А.В. Графкин

ПРИНЦИПЫ ПРОГРАММНОГО УПРАВЛЕНИЯ МОДУЛЯМИ ICP DAS СЕРИИ I-7000 В ЗАДАЧАХ ПРОМЫШЛЕННОЙ АВТОМАТИЗАЦИИ

САМАРА 2010

УДК 004.9 (075)

Рецензенты:
Заслуженный работник высшей школы РФ, д.т.н.,
профессор Прохоров С.А.;
д.т.н., профессор Кузнецов П.К.

А.В. Графкин
Принципы программного управления модулями ICP DAS СЕРИИ I-7000 в задачах промышленной автоматизации / СНЦ РАН, 2010. – 133 с.: ил.

ISBN 978-5-93424-475-1

Монография содержит описание особенностей, которые необходимо учитывать при работе с модулями ICP DAS серии I-7000. Даны рекомендации по использованию модулей, а также приведены примеры программ в среде Windows.

Предназначена для преподавателей, научных сотрудников, инженеров, аспирантов и студентов как руководство по изучению основ проектирования автоматизированных систем обработки информации и управления.

Печатается по решению издательского совета Самарского научного центра Российской академии наук.

ISBN 978-5-93424-475-1

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время все большее распространение получают автоматизированные системы на основе последовательных интерфейсов. Наиболее широко они используются в автоматизированных системах управления технологическими процессами (АСУ ТП) для сбора, обработки информации и управления территориально распределенными объектами.

АСУ ТП на базе последовательных интерфейсов по сравнению с традиционными централизованными системами имеют несколько преимуществ:

1. Повышение надежности системы управления. По надежности цифровой метод передачи данных намного превосходит аналоговый. Передача в цифровом виде малочувствительна к помехам и гарантирует доставку информации благодаря специальным механизмам, встроенным в протоколы промышленных сетей (контрольные суммы, повтор передачи искаженных пакетов данных). Повышение надежности функционирования и живучести АСУ ТП на базе промышленных сетей также связано с распределением функций контроля и управления по различным узлам сети. Выход из строя одного узла не влияет либо влияет незначительно на отработку технологических алгоритмов в остальных узлах. Для критически важных технологических участков возможно дублирование линий связи или наличие альтернативных путей передачи информации. Это позволяет сохранить работоспособность системы в случае повреждения кабельной сети.

2. Гибкость и модифицируемость. Добавление или удаление отдельных точек ввода-вывода и даже целых узлов требует минимального количества монтажных работ и может производиться без остановки системы автоматизации. Переконфигурация системы осуществляется на уровне программного обеспечения и также занимает минимальное время.

3. Использование принципов открытых систем, открытых технологий, что позволяет успешно интегрировать в единую систему изделия от различных производителей.

4. Приближение мощности вычислительных средств к объекту управления.

5. Повышение живучести всей системы за счет использования «горячей» замены, принципа автоконфигурации, дублирования критически важных узлов [2, 5, 11].

6. Существенная экономия кабельной продукции. Вместо километров дорогих кабелей, используемых для передачи аналоговой ин-

формации требуется несколько сотен метров дешевой витой пары. Также сокращаются расходы на вспомогательное оборудование (кабельные каналы, клеммы, шкафы).

Снижение затрат при переходе на цифровые сети иллюстрирует рис. В1 [10].

