователем в *No (Нет)*, выполняются дополнительные действия, цель которых – по возможности сохранить неизменными значения всех внутренних, входных и выходных переменных предыдущего приложения, которое подвергается обновлению.
Значения внутренних, входных и выходных переменных предыдущего приложения сохраняются неизменными, если:

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

15. По окончании этапа связывания объектов, нуждающихся в обмене данными, с образом процесса пользовательская функция-обработчик системного событий *OnPowerOn* не вызывается, но последовательно вызываются *OnInit* и *OnStart*.
16. Производится перестановка первичного и вторичного хранилищ приложения: первичное хранилище, в котором находилось старое приложение, помечается, как пустое, после чего делается вторичным. Вторичное хранилище, в которое было загружено новое приложение, делается первичным.

4.2.4. Исполнение приложения пользователя

4.2.4.1. Общие сведения

Описание состава приложения пользователя приведено в п. 4.1.1. Сокращенная архитектура адаптированной среды исполнения CoDeSys в упрощенной нотации UML представлена на рис. 11.

Одним из основных активных элементов архитектуры адаптированной среды исполнения CoDeSys является сервисная задача, исполняемая на контексте отдельного высокоприоритетного потока операционной системы. Сервисной задаче принадлежат множества объектов, представляющих циклические и ациклические задачи, описания которых формируются средой разработки CoDeSys, а также глобальная область данных, полученная у операционной системы на основе информации, сгенерированной средой разработки.

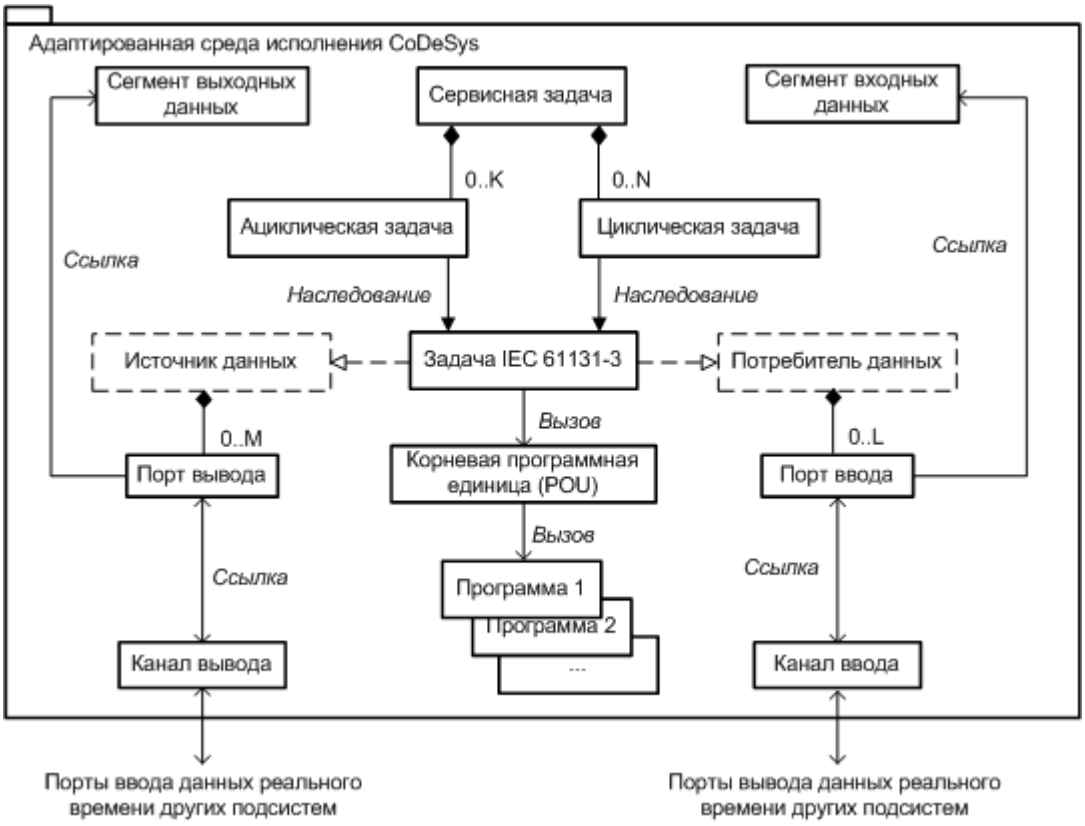


Рис. 11. Архитектура адаптированной среды исполнения CoDeSys

На сервисную задачу возлагается выполнение следующих основных функций:

1. Прием и обслуживание запросов среды разработки CoDeSys, поступающих по сети или через интерфейс прямого соединения P2P, включая чтение/запись переменных, загрузку приложения, чтение/запись файлов, вставку и снятие точек останова во время отладки и т.п.
2. Вызов большинства пользовательских функций обработки системных событий
3. Управление ациклическими задачами, включая проверку переменных-событий ациклических задач на наличие перехода их значений с FALSE к TRUE, обмен (ввод-вывод) данными ациклических задач с образом процесса и вызов корневых программных единиц ациклических задач. Корневая программная единица некоторой задачи является функцией, которая сгенерирована компилятором среды разработки CoDeSys, и содержит вызовы (запуски) программ, ассоциированных пользователем с данной задачей, в порядке, установленном для программ задачи в ресурсе **Tasks Configuration**.
4. Диагностику работы циклических задач, включая анализ значений счетчиков циклов и запаздываний каждой циклической задачи, обновление соответствующей информации в подобласти диагностики системы образа процесса, а также перевод контроллера в безопасный режим в случае, если какая-либо из циклических задач не выполнила ни одного цикла в течение более чем 30 с.

5. Взаимодействие со сторожевым таймером с целью предотвращения зависания контроллера при вызовах пользовательских обработчиков системных событий или ациклических задач.
6. Сохранение энергонезависимых (*VAR RETAIN*) переменных в энергонезависимой памяти контроллера.

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

Корневой программной единицей является скрытая от пользователя программа, автоматически создаваемая компилятором, в которую вставлены вызовы ассоциированных с пользовательской задачей программ в порядке их перечисления в окне ресурса **Tasks Configuration**, как показано на рис. 12.

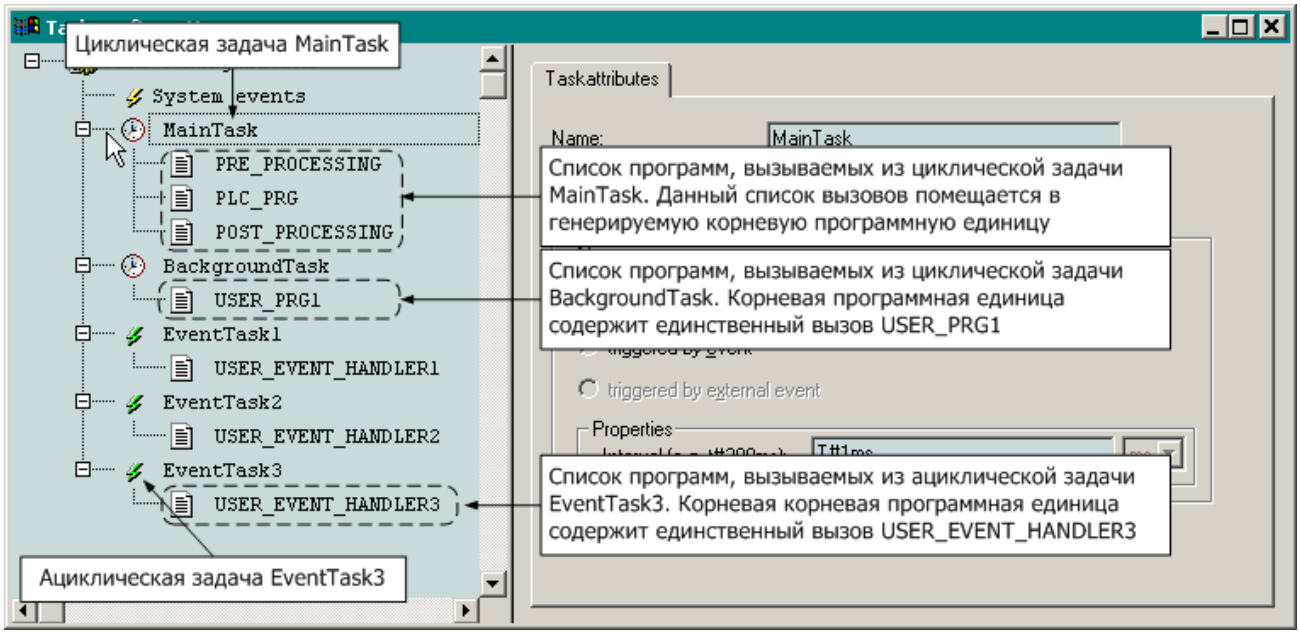


Рис. 12. Списки программных единиц, вызываемых из задач

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

Принцип формирования описателей ссылок иллюстрируется рис. 13. 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 равна 0.

ВНИМАНИЕ!

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

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

Перед началом связывания по оптимизированным множествам описателей ссылок задач на образ процесса для каждой пользовательской задачи сначала создаются специальные объекты сервиса обмена данными реального времени контроллера, называемые входными и выходными портами, через которые осуществляются чтение и запись образа процесса. Кроме того, перед связыванием создаются два объекта связи, представляющие образ процесса для всех источников и потребителей данных реального времени системы и называемые каналами ввода и вывода.

Глобальная область памяти, выделяемая адаптированной средой исполнения на основе соответствующей информации в загруженном приложении пользователя, разделена на следующие основные сегменты: сегмент входных данных, сегмент выходных данных, сегмент энергонезависимых переменных и сегмент внутренних/глобальных переменных и экземпляров функциональных блоков (экземпляр – объект-переменная некоторого типа, каковым является функциональный блок).

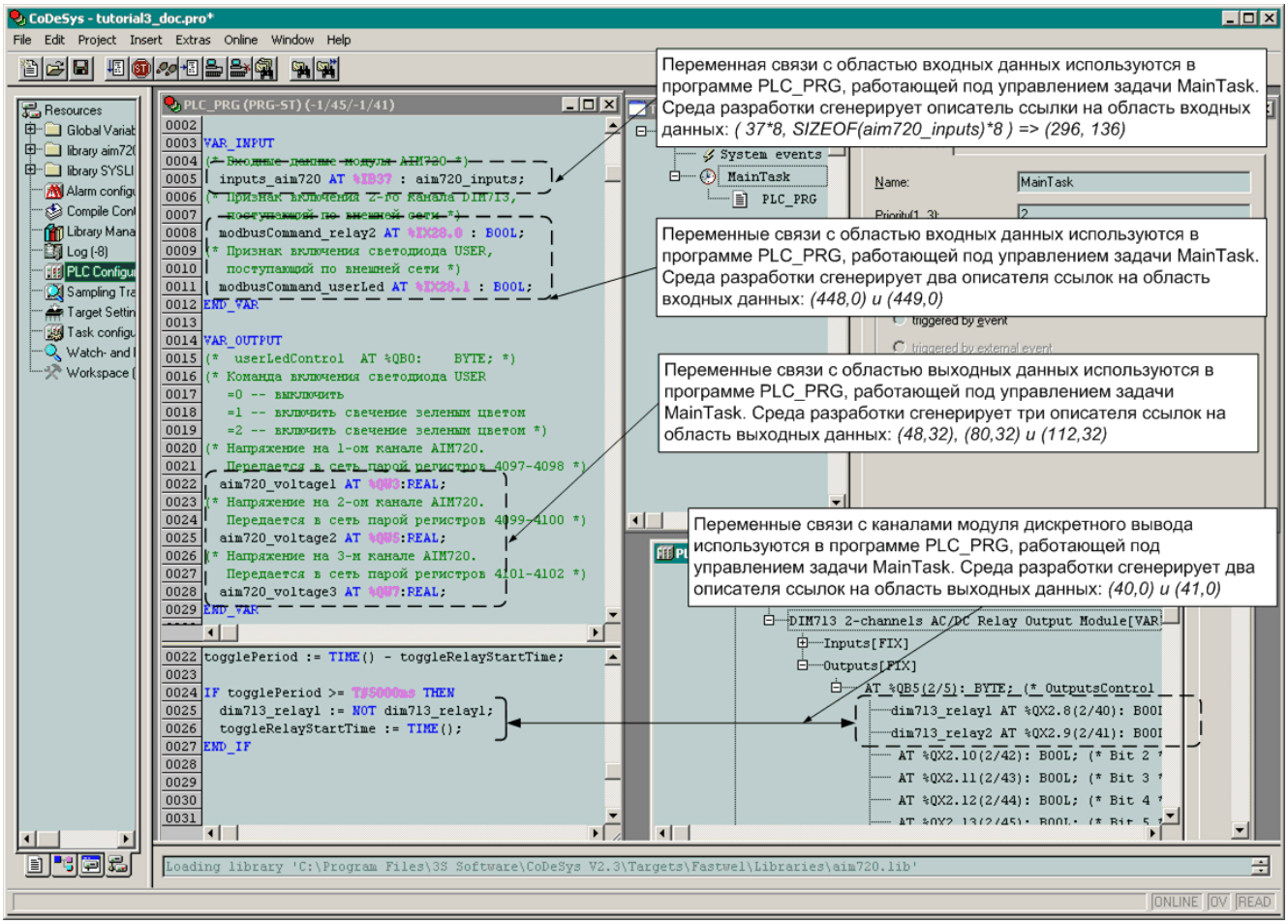


Рис. 13. Принцип формирования описателей ссылок на области входных и выходных данных приложения

Переменные, ссылающиеся на адреса во входной области образа процесса, размещаются компилятором в индивидуальном входном сегменте данных каждой задачи, а переменные, ссылающиеся на адреса в выходной области образа процесса – в выходном сегменте.

Каждый входной порт задачи содержит смещение одиночной входной переменной или группы входных переменных, занимающих во входном сегменте непрерывный участок памяти, а также длину переменной или группы переменных. То же самое касается каждого выходного порта по отношению к выходному сегменту приложения. Образ процесса, о котором до сих говорилось в настоящем документе, буквально является парой буферов, размеры которых в точности совпадают с размерами входного и выходного сегментов приложения. Указанные буфера ассоциируются с каналами ввода и вывода при создании или повторном конфигурировании последних.

Процесс связывания 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 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 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, и синхронизировать доступ к входным и выходым данным приложения пользователя и других сервисов системы исполнения.

4.2.4.2. Исполнение циклических задач

Каждая циклическая задача исполняется на контексте отдельного потока операционной системы. Алгоритм функционирования циклической задачи представлен на рис. 14.

В процессе выполнения производятся следующие действия:

1. У операционной системы получается текущее время, которое требуется для точного вычисления времени выполнения кода всех программ, вызываемых из корневой программной единицы данной задачи, а также времени, затраченного на ввод-вывод данных.
2. Выполняется последовательное чтение всех входных портов задачи. Во время чтения портов канал ввода блокирован (объектом синхронизации типа "критическая секция") для записи со стороны связанных с ним выходных портов других участников обмена данными (сервисов внешней сети и ввода-вывода).
3. Выполняется вызов корневой программной единицы задачи.

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

   Если в пользовательском коде имеется деление на 0 целочисленных операндов, возникнет исключение EXCEPTION_INT_DIVIDE_BY_ZERO, и контроллер перейдет в безопасный режим.

   Если в пользовательском коде имеется деление на 0 операндов с плавающей точкой, возникнет исключение EXCEPTION_FLT_DIVIDE_BY_ZERO, и контроллер перейдет в безопасный режим.

   Если в пользовательском коде имеется ошибка кодогенератора CoDeSys, с высокой вероятностью возникнет исключение EXCEPTION_ILLEGAL_INSTRUCTION или EXCEPTION_INVALID_DISPOSITION, и контроллер перейдет в безопасный режим.
4. При успешном завершении работы корневой программной единицы задачи выполняется последовательная запись во все выходные порты задачи. Во время записи в порты канал

вывода блокирован (объектом синхронизации типа "критическая секция") для чтения со стороны связанных с ним входных портов других участников обмена данными.

5. Вычисляется суммарное время ввода-вывода и исполнения текущего цикла в микросекундах.
6. Вычисляется время в миллисекундах, в течение которого данная задача должна находиться в состоянии пассивного ожидания ("спать"), отдав процессор другим задачам. Если оказывается, что время исполнения и ввода-вывода на текущем цикле превысило период цикла, задача в обязательном порядке будет "спать" в течение 1 мс, чтобы не возникало перегрузки системы.
7. Увеличивается счетчик циклов данной циклической задачи. Если время исполнения и ввода-вывода на текущем цикле превысило заданный период цикла, увеличивается счетчик запаздываний данной задачи.
8. Задача переводится в состояние пассивного ожидания на время, вычисленное при выполнении шага 6.
9. Задача "просыпается" и переходит к шагу 1 данного алгоритма.

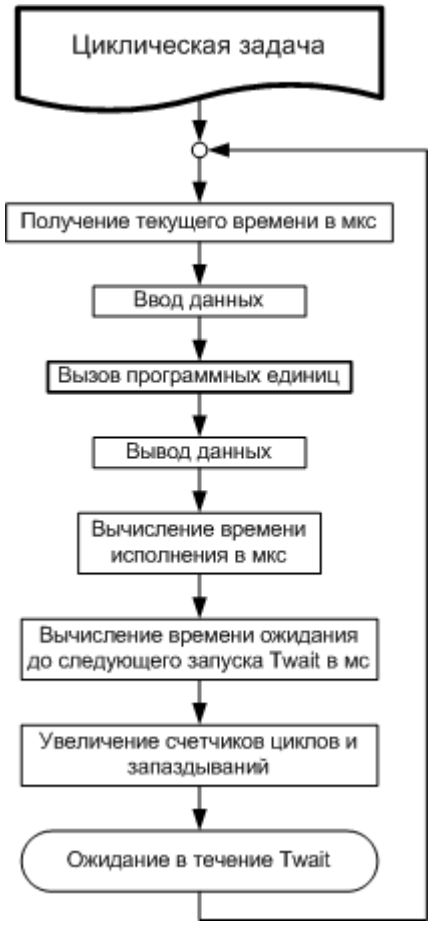


Рис. 14. Алгоритм функционирования циклической задачи

Если имеется несколько циклических задач с разными приоритетами, управление в первую очередь получает задача с наибольшим по значению приоритетом. Как только она переводится в состояние пассивного ожидания (шаг 8), управление получает задача с приоритетом ниже данной на 1 и т.д.

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

4.2.4.3. Исполнение ациклических задач

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

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

1. Вызывается функция-обработчик системного события *OnTimer*, если она была установлена пользователем в окне ресурса **Tasks Configuration–System events** приложения.
2. Если очередь ациклических задач пуста, просмотр очереди завершается и осуществляется переход к предпоследнему шагу данного алгоритма. Если очередь не пуста, из ее "головы" извлекается находящаяся там ациклическая задача.
3. Считывается значение переменной-события извлеченной задачи. Если переменная-событие находится в сегменте входных данных, ее значение считывается по соответствующему адресу из образа процесса. Если же переменная-событие расположена сегменте выходных данных или в сегменте внутренних переменных приложения, то значение переменной считывается непосредственно из того сегмента приложения. Считанное значение запоминается в объекте-задаче.
4. Если значение переменной-события, считанное на предыдущем цикле просмотра очереди ациклических задач, было равным FALSE (0), а на считанное на текущем цикле просмотра – TRUE (1), то считается, что произошло событие, по которому должна быть вызвана корневая программная единица данной задачи. В противном случае осуществляется переход к шагу 2 настоящего алгоритма.
5. Выполняется последовательное чтение всех входных портов задачи. Во время чтения портов канал ввода блокирован (объектом синхронизации типа "критическая секция") для записи со стороны связанных с ним выходных портов других участников обмена данными.
6. Выполняется вызов корневой программной единицы задачи.

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

   Если в пользовательском коде имеется деление на 0 целочисленных операндов, возникнет исключение EXCEPTION_INT_DIVIDE_BY_ZERO, и контроллер перейдет в безопасный режим.

   Если в пользовательском коде имеется деление на 0 операндов с плавающей точкой, возникнет исключение EXCEPTION_FLT_DIVIDE_BY_ZERO, и контроллер перейдет в безопасный режим.

   Если в пользовательском коде имеется ошибка кодогенератора CoDeSys, с высокой вероятностью возникнет исключение EXCEPTION_ILLEGAL_INSTRUCTION или EXCEPTION_INVALID_DISPOSITION, и контроллер перейдет в безопасный режим.
7. При успешном завершении работы корневой программной единицы задачи выполняется последовательная запись во все выходные порты задачи. Во время записи в порты канал вывода блокирован (объектом синхронизации типа "критическая секция") для чтения со стороны связанных с ним входных портов других участников обмена данными.
8. Выполняется сравнение содержимого сегмента энергонезависимых переменных приложения с содержимым этого же сегмента, запомненным в специальном буфере на предыдущем цикле сервисной задачи. Если найдено хотя бы одно отличие, выполняется

обновление буфера и сохранение всего сегмента в специальной энергонезависимой памяти типа FRAM. Данная операция включает в себя вычисление циклической контрольной суммы по всем байтам сегмента и сохранение двух копий сегмента в энергонезависимой памяти. Если пользователь использует в приложении все 2 кбайт энергонезависимых переменных, то время сохранения составляет от 40 до 60 мс. В течение этого времени циклические задачи приложения вытеснены сервисной задачей, и не исполняются.

9. Осуществляется переход к шагу 2 настоящего алгоритма
10. С периодом 500 мс выполняется диагностика исполнения циклических задач, в том числе:

    суммируются общие количества циклов и запаздываний всех циклических задач, полученные значения выводятся в диагностические каналы *MK905 Programmable Automation Controller – Application Diagnostics–CyclesCounter* и *MK905 Programmable Automation Controller–Application Diagnostics–OverrunsCounter*;

    формируется маска состояния циклических задач (0 – неактивна; 1 – активна и успевает; 2 – активна и иногда не успевает; 3 – активна и никогда не успевает) и выводится в диагностические каналы *MK905 Programmable Automation Controller–Application Diagnostics–Cyclic Tasks Status*;

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

    в зависимости от выявленного состояния циклических задач формируется индикация светодиода LED1.

11. Сервисная задача переводится в состояние пассивного ожидания на время, заданное параметром *MK905 Programmable Automation Controller:EventsRate*, отдавая процессор другим задачам среды исполнения.
12. По прошествии времени *MK905 Programmable Automation Controller:EventsRate* сервисная задача "просыпается" и переходит к шагу 1 настоящего алгоритма.

ВНИМАНИЕ!

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

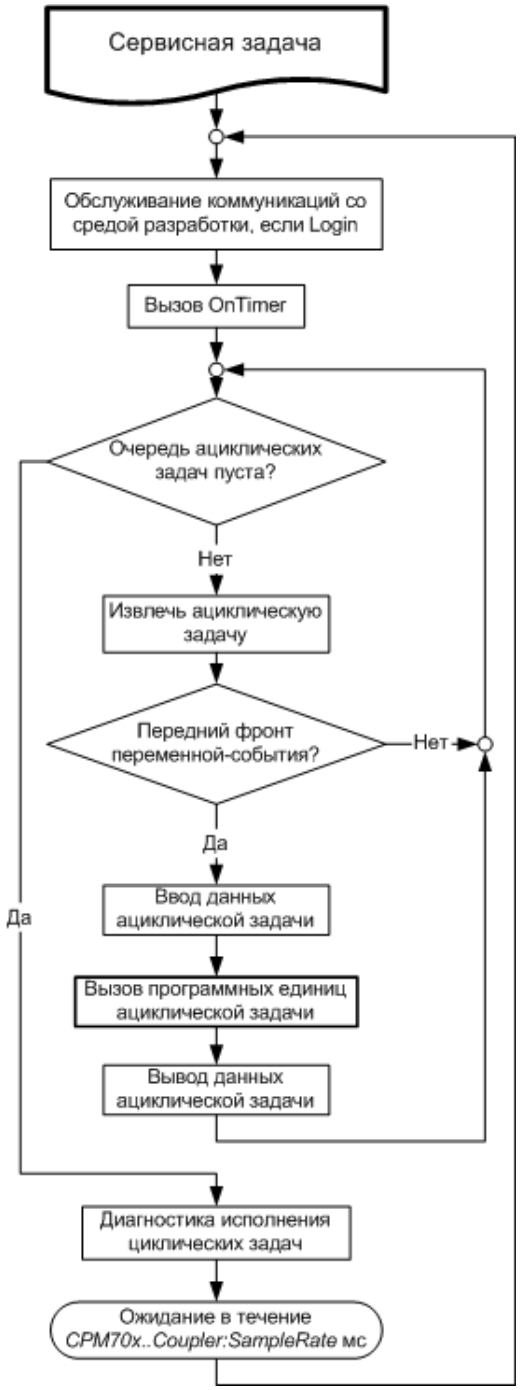


Рис. 15. Алгоритм функционирования сервисной задачи

4.2.4.4. Вызов обработчиков системных событий

Перечень системных событий, поддерживаемых адаптированной средой исполнения CoDeSys, приведен в табл. 2.

Таблица 2

Поддерживаемые системные события Tasks Configuration–System events		
Тип события	**Значение входного параметра**	**Вызов**
OnStart	**F_EVENT_START (1)**	▪ Перед первым циклом сервисной задачи после события OnInit; ▪ По команде **Online–Run**, выполняемой после команды **Online–Stop**
OnStop	**F_EVENT_STOP (2)**	По команде **Online–Stop**
BeforeReset	**F_EVENT_BEFORE_RESET (3)**	Непосредственно перед перезапуском контроллера: ▪ по командам **Online–Reset**; **Online–Reset (cold)**; **Online–Reset (original)**; ▪ по окончании загрузки новой версии системного ПО контроллера после события OnProgramChange
OnLogin	**F_EVENT_LOGIN (501)**	▪ по команде **Online–Login**; ▪ перед запуском вновь загруженного приложения перед событием OnInit
OnLogout	**F_EVENT_LOGOUT (1503)**	▪ по команде **Online–Logout**, выполняемой после команды **Online–Login**; ▪ перед началом процесса конфигурирования вновь загруженной программы, описанного в п. 4.2.3, до вызова обработчика OnProgramChange; ▪ при разрыве связи со средой разработки, находившейся в состоянии Login на данном контроллере.
OnProgramChange	**F_EVENT_ONLINE_CHANGE (33)**	▪ перед началом процесса конфигурирования вновь загруженной программы, описанного в п. 4.2.3 ▪ по окончании загрузки новой версии системного ПО контроллера перед перезапуском контроллера
OnDownload	**F_EVENT_BEFORE_DOWNLOAD (34)**	Перед началом загрузки из среды разработки секции кода нового приложения
OnInit	**F_EVENT_ON_INIT (1501)**	Перед запуском приложения. **Вызов функции F_IecTasks_linkVariables() из внешней библиотеки FastwelTasksExchange.lib допускается делать только при обработке данного события!**
OnPowerOn	**F_EVENT_POWER_ON (1502)**	Перед запуском приложения перед OnInit в случае, если контроллер стартует по включению питания
OnTimer	**F_EVENT_TIMER (30)**	Каждый цикл сервисной задачи с периодом МК905 *Programmable Automation Controller:EventsRate*

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

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

ВНИМАНИЕ!

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

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

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

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

Пример диспетчеризации системных событий:

```
FUNCTION SystemEventsDispatcher : DWORD
   VAR_INPUT
         eventType : DWORD;
   END_VAR
   CASE eventType OF
   F_EVENT_TIMER:
         ActualEventTimerHandler();
   F_EVENT_ONLINE_CHANGE:
```

```
            SaveMyPersistentVariables();
      F_EVENT_ON_INIT:
            LinkTasks();
            LoadMyPersistentVeariables();
      F_EVENT_POWER_ON:
            LoadMyRetainVariables();
      ELSE
            (* ничего не делаем *);
      END_CASE;
END_FUNCTION
```

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

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

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

Событие *OnProgramChange* может использоваться для сохранения значений некоторых переменных, которые должны использоваться вновь загруженной программой. Кроме того, настоятельно рекомендуется использовать событие *OnProgramChange* для закрытия всех открытых файлов и коммуникационных портов, если они были открыты старым приложением при помощи вызовов функций соответствующих внешних библиотек.

Событие *OnInit* может служить для реализации в пользовательском приложении однократных инициализирующих действия над каким-либо переменными приложения, для открытия файлов и т.п., а также для связывания задач приложения по переменным в соответствии с п. 4.2.4.5 перед запуском задач приложения. Если активизирована функция горячего обновления приложения (параметр *MK905 Programmable Automation Controller:HotUpdateDisabled* имеет значение *No*), необходимо соблюдать особую осторожность при открытии файлов (при помощи функции *FwSysFileOpen()* из библиотеки *FastwelSysLibFile.lib*) и коммуникационных портов (при помощи функции *FwSysComOpen()* из библиотеки *FastwelSysLibCom.lib*) – если в процессе обновления окажется, что структуры данных, размеры образа процесса и связи задач с образом процесса не изменились, то может произойти повторное открытие одних и тех же файлов и неудачное открытие ранее открытых портов. Поэтому наиболее правильным способом борьбы с такой ситуацией является закрытие всех файлов и порта во время обработки события *OnProgramChange*.

Пример:

```
(* диспетчер системных событий *)
FUNCTION SystemEventsDispatcher : DWORD
   VAR_INPUT
         eventType : DWORD;
   END_VAR
   CASE eventType OF
   F_EVENT_ONLINE_CHANGE:
         (* закрываем все открытые хэндлы *)
         CloseAllHandles();
   F_EVENT_ON_INIT:
         (* связываем задачи *)
         LinkTasks();
   F_EVENT_POWER_ON:
   ELSE
         (* ничего не делаем *);
   END_CASE;
END_FUNCTION
```

4.2.4.5. Обмен данными между задачами

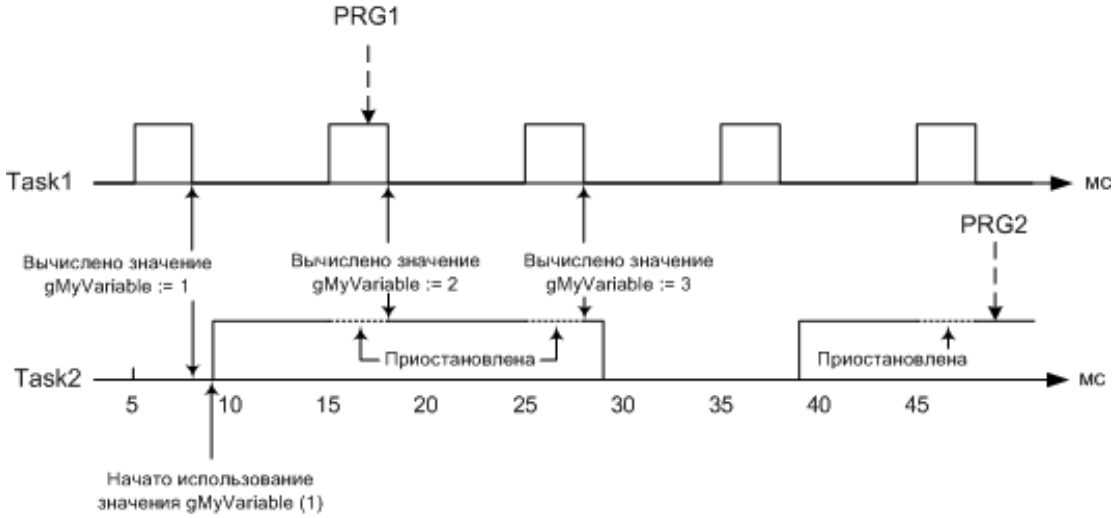
Традиционно при разработке приложений для программируемых логических контроллеров применяется подход, при котором все входные переменные (их значения получаются приложением от входных каналов модулей ввода-вывода и входящих коммуникационных объектов внешней сети) и

большинство внутренних переменных алгоритма, реализуемого приложением, делаются глобальными, т.е. доступными любой программной единице приложения.

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

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

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



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

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

Решение данной проблемы путем копирования gMyVariable в какую-нибудь внутреннюю или входную переменную PRG2 в начале каждого цикла Task2 является корректным только для переменных длиной не более разрядности процессора, которая в контроллере MK905 составляет 32 бита (4 байта). Переменные большей длины, включая LREAL, массивы, строки и структуры, длина которых превышает 4 байта, невозможно скопировать атомарно – т.е. исключив возможность вытеснения во время копирования другой, более приоритетной, задачей, которая изменит копируемое значение, в результате чего во внутреннюю переменную PRG2 будет скопирована часть старого значения и часть нового, перевычисленного "вклинившейся" более приоритетной задачей.

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

Для того, чтобы пользователь был способен избежать указанных явлений при обмене данными между программами, вызываемыми из разных задач, в комплект адаптации CoDeSys для Fastwel I/O и MK905 входит библиотека поддержки *FastwelTasksExchange.lib*.

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

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

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

Примечание. Организовывать обмен данными описанным способом между ациклическими задачами не требуется, поскольку они исполняются друг за другом под управлением одной и той же сервисной задачи (синхронно относительно друг друга). Указанный способ должен применяться для связи по переменным большого размера между циклических задачами, а также между циклическими и ациклическими задачами.

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

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

Входные параметры:

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

```
TYPE F_LINK_DESCRIPTOR :
   STRUCT
    (*     Указатель на переменную, которую требуется связать с другой переменной.
           Пример: descr.variableAddress := ADR(SomeProgram.SomeVariable); *)
     variableAddress : DWORD;
    (*     Размер переменной (в байтах)
           Пример: descr.variableSize := SIZEOF(SomeProgram.SomeVariable); *)
     variableSize : INT;
    (*     Индекс программной единицы (POU), содержащей описываемую переменную.
           Пример: descr.pouIndex := INDEXOF(SomeProgram);       *)
     pouIndex : INT;
   END_STRUCT
END_TYPE
```

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

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

Возвращаемый результат:

```
F_LINK_UNCERTAIN := 0
```

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

F_LINK_OK := 1

Связывание выполнено успешно.

F_LINK_INVALID_SOURCE := 2

Неправильный описатель переменной-источника данных. Ситуации:

pSourceDescriptor равен 0;

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

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

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

F_LINK_INVALID_DESTINATION := 3

Неправильный описатель переменной-получателя данных. Ситуации:

pDestinationDescriptor равен 0;

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

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

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

F_LINK_INVALID_SOURCE_LEN := 4

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

F_LINK_INVALID_DESTINATION_LEN := 5

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

F_LINK_SOURCE_DESTINATION_LEN_NOT_EQUAL := 6

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

F_LINK_ONE_TASK_CONNECTION_NOT_ALLOWED := 7

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

F_LINK_CONNECTION_ALLOWED_ON_INIT_INTARGET_ONLY := 8

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

F_LINK_NOT_ENOUGH_RESOURCES := 9

Не хватило системных ресурсов для связывания.

F_LINK_INPORT_EXIST_FOR_DESTINATION := 10

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

Пример:

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

FUNCTION LinkTasks: DWORD
(*********************************************************************
Функция, вызываемая из пользовательского диспетчера системных событий по событию
OnInit.
НИКОГДА не ставьте ее непосредственно в качестве обработчика OnInit в диалоге Tasks
Configuration-System events среды разработки CoDeSys, иначе контроллер гарантированно
упадет после вызова!
*********************************************************************)
VAR_INPUT
   dwEventType : DWORD;
END_VAR
VAR
   linkResult : F_LINK_RESULT;
   sourceDesc : F_LINK_DESCRIPTOR;
   destDesc : F_LINK_DESCRIPTOR;
END_VAR
(* Тело функции *)
IF dwEventType = F_EVENT_ON_INIT THEN
(* Устанавливаем связь по переменным, хранящим уставки температуры резисторов,
между POU "PLC_PRG" и "RESISTORS_CONTROL" *)
(* Описатель источника данных *)
(* Индекс POU PLC_PRG*)
   sourceDesc.pouIndex := INDEXOF(PLC_PRG);
(* Адрес переменной-источника данных *)
   sourceDesc.variableAddress := ADR(PLC_PRG.arResistorsValue[2]);
(* Длина переменной-источника данных, два элемента массива (2-й и 3-й) *)
```

```
   sourceDesc.variableSize := 2 * SIZEOF(PLC_PRG.arResistorsValue[2]);
(* Описатель получателя данных *)
(* Индекс POU RESISTORS_CONTROL *)
   destDesc.pouIndex := INDEXOF(RESISTORS_CONTROL);
(* Адрес переменной-получателя данных *)
   destDesc.variableAddress := ADR(RESISTORS_CONTROL.arResistorsValue[2]);
(* Длина переменной-получателя данных, два элемента массива (2-й и 3-й) *)
   destDesc.variableSize := 2 * SIZEOF(PLC_PRG.arResistorsValue[2]);
   linkResult := F_IecTasks_linkVariables(ADR(sourceDesc), ADR(destDesc));

   globalLinkResult := linkResult;

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

4.2.5. Диагностика

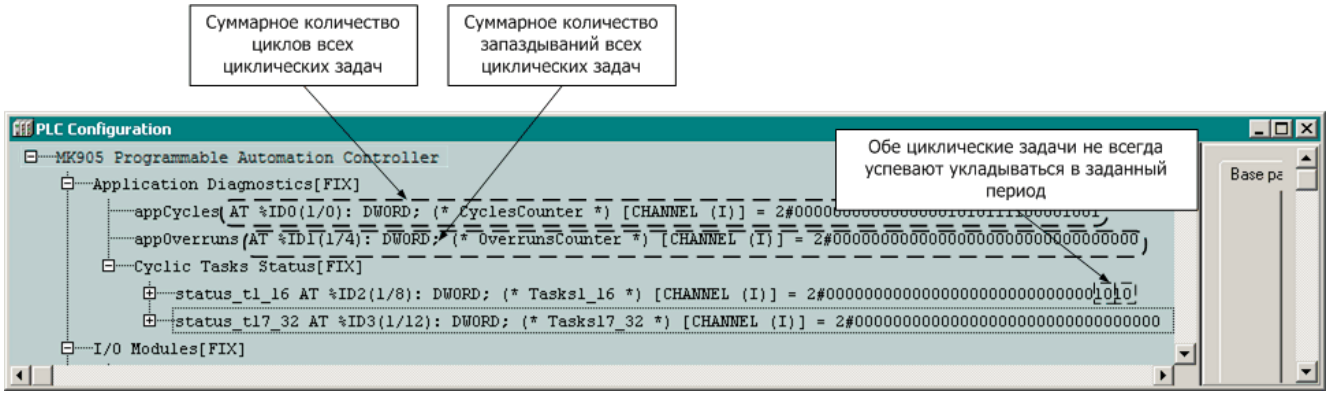


Рис. 16. Диагностические каналы среды исполнения CoDeSys

В конфигурации контроллера имеется секция *Application Diagnostics*, в которой определены два входных канала, позволяющих приложению во время выполнения получить общее количество циклов всех циклических задач и общее количество циклов, во время которых циклические задачи не успели завершить исполнение в течение заданных периодов. Назначение каналов представлено в табл. 3.

Таблица 3

Описание секции Application Diagnostics конфигурации контроллера MK905			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
CyclesCounter	%ID0	DWORD	Общее количество циклов всех циклических задач
OverrunsCounter	%ID1	DWORD	Общее количество циклов циклических задач, во время которых они не успели завершить выполнение в течение своих заданных периодов
Описание секции Application Diagnostics–Cyclic Tasks Status			
Tasks1_16	%ID2	DWORD	Статус циклических задач с 1-й по 16-ю
Tasks17_32	%ID3	DWORD	Статус циклических задач с 16-й по 32-ю

Кроме того, секция *Application Diagnostics–Cyclic Tasks Status* содержит два канала, пары бит которых представляют текущий статус циклических задач в порядке их следования в списке ресурса **Tasks Configuration**: младшие два бита соответствуют задаче с наименьшим номером. Статус задачи может принимать следующие значения:

0: неактивная задача;

1: задача активна и успевает укладываться в заданный период;

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

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

"Успевать укладываться в заданный период" для задачи означает, что за интервал анализа (500 мс) время исполнения каждого цикла задачи ни разу не превысило периода цикла, и задача выполнила N-1 циклов, где N = 500 / Период_цикла.

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

4.3. Принцип работы сервиса ввода-вывода

4.3.1. Общие сведения

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

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

Схема подключения модулей ввода-вывода Fastwel I/O к MK905 представлена на рис. 17.

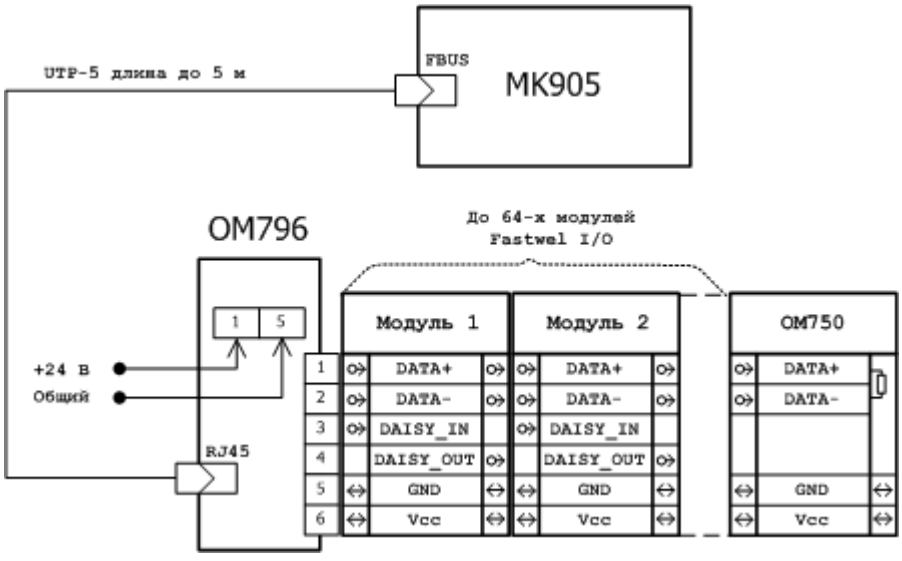


Рис. 17. Линии среды обмена данными FBUS и схема подключения модулей ввода-вывода к MK905

Таблица 4

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

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

ВНИМАНИЕ!

Для организации обмена данными между приложением и модулями ввода-вывода требуется добавить описания модулей ввода-вывода в секцию *MK905 Programmable Automation Controller–I/O Modules* ресурса **PLC Configuration** с точным соблюдением порядка физического подключения к внутренней шине контроллера. Например, если к контроллеру подключены модули в следующем порядке: AIM720 (самый ближний к контроллеру); DIM713; DIM717; DIM717; DIM718, то конфигурация сервиса ввода-вывода должна выглядеть, как показано на рис. 18.

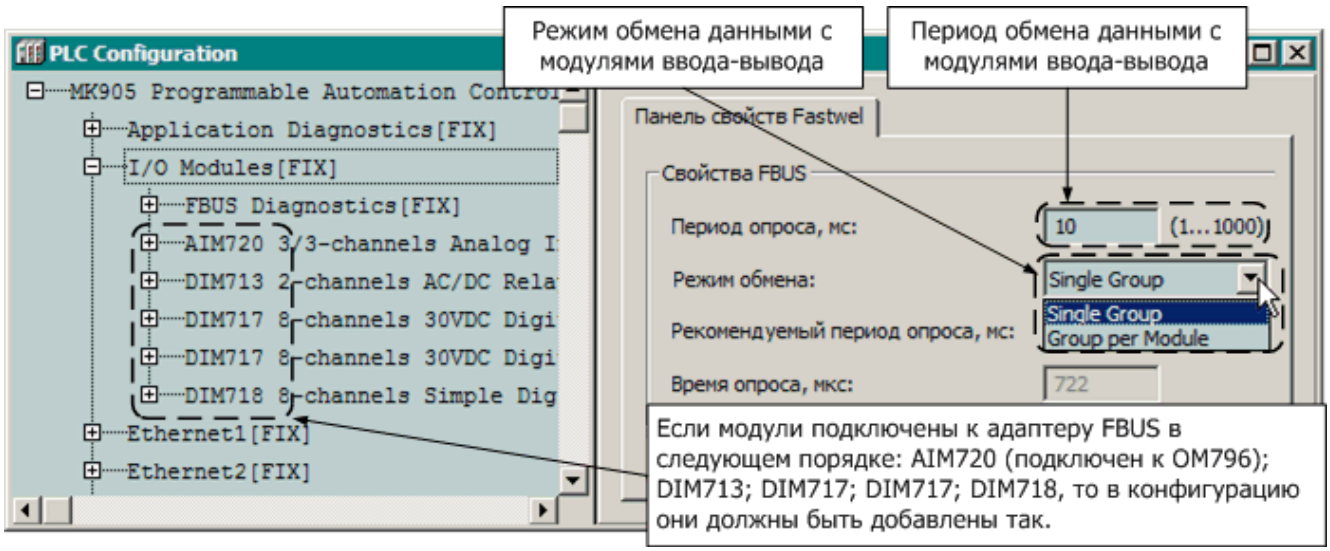


Рис. 18. Конфигурация сервиса ввода-вывода контроллера

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

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

Период обмена с модулями ввода-вывода определяемую параметром *MK905 Programmable Automation Controller –I/O Modules:SampleRate*. При использовании режима индивидуального обмена пропускная способность внутренней шины существенно ухудшается за счет появления пауз длительностью 500 мкс между обменами с соседними модулями, а также за счет передачи в групповом

ответе каждого модуля дополнительных пяти байт (идентификатора мастера и поля контрольной суммы).

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

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

4.3.2. Инициализация шины

Инициализация шины выполняется сервисом ввода-вывода в несколько этапов:

Этап 1:

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

Этап 2:

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

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

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

Если сервис ввода-вывода настроен на работу в режиме индивидуального обмена с модулями, то в случае, если при инициализации обнаружена хотя бы какая-то часть модулей, добавленных пользователем в конфигурацию контроллера (в точном соответствии с порядком следования в списке *MK905 Programmable Automation Controller–I/O Modules*), то сервис ввода-вывода будет выполнять обмен данными только с ними, а поиск остальных выполняться не будет.

4.3.3. Обмен данными с модулями ввода-вывода

4.3.3.1. Групповой режим (*Single Group*)

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

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

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

Осциллограмма одной групповой операции обмена данными с двумя модулями ввода-вывода показана на рис. 19.

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

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

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

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

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

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

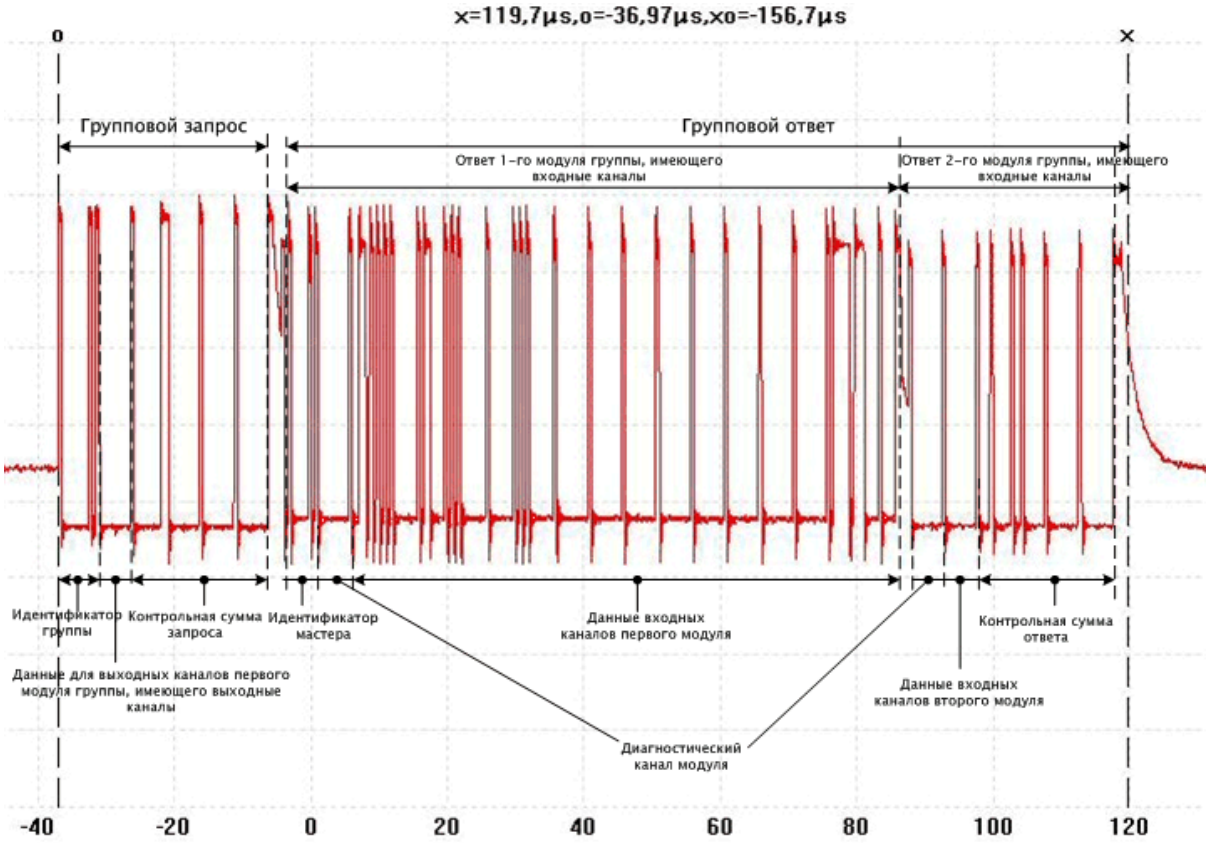


Рис. 19. Осциллограмма группового обмена с модулями AIM720 и DIM713

Сервис ввода-вывода, получив ответ первого модуля (первой группы), проверяет контрольную сумму и, в случае ее правильности, сбрасывает счетчик ошибок данной группы, записывает данные в соответствующий участок входной части образа процесса.