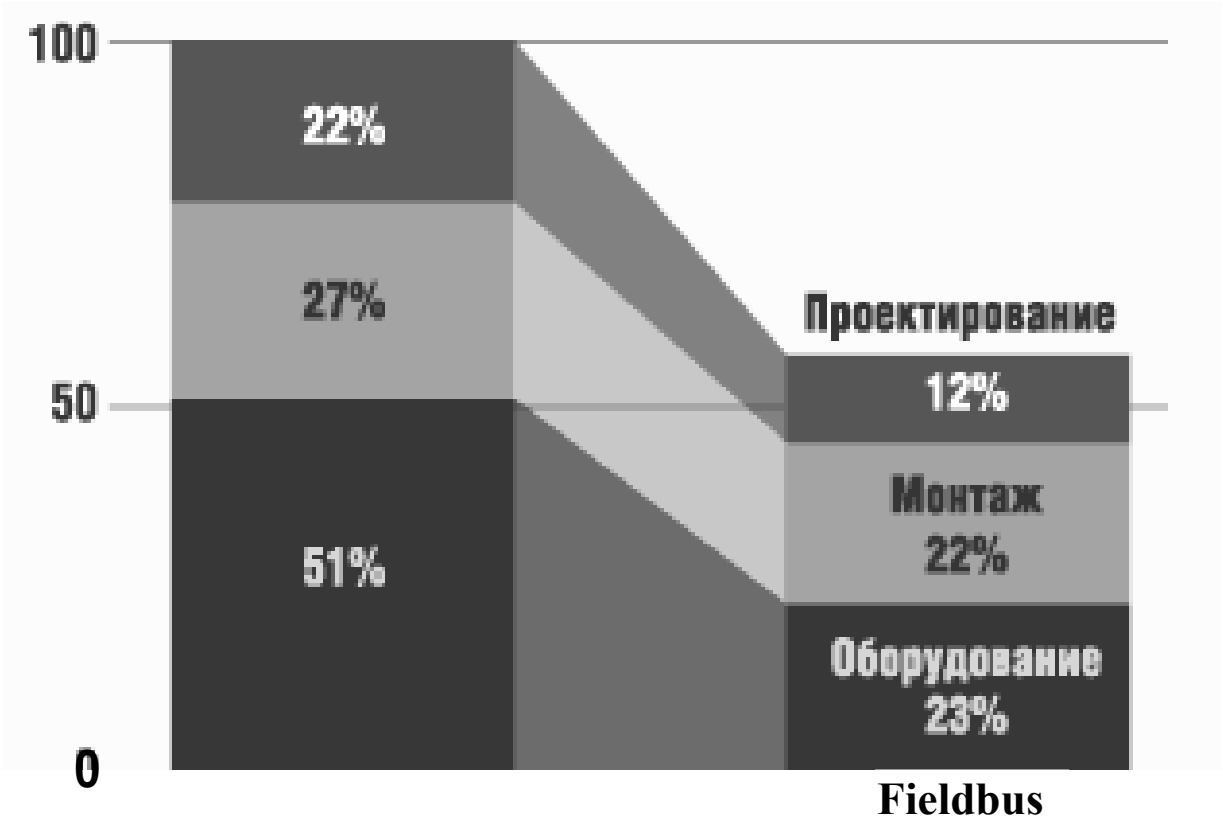


Рис. В1. Снижение затрат при переходе на цифровые сети

На рис. В1 левый столбец соответствует структуре затрат пользователя при использовании аналогового способа подключения устройств, а правый – структуре затрат при переходе на цифровые сети.

Особенностью современного производства является тесное взаимодействие технологических (производственных) и информационных потоков.

В этих условиях существенно возрастает роль данных, собираемых на всех уровнях АСУ ТП. Требования, предъявляемые со стороны потребителей этой информации, все более ужесточаются в части объема, скорости и надежности получения данных, поэтому вопросы обеспечения коммуникаций становятся высокоприоритетными.

На рис. В2 показана обобщенная сетевая структура организации информационного обмена на условном промышленном предприятии [5, 8].

Каждый уровень имеет свои особенности по виду передаваемой информации, среде передачи, ее объему, допустимому времени ожи-

дания и доставки данных, требованиям к защите информации и так далее.