Далее таким же образом выполняется обмен с остальными модулями.

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

Осциллограмма обмена данными с двумя модулями ввода-вывода в режиме индивидуального обмена показана на рис. 20.

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

При использовании режима *Group per Module* сервис ввода-вывода потребляет существенно больше вычислительных ресурсов, чем в режиме *Single Group*. В результате производительность при исполнении программных единиц, вызываемых на контексте циклических задач, может ухудшиться 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.

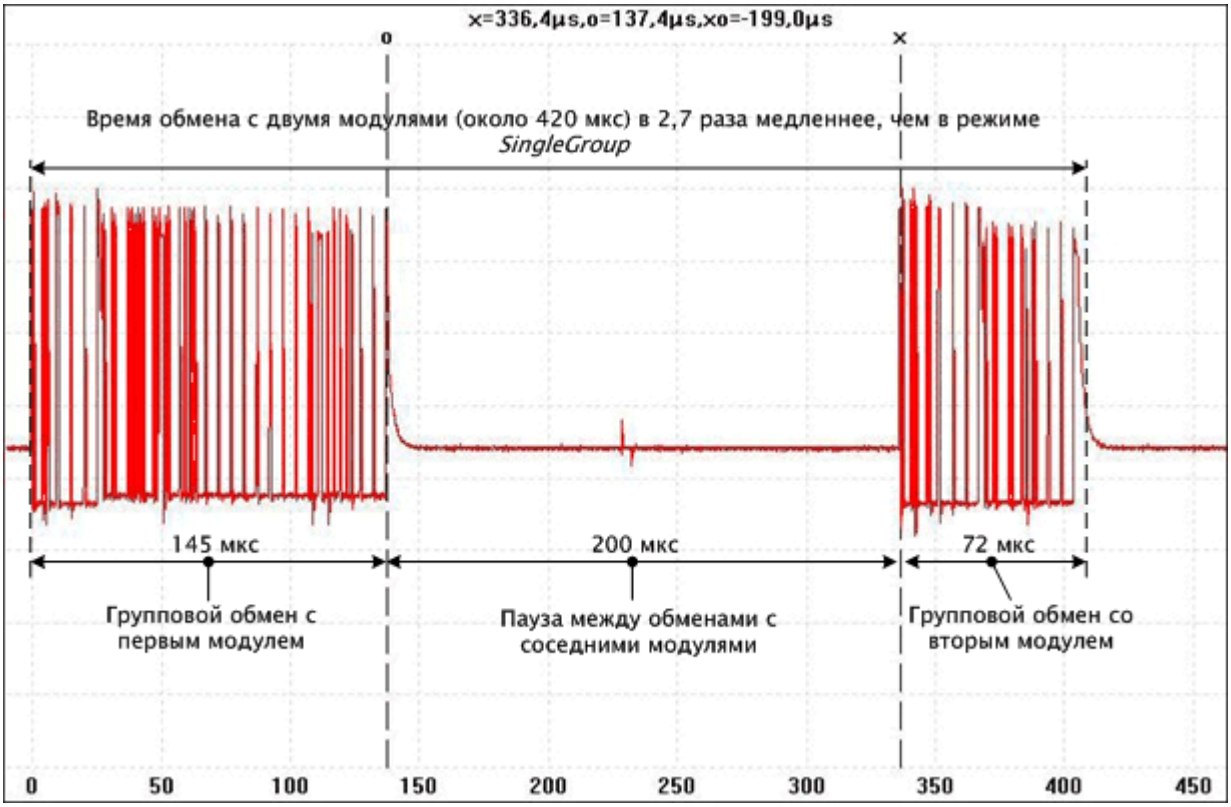


Рис. 20. Осциллограмма обмена с модулями AIM720 и DIM713 в режиме *Group per Module*

ВНИМАНИЕ!

При конфигурировании контроллера МК905 **не рекомендуется** устанавливать значение параметра *I/O Modules:SampleRate* меньшим 5 мс, т.к. это отрицательно сказывается на общей производительности системы.

4.3.4. Обработка нештатных ситуаций

4.3.4.1. Ошибка инициализации при запуске контроллера

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

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

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

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

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

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

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

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

4.3.5. Диагностика

4.3.5.1. Индикация

Индикация функционирования сервиса ввода-вывода осуществляется при помощи светодиодного индикатора LED2:

Непрерывное свечение – 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**–*MK905 Programmable Automation Controller–I/O Modules*).

Прерывистое свечение – обмен с модулями ввода-вывода, подключенными к контроллеру, не может быть выполнен за время, заданное в конфигурации контроллера (**Resources–PLC Configuration**– *MK905 Programmable Automation Controller–I/O Modules:SampleRate*), в связи с чем используется расчетное минимально достижимое время обмена данными со всеми модулями при загрузке внутренней шины около 50%.

Отсутствие свечения – в конфигурации загруженного приложения отсутствуют описания модулей ввода-вывода или конфигурация модулей ввода-вывода, определенная в проекте, не совпадает с аппаратной конфигурацией модулей, подключенных к контроллеру.

4.3.5.2. Диагностические каналы сервиса ввода-вывода

Диагностические канала сервиса ввода-вывода предназначены для реализации контроля состояния внутренней шины в прикладной программе. Описания каналов приведены в табл. 5.

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

Таблица 5

Описание области I/O Modules–FBUS Diagnostics конфигурации MK905			
Элемент/канал	**Адрес**	**Тип**	**Назначение**
IOStatus0	%IB11	DWORD	Битовая маска наличия модулей ввода-вывода с 1-го по 32-й, соответствующих перечисленным в конфигурации контроллера
IOStatus1	%IB15	DWORD	Битовая маска наличия модулей ввода-вывода с 33-го по 64-й, соответствующих перечисленным в конфигурации контроллера
TransactionsCount	%IB19	DWORD	Общее количество обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера
ErrorsCount	%IB23	DWORD	Количество неудачных обменов данными/командами с модулями ввода-вывода по внутренней шине контроллера

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

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

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

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

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

5. Указания по разработке приложений

5.1. Общие сведения

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

1. Создание проекта для платформы *Fastwel MK905 Programmable Controller*
2. Создание программ и конфигурации задач
3. Создание обработчиков системных событий
4. Трансляция приложения
5. Загрузка приложения в контроллер
6. Отладка приложения
7. Мониторинг переменных
8. Трассировка переменных
9. Обновление приложения в контроллере
10. Запись файлов в контроллер
11. Чтение файлов из контроллера
12. Обновление системного программного обеспечения контроллера

5.2. Создание проекта

Для создания проекта:

1. Запустите среду разработки CoDeSys и выберите команду **File–New**
2. В появившейся диалоговой панели **Target Settings** установите опцию **Configuration :** *Fastwel MK905 Programmable Controller* и нажмите кнопку **OK**
3. В появившейся диалоговой панели **New POU** будет предложено создать программу с именем *PLC_PRG*. Нажмите кнопку **OK**, если намереваетесь иметь в проекте программу с таким именем. В противном случае измените имя создаваемой программы либо вообще откажитесь от создания программы на данном этапе проектирования.

   Если в проекте имеется программа с именем *PLC_PRG*, содержащая в своем теле хотя бы один пустой оператор (**;**) или выражение, а в ресурсе **Tasks Configuration** не создано ни одного описания циклической или ациклической задачи, то при трансляции проекта командой **Project–Build All** будет создано приложение с одной, скрытой от пользователя, циклической задачей с именем *DefaultTask*, период которой будет равен значению параметра *MK905 Programmable Automation Controller:EventsRate*. Из корневой программной единицы данной задачи будет вызываться программа *PLC_PRG*.

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

   Если программа *PLC_PRG* (или любая программная единица с любым именем) создана, однако на данном этапе предполагается транслировать проект без написания кода данной программной единицы, в ее теле введите единственный оператор: **;**
4. Выберите **File–Save** и сохраните создаваемый проект в файле. При необходимости можно ввести информацию о проекте, включая заголовок, имя автора и т.п., выбрав команду **Project–Project Info**.
5. Для пробной трансляции проекта выберите команду **Project–Rebuild All**. Успешная трансляция будет завершена без диагностических сообщений красного цвета в панели вывода среды CoDeSys. Информация о сообщениях, выводимых средой разработки

CoDeSys в панели вывода, может быть получена во встроенной справочной системе или документации на среду разработки CoDeSys.

5.3. Создание и редактирование конфигурации контроллера

Основными элементами конфигурации контроллера являются описания физических устройств и/или их подсистем, параметры физических устройств/подсистем и каналы ввода-вывода, как показано на рис. 21. Каналы ввода-вывода элементов конфигурации определяют структуру образа процесса, который используется для взаимодействия приложения CoDeSys с окружением.

Основной элемент конфигурации *MK905 Programmable Automation Controller* имеет параметр *EventsRate*, который определяет:

1. При отсутствии в ресурсе **Tasks Configuration** задач, добавленных пользователем, – период цикла "автоматической" циклической задачи с именем *DefaultTask*, на контексте которой исполняется программа *PLC_PRG*
2. Период цикла сервисной задачи адаптированной среды исполнения CoDeSys в диапазоне от 1 до 100 мс (по умолчанию 10 мс). Более подробная информация сервисной задаче приведена в п. 4.2.4.1 настоящего руководства.

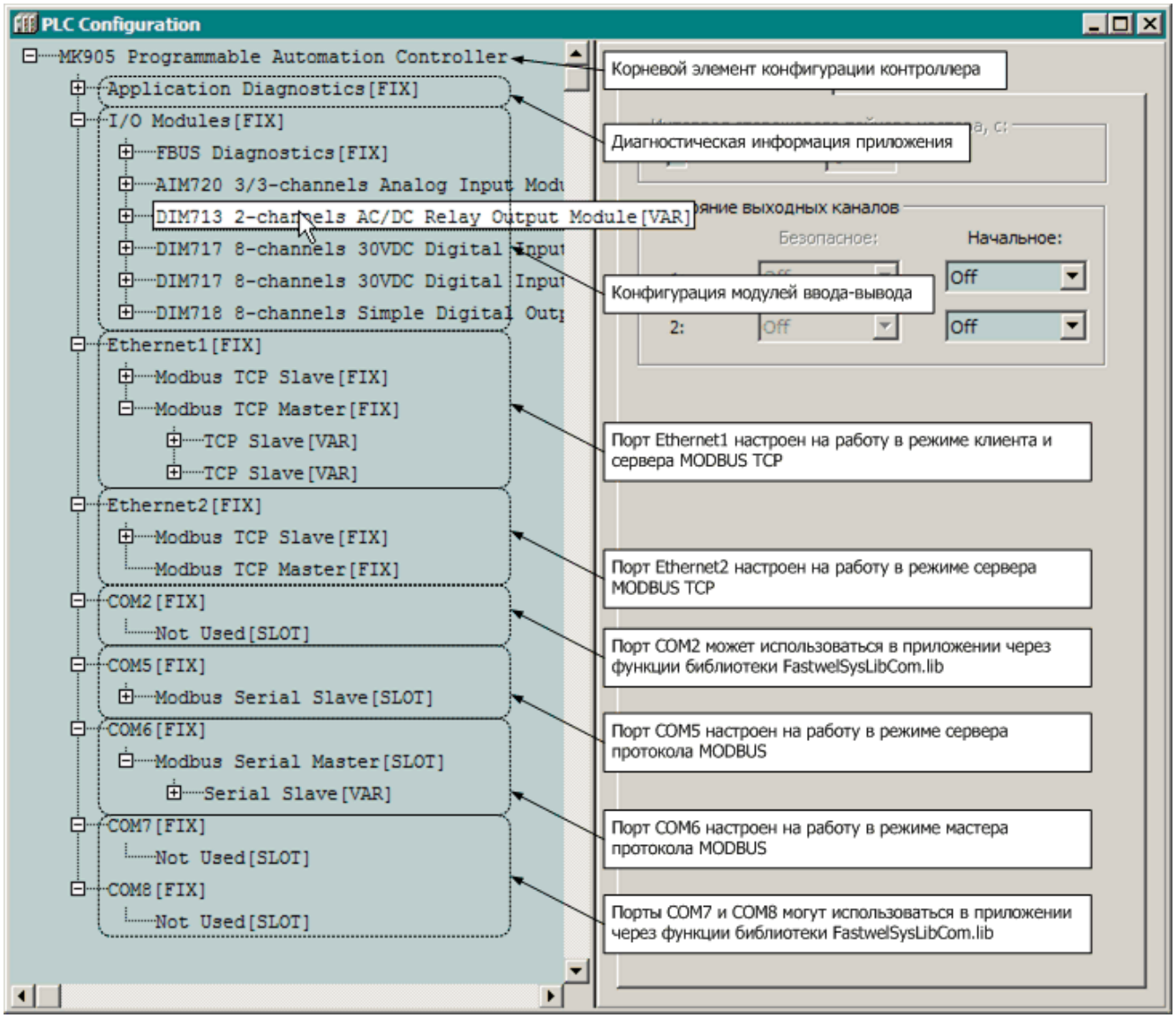


Рис. 21. Элементы конфигурации контроллера

Параметр *Autostart*, имеющий исходное значение *Yes*, позволяет после загрузки приложения оставить контроллер в состоянии останова. Для последующего запуска контроллера следует выполнить команду **Online–Login**, а затем **Online–Run**, либо нажать кнопку **Start** в окне консоли системы исполнения.

Параметр *HotUpdateDisabled*, имеющий исходное значение *Yes*, предназначен для блокировки или активизации функции горячего обновления приложения в соответствии с п. 4.2.3.

Элемент *MK905 Programmable Automation Controller–I/O Modules* является списком, который предназначен для добавления в конфигурацию описаний модулей ввода-вывода. Модули добавляются по командам контекстного меню, вызываемым как над самим элементом *I/O Modules* (**Append Subelement**), так и над описаниями самих модулей (**Insert Subelement**).

5.4. Создание программных единиц и задач

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

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

Максимальное количество программных единиц, поддерживаемых адаптированной средой исполнения CoDeSys для MK905, составляет 16384.

Максимальный размер секции кода, генерируемого CoDeSys, для платформы MK905 составляет 2 Мбайт, из которых от 1 до 5% приходится на служебную информацию. Обратите внимание, что среда разработки CoDeSys генерирует исполняемый код для форм визуализации, добавленных в проект.

Если после трансляции программы оказалось, что максимальный размер секции кода превышен:

1. Выберите команду меню **Project–Options**, щелкните на элементе *Build* в списке диалоговой панели **Options**, после чего снимите флажок с опции **Debugging**. В результате после трансляции проекта пошаговая отладка станет недоступной, однако размер сгенерированного кода уменьшится.
2. Главным резервом для оптимизации размера приложения являются операции с типом BOOL. Особенно это касается доступа к битовым полям в образе процесса и статической инициализации больших массивов элементов типа BOOL. По возможности для доступа к большому количеству смежных битовых полей старайтесь использовать операции сдвига и маскирования битов в целочисленных переменных.

Адаптированной среда исполнения CoDeSys для MK905 поддерживает до 32-х циклических и до 64-ти ациклических задач.

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

При создании ациклических задач следует помнить, что длительные операции и циклы под их управлением недопустимы, поскольку имеющимся в приложении циклическим задачам и сетевым сервисам может не хватить процессорного времени.

5.5. Связывание программ с окружением и ввод-вывод данных

5.5.1. Общие сведения

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

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

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

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

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

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

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

В конструкциях *VAR, 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. 4.2.4.1), а отображение массивов типа BOOL не поддерживается вовсе.

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%IB11    : FIODiagnostics;
END_VAR
```

Пусть в конфигурацию контроллера добавлены 6 модулей аналогового ввода типа AIM726 и 9 модулей аналогового ввода типа AIM728, причем однотипные модули располагаются в конфигурации друг за другом. Также пусть требуется выводить в сеть MODBUS значения напряжения на каналах модулей AIM726 и AIM729. Суммарное количество каналов составляет 2 * 6 + 4 * 9 = 48, а значит в конфигурации сети контроллера для передачи 48-ми значений типа REAL должно быть создано не менее 96-ти входных регистров со смежными идентификаторами (адресами). Пусть первый из 96-ти созданных регистров имеет адрес %QB7 в области выходных данных среды исполнения.

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

```
PROGRAM PLC_PRG
   VAR CONSTANT
         AIM726_ARRAY_SIZE :INT := 5;
```

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

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

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

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

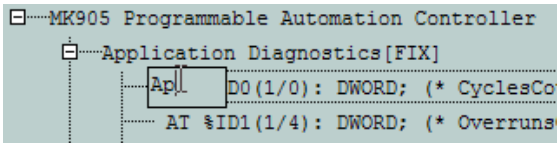
Основным недостатком данного способа является необходимость коррекции ссылок AT% в декларациях переменных при изменении структуры образа процесса, например, из-за вставки или удаления модулей ввода-вывода или коммуникационных объектов.

5.5.3. Создание символических имен каналов в ресурсе PLC Configuration

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

Для создания символического имени следует:

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



3. Нажать клавишу Enter.

5.5.4. Использование ресурса VAR_CONFIG

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

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

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

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

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

5.6. Создание обработчиков системных событий

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

Пример диспетчеризации системных событий:

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

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

Для добавления обработчика системного события:

1. Откройте окно ресурса **Tasks Configuration** и щелкните на элементе древовидного списка *System events*. В правой панели окна появится вкладка **System events**, показанная на рис. 22.
2. Дважды щелкните в ячейке таблицы *called POU* напротив названия системного события, для которого необходимо установить функцию-обработчик, и введите имя существующей функции (осторожно! у функции должен быть один входной параметр типа DWORD и возвращаемый результат типа DWORD и никаких внутренних переменных!) либо функции, которую собираетесь добавить сейчас же, после чего щелкните мышью на ячейке, расположенной слева от имени создаваемой функции, как показано на рис. 22.
3. Если для обработки события создается новая функция, станет доступной для нажатия кнопка **Create POU <*имя функции*>**. Нажмите ее, и в список **POUs** главного окна среды разработки будет добавлена функция с введенным именем.

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

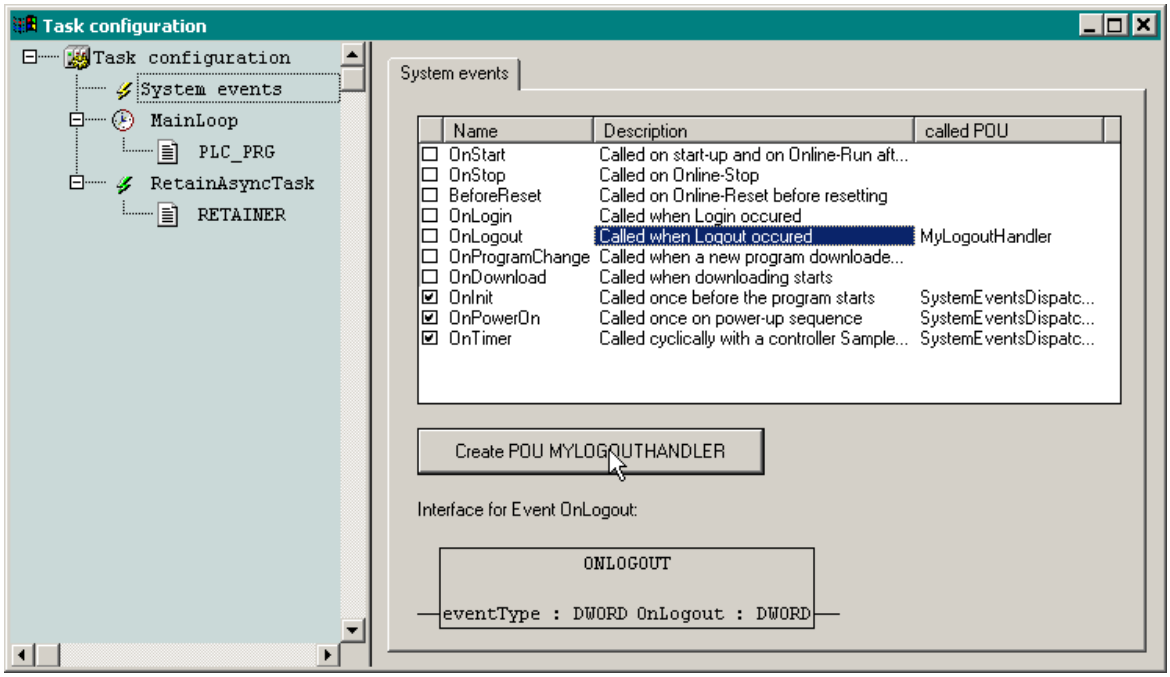


Рис. 22. Добавление обработчика системного события

5.7. Трансляция приложения

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

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

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

соответствующей области образа процесса. В таком случае после загрузки программы контроллер переходит в безопасный режим по ошибке связывания (*CoDeSys2. FAILED_TO_LINK_TASK*).

5.8. Загрузка приложения в контроллер и отладка

5.8.1. Общие сведения

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

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

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

Перед началом выполнения любых операций с удаленным контроллером должна быть выполнена настройка параметров драйвера коммуникационного сервера CoDeSys Gateway Server. Настройка выполняется в соответствии с указаниями п. 7.7.

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

5.8.2. Login

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

При успешном выполнении **Login** строка состояния главного окна среды CoDeSys принимает вид, аналогичный приведенному на рис. 23.

Рис. 23. 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. 24.

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*.

5.8.3. Загрузка приложения в контроллер

Для загрузки в контроллер:

1. Выберите команду **Online–Login**. При успешном соединении на экран будет выведена диалоговая панель, показанная на рис. 24.

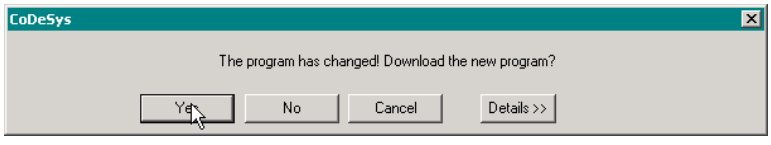


Рис. 24. Предложение загрузить программу

2. Нажмите кнопку **Yes**. На экран будет выведено окно, отображающее ход загрузки программы, показанное на рис. 25. Обратите внимание, что в данном окне отображается ход загрузки только исполняемого кода приложения. 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 коммуникационных объектов сетевых протоколов.

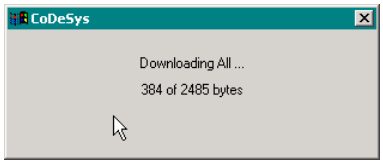


Рис. 25. Отображение хода загрузки программы

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

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

5.8.4. Просмотр и установка значений переменных

Предполагается, что приложение загружено в контроллер, а в среде CoDeSys открыт проект, в котором разрабатывалось данное приложение.

Для просмотра значений переменных приложения, исполняющегося в контроллере, выберите команду **Online–Login**. После успешного соединения среды CoDeSys с контроллером текущее активное окно редактора программ примет вид, показанный на рис. 26.

Для подготовки к записи нового значения в переменную дважды щелкните над переменной в верхней или правой области просмотра переменных и в появившейся диалоговой панели введите новое значение, как показано на рис. 27, и нажмите кнопку **OK**. Новое значение появится справа от текущего в треугольных скобках, как показано на рис. 28.

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

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

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

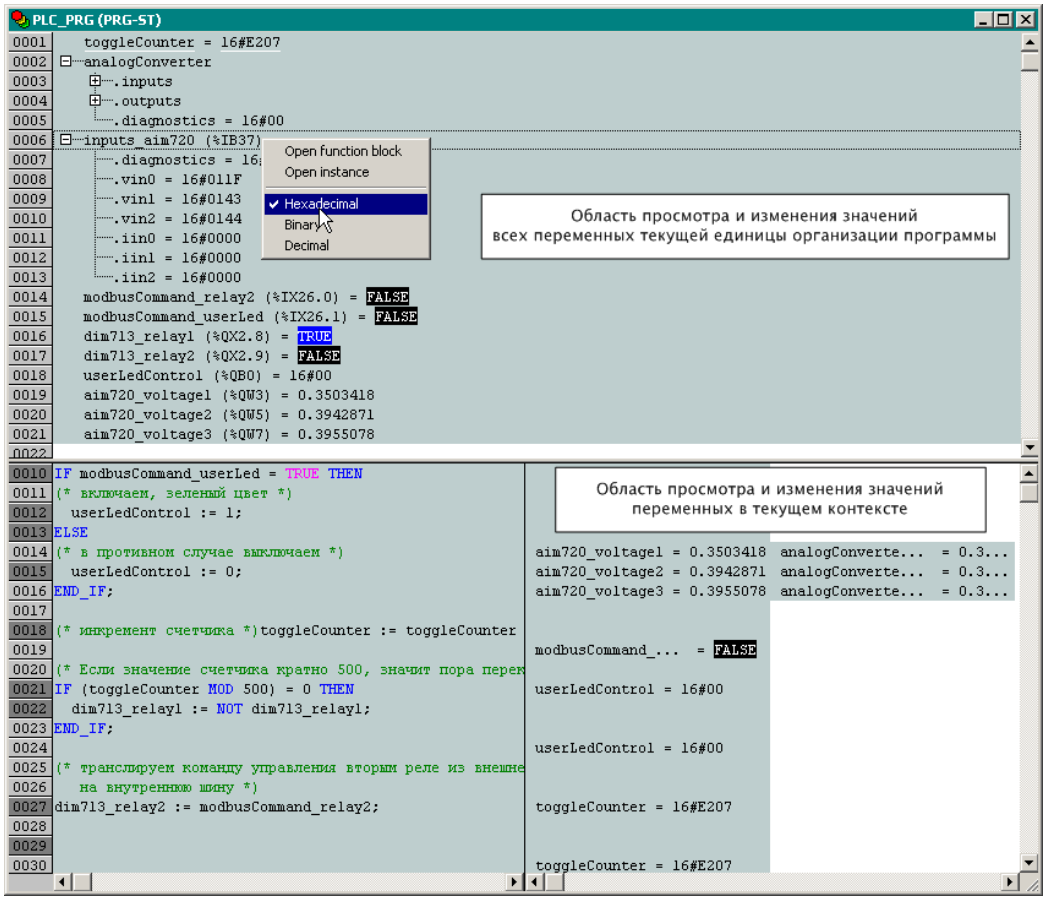


Рис. 26. Просмотр переменных

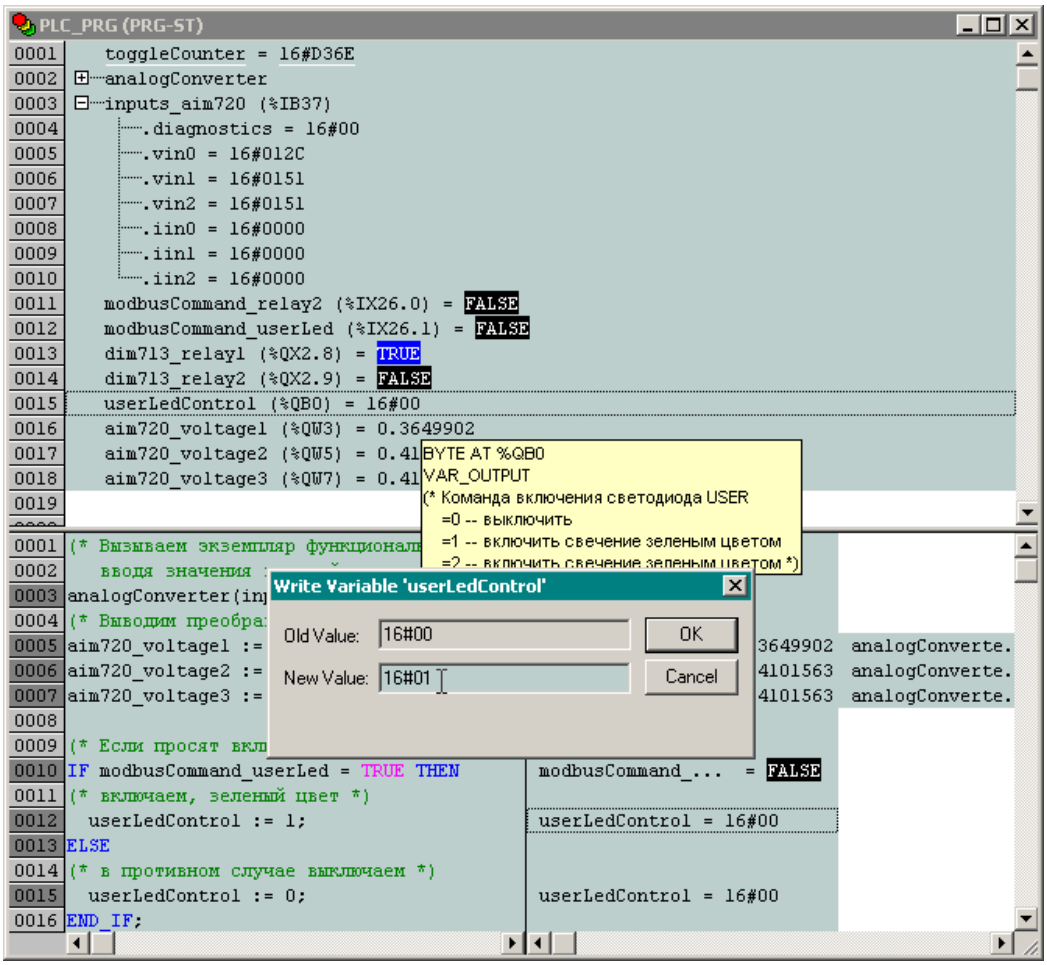


Рис. 27. Подготовка к изменению значения переменной

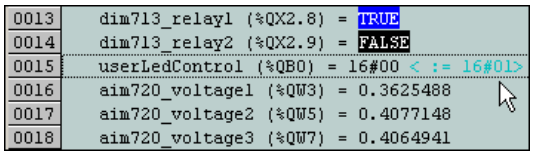


Рис. 28. Подготовленное значение переменной

5.9. Запись файлов в контроллер

Запись файлов в целевую платформу может быть необходима для загрузки энергонезависимых настроечных данных пользовательского приложения, для обновления системного программного обеспечения контроллера или для загрузки DLL расширения системы исполения.

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

Если выполняется обновление системного программного обеспечения или загрузка DLL расширения (загрузка файла *norm.dnl*), то по окончании загрузки произойдет автоматический перезапуск системы исполнения.

5.10. Чтение файлов из контроллера

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

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

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

При взаимодействии среды разработки CoDeSys с контроллерами Fastwel I/O на экране могут появляться сообщения с числовыми кодами ошибок последней операции взаимодействия. Перечень кодов представлен в табл. 6. К сожалению, среда разработки CoDeSys никак не реагирует на эти сообщения и не позволяет задать для них строки, удобные для восприятия.

Таблица 6

Код	Описание
50	Сервис не поддерживается данной платформой
104	При трассировке переменных запрошен слишком большой размер трассы
20001	Неправильная длина запроса на установление отладочной задачи по команде **Extras–Set Debug Task**
20002	В запросе на установление отладочной задачи по команде **Extras–Set Debug Task** поступил номер отсутствующей задачи.
20003	При включении **Online–Display Flow Control** оказалось, что текущая просматриваемая программная единица находится не в отладочной задаче. Для просмотра потоков данных необходимо сначала установить в качестве отладочной задачу, которая содержит требуемую программную единицу, а затем включить опцию **Online–Display Flow Control**.
20004	При включении **Online–Display Flow Control** оказалось, что текущая просматриваемая программная единица вызывается из нескольких задач. В таком случае просмотр потоков данных в текущей программной единице невозможен.
20005	Среда разработки при включении **Online–Display Flow Control** прислала позицию просмотра, которая не найдена в коде исполняющегося приложения или не содержит код команды временной точки останова.
20006	Среда разработки при включении **Online–Display Flow Control** прислала слишком много позиций для просмотра, которые не могут быть обработаны средой исполнения
20007	Просмотр стека вызовов по команде **Online–Show Call Stack** возможен только когда контроллер остановлен на установленной пользователем точке останова
20008	При включении **Online–Display Flow Control** оказалось, что текущая просматриваемая программная единица вызывается из ациклической задачи.
20009	Попытка записи или чтения файла с нулевой длиной имени
20010	Попытка чтения отсутствующего файла
20011	Возникла ошибка чтения файла
20012	Ошибка чтения информации о файле
20013	резерв
20014	Ошибка загрузки секции кода приложения в контроллер по команде **Online–Login**
20015	Ошибка загрузки секции конфигурации приложения
20016	Ошибка загрузки секции конфигурации задач
20017	Ошибка загрузки нового приложения
20018	Ошибка загрузки секции информации о проекте
20019	Запись системного файла запрещена
20020	Попытка перезаписи файла, открытого приложением или системным программным обеспечение контроллера
20021	При попытке записи по команде **Online–Write file to PLC** не удалось создать файл с заданным именем
20022	Возникла ошибка записи в файл
20023	Ошибка чтения информации о загруженном проекте
20024	Запрошенный список переменных не укладывается в буфер
20025	Не удалось начать загрузку нового приложения
20026	Попытка начать загрузку нового приложения во время другой незаконченной сессии загрузки
20027	Чтение системного файла запрещено
20028	резерв
20029	Команда **Online–Display Flow Control** запрещена, если в коде имеется хотя бы одна точка останова
20030	Установка точек останова запрещена во время выполнения **Online–Display Flow Control**

5.12. Трассировка переменных

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

6. Системные библиотеки

6.1. Общие сведения

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

Подсистема исполнения приложений CoDeSys контроллера МК905 поддерживает следующие системные библиотеки:

1. Standard.lib – стандартная библиотека функций базовых преобразований и блоков.
2. FastwelSysLibFile.lib – библиотека доступа к файловой системе.
3. FastwelTasksExchange.lib – библиотека для организации межзадачного обмена.
4. FastwelSysLibCom.lib – библиотека доступа к портам COM2–COM6.
5. FastwelSysLibSockets.lib – поддерживает работу с сокетами для коммуникаций по TCP/IP и UDP.
6. SysLibRtc.lib – библиотека доступа к часам-календарю с автономным питанием, входящим в состав МК905. Информация по использованию функций данной библиотеки приведена в документации и справочной системе на системные библиотеки CoDeSys. В текущей версии среды исполнения не реализованы и всегда возвращают 1 функция *SysRtcCheckBattery* и *SysRtcGetHourMode*.
7. FastwelUtils.lib – библиотека вспомогательных функций.

6.2. Библиотека FastwelSysLibFile.lib

6.2.1. Общие сведения

6.2.1.1. Особенности библиотеки FastwelSysLibFile.lib

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

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

6.2.1.2. Относительный путь

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

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

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

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

6.2.1.3. Абсолютный путь

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

```
\USB Storage\Temp1\my_file.txt
```

указывает на файл *my_file.txt* в каталоге *Temp1* съемного дискового накопителя, подключенного к интерфейсу USB контроллера первым.

6.2.1.4. Доступ к съемным USB-накопителям

К интерфейсу USB контроллера может быть подключено до четырех съемных накопителей, отформатированных с использованием файловой системы FAT или FAT32.

Съемные USB-накопители монтируются к корневому каталогу \ файловой системы контроллера в виде подкаталогов с именами *USB Storage* и *USB Storage2*.

Если к контроллеру подключено два съемных USB-накопителя, то накопитель, подключенный первым, будет подмонтирован к корневому каталогу файловой системы контроллера с именем *\USB Storage*, а подключенный вторым – *\USB Storage2*.

Перед любым обращением к файлу или каталогу на съемном USB-накопителе необходимо обязательно убедиться в том, что накопитель подключен к контроллеру:

```
IF FwSysDirExist('\USB Storage') <> 0 THEN
(* первый съемный USB-накопитель подключен, можно обращаться *)
(* к его файлам и каталогам *)
END_IF
```

Данное правило необходимо соблюдать особенно тщательно, если предполагается создавать и удалять каталоги на USB-накопителе:

```
IF FwSysDirExist('\USB Storage') <> 0 THEN
   (* первый съемный USB-накопитель подключен, можно создать на нем каталог, *)
   (* если он не был создан ранее *)
   IF FwSysDirExist('\USB Storage\Archive') = 0 THEN
   (* если на 1-м USB-накопителе отсутствует каталог Archive, *)
   (* создаем его *)
       FwSysDirCreate('\USB Storage\Archive');
   END_IF
END_IF
```

Если выполнить операции:

```
   IF FwSysDirExist('\USB Storage\Archive') = 0 THEN
   (* если отсутствует каталог Archive, *)
   (* создаем его *)
       FwSysDirCreate('\USB Storage\Archive');
   END_IF
```

при отсутствии USB-накопителя, подключенного к контроллеру, то каталог '\USB Storage\Archive' будет создан в корневом каталоге файловой системы, содержимое которого (за исключением подмонтированных энергонезависимых накопителей) исчезнет после выключения питания контроллера.

ВНИМАНИЕ! Извлечение USB-накопителя во время записи на него данных может привести к разрушению файловой системы накопителя.

6.2.2. Описание функций

6.2.2.1. FwSysFileGetSize

Возвращает размер файла, имя которого передано в качестве параметра.

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

Входные параметры:

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

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

Возвращаемый результат:

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

6.2.2.2. FwSysFileExists

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

Входные параметры:

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

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

Возвращаемый результат:

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

6.2.2.3. FwSysFileGetTime

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

Входные параметры:

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

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

ftFileTime: POINTER TO FILETIME;
Указатель на структуру, содержащую информацию о времени создания, последнего доступа и последней модификации файла.
Структура имеет следующий вид:

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

Возвращаемый результат:

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

6.2.2.4. FwSysFileCopy

Создает копию файла, имя которого указано в качестве второго параметра

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

Входные параметры:

FileDest: STRING
Имя файла-копии. Файл создается относительно каталога \user, если используются относительные пути.. Пример имен, в которых используются относительные пути:

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

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

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

Возвращаемый результат:

возвращает количество скопированных байт.

6.2.2.5. FwSysFileDelete

Функция удаляет файл с заданным именем, возвращая ненулевое значение в случае успеха. Поиск файла выполняется относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

6.2.2.6. FwSysFileRename

Функция изменяет имя файла, заданное в качестве первого параметра, на имя, заданное в качестве второго параметра, возвращая ненулевое значение в случае успеха. Поиск файла выполняется относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

6.2.2.7. FwSysFileOpen

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

Максимальное количество одновременно открытых файлов ограничено 16-ю.

По завершении работы с файлом его необходимо закрыть, иначе операции записи могут быть не завершены. При загрузке приложения пользователя непосредственно перед началом переконфигурирования контроллера система автоматически закрывает все незакрытые пользователем файлы.

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

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

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

Входные параметры:

FileName: STRING

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

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

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

```
(* временный файл в каталоге \Temp *)
fileName1 : STRING := '\Temp\MyData.dat';
(* файл в каталоге \subdir на первом USB-накопителе *)
fileName2 : STRING := '\USB Storage\subdir\MyData.dat';
```

Mode: STRING

Режим открытия:

'r' – открыть только для чтения;

'w' – открыть только для записи;

'rw' – открыть для чтения и записи;

'cw' – открыть для чтения и записи, если файл не найден – создать;

Возвращаемый результат:

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

Пример:

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

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

VAR_INPUT
   FileName            : STRING := '';
   (* Имя файла *)
   pData               : POINTER TO BYTE := 0;
   (* Указатель на данные, подлежащие сохранению *)
   DataSize            : DWORD :=0;
   (* Количество байт для сохранения *)
   AppendMode          : BOOL := FALSE;
   (* Режим сохранения: TRUE -- добавление в конец файла *)
END_VAR

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

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

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

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

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

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

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

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

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

```
    END_IF

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

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

    FwSysFileClose(handle);
END_IF

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

END_PROGRAM
```

6.2.2.8. FwSysFileClose

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

6.2.2.9. FwSysFileGetPos

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

6.2.2.10. FwSysFileSetPos

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

6.2.2.11. FwSysFileRead

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

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

Входные параметры:

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

Возвращаемый результат:

Количество прочитанных байт.

6.2.2.12. FwSysFileWrite

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

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

Входные параметры:

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

Возвращаемый результат:

Количество записанных байт.

6.2.2.13. FwSysFileEOF

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

6.2.2.14. FwSysDirCreate

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

Пример относительного пути к создаваемому каталогу:

```
MyDir1\Temp1\TargetDir
```

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

Пример абсолютного пути к создаваемому каталогу:

```
\USB Storage\Temp1\TargetDir
```

В данном случае в корневом каталоге \ файловой системы контроллера сначала будет выполнен поиск каталога USB Storage. При отрицательном результате USB Storage будет создан в каталоге \, после чего в нем будет создан каталог Temp1, а в каталоге Temp1 – TargetDir. Таким образом, перед созданием каталога на съемном USB-накопителе следует убедиться в наличии подключенного накопителя путем вызова FwSysDirExist с передачей ‘\USB Storage’ или ‘\USB Storage2’ в качестве параметра.

Прототип функции FwSysDirCreate

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

Входные параметры:

DirName: STRING
Полный путь к создаваемому каталогу. Компоненты пути являются именем каталогов, которые будут созданы в существующих каталогах, представленных предшествующими компонентами пути.
Создание производится относительно каталога \user, если используются относительные пути. Пример имен, в которых используются относительные пути:

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

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

```
(* в каталоге \ будет создан подкаталог \Temp *)
dirName1 : STRING := '\Temp';
(* Если 1-й USB-накопитель подключен, то на нем *)
(* будут созданы каталоги \subdir и \subdir\mydir, *)
(* Если 1-й USB-накопитель не подключен, то каталоги \subdir и \subdir\mydir *)
(* будут созданы в корневом каталоге файловой системы \ *)
dirName2 : STRING := '\USB Storage\subdir\mydir';
```

Возвращаемый результат:

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

6.2.2.15. FwSysDirExist

Возвращает ненулевое значение, если найден каталог с заданным именем.

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

Входные параметры:

DirName: STRING

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

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

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

```
(* в каталоге \ ищется подкаталог \Temp *)
dirName1 : STRING := '\Temp';
(* Если 1-й USB-накопитель подключен, то на нем *)
(* будет выполнен поиск подкаталога \subdir\mydir, *)
(* Если 1-й USB-накопитель не подключен, то *)
(* поиск \subdir\mydir будет выполнен в корневом каталоге файловой системы \ *)
dirName2 : STRING := '\USB Storage\subdir\mydir';
```

6.2.2.16. FwSysDirRemove

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

Пример относительного пути к удаляемому каталогу:

```
MyDir1\Temp1\TargetDir
```

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

Пример абсолютного пути к удаляемому каталогу:

```
\USB Storage\Temp1\TargetDir
```

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

Причиной неудачного удаления может быть наличие файлов в удаляемом каталоге.

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

Входные параметры:

DirName: STRING

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

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

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

```
(* в каталоге \ будет удален подкаталог \Temp *)
dirName1 : STRING := '\Temp';
(* Если 1-й USB-накопитель подключен, то на нем *)
(* будет выполнено удаление подкаталога \subdir\mydir, *)
(* Если 1-й USB-накопитель не подключен, то *)
(* удаление \subdir\mydir будет выполнено в корневом каталоге файловой системы \ *)
dirName2 : STRING := '\USB Storage\subdir\mydir';
```

6.3. Библиотека FastwelTasksExchange.lib

6.3.1. Общие сведения

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

F_IecTasks_getCount – возвращает общее количество циклических и ациклических задач;

F_getProjectName – возвращает имя проекта, загруженного в контроллер;

F_IecTasks_getInfo – возвращает диагностическую информацию о циклической или ациклической задаче.

Описание применения функции F_IecTasks_linkVariable приведено в п. 4.2.4.5 настоящего руководства.

6.3.2. Функция F_IecTasks_getInfo

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

Структура F_TASK_INFO определена следующим образом:

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

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

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

6.4. Библиотека FastwelSysLibCom.lib

6.4.1. Общие сведения

Функции данной библиотеки предназначены для непосредственного доступа к коммуникационным портам COM2, COM5...COM8 из приложений пользователя. По сути данная библиотека является полной копией исходной библиотеки SysLibCom.lib, входящей в стандартную среду исполнения CoDeSys, однако из-за ряда особенностей компилятора среды CoDeSys, присутствовавших в одной из предыдущих версий CoDeSys, в FastwelSysLibCom.lib все возвращаемые результаты типа BOOL заменены на тип WORD. При этом сохранена семантика возвращаемого результата: 0 означает FALSE, а не 0 – TRUE.

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

Пример использования функций библиотеки имеется в проекте \Examples\MK905\MK905_SysLibComTest.pro, который входит в пакет адаптации CoDeSys для MK905. Открывать коммуникационный порт следует при обработке системного события *OnInit*. При обработке системного события *OnProgramChange* порт обязательно необходимо закрыть, иначе новая версия загруженной программы не получит к нему доступ.

Для того, чтобы функции библиотеки имели возможность доступа к коммуникационным портам, необходимо, чтобы в конфигурации контроллера для соответствующих портов была выбрана опция *Not Used*.

6.4.2. Описание функций

6.4.2.1. FwSysComOpen

Функция открывает коммуникационный порт контроллера для использования.

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

Входные параметры:

Port: PORTS
Идентификатор порта перечислимого типа PORTS: (COM1:=1, COM2, COM3, COM4, COM5, COM6, COM7, COM8). Допустимые значение для контроллера: COM2, COM5–COM8.

Возвращаемый результат:

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

6.4.2.2. FwSysComClose

Функция освобождает коммуникационный порт.

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

Входные параметры:

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

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успеха.

6.4.2.3. FwSysComSetSettings

Устанавливает параметры коммуникационного порта: скорость и параметры передачи, размер коммуникационного буфера. Для портов COM5–COM8 устанавливаемая скорость обмена должна быть от 1200 бит/с.

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

Входные параметры:

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

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

Размер приемного и передающего буферов будут равны dwBufferSize.

Возвращаемый результат:

Функция возвращает ненулевое значение в случае успеха.

6.4.2.4. FwSysComRead

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

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

Входные параметры:

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

Возвращаемый результат:

Функция возвращает фактический размер считанных данных.

6.4.2.5. FwSysComWrite

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

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

Входные параметры:

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

Возвращаемый результат:

Количество записанных байт.

6.5. Библиотека FastwelSysLibSockets.lib

6.5.1. Общие сведения

Данная библиотека поддерживает работу с сокетами для коммуникаций по TCP/IP и UDP, по сути являясь аналогом исходной библиотеки SysLibSockets.lib, входящей в стандартную среду исполнения CoDeSys.

Следует обратить внимание на тот факт, что в библиотеке FastwelSysLibSockets.lib реализованы только неблокирующиеся сокеты, и выполнение функций асинхронно по отношению к потоку исполнения, из которого вызываются функции. Таким образом, функции библиотеки не используют таймаутов ожидания и не приостанавливают выполнение вызывающей программы CoDeSys.

Пример использования библиотеки имеется в проектах *fsyslibsock_srv_mk905.pro* и *fsyslibsock_clnt_mk905.pro*, которые входят в пакет адаптации CoDeSys для Fastwel I/O.

6.5.2. Описание функций

6.5.2.1. FwSysSockCreate

Функция создает новый сокет. Возвращает дескриптор сокета, используемый другими функциями (например, FwSysSockBind, FwSysSockConnect).

```
FUNCTION FwSysSockCreate : DINT
VAR_INPUT
   diAddressFamily:DINT;
   diType:DINT;
   diProtocol:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diAddressFamily:DINT
Идентификатор формата адреса. Единственное допустимое значение: SOCKET_AF_INET.
diType:DINT

Идентификатор типа создаваемого сокета. Одно из двух допустимых значений:
SOCKET_STREAM – используется при создании сокета для TCP соединения;
SOCKET_DGRAM – используется при создании сокета для UDP соединения.

diProtocol:DINT

Идентификатор протокола. Одно из двух допустимых значений:
SOCKET_IPPROTO_TCP – используется при создании сокета для TCP соединения;
SOCKET_IPPROTO_UDP – используется при создании сокета для UDP соединения.

Возвращаемый результат:

Системный идентификатор сокета, который в дальнейшем используется в вызовах других функций библиотеки. В случае ошибки (если сокет не может быть создан) возвращается значение: SOCKET_INVALID. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError.**

6.5.2.2. FwSysSockClose

Функция закрывает сокет, разрывая соединение с удаленным узлом.

```
FUNCTION FwSysSockClose : BOOL
VAR_INPUT
   diSocket:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть уточнен вызовом функции **FwSysSockGetLastError**.

6.5.2.3. FwSysSockListen

Функция устанавливает сокет в режим сервера, ожидающего запросы на установление соединений с удаленными клиентами.