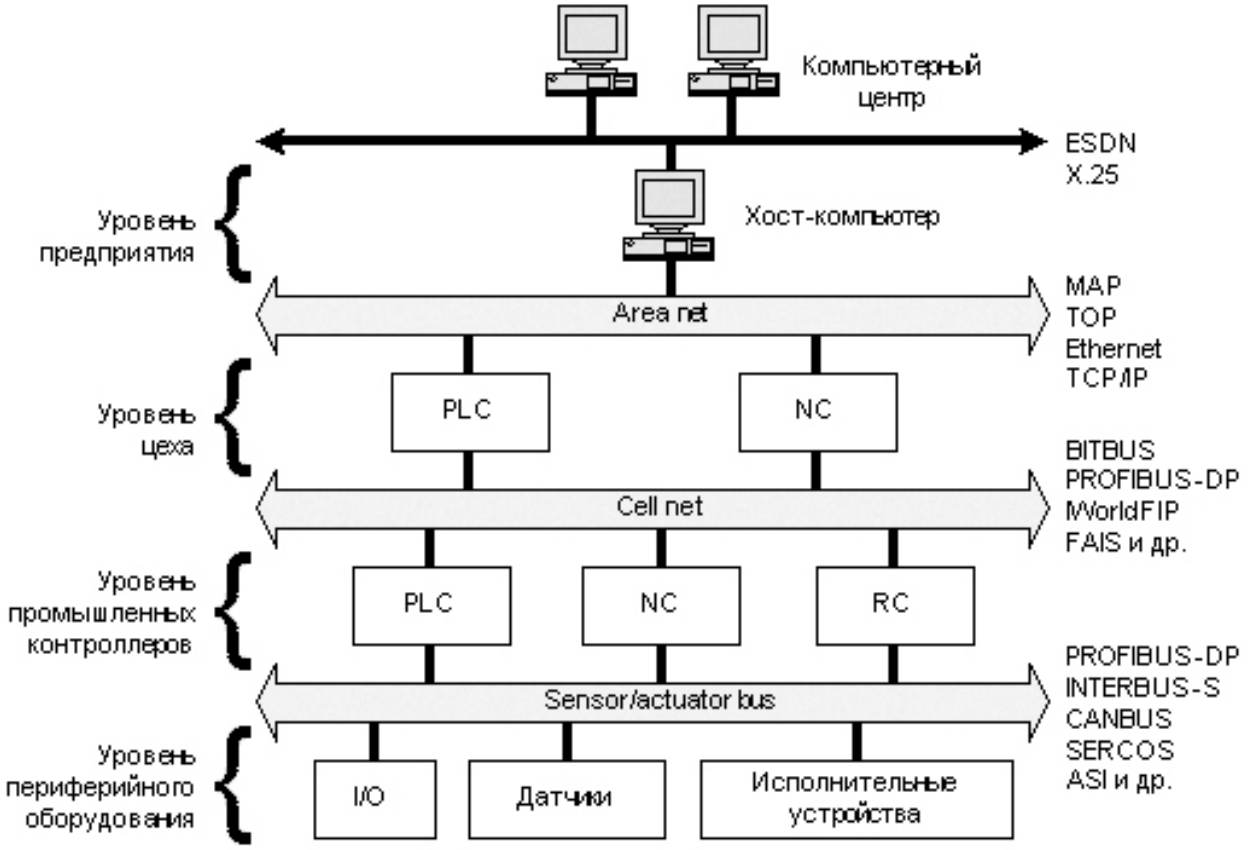


Рис. В2. Обобщенная структура информационных сетей предприятия

На уровне планирования (предприятия) данные, подготовленные при проектировании, пересылаются к рабочим местам. Для этого уровня характерны огромные объемы некритичной ко времени информации, но доступ к информации должен быть авторизован.

Для уровня управления производством характерными являются апериодические сообщения относительно большого объема, адресованные различным подсистемам. Если обмен информацией выполняется в ходе технологического процесса, то могут быть введены ограничения на время обмена.

На цеховом уровне происходит обмен информацией между различными автономными центрами обработки и управления, объединенными общими глобальными задачами (например, обрабатывающие центры цеха). Здесь необходима синхронизация работы в режиме реального времени, что несовместимо с длинными пакетами передаваемых сообщений. Но требования к допустимым задержкам не такие жесткие, как на нижнем уровне.

Уровень периферийного оборудования (полевой) включает устройства, соответствующие данному технологическому процессу (датчики, исполнительные устройства, средства ввода/вывода). Задача этих устройств – формировать и передавать информацию о состоянии производимого продукта и технологического процесса. Измеренные значения являются базисом для вмешательства в технологический процесс или управление.

Для разных технологических процессов виды измерения и управления полевого уровня могут существенно отличаться. Они могут быть цифровыми или аналоговыми, доступ может быть динамическим (короткие промежутки цикла опроса) или статическим с циклическим и произвольным доступом, при этом необходимо учитывать появление критических результатов, которые должны обрабатываться с максимальной скоростью, и так далее. Общим для данного уровня является: работа в реальном масштабе времени с ограниченным временем отклика, информация о событиях не отличается большим объемом, частая реконфигурация устройств, наличие интеллектуальных и неинтеллектуальных устройств, возможность работы с несколькими ведущими устройствами, длинные линии передачи, работа в условиях промышленных помех, повышенная достоверность передаваемой информации.

В соответствии с перечисленными задачами для разных уровней используются соответствующие сетевые технологии.

Для верхнего уровня наиболее широко применяется Ethernet и протокол TCP/IP.

На среднем и нижнем уровнях используют промышленные сети (fieldbus), к числу которых относятся **Profibus, CANbus, Modbus** и многие другие.

Принципы взаимодействия систем в сети описываются семиуровневой моделью взаимодействия открытых систем (Open System Interconnection, OSI, или модель ISO/OSI), разработанной Международной организацией по стандартам (International Standards Organization, ISO). Данная модель четко определяет различные уровни взаимодействия систем, дает им стандартные имена и указывает, какую работу должен выполнять каждый уровень. При этом уровни поддерживают интерфейсы с выше- и нижележащими уровнями.

Модель OSI описывает только системные средства взаимодействия, не касаясь приложений конечных пользователей, которые реализуют свои

собственные протоколы взаимодействия, обращаясь к системным средствам. Приложение может взять на себя функции некоторых верхних уровней модели OSI. В этом случае при необходимости межсетевого обмена оно обращается напрямую к системным средствам, выполняющим функции оставшихся нижних уровней.

При обращении пользовательского приложения с запросом к прикладному уровню модели (например, к файловому сервису) программное обеспечение указанного уровня формирует сообщение (message) стандартного формата, в которое помещает служебную информацию (заголовок) и, возможно, передаваемые данные. Затем это сообщение направляется представительному уровню. Представительный уровень добавляет к сообщению свой заголовок и передает результат вниз сеансовому уровню, который, в свою очередь, добавляет заголовок со служебной информацией данного уровня, и т.д. Некоторые реализации протоколов предусматривают наличие в сообщении не только заголовка, но и поля окончания пакета. Наконец, сообщение достигает самого низкого, физического уровня, который действительно передает его по линиям связи.

Когда сообщение по сети поступает на другую машину, оно последовательно перемещается вверх с уровня на уровень. Каждый уровень анализирует, обрабатывает и удаляет заголовок своего уровня, выполняет соответствующие данному уровню функции и передает сообщение вышележащему уровню. При обозначении единицы обмена данными (сообщений) в стандартах ISO для протоколов любого уровня используется термин «протокольный блок данных» - Protocol Data Unit (PDU). Кроме этого часто используются такие названия как кадр (frame), пакет (packet), дейтаграмма (datagram).

Физический уровень определяет электрические, конструктивные и информационные характеристики, обеспечивающие взаимодействие между конечными системами (устройствами). Спецификации физического уровня определяют такие характеристики, как величины напряжений, требования к параметрам импульсов, волновое сопротивление, способы кодирования информации, параметры синхронизации, скорость передачи физической информации, топологию, максимальные расстояния передачи информации, физические соединители (разъемы) и другие аналогичные характеристики. Физической средой в различных телекоммуникационных системах могут быть самые разнообразные средства от простейшей пары проводов до иерархической системы, использующей оптоволоконные и/или беспроводные каналы передачи цифровой информации.

Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня

выполняются сетевым адаптером. К оборудованию, которое работает только на физическом уровне, относится, например, репитер или повторитель шины, служащий для усиления сигналов при увеличении длины линии связи.

Примером протокола физического уровня может служить спецификация 10Base-T технологии Ethernet, которая определяет в качестве используемого кабеля неэкранированную витую пару категории 3 с волновым сопротивлением 100 Ом, разъем RJ-45, максимальную длину физического сегмента 100 метров, манчестерский код для представления данных на кабеле и другие характеристики среды и электрических сигналов.

На физическом уровне модели ISO/OSI [5] часто находят применение интерфейсы:

- EIA-RS-232-C, CCITT V.24/V.28 – механические/электрические характеристики несбалансированного последовательного интерфейса;
- EIA-RS-422/449, CCITT V.10 - механические, электрические и оптические характеристики сбалансированного последовательного интерфейса;
- RS-485;
- IEEE 802.3 - Ethernet;
- IEEE 802.5 - Token ring.

Следует отметить, что модель OSI представляет хотя и очень важную, но только одну из многих моделей коммуникаций. Эти модели и связанные с ними стеки протоколов могут отличаться количеством уровней, их функциями, форматами сообщений, сервисами, предоставляемыми на верхних уровнях, и прочими параметрами.

1. АЛГОРИТМ УПРАВЛЕНИЯ МОДУЛЯМИ

При проектировании алгоритмического обеспечения необходимо учитывать особенности обмена информацией в системах на базе модулей ICP DAS, функции, реализуемые модулями, и инструментальные средства, необходимые для разработки программного обеспечения выбранного измерительного комплекса.

В системах на базе модулей ICP DAS обмен информацией ведется по принципу Master-Slave. Ведущим устройством является ПЭВМ, управляющая работой автоматизированной системы в соответствии с ее назначением. Обмен информацией выполняется в кодах ASCII. Символы ASCII передаются в режиме асинхронного обмена 10-битовыми пакетами, соответствующими формату данных RS-232 ПЭВМ.

Конфигурация автоматизированной системы задается или хранится в памяти ПЭВМ. Идентификация модулей и их настроек выполняется на основе информации, записанной в EEPROM. Перед включением модуля в систему в EEPROM записываются коды идентификации модуля, микропрограммного обеспечения, режим работы (устанавливаемый по умолчанию), адрес, скорость обмена и ряд других параметров.