```
FUNCTION FwSysSockListen : BOOL
VAR_INPUT
   diSocket:DINT;
   diMaxConnections:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета, созданного функцией **FwSysSockCreate.**

diMaxConnections:DINT
Максимальное число запросов в очереди на соединение. Допустимое значение от 1 до 5.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.4. FwSysSockAccept

Функция принимает входящее соединение, ожидаемое на серверном сокете и, в случае успеха, создает сокет соединения с клиентом. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockAccept : DINT
VAR_INPUT
   diSocket:DINT;
   pSockAddr:DWORD;
   piSockAddrSize:DWORD;
END_VAR
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Идентификатор серверного сокета, установленного в режим ожидания входящих соединений функцией **FwSysSockListen**.
pSockAddr:DWORD
Указатель на переменную типа SOCKADDR, которая будет заполнена адресом вызывающего клиента. Если передается 0 информация о клиенте не возвращается.
piSockAddrSize:DWORD
Указатель на переменную типа DINT, содержащую размер структуры SOCKADDR (может быть получен оператором SIZEOF). Если передается 0, информация о клиенте не возвращается.

Возвращаемый результат:

Системный идентификатор сокета нового клиентского соединения. В случае ошибки возвращается значение SOCKET_INVALID. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.5. FwSysSockBind

Функция ассоциирует сокет с локальным IP адресом и портом. Используется на вновь созданном сокете до вызова **FwSysSockConnect** или **FwSysSockListen.**

```
FUNCTION FwSysSockBind : BOOL
VAR_INPUT
   diSocket:DINT;
   pSockAddr:DWORD;
   diSockAddrSize:DINT;
END_VAR
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета созданного функцией **FwSysSockCreate**.
pSockAddr:DWORD
Указатель на переменную типа SOCKADDR.
diSockAddrSize:DINT
Размер структуры SOCKADDR (может быть получен оператором SIZEOF).

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.6. FwSysSockConnect

Функция устанавливает соединение с удаленным сервером. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockConnect : BOOL
VAR_INPUT
   diSocket:DINT;
   pSockAddr:DWORD;
   diSockAddrSize:DINT;
END_VAR
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета созданного функцией **FwSysSockCreate**.
pSockAddr:DWORD
Указатель на переменную типа SOCKADDR.
diSockAddrSize:DINT
Размер структуры SOCKADDR (может быть получен оператором SIZEOF).

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**. Значение кода ошибки SOCKET_EWOULDBLOCK указывает на то, что процесс установки соединения инициирован, но не завершен к настоящему времени. В этом случае фактический результат установки соединения с сервером должен быть в дальнейшем получен проверкой статусов «записи» и «ошибок» для данного сокета функцией **FwSysSockSelect**.

6.5.2.7. FwSysSockGetOption

Функция получает значение соответствующей опции сокета.

```
FUNCTION FwSysSockGetOption : BOOL
VAR_INPUT
   diSocket:DINT;
   diLevel:DINT;
   diOption:DINT;
   pOptionValue:DWORD;
   piOptionLength:DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета, созданного функцией **FwSysSockCreate**.
diLevel:DINT
Специфический уровень протокола:
SOCKET_SOL – уровень сокетов,
SOCKET_IPPROTO_TCP – уровень протокола TCP.
diOption:DINT
Идентификатор имени опции. **FwSysSockSetOption** поддерживаются следующие опции:
Уровень SOCKET_IPPROTO_TCP:

Код	Тип	Значение
SOCKET_TCP_NODELAY	BOOL	Выключает Nagle алгоритм (запрещает передачу новых TCP сегментов при поступлении новых выходных данных пользователя, если имеются ранее отправленные неподтвержденные данные).

Уровень SOCKET_SOL:

Код	Тип	Значение
SOCKET_SO_BROADCAST	BOOL	Позволяет передавать и принимать широковещательные сообщения.
SOCKET_SO_KEEPALIVE	BOOL	Посылает сообщения “keep-alive”.
SOCKET_SO_ACCEPTCONN	BOOL	Режим приема входящих соединений.
SOCKET_SO_ERROR	DINT	Чтение и очистка статуса ошибки.
SOCKET_SO_TYPE	DINT	Тип сокета.
SOCKET_SO_RCVBUF	DINT	Размер буфера приемной очереди.
SOCKET_SO_REUSEADDR	BOOL	Позволяет связать сокет с локальным адресом, который уже используется на другом открытом сокете.
SOCKET_SO_SNDBUF	DINT	Размер буфера выходной очереди.

pOptionValue:DWORD
Указатель на буфер, в который будет записано значение опции.
piOptionLength:DWORD
Указатель на размер буфера.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.8. FwSysSockSetOption

Функция устанавливает значение соответствующей опции сокета.

```
FUNCTION FwSysSockSetOption : BOOL
VAR_INPUT
   diSocket:DINT;
   diLevel:DINT;
   diOption:DINT;
   pOptionValue:DWORD;
   diOptionLength:DINT;
END_VAR
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета созданного функцией **FwSysSockCreate**.
diLevel:DINT
Специфический уровень протокола:
SOCKET_SOL – уровень сокетов,
SOCKET_IPPROTO_TCP – уровень протокола TCP.
diOption:DINT
Идентификатор имени опции. **FwSysSockSetOption** поддерживаются следующие опции:
Уровень SOCKET_SOL:

Код	Тип	Значение
SOCKET_SO_BROADCAST	BOOL	Позволяет передавать и принимать широковещательные сообщения.
SOCKET_SO_KEEPALIVE	BOOL	Посылает сообщения “keep-alive”.
SOCKET_SO_RCVBUF	DINT	Устанавливает размер буфера приемной очереди для сокета.
SOCKET_SO_REUSEADDR	BOOL	Позволяет связать сокет с локальным адресом, который уже используется на другом открытом сокете.
SOCKET_SO_SNDBUF	DINT	Устанавливает размер буфера выходной очереди для сокета.

Уровень SOCKET_IPPROTO_TCP:

Код	Тип	Значение
SOCKET_TCP_NODELAY	BOOL	Выключает Nagle алгоритм (запрещает передачу новых TCP сегментов при поступлениии новых выходных данных пользователя, если на соединении остаются неподтвержденными ранее отправленные данные).

pOptionValue:DWORD
Указатель на буфер, в который записано значение опции.
diOptionLength:DINT
Размер буфера значения опции.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.9. FwSysSockIoctl

Функция управления режимом ввода/вывода сокета.

```
FUNCTION FwSysSockIoctl : DINT
VAR_INPUT
   diSocket:DINT;
   diCommand:DINT;
   piParameter:DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета созданного функцией **FwSysSockCreate**.
diCommand:DINT
Команда, которую нужно применить к сокету. Допустимые значения: SOCKET_FIONBIO, SOCKET_FIONREAD.
piParameter:DWORD
Указатель на параметр команды.

Команда	Тип параметра	Значение
SOCKET_FIONBIO	DINT	Не 0 – устанавливает неблокирующий режим ввода/вывода для сокета. 0 – устанавливает блокирующий режим. **Внимание! В текущей реализации поддерживается только неблокирующий режим, который устанавливается по умолчанию при создании сокета. Попытка задания блокирующего режима заканчивается неудачей.**
SOCKET_FIONREAD	DINT	Используется для получения числа байт, находящихся в приемном буфере сокета и готовых для считывания.

Возвращаемый результат:

Возвращает 0 в случае успеха, иначе SOCKET_INVALID. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.10. FwSysSockSelect

Функция проверки одного или нескольких сокетов на готовность к определенным коммуникационным действиям. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockSelect : DINT
VAR_INPUT
   diWidth:DINT;
   fdRead:DWORD;
   fdWrite:DWORD;
   fdExcept:DWORD;
   ptvTimeout:DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diWidth:DINT
Единственное допустимое значение SOCKET_FD_SETSIZE.
fdRead:DWORD
Опциональный указатель на структуру, определяющую сокеты, для которых необходимо проверить статус операции чтения. Может быть 0.
fdWrite:DWORD
Опциональный указатель на структуру, определяющую сокеты, для которых необходимо проверить статус операции записи. Может быть 0.
fdExcept:DWORD
Опциональный указатель на структуру, определяющую сокеты, для которых необходимо проверить статус ошибок. Может быть 0.
ptvTimeout:DWORD
Игнорируется.

Возвращаемый результат:

В случае успеха возвращается суммарное число сокетов, удовлетворяющих условиям проверки. В случае ошибки возвращается значение SOCKET_INVALID. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.11. FwSysSockRecv

Функция считывает данные, полученные от удаленного узла соединения. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockRecv : DINT
VAR_INPUT
   diSocket:DINT;
   pbyBuffer:DWORD;
   diBufferSize:DINT;
   diFlags:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета с установленным соединением.
pbyBuffer:DWORD
Адрес буфера данных.
diBufferSize:DINT
Размер буфера данных.
diFlags:DINT
Опции вызова функции скомбинированные битовым оператором ИЛИ:

Имя	Значение
SOCKET_MSG_PEEK	Данные копируются в буфер пользователя, но не удаляются из входной очереди. Возвращает число байт данных, которое может быть считано одной операцией чтения.

Возвращаемый результат:

Возвращает число принятых байт (если установлена опция SOCKET_MSG_PEEK, возвращает число байт, которое может быть считано одной операцией чтения), или SOCKET_INVALID в случае ошибки. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**. Если соединение было закрыто удаленным узлом, возвращаемое значение равно 0.

6.5.2.12. FwSysSockRecvFrom

Функция считывает данные, полученные от удаленного узла соединения. Для сокетов UDP соединений (тип SOCKET_DGRAM) сохраняет адрес узла, являющегося источником данных. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockRecvFrom : DINT
VAR_INPUT
   diSocket:DINT;
   pbyBuffer:DWORD;
   diBufferSize:DINT;
   diFlags:DINT;
   pSockAddr:DWORD;
   diSockAddrSize:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета с установленным соединением.
pbyBuffer:DWORD
Адрес буфера данных.
diBufferSize:DINT
Размер буфера данных.
diFlags:DINT
Опции вызова функции скомбинированные битовым оператором ИЛИ:

Имя	Значение
SOCKET_MSG_PEEK	Данные копируются в буфер пользователя, но не удаляются из входной очереди. Возвращает число байт данных, которое может быть считано одной операцией чтения.

pSockAddr:DWORD
Указатель на переменную типа SOCKADDR, в которую будет записан адрес узла, отправителя данных.
diSockAddrSize:DINT
Размер структуры SOCKADDR (может быть получен оператором SIZEOF).

Возвращаемый результат:

Возвращает число принятых байт (если установлена опция SOCKET_MSG_PEEK возвращает число байт, которое может быть считано одной операцией чтения), или SOCKET_INVALID в случае ошибки. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**. Если соединение было закрыто удаленным узлом, возвращаемое значение равно 0.

6.5.2.13. FwSysSockSend

Функция записывает данные в буфер передачи сокета. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockSend : DINT
VAR_INPUT
   diSocket:DINT;
   pbyBuffer:DWORD;
   diBufferSize:DINT;
   diFlags:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета с установленным соединением.
pbyBuffer:DWORD
Адрес буфера данных.
diBufferSize:DINT
Размер буфера данных.

diFlags:DINT
Опции вызова функции. В текущей реализации значение аргумента игнорируется.

Возвращаемый результат:

Возвращает число записанных байт, которое может быть меньше переданного числа байт, или SOCKET_INVALID в случае ошибки. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.14. FwSysSockSendTo

Функция записывает данные в буфер передачи сокета для отправки данных по указанному адресу. Функция не блокирует выполнение вызывающей программы.

```
FUNCTION FwSysSockSendTo : DINT
VAR_INPUT
   diSocket:DINT;
   pbyBuffer:DWORD;
   diBufferSize:DINT;
   diFlags:DINT;
   pSockAddr:DWORD;
   diSockAddrSize:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета с установленным соединением.
pbyBuffer:DWORD
Адрес буфера данных.
diBufferSize:DINT
Размер буфера данных.
diFlags:DINT
Опции вызова функции. В текущей реализации значение аргумента игнорируется.
pSockAddr:DWORD
Указатель на переменную типа SOCKADDR, в которую записан адрес узла, являющегося получателем данных.
diSockAddrSize:DINT
Размер структуры SOCKADDR (может быть получен оператором SIZEOF). Для сокетов TCP соединений (тип SOCKET_STREAM) аргументы **pSockAddr** и **pSockAddr** игнорируются. Вызов **FwSysSockSendTo** эквивалентен вызову **FwSysSockSend**.

Возвращаемый результат:

Возвращает число записанных байт, которое может быть меньше переданного числа байт, или SOCKET_INVALID в случае ошибки. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.15. FwSysSockShutdown

Функция запрещает операции передачи и/или чтения данных для сокета.

```
FUNCTION FwSysSockShutdown : BOOL
VAR_INPUT
   diSocket:DINT;
   diHow:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

diSocket:DINT
Системный идентификатор сокета созданного функцией **FwSysSockCreate**.
diHow:DINT
Тип запрещаемых действий. Допустимые значения: SOCKET_SD_RECEIVE, SOCKET_SD_SEND, SOCKET_SD_BOTH.

Имя	Значение
SOCKET_SD_RECEIVE	Запрещает последующие вызовы операций чтения данных.
SOCKET_SD_SEND	Запрещает последующие вызовы операций передачи данных.
SOCKET_SD_BOTH	Запрещает последующие вызовы операций чтения и передачи данных.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.16. FwSysSockHtons

Функция конвертирует переменную типа WORD в соответствии с порядком байт в сетях TCP/IP.

```
FUNCTION FwSysSockHtons : WORD
VAR_INPUT
   wHost:WORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

wHost:WORD
Значение для конвертирования.

Возвращаемый результат:

Возвращает конвертированное значение.

6.5.2.17. FwSysSockHtonl

Функция конвертирует переменную типа DWORD в соответствии с порядком байт в сетях TCP/IP.

```
FUNCTION FwSysSockHtonl : DWORD
VAR_INPUT
   dwHost:DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

dwHost:DWORD
Значение для конвертирования.

Возвращаемый результат:

Возвращает конвертированное значение.

6.5.2.18. FwSysSockNtohs

Функция конвертирует переменную типа WORD, представленную в соответствии с порядком байт в сетях TCP/IP в порядок хоста.

```
FUNCTION FwSysSockNtohs : WORD
VAR_INPUT
   wNet:WORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

wNet:WORD
Значение для конвертирования.

Возвращаемый результат:

Возвращает конвертированное значение.

6.5.2.19. FwSysSockNtohl

Функция конвертирует переменную типа DWORD, представленную в соответствии с порядком байт в сетях TCP/IP в порядок хоста.

```
FUNCTION FwSysSockNtohl : DWORD
VAR_INPUT
   dwNet:DWORD;
END_VAR
;
END_FUNCTION
```

Входные параметры:

dwNet:DWORD
Значение для конвертирования.

Возвращаемый результат:

Возвращает конвертированное значение.

6.5.2.20. FwSysSockGetLastError

Функция возвращает код ошибки последней операции.

```
FUNCTION FwSysSockGetLastError : INT
VAR_INPUT
END_VAR
;
END_FUNCTION
```

6.5.2.21. FwSysSockGetHostName

Функция получает имя хоста.

```
FUNCTION FwSysSockGetHostName : BOOL
VAR_INPUT
   stHostName:POINTER TO STRING;
   diNameLength:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

stHostName:POINTER TO STRING
Указатель на переменную типа STRING, в которую будет записано имя хоста.
diNameLength:DINT
Размер буфера строки.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.22. FwSysSockInetAddr

Функция конвертирует строку, содержащую интернет адрес, в адрес, используемый в структуре INADDR.

```
FUNCTION FwSysSockInetAddr : DWORD
VAR_INPUT
   stIPAddr:STRING;
END_VAR
;
END_FUNCTION
```

Входные параметры:

stIPAddr:STRING
IP адрес.

Возвращаемый результат:

Возвращает конвертированное значение.

6.5.2.23. FwSysSockInetNtoa

Функция конвертирует сетевой интернет адрес в строку стандартного формата.

```
FUNCTION FwSysSockInetNtoa : BOOL
VAR_INPUT
   pInAddr:DWORD;
   stIPAddr:STRING;
   diIPAddrSize:DINT;
END_VAR
;
END_FUNCTION
```

Входные параметры:

pInAddr:DWORD
Указатель на структуру INADDR, содержащую Интернет адрес.
stIPAddr:STRING
Указатель на строку, в которую будет записано конвертированное значение.
diIPAddrSize:DINT
Размер буфера строки.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.2.24. FwSysSockSetIPAddress

Функция устанавливает IP адрес заданной сетевой платы.

```
FUNCTION FwSysSockSetIPAddress : BOOL
VAR_INPUT
   stCardName:STRING;
   stIPAddress:STRING;
   stMask:STRING;
END_VAR
;
END_FUNCTION
```

Входные параметры:

stCardName:STRING
Имя сетевой платы: "Ethernet1" или "Ethernet2".
stIPAddress:STRING
IP адрес.
stMask:STRING
Маска подсети.

Возвращаемый результат:

Возвращает TRUE в случае успеха, иначе FALSE. Код ошибки может быть получен вызовом функции **FwSysSockGetLastError**.

6.5.3. Описание типов данных

6.5.3.1. INADDR

```
TYPE INADDR :
STRUCT
   S_addr:DWORD;
END_STRUCT
END_TYPE
```

Поля структуры:

S_addr:DWORD;
Интернет адрес.

6.5.3.2. SOCKADDRESS

Тип данных SOCKADDRESS используется для представления информации об адресе IP соединения.

```
TYPE SOCKADDRESS :
STRUCT
   sin_family:INT;
   sin_port:UINT;
   sin_addr:UDINT;
   sin_zero:ARRAY [0..7] OF SINT;
END_STRUCT
END_TYPE
```

Поля структуры:

sin_family:INT;

Идентификатор формата адреса. Единственное возможное значение при работе с функциями библиотеки – SOCKET_AF_INET.

sin_port:UINT;

Порт. Значение поля отображается в соответствии с порядком байт в сетях TCP/IP.

sin_addr:UDINT;

IP-адрес. Значение поля отображается в соответствии с порядком байт в сетях TCP/IP.

sin_zero:ARRAY [0..7] OF SINT;

Буфер.

6.5.3.3. SOCKET_FD_SET

Тип данных SOCKET_FD_SET описывает некоторое множество (набор) сокетов. Используется с функцией **FwSysSockSelect** с целью проверки сокетов набора на готовность к определенным коммуникационным действиям.

```
TYPE SOCKET_FD_SET :
STRUCT
   fd_count:UDINT;
   fd_array:ARRAY [0..MAX_SOCKET_FD_SETSIZE] OF DINT;
END_STRUCT
END_TYPE
```

Поля структуры:

fd_count:UDINT

Число сокетов в наборе.

fd_array:ARRAY [0..MAX_SOCKET_FD_SETSIZE] OF DINT

Массив идентификаторов сокетов. Идентификаторы находятся в первых **fd_count** ячейках массива.

6.5.4. Коды ошибок

В списке приведены коды ошибок возвращаемые функцией **FwSysSockGetLastError**. Ошибки перечислены в порядке возрастания их числового значения:

Имя / значение	Описание
SOCKET_EIO 5	*Ошибка ввода/вывода*. Ошибка инициализации библиотеки или другая ошибка сетевого уровня.
SOCKET_EBADF 9	*Неверное значение идентификатора*. В качестве идентификатора сокета передано неверное или неизвестное значение.
SOCKET_EWOULDBLOCK 11	*Ресурс временно не доступен.* Поскольку библиотека поддерживает только неблокирующие сокеты, ошибка возникает на операциях, которые не могут быть завершены без ожидания. Например, при вызове функции FwSysSockRecv в случае отсутствия принятых данных. Или при вызове FwSysSockConnect для установки TCP соединения, т.к. требуется время для выполнения фактического соединения с удаленным узлом.
SOCKET_ENOMEM 12	*Не хватает памяти.* Операция на сокете не была выполнена из-за недостаточного размера буфера памяти, или переполнения очереди заданий.
SOCKET_EFAULT 14	*Неверное значение адреса*. Неправильное значения указателя на объекта или неправильное значение размера объекта при передаче объекта в параметрах вызова.
SOCKET_EINVAL 22	*Недопустимое значение аргумента*. Неправильное или недопустимое значение параметра при вызове. Например, указан недостаточный размер буфера объекта при записи/чтении опции сокета в вызове FwSysSockSetOption/FwSysSockGetOption. Или недопустимое использование вызова. Например, вызов функции FwSysSockAccept для сокета, не находящегося в состоянии listen.
SOCKET_ENFILE 23	*Нет свободных дескрипторов сокетов*. Закончилось зарезервированное число сокетов (для СРВ902 - 64).
SOCKET_ENOSYS 38	*Операция не поддерживается*. Функция, команда или опция управления не поддерживается реализацией библиотеки для данного контроллера.

Имя / значение	Описание
SOCKET_ENOPROTOOPT 92	*Неверная опция протокола*. Неизвестная или неподдерживаемая опция протокола при вызове FwSysSockSetOption, FwSysSockGetOption.
SOCKET_EAFNOSUPPORT 97	*Формат адреса не поддерживается*. Единственное допустимое значение для сокетов библиотеки - SOCKET_AF_INET.
SOCKET_EADDRINUSE 98	*Адрес используется*. Например, при попытке связать сокет с локальным адресом (значением адрес/порт) которое уже используется на другом открытом сокете.
SOCKET_EADDRNOTAVAIL 99	*Недопустимое значение адреса*. Например, при попытке связать сокет с несуществующим локальным адресом.
SOCKET_ECONNABORTED 103	*Соединение разорвано*. Установленное соединение было разорвано, возможно, из-за ошибок или истечения таймаутов протокола
SOCKET_ECONNRESET 104	*Соединение разорвано удаленным узлом*. Установленное соединение было разорвано по инициативе удаленного узла.
SOCKET_ENOBUFS 105	*Нет свободных дескрипторов соединений*.
SOCKET_EISCONN 106	*Соединение уже установлено*. Запрос операции, требующей соединения на сокете с уже установленным соединением.
SOCKET_ENOTCONN 107	*Соединение не установлено*. Вызов функций чтения/передачи данных для сокета с неустановленным соединением.
SOCKET_ESHUTDOWN 108	*Операция отключена*. Попытка чтения/передачи данных для сокета с выключенной функцией приема/передачи.
SOCKET_EALREADY 114	*Операция выполняется*.
SOCKET_EINPROGRESS 115	*Предыдущая операция с сокетом не завершена*. Выполняется операция с сокетом, требующая блокировки. Данная ошибка также может проявляться в случае конкурентного доступа к сокету из разных задач, что не рекомендуется.

6.6. Библиотека FastwelUtils.lib

6.6.1. FwCheckSum16

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

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

Входные параметры:

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

bufferLength : WORD
Размер участка.

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

Возвращаемый результат:

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

6.6.2. FwCheckSum32

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

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

Входные параметры:

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

bufferLength : DWORD
Размер участка.

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

Возвращаемый результат:

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

6.6.3. FwCRC16

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

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

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

Входные параметры:

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

bufferLength : WORD
Размер участка.

Возвращаемый результат:

16-разрядная CRC всех байт заданного участка памяти.

6.6.4. FwCRC32

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

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

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

Входные параметры:

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

bufferLength : DWORD
Размер участка.

startCRC : DWORD
Начальное значение CRC32.

Возвращаемый результат:

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

6.6.5. FwIsPOUExist

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

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

Входные параметры:

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

Возвращаемый результат:

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

6.6.6. FwGetPOU_CRC32

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

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

Входные параметры:

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

Возвращаемый результат:

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

6.6.7. FwMemCompare

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

Если содержимое участков отличается, возвращает ненулевое значение.

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

6.6.8. FwMemCopy

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

Функция возвращает TRUE тогда, и только тогда, когда:

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

6.7. SysLibGetAddress.lib

6.7.1. Общие сведения

Библиотека содержит две функции, позволяющие получить адреса и размеры сегментов данных приложения. Для идентификации сегментов используется библиотечный перечислимый тип ADDRESS_SEGMENTS:

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

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

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

Значение DATAID_RETAIN идентифицирует сегмент энергонезависимых переменных.

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

6.7.2. SysLibGetAddress

Функция возвращает указатель на начало сегмента данных приложения, идентификатор которого передан функции в качестве параметра..

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

6.7.3. SysLibGetSize

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

Для идентификатора DATAID_MEMORY возвращается **полный** размер сегмента флагов (65536), независимо от того, каков реальный суммарный размер размещенных в нем переменных приложения.

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

7. Сетевые средства

7.1. Общие сведения

Предустановленное системное программное обеспечение MK905 включает сервисы внешней сети, которые могут использоваться для обмена данными между приложением *CoDeSys* и другими устройствами по протоколу прикладного уровня MODBUS в качестве мастера – сервис клиента MODBUS; или подчиненного узла сети – сервис сервера MODBUS.

Портами сервисов внешней сети контроллера являются коммуникационные порты аппаратных интерфейсов Ethernet: *Ethernet1*, *Ethernet2* (реализуется функциональность протокола MODBUS TCP); и УАПП интерфейсов RS-232/RS-485: *COM2*, *COM5*, *COM6*, *COM7*, *COM8* (реализуется функциональность протоколов MODBUS RTU, ASCII).

Наиболее актуальную спецификацию протоколов MODBUS можно загрузить с Web-узла http://www.modbus.org.

Порты COM5–COM8 обеспечивают возможность функционирования только в режиме RS-485.

7.2. Характеристики сетевых средств

7.2.1. Характеристики сервера MODBUS TCP

Сервис сервера MODBUS TCP реализует функциональность подчиненного узла протокола MODBUS TCP. Перечень основных характеристик сервиса приведен в табл. 7.

Таблица 7

Тип узла протокола MODBUS	Подчиненный	
Скорость обмена, Мбит/с	10, 100 (автоопределение)	
Адресация IP	статическая, по конфигурации контроллера из проекта CoDeSys	
Адрес по умолчанию	10.0.0.1	
Количество клиентских соединений по MODBUS TCP	не более 32-х	
Операции протокола MODBUS	*Тип*	*Описание*
	01	Выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для записи
	02	Выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для чтения
	03	Выдача за один запрос от 1 до 125 смежных регистров, доступных для записи
	04	Выдача за один запрос от 1 до 125 смежных регистров, доступных для чтения
	05	Прием значения одного битового поля, доступного для записи
	06	Прием значения одного регистра, доступного для записи
	15	Прием за один запрос значений до 2000 смежных битовых полей, доступных для записи
	16	Прием за один запрос значений до 125 смежных регистров, доступных для записи
	22	Изменение содержимого заданного регистра, доступного для записи с использованием комбинации масок И, ИЛИ с текущим содержимым регистра для индивидуального сброса или установки бит регистра
	23	Прием и выдача за один запрос значений до 125 смежных регистров, доступных для записи
	43	Сервис инкапсулированного транспорта. Функция 128: обмен со средой разработки CoDeSys Функция 14: выдача расширенной идентификационной информации
Количество регистров	До 4096	
Количество битовых полей	До 65536	

7.2.2. Характеристики клиента MODBUS TCP

Сервис клиента MODBUS TCP реализует функциональность мастера протокола MODBUS TCP. Перечень основных характеристик сервиса приведен в табл. 8.

Таблица 8

Тип узла протокола MODBUS	Мастер	
Скорость обмена, Мбит/с	10, 100 (автоопределение)	
Адресация IP	статическая, по конфигурации контроллера из проекта CoDeSys	
Адрес по умолчанию	10.0.0.1	
Количество серверных соединений по MODBUS TCP	Ограничено ресурсами конфигурационных данных проекта CoDeSys	
Операции протокола MODBUS	*Тип*	*Описание*
	01	Чтение от 1 до 2000 смежных битовых полей, доступных для записи
	02	Чтение от 1 до 2000 смежных битовых полей, доступных для чтения
	03	Чтение от 1 до 125 смежных регистров, доступных для записи
	04	Чтение от 1 до 125 смежных регистров, доступных для чтения
	05	Запись одного битового поля, доступного для записи
	06	Запись значения одного регистра, доступного для записи
	15	Запись значений до 2000 смежных битовых полей, доступных для записи
	16	Запись значений до 125 смежных регистров, доступных для записи
	23	Чтение и чтение за один запрос значений до 125 смежных регистров, доступных для записи

7.2.3. Характеристики сервера MODBUS SERIAL

Сервис сервера MODBUS SERIAL реализует функциональность подчиненного узла протоколов MODBUS RTU, ASCII. Перечень основных характеристик сервиса приведен в табл. 9.

Таблица 9

Тип узла протокола MODBUS	Подчиненный	
Поддерживаемые режимы	**RTU** или ASCII	
Скорость обмена, бит/с	1200, 2400, 9600, 19200, **38400**, 57600, 115200	
Контроль четности	**Even**, Odd, None. Если используется значение None, количество стоп-бит рекомендуется устанавливать равным 2.	
Количество бит данных	7, **8**	
Количество стоповых бит	**1**, 2	
Операции протокола MODBUS	*Тип*	*Описание*
	01	Выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для записи
	02	Выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для чтения
	03	Выдача за один запрос от 1 до 125 смежных регистров, доступных для записи
	04	Выдача за один запрос от 1 до 125 смежных регистров, доступных для чтения
	05	Прием значения одного битового поля, доступного для записи
	06	Прием значения одного регистра, доступного для записи
	15	Прием за один запрос значений до 2000 смежных битовых полей, доступных для записи
	16	Прием за один запрос значений до 125 смежных регистров, доступных для записи
	17	Выдача идентификационной информации устройства
	22	Изменение содержимого заданного регистра, доступного для записи с использованием комбинации масок И, ИЛИ с текущим содержимым регистра для индивидуального сброса или установки бит регистра
	23	Прием и выдача за один запрос значений до 125 смежных регистров, доступных для записи

	43	Сервис инкапсулированного транспорта. Функция 128: обмен со средой разработки CoDeSys Функция 14: выдача расширенной идентификационной информации
Количество регистров	До 4096	
Количество битовых полей	До 65536	
Задержка ответа на запрос	*Скорость, бит/с*	*Номинальное значение (режим RTU), мкс*
	1200	37600
	2400	18700
	9600	5040
	19200	2630
	38400	1470
	57600	1040
	115200	710
Адрес в безопасном режиме	247	

7.2.4. Характеристики клиента MODBUS SERIAL

Сервис клиента MODBUS SERIAL реализует функциональность мастера протоколов MODBUS RTU, ASCII. Перечень основных характеристик сервиса приведен в табл. 10.

Таблица 10

Тип узла протокола MODBUS	Мастер	
Поддерживаемые режимы	**RTU** или ASCII	
Скорость обмена, бит/с	1200, 2400, 9600, 19200, **38400**, 57600, 115200	
Контроль четности	**Even**, Odd, None. Если используется значение None, количество стоп-бит должно быть равным 2	
Количество бит данных	7, **8**	
Количество стоповых бит	**1**, 2	
Операции протокола MODBUS	*Тип*	*Описание*
	01	Чтение от 1 до 2000 смежных битовых полей, доступных для записи
	02	Чтение от 1 до 2000 смежных битовых полей, доступных для чтения
	03	Чтение от 1 до 125 смежных регистров, доступных для записи
	04	Чтение от 1 до 125 смежных регистров, доступных для чтения
	05	Запись одного битового поля, доступного для записи
	06	Запись значения одного регистра, доступного для записи
	15	Запись значений до 2000 смежных битовых полей, доступных для записи
	16	Запись значений до 125 смежных регистров, доступных для записи
	23	Чтение и чтение за один запрос значений до 125 смежных регистров, доступных для записи

7.3. Принцип работы и способы конфигурирования сервера MODBUS

7.3.1. Общие сведения

Сервис сервера MODBUS реализует функциональность подчиненного узла (сервера) протокола и может использоваться для обмена данными между прикладной программой контроллера МК905 и другим устройством, реализующим функции мастера (клиента) сети MODBUS.

Сервис может быть подключен к порту интерфейса RS-232 *COM2* и портам интерфейса RS-422/RS-485 *COM5*, *COM6*, *COM7*, *COM8* контроллера МК905. Сервер MODBUS TCP всегда подключен к портам *Ethernet1* и *Ethernet2*, поскольку в контроллере МК905 обеспечивается возможность одновременной работы клиента и сервера MODBUS TCP.

Для того, чтобы подключить сервис сервера MODBUS RTU/ASCII к аппаратному интерфейсу некоторого последовательного порта, необходимо в окне **PLC Configuration** в конфигурации контроллера для соответствующего порта выбрать опцию *Modbus Serial Slave*, как показано на рис. 29.

В безопасном режиме контроллера сервис сервера MODBUS подключен к коммуникационным интерфейсам: *Ethernet1*, *COM5*.

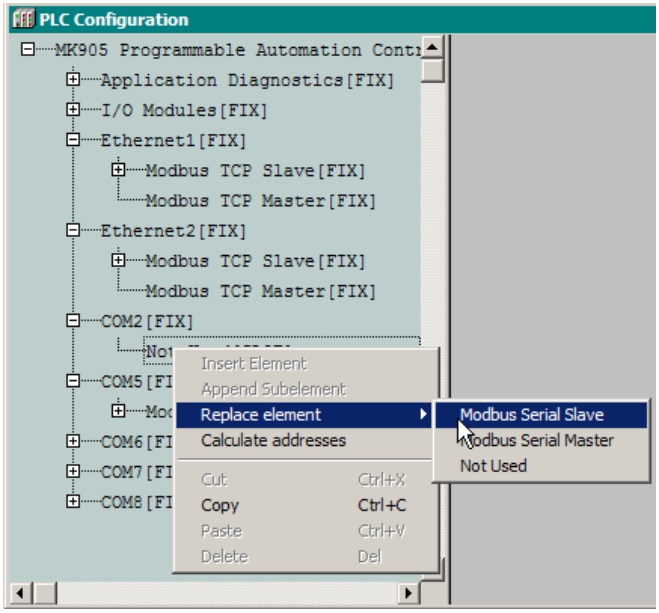


Рис. 29. Пример подключения серверов MODBUS к коммуникационным интерфейсам MK905

Для конфигурирования сервера MODBUS следует:

1. Настроить параметры подчиненного узла сети.
2. Настроить параметры коммуникационных объектов протокола, отображающих области памяти прикладной программы на множества регистров MODBUS.

7.3.2. Параметры протокола сервера MODBUS SERIAL

Перечень параметров протокола MODBUS SERIAL приведен в табл. 11.

Таблица 11

Параметр	Назначение
Адрес узла	Адрес подчиненного узла в диапазоне от 1 до 247. В безопасном режиме и при отсутствии прикладной программы – 247
Тип протокола	Режим протокола MODBUS: RTU или ASCII. В безопасном режиме, и при отсутствии прикладной программы RTU.
Скорость обмена	Скорость обмена по внешней сети из ряда значений: 1200, 2400, 9600, 19200, 38400, 57600, 115200 бит/с. По умолчанию 38400. В принудительном безопасном режиме и при отсутствии прикладной программы – 38400
Кол-во бит данных	Количество бит данных в кадре: 7 или 8. В безопасном режиме и при отсутствии прикладной программы – 8.
Четность	Режим контроля четности бит в кадре: None, Even, Odd. В безопасном режиме и при отсутствии прикладной программы – Even. Если используется значение None, количество стоп-бит должно быть установлено равным 2
Кол-во стоп-битов	Количество стоповых бит в кадре: 1 или 2. В безопасном режиме и при отсутствии прикладной программы – 1. Если для параметра Parity используется значение None, количество стоп-бит должно быть установлено равным 2.

Если контроллер запустился в безопасном режиме по ошибке, его коммуникационные параметры (Режим протокола, скорость обмена, количество бит данных, контроль четности, количество стоп бит) примут значения из последней удачно загруженной конфигурации!

Параметры протокола становятся доступными для редактирования в таблице параметров после щелчка мышью над элементом *Modbus Serial Slave* в древовидном списке конфигурации контроллера.

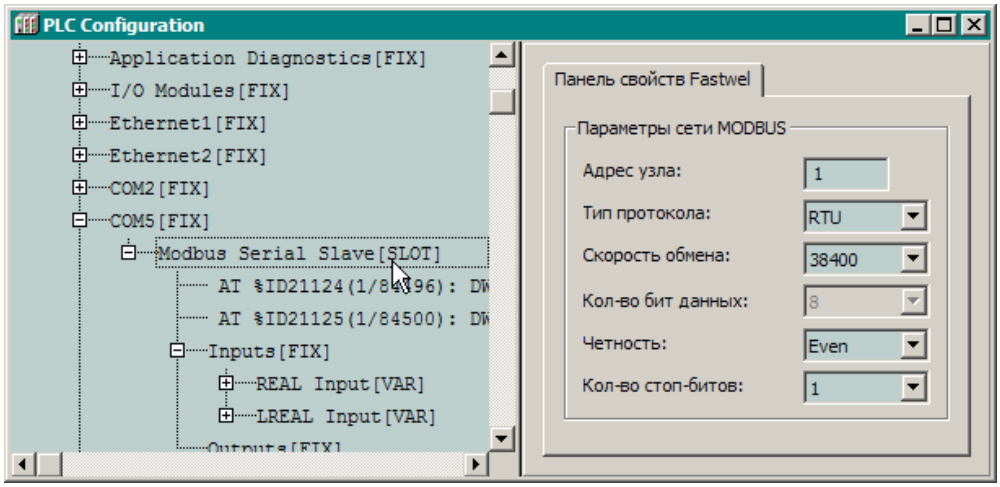


Рис. 30. Конфигурирование параметров протокола сервера MODBUS SERIAL

7.3.3. Параметры протокола сервера MODBUS TCP

IP-параметры Ethernet-портов устанавливаются в панели свойств Fastwel, выводимой при выборе каждого порта в дереве конфигурации контроллера в ресурсе PLC Configuration, как показано на рис. 31. Единственный собственный параметр сервера MODBUS TCP представлен в табл. 12.

Таблица 12

Параметр	Назначение
UnitID	Идентификатор узла. В безопасном режиме и при отсутствии прикладной программы – 0

Если контроллер запустился в безопасном режиме по ошибке, его коммуникационные параметры (IP-адрес, адрес шлюза и маска подсети) примут значения из последней удачно загруженной конфигурации!

Параметры протокола становятся доступными для редактирования в таблице параметров после щелчка мышью над элементом *Modbus TCP Slave* в древовидном списке конфигурации контроллера.

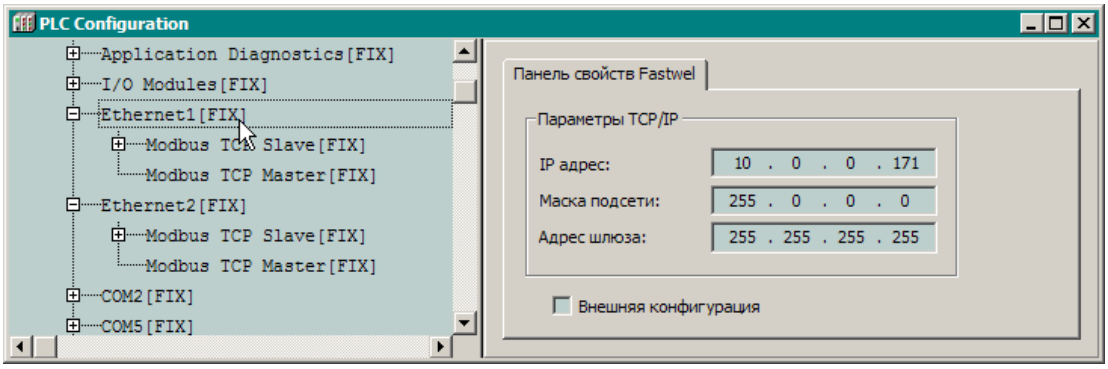


Рис. 31. Конфигурирование параметров протокола сервера MODBUS TCP

7.3.4. Коммуникационные объекты сервера MODBUS

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* содержит список объектов доступа к данным, поступающим по сети от удаленных клиентов. Секция *Outputs* содержит список объектов доступа к данным для передачи в сеть.

Перечень типов и назначение используемых объектов данных приведены в табл. 13.

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. 33.

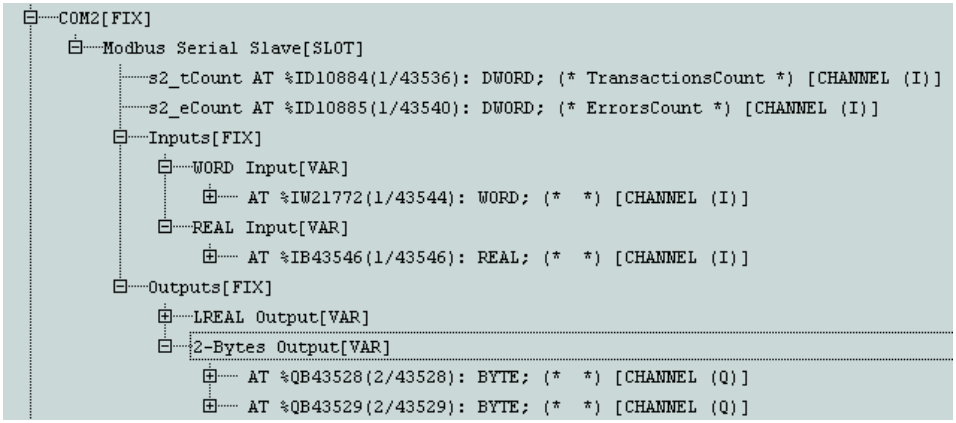


Рис. 32. Конфигурирование коммуникационных объектов сервера MODBUS

Таблица 13

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

Для добавления объекта данных в список сервера щелкните правой кнопкой мыши на позиции соответствующей секции, выберите в контекстном меню команду **Append Subelement** и затем требуемый тип объекта.

Для вставки объекта в список щелкните правой кнопкой мыши на соответствующей позиции в списке объектов данных сервера, выберите в контекстном меню команду **Insert Element** и затем требуемый тип объекта.

Для удаления объекта данных щелкните правой кнопкой мыши на его позиции и выберите в контекстного меню команду **Delete** или выберите его в списке и нажмите клавишу **Del.**

7.3.5. Обмен данными с клиентами MODBUS

Стандартные сетевые операции протокола MODBUS, поддерживаемые сервисом сервера MODBUS, перечислены в п.7.2.1, п.7.2.3 настоящего руководства.

Диапазоны регистровых адресов коммуникационных объектов сервера, регистровый адрес, тип и количество регистров для доступа к данным объектов со стороны клиента (мастера) можно получить в таблице параметров, выполнив следующие действия:

1. Выберите секцию *Inputs* или *Outputs*, которой принадлежит требуемый объект, в древовидном списке конфигурации сервера. Параметры доступа ко всем объектам данной секции будут отображены в таблице **Modbus Access Properties**, как показано на рис. 33.

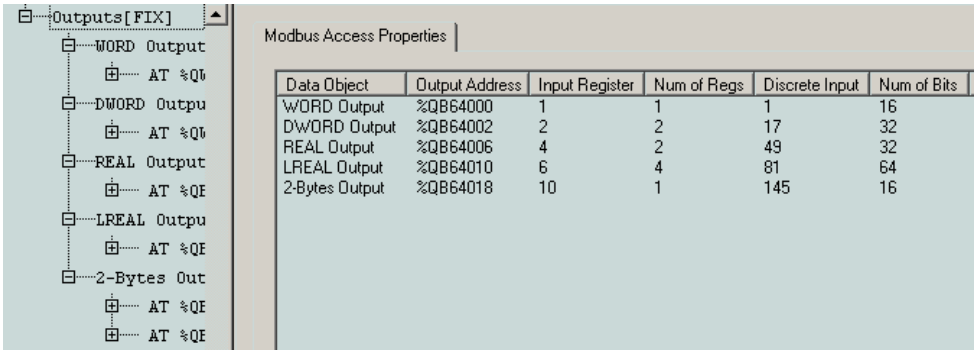


Рис. 33. Список и параметры доступа к выходным объектам данных сервера MODBUS

2. Выберите элемент объекта для отображения параметров доступа к отдельному объекту, как показано на рис. 34.

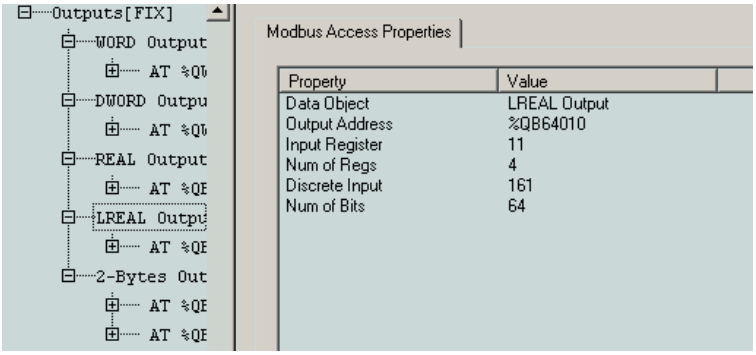


Рис. 34. Параметры доступа к объекту данных сервера MODBUS

Описание назначения отдельных свойств входных и выходных коммуникационных объектов в таблице свойств Modbus Access Properties для секций *Inputs* и *Outputs* приведено в табл. 14.

Таблица 14

Секция Outputs	Объекты данных, передаваемые в сеть мастеру в ответ на запрос чтения	
Столбец	**Наименование**	**Назначение**
Data Object	Тип объекта данных	Название типа объекта данных (см. табл. 13)
Output Address	Начальный адрес в области выходных данных приложения	Смещение в сегменте выходных данных приложения, на которое отображен объект данных
Input Register	Начальный адрес регистра в запросе чтения	Начальный адрес входного регистра (Input Register), используемый для чтения данного объекта или группы объектов, расположенных ниже текущего, по сети. Адрес, передаваемый в запросе MODBUS, должен быть на 1 меньше отображаемого значения.
Num of Regs	Количество регистров в запросе чтения	Количество регистров, которое должно быть указано в сетевом запросе чтения для атомарного получения значения данного объекта с сохранением целостности данных.
Discrete Input	Начальный адрес битового поля в запросе чтения	Начальный адрес битового поля (Discrete Input), доступного для чтения, используемый для чтения данного объекта или группы объектов, расположенных ниже текущего, по сети. Адрес, передаваемый в запросе MODBUS, должен быть на 1 меньше отображаемого значения.
Num of Bits	Количество битовых полей в запросе чтения	Количество битовых полей , которое должно быть указано в сетевом запросе чтения для атомарного получения значения данного объекта с сохранением целостности данных.
Секция Inputs	Объекты данных, получаемые по сети от мастера в запросе записи	
Столбец	**Наименование**	**Назначение**
Data Object	Тип объекта данных	Название типа объекта данных (см. табл. 13)
Input Address	Начальный адрес в области входных данных приложения	Смещение в сегменте входных данных приложения, на которое отображен объект данных
Holding Register	Начальный адрес регистра в запросе записи или чтения	Начальный адрес выходного регистра (Holding Register), используемый для записи или чтения по сети данного объекта или группы объектов, расположенных ниже текущего. Адрес, передаваемый в запросе MODBUS, должен быть на 1 меньше отображаемого значения.
Num of Regs	Количество регистров в запросе записи или чтения	Количество регистров, которое должно быть указано в сетевом запросе записи или чтения для атомарного изменения или получения значения данного объекта с сохранением целостности данных.
Coil	Начальный адрес битового поля в запросе записи или чтения	Начальный адрес битового поля (Coil), доступного для записи или чтения, используемый для записи или чтения данного объекта или группы объектов, расположенных ниже текущего, по сети. Адрес, передаваемый в запросе MODBUS, должен быть на 1 меньше отображаемого значения.
Num of Bits	Количество битовых полей в запросе записи или чтения	Количество битовых полей , которое должно быть указано в сетевом запросе записи или чтения для атомарного изменения или получения значения данного объекта с сохранением целостности данных.

Например, пусть в приложении контроллера требуется иметь 16 переменных типа BOOL и одну переменную типа LREAL, значения которых должны доставляться контроллеру по сети от мастера протокола MODBUS. При этом для оптимизации сетевого траффика значения переменных типа BOOL и LREAL должны поступать в одном сетевом запросе записи (тип 3).

Тогда в конфигурацию сервера MODBUS контроллера могут быть добавлены объекты типа *2-Bytes Input* и *LREAL Input*, как показано на рис. 35.

Мастер сети MODBUS может выполнять запись значений в данные объекты следующими регистровыми операциями:

1. Один запрос записи (тип 16) с начальным адресом регистра 0 и количеством регистров 5. Таким образом будет произведено одновременное изменение всех данных, описываемых указанными двумя объектами.
2. Два отдельных запроса записи регистров: для 2-Bytes Input – с начальным адресом 0 и количеством объектов 1; для LREAL Input – с начальным адресом 1 и количеством объектов 4.

Чтение ранее записанных значений при помощи регистровых операций чтения выполняется аналогично.

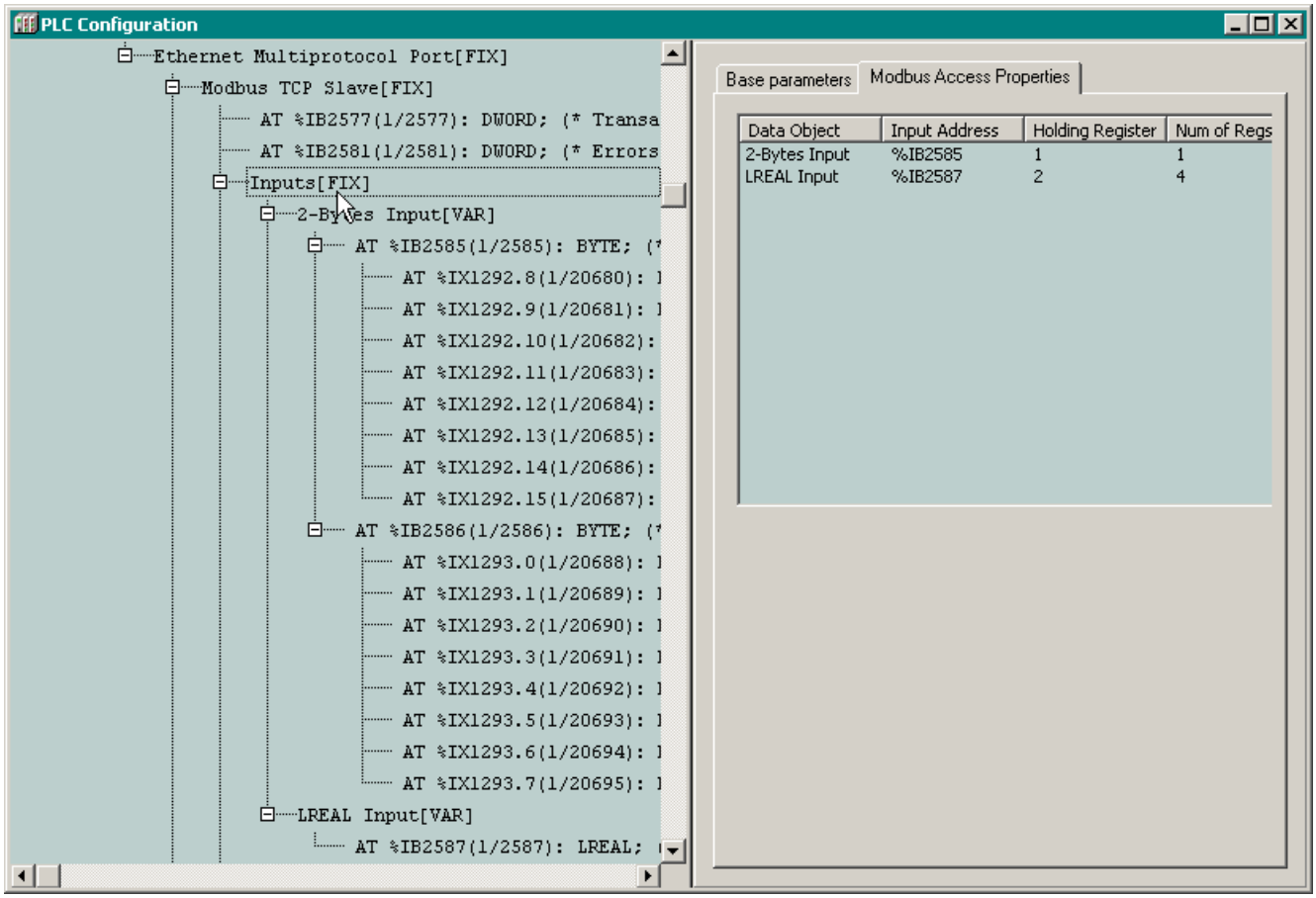


Рис. 35. Пример конфигурации входных объектов сервера

Запись битовых полей объекта *2-Bytes Input* в рассматриваемом примере может выполняться путем запроса записи (тип 15) с адресом 0 и количеством битовых полей 16, либо индивидуально с использованием запроса записи типа 5.

Аналогичным образом выполняется чтение мастером MODBUS объектов, расположенных в области конфигурации *Outputs*.

7.3.6. Формат запроса/ответа на чтение расширенной идентификационной информации

Таблица 15

Поле	Длина, байт	Описание		
Запрос				
Код функции	1	43 (2Bh) – инкапсулированный транспорт		
Тип запроса	1	14 (0Eh) – чтение идентификатора устройства		
Код запроса	1	*Категория запрашиваемой информации*		
		1	базовая информация	
		2	регулярная	
		3	расширенная (не поддерживается)	
		4	доступ к индивидуальным объектам	
Id объекта	1	*Код запрашиваемого объекта*		
		0	Производитель	Fastwel Co.Ltd.
		1	Код продукта	Fastwel MK905 PAC
		2	Код версии продукта	2.59.23936
		3	URL производителя	www.fastwel.com
		4	Наименование продукта	Fastwel MK905 Programmable Automation Controller
		5	Наименование модели	MK905
		6	Имя приложения пользователя	имя файла проекта
Ответ				
Код функции	1	43 (2Bh) – инкапсулированный транспорт		
Тип запроса	1	14 (0Eh) – чтение идентификатора устройства		
Код запроса	1	*Категория запрашиваемой информации*		
		1	базовая информация	
		2	регулярная	
		3	расширенная (не поддерживается)	
		4	доступ к индивидуальным объектам	
Уровень совместимости	1	В соответствии со спецификацией		
Признак продолжения	1	=0 – есть продолжение; =255 – нет продолжения		
Число объектов	1	Количество объектов в списке		
Список объектов				
Id объекта	1	...		
Длина объекта	1	...		
Значение	длина объекта	...		

7.3.7. Обслуживание сетевых запросов

Сервис сервера MODBUS активизируется при возникновении прерывания от коммуникационного порта либо при необходимости передать данные в сеть.

Запрос чтения одного или нескольких регистров приводит к тому, что буферизованные значения, ранее выведенные из сегмента выходных данных приложения в участок образа процесса, на который отображены регистры, упаковываются в ответное сообщение и передаются по сети мастеру.

Запрос записи одного или нескольких регистров приводит к тому, что поступившие значения записываются в участок входной части образа процесса, на который отображены соответствующие регистры.

ВНИМАНИЕ!

Одновременно к серверу протокола MODBUS TCP может быть подключено не более 32-х клиентов. При попытке подключения каждого последующего клиента сервер MODBUS TCP контроллера закрывает соединение с клиентом, активность которого не проявлялась в последние 30 секунд, и использует освободившееся соединение для работы с новым клиентом. Клиент, чье соединение было только что разорвано, должен будет повторно его устанавливать перед очередным MODBUS-запросом. Такое поведение поддерживается не всеми клиентами MODBUS TCP, поэтому не следует превышать максимальное количество одновременно подключенных клиентов.

В соответствии с п. 4.2.1 документа "MODBUS MESSAGING ON TCP/IP. IMPLEMENTATION GUIDE" каждый очередной запрос к MODBUS TCP серверу должен передаваться в отдельном сетевом пакете. Поэтому сервер MODBUS TCP контроллера, приняв очередной пакет в сокет порта 502, извлекает сообщение MODBUS TCP (MODBUS ADU) один раз.

Адреса регистров и битовых полей в сетевых запросах к серверу MODBUS должны быть на единицу меньше отображаемых в диалоговых панелях, показанных на рис. 34 в свойствах Input Register/Dicrete Input и на рис. 35 в свойствах Holding Register/Coil.

7.3.8. Диагностика

Элемент конфигурации Modbus TCP Slave имеет два входных канала типа DWORD, отображающих количество принятых сетевых запросов и количество сетевых запросов, принятых с ошибками. Описание каналов приведено в табл. 16. Обновление значений данных каналов производится системой исполнения контроллера с периодичностью около 1 с.

Таблица 16

Элемент/канал	Тип	Назначение
TransactionsCount	DWORD	Количество принятых входящих сообщений
ErrorsCount	DWORD	Количество входящих сообщений, при приеме которых обнаружены ошибки

При поступлении по сети корректного пакета протокола MODBUS значение на канале *MessagesCount* увеличивается на 1. При обнаружении ошибки в принятом сообщении значение на канале *ErrorsCount* увеличивается на 1.

7.4. Принцип работы и способы конфигурирования клиента MODBUS

7.4.1. Общие сведения

Сервис клиента MODBUS реализует функциональность мастера данного протокола и может использоваться для обмена данными между прикладной программой контроллера MK905 и другими устройствами, реализующими функции подчиненного узла (сервера) сети MODBUS.

Сервис может быть подключен к аппаратным интерфейсам сети Ethernet, порты: *Ethernet1*, *Ethernet2*; и/или к УАПП *COM2*, *COM5*, *COM6*, *COM7*, *COM8* контроллера MK905. Число одновременно и независимо функционирующих клиентов MODBUS на контроллере MK905 может быть от нуля до семи.

Для того, чтобы подключить сервис клиента MODBUS к портам COM2, COM5–COM7, необходимо в конфигурации контроллера для соответствующего порта выбрать опцию *Modbus Serial Master*, как показано на рис. 36.

Сервис клиента MODBUS TCP всегда подключен к портам Ethernet1 и Ethernet2, если контроллер функционирует в нормальном режиме.

В безопасном режиме контроллера сервисы клиента MODBUS отключены от коммуникационных интерфейсов!

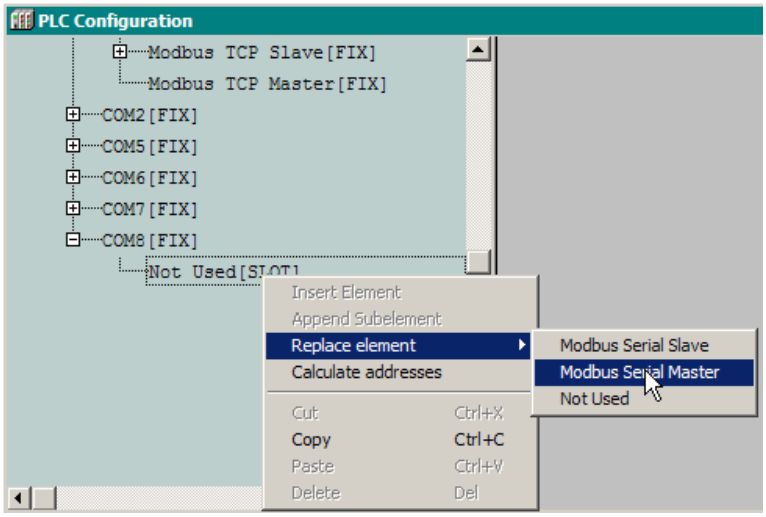


Рис. 36. Пример подключения клиентов MODBUS к коммуникационным интерфейсам MK905

Конфигурирование клиента MODBUS состоит в выполнении следующих действий:

1. Установка параметров узла мастера.

2. Создание списка подчиненных узлов, с которыми мастер должен осуществлять обмен данными и настройка параметров соединений;
3. Создание списков коммуникационных объектов подчиненных узлов, описывающих области считываемых и записываемых данных, и настройка их параметров, а также настройка параметров расписания обмена.

7.4.2. Параметры мастера MODBUS TCP

IP-параметры устанавливаются в панели свойств Fastwel, показанной на рис. 31.

Параметры мастера MODBUS TCP становятся доступными для редактирования в панели свойств Fastwel после щелчка мышью над элементом *Ethernet1(2)– Modbus TCP Master* в древовидном списке конфигурации контроллера, как показано на рис. 37.

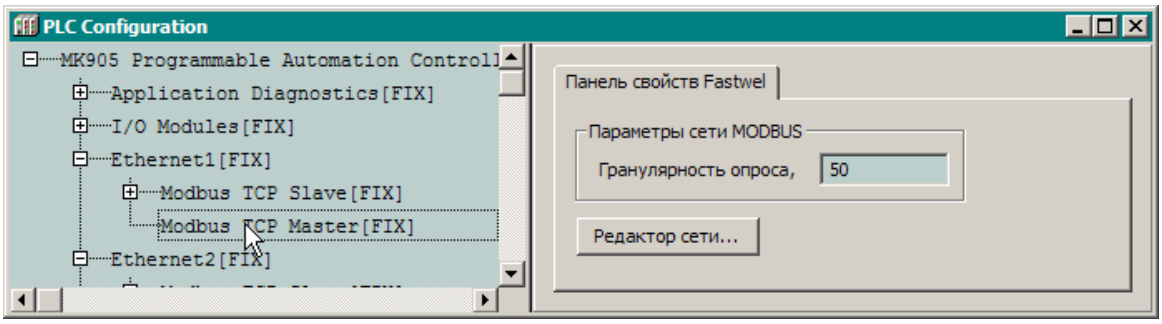


Рис. 37. Параметры мастера MODBUS TCP

Единственный параметр **Гранулярность опроса** определяет временной квант периода обмена с подчиненными узлами (в миллисекундах). Чтение и запись коммуникационных объектов, описанных в конфигурации каждого подчиненного узла данного мастера, будут выполняться с периодами, определяемыми произведением значения в поле **Гранулярность опроса** на значения, заданные для параметров **Тип передачи** коммуникационных объектов.

Для добавления подчиненного узла в список мастера щелкните правой кнопкой мыши на его имени и выберите команду **Append TCP Slave** в появившемся контекстном меню.

Для вставки узла в список щелкните правой кнопкой мыши на соответствующей позиции в списке подчиненных узлов и выберите в контекстном меню команду **Insert TCP Slave**.

Для удаления ранее добавленного подчиненного узла щелкните правой кнопкой мыши на его имени и выберите в контекстном меню команду **Delete** или выберите подчиненный узел в списке и нажмите клавишу **Del**.

Параметры подчиненного узла клиента MODBUS TCP становятся доступными для редактирования в панели свойств Fastwel после щелчка мышью над элементом *TCP Slave* в соответствующей позиции списка конфигурации, как показано на рис. 38.

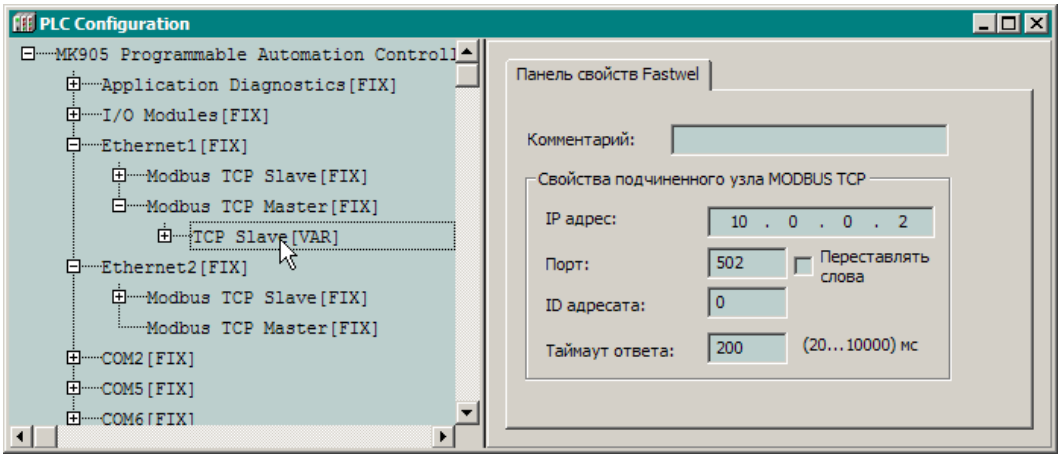


Рис. 38. Параметры подчиненного узла мастера MODBUS TCP

Перечень и назначение параметров подчиненного узла MODBUS TCP приведены в табл. 17.

Поле **Комментарий** позволяет задать текстовую строку, идентифицирующую подчиненный узел в списке подчиненных узлов *Modbus TCP Master*.

Таблица 17

Обозначение	Назначение
IP адрес	IP-адреса подчиненного узла
Порт	TCP-порт сервера MODBUS TCP. По умолчанию 502.
ID адресата	Идентификатор узла
Таймаут ответа	Максимальное время ожидания ответа на запрос
Переставлять слова	Флаг перестановки байт при передаче/приеме слов данных узлу. Клиент использует принятое стандартом Big-Endian представление для адресов и слов данных.. Для серверов, использующих Little-Endian представление, отметить флажок.

7.4.3. Параметры мастера MODBUS SERIAL

Параметры мастера MODBUS SERIAL становятся доступными для редактирования в панели свойств Fastwel после щелчка мышью над элементом *COM2(5,6,7,8) / Modbus Serial Master* в древовидном списке конфигурации контроллера, как показано на рис. 39.

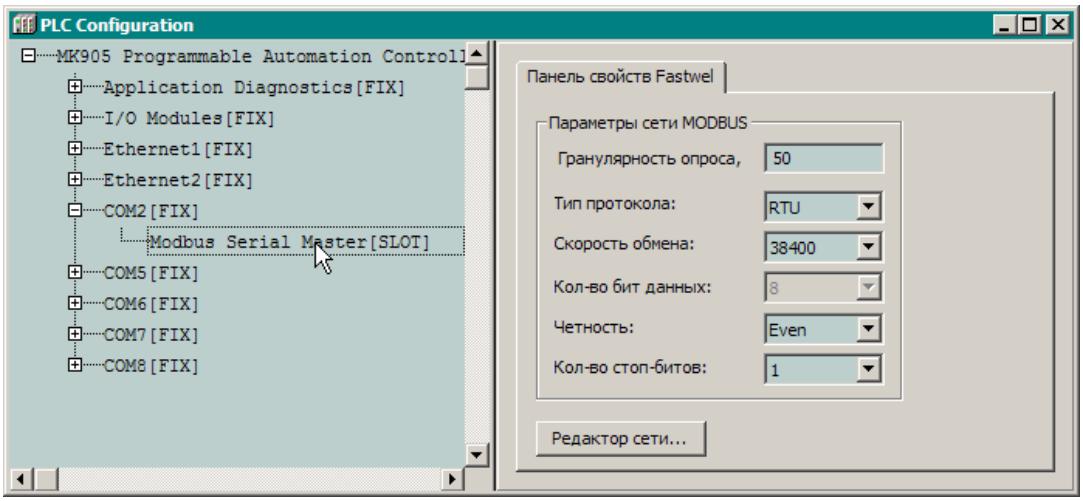


Рис. 39. Параметры мастера MODBUS SERIAL

Перечень и назначение параметров мастера MODBUS SERIAL приведены в табл.18.

Таблица 18

Обозначение	Назначение
Гранулярность опроса, мс	Определяет временной квант периода обмена с подчиненными узлами (в мс). Чтение и запись коммуникационных объектов каждого подчиненного узла, обслуживаемого данным мастером, будут выполняться с периодами, определяемыми произведением значения в поле **Гранулярность опроса** на значения, заданные для параметров **Тип передачи** коммуникационных объектов.
Тип протокола	Режим протокола: RTU или ASCII
Скорость обмена	Скорость обмена, бит/с
Кол-во бит данных	Количество бит данных
Четность	Контроль четности
Кол-во стоп-битов	Количество стоповых бит

Для добавления подчиненного узла в список мастера щелкните правой кнопкой мыши на его имени и выберите команду **Append Serial Slave** в появившемся контекстном меню.

Для вставки узла в список щелкните правой кнопкой мыши на соответствующей позиции в списке подчиненных узлов и выберите в контекстном меню команду **Insert Serial Slave**.

Для удаления ранее добавленного подчиненного узла щелкните правой кнопкой мыши на его имени и выберите в контекстном меню команду **Delete** или выберите подчиненный узел в списке и нажмите клавишу **Del**.

Параметры подчиненного узла клиента MODBUS SERIAL становятся доступными для редактирования в панели свойств Fastwel после щелчка мышью над элементом *Serial Slave* в соответствующей позиции списка конфигурации, как показано на рис. 40.

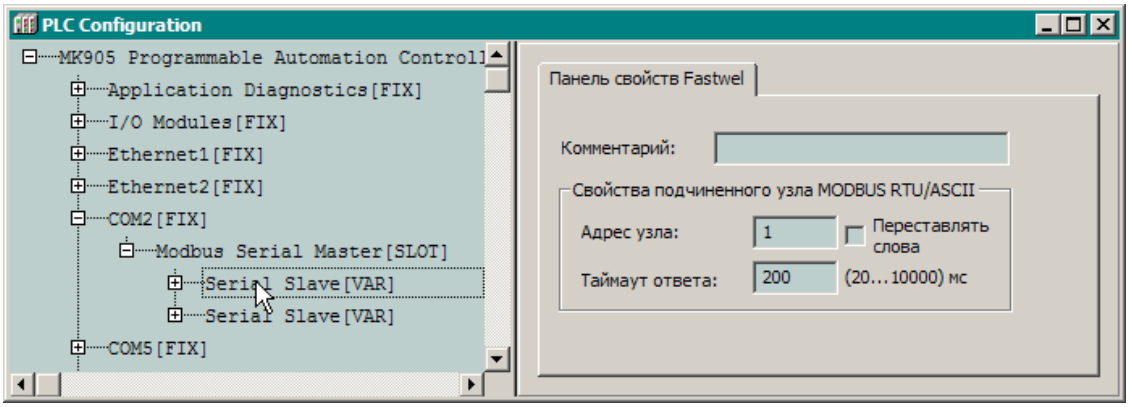


Рис. 40. Параметры подчиненного узла мастера MODBUS SERIAL

Перечень и назначение параметров подчиненного узла MODBUS SERIAL приведены в табл. 19.

Поле **Комментарий** позволяет задать текстовую строку, идентифицирующую подчиненный узел в списке подчиненных узлов *Modbus Serial Master*.

Таблица 19

Обозначение	Назначение
Адрес узла	Идентификатор узла в сети MODBUS
Таймаут ответа	Максимальное время ожидания ответа на запрос (в мс)
Переставлять слова	Флаг перестановки байт при передаче/приеме слов данных узлу. Клиент использует принятое стандартом Big-Endian представление для адресов и слов данных.. Для узлов, использующих Little-Endian представление, необходимо установить флажок.

7.4.4. Обмен данными с подчиненными устройствами

Конфигурирование обмена данными с подчиненным устройством (сервером MODBUS) осуществляется посредством описания его коммуникационных объектов, показанных на рис. 41. Коммуникационный объект определяет области данных на стороне клиента и сервера, расписание и направление операций доступа к данным (чтение/запись/запись-чтение).

Перечень типов и назначение используемых коммуникационных объектов приведены в табл. 20.

Таблица 20

Обозначение	Назначение
Read-Write Coil	Используется для доступа по записи-чтению к данным типа Coil сервера
Read-Write Holding Register	Используется для доступа по записи-чтению к данным типа Holding Registers сервера
Write-Only Coil	Используется для доступа по записи к данным типа Coil сервера
Write-Only Holding Register	Используется для доступа по записи к данным типа Holding Registers сервера
Read-Only Input	Используется для доступа по чтению к данным типа Discrete Input сервера
Read-Only Register	Используется для доступа по чтению к данным типа Input Registers сервера
Read-Only Coil	Используется для доступа по чтению к данным типа Coil сервера
Read-Only Holding Register	Используется для доступа по чтению к данным типа Holding Registers сервера

Для добавления коммуникационного объекта в список сервера щелкните правой кнопкой мыши на позиции подчиненного узла, выберите в контекстном меню команду **Append Subelement** и затем требуемый тип объекта.

Для вставки коммуникационного объекта в список щелкните правой кнопкой мыши на соответствующей позиции в списке коммуникационных объектов сервера, выберите в контекстном меню команду **Insert Element** и затем требуемый тип объекта.

Для удаления коммуникационного объекта щелкните правой кнопкой мыши на его позиции и выберите в контекстном меню команду **Delete** или выберите объект в списке и нажмите клавишу **Del**.

Параметры коммуникационного объекта становятся доступными для редактирования в панели свойств Fastwel после щелчка мышью над его именем, как показано на рис. 41.

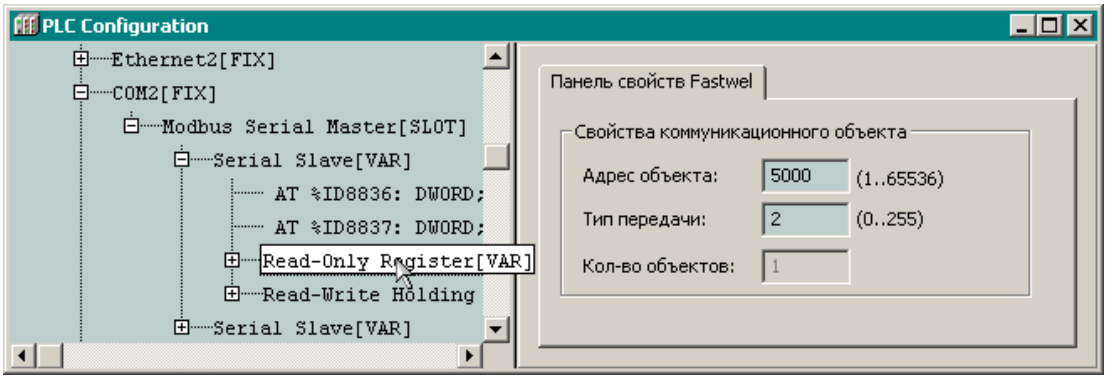


Рис. 41. Коммуникационные объекты мастера MODBUS

Перечень и назначение параметров приведены в табл. 21.

Таблица 21

Обозначение	Назначение	
Адрес объекта	Указывается MODBUS адрес области данных на сервере. **Данное значение должно быть на единицу больше передаваемого в сетевом запросе серверу MODBUS.**	
Тип передачи	0	Указывает на то, что клиент не проводит сетевые операции с данными объекта (не активный объект)
	1...250	Значение N от 1 до 250 указывает на то, что клиент проводит сетевые операции с данными объекта с периодом P = N * To, где To – значение параметра **Гранулярность опроса**, заданного в конфигурации мастера.
	251...254	не используются
	255	Для объектов с доступом только по чтению не используется. Для объектов с доступом по записи указывает на то, что сетевые операции будут производиться по изменению данных, передаваемых серверу

Тип (Read-Only/Write-Only/Read-Write) и адрес коммуникационного объекта определяют адрес области данных (регистр MODBUS) на стороне сервера и тип операции доступа (чтение/запись/запись-чтение). Параметр **Тип передачи** задает расписание обмена.

Область данных со стороны приложения задается списком переменных объекта. Перечень используемых переменных и схема адресации в зависимости от типа объекта приведены в табл. 22.

Таблица 22

Тип объекта	Тип переменной	Область данных переменной
Read-Write Coil	8-Bit Input-Output	Однобайтовый входной канал типа BYTE
		Однобайтовый выходной канал типа BYTE
	16-Bit Input-Output	Двухбайтовый входной канал типа WORD
		Двухбайтовый выходной канал типа WORD
	32-Bit Input-Output	Четырехбайтовый входной канал типа DWORD
		Четырехбайтовый выходной канал типа DWORD
Read-Write Holding Register	WORD Input-Output	Двухбайтовый входной канал типа WORD
		Двухбайтовый выходной канал типа WORD
	DWORD Input-Output	Четырехбайтовый входной канал типа DWORD
		Четырехбайтовый выходной канал типа DWORD
	REAL Input-Output	Четырехбайтовый входной канал типа REAL
		Четырехбайтовый выходной канал типа REAL
	LREAL Input-Output	Восьмибайтовый входной канал типа LREAL
		Восьмибайтовый выходной канал типа LREAL
	2-Bytes Input-Output	2 однобайтовых входных канала типа BYTE
		2 однобайтовых выходных канала типа BYTE
Write-Only Coil	8-Bit Output	Однобайтовый выходной канал типа BYTE
	16-Bit Output	Двухбайтовый выходной канал типа WORD
	32-Bit Output	Четырехбайтовый выходной канал типа DWORD
Write-Only Holding Register	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
Read-Only Input	8-Bit Input	Однобайтовый входной канал типа BYTE
	16-Bit Input	Двухбайтовый входной канал типа WORD
	32-Bit Input	Четырехбайтовый входной канал типа DWORD
Read-Only Register	WORD Input	Двухбайтовый входной канал типа WORD
	DWORD Input	Четырехбайтовый входной канал типа DWORD
	REAL Input	Четырехбайтовый выходной канал типа REAL

Тип объекта	Тип переменной	Область данных переменной
	LREAL Input	Восьмибайтовый входной канал типа LREAL
	2-Bytes Input	2 однобайтовых входных канала типа BYTE
Read-Only Coil	8-Bit Input	Однобайтовый входной канал типа BYTE
	16-Bit Input	Двухбайтовый входной канал типа WORD
	32-Bit Input	Четырехбайтовый входной канал типа DWORD
Read-Only Holding Register	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

Для добавления переменной к коммуникационному объекту щелкните правой кнопкой мыши на его позиции, выберите в контекстном меню команду **Append Subelement** и затем требуемый тип переменной.

Для вставки переменной в список щелкните правой кнопкой мыши на соответствующей позиции в списке переменных коммуникационного объекта, выберите в контекстном меню команду **Insert Element** и затем требуемый тип переменной.

Для удаления переменной щелкните правой кнопкой мыши на ее позиции и выберите в контекстном меню команду **Delete** или выберите ее в списке и нажмите клавишу **Del.**

Сервис клиента MODBUS активизируется при наступлении события, требующего выполнения сетевой операции доступа к данным подчиненного устройства.

Выполнение операции записи в удаленный узел приводит к тому, что буферизованные значения, ранее выведенные из сегмента выходных данных приложения в участок образа процесса, на который отображены переменные коммуникационных объектов, передаются по сети подчиненному узлу.

Выполнение операции чтения из удаленного узла приводит к тому, что поступившие по сети из подчиненного устройства значения записываются в участок входной части образа процесса, на который отображены соответствующие переменные коммуникационного объекта.

Стандартные сетевые операции протокола MODBUS, используемые сервисами клиентов MODBUS TCP, MODBUS SERIAL для выполнения транзакций обмена данными с подчиненными узлами, перечислены в п. 7.2.2, п. 7.2.4 настоящего руководства.

7.4.5. Диагностика

В конфигурации подчиненного узла мастера MODBUS (*TCP Slave/ Serial Slave*) имеется секция, в которой определены два диагностических входных канала, позволяющих приложению во время выполнения получить общее количество инициированных мастером транзакций обмена данными с удаленным узлом и общее количество транзакций завершившихся с ошибкой, табл. 23.

Таблица 23

Элемент/канал	Тип	Назначение
TransactionsCount	DWORD	Количество инициированных мастером транзакций обмена данных с сервером
ErrorsCount	DWORD	Количество транзакций, при выполнении которых обнаружены ошибки

7.5. Инициализация сервисов сети MODBUS

Системное программное обеспечение при включении питания или перезапуске контроллера считывает конфигурацию сервисов сети MODBUS, задает параметры обмена, создает и связывает объекты данных, описания которых имеются в конфигурации. Если в процессе конфигурирования какого либо из сервисов внешней сети произошли критические ошибки, контроллер будет переведен в безопасный режим.

7.6. Доступ к полям данных коммуникационных объектов из приложения

Доступ к данным для передачи в сеть осуществляется посредством выходных переменных программы, ссылающихся на выходные каналы объектов, как показано в табл. 24.

Таблица 24

<table>
<tr><th>Объект выходных данных</th><th colspan="4">Выходные каналы</th><th>Выходные переменные</th></tr>
<tr><td rowspan="8">DWORD Output</td><td>%QX0.0-%QX0.7</td><td>%QB0</td><td rowspan="2">%QW0</td><td rowspan="8">%QD0</td><td rowspan="8">VAR
dwOut AT%QD0 : DWORD;
wOut0 AT%QW1 : WORD;
wOut1 AT%QW1 : WORD;
byteOut0 AT%QB0 : BYTE;
byteOut1 AT%QB1 : BYTE;
byteOut2 AT%QB2 : BYTE;
byteOut3 AT%QB3 : BYTE;
bitOut25 AT%QX1.9 : BOOL;
END_VAR</td></tr>
<tr><td>%QX0.8-%QX0.15</td><td>%QB1</td></tr>
<tr><td>%QX1.0-%QX1.7</td><td>%QB2</td><td rowspan="2">%QW1</td></tr>
<tr><td>%QX1.8-%QX1.15</td><td>%QB3</td></tr>
<tr><td rowspan="8">REAL Output</td><td>%QX2.0-%QX2.7</td><td>%QB4</td><td rowspan="2">%QW2</td><td rowspan="8">%QD1</td><td rowspan="8">VAR
realOut AT%QB4 : REAL;
END_VAR</td></tr>
<tr><td>%QX2.8-%QX2.15</td><td>%QB5</td></tr>
<tr><td>%QX3.0-%QX3.7</td><td>%QB6</td><td rowspan="2">%QW3</td></tr>
<tr><td>%QX3.8-%QX3.15</td><td>%QB7</td></tr>
</table>

Доступ к данным, поступающим по сети, осуществляется через входные переменные приложения, ссылающиеся на каналы объектов данных, как показано в табл. 25.

Таблица 25

<table>
<tr><th>Объект входных данных</th><th colspan="3">Входные каналы</th><th>Входные переменные</th></tr>
<tr><td rowspan="2">2-Bytes Input</td><td>%IX0.0-%IX0.7</td><td>%IB0</td><td rowspan="2">%IW0</td><td rowspan="2">VAR
wIn AT%IW0 : WORD;
byteIn0 AT%IB0 : BYTE;
byteIn1 AT%IB1 : BYTE;
bitIn1 AT%QX0.1 : BOOL;
END_VAR</td></tr>
<tr><td>%IX0.8-%IX0.15</td><td>%IB1</td></tr>
<tr><td rowspan="4">REAL Input</td><td>%IX1.0-%IX1.7</td><td>%IB2</td><td rowspan="2">%IW1</td><td rowspan="4">VAR
realIn AT%IB2 : REAL;
END_VAR</td></tr>
<tr><td>%IX1.8-%IX1.15</td><td>%IB3</td></tr>
<tr><td>%IX2.0-%IX2.7</td><td>%IB4</td><td rowspan="2">%IW2</td></tr>
<tr><td>%IX2.8-%IX2.15</td><td>%IB5</td></tr>
</table>

Имеется возможность создания символических ссылок на каналы объектов данных в ресурсе **PLC Configuration**, как показано на рис. 42.

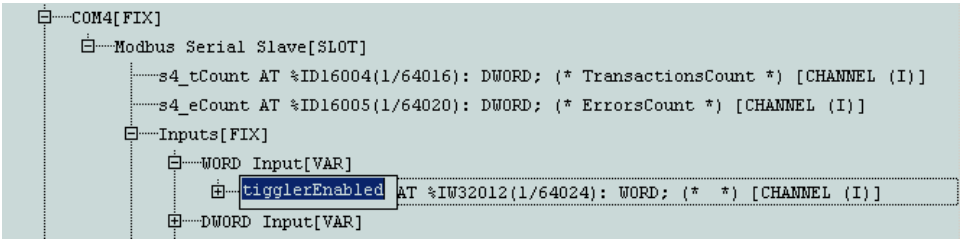


Рис. 42. Создание символических ссылок

ВНИМАНИЕ!

Пусть в конфигурации сервисов внешней сети имеется входной (выходной) объект данных, канал которого ссылается на некоторый адрес в области выходных (входных) данных приложения. При последующей вставке или удалении в конфигурации контроллера описаний секций, имеющих выходные (входные) каналы, канал данного объекта будет ссылаться на другой адрес в области выходных (входных) данных программы, что потребует ручной коррекции адресов выходных (входных) переменных, ссылающихся на канал данного объекта данных.

Данная проблема может быть частично решена путем использования символических ссылок на каналы отдельных каналов регистров.

7.7. Коммуникационные средства верхнего уровня

7.7.1. Общие сведения

CoDeSys Gateway Server предназначен для организации информационного обмена между средой разработки CoDeSys, функционирующей на компьютере, и средой исполнения CoDeSys на удаленном контроллере через интерфейс внешней сети, поддерживаемый контроллером.

Поддержка той или иной сети реализуется при помощи коммуникационных драйверов, с которыми взаимодействует Gateway Server, как показано на рис. 43.

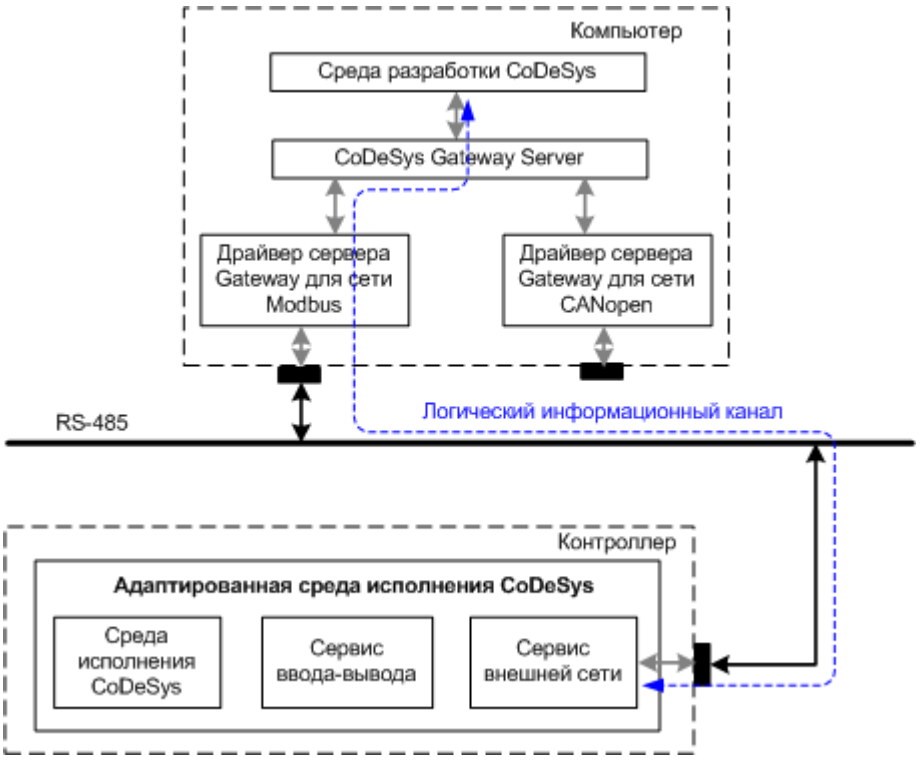


Рис. 43. Архитектура информационного обмена между средой разработки и средой исполнения CoDeSys

Коммуникационный драйвер GDrvFastwel.dll имеет встроенную поддержку протоколов MODBUS over Serial Line и MODBUS TCP.

При установлении логического информационного канала между средой разработки и удаленным контроллером по сети MODBUS используется интерфейс инкапсуляции транспорта протокола MODBUS (Encapsulated Interface Transport, код функции 2Bh, тип 80h).

Настоящий подраздел содержит указания по настройке параметров коммуникационного драйвера GDrvFastwel.dll, поставляемого в комплекте с адаптированной средой CoDeSys для Fastwel I/O и MK905.

7.7.2. Установка коммуникационного драйвера CoDeSys Gateway Server

Коммуникационный драйвер CoDeSys Gateway Server включает в себя два компонента (GDrvFastwel.dll и modbusDLL.dll), автоматически устанавливаемые в подкаталог \System32 каталога установки Windows в процессе установки пакета адаптации среды CoDeSys для Fastwel I/O.

7.7.3. Создание логического информационного канала между средой разработки и контроллером по протоколу MODBUS TCP

Логический информационный канал между средой разработки и средой исполнения CoDeSys на удаленном контроллере служит для выполнения операций по сети, инициируемых из меню **Online** среды разработки CoDeSys.

Установление соединения между средой разработки CoDeSys и MK905 по протоколу MODBUS TCP возможно в случае, если в конфигурации контроллера для соответствующего порта Ethernet заданы, как минимум, IP-адрес и маска подсети.

Для создания информационного канала выполните следующие действия:

1. Запустите среду разработки CoDeSys.
2. Выберите команду меню **Online–Communication Parameters**… На экран будет выведена диалоговая панель **Communication Parameters**, показанная на рис. 44.

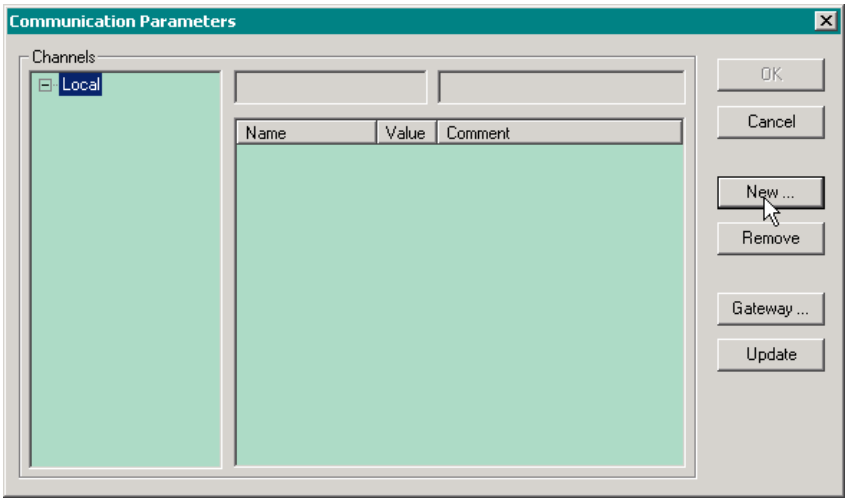


Рис. 44. Диалоговая панель параметров коммуникационного сервера связи с удаленным контроллером

3. Для создания логического информационного канала нажмите кнопку **New** и в появившейся диалоговой панели введите имя создаваемого канала, а в списке **Device** выберите строку *ModbusTCP: Fastwel Modbus TCP driver*, как показано на рис. 45 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. 46.

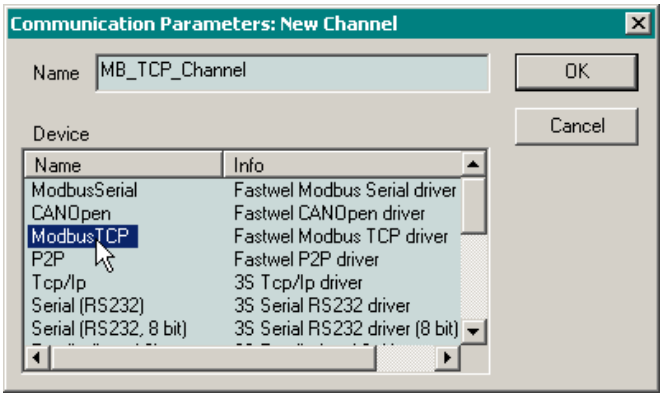


Рис. 45. Создание канала с использованием драйвера Fastwel MODBUS driver

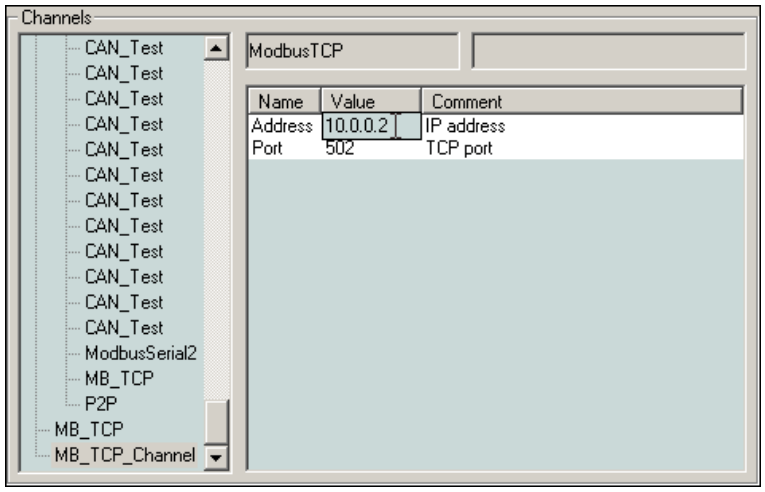


Рис. 46. Настройка параметров канала

4. Дважды щелкните на значении параметра *Address* и введите значение IP-адреса подчиненного узла, с которым предполагается установить связь.
5. Закройте диалоговую панель **Communication Parameters** нажатием кнопки **OK**.

7.7.4. Создание логического информационного канала между средой разработки и контроллером по протоколу MODBUS Serial

Установление соединения между средой разработки CoDeSys и MK905 по протоколу MODBUS RTU/ASCII возможно в случае, если в конфигурации контроллера для соответствующего последовательного порта (COM2, COM5–COM8) выбран протокол *Modbus Serial Slave*.

Для создания информационного канала выполните следующие действия:

1. Запустите среду разработки CoDeSys.
2. Выберите команду меню **Online–Communication Parameters**… На экран будет выведена диалоговая панель **Communication Parameters**, показанная на рис. 47.

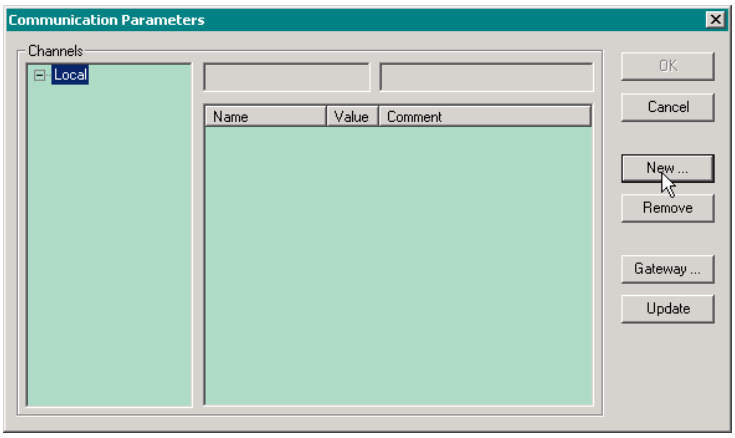


Рис. 47. Диалоговая панель параметров коммуникационного сервера связи с удаленным контроллером

3. Для создания логического информационного канала нажмите кнопку **New** и в появившейся диалоговой панели введите имя создаваемого канала, а в списке **Device** выберите строку *Modbus: Fastwel Modbus driver*, как показано на рис. 48 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. 49.

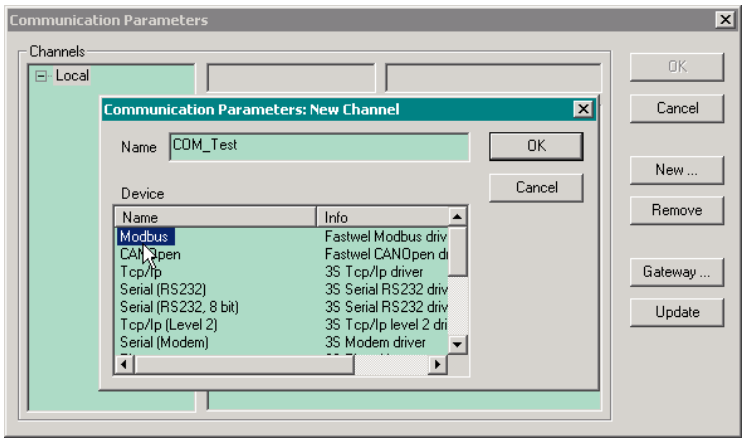


Рис. 48. Создание канала с использованием драйвера Fastwel MODBUS driver

4. 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 по сети MODBUS, и нажмите клавишу Enter.
5. Дважды щелкните на значении параметра *Parity* и клавишей ↑ ("стрелка вверх") установите требуемый вид контроля по четности.

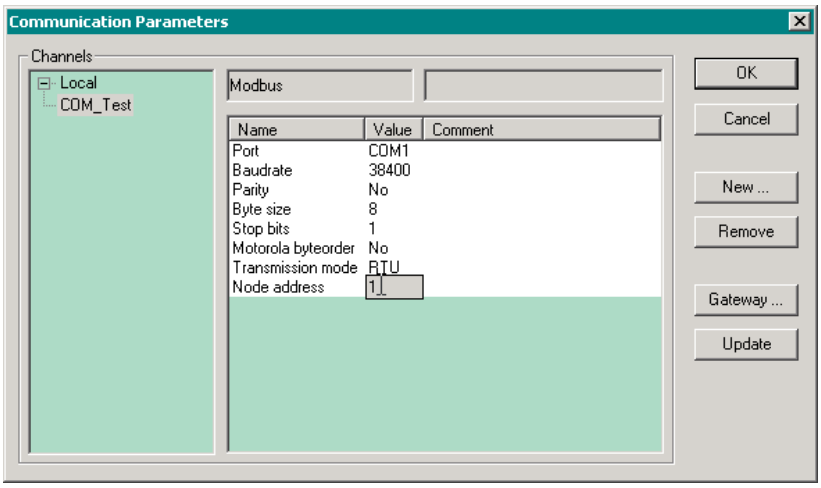


Рис. 49. Настройка параметров канала

6. Дважды щелкните на значении параметра *Node address* и клавишами ↑ ("стрелка вверх") или ↓ ("стрелка вниз") установите значение адреса подчиненного узла, с которым предполагается установить связь.
7. Закройте диалоговую панель **Communication Parameters** нажатием кнопки **OK**.

7.7.5. Создание логического информационного канала между средой разработки и контроллером по последовательному каналу связи (P2P)

Для создания информационного канала выполните следующие действия:

1. Соедините последовательный порт ПК с портом COM1 контроллера.
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. 50, 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. 51.

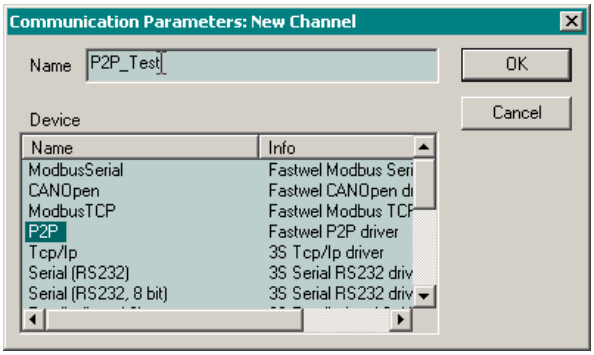


Рис. 50. Создание канала с использованием драйвера Fastwel P2P driver

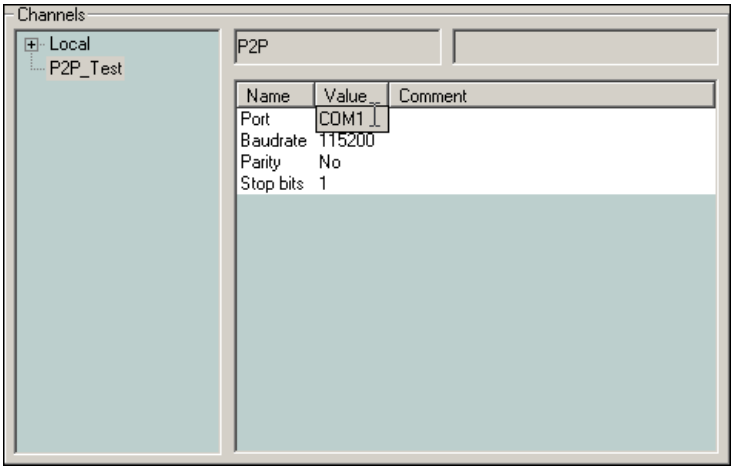


Рис. 51. Настройка параметров канала

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**

ВНИМАНИЕ!

При наличии интенсивного обмена через коммуникационные порты COM5–COM8 загрузка приложения и файлов через порт P2P (COM1) контроллера может стать невозможной. Для решения данной проблемы загружайте приложение или файлы через один из интерфейсов Ethernet.

7.7.6. Дополнительные замечания

ВНИМАНИЕ!

Иногда после копирования файла проекта CoDeSys с расширением *.pro с одного компьютера на другой или после удаления вспомогательных файлов проекта при попытке выполнить команду **Online–Login** среда разработки CoDeSys выводит на экран монитора сообщение, показанное на рис. 54.

Для решения данной проблемы нажмите кнопку **Gateway...** в диалоговой панели **Communication Parameters**, после чего в появившейся диалоговой панели **Communication Parameters: Gateway** установите опцию **Connection** : *Local*, затем нажмите **OK**, и всё будет в порядке.

Рис. 52. Сообщение о невозможности установить соединение с удаленным сервером Gateway

8. Дополнительные функциональные возможности

8.1. Общие сведения

Контроллер функционирует под управлением операционной системы Windows CE 5.0, в состав которой включены следующие дополнительные функциональные возможности:

1. Драйвер USB-накопителей
2. Драйвер мыши и клавиатуры с интерфейсом USB
3. Сервер протокола FTP
4. Сервер и клиент службы времени SNTP.

Для получения подробной информации о настройке и использовании перечисленных функциональных возможностей обратитесь к документации на Windows CE 5.0.

В настоящем разделе приведена информация о копировании файлов между контроллером и инструментальным компьютером (далее – ПК) с использованием дополнительных средств.

Копирование файлов может быть осуществлено следующими способами:

1. При помощи USB-накопителя, подключаемого к одному из гнезд контроллера.
2. По протоколу ftp при помощи любого клиента ftp, например, встроенного в утилиту управления файлами Total Commander (но почему-то кроме встроенного в браузер Mozilla Firefox).

8.2. Копирование файлов по протоколу ftp

Для установления соединения с MK905 по протоколу ftp требуется предварительно задать пароль сетевого подключения к MK905:

1. Отключите питание MK905.
2. Подключите к MK905 монитор, клавиатуру и мышь. Клавиатура и мышь могут быть подключены к незанятым портам USB.
3. Включите питание MK905.
4. По истечении около 15 секунд на экране монитора, подключенного к MK905, появится рабочий стол Windows CE 5.0 с отключенной панелью задач.
5. На клавиатуре нажмите клавишу Windows () и в появившемся меню выберите команду **Settings**. На экран монитора будет выведено окно **Control Panel**.
6. Дважды щелкните на ярлыке **Password**. В появившейся диалоговой панели **Password Properties** задайте пароль в полях **Password** и **Confirm Password**, после чего закройте **Password Properties** нажатием кнопки **OK**.

В дальнейшем для соединения с контроллером по протоколу ftp следует использовать имя пользователя *FastwelMK905-CDS* и пароль, введенный при выполнении приведенных выше указаний.

Остальные параметры соединения с контроллером по протоколу ftp при помощи утилиты Total Commander показаны на рис. 53. В конкретном случае может потребоваться ввести IP-адрес, соответствующий заданному в свойствах порта Ethernet1 или Ethernet2 конфигурации проекта, загруженного в контроллер.

При использовании *FixedDisk1* в качестве значения параметра **Remote Dir:** после успешного подключения к MK905 по протоколу ftp в одной из панелей Total Commander будет отображен список файлов и каталогов встроенного NAND Flash диска контроллера.

При использовании *FixedDisk2* в качестве значения параметра **Remote Dir:** будет выполнено подключение к накопителю, установленному в гнездо CompactFlash контроллера.

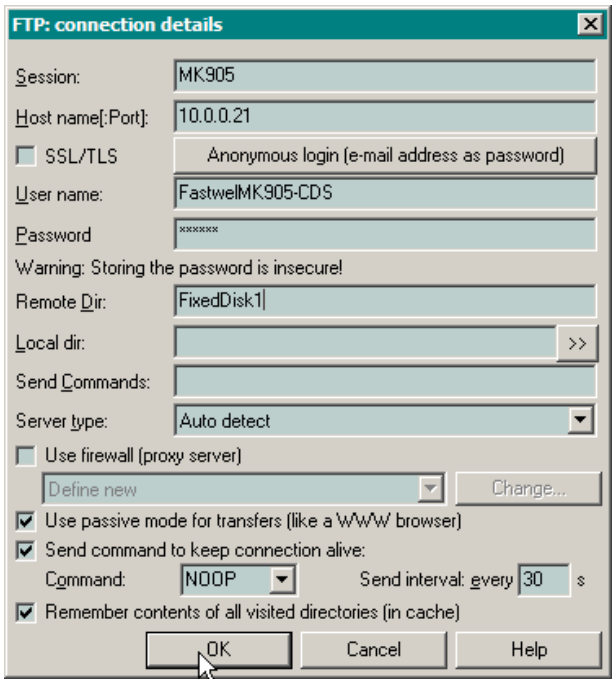


Рис. 53. Параметры соединения по протоколу ftp утилитой Total Commander

Для выгрузки файлов из контроллера по протоколу ftp при помощи браузера Internet Explorer введите в адресной строке префикс протокола ftp, после него IP-адрес и нажмите Enter. Например:

`ftp://10.0.0.251`

После ввода имени пользователя и пароля в окне браузера появится содержимое корневого каталога файловой системы контроллера. Перейдите в требуемый каталог, щелкните правой кнопкой мыши над требуемыми файлами и выберите команду Copy.

8.3. Перенос файлов на USB-накопителе

8.3.1. Локальное копирование

Если в распоряжении пользователя имеются USB-мышь и видеомонитор, то копирование файлов с подключаемого к контроллеру USB-накопителя или на USB-накопитель может быть выполнено практически в точности так же, как это делается на обычном компьютере:

1. Подключите к USB-гнездам контроллера мышь и клавиатуру с интерфейсом USB.
2. Подключите к контроллеру видеомонитор разрешением не хуже 800×600.
3. Включите питание монитора и нажмите клавишу Windows на клавиатуре, подключенной к контроллеру, после чего выберите **Programs–Windows Explorer** в появившемся меню. На экран монитора, подключенного к контроллеру, будет выведено окно Pocket Windows Explorer, показанное на рис. 54.
4. Присоедините USB-накопитель к контроллеру и убедитесь, что через 3-5 секунд в окне Windows Explorer появился ярлык *USB Storage*, как показано на рис. 54.
5. Скопируйте требуемые файлы с USB-накопителя в пользовательский специальный каталог *\FixedDisk1\user* или наоборот, пользуясь традиционными приемами, принятыми в пользовательском интерфейсе операционной системы Windows.

8.3.2. Удаление файлов

Для удаления файлов с накопителей дисков FixedDisk1 или FixedDisk2:

1. Подключите к USB-гнездам контроллера мышь и клавиатуру с интерфейсом USB.
2. Подключите к контроллеру видеомонитор разрешением не хуже 800×600.
3. Включите питание монитора и нажмите клавишу Windows на клавиатуре, подключенной к контроллеру, после чего выберите **Programs–Windows Explorer** в

появившемся меню. На экран монитора, подключенного к контроллеру, будет выведено окно Pocket Windows Explorer, показанное на рис. 54.

4. Дважды щелкните на требуемом диске (FixedDisk1 или FixedDisk2), щелкните правой кнопкой мыши на файле, подлежащему удалению, после чего выберите команду Delete в появившемся контекстном меню.

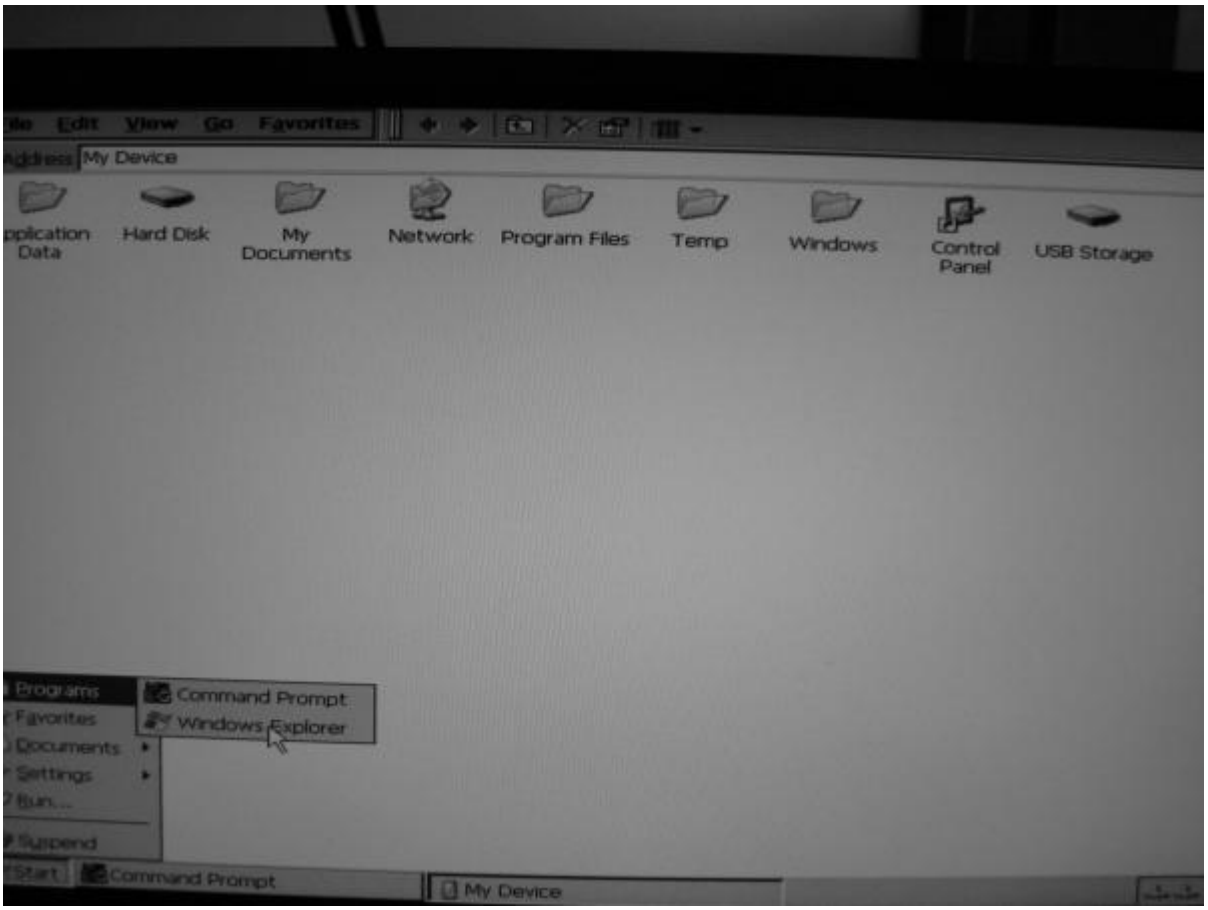


Рис. 54. Содержимое окна Windows Explorer

8.3.3. Копирование без использования монитора и клавиатуры

1. Подключите USB-накопитель к одному из гнезд контроллера.
2. Установите соединение между инструментальным ПК и контроллером по протоколу ftp в соответствии с указаниями п. 8.2. При установлении соединения по протоколу ftp в качестве начального каталога используйте *\USB Storage* (для второго USB-накопителя – *\USB Storage2*).
3. Скопируйте требуемые файлы на диск инструментального ПК или наоборот.

9. Подсистема целевой визуализации

9.1. Общие сведения

Настоящий раздел содержит информацию об особенностях применения подсистемы целевой визуализации CoDeSys, поддержка которой реализована в контроллере MK905.

Подробная информация об использовании целевой визуализации приведена в документе «Визуализация CoDeSys. Дополнение к руководству пользователя по программированию ПЛК в CoDeSys 2.3» (CoDeSys_Visu_V23_RU.pdf), который находится в подкаталоге документации пакета адаптации CoDeSys для Fastwel I/O. При установке пакета адаптации с параметрами по умолчанию местоположение документа:

C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Doc\CoDeSys\Russian\ CoDeSys_Visu_V23_RU.pdf

Кроме того, документ может быть загружен по адресу:

ftp://ftp.prosoft.ru/pub/hardware/Fastwel/Fastwel_IO/Version2/Doc/CoDeSys/Russian/CoDeSys_Visu_V23_RU.pdf

Для того, чтобы в приложении, разрабатываемом для контроллера MK905, появилась возможность использования функций целевой визуализации, в свойствах целевой платформы *Fastwel MK905 Programmable Automation Controller* создаваемого приложения во вкладке **Visualization** должен быть установлен флажок **Target Visualization**. При необходимости использования архивации значений переменных, отображаемых элементом Тренд, следует также отметить флажок **Store trend data in PLC**.

Для выяснения текущей версии системного программного обеспечения контроллера:

1. В среде разработки CoDeSys откройте проект (файл с расширением *.pro), посредством которого была получена текущая пользовательская программа контроллера. Если в контроллере отсутствует исполняющееся приложение, создайте новый проект в соответствии с указаниями п. 5.2 настоящего руководства.
2. Если контроллер подключен к сети (MODBUS или MODBUS TCP, в зависимости от типа контроллера), установите соединение между средой разработки CoDeSys и контроллером через соответствующий коммуникационный канал согласно указаниям п. 7.7 путем выполнения команды **Online–Login** в среде разработки CoDeSys.

   Если контроллер не подключен к сети, подключите коммуникационный порт COM1 контроллера к последовательному порту компьютера, на который установлена среда разработки CoDeSys, при помощи нуль-модемного кабеля последовательной связи, после чего установите соединение между средой разработки CoDeSys и контроллером через коммуникационный канал P2P.
3. Если после выполнения команды **Online-Login** на экране монитора появится диалоговая панель *The program has changed...*, нажмите в ней кнопку **Details**. Диалоговая панель CoDeSys примет вид, показанный на рис. 55.
4. Номер версии системного программного обеспечения, загруженного в контроллер, отображается в скобках после префикса *FW:* в поле **Project in PLC – Version**, как показано на рис. 55.

В случае необходимости обновление системного программного обеспечения контроллера выполнять в соответствии с указаниями п. 3.6 настоящего руководства.

Наиболее актуальная версия пакета адаптации CoDeSys 2.3 для Fastwel I/O доступна по адресу:

ftp://ftp.prosoft.ru/pub/hardware/Fastwel/Fastwel_IO/Version2/Setup/

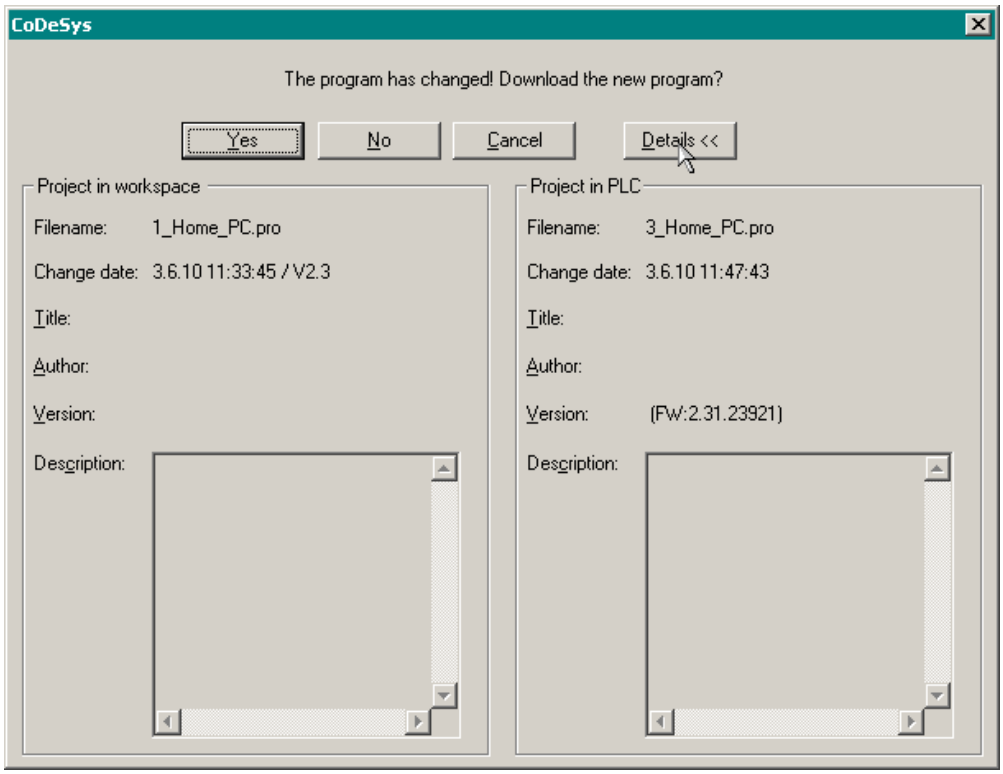


Рис. 55. Просмотр информации о версии системного ПО контроллера

9.2. Функциональные возможности и характеристики

9.2.1. Функциональные возможности целевой визуализации MK905

В подсистеме целевой визуализации MK905 реализованы следующие функциональные возможности:

1. Отображение базовых графических примитивов визуализации CoDeSys, включая перечисленные ниже:

   прямоугольник, эллипс, прямоугольник со скругленными углами;

   линия, полигон, ломаная линия, кривая;

   содержимое файлов в формате bmp и jpeg.

2. Отображение составных графических примитивов визуализации CoDeSys, включая перечисленные ниже:

   кнопка;

   столбчатый индикатор;

   стрелочный индикатор;

   ссылка на другую форму визуализации;

   тренд.

3. Отображение следующих типов анимации графических элементов:

   динамический текст;

   изменение цвета;

   изменение видимости;

   перемещение;

   вращение;

   масштабирование;

   смещение отдельных граней;

активизация/блокировка кнопок.

4. Реакция на действия оператора/пользователя, включая:

   кратковременное переключение (tap) и изменение (toggle) значений булевых переменных;

   ввод значений числовых и строковых переменных;

   переключение между формами (окнами) визуализации;

   переключение языка интерфейса оператора.

5. Выполнение следующих специальных функций:

   отображение всплывающих подсказок над графическими примитивами;

   отображение фоновых рисунков в формате bmp и jpeg в формах визуализации и кнопках;

   архивация и прокрутка истории значений переменных, определенных для элемента отображения типа Тренд;

   исполнение программируемых выражений.

В подсистеме визуализации контроллера MK905 реализованы следующие функциональные возможности, изначально не предусмотренные в среде исполнения CoDeSys:

1. Отображение значений переменных типа STRING приложения на русском языке.
2. Ввод текста на русском языке для переменных типа STRING приложения.
3. Переключение раскладки диалога клавишной панели (Keypad) на русскую.

В подсистеме визуализации контроллера MK905 не поддерживается отображение элементов управления ActiveX.

9.2.2. Основные характеристики целевой визуализации MK905

Параметр	**Единица**	**Количество**
Количество одновременно отображаемых окон форм визуализации	шт	1
Максимальная длина строки	символов	256
Максимальная длина строки для всплывающей подсказки	символов	1024
Максимальное количество обновляемых регионов	шт	2000
Максимальное количество растровых изображений	шт	1024
Максимальное количество полигонов	шт	200
Максимальное количество вершин полигона	шт	512
Максимальное количество прямоугольников	шт	200
Максимальное количество запоминаемых перемещений курсора	шт	25
Максимальное количество запоминаемых нажатий клавиш	шт	30
Максимальное количество запоминаемых щелчков мыши	шт	2
Минимальный период задачи контроля пользовательского ввода VISU_INPUT_TASK	мс	50
Минимальный период задачи визуализации VISU_TASK	мс	100
Минимальный период задачи архивации TREND_TASK	мс	200

Видеоподсистема контроллера MK905 поддерживает следующие видеорежимы:

640×480 (60 Гц, 75 Гц); 800×600 (60 Гц, 75 Гц); 1024×768 (60 Гц, 75 Гц); 1152×864 (60 Гц, 75 Гц); 1280×1024 (60 Гц, 75 Гц).

9.3. Принцип работы подсистемы целевой визуализации

9.3.1. Общие сведения

Среда разработки CoDeSys 2.3 обеспечивает возможность создания графических мнемосхем оператора (далее – форм визуализации) в проектах приложений для контроллеров, среда исполнения которых поддерживает функциональность целевой визуализации (Target Visualization).

Функциональность целевой визуализации в контроллере реализуется за счет наличия в его среде исполнения поддержки системной библиотеки SysLibTargetVisu.lib, содержащей функции отображения графических примитивов, растровых изображений, текста, обнаружения событий, связанных с действиями оператора и т.д.

По умолчанию при создании проекта для контроллера MK905 функциональность целевой визуализации отключена. Для включения поддержки целевой визуализации в проекте CoDeSys для MK905:

1. Выберите вкладку **Resources** в левой области главного окна CoDeSys и дважды щелкните на ресурсе **Target Settings**.
2. В появившейся диалоговой панели **Target Settings** щелкните на вкладке **Visualization** и отметьте флажок **Target visualization**, как показано на рис. 56. При необходимости использования архивации значений переменных, отображаемых элементом Тренд, следует также отметить флажок **Store trend data in PLC**.
3. В полях **Display width in pixel** и **Display height in pixel** задайте разрешение экрана монитора, который предполагается подключить к контроллеру. Видеоподсистема MK905 поддерживает следующие разрешения экрана: 640×480 (60 Гц, 75 Гц); 800×600 (60 Гц, 75 Гц); 1024×768 (60 Гц, 75 Гц); 1152×864 (60 Гц, 75 Гц); 1280×1024 (60 Гц, 75 Гц). По умолчанию при поставке для видеоподсистемы MK905 установлено разрешение 800×600. Для изменения разрешения следуйте указаниям п. 9.4.1.2 настоящего документа.
4. Закройте диалоговую панель **Target Settings** нажатием кнопки **OK**.

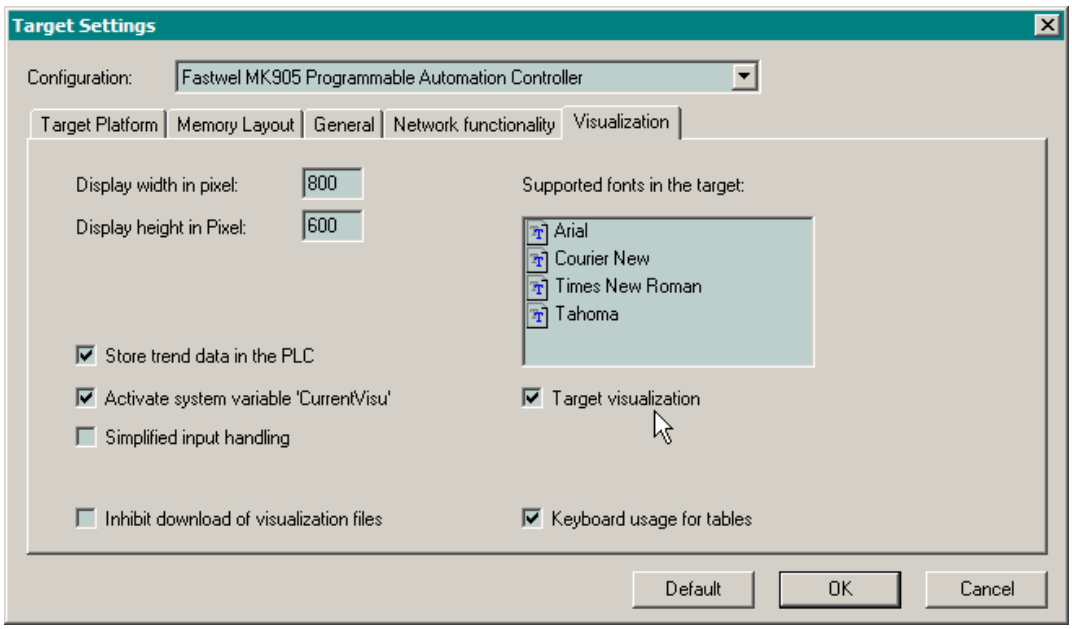


Рис. 56. Активизация функциональности целевой визуализации MK905

Разработка форм визуализации в проекте выполняется в соответствии с указаниями документа Визуализация CoDeSys. Дополнение к руководству пользователя по программированию ПЛК в CoDeSys 2.3. Подсистема целевой визуализации MK905 поддерживает все функциональные возможности целевой визуализации, исключая отображение элементов управления ActiveX, Web visualization и обработку тревог (**Alarmhandling in the PLC**).

При выполнении команды **Project–Build** или **Project–Rebuild All** среда разработки CoDeSys автоматически генерирует код, который после загрузки в контроллер будет выполнять отображение созданных в проекте форм визуализации и обработку действий пользователя (нажатие клавиш

клавиатуры, щелчки кнопками мыши и т.п.) путем вызовов функций библиотеки SysLibTargetVisu.lib, которая автоматически подключается к проекту, как только пользователь активизирует в нем функциональность целевой визуализации. Кроме автоматического подключения библиотеки SysLibTargetVisu.lib, в проект будут автоматически добавлены три задачи:

1. VISU_TASK – данная циклическая задача выполняет рендеринг текущей формы визуализации. По умолчанию данная задача получает приоритет 15 и период 200 мс.
2. VISU_INPUT_TASK – данная циклическая задача фиксирует действия оператора, включая нажатия и отпускания клавиш клавиатуры, кнопок мыши и перемещения курсора. По умолчанию данная задача получает приоритет 14 и период 50 мс.
3. TREND_TASK – если в приложении используется архивация значений переменных (отмечен флажок **Target Setting–Store trend data in PLC**), то данная циклическая задача выполняет сбор значений архивируемых переменных, формирование записей для сохранения в файлах архива и вызов функций архивирования, которые реализованы в библиотеке SysLibAlarmTrend.lib. По умолчанию данная задача получает приоритет 13 и период 200 мс.

Не рекомендуется изменять периоды задач VISU_TASK и VISU_INPUT_TASK, поскольку это может отрицательно сказаться на общей производительности системы.

В среде исполнения контроллера и в пакете адаптации CoDeSys для MK905 количество приоритетов циклических задач увеличено до 32-х.

9.3.2. Функционирование в нормальном режиме

Если в контроллер загружено приложение, в котором отсутствуют формы визуализации или для которого не была активизирована поддержка целевой визуализации, на экране монитора контроллера отображается окно, показанное на рис. 57 и далее называемое окном консоли контроллера.

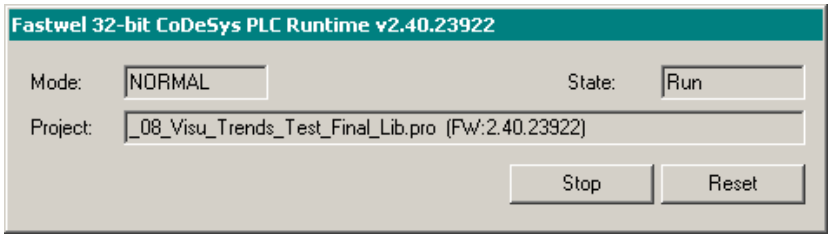


Рис. 57. Окно консоли ПЛК в нормальном режиме

Поле **Mode** содержит название текущего режима работы – *NORMAL* (нормальный, см. п. 4.2.1.2).

В поле **State** отображается состояние среды исполнения приложения CoDeSys:

Run – приложение выполняется;

Stop – приложение приостановлено;

Breakpoint – приложение остановлено на точке останова.

Поле **Project** содержит имя файла проекта приложения, загруженного в контроллер, а также информацию о версии приложения, введенную в поле **Version** диалоговой панели **Project Information** среды разработки CoDeSys, а также информацию о версии системного программного обеспечения контроллера – в скобках с префиксом *FW:*.

Кнопка **Reset** позволяет выполнить «мягкую» перезагрузку контроллера, при которой происходит перезапуск только среды исполнения.

Кнопка **Stop** позволяет приостановить выполнение приложения. После ее нажатия надпись на кнопке изменится на **Start**, а в поле **State** будет отображено состояние среды исполнения *Stop*.

Если в контроллер загружено приложение, содержащее хотя бы одну форму визуализации при том, что в свойствах платформы **Target Settings** включена поддержка целевой визуализации, на экране монитора, подключенного к контроллеру, будет отображаться окно формы визуализации, первой в списке форм загруженного проекта, отображаемом во вкладке **Visualizations** среды разработки CoDeSys. Окно консоли контроллера при этом будет скрыто.

Для того, чтобы сделать видимым окно консоли контроллера, следует щелкнуть левой кнопкой мыши в окне визуализации, после чего нажать комбинацию клавиш Shift-Escape (один или два раза) на

клавиатуре, подключенной к контроллеру. Для того, чтобы скрыть отображаемое окно консоли контроллера, следует нажать клавишу Escape.

9.3.3. Функционирование в безопасном режиме

Если приложение в контроллер не загружено или если в процессе работы приложения, ранее загруженного в контроллер, произошла ошибка, контроллер будет функционировать в безопасном режиме (см. п. 4.2.1.1).

При отсутствии в контроллере загруженного приложения на экране монитора контроллера отображается окно консоли, показанное на рис. 58.

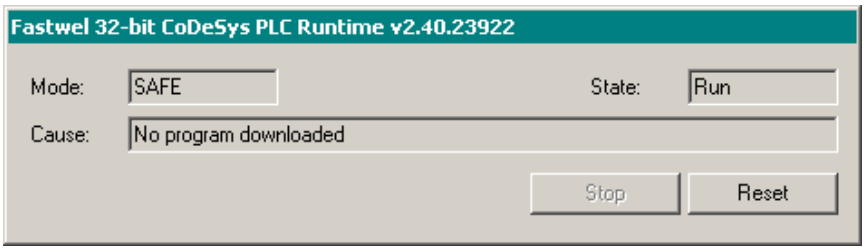


Рис. 58. Окно консоли ПЛК в безопасном режиме при отсутствии загруженного приложения

Поле **Mode** содержит название текущего режима – *SAFE* (безопасный), а поле **Cause** – причину безопасного режиме. О том, что в контроллере отсутствует загруженное приложение, свидетельствует сообщение *No program downloaded* в поле **Cause**.

Например, при работе в безопасном режиме, в который контроллер перешел по ошибке целочисленного деления на 0, окно консоли контроллера будет иметь вид, показанный на рис. 59. Информация о кодах причин перехода в безопасный режим приведена в разделе **Ошибка! Источник ссылки не найден.** настоящего руководства.

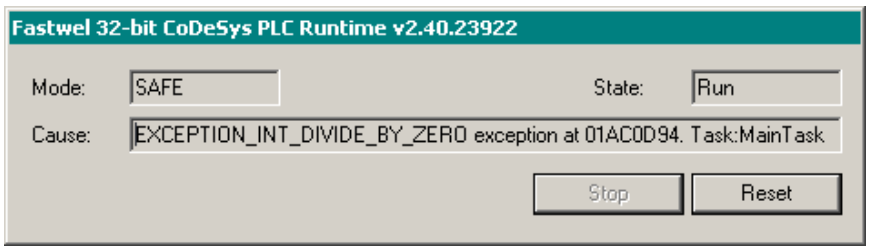


Рис. 59. Окно консоли ПЛК в безопасном режиме по ошибке в приложении

9.4. Указания по применению

9.4.1. Настройка свойств целевой визуализации

9.4.1.1. Общие сведения

Настройка свойств целевой визуализации может состоять из двух частей:

1. настройки разрешения видеоподсистемы контроллера;
2. активизации целевой визуализации в проекте с выбором размеров форм визуализации в проекте CoDeSys.

9.4.1.2. Настройка разрешения видеоподсистемы контроллера

Перед настройкой разрешения видеоподсистемы необходимо подключить к контроллеру клавиатуру и мышь. При использовании клавиатуры и мыши с интерфейсом USB допускается выполнять подключение без отключения питания контроллера.

1. Выполните команду **Start**–**Run**, введите команду regedit в появившейся диалоговой панели **Run** и нажмите Enter. На экран монитора контроллера будет выведено главное окно приложения просмотра и редактирования реестра.
2. Раскройте и выберите ключ реестра *HKEY_LOCAL_MACHINE\Software\Geode* в дереве, расположенном в левой области главного окна редактора реестра.

3. В правой области главного окна редактора реестра дважды щелкните левой кнопкой мыши над параметром **Resolution**, после чего в появившейся диалоговой панели **Edit STRING Value** в поле **Value data** введите новое значение разрешения по горизонтали, вертикали и частоту горизонтальной развертки из ряда:
```
«640 480 60»
«640 480 75»
«800 600 60»
«800 600 75»
«1024 768 60»
«1024 768 75»
«1152 864 60»
«1152 864 75»
«1280 1024 60»
«1280 1024 75»
```
и закройте диалоговую панель нажатием кнопки **OK** или нажатием клавиши Enter.
4. Завершите работу программы regedit.
5. Перезапустите контроллер, выключив и повторно включив питание.

ВНИМАНИЕ!

Если в процессе настройки разрешения видеоподсистемы контроллера введены такие значения параметров, что после перезапуска контроллера содержимое экрана монитора, подключенного к MK905, искажено или отсутствует:

1. Выключите питание MK905, отверните винт, фиксирующий защитную планку гнезда CompactFlash-диска MK905, расположенного на задней панели MK905.
2. Извлеките CompactFlash-диск, нажав кнопку экстрактора гнезда.
3. Подключите устройство чтения CompactFlash-дисков (CardReader) к инструментальному компьютеру и установите CompactFlash-диск MK905 в гнездо CardReader, после чего откройте *My Computer* (*Мой компьютер*).
4. Откройте диск MK905 и удалите со всем содержимым подкаталог *Registry*, расположенный в корневом каталоге CompactFlash MK905.
5. Щелкните правой кнопкой мыши над буквой дискового накопителя, присвоенной Windows CompactFlash-диску MK905, выполните команду **Eject** (**Извлечь**), после чего извлеките CompactFlash-диск MK905 из CardReader.
6. Установите CompactFlash-диск MK905 в гнездо, расположенное на задней панели контроллера, установите защитную планку и заверните фиксирующий винт.
7. Включите питание MK905. Контроллер запустится с исходными параметрами видеоподсистемы: 800×600×16.

9.4.1.3. Настройка параметров целевой визуализации в проекте

Для включения поддержки целевой визуализации в проекте CoDeSys для MK905:

1. Выберите вкладку **Resources** в левой области главного окна CoDeSys и дважды щелкните на ресурсе **Target Settings**.
2. В появившейся диалоговой панели **Target Settings** щелкните на вкладке **Visualization** и отметьте флажок **Target visualization**, как показано на рис. 56. При необходимости использования архивации значений переменных, отображаемых элементом Тренд, следует также отметить флажок **Store trend data in PLC**.
3. В полях **Display width in pixel** и **Display height in pixel** задайте разрешение экрана монитора, который предполагается подключить к контроллеру.
4. Закройте диалоговую панель **Target Settings** нажатием кнопки **OK**.

9.4.1.4. Отключение загрузки вспомогательных файлов визуализации

При редактировании и/или отладке некоторого проекта, в процессе чего приходится часто загружать обновленный проект в контроллер, возможно отключить загрузку неизменных вспомогательных файлов визуализации, относящихся к текущему проекту:

1. Выберите вкладку **Resources** в левой области главного окна CoDeSys и дважды щелкните на ресурсе **Target Settings**.

2. В появившейся диалоговой панели **Target Settings** щелкните на вкладке **Visualization** и отметьте флажок **Inhibit download of visualization files**.
3. Закройте диалоговую панель **Target Settings** нажатием кнопки **OK**.

Впоследствии при необходимости обновления изменившихся вспомогательных файлов визуализации достаточно снять флажок **Target Settings – Visualization – Inhibit download of visualization files**.

9.4.2. Создание и редактирование форм визуализации

Создание и редактирование форм визуализации в проекте выполняется в соответствии с указаниями документа «Визуализация CoDeSys. Дополнение к руководству пользователя по программированию ПЛК в CoDeSys 2.3». При установке пакета адаптации с параметрами по умолчанию местоположение документа:

C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Doc\CoDeSys\Russian\ CoDeSys_Visu_V23_RU.pdf

Кроме того, документ может быть загружен по адресу:

ftp://ftp.prosoft.ru/pub/hardware/Fastwel/Fastwel_IO/Version2/Doc/CoDeSys/Russian/CoDeSys_Visu_V23_RU.pdf

Подсистема целевой визуализации MK905 поддерживает все функциональные возможности целевой визуализации, исключая отображение элементов управления ActiveX, Web visualization и обработку тревог (**Alarmhandling in the PLC**).

9.4.3. Особенности работы с растровыми изображениями

Подсистема целевой визуализации контроллера обеспечивает возможность отображения изображений в формате bmp и jpeg в качестве фоновых рисунков форм визуализации и кнопок.

Перед прикреплением файла изображения в качестве фонового рисунка (по команде меню **Extras–Select background bitmap** – для текущей формы визуализации, или в категория **Bitmap** свойств текущей выбранной кнопки) следует поместить требуемые файлы изображений в каталог расположения файла проекта. При этом настоятельно рекомендуется предварительно обработать каждый файл изображения любым доступным графическим редактором таким образом, чтобы он имел минимально возможный размер. Это обусловлено следующими соображениями:

1. Файлы изображений загружаются в контроллер сразу после загрузки основных секций проекта по команде **Online–Login**. Загрузка файлов большого размера может занимать довольно длительное время.
2. Загрузка изображений большого размера в память при запуске приложения может занимать довольно длительное время и требовать большого количества системных ресурсов контроллера.
3. При обновлении проекта в контроллере наличие большого количества изображений может привести к невозможности загрузки некоторых из них из-за фрагментации памяти.

9.4.4. Особенности отображения и ввода текстовой информации

9.4.4.1. Общие сведения

Адаптированная версия подсистемы целевой визуализации CoDeSys обеспечивает возможность отображения и ввода пользователем строковой информации символами кириллицы без использования специальных механизмов динамического текста, предлагаемого средой CoDeSys (см. п. 9.4.5). В настоящем подразделе рассматриваются примеры вывода статической и динамической строковой информации символами кириллицы.

9.4.4.2. Вывод статической текстовой информации символами кириллицы

Под статической текстовой информацией подразумеваются надписи на кнопках и других графических примитивах, которые не изменяются в процессе работы приложения.

Рассмотрим пример создания кнопки с надписью на русском языке.

1. Добавьте в проект форму визуализации, для чего щелкните на вкладке **Visualizations** в левой области главного окна CoDeSys, щелкните правой кнопкой мыши над элементом

дерева **Visualizations**, выберите команду **Add Object** в контекстном меню, введите имя формы в поле **Name of the new Visualization** появившейся диалоговой панели и нажмите клавишу Enter или нажмите кнопку **OK**. Вновь созданная форма появится в правой области главного окна CoDeSys.

2. Выберите команду **Insert–Button** в главном меню или щелкните на пиктограмме в панели инструментов, после чего нарисуйте кнопку в окне редактирования формы визуализации, как показано на рис. 60. Для этого нажмите левую кнопку мыши в точке формы, где должен быть расположен верхний левый угол кнопки, после чего, оставив левую кнопку мыши нажатой, переместите курсор в точку, в которой должен находиться правый нижний угол кнопки, и отпустите левую кнопку мыши.
3. Дважды щелкните над вновь созданной кнопкой и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration**.
4. Нажмите кнопку **Default Font** в свойствах текста кнопки.
5. Переключите раскладку клавиатуры компьютера на русскую и в поле **Content** свойств текста кнопки введите надпись, которая должна отображаться на кнопке, на русском языке, как показано на рис. 61.
6. Нажмите кнопку **Font** в свойствах кнопки и выберите требуемый тип, размер и оформление шрифта, которым будет отображаться надпись на созданной кнопке.
7. Закройте диалоговую панель свойств кнопки **Regular Element Configuration** нажатием кнопки **OK**. Введенная надпись на русском языке будет отображена на кнопке выбранным типом шрифта.

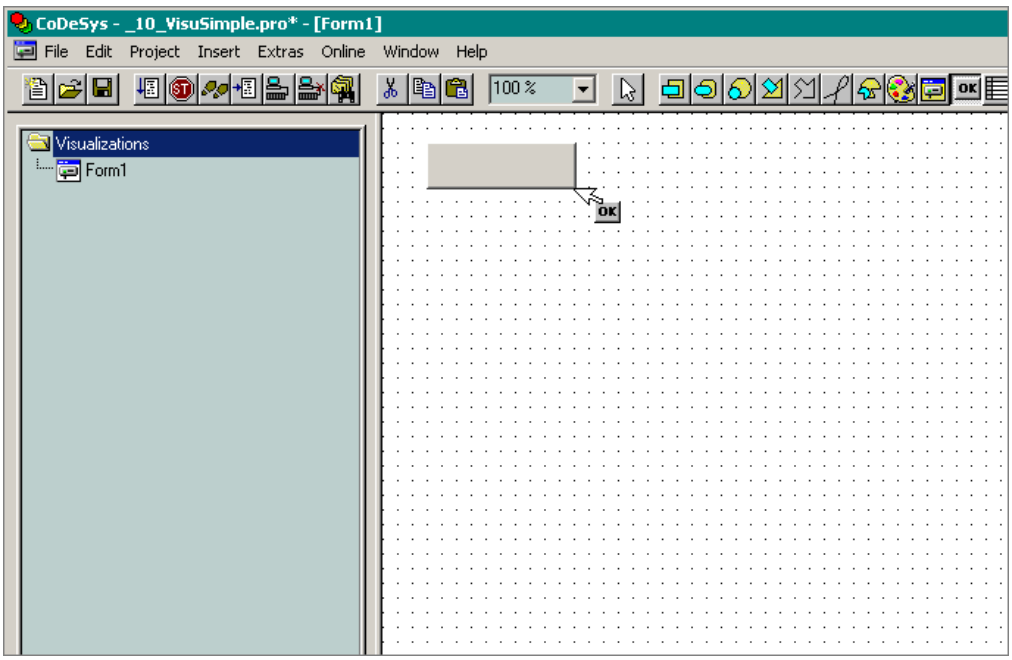


Рис. 60. Добавление новой формы

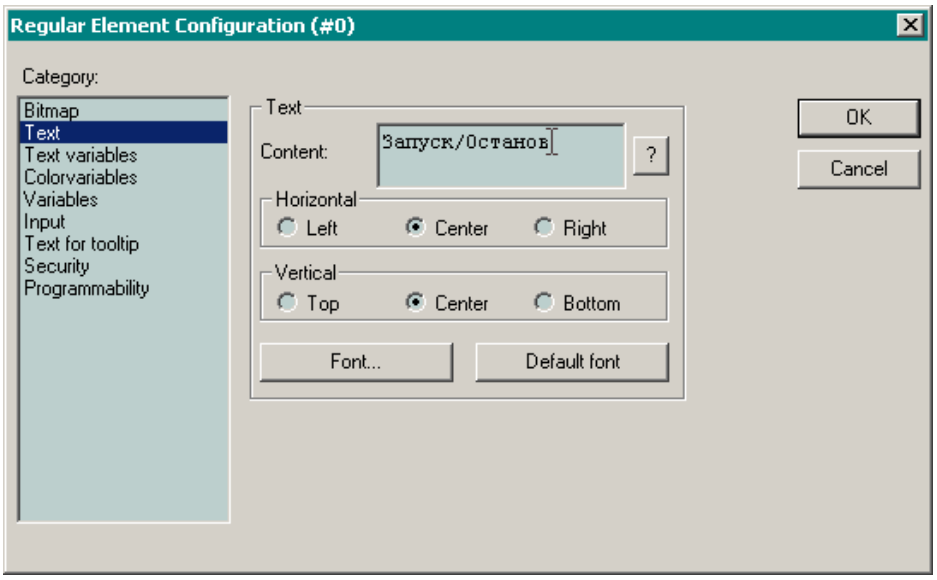


Рис. 61. Ввод статической надписи для отображения на кнопке

8. После загрузки проекта в контроллер надпись на кнопке будет отображена на русском языке выбранным типом и оформлением шрифта, как показано на рис. 62.

ПРИМЕЧАНИЕ. Весьма важно нажать кнопку **Default font** в свойствах текста перед вводом строки символами кириллицы. При этом, если установка адаптации среды CoDeSys для Fastwel I/O выполнялась в точном соответствии указаниям п. 3.2, для текста графического примитива будет установлен набор символов кодовой страницы 1251.

Рис. 62. Отображение статической надписи на кнопке символами кириллицы

9.4.4.3. Вывод динамической информации символами кириллицы

Под динамической текстовой информацией подразумеваются надписи на кнопках и других графических примитивах, которые должны изменяться в процессе работы приложения в зависимости от выполнения некоторых условий.

Рассмотрим пример создания кнопки с надписью на русском языке, которая изменяется в зависимости от значения булевой переменной, переключаемого кнопкой.

1. В область объявления переменных программы (например, PLC_PRG) добавьте следующие переменные:

```
VAR

   (* Переменная, значение которой будет переключаться кнопкой *)
   TOGGLE_VAR : BOOL := FALSE;
   (* Надпись на отпущенной кнопке *)
   LABEL_START : STRING := 'Запуск';
   (* Надпись на нажатой кнопке *)
   LABEL_STOP : STRING := 'Останов';
   (* Переменная, с которой будет связана надпись на кнопке *)
   TOGGLE_BUTTON_LABEL : STRING;

END_VAR
```

2. Добавьте в тело программы (например, PLC_PRG) следующий исходный текст:

```
(* Если TOGGLE_VAR = TRUE, отображаем надпись Останов *)
(* Если TOGGLE_VAR = FALSE, отображаем надпись Запуск *)
IF TOGGLE_VAR THEN
   TOGGLE_BUTTON_LABEL := LABEL_STOP;
```

```
ELSE
   TOGGLE_BUTTON_LABEL := LABEL_START;
END_IF
```

3. Выберите команду **Insert–Button** в главном меню или щелкните на пиктограмме  в панели инструментов, после чего нарисуйте кнопку в окне редактирования формы визуализации, как показано на рис. 60. Для этого нажмите левую кнопку мыши в точке формы, где должен быть расположен верхний левый угол кнопки, после чего, оставив левую кнопку мыши нажатой, переместите курсор в точку, в которой должен находиться правый нижний угол кнопки, и отпустите левую кнопку мыши.
4. Дважды щелкните над вновь созданной кнопкой и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration**.
5. Нажмите кнопку **Default Font** в свойствах текста кнопки.
6. В поле **Content** категории свойств **Text** введите спецификатор вывода текстовой строки %s, как показано на рис. 63.

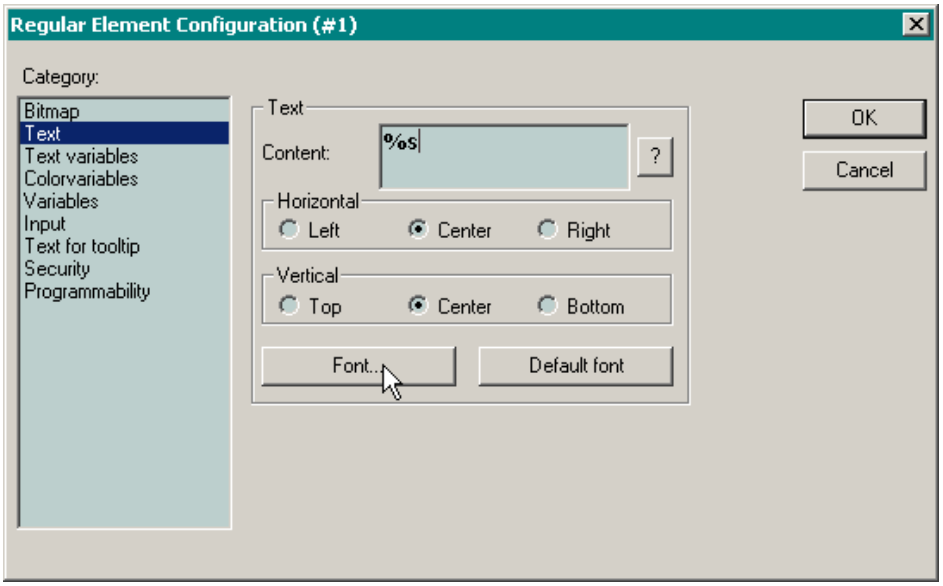


Рис. 63. Ввод спецификатора вывода текстовой строки

7. Нажмите кнопку **Font** в свойствах кнопки и выберите требуемый тип, размер и оформление шрифта, которым будет отображаться надпись на созданной кнопке.
8. Выберите категорию свойств **Variables** в диалоговой панели **Regular Element Configuration** и в поле **Textdisplay** введите имя переменной PLC_PRG. TOGGLE_BUTTON_LABEL. Ввод может быть осуществлен при помощи диалоговой панели **Input assistant**, вызываемой по нажатию клавиши F2.
9. Выберите категорию свойств **Input** в диалоговой панели **Regular Element Configuration**, отметьте флажок **Toggle variable** и справа от него введите имя переменной PLC_PRG. TOGGLE_VAR. Ввод может быть осуществлен при помощи диалоговой панели **Input assistant**, вызываемой по нажатию клавиши F2.
10. Закройте диалоговую панель свойств кнопки **Regular Element Configuration** нажатием кнопки **OK**, после чего загрузите проект в контроллер. Сразу после загрузки кнопка будет иметь надпись **Запуск**.
11. Нажмите кнопку и убедитесь, что в надпись на кнопке в нажатом состоянии изменилась на **Останов**, как показано на рис. 64.

Рис. 64. Динамическая смена надписи на кнопке

9.4.5. Указания по реализации поддержки нескольких языков интерфейса оператора

9.4.5.1. Общие сведения

Настоящий подраздел содержит краткие иллюстративные указания по реализации поддержки нескольких языков интерфейса оператора в приложении CoDeSys для контроллера MK905. Более подробная информация приведена в документе Визуализация CoDeSys. Дополнение к руководству пользователя по программированию ПЛК в CoDeSys 2.3.

Процесс создания многоязычного интерфейса в приложении состоит из следующих шагов:

1. Создание одного или нескольких текстовых файлов в формате xml, содержащих список поддерживаемых языков, а также префиксы, идентификаторы и значения строк для поддерживаемых языков. Данные файлы далее называются файлами динамического текста.
2. Подключение файлов динамического текста к проекту CoDeSys.
3. Связывание свойств **Text**–**Content** и **Variables**–**Textdisplay** графических примитивов, отображающих текст, с префиксами и идентификаторами строк соответственно.
4. Загрузка в контроллер проекта.

В данном подразделе перечисленные шаги будут рассмотрены на примере создания простой формы визуализации, содержащей кнопки выбора языка и запуска/останова изменения некоторой целочисленной переменной, а также надпись отображаемого значения и само отображаемое значение некоторой переменной, как показано на рис. 65.

Нажатие кнопки **Запуск/Останов** будет приводить к запуску изменения внутренней переменной приложения в диапазоне от 0 до 360, а отпускание – к прекращению изменения. Нажатие кнопки **Выбор языка...** будет приводить к появлению системной диалоговой панели выбора языка. В случае, если пользователь меняет язык интерфейса пользователя, все надписи формы визуализации автоматически изменяются на соответствующие строки на выбранном языке.

При создании формы визуализации далее предполагается, что пользователь создал проект для MK905 в среде CoDeSys, включил поддержку визуализации и создал одну пустую форму визуализации. Кроме того, предполагается, что в проекте имеется программа PLC_PRG.

Рис. 65. Простая форма визуализации с возможностью динамической смены языка

9.4.5.2. Создание файла динамического текста

Файл динамического текста в формате xml состоит из секции заголовка, описывающего доступные языки и свойства отображения текста, и секции строковых ресурсов, каждый из которых содержит префикс и идентификатор ресурса и значения описываемой строки на каждом доступном языке.

В текстовом редакторе или в xml-редакторе создайте файл TutorLangs.xml следующего содержания:

```
<dynamic-text>
<header>
   <default-language>rus</default-language>
   <default-font>
          <language>rus</language>
          <font-name> Tahoma </font-name>
          <font-color>0,0,0</font-color>
          <font-height>-15</font-height>
          <font-weight>800</font-weight>
          <font-italic>false</font-italic>
          <font-underline>false</font-underline>
          <font-strike-out>false</font-strike-out>
          <font-char-set>0</font-char-set>
   </default-font>
   <default-font>
          <language>eng</language>
          <font-name> Tahoma </font-name>
          <font-color>0,0,0</font-color>
          <font-height>-15</font-height>
          <font-weight>800</font-weight>
          <font-italic>false</font-italic>
          <font-underline>false</font-underline>
          <font-strike-out>false</font-strike-out>
          <font-char-set>0</font-char-set>
   </default-font>
</header>

<text-list>
   <text prefix="TEXTHOLDER" id="1">
          <rus> <![CDATA[Выбор языка...]]> </rus>
          <eng> <![CDATA[Select Language...]]> </eng>
   </text>
   <text prefix="TEXTHOLDER" id="2">
          <rus> <![CDATA[Симуляция угла:]]> </rus>
          <eng> <![CDATA[Angle Simulation:]]> </eng>
   </text>
   <text prefix="TEXTHOLDER" id="3">
          <rus> <![CDATA[Запуск/Останов]]> </rus>
          <eng> <![CDATA[Start/Stop]]> </eng>
   </text>
</text-list>
</dynamic-text>
```

Парные теги **dynamic-text** ограничивают содержимое файла динамического текста.

Парные теги **header** определяют секцию заголовка, в которой имеются теги **default-language** и **default-font**, определяющие исходный язык интерфейса оператора при запуске приложения и перечень свойств шрифтов для поддерживаемых языков. Теги **language** внутри **default-font** определяют доступные языки интерфейса оператора.

Парные теги **text-list** ограничивают список строковых ресурсов, описанных в данном текстовом файле. Каждая строка (строковый ресурс) представлена подсекцией **text** с атрибутами **prefix** и **id**, а также значениями строк для каждого языка, определенного в заголовке. Например, описание ресурса:

```
<text prefix="TEXTHOLDER" id="1">
     <rus> <![CDATA[Выбор языка...]]> </rus>
     <eng> <![CDATA[Select Language...]]> </eng>
</text>
```

определяет строковый ресурс с префиксом TEXTHOLDER и идентификатором 1, для которого определены строковые значения для языков **rus** и **eng**, которые, в свою очередь, ранее определены в секции **header.**

9.4.5.3. Подключение файла динамического текста к проекту CoDeSys

Для подключения к проекту файла динамического текста, созданного в п. 9.4.5.2:

1. Сделайте видимой форму визуализации в редакторе форм, для чего дважды щелкните на ее названии в списке, доступном во вкладке **Visualizations**.
2. Выберите команду **Extras–Settings** в главном меню CoDeSys. На экран монитора будет выведена диалоговая панель **Visualization settings**.
3. Выберите категорию свойств **Language**, отметьте флажок **Dynamic texts**, нажмите кнопку **Add** и выберите файл динамического текста TutorLangs.xml, созданный в п. 9.4.5.2. Имя файла появится в списке подключенных файлов динамического текста под флажком **Dynamic texts**, а выпадающий список **Language**, расположенный в нижней части диалоговой панели, будет содержать название языка, заданного тегом **`default-language`** в секции заголовка подключенного файла динамического текста. Содержимое диалоговой панели **Visualization settings** после подключения файла динамического текста показано на рис. 66.

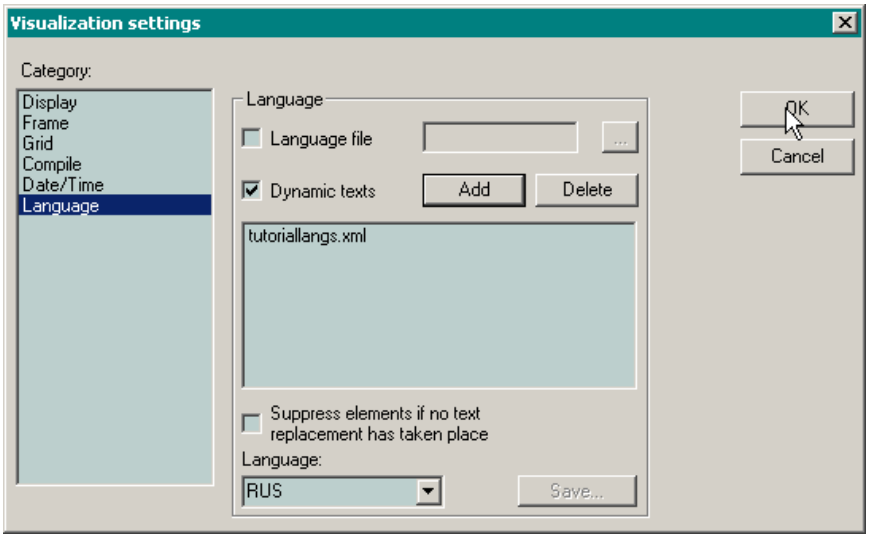


Рис. 66. Диалоговая панель Visualization settings, категория Language после подключения файла динамического текста

4. Закройте диалоговую панель **Visualization settings** нажатием кнопки **OK**.

9.4.5.4. Создание программы

1. Щелкните на вкладке **POUs** в главном окне CoDeSys и дважды щелкните на имени программы PLC_PRG.
2. В области деклараций переменных редактора Structured Text добавьте следующие декларации переменных:

```
VAR
   (* Переменная, активизирующая изменение ANGLE_VAR *)
   TOGGLE_VAR : BOOL := FALSE;
   (* Переменная, изменяющаяся от 0 до 360, если TOGGLE_VAR=TRUE*)
   ANGLE_VAR : WORD := 0;
END_VAR
```

3. В теле PLC_PRG введите следующий исходный текст:

```
(* Если TOGGLE_VAR=TRUE, ANGLE_VAR циклически изменяется от 0 до 360 *)
IF TOGGLE_VAR THEN
   ANGLE_VAR := (ANGLE_VAR + 1) MOD 360;
END_IF
```

Значение переменной TOGGLE_VAR будет переключаться с FALSE на TRUE одной из кнопок на форме визуализации, а значение ANGLE_VAR будет отображаться в текстовом поле этой формы.

9.4.5.5. Создание кнопки выбора языка

1. Сделайте видимой форму визуализации в редакторе форм, для чего дважды щелкните на ее названии в списке, доступном во вкладке **Visualizations**.

2. Выберите команду **Insert–Button** в главном меню или щелкните на пиктограмме  в панели инструментов, после чего нарисуйте кнопку в окне редактирования формы визуализации, как показано на рис. 67. Для этого нажмите левую кнопку мыши в точке формы, где должен быть расположен верхний левый угол кнопки, после чего, оставив левую кнопку мыши нажатой, переместите курсор в точку, в которой должен быть расположен правый нижний угол кнопки, и отпустите левую кнопку мыши.

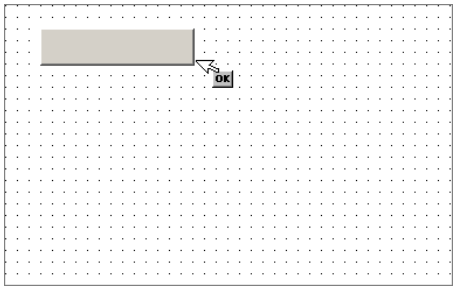


Рис. 67. Создание кнопки выбора языка

3. Дважды щелкните над вновь созданной кнопкой и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration**.
4. Используя синтаксис динамического текста, введите в поле **Content** имя префикса текстового ресурса, как показано на рис. 68.

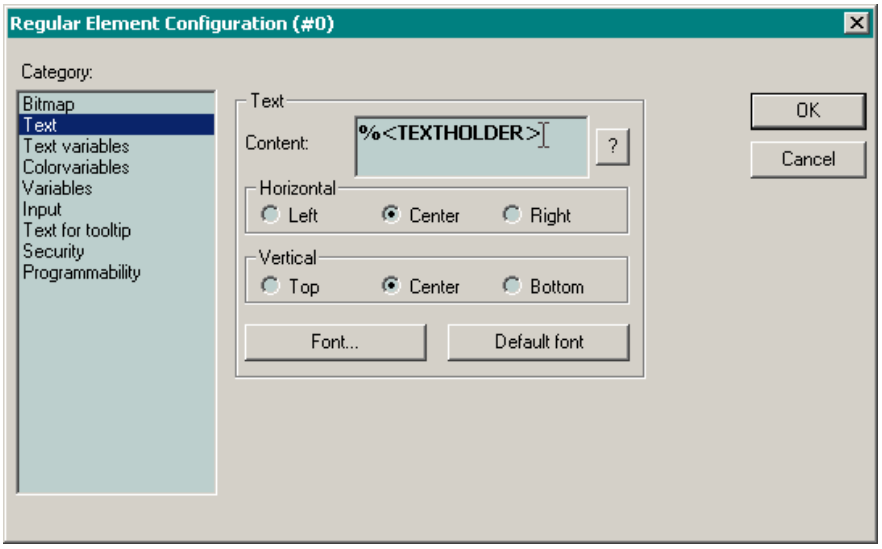


Рис. 68. Добавление префикса текстового ресурса, доступного в подключенном файле динамического текста

Введенный префикс должен совпадать с использованным в файле динамического текста TutorLangs.xml.

5. Для выбора надписи, которая будет отображаться на кнопке, щелкните на категории **Variables** и в поле **Textdisplay** введите числовой идентификатор соответствующей строки. В данном случае кнопке выбора языка соответствует идентификатор 1, как показано на рис. 69.

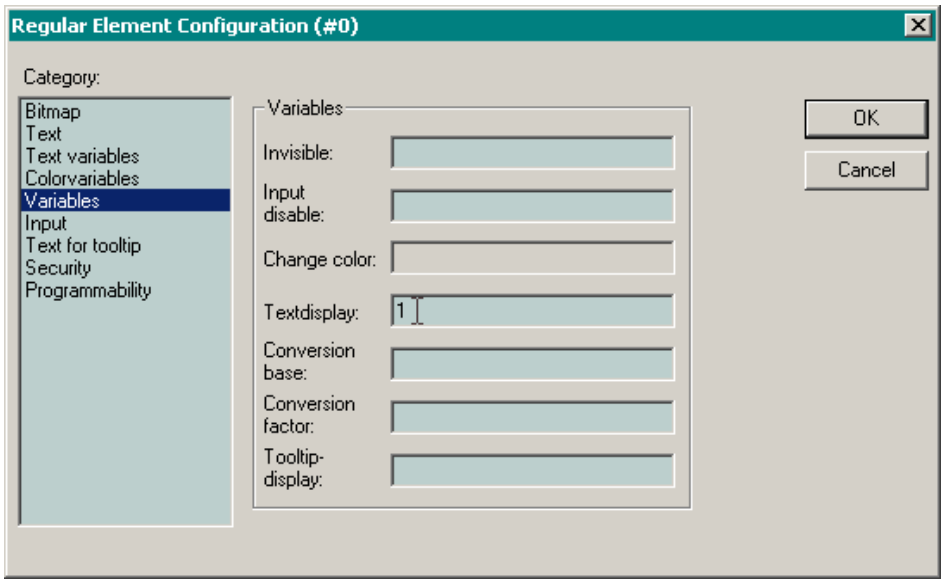


Рис. 69. Ввод идентификатора текстового ресурса, доступного в подключенном файле динамического текста

6. Для выбора действия, выполняемого при нажатии данной кнопки, щелкните на категории свойств **Input**, отметьте флажок **Execute program** и нажмите кнопку **...**, расположенную справа от флажка. В выпадающем списке появившейся диалоговой панели **Configure programs** выберите действие LANGUAGEDIALOG и нажмите кнопку **Add**, после чего закройте диалоговую панель **Configure programs** нажатием кнопки **OK**. Страница свойств категории **Input** диалоговой панели **Regular Element Configuration** примет вид, показанный на рис. 70.

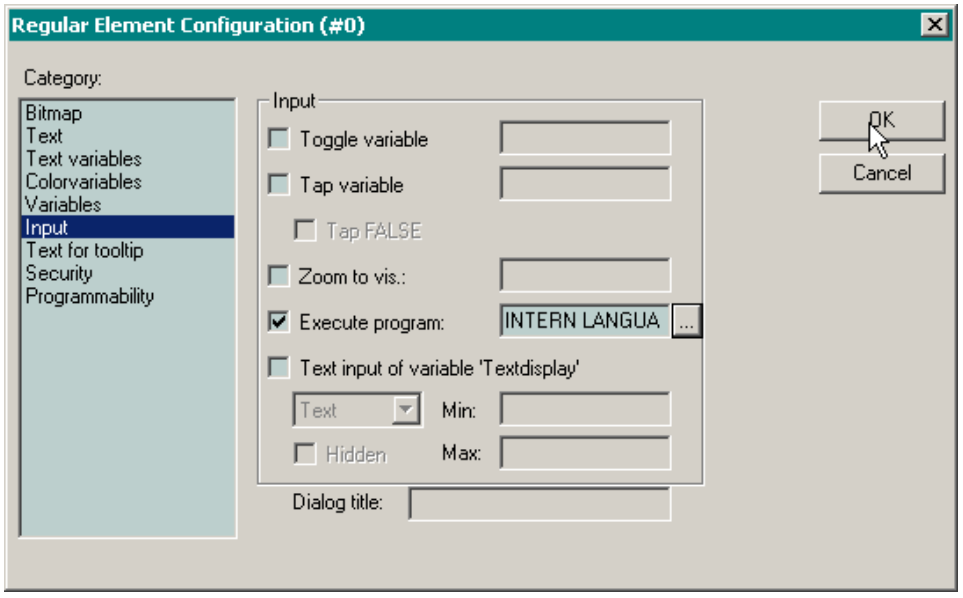


Рис. 70. Завершение редактирования свойств кнопки выбора языка

7. Закройте диалоговую панель **Regular Element Configuration** нажатием кнопки **OK**.

9.4.5.6. Создание кнопки переключения TOGGLE_VAR

1. Добавьте на форму еще одну кнопку, для чего выполните действия пп. 2–5 п. 9.4.5.5. В качестве идентификатора строки, отображаемой на кнопке, для свойства **Variables–Textdisplay** введите 3.
2. Выберите категорию свойств **Input** в диалоговой панели **Regular Element Configuration**, отметьте флажок **Toggle variable** и справа от него введите имя переменной PLC_PRG.TOGGLE_VAR. Ввод может быть осуществлен при помощи диалоговой панели **Input assistant**, вызываемой по нажатию клавиши F2.
3. Закройте диалоговую панель **Regular Element Configuration** нажатием кнопки **OK**.

9.4.5.7. Создание статической надписи для отображаемого значения ANGLE_VAR

Статические надписи могут создаваться на основе графического примитива Прямоугольник.

1. Выберите команду **Insert–Rectangle** в главном меню или щелкните на пиктограмме  в панели инструментов, после чего нарисуйте прямоугольник в окне редактирования формы визуализации, как показано на рис. 71. Для этого нажмите левую кнопку мыши в точке формы, где должен быть расположен верхний левый угол прямоугольника, после чего, оставив левую кнопку мыши нажатой, переместите курсор в точку, в которой должен находиться правый нижний угол прямоугольника, и отпустите левую кнопку мыши.

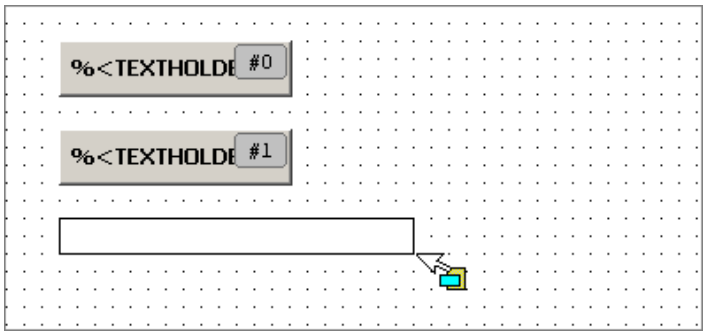


Рис. 71. Создание графического примитива Прямоугольник

2. Дважды щелкните над вновь созданным прямоугольником и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration**.
3. Установите выравнивание текста по левой границе, для чего установите переключатель **Horizontal–Left**.
4. Нажмите кнопку **Default font**, после чего нажмите кнопку **Font** и выберите требуемый тип, размер и оформление шрифта, которым будет отображаться надпись.
5. Используя синтаксис динамического текста, введите в поле **Content** имя префикса текстового ресурса, как показано на рис. 72.

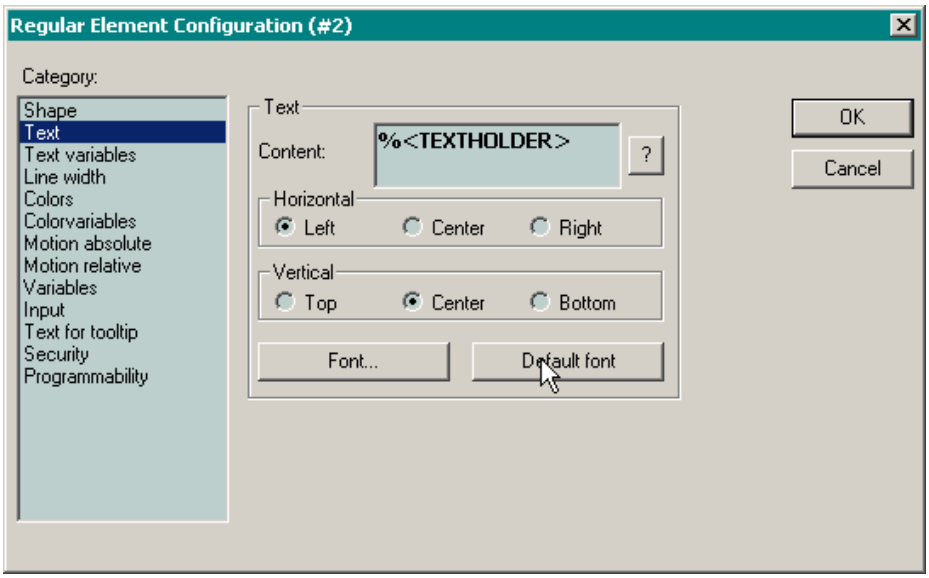


Рис. 72. Ввод префикса текстового ресурса, доступного в подключенном файле динамического текста

6. Для выбора отображаемой надписи щелкните на категории **Variables** и в поле **Textdisplay** введите числовой идентификатор соответствующей строки. В данном случае статической надписи отображаемого значения соответствует идентификатор 2, как показано на рис. 73.

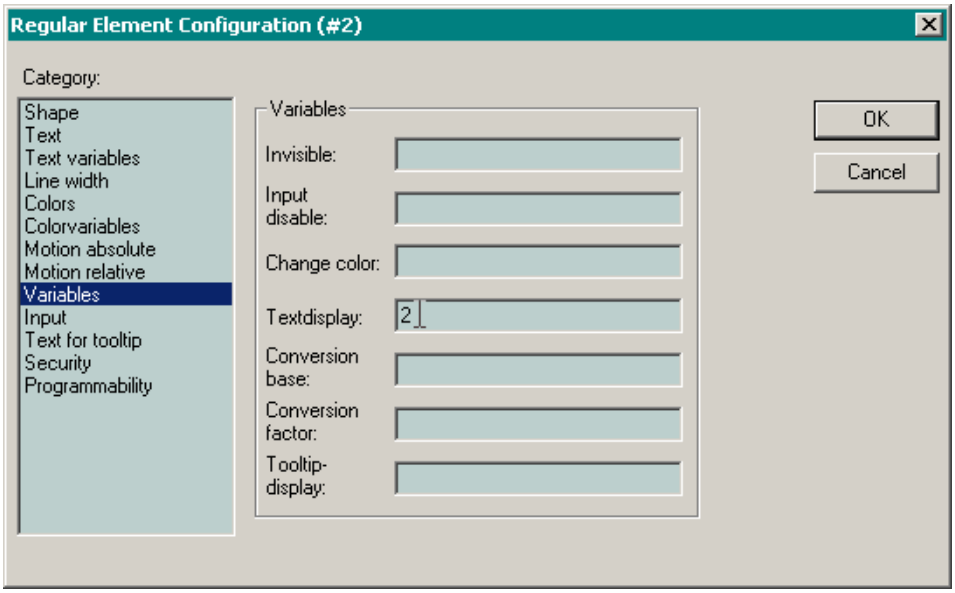


Рис. 73. Ввод идентификатора текстового ресурса, доступного в подключенном файле динамического текста

7. Для того, чтобы сделать невидимыми границы прямоугольника и сделать фон надписи прозрачным, выберите категорию **Colors** и отметьте в ней флажки **No color inside** и **No frame color**, как показано на рис. 74, после чего закройте диалоговую панель **Regular Element Configuration** нажатием кнопки **OK**.

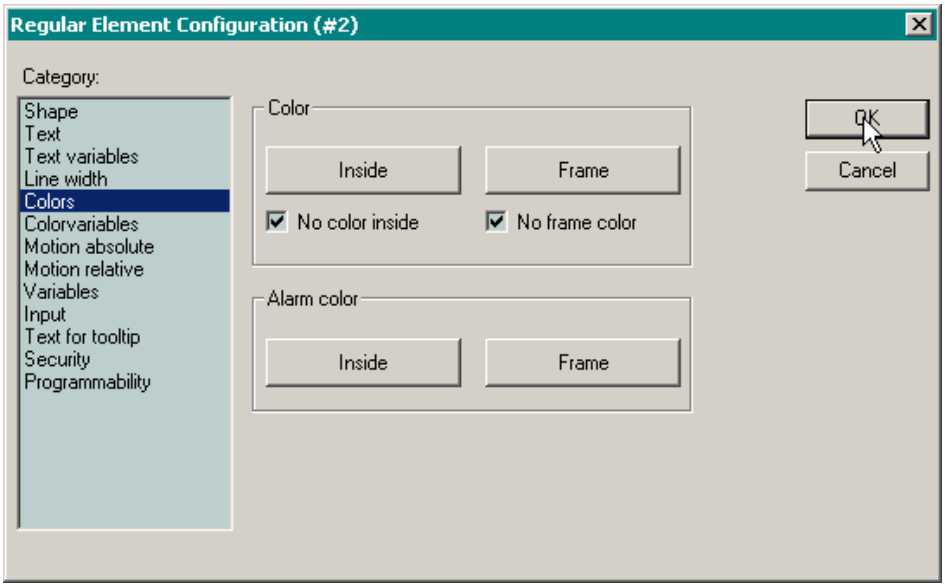


Рис. 74. Отключение отображения границ и фона надписи

9.4.5.8. Создание динамического элемента отображения значения ANGLE_VAR

Динамические элементы отображения числовых и строковых значений могут создаваться на основе графических примитивов Прямоугольник.

1. Выберите команду **Insert–Rectangle** в главном меню или щелкните на пиктограмме в панели инструментов, после чего нарисуйте прямоугольник в окне редактирования формы визуализации справа от статической надписи, созданной в п. 9.4.5.7, как показано на рис. 75.
2. Дважды щелкните над вновь созданным прямоугольником и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration**.
3. Установите выравнивание текста по левой границе, для чего установите переключатель **Horizontal–Left**.
4. Нажмите кнопку **Default font**, после чего нажмите кнопку **Font** и выберите требуемый тип, размер и оформление шрифта, которым будет отображаться надпись.
5. В поле **Content** категории свойств **Text** введите спецификатор вывода целого беззнакового значения %u, как показано на рис. 76.

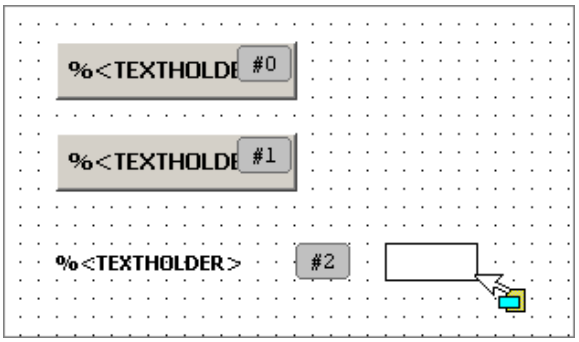


Рис. 75. Добавление прямоугольника для отображения значения переменной ANGLE_VAR

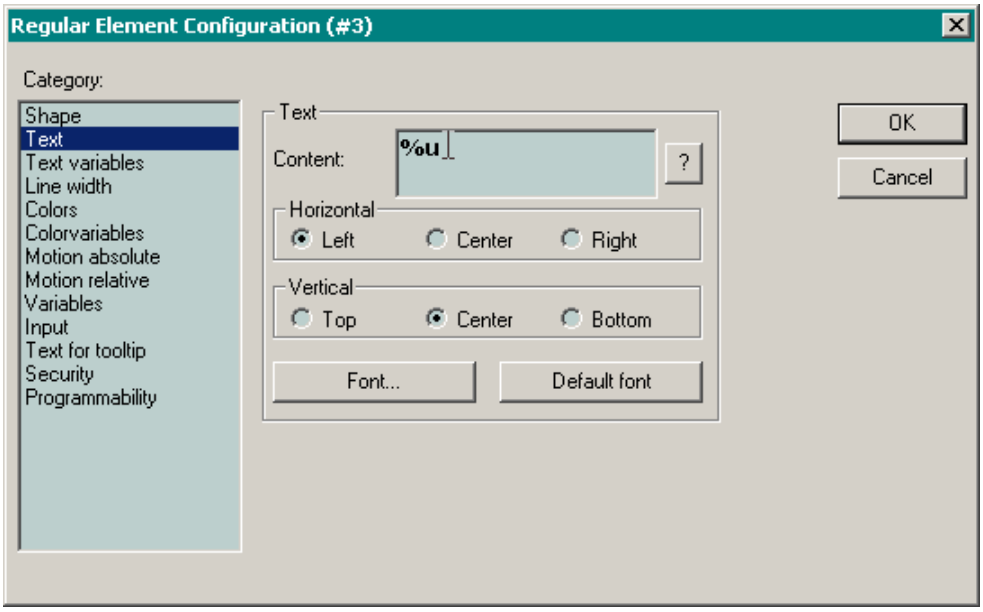


Рис. 76. Ввод спецификатора отображения беззнакового целого значения

6. Выберите категорию свойств **Variables** в диалоговой панели **Regular Element Configuration** и в поле **Textdisplay** введите имя переменной PLC_PRG.ANGLE_VAR, чье значение должно отображаться редактируемым графическим примитивом, как показано на рис. 77. Ввод может быть осуществлен при помощи диалоговой панели **Input assistant**, вызываемой по нажатию клавиши F2.

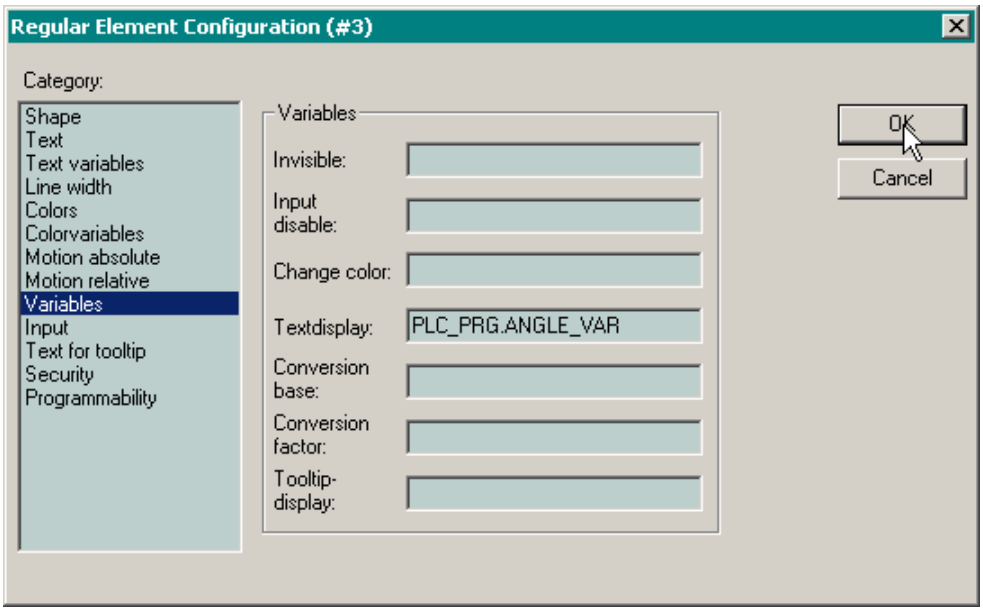


Рис. 77. Связывание элемента отображения с переменной

7. Для того, чтобы сделать невидимыми границы прямоугольника и сделать фон прозрачным, выберите категорию **Colors** и отметьте в ней флажки **No color inside** и **No frame color**, после чего закройте диалоговую панель **Regular Element Configuration** нажатием кнопки **OK**.

9.4.5.9. Загрузка приложения в контроллер

1. Выполните команду **Project–Build** и загрузите приложение в контроллер. По окончании загрузки на экране монитора, подключенного к контроллеру, появится окно созданной формы визуализации, фрагмент которого показан на рис. 78.

Рис. 78. Окно формы визуализации после загрузки приложения в контроллер

2. Нажмите кнопку **Запуск/Останов** и контролируйте изменение значения в диапазоне от 0 до 360 справа от надписи **Симуляция угла**.
3. Нажмите кнопку **Выбор языка**. На экран монитора, подключенного к контроллеру, будет выведена системная диалоговая панель **Change language**, показанная на рис. 79.

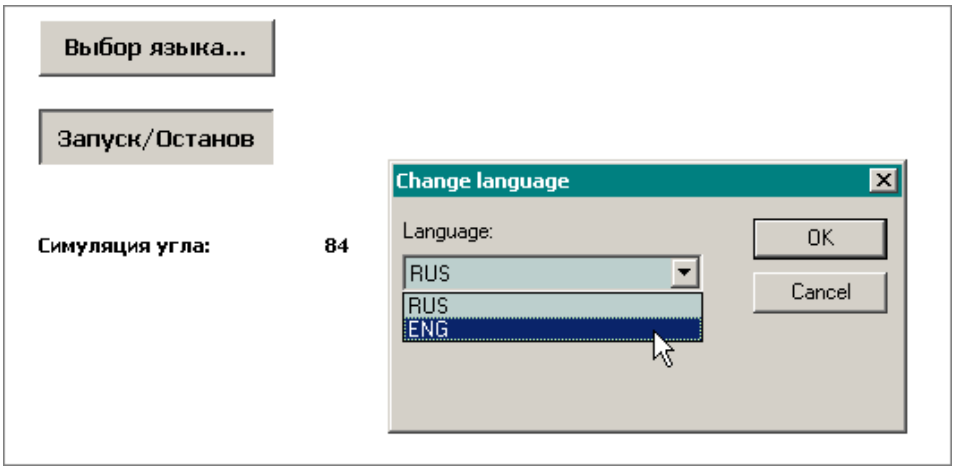


Рис. 79. Выбор языка интерфейса оператора

В выпадающем списке **Language** выберите ENG и нажмите **OK**. Надписи на кнопках и статическая надпись значения изменятся на строки, введенные в файле динамического текста для английского языка, как показано на рис. 80.

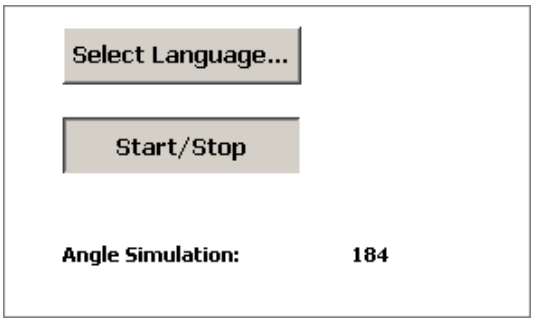


Рис. 80. Внешний вид формы после смены языка интерфейса оператора на английский

9.4.6. Особенности реализации реакции на действия оператора/пользователя

9.4.6.1. Общие сведения

В подсистеме целевой визуализации CoDeSys реализован ряд функций, обеспечивающих возможность реакции на действия оператора/пользователя над элементами форм визуализации при помощи клавиатуры и указательного устройства, подключенных к контроллеру. При этом для ограничения возможности ввода для разных категорий пользователей поддерживается до 8 уровней безопасности.

Вид реакции на действия пользователя над некоторым элементом формы визуализации устанавливается при редактировании его свойств в категории свойств **Input** диалоговой панели **Regular Element Configuration**, показанной на рис. 81.

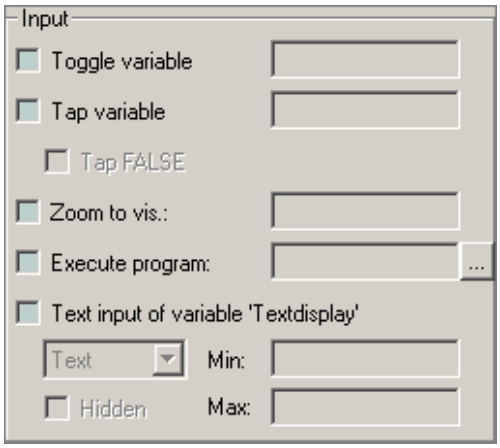


Рис. 81. Категория свойств Input элемента визуализации

Настоящий подраздел содержит краткое описание различных видов реакции на действия пользователя с учетом особенностей адаптации подсистемы целевой визуализации для контроллера MK905.

9.4.6.2. Фиксированное переключение значения булевой переменной

Для переключения значения булевой переменной с текущего на противоположное с последующей фиксацией результирующего значения в переменной служит флажок **Input–Toggle variable**.

Если флажок **Input–Toggle variable** установлен, справа от него становится доступным поле ввода имени переменной, в отношении которой будет выполнено переключение значения с текущего на противоположное при щелчке мышью над данным элементом визуализации во время исполнения приложения в контроллере. Например, если в программе имеется некоторая булева переменная, имя которой введено в поле **Input–Toggle variable**, то после загрузки приложения в контроллер каждый щелчок мышью над соответствующим элементом визуализации будет приводить к смене текущего значения переменной на противоположное до следующего щелчка.

Для ввода имени переменной можно воспользоваться диалоговой панелью **Input assistant**, выводимой при нажатии клавиши F2.

9.4.6.3. Нефиксированное переключение значения булевой переменной

Флажок **Input–Tap variable** служит для кратковременного переключения значения булевой переменной с FALSE на TRUE (если не установлен флажок **Tap FALSE**) и возврата исходного значения FALSE (если не установлен флажок **Tap FALSE**). Если одновременно установлен флажок **Tap FALSE**, то кратковременное переключение будет иметь инверсную логику: TRUE→FALSE→TRUE.

Если флажок **Input–Tap variable** установлен, справа от него становится доступным поле ввода имени переменной, в отношении которой будет выполнено кратковременное переключение значения при щелчке мышью над данным элементом визуализации во время исполнения приложения в контроллере.

Для ввода имени переменной можно воспользоваться диалоговой панелью **Input assistant**, выводимой при нажатии клавиши F2.

9.4.6.4. Переключение между окнами форм визуализации

Подсистема целевой визуализации CoDeSys в каждый момент времени позволяет отображать одно окно формы визуализации. Для переключения окон форм визуализации пользователем вручную служит флажок **Input–Zoom to vis:**.

После отметки данного флажка становится доступным поле для ввода имени формы визуализации, на которую требуется переключиться при щелчке мышью над данным элементом визуализации. В данное поле может быть введено (в том числе при помощи **Input assistant** по клавише F2):

1. Имя формы, окно которой должно быть выдвинуто на передний план.

2. Имя переменной типа STRING, которая во время выполнения приложения будет содержать имя формы визуализации.
3. Специальная команда ZOOMTOCALLER, предназначенная для возврата к форме визуализации, с которой был осуществлен переход на текущую форму.

9.4.6.5. Ввод числовой и текстовой информации по месту отображения

Элемент формы визуализации может использоваться для ввода числовых или текстовых значений для переменной, заданной в поле **Variables–Textdisplay**, в однострочном текстовом редакторе, отображаемом непосредственно по месту расположения элемента, над которым совершен щелчок мышью, как показано на рис. 82.

Рис. 82. Ввод информации по месту отображения

Для активизации редактирования по месту необходимо отметить флажок **Input–Text input of variable 'Textdisplay'** и в выпадающим списке под флажком выбрать опцию **Text**. Для контроля диапазона вводимых значений следует ввести минимальное и максимальное допустимые значения в поля **Min** и **Max**. Для переменных типа STRING в полях **Min** и **Max** можно ввести минимальное и максимальное допустимые значения длины вводимой строки.

ВНИМАНИЕ!

В среде исполнения CoDeSys значение переменной, введенное оператором при помощи данного действия, немедленно записывается в память на контексте задачи VISU_INPUT_TASK, что может привести к неправильному выполнению алгоритмов, использующих значение данной переменной на контекстах других циклических задач, вытесненных VISU_INPUT_TASK. В связи с этим переменные, которые могут быть изменены оператором из форм визуализации, должны передаваться в качестве входных в использующие их POU.

9.4.6.6. Ввод числовой информации при помощи системной диалоговой панели Numpad

Элемент формы визуализации может использоваться для ввода числовых значений для переменной, заданной в поле **Variables–Textdisplay**, в системной диалоговой панели **Numpad**, показанной на рис. 83.

Для активизации редактирования с помощью системной диалоговой панели **Numpad** необходимо отметить флажок **Input–Text input of variable 'Textdisplay'** и в выпадающим списке под флажком выбрать опцию **Numpad**. Для контроля диапазона вводимых значений следует ввести минимальное и максимальное допустимые значения в поля **Min** и **Max**. Для переменных типа STRING в полях **Min** и **Max** можно ввести минимальное и максимальное допустимые значения длины вводимой строки.

Поле **Dialog title** позволяет задать строку, которая будет отображаться в области заголовка диалоговой панели **Numpad**.

Кнопка **BACK** диалоговой панели **Numpad** служит для стирания текущего введенного символа. Кнопка **CL** служит для стирания всех введенных символов.

Для ввода значения следует щелкнуть мышью над кнопкой **OK** диалоговой панели, а для скрытия диалоговой панели с отказом от ввода значения служит кнопка **ESC**.

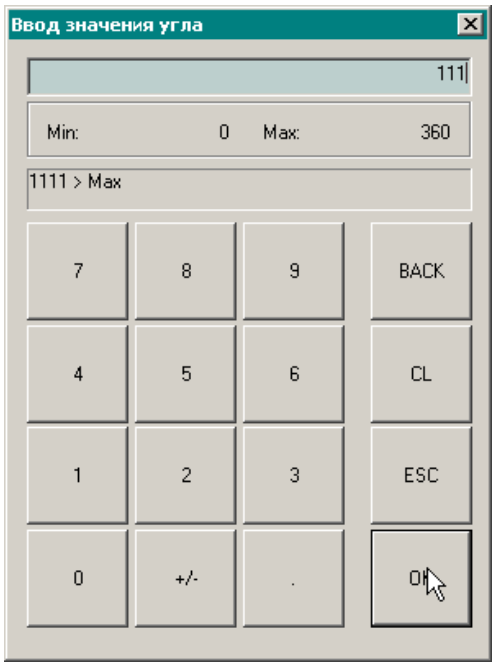


Рис. 83. Ввод информации при помощи Numpad

ВНИМАНИЕ!

В среде исполнения CoDeSys значение переменной, введенное оператором при помощи данного действия, немедленно записывается в память на контексте задачи VISU_INPUT_TASK, что может привести к неправильному выполнению алгоритмов, использующих значение данной переменной на контекстах других циклических задач, вытесненных VISU_INPUT_TASK. В связи с этим переменные, которые могут быть изменены оператором из форм визуализации, должны передаваться в качестве входных в использующие их POU.

9.4.6.7. Ввод информации при помощи системной диалоговой панели Keypad

Элемент формы визуализации может использоваться для ввода текстовых или числовых значений для переменной, заданной в поле **Variables–Textdisplay**, в системной диалоговой панели **Keypad**, показанной на рис. 84.

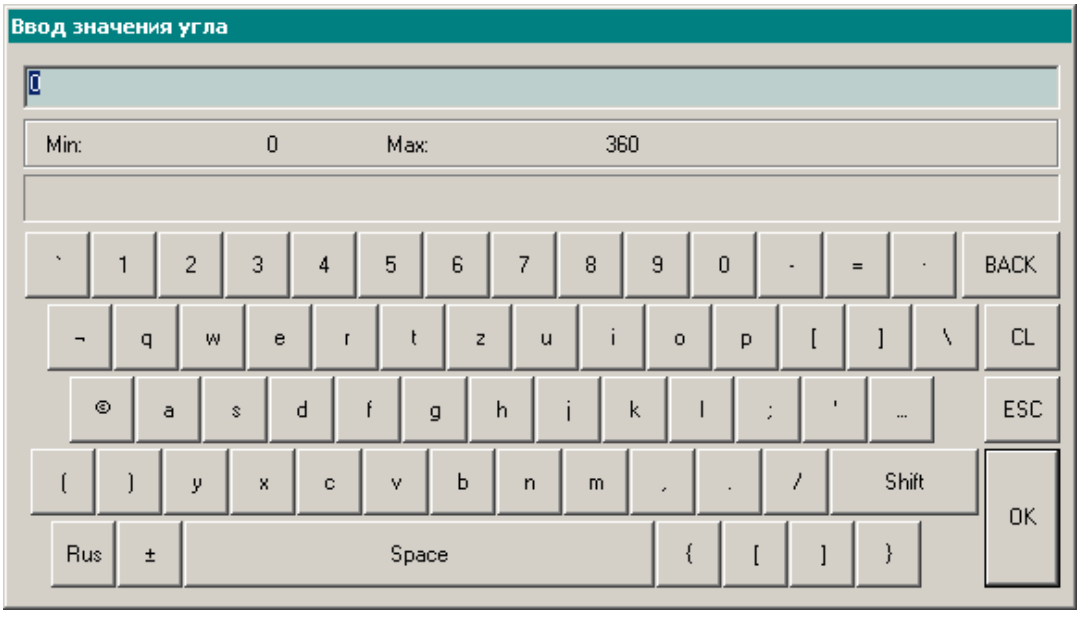


Рис. 84. Ввод информации при помощи Keypad

Для активизации редактирования с помощью системной диалоговой панели **Keypad** необходимо отметить флажок **Input–Text input of variable 'Textdisplay'** и в выпадающим списке под флажком выбрать опцию **Keypad**. Для контроля диапазона вводимых значений следует ввести минимальное и максимальное допустимые значения в поля **Min** и **Max**. Для переменных типа STRING в полях **Min** и **Max** можно ввести минимальное и максимальное допустимые значения длины вводимой строки.

Поле **Dialog title** позволяет задать строку, которая будет отображаться в области заголовка диалоговой панели **Keypad**.

Кнопка **BACK** служит для стирания текущего введенного символа, а кнопка **CL** – для стирания всех введенных символов.

Для ввода набранного значения следует щелкнуть мышью над кнопкой **OK** диалоговой панели, а для скрытия диалоговой панели с отказом от ввода значения служит кнопка **ESC**.

Кнопка **Rus/En** позволяет переключать раскладку клавиатуры **Keypad** на русскоязычную и наоборот.

ВНИМАНИЕ!

В среде исполнения CoDeSys значение переменной, введенное оператором при помощи данного действия, немедленно записывается в память на контексте задачи VISU_INPUT_TASK, что может привести к неправильному выполнению алгоритмов, использующих значение данной переменной на контекстах других циклических задач, вытесненных VISU_INPUT_TASK. В связи с этим переменные, которые могут быть изменены оператором из форм визуализации, должны передаваться в качестве входных в использующие их POU.

9.4.6.8. Выполнение простых выражений с присвоением результата некоторой переменной

Элемент формы визуализации может использоваться для выполнения одного или нескольких выражений на языке ST, изменяющих значение одной или нескольких переменных по щелчку над элементом во время исполнения приложения в контроллере.

Для активизации данного действия:

1. Отметьте флажок **Input–Execute program** и нажмите кнопку **...** справа от сделавшегося доступным для редактирования поля ввода. На экран монитора будет выведена диалоговая панель **Configure programs**, показанная на рис. 85.

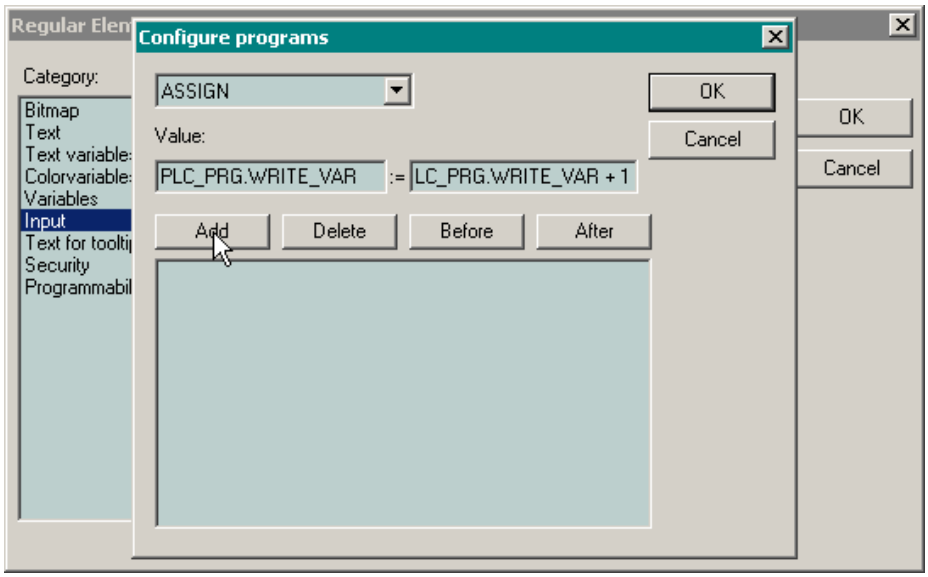


Рис. 85. Настройка действия ASSIGN

2. В выпадающем списке диалоговой панели **Configure programs** выберите действие ASSIGN. Под выпадающим списком станут доступными два поля для ввода выражения, разделенные оператором присваивания.
3. В поле **Value** введите имя переменной, значение которой должно быть вычислено, а справа от оператора присваивания введите выражение на языке ST, посредством которого должно вычисляться значение переменной, после чего нажмите кнопку **Add**. Введенное выражение будет добавлено в список выполняемых действий в качестве параметра системной команды INTERN ASSIGN.
4. При необходимости добавьте другие выражения в отношении той же или других переменных, нажимая кнопку **Add**. При необходимости изменить порядок выполнения команд воспользуйтесь кнопками **Before** (**Перед**) или **After** (**После**), выбирая команды в списке добавленных, чей порядок выполнения должен быть изменен.

ПРИМЕЧАНИЕ. Не злоупотребляйте данным функционалом, поскольку часть логики приложения будет скрыта в коде, автоматически генерируемом для формы визуализации, что затрудняет отладку и поиск ошибок.

5. Закройте диалоговую панель нажатием кнопки **OK**.

ВНИМАНИЕ!

В среде исполнения CoDeSys значение переменной, введенное оператором при помощи данного действия, немедленно записывается в память на контексте задачи VISU_INPUT_TASK, что может привести к неправильному выполнению алгоритмов, использующих значение данной переменной на контекстах других циклических задач, вытесненных VISU_INPUT_TASK. В связи с этим переменные, которые могут быть изменены оператором из форм визуализации, должны передаваться в качестве входных в использующие их POU.

9.4.6.9. Выбор языка интерфейса оператора

Для изменения языка интерфейса оператора во время исполнения приложения в контроллере по щелчку над элементом визуализации в категории его свойств **Input** отметьте флажок **Input–Execute program** и нажмите кнопку **...** справа от сделавшегося доступным для редактирования поля ввода, после чего в выпадающем списке диалоговой панели **Configure programs** выберите команду LANGUAGEDIALOG, нажмите **Add** и закройте диалоговые панели **Configure programs** и **Regular Element Configuration** нажатием кнопок **OK**.

После загрузки приложения в контроллер щелчок мышью над данным элементом визуализации будет приводить к появлению на экране системной диалоговой панели **Change language**, показанной на рис. 79.

9.4.6.10. Переключение языка интерфейса оператора

Для переключения языка интерфейса оператора на требуемый во время исполнения приложения в контроллере по щелчку над элементом визуализации в категории его свойств **Input** отметьте флажок **Input–Execute program** и нажмите кнопку **...** справа от сделавшегося доступным для редактирования поля ввода, после чего в выпадающем списке диалоговой панели **Configure programs** выберите команду LANGUAGE, в поле **Value** введите идентификатор языка (например, RUS или ENG), заданный в файле динамического текста, нажмите **Add** и закройте диалоговые панели **Configure programs** и **Regular Element Configuration** нажатием кнопок **OK**.

После загрузки приложения в контроллер щелчок мышью над данным элементом визуализации будет приводить к переключению языка интерфейса оператора.

9.4.6.11. Запуск внешнего приложения

Для запуска внешнего приложения во время исполнения приложения CoDeSys в контроллере по щелчку над элементом визуализации в категории его свойств **Input** отметьте флажок **Input–Execute program** и введите имя файла внешнего приложения (например, regedit.exe), после чего закройте диалоговую панель **Regular Element Configuration** нажатием кнопок **OK**.

После загрузки приложения в контроллер щелчок мышью над данным элементом визуализации будет приводить к запуску заданного внешнего приложения.

9.4.6.12. Изменение текущего уровня пользователя

Перед использованием возможностей ограничения доступа разных категорий пользователей к элементам форм визуализации во вкладке **Resources** главного окна CoDeSys щелкните на ресурсе **Global Variables** и добавьте следующую секцию энергонезависимых переменных:

```
VAR_GLOBAL RETAIN
   VisuDoExecuteUserlevelInit : BOOL := TRUE;
   CurrentUserLevel : INT := 0;
   CurrentPasswords : ARRAY[0..7] OF STRING[20] :=
   'pwd0', 'pwd1', 'pwd2', 'pwd3', 'pwd4', 'pwd5', 'pwd6', 'pwd7';
END_VAR
```

Переменная **`CurrentUserLevel`** будет определять текущий уровень пользователя, установленный для приложения.

Массив **CurrentPasswords**, состоящий из восьми строк, определяет текущие пароли для каждого из восьми уровней пользователя. В качестве инициализирующих значений для каждого элемента массива задайте желаемые пароли.

Для изменения текущего уровня пользователя по щелчку над элементом визуализации во время исполнения приложения CoDeSys в контроллере в категории его свойств **Input** отметьте флажок **Input–Execute program**, нажмите кнопку ... справа от сделавшегося доступным для редактирования поля ввода, после чего в выпадающем списке диалоговой панели **Configure programs** выберите команду CHANGEUSERLEVEL, нажмите **Add** и закройте диалоговые панели **Configure programs** и **Regular Element Configuration** нажатием кнопок **OK**.

После загрузки приложения в контроллер щелчок мышью над данным элементом визуализации будет приводить к появлению на экране системной диалоговой панели **Change user level**, показанной на рис. 86. Для смены уровня необходимо выбрать требуемый уровень пользователя в выпадающем списке **User Level**, ввести пароль в поле **Password** и нажать **OK**.

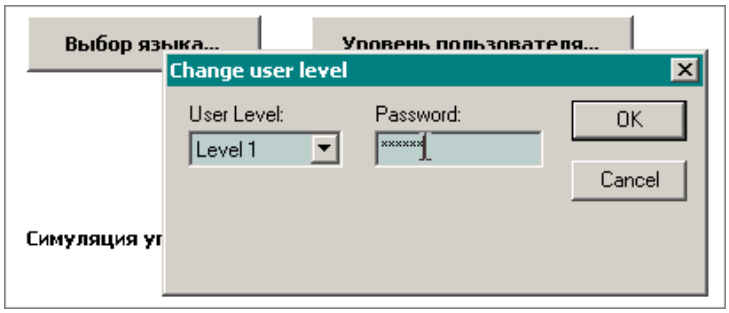


Рис. 86. Смена уровня пользователя во время исполнения приложения в контроллере

После успешной смены уровня пользователя в текущей форме визуализации будут отображены все элементы визуализации, для которых в свойствах **Security** задан уровень пользователя, меньший или равный текущему выбранному.

9.4.6.13. Изменение паролей для уровней пользователей

Перед использованием возможностей ограничения доступа разных категорий пользователей к элементам форм визуализации во вкладке **Resources** главного окна CoDeSys щелкните на ресурсе **Global Variables** и добавьте следующую секцию энергонезависимых переменных:

```
VAR_GLOBAL RETAIN
   VisuDoExecuteUserlevelInit : BOOL := TRUE;
   CurrentUserLevel : INT := 0;
   CurrentPasswords : ARRAY[0..7] OF STRING[20] :=
   'pwd0', 'pwd1', 'pwd2', 'pwd3', 'pwd4', 'pwd5', 'pwd6', 'pwd7';
END_VAR
```

Переменная **CurrentUserLevel** будет определять текущий уровень пользователя, установленный для приложения.

Массив **CurrentPasswords**, состоящий из восьми строк, определяет текущие пароли для каждого из восьми уровней пользователя. В качестве инициализирующих значений для каждого элемента массива задайте желаемые пароли.

Для обеспечения возможности изменения паролей для уровней пользователя во время исполнения приложения CoDeSys в контроллере по щелчку над элементом визуализации в категории его свойств **Input** отметьте флажок **Input–Execute program**, нажмите кнопку ... справа от сделавшегося доступным для редактирования поля ввода, после чего в выпадающем списке диалоговой панели **Configure programs** выберите команду CHANGEPASSWORD, нажмите **Add** и закройте диалоговые панели **Configure programs** и **Regular Element Configuration** нажатием кнопок **OK**.

После загрузки приложения в контроллер щелчок мышью над данным элементом визуализации будет приводить к появлению на экране системной диалоговой панели **Change password of user level**, показанной на рис. 87. Для смены пароля необходимо выбрать требуемый уровень пользователя в выпадающем списке **User Level**, ввести старый пароль в поле **Old Password**, новый пароль в поля **New Password** и **Acknowledge new** и нажать **OK**.

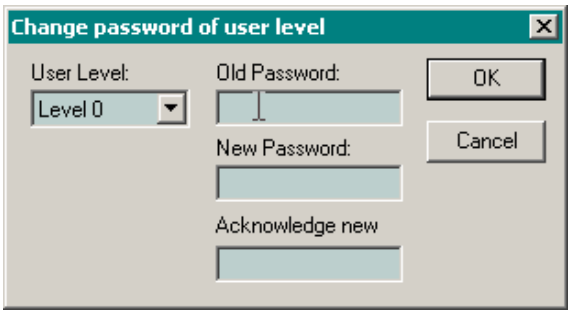


Рис. 87. Смена пароля для уровня пользователя

9.4.7. Отображение графиков и архивация данных

9.4.7.1. Общие сведения

Подсистема целевой визуализации контроллера поддерживает элемент визуализации типа Тренд, предназначенный для отображения графических зависимостей значений переменных приложения от времени. Данный элемент также позволяет выполнять архивацию значений переменных с возможностью просмотра содержимого архива, если в свойствах платформы отмечен флажок **Visualization–Store trend data in PLC**.

При использовании элемента Тренд учитывайте следующие особенности его реализации в подсистеме целевой визуализации CoDeSys:

1. Элемент визуализации Тренд, посредством которого предполагается отображать график значения одной переменной, занимает в сегменте кода до 33 кбайт.
2. Отображение элемента визуализации Тренд требует большого количества системных ресурсов, что может негативно сказываться на производительности системы.
3. Архивация значений переменной средствами элемента визуализации Тренд выполняется на контексте автоматически создаваемой циклической задачи TREND_TASK, период цикла которой по умолчанию устанавливается равным 200 мс. По возможности увеличьте период задачи TREND_TASK до 500-1000 мс для уменьшения количества обращений к файловой системе контроллера.

В настоящем подразделе приведены краткие указания по настройке и использованию элемента визуализации Тренд на примере небольшого приложения, в котором выполняются отображение графика и архивация значений одной переменной на съемном USB-накопителе.

9.4.7.2. Создание приложения с поддержкой целевой визуализации

После выполнения приведенных ниже указаний в среде разработки CoDeSys будет создан проект для контроллера MK905 с поддержкой целевой визуализации. Проект будет содержать единственную программную единицу PLC_PRG, ассоциированную с циклической задачей PLC_PRG_TASK (приоритет – 1, период – 10 мс). В конфигурацию задач проекта будут автоматически добавлены три задачи: VISU_TASK, VISU_INPUT_TASK и TREND_TASK. Кроме того, к проекту будут автоматически подключены следующие системные библиотеки: FastwelSysLibFile.lib, HMI\SysLibMem.lib, HMI\SysLibTime.lib, SysLibTargetVisu.lib, SysLibAlarmTrend.lib, Standard.lib м SysLibCallback.lib.

1. Запустите среду разработки CoDeSys и выберите команду **File–New** в главном меню.
2. В выпадающем списке **Configuration** появившейся диалоговой панели **Target Settings** выберите опцию *Fastwel CPB902 WinCE Runtime*.
3. Щелкните на вкладке **Visualization** диалоговой панели **Target Settings** и отметьте флажки **Target visualization** и **Store trend data in PLC**, как показано на рис. 88, после чего закройте диалоговую панель **Target Settings** нажатием кнопки **OK**.
4. Нажмите кнопку **OK** в появившейся диалоговой панели **New POU**. Главное окно среды разработки CoDeSys примет вид, показанный на рис. 89.
5. В главном меню CoDeSys выберите команду **File–Save**, после чего в поле **File name** (**Имя файла**) диалоговой панели **Save as** (**Сохранить как**) введите имя файла проекта *MK905_TrendLogging* и нажмите кнопку **Save** (**Сохранить**).

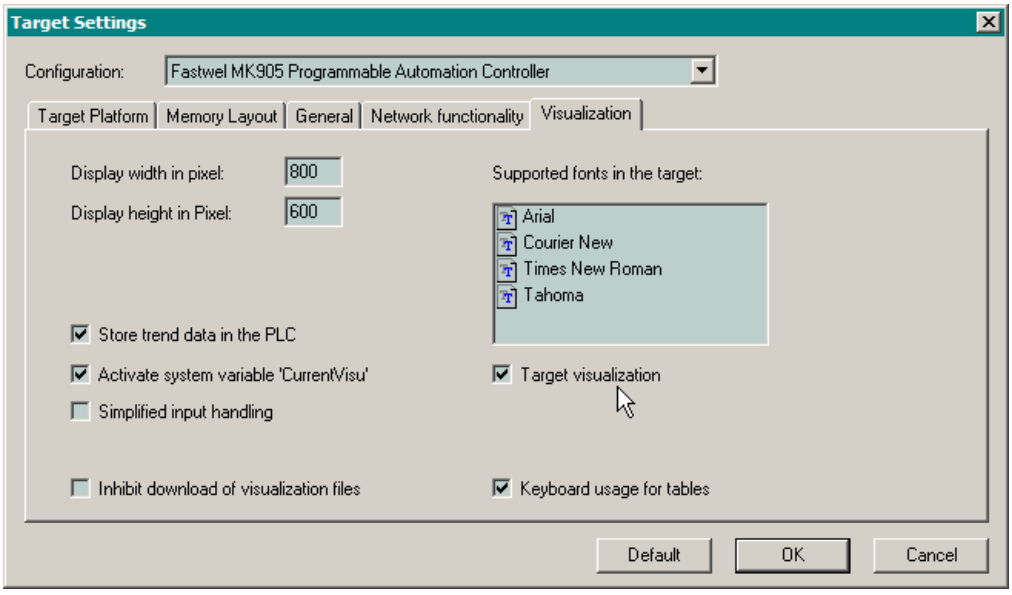


Рис. 88. Настройка параметров платформы для создаваемого проекта

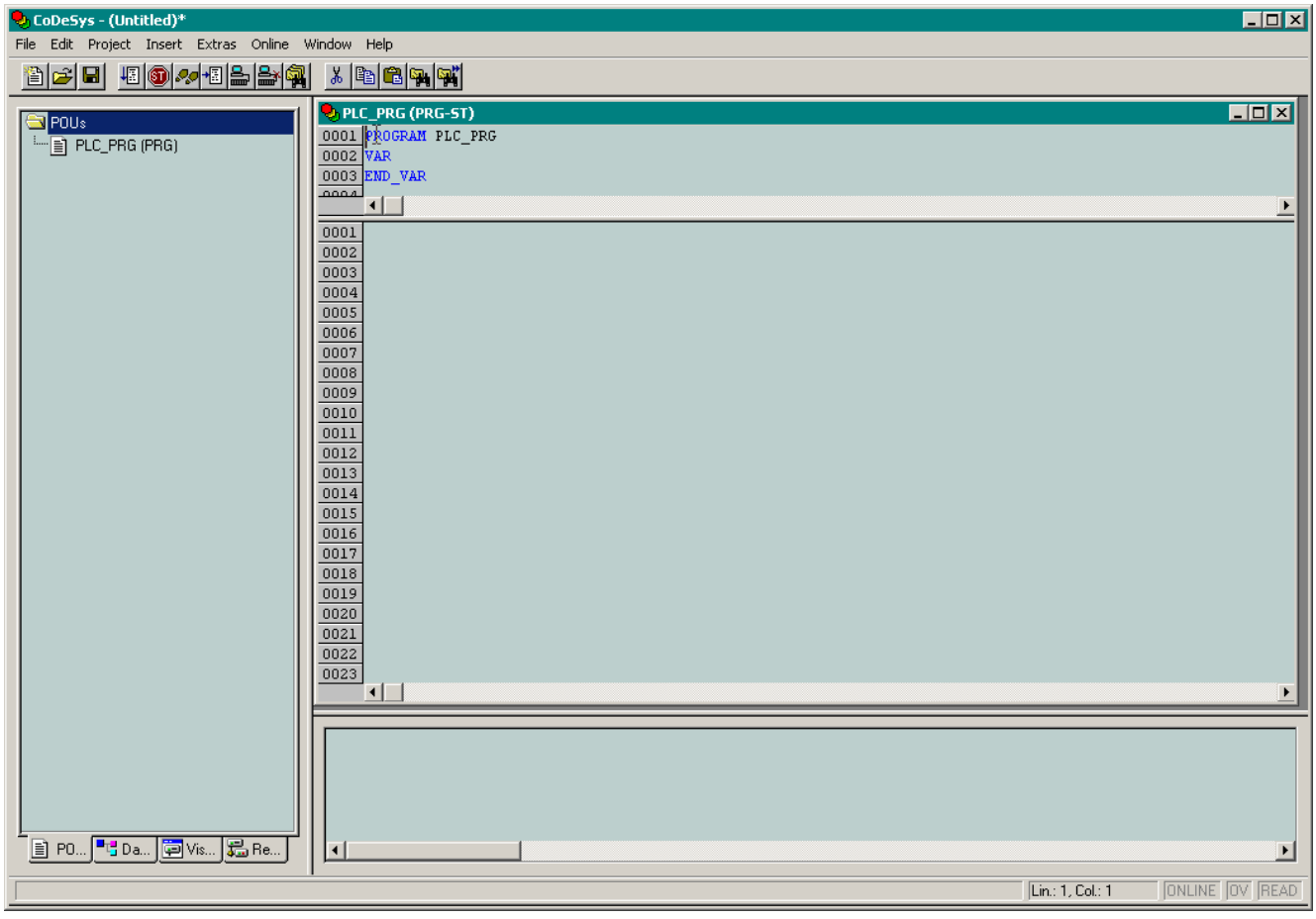


Рис. 89. Внешний вид главного окна CoDeSys после создания проекта

9.4.7.3. Разработка программы

Программа PLC_PRG будет исполняться на контексте циклической PLC_PRG_TASK с периодом 10 мс и приоритетом 32 и выполнять следующие действия:

1. Циклически увеличивать на 1 значение некоторой переменной в диапазоне от 0 до 359.
2. Формировать статусную информацию о наличии присоединенного USB-накопителя для отображения в окне формы визуализации.
3. Формировать статусную информацию о наличии на присоединенном USB-накопителя каталога, в который будут помещаться файлы архивов.

В области декларации переменных программы PLC_PRG введите объявления следующих констант:

```
VAR CONSTANT

   TrendsStorage : STRING := '\USB Storage';
   (* Имя USB-накопителя *)

   DiskExistLabel : STRING := 'подключен';
   (* Строка для отображения статуса наличия диска *)

   DiskAbsentLabel : STRING := 'не подключен';
   (* Строка для отображения статуса отсутствия диска *)

   TrendsStoragePath : STRING := '\USB Storage\TestDir';
   (* Путь к каталогу архивации на USB-накопителе *)

   LoggingPathAvailableLabel : STRING := 'доступен';
   (* Строка для отображения статуса наличия каталога архивации *)

   LoggingPathNotAvailableLabel : STRING := 'не доступен';
   (* Строка для отображения статуса отсутствия каталога архивации *)

END_VAR
```

В области декларации переменных программы PLC_PRG введите объявления следующих переменных:

```
VAR

   VarForTrend : WORD := 0;
   (* Переменная для отображения на графике и архивации *)

   StorageExist : BOOL := FALSE;
   (* Признак наличия присоединенного USB-накопителя *)

   PathExist : BOOL := FALSE;
   (* Признак наличия каталога архивации на присоединенном USB-накопителе *)

   StorageStatusLabel : STRING;
   (* Строка для надписи статуса наличия диска *)

   StorageStatusColor : DWORD := 16#FFFFFF;
   (* Цвет надписи статуса наличия диска, по умолчанию -- белый *)

   LoggingStatusLabel : STRING;
   (* Строка для надписи статуса наличия каталога архивации *)

   LoggingStatusColor : DWORD := 16#FFFFFF;
   (* Цвет надписи статуса наличия каталога архивации, по умолчанию -- белый *)

END_VAR
```

В области тела PLC_PRG введите следующий исходный текст:

```
(* На каждом цикле PLC_PRG_TASK увеличиваем значение VarForTrend на 1. *)
(* По достижении 360 значение возвращается к 0. *)
VarForTrend := (VarForTrend + 1) MOD 360;

(* Устанавливаем признак наличия присоединенного USB-накопителя *)
StorageExist := FwSysDirExist(TrendsStorage) <> 0;

IF StorageExist THEN

   (* Если накопитель присоединен, надпись статуса -- 'подключен' *)
   StorageStatusLabel := DiskExistLabel;

   (* Надпись статуса накопителя будет иметь зеленый цвет *)
   StorageStatusColor := 16#00_FF_00;

   (* Устанавливаем признак наличия каталога архивации *)
   PathExist := FwSysDirExist(TrendsStoragePath) <> 0;
```

```
    IF PathExist THEN

       (* Если каталог архивации найден, надпись статуса -- 'доступен' *)
       LoggingStatusLabel := LoggingPathAvailableLabel;

       (* Надпись статуса каталога будет иметь зеленый цвет *)
       LoggingStatusColor := 16#00_FF_00;

    ELSE

       (* Если каталог архивации не найден, надпись статуса -- 'не доступен' *)
       LoggingStatusLabel := LoggingPathNotAvailableLabel;

       (* Надпись статуса каталога будет иметь красный цвет *)
       LoggingStatusColor := 16#00_00_FF;

    END_IF

ELSE

    (* Если накопитель не присоединен, надпись статуса -- 'не подключен' *)
    StorageStatusLabel := DiskAbsentLabel;

    (* Надпись статуса накопителя будет иметь красный цвет *)
    StorageStatusColor := 16#00_00_FF;

    (* Надпись статуса каталога архивации будет содержать строку 'не доступен' *)
    LoggingStatusLabel := LoggingPathNotAvailableLabel;

    (* Надпись статуса каталога будет иметь красный цвет *)
    LoggingStatusColor := 16#00_00_FF;

END_IF
```

Для установки приоритета задачи PLC_PRG_TASK:

1. Щелкните на вкладке **Resources** в левой области главного окна CoDeSys и дважды щелкните на ресурсе **Task configuration**.
2. В появившемся окне щелкните на элементе дерева PLC_PRG_TASK и во вкладке свойств задачи **Task Attributes** установите значение параметра **Priority(1..32)** равным 32, что соответствует максимальному приоритету.

9.4.7.4. Создание формы визуализации и добавление элемента Тренд

После выполнения приведенных ниже указаний в проект будет добавлена форма визуализации и на форме будет размещен элемент визуализации Тренд.

1. Щелкните на вкладке **Visualizations** в левой области главного окна CoDeSys, после чего щелкните правой кнопкой мыши на корневым элементом древовидного списка форм визуализации данного проекта и выберите команду **Add Object** в появившемся контекстном меню.
2. Введите имя Trender для создаваемой формы визуализации в поле **Name of the new visualization** появившейся диалоговой панели **New Visualization** и нажмите клавишу Enter (или щелкните **OK** в диалоговой панели **New Visualization**). В правой области главного окна CoDeSys появится пустое окно формы визуализации.
3. Выберите команду **Insert–Trend** в главном меню или щелкните на пиктограмме  в панели инструментов CoDeSys и нарисуйте элемент визуализации Тренд, как показано на рис. 90. Для этого нажмите левую кнопку мыши в точке формы, где должен быть расположен верхний левый угол элемента, после чего, оставив левую кнопку мыши нажатой, переместите курсор в точку, в которой должен находиться правый нижний угол элемента, и отпустите левую кнопку мыши.

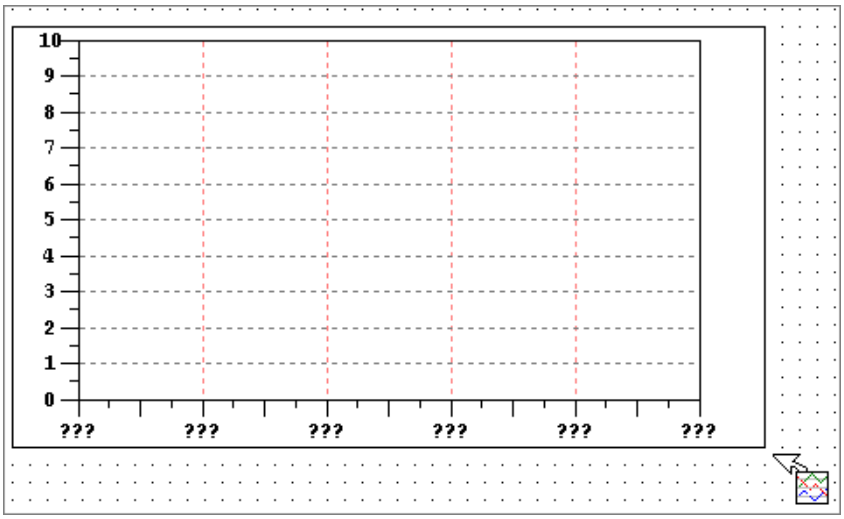


Рис. 90. Рисование элемента визуализации Тренд

9.4.7.5. Настройка элемента Тренд

После выполнения приведенных ниже указаний элемент визуализации Тренд, добавленный в проект при выполнении указаний п. 9.4.7.4, будет настроен на отображение графика значений переменной PLC_PRG.VarForTrend.

1. Дважды щелкните левой кнопкой мыши над нарисованным элементом Тренд. На экран монитора будет выведена диалоговая панель **Regular Element Configuration**, показанная на рис. 91.

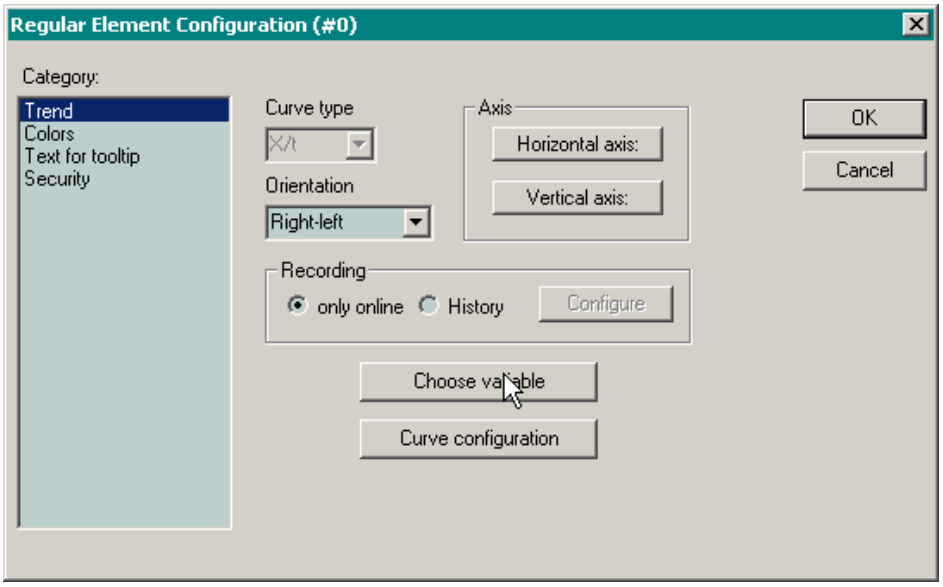


Рис. 91. Диалоговая панель настройки параметров элемента визуализации Тренд

2. Для выбора переменной, чьи значения должны отображаться на графике, нажмите кнопку **Choose variable** в категории параметров **Trend** и в появившейся диалоговой панели **Variables** щелкните над ячейкой таблицы переменных, после чего введите имя переменной PLC_PRG.VarForTrend или нажмите F2 и выберите ее в окне **Input assistant**.

   Щелчок над ячейкой **Color** справа от имени переменной позволяет выбрать цвет линии, которой будет отображаться график ее значений.

   Выпадающий список **Linetype** позволяет выбрать тип линии. Обратите внимание, что операционная система контроллера MK905 поддерживает отображение только сплошных (*Normal*) и штриховых (*Dashed*) линий.

   Ячейка **Marker** в подсистеме целевой визуализации не используется.

   Закройте диалоговую панель **Variables** нажатием кнопки **OK**.

3. Для настройки параметров горизонтальной оси графика нажмите кнопку **Axis–Horizontal axis** в категории параметров **Trend**. На экран монитора будет выведена диалоговая панель **Horizontal axis**, показанная на рис. 92.

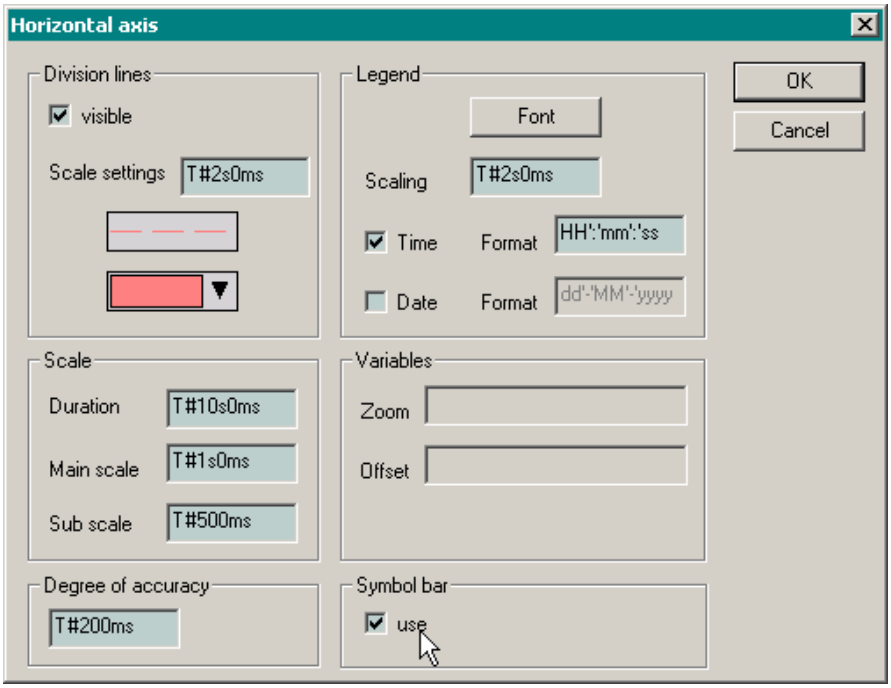


Рис. 92. Настройка параметров горизонтальной оси

Группа параметров **Division lines** позволяет настроить свойства линий сетки графика для оси времени. При необходимости отобразить линии сетки отметьте флажок **visible** и в поле **Scale settings** задайте интервал времени между соседними линиями.

Щелчок мышью над полем  позволяет выбрать тип линий сетки для оси времени. Обратите внимание, что операционная система контроллера MK905 поддерживает отображение только сплошных (*Normal*) и штриховых (*Dashed*) линий.

Щелчок мышью над полем  позволяет выбрать цвет линий сетки для оси времени.

Группа параметров **Scale** позволяет задать полный диапазон горизонтальной оси графика (поле **Duration**) и шаг основных (**Main scale**) и дополнительных (**Sub scale**) делений, отображаемых на горизонтальной оси.

Поле **Degree of accuracy** предназначено для установки дискретности отображаемых значений. Данное значение не рекомендуется устанавливать меньшим, чем период задачи TREND_TASK.

Группа параметров **Legend** позволяет настроить свойства надписей для горизонтальной оси, включая тип, цвет и размер шрифта (**Font**), шаг отображения надписей и формат надписей.

Группа **Variables**, доступная для редактирования при снятом флажке **Symbol bar–use**, позволяет выбрать переменные приложения, значения которых будут определять масштаб отображения графика и сдвиг начала горизонтальной оси.

Флажок **Symbol bar–use** позволяет отобразить набор кнопок управления масштабированием и сдвигом начала горизонтальной оси.

Установите значения параметров горизонтальной оси, как показано на рис. 92, после чего закройте диалоговую панель **Horizontal axis** нажатием кнопки **OK**.

4. Для настройки параметров вертикальной оси графика нажмите кнопку **Axis–Vertical axis** в категории параметров **Trend**. На экран монитора будет выведена диалоговая панель **Vertical axis**, показанная на рис. 93.

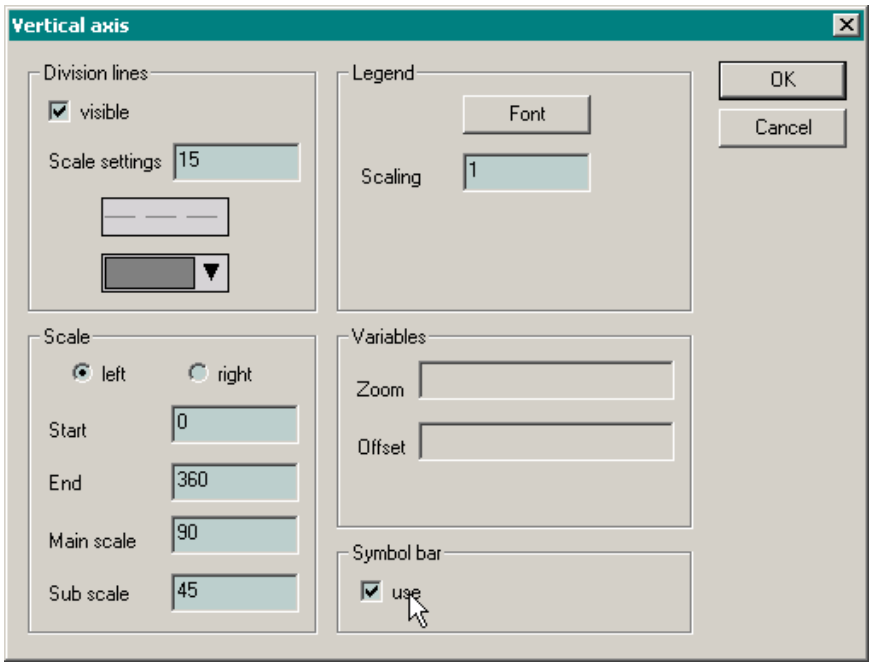


Рис. 93. Настройка параметров вертикальной оси

Группа параметров **Division lines** позволяет настроить свойства линий сетки графика для вертикальной оси. При необходимости отобразить линии сетки отметьте флажок **visible** и в поле **Scale settings** задайте интервал времени между соседними линиями.

Щелчок мышью над полем  позволяет выбрать тип линий сетки для вертикальной оси. Обратите внимание, что операционная система контроллера MK905 поддерживает отображение только сплошных (*Normal*) и штриховых (*Dashed*) линий.

Щелчок мышью над полем  позволяет выбрать цвет линий сетки для вертикальной оси.

Группа параметров **Legend** позволяет настроить свойства надписей для вертикальной оси, включая тип, цвет и размер шрифта (**Font**), а также шаг отображения надписей.

Группа **Variables**, доступная для редактирования при снятом флажке **Symbol bar–use**, позволяет выбрать переменные приложения, значения которых будут определять масштаб отображения графика и сдвиг начала вертикальной оси.

Флажок **Symbol bar–use** позволяет отобразить набор кнопок управления масштабированием и сдвигом начала вертикальной оси.

Группа параметров **Scale** служит для настройки полного диапазона вертикальной оси (поле **Start** – для начального значения; поле **End** – для конечного значения) и шага основных (**Main scale**) и дополнительных (**Sub scale**) делений, отображаемых на вертикальной оси.

Переключатель **left/right** позволяет выбрать сторону (левая или правая), с которой будут нанесены деления и надписи вертикальной оси.

Установите значения параметров вертикальной оси, как показано на рис. 93, задайте шаг нанесения надписей **Legend–Scaling**, равным 20, после чего закройте диалоговую панель **Horizontal axis** нажатием кнопки **OK**.

9.4.7.6. Настройка параметров архивации

Для настройки параметров архивации значений переменной в категории параметров **Trend** установите переключатель **Recording** в положение **History** и нажмите кнопку **Configure**. На экран монитора будет выведена диалоговая панель **Configure database**, показанная рис. 94. Если переключатель **Recording** в категории параметров **Trend** установлен в положение **only online**, то архивация выполняться не будет.

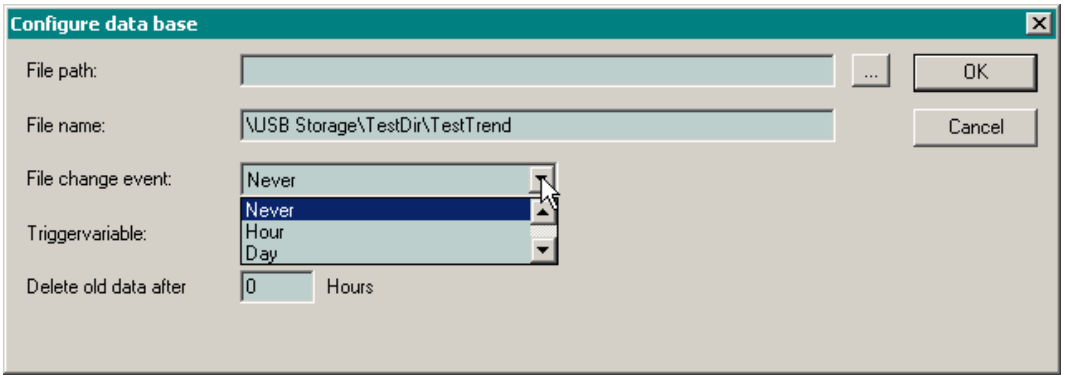


Рис. 94. Настройка параметров архивации

Оставьте пустым поле **File path**, поскольку оно не используется.

Поле **File name** служит для выбора префикса имени файлов архива, а также для установки пути к файлам архива.

Введите в поле **File name** абсолютный путь \USB Storage\TestDir\TestTrend. Тогда во время исполнения при подключении съемного USB-накопителя, в корневом каталоге которого имеется подкаталог TestDir, в данном подкаталоге будет создан файл архива с префиксом имени TestTrend и расширением trd.

Если в поле **File name** введено имя файла без разделителей пути '\', файлы архива будут создаваться в каталоге пользователя user. В контроллере MK905 полный путь к каталогу пользователя: \FixedDisk1\user. Например, если ввести в поле **File name** префикс имени TestTrend, то в каталоге пользователя user (полный путь: \ FixedDisk1\user) в процессе работы приложения будет создан файл с префиксом TestTrend и расширением trd.

Если в поле **File name** введено имя файла с относительным путем (например, TestFolder\TestTrend), файлы архива будут создаваться в соответствующем подкаталоге каталога user, если данный подкаталог присутствует в каталоге user. Более подробная информация об относительном пути приведена в п. 6.2.1.2 настоящего руководства.

Если в поле **File name** введено имя файла с полным путем (например, \USB Storage\TestFolder\TestTrend), файлы архива будут создаваться по соответствующего абсолютному пути, если все подкаталоги по данному пути присутствуют до запуска архивации. Более подробная информация об абсолютном пути приведена в п. 6.2.1.3 настоящего руководства.

ВНИМАНИЕ! При наличии в относительном или абсолютном пути к файлу хотя бы одного подкаталога соответствующие подкаталоги должны физически присутствовать на целевом накопителе.

Выпадающий список **File change event** позволяет выбрать алгоритм смены файлов архива во время выполнения приложения:

Never – запись архивных данных будет производиться в один и тот же файл. Использование данной опции означает потенциально неограниченный рост размера одного файла архива. Скорость роста размера при периоде задачи TREND_TASK, равном 200 мс, для значения одной архивируемой переменной составит около 300-360 байт/с. При использовании данной опции возможно организовать автоматическую выгрузку и удаление единственного файла архива при помощи удаленного ftp-клиента, в результате чего после удаления будет автоматически создаваться новый файл архива с тем же именем.

Hour – новый файл архива с заданным префиксом имени будет создаваться с периодичностью, равной одному часу. Предыдущий файл архива при этом будет удален. При периоде задачи TREND_TASK, равном 200 мс, для значения одной архивируемой переменной размер каждого файла будет составлять около 1,2 Мбайт.

Day – новый файл архива с заданным префиксом имени будет создаваться с периодичностью, равной одним суткам. Предыдущий файл архива при этом будет удален.

Week – новый файл архива с заданным префиксом имени будет создаваться с периодичностью, равной одной неделе. Предыдущий файл архива при этом будет удален. При периоде задачи TREND_TASK, равном 200 мс, для значения одной архивируемой переменной размер каждого файла будет составлять около 207 Мбайт. В связи с этим по возможности избегайте использовать данную опцию.

Month – новый файл архива с заданным префиксом имени будет создаваться с периодичностью, равной одному месяцу. Предыдущий файл архива при этом будет удален. **Не рекомендуется использовать данную опцию.**

Variable – новый файл архива с заданным префиксом имени будет создаваться по переднему фронту булевой переменной, имя которой задано в поле **Triggervariable**. Предыдущий файл архива при этом будет удален.

Records – файл архива с заданным префиксом имени будет создаваться, когда количество записей в предыдущем файле достигло значения, заданного в поле **Number of records**. Предыдущий файл архива при этом будет удален.

Оставьте неизменной (*Never*) опцию в выпадающем списке **File change event**.

Поле **Delete old data after** позволяет задать период времени в часах, с которым будет удалятся предыдущий файл архива.

9.4.7.7. Создание дополнительных элементов визуализации

После выполнения приведенных ниже указаний под элементом Тренд будут созданы две надписи. Первая надпись предназначена для отображения статуса съемного USB-накопителя:

если во время исполнения приложения USB-накопитель подключен к контроллеру, надпись имеет зеленый цвет фона и отображает сообщение «USB-диск подключен»;

если USB-накопитель не подключен, надпись имеет красный цвет фона и отображает сообщение «USB-диск не подключен».

Вторая надпись служит для отображения статуса наличия пути к файлам архива на съемном USB-накопителе: если во время исполнения приложения USB-накопитель подключен к контроллеру и на нем имеется каталог TestDir, надпись имеет зеленый цвет и отображает сообщение «Путь к архиву доступен». В противном случае надпись имеет красный цвет фона и отображает сообщение «Путь к архиву не доступен».

Внешний вид формы визуализации после добавления надписей показан на рис. 95.

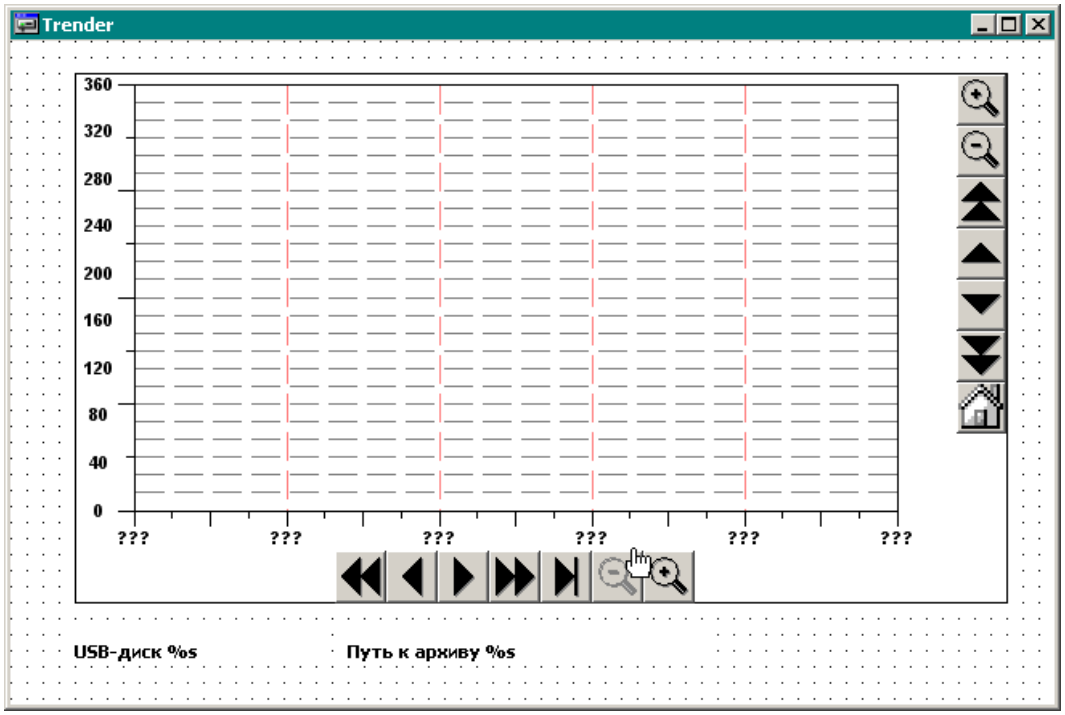


Рис. 95. Статусные надписи

Для добавления надписи статуса USB-накопителя:

1. Выберите команду **Insert–Rectangle** в главном меню или щелкните на пиктограмме в панели инструментов, после чего нарисуйте прямоугольник в окне редактирования формы визуализации слева под элементом Тренд.
2. Дважды щелкните над вновь созданным прямоугольником и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration.**

Установите выравнивание текста по левой границе, для чего установите переключатель **Horizontal–Left**.

3. Нажмите кнопку **Default font**, после чего нажмите кнопку **Font** и выберите требуемый тип, размер и оформление шрифта, которым будет отображаться надпись.
4. В поле **Content** введите надпись *USB-диск %s*, которая означает, что во время исполнения приложения вместо спецификатора %s будет отображаться значение строковой переменной, заданной в группе свойств **Variables–Textdisplay**.
5. Перейдите в категорию свойств **Variables** и в поле **Textdisplay** введите имя переменной PLC_PRG.StorageStatusLabel (при этом можно воспользоваться помощником ввода, нажав клавишу F2).
6. Перейдите в категорию свойств **Colors** и установите флажок **Color–No frame color**.
7. Перейдите в категорию свойств **Colorvariables** и в поле **Variables for colors–Fill color** введите имя переменной PLC_PRG.StorageStatusColor (при этом можно воспользоваться помощником ввода, нажав клавишу F2), после чего закройте диалоговую панель, нажав кнопку **OK**.

Для добавления надписи статуса целевого каталога архива на USB-накопителе:

1. Выберите команду **Insert–Rectangle** в главном меню или щелкните на пиктограмме  в панели инструментов, после чего нарисуйте прямоугольник в окне редактирования формы визуализации под элементом Тренд справа от надписи статуса USB-накопителя.
2. Дважды щелкните над вновь созданным прямоугольником и выберите категорию свойств **Text** в появившейся диалоговой панели **Regular Element Configuration**. Установите выравнивание текста по левой границе, для чего установите переключатель **Horizontal–Left**.
3. Нажмите кнопку **Default font**, после чего нажмите кнопку **Font** и выберите требуемый тип, размер и оформление шрифта, которым будет отображаться надпись.
4. В поле **Content** введите надпись *Путь к архиву %s*, что означает, что во время исполнения приложения вместо спецификатора %s будет отображаться значение строковой переменной, заданной в группе свойств **Variables–Textdisplay**.
5. Перейдите в категорию свойств **Variables** и в поле **Textdisplay** введите имя переменной PLC_PRG. LoggingStatusLabel (при этом можно воспользоваться помощником ввода, нажав клавишу F2).
6. Перейдите в категорию свойств **Colors** и установите флажок **Color–No frame color**.
7. Перейдите в категорию свойств **Colorvariables** и в поле **Variables for colors–Fill color** введите имя переменной PLC_PRG. LoggingStatusColor (при этом можно воспользоваться помощником ввода, нажав клавишу F2), после чего закройте диалоговую панель, нажав кнопку **OK**.

9.4.7.8. Установка цвета фона формы визуализации

Редактор форм визуализации CoDeSys не позволяет явно установить цвет фона окна формы. Для установки цвета фона следует в качестве нулевого элемента визуализации использовать прямоугольник требуемого цвета, ширина и высота которого совпадают с шириной и высотой формы визуализации.

1. Выберите команду **Insert–Rectangle** в главном меню CoDeSys или щелкните на пиктограмме  в панели инструментов и нарисуйте прямоугольник небольших размеров в верхнем левом углу формы, для чего нажмите левую кнопку мыши в точке формы, где должен быть расположен верхний левый угол прямоугольника, после чего, оставив левую кнопку мыши нажатой, переместите курсор в точку, в которой должен находиться правый нижний угол прямоугольника, и отпустите левую кнопку мыши.
2. В главном меню CoDeSys выберите команду **Extras–Elementlist**. На экран монитора будет выведена диалоговая панель **Element list**, показанная на рис. 96.
3. Щелкните на последнем элементе Rectangle в отображаемом списке, который представляет только что нарисованный прямоугольник, и нажмите кнопку **To back**. Элемент будет перемещен в начало списка, т.е. станет первым в списке отображаемых элементов.

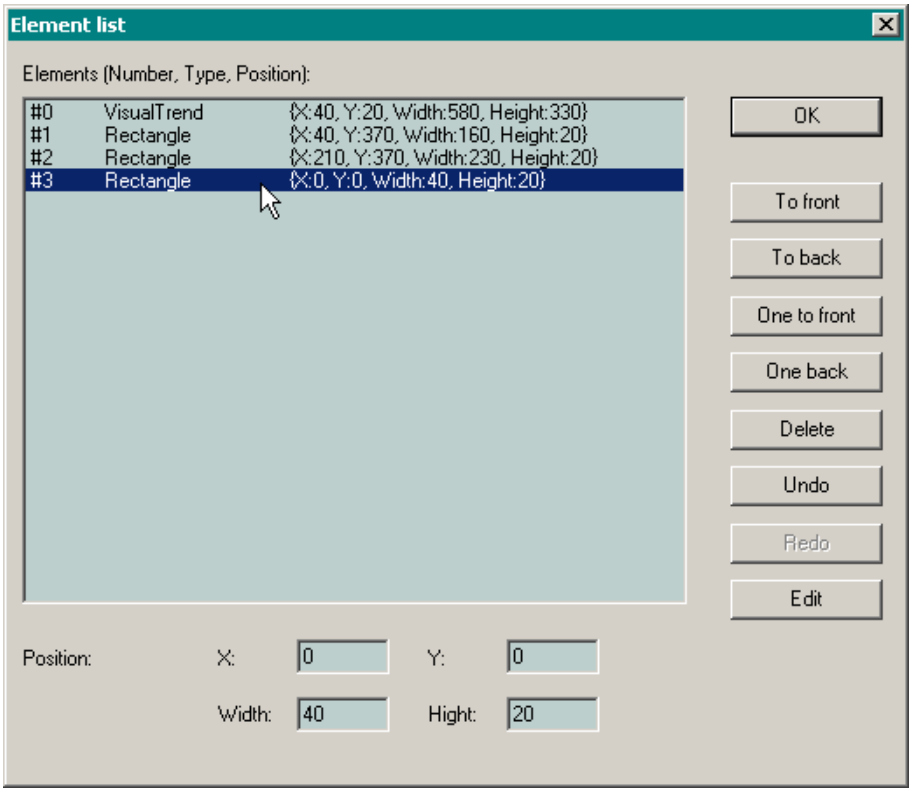


Рис. 96. Диалоговая панель отображения списка элементов визуализации

4. В группе параметров **Position** установите **X** и **Y** равными 0, а **Width** и **Height** сделайте равными ширине и высоте формы визуализации (по умолчанию 800 и 600), после чего закройте диалоговую панель нажатием кнопки **OK**.
5. Дважды щелкните над прямоугольником, в появившейся диалоговой панели **Regular Element Configuration** выберите категорию свойств **Color**, отметьте флажок **No frame color**, а затем, нажав кнопку **Color–Inside**, выберите требуемый цвет фона (скажем, серый), после чего закройте диалоговую панель **Regular Element Configuration** нажатием кнопки **OK**.

9.4.7.9. Загрузка приложения в контроллер

При необходимости последующей выгрузки файлов архива из контроллера по протоколу ftp в соответствии с указаниями п. 8.2, настройте параметры сетевых интерфейсов контроллера в окне **PLC Configuration**, после чего загрузите проект в контроллер в соответствии с указаниями п. 5.8.2 или п. 7.7.

После загрузки основных секций приложения в контроллер будут загружены файлы, относящиеся к подсистеме визуализации, включая файлы изображений, отображаемых на кнопках панелей инструментов элемента Тренд, а также файл textlistheader.thl, содержащий информацию о динамическом тексте, и файл, имя которого сформировано из имени файла проекта, суффикса vis, имеющего расширение txt.

Внешний вид фрагмента формы визуализации, отображаемой во время исполнения приложения на экране монитора, подключенного к контроллеру, представлен на рис. 97.

9.4.7.10. Управление элементом Тренд во время исполнения приложения

Кнопки  и  панелей инструментов горизонтальной и вертикальной осей позволяют масштабировать отображение графика по горизонтали и вертикали соответственно.

Кнопки  и  панели инструментов вертикальной оси предназначены для задания положительного или отрицательного сдвига начальной координаты вертикальной оси.

Кнопки  и  панели инструментов вертикальной оси позволяют выполнить сдвиг начальной координаты вертикальной оси на диапазон вверх или вниз.

Кнопка  панели инструментов вертикальной оси предназначена для восстановления в исходное положение сдвинутой координаты вертикальной оси.

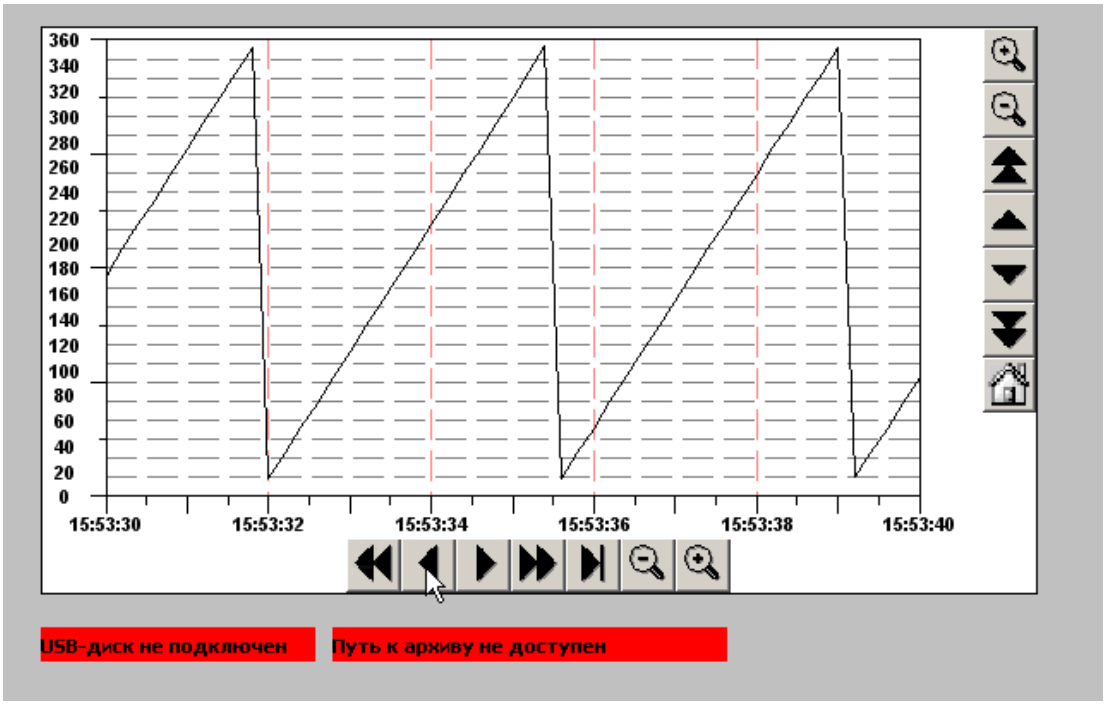


Рис. 97. Диалоговая панель отображения списка элементов визуализации

Кнопки с одинарными и двойными стрелками панели инструментов горизонтальной оси предназначены для отрицательного и положительного сдвига начала горизонтальной оси и функциональны в случае, если выполняется архивация данных.

На инструментальном ПК подготовьте USB-накопитель, для чего создайте в его корневом каталоге подкаталог TestDir, а затем присоедините USB-накопитель к одному из USB-гнезд контроллера. Цвета надписей под элементом Тренд формы визуализации сменят красный цвет на фона на зеленый и будут отображать надписи «USB-диск подключен» и «Путь к архиву доступен». После этого появится возможность выполнять сдвиг начала горизонтальной оси элемента Тренд. При этом кнопка  позволяет восстановить в исходное положение координату ранее сдвинутого начала горизонтальной оси.

Файлы архива содержат записи вида:

```
1282003200;58284703;17.08.2010;16:11:24;PLC_PRG.VarForTrend;141;
```

Первые два поля являются датой (DATE) и временем (TIME) записи, после чего следуют даты, времена, имена и значения переменных в данной записи. Значения представляются в формате с плавающей точкой одинарной точности.

Для безопасного отсоединения USB-накопителя во время работы контроллера нажмите комбинацию клавиш Shift–Escape (возможно, два раза), затем нажмите кнопку **Stop** в появившемся окне консоли контроллера, отсоедините USB-накопитель, после чего нажмите кнопку **Start**.

Имя файла архива состоит из префикса, заданного в поле **File name** диалоговой панели **Configure database**, показанной на рис. 94, числового суффикса и расширения trd. Для того, чтобы числовой суффикс следовал с 0, необходимо в окне **Library Manager** удалить автоматически добавленную библиотеку SysLibAlarmTrend.lib, после чего тут же добавить ее вручную, в результате чего она окажется первой в списке библиотек, подключенных к проекту.

10. Реализация прикладных алгоритмов на C/C++ и расширение системы исполнения приложений CoDeSys

10.1. Общие сведения

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

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

DLL создается при помощи средств разработки Microsoft семейства Visual Studio или Microsoft eMbedded Visual C++ 4.0 SP4 с использованием SDK для образа операционной системы Windows CE 5.0, загруженной в MK905.

DLL размещается в корневом каталоге CompactFlash-диска контроллера либо путем загрузки из среды CoDeSys (с предварительной упаковкой в zip-архив и переименованием файла архива в norm.dnl), либо по протоколу ftp в соответствии с указаниями п. 8.2 настоящего руководства.

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

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

10.2. Требования к рабочему месту разработчика

10.2.1. Требования к конфигурации программного обеспечения

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

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

10.2.2. Средства разработки для Windows CE 5.0

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

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

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

Среда разработка Microsoft eMbedded Visual C++ 4.0, указания по установке, включая информацию о ключе продукта, и пакеты обновлений могут быть загружены с Web-узла компании Microsoft.

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

Кроме того, на инструментальном ПК должен быть установлен пакет программной поддержки MK905 для Windows CE 5.0 (Windows CE 5.0 SDK) – MK905_WinCE500_SDK.msi, который можно загрузить по адресу:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/SDK/MK905

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

"C:\Program Files\Windows CE Tools\wce500\Fastwel MK905 Programmable Automation Controller"

10.3. Взаимодействие системы исполнения контроллера с DLL пользователя

10.3.1. Общие сведения

Расширение системы исполнения контроллера осуществляется путем размещения DLL с именем FwCdsUsr.dll в корневом каталоге NAND Flash-диска контроллера (абсолютный путь \FixedDisk1). DLL должна быть построена для целевой операционной системы Windows CE 5.0 с использованием SDK для MK905.

Файл FwCdsUsr.dll может быть скопирован на NAND Flash-диск MK905 одним из следующих способов:

1. Файл FwCdsUsr.dll упаковывается на инструментальном ПК в zip-архив, файл архива переименовывается в norm.dnl, после чего полученный файл norm.dnl загружается в контроллер из среды разработки CoDeSys командой **Online–Write file to PLC**. По завершении загрузки файла произойдет автоматический перезапуск системы исполнения контроллера и распаковка загруженного архива, после чего система исполнения попытается загрузить DLL и выполнить связывание.
2. Если файл FwCdsUsr.dll отсутствует в корневом каталоге NAND Flash-диска, его можно скопировать с инструментального ПК по протоколу ftp в соответствии с указаниями п. 8.2 настоящего руководства.

FwCdsUsr.dll, помимо точки входа (**DllMain**), должна содержать единственную экспортируемую функцию **F_CdsUsr_link**, описание которой приведено в п. 10.3.2 настоящего руководства.

10.3.2. Функция связывания системы исполнения с DLL пользователя F_CdsUsr_Link

DLL в обязательном порядке должна содержить экспортируемую функцию:

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

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

После успешной загрузки DLL система исполнения выполняет поиск данной функции в загруженной DLL. Если функция не найдена, DLL выгружается, а при успешном обнаружении – производится вызов данной функции. В качестве параметра функции система исполнения передает указатель на интерфейс доступа к сегментам данных приложения CoDeSys (см. п. 10.3.3.1). В качестве результата функция должна вернуть системе исполнения указатель на интерфейс, предоставляемый DLL для координации совместной работы с системой исполнения (см. п. 10.3.4.1).

10.3.3. Интерфейс доступа к сегментам данных приложения CoDeSys

10.3.3.1. F_USR_CDS_IFACE

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

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

```
struct F_USR_CDS_IFACE
{
```

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

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

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

  // Функция получения текущего статуса системы исполнения
  PF_USR_CDS_GET_STATUS     pfGetStatus;

  // Функция получения текущего режима системы исполнения
  PF_USR_CDS_GET_MODE       pfGetMode;

};
```

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

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

10.3.3.2. pfReadVariable/pfWriteVariable

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

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

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

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

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

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

};
```

Идентификатор сегмента представлен типом **F_CDS_RT_SEGMENT**:

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

и может принимать следующие значения:

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

// Сегмент выходных данных
#define CDS_SEG_DATAID_OUTPUT     2

// Сегмент энергонезависимых переменных
#define CDS_SEG_DATAID_RETAIN     3

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

Обе функции в качестве результата возвращают статус операции:

```
// Результат обращения DLL к системе исполнения
```

```
typedef enum F_USR_RESULT
{
  // Успех
  F_UC_OK,

  // Один или оба параметра равны NULL.
  F_UC_BAD_PARAM,

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

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

}F_USR_RESULT;
```

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

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

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

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

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

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

Более подробное описание работы функций чтения и записи сегментов данных приложения приведено в п. 10.3.6 настоящего руководства.

10.3.3.3. Функция чтения статуса приложения

Для получения текущего статуса системы исполнения может быть использовано макроопределение:

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

которое возвращает одно из следущих значений:

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

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

// Приложение CoDeSys функционирует нормально
#define CDSS_RUN        1

// Приложение CoDeSys остановлено
#define CDSS_STOP       2

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

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

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

10.3.4. Интерфейс координации совместной работы с DLL

10.3.4.1. F_CDS_USR_IFACE

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

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

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

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

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

};
```

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

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

10.3.4.2. pfAccept

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

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

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

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

```
typedef size_t F_USR_CODE_RESULT;
```

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

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

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

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

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

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

  // Размер сегмента входных данных
  size_t  inputs_size;

  // Размер сегмента выходных данных
  size_t  outputs_size;

  // Размер сегмента энергонезависимых данных
  size_t  retains_size;

  // Размер сегмента глобальных данных
  size_t  global_size;

};

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

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

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

  // Заголовок проекта
  char             title[CDS_TEXT_SIZE];

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

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

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

  // Информация о размерах сегментов данных приложения
  F_CDS_SEGS_INFO  segments_info;

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

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

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

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

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

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

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

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

10.3.4.3. pfEvent

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

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

Параметр типа **F_CDS_EVENT_TYPE** содержит код события:

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

// При выполнении Online-Stop
#define CDS_EVENT_STOP            2

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

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

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

// При выполнении Online-Login
#define CDS_EVENT_LOGIN           501

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

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

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

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

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

10.3.4.4. pfDoCycle

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

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

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

Первый параметр является указателем на структуру:

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

  // Размер сегмента
  size_t size;

};
```

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

Второй и третий параметры являются кодами режима и статуса системы исполнения перед вызовом **pfDoCycle**. Код режима в текущей версии системы исполнения всегда равен 1, что соответствует нормальному режиму. Описание значений кода статуса приведено в п. 10.3.3.3 настоящего руководства.

Описание типа возвращаемого значения приведено в п. 10.3.4.2. Таким образом, пользовательская DLL может в любой момент перевести контроллер в безопасный режим либо прекратить дальнейшие вызовы своего кода, реализуемого функциями **pfEvent** и **pfDoCycle.**

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

10.3.5. Вызов функций интерфейса координации совместной работы с DLL пользователя

10.3.5.1. Последовательность обращения к функциям интерфейса DLL

Взаимодействие между системой исполнения и пользовательской DLL осуществляется следующим образом:

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

   Если загрузка DLL не удалась, система исполнения продолжает работу в нормальном режиме.
2. Если загрузка DLL произведена успешно, система исполнения выполняет поиск в DLL функции **F_CdsUsr_link.** Если функция не найдена, DLL выгружается, и система исполнения продолжает работу в нормальном режиме.
3. Если функция **F_CdsUsr_link** найдена, система исполнения вызывает ее, передав в качестве параметра указатель на интерфейс доступа к сегментам данных приложения (см. п. 10.3.3).

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

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

После успешного вызова функции **F_CdsUsr_link** проверяется, вернула ли она указатель на интерфейс координации совместной работы с DLL (см. п. 10.3.4).

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

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

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

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

   Если функция **pfAccept** реализована в интерфейсе DLL, система исполнения вызывает ее, передав в качестве параметра указатель на структуру **F_CDS_PROJECT_INFO**, содержащую информацию о текущем приложении, загруженном в контроллер (см. п. 10.3.4.2).

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

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

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

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

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

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

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

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

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

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

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

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

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

10.3.5.2. Обработка исключений в коде функций интерфейса DLL

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

10.3.6. Вызов функций интерфейса системы исполнения из DLL пользователя

10.3.6.1. Общие сведения

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

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

10.3.6.2. Вызов функции чтения сегментов приложения

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

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

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

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

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

10.3.6.3. Вызов функции записи в сегменты приложения

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

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

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

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

10.3.7. Получение информации о размещении переменных приложения CoDeSys

10.3.7.1. Формирование файла символической информации о проекте CoDeSys

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

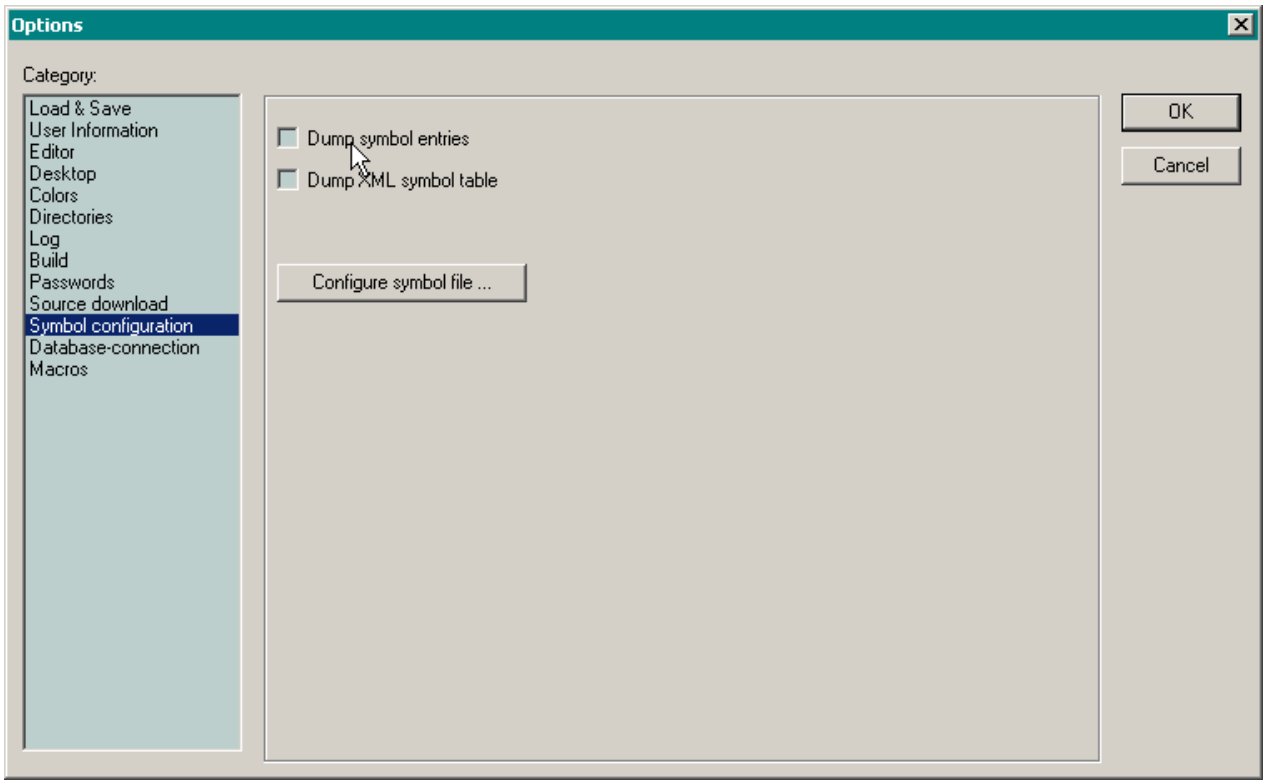


Рис. 98. Конфигурация символической информации в диалоговой панели Options

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

1. Выполните команду **Project–Options** в главном меню CoDeSys и в появившейся диалоговой панели **Options** выберите категорию опций **Symbol Configuration**, как показано на рис. 98.
2. Отметьте флажок **Dump symbol entries** и нажмите кнопку **Configure symbol file...**. На экран монитора будет выведена диалоговая панель **Set object attributes**, показанная на рис. 99.

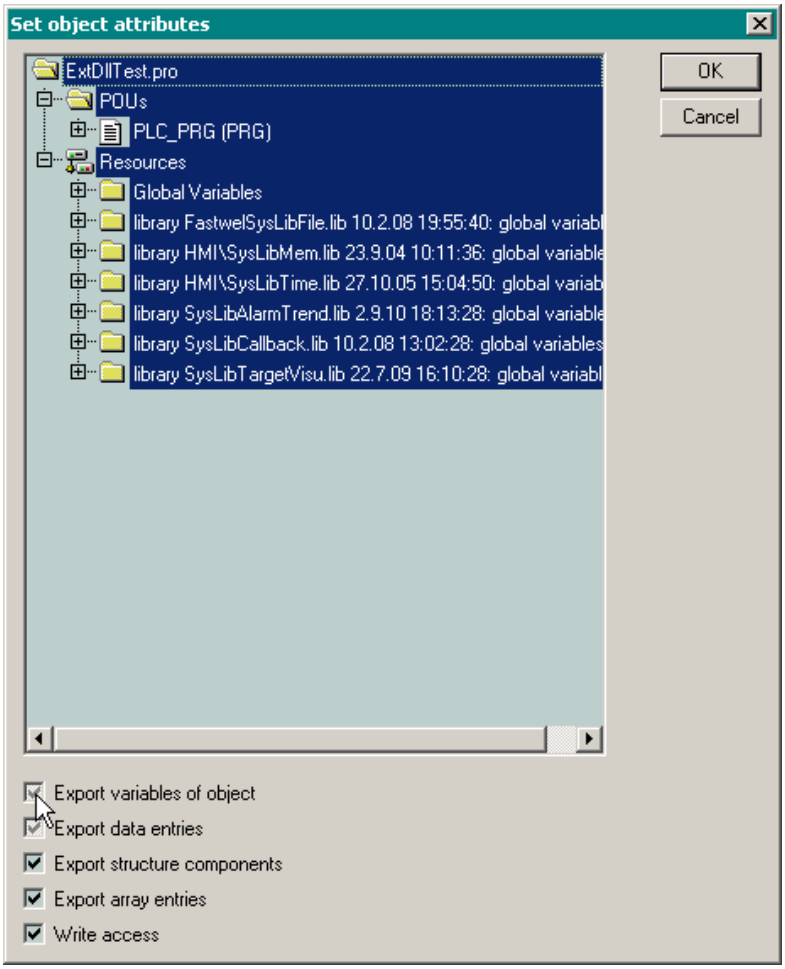


Рис. 99. Внешний вид диалоговой панели Set object attributes

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

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

10.3.7.2. Формат символической информации

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

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

где:

POU – имя программной единицы, содержащей переменную;

var_name – имя переменной или элемента массива;

var_subname – имя поля структурной переменной;

iec_type – имя типа переменной IEC 61131-3;

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

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

seg_off – байтовое смещение переменной в сегменте;

len – длина переменной в байтах.

Например:

Массив

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

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

Элемент массива

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

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

Структура

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

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

Поле структуры

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

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

Переменная примитивного типа

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

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

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

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

где:

POU – имя программной единицы, содержащей переменную;

var_name – имя переменной;

BOOL – обозначение булева типа;

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

seg_bit_off – битовое смещение переменной в сегменте;

0 – длина переменной в байтах.

Например:

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

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

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

10.4. Пример реализации DLL пользователя

10.4.1. Общие сведения

Настоящий подраздел содержит информацию о небольшом проекте, поставляемом в составе пакета адаптации CoDeSys для контроллера MK905, который не содержит ни одной строчки исполняемого кода на языках IEC 61131-3 и взаимодействует с DLL, формирующей данные для формы визуализации, входящей в проект.

10.4.2. Проект CoDeSys

Проект CoDeSys с именем ExtDllTest.pro (по умолчанию расположен в каталоге C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Examples\CdsExtDllExample\CoDeSys Project\MK905) содержит декларации трех глобальных переменных, значения которых формируются в DLL и отображаются в окне формы визуализации DiagForm.

Переменная Runtime_Information типа F_APP_INFO предназначена хранения общего количества циклов и запаздываний системы исполнения, которые DLL записывает в данную переменную, предварительно считав их по адресам %ID0 и %ID1 области входных данных приложения.

Переменная IO_Information типа FBUS_INFO содержит диагностическую информацию сервиса ввода-вывода, которую DLL записывает в данную переменную, предварительно считав по адресам участка MK905 Programmable Automation Controller –I/O Modules–FBUS Diagnostics области входных данных приложения.

Значение переменной ExternalRamp типа LREAL формируется DLL и отображается на графике формы визуализации DiagForm.

Программа PLC_PRG ассоциирована с ациклической задачей PLC_PRG_TASK и не вызывается ни разу, поскольку переменная чувствительности задачи PLC_PRG_TASK всегда остается неизменной и равной FALSE.

После загрузки DLL и приложения CoDeSys в контроллер на экране монитора, подключенного к MK905, будут отображаться значения переменных, передаваемые из DLL.

10.4.3. Проект DLL

10.4.3.1. Общие сведения

Исходный текст единственного модуля MK905.cpp DLL по умолчанию расположен в каталоге: C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Examples\CdsExtDllExample\src\MK905.

Построение DLL может быть выполнено в среде разработки MS Visual Studio 2008 либо MS eMbedded Visual C++ 4.0 SP4. Соответствующие проекты расположены в каталогах:

C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Examples\CdsExtDllExample\projects\MSVS2008

C:\Program Files\Fastwel\Fastwel CoDeSys Adaptation\Examples\CdsExtDllExample\projects\eVC4

Перед построением на инструментальный ПК требуется установить SDK для образа операционной системы Windows CE 5.0 MK905, установочный комплект которого можно загрузить по адресу:

ftp://ftp.prosoft.ru/pub/Hardware/Fastwel/Fastwel_IO/Version2/Setup/SDK/MK905 .

Перед построением проекта в MS Visual Studio 2008 в операционной системе Windows должна быть создана переменная окружения CESDK, имеющая значение, совпадающее с каталогом установки SDK. По умолчанию таковым является:

"C:\Program Files\Windows CE Tools\wce500\Fastwel MK905 Programmable Automation Controller"

10.4.3.2. Описание исходного текста DLL

Реализация DLL находится в модуле MK905.cpp. Ниже приведен листинг модуля с комментариями на русском языке.

```
#include <windows.h>
#include <cdsdll.h>

// Здесь будет храниться указатель на интерфейс, переданный системой исполнения.
static PF_USR_CDS_IFACE spRuntime = NULL;

// Допустимое имя проекта CoDeSys, с которым будет функционировать DLL
static const char* sAllowedProjectName = "ExtDllTest.pro";

// Минимально допустимый размер сегмента глобальных данных приложения CoDeSys
#define CDS_GLOBAL_VARS_MIN_SIZE 289

// Минимально допустимый размер сегмента входных данных
#define CDS_INPUT_MIN_SIZE 32

// Структура представляет первые 32 байта области входных данных МК905
struct CdsProjectInputs
{

  // Общее количество циклов системы исполнения
  unsigned long cycles_count;

  // Общее количество запаздываний задач
  unsigned long overruns_count;

  // Статус задач (не используется)
  unsigned long tasks_status[2];

  // Статус наличия связи с первыми 32-мя модулями ввода-вывода
  unsigned long io_status_0;

  // Статус наличия связи со следующими 32-мя модулями ввода-вывода
  unsigned long io_status_1;

  // Количество транзакций с модулями ввода-вывода
  unsigned long io_transactions_count;

  // Количество неудачных транзакций с модулями ввода-вывода
  unsigned long io_errors_count;

};

/**
 *  Реализация pfAccept.
 *
 *  Проверяет правильность размера pProjectInfo, имя проекта
 *  и размеры входного и глобального сегмента
 *
 **/
static F_USR_CODE_RESULT cds_test_dll_accept(const PF_CDS_PROJECT_INFO pProjectInfo)
{
  // Если загружен неправильный проект, просим перейти
  // в безопасный режим.
  F_USR_CODE_RESULT res = F_UCR_ABORT;

  if(pProjectInfo->this_size == sizeof(F_CDS_PROJECT_INFO) &&
     0 == strcmp(pProjectInfo->project, sAllowedProjectName) &&
     pProjectInfo->segments_info.global_size >= CDS_GLOBAL_VARS_MIN_SIZE &&
     pProjectInfo->segments_info.inputs_size >= CDS_INPUT_MIN_SIZE)
  {
    // Все в порядке, можно работать
    res = F_UCR_OK;
  }

  return res;
}

/**
 *  Реализация pfEvent.
 **/
static void cds_test_dll_event(F_CDS_EVENT_TYPE type)
{
  // Ничего не делаем
}
```

```
// Смещение и длина переменной Runtime_Information
#define CDS_RT_INFO_OFFSET 331
#define CDS_RT_INFO_LENGTH 8

// Смещение и длина переменной IO_Information
#define CDS_IO_INFO_OFFSET 339
#define CDS_IO_INFO_LENGTH 16

// Смещение и длина переменной ExternalRamp
#define CDS_RAMP_OFFSET 355
#define CDS_RAMP_LENGTH sizeof(double)

// Для наивной реализации генератора треугольных импульсов
struct RampGen
{
  double  value;
  int     slope;
};

static RampGen ramp_gen = { 0.0, 1 };

/**
 *  Реализация pfDoCycle.
 **/
static F_USR_CODE_RESULT cds_test_dll_cycle(const PF_CDS_DATA_SEGMENT pInputSegment,
                                            F_CDS_RT_MODE mode,
                                            F_CDS_RT_STATUS status)
{
  F_USR_CODE_RESULT res = F_UCR_ABORT;

  // Еще раз проверяем размер области входных данных
  if(pInputSegment->size >= CDS_INPUT_MIN_SIZE)
  {
    // Локальные первые 32 байта области входных данных.
    // На самом деле, можно сделать указатель и привести к нему pInputSegment->pData.
    CdsProjectInputs status_inputs;

    // Дескриптор переменной CoDeSys, будет использоваться для записи
    F_CDS_USR_VAR cds_var_descriptor;

    // Вернем Ok системе исполнения
    res = F_UCR_OK;

    // Сбрасываем в 0
    memset(&status_inputs, 0, sizeof(status_inputs));

    // Копируем к себе 32 байта из pInputSegment->pData
    memcpy(&status_inputs, pInputSegment->pData, sizeof(status_inputs));

    // Будем писать 8 байт по смещению CDS_RT_INFO_OFFSET
    // в сегмент глобальных данных (в переменную Runtime_Information).
    cds_var_descriptor.segment_id = CDS_SEG_DATAID_GLOBVARS;
    cds_var_descriptor.offset = CDS_RT_INFO_OFFSET;
    cds_var_descriptor.len = CDS_RT_INFO_LENGTH;

    // Записываем в переменную Runtime_Information
    F_CdsUsr_writeVariable(spRuntime, &cds_var_descriptor, &status_inputs);

    // Далее будем писать 16 байт по смещению CDS_IO_INFO_OFFSET
    // в сегмент глобальных данных (в переменную IO_Information).
    cds_var_descriptor.offset = CDS_IO_INFO_OFFSET;
    cds_var_descriptor.len = CDS_IO_INFO_LENGTH;

    // Пишем задние 16 bytes status_inputs в переменную IO_Information
    F_CdsUsr_writeVariable(spRuntime, &cds_var_descriptor,
                           &status_inputs.io_status_0);

    // Наивная реализация генератора треугольных импульсов
    ramp_gen.value +=  ramp_gen.slope * 0.25;

    if(ramp_gen.slope > 0 &&  ramp_gen.value > 100.0)
    {
      ramp_gen.value = 100.0;
      ramp_gen.slope = -1;
    }
```

```
      else if(ramp_gen.slope < 0 &&  ramp_gen.value < 0.0)
      {
        ramp_gen.value = 0.0;
        ramp_gen.slope = 1;
      }

      // Пишем текущее значение генератора в ExternalRamp
      cds_var_descriptor.offset = CDS_RAMP_OFFSET;
      cds_var_descriptor.len = CDS_RAMP_LENGTH;

      F_CdsUsr_writeVariable(spRuntime, &cds_var_descriptor, &ramp_gen.value);

    }

    return res;
}

/**
 *  Интерфейс DLL-to-Runtime
 **/
static F_CDS_USR_IFACE to_cds_iface =
{
  sizeof(F_CDS_USR_IFACE),
  cds_test_dll_accept,
  cds_test_dll_event,
  cds_test_dll_cycle
};


/**
 *  Функция связывания DLL с системой исполнения
 **/
PF_CDS_USR_IFACE F_CdsUsr_link(PF_USR_CDS_IFACE pRuntimeIface)
{
  spRuntime = pRuntimeIface;
  return &to_cds_iface;
}


/**
 *  Точка входа в DLL
 **/
BOOL WINAPI DllMain( HANDLE hModule, DWORD  ul_reason_for_call, LPVOID lpReserved )
{
  if (ul_reason_for_call == DLL_PROCESS_ATTACH)
  {
  }

  if (ul_reason_for_call == DLL_PROCESS_DETACH)
  {
  }

  return TRUE;
}
```

11. Сообщения об ошибках

11.1. Общие сведения

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

Существуют следующие классы причин перехода системы исполнения в безопасный режим:

4. Исключение – причины данного класса связаны с проявлением дефектов в пользовательском коде приложения или в системном программном обеспечении, при наличии которых система исполнения не может обеспечить предсказуемое выполнение пользовательских алгоритмов и функционирование периферийных устройств, подключенных к контроллеру.
5. Дефицит системных ресурсов – причины данного класса связаны с нехваткой памяти или других ресурсов, возникающей после загрузки приложения или в процессе его работы.
6. Зацикливание в пользовательском коде – причины данного класса связаны с наличием бесконечных (или почти бесконечных) циклов в пользовательском коде либо при остановке пользовательского кода в функциях интерфейса, возвращаемого системе исполнения из DLL расширения. Контроль дефектов данного класса выполняется системой исполнения при помощи программного сторожевого таймера с интервалом от 20 до 30 с.
7. Ошибка конфигурации приложения – возникают при запуске приложения и могут быть связаны с неправильными значениями параметров **PLC Configuration**, превышением допустимых значений параметров системы исполнения и другими причинами, делающими невозможным дальнейшее функционирования системы исполнения в нормальном режиме.

11.2. Исключения

11.2.1. Типы исключений

Коды типов исключений приведены в табл. 26.

Таблица 26

Обозначение	Причина
EXCEPTION_ACCESS_VIOLATION	Ошибка доступа к памяти по чтению или записи
EXCEPTION_DATATYPE_MISALIGNMENT	Попытка доступа к памяти с нарушением выравнивания
EXCEPTION_ARRAY_BOUNDS_EXCEEDED	Попытка доступа за пределы массива при наличии аппаратного контроля
EXCEPTION_FLT_DENORMAL_OPERAND	Попытка выполнения операции с плавающей точкой над ненормализованным операндом
EXCEPTION_FLT_DIVIDE_BY_ZERO	Попытка деления на 0 при выполнении операции с плавающей точкой
EXCEPTION_FLT_INEXACT_RESULT	Дробная часть результата выполнения операции с плавающей точкой не может быть представлена целым числом
EXCEPTION_FLT_INVALID_OPERATION	Некорректная операция с плавающей точкой
EXCEPTION_FLT_OVERFLOW	Экспонента результата выполнения операции с плавающей точкой превышает порядок, допустимый для типа операндов
EXCEPTION_FLT_STACK_CHECK	Переполнение стека при выполнении операции с плавающей точкой
EXCEPTION_FLT_UNDERFLOW	Экспонента результата выполнения операции с плавающей точкой менее порядка, допустимого для типа операндов
EXCEPTION_INT_DIVIDE_BY_ZERO	Попытка целочисленного деления на 0
EXCEPTION_INT_OVERFLOW	Переполнение при выполнении операции с целочисленными операндами
EXCEPTION_PRIV_INSTRUCTION	Попытка исполнения привилегированной инструкции
EXCEPTION_NONCONTINUABLE_EXCEPTION	Попытка продолжения исполнения после исключения, не предусматривающего возможность продолжения исполнения
EXCEPTION_ILLEGAL_INSTRUCTION	Попытка исполнения неизвестной инструкции
EXCEPTION_STACK_OVERFLOW	Переполнение стека

11.2.2. Исключения при исполнении кода приложения CoDeSys

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

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

где:

EXCPT_TYPE – тип исключения в соответствии с табл. 26;

addr – абсолютный адрес в сегменте кода;

task_name – имя задачи приложения, в которой произошло исключение;

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

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

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

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

11.2.3. Исключение вне кода приложения CoDeSys

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

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

где:

EXCPT_TYPE – тип исключения в соответствии с табл. 26;

addr – абсолютный адрес в сегменте кода;

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

При возникновении такого исключения следует переслать информацию о данном инциденте по адресу службы технической поддержки Fastwel (см. п. **Ошибка! Источник ссылки не найден.)**.

11.2.4. Исключения в коде пользовательской DLL расширения системы исполнения

Информация об исключениях в коде DLL расширения системы исполнения приведена в п. **Ошибка! Источник ссылки не найден.** настоящего руководства.

11.3. Сообщения о дефиците системных ресурсов

Перечень сообщений о дефиците системных ресурсов приведен в табл. 27.

Таблица 27

Обозначение	Причина
Too many target visu function calls	Приложение содержит форму визуализации, которая не может быть отображена системой исполнения.
Too many update regions	
Failed to create ShMemory lock	Недостаточное количество ресурсов операционной системы при первоначальном запуске системы исполнения. Вероятно, наряду с системой исполнения на данном вычислительном устройстве запущено некоторое приложение, потребляющее чрезмерное количество системных ресурсов. Рассмотрите возможность функционирования системы без дополнительных приложений либо устраните в нем утечку памяти или других ресурсов.
Failed to allocate RPC input buffer	
Failed to allocate RPC output buffer	
Failed to init temporary buffer	
Failed to init PLC:	
Failed to init DXS2	
Failed to init Tasks Manager	
Failed to init 3S HMI files manager	
Failed to init User DLL Manager	
Commons initialization failure for modbus masters	
Modbus Servers Commons initialization failure	
Failed to initialize ModbusServer	
Failed to init 3S HMI	Недостаточное количество ресурсов операционной системы или отсутствует/поврежден файл ресурсов системы исполнения *cdshmirc_res.dll*. Система исполнения не сможет функционировать даже в безопасном режиме. Если поврежден или отсутствует *cdshmirc_res.dll*, восстановите его, скопировав на диск контроллера файл обновления системного ПО *norm.dnl* и повторно запустив контроллер.
Code initialization failed:	Возникла ошибка при инициализации сегмента кода приложения CoDeSys. Справа от двоеточия отображается символьный код уточняющей информации. Перечень символьных кодов приведен в табл. 28.
Tasks configuring failed:	Возникла ошибка при конфигурировании задач приложения CoDeSys. Справа от двоеточия отображается символьный код уточняющей информации. Перечень символьных кодов приведен в табл. 28.
Tasks linking failed	Ошибка связывания задач приложения CoDeSys с образом процесса по одной из следующих причин: 1. Среда разработки CoDeSys сформировала для некоторой задачи ссылку на несуществующий участок образа процесса. 2. Для ациклической задачи задана несуществующая переменная чувствительности. В среде разработки CoDeSys откройте окно ресурса **PLC Configuration** и выполните команду меню **Extras–Calculate Addresses**, затем **Project–Clean all** и **Project Rebuild all**, после чего вновь загрузите приложение в контроллер.
Failed to update the call set for task <task_num>	Пользовательская задача с номером *task_num* (порядковый номер в списке задач в окне ресурса **Task Configuration**) содержит POU, в котором имеется слишком много вложенных вызовов других POU. Попробуйте упростить структуру вызовов либо обратитесь в службу поддержки Fastwel.
Failed to exchange storages	Не удалось выполнить перестановку вторичного хранилища приложения с первичным. Вероятно, повреждена файловая система основного дискового накопителя контроллера. Обратитесь в службу поддержки Fastwel.
Failed to initialize FBUS	Не удалось выполнить инициализацию сервиса ввода-вывода. По всей видимости, поврежден или отсутствует файл драйвера шины FBUS *ce_fbus.dll*. Восстановите его, скопировав на диск контроллера файл обновления системного ПО *norm.dnl* и повторно запустив контроллер.
Failed to link groups	Недостаточное количество системных ресурсов для связывания коммуникационных объектов сервиса FBUS с образом процесса. Рассмотрите возможность функционирования системы без дополнительных приложений либо устраните в нем утечку памяти или других ресурсов.

11.4. Зацикливания в пользовательском коде

11.4.1. Зацикливание в циклической задаче

Информация о зацикливании в циклической задаче представляется сообщением:

`System overrun: <task_name>`

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

11.4.2. Зацикливание в сервисной задаче

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

```
CoDeSys2.SERVICE_TASK_STALLED
```

11.5. Ошибки конфигурации приложения

Символьные коды ошибок подсистемы исполнения кода CoDeSys перечислены в табл. 28. Каждый символьный код ошибки подсистемы исполнения кода CoDeSys предваряется префиксом **CoDeSys2.**. Некоторые из перечисленных кодов являются уточняющими для сообщения об ошибке дефицита системных ресурсов.

Таблица 28

Обозначение	Причина
Ошибки конфигурирования задач	
FAILED_TO_CREATE_TASK	Не удалось создать поток операционной системы. См. сообщение Failed to init PLC в табл. 27.
FAILED_TO_CONFIGURE_TASK	Не удалось сконфигурировать задачу приложения. Среда разработки ассоциировала с задачей несуществующий корневой программной единицей либо причина аналогична описанной для сообщения Failed to update the call set for task <task_num> в табл. 27. В первом случае обратитесь в службу поддержки Fastwel.
FAILED_TO_LINK_TASK	См. сообщение Tasks linking failed в табл. 27.
REINIT_TASKS_FAILURE	Одна из задач содержит слишком много ссылок на образ процесса.
Ошибки управления хранилищем проектной информации приложения	
CODE_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию кода. Попытайтесь повторно загрузить приложение.
CONFIG_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию конфигурации. Попытайтесь повторно загрузить приложение.
TASKS_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию конфигурации задач. Попытайтесь повторно загрузить приложение.
PRJINFO_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию информации о проекте. Попытайтесь повторно загрузить приложение.
IODESC_SECTION_WRITE_FAILURE	При загрузке нового приложения не удалось выполнить запись в секцию информации о ссылках задач на образ процесса. Попытайтесь повторно загрузить приложение.
STORAGE_FAILURE	После загрузки нового приложения в одной из секций обнаружена ошибка контрольной суммы. Попытайтесь повторно загрузить приложение.
REPLACE_STORAGES_FAILURE	Не удалось выполнить перестановку вторичного хранилища приложения с первичным. Попытайтесь повторно загрузить приложение. В случае повторной ошибки обратитесь в службу поддержки Fastwel.
LOAD_PRJINFO_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции информации о проекте. Попытайтесь повторно загрузить приложение.
LOAD_PRJINFO_FAILURE	Ошибка контрольной суммы при чтении секции информации о проекте. Попытайтесь повторно загрузить приложение.
LOAD_CODE_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции кода. Попытайтесь повторно загрузить приложение.
LOAD_CODE_FAILURE	Ошибка контрольной суммы при чтении секции кода. Попытайтесь повторно загрузить приложение.
NOT_ENOUGH_MEMORY_FOR_CODE	Размер секции кода превышает предельно допустимое значение для данной системы исполнения. Обратитесь в службу поддержки Fastwel.
CODE_REINIT_FAILURE	Ошибка инициализации сегмента кода. Обратитесь в службу поддержки Fastwel.
LOAD_CONFIG_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции конфигурации приложения. Попытайтесь повторно загрузить приложение.
LOAD_CONFIG_FAILURE	Ошибка контрольной суммы при чтении секции конфигурации приложения. Попытайтесь повторно загрузить приложение.
LOAD_TASKS_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции конфигурации задач. Попытайтесь повторно загрузить приложение.
LOAD_TASKS_FAILURE	Ошибка контрольной суммы при чтении секции конфигурации задач. Попытайтесь повторно загрузить приложение.

Обозначение	Причина
LOAD_IODESC_LEN_FAILURE	Ошибка контрольной суммы при чтении длины секции информации о ссылках задач на образ процесса. Попытайтесь повторно загрузить приложение.
LOAD_IODESC_FAILURE	Ошибка контрольной суммы при чтении секции информации о ссылках задач на образ процесса. Попытайтесь повторно загрузить приложение.
REINIT_IODESC_FAILURE	Одна из задач содержит слишком много ссылок на образ процесса.
PARSE_CONFIG_FAILURE	Не хватило памяти для загрузки дерева конфигурации приложения, либо структура дерева конфигурации не соответствует ожидаемой.
INVALID_CONFIG_FORMAT	В конфигурации приложения не найден требуемый элемент либо обнаружен элемент, не поддерживаемый данной системой исполнения.
INVALID_CONTROLLER_ID	В контроллер загружена конфигурация приложения, не соответствующая данной системе исполнения (типу контроллера).
Ошибки управления образом процесса	
PROCESS_IMAGE_REINIT_FAILURE	При загрузке нового приложения не удалось увеличить размер образа процесса. Попытайтесь повторно загрузить приложение.
FAILED_TO_INIT_DATA_MANAGER	См. сообщение Failed to init PLC: в табл. 27.
Ошибки инициализации буферов, сегментов кода и данных приложения	
BUFFERS_INITIALIZATION_FAILURE	Не хватило памяти для служебных буферов Online-сервисов взаимодействия со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
INVALID_RETAIN_DATA_SIZE	Не хватило системных ресурсов для управления энергонезависимыми переменными. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_RECEIVE_BUFFER	Не хватило памяти для буфера приема, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_RETAIN_SEGMENT	Не хватило памяти для создания сегмента энергонезависимых переменных. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_SEND_BUFFER	Не хватило памяти для буфера передачи, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_VARLIST_BUFFER	Не хватило памяти для буфера чтения/записи списков переменных, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_FORCELIST_BUFFER	Не хватило памяти для буфера форсирования значений переменных, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_TEMP_BUFFER	Не хватило памяти для вспомогательного буфера, используемого при взаимодействии со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_RPC_BUFFER	Не хватило памяти для буфера сервиса вызова удаленных процедур, реализующего взаимодействие со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_RESOURCES_FOR_RPC	Не хватило системных ресурсов для сервиса вызова удаленных процедур, реализующего взаимодействие со средой разработки CoDeSys. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_PRIMARY_CODESEG	Не хватило памяти для сегмента кода. См. сообщение Failed to init PLC: в табл. 27.
NOT_ENOUGH_MEMORY_DATASEG	Не хватило памяти для глобального сегмента данных. См. сообщение Failed to init PLC: в табл. 27.
Ошибки загрузки и связывания с библиотечными функциями	
CODE_INIT_FAILURE	Не хватило системных ресурсов для управления библиотекой Standard.lib.
GLOBAL_INIT_RESOLUTION_FAILURE	Не удалось найти функцию инициализации глобальных данных в сегменте кода. Обратитесь в службу поддержки Fastwel.
GLOBAL_INIT_FAILED	После вызова функции инициализации глобальных, обнаруженной в сегменте кода, обнаружено повреждение памяти. Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
INVALID_DATA_SIZE	Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
INVALID_CODE_SIZE	Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
RETAIN_SIZE_EXCEEDED	Загружено приложение, у которого превышен допустимый размер сегмента энергонезависимых переменных. Появление данного кода маловероятно, однако если он появился, в среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
INVALID_PROGRAM_CHECKSUM	Неправильная контрольная сумма загруженного приложения. В среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.

Обозначение	Причина
DATA_RELOCATION_FAILURE	Не удалось выполнить релокацию ссылок на данные в коде приложения. В среде CoDeSys выполните последовательность **Project–Clean all**, **Project Rebuild all**, после чего повторно загрузите приложение.
DYNAMIC_LINKAGE_FAILURE	Не удалось найти функцию внешней библиотеки. В коде приложения имеется вызов библиотечной функции, которая не поддерживается системой исполнения. Информация о поддерживаемых внешних библиотеках приведена в разделе **Ошибка! Источник ссылки не найден.** настоящего руководства.

ПРИЛОЖЕНИЕ 1. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Версия	**Дата**	**Ссылка**	**Статус**	**Примечания**
2.59.23937	05.12.2012	Документ	создан	Первый выпуск