Структура кадра, формируемого ПЭВМ, представлена на рис. 1.

Код операции	Адрес модуля	Модификатор кода операции	Поле данных	Контрольная сумма	CR

Рис. 1. Структура кадра команды

Длина кадра может изменяться в зависимости от кода операции от 4 до 12 байт. В некоторых кадрах может отсутствовать поле данных и поле контрольной суммы.

Достоверность обмена обеспечивается подсчетом контрольной суммы по модулю 256, сторожевым таймером, пакетом подтверждения.

Контрольная сумма образуется как сумма всех кодов ASCII символов, входящих в строку команды или ответа модуля, за исключением кода символа CR (0Dh). Два младших 16-ричных разряда в полученной сумме являются контрольной суммой.

Рассмотрим пример определения контрольной суммы для команды $012(cr). Сумма кодов символов команды: “$” + ”0” + ”1” + ”2” = 24h + +30h + 31h + 32h = B7h. Контрольная сумма равна B7h, а команда с байта-

ми контрольной суммы имеет вид: $012B7(cr).

Ответ модуля - !01200600(cr). Сумма кодов символов ответа модуля: “!” + ”0” + ”1” + ”2” + ”0” + ”0” + ”6” + ”0” + ”0” = 21h + 30h + 31h + 32h + +30h + 30h + 36h + 30h + 30h = 1AAh. Контрольная сумма равна AAh, ответ модуля с байтами контрольной суммы: $01200600AA(cr).

Длина пакета подтверждения, формируемого модулем, также зависит от кода операции и находится в диапазоне от 1 до 56 байт (при чтении каналов АЦП 7018). Как правило, пакет подтверждения при успешном выполнении команды содержит символ «!» и адрес модуля. Если по какой-либо причине команда не выполнена, то передается символ «?». **Однако возможны и исключения из этих правил.**

Кадр начинается полем кода операции и завершается кодом CR (возврат каретки 0Dh).

Система команд используемых модулей приведена в приложениях П.1-П.6.

При кодировании команд приняты следующие обозначения:

%, $, #, ~, @ - символ, определяющий группу команд;

AA - текущий адрес модуля (00÷FF);

NN - новый адрес модуля;

TT - код режима работы (I-7044, I-7080), тип аналогового входа (I-7018, I-7013(33)) или выхода (I-7021);

CC - код скорости передачи по RS-485;

FF - код формата данных, выбора режекторного фильтра и контрольной суммы;

CHK - значение контрольной суммы по модулю 256 (2 символа). Если формирование контрольной суммы запрещено, то этот параметр в кадре отсутствует;

CR - символ окончания кадра (0D0h);

! - начало кадра подтверждения, если **команда** допустима. **Исключение составляет модуль I-7021, в котором по команде #AA(данные) символ «!» используется как признак недопустимой команды**;

> - начало кадра подтверждения при допустимой команде, если передаются или принимаются **данные;**

? - начало кадра подтверждения, если команда недопустима. Указывает на наличие ошибки кадра, нарушение требуемой последовательности команд и т.д. **В случае**

синтаксической или коммуникационной ошибки ответное сообщение может быть не принято;

(данные) - значение входного или выходного сигнала модулей. Данные кодируются в формате, соответствующем полю FF команды конфигурации. При кодировании команд формат данных должен строго соответствовать формату, указанному в соответствующих таблицах документации модулей. При нарушении длины формата фиксируется ошибка.

Остальные символы, используемые при кодировании команд, будут разъяснены по мере их применения.

В зависимости от задач, стоящих перед системой, выбирается количество и типы модулей, распределяются адреса на шине RS-485, выполняются необходимые электрические соединения.

На подготовительном этапе первичные параметры модулей могут быть установлены и проверены DCON Utility.

Перед началом работы следует разработать интерфейс пользователя, который бы позволял выполнять основные процедуры по сбору, обработке, отображению и документированию информации.

Пользовательский интерфейс (англ. user interface, UI) является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и компьютера. Лучший пользовательский интерфейс - это интерфейс, которому пользователь не должен уделять много внимания, почти не замечать его. Чтобы создать эффективный интерфейс, который делал бы работу с программой приятной, нужно понимать, какие задачи будут решать пользователи с помощью данной программы и какие требования к интерфейсу могут возникнуть у пользователей. Большую роль играет интуиция - если разработчик сам терпеть не может некрасивые и неудобные интерфейсы, то при создании собственной программы он будет чувствовать, где и какой именно элемент нужно убрать или добавить. Необходимо иметь художественный вкус, чтобы понимать, что именно придаст интерфейсу красоту и привлекательность.

Если говорить о самых **общих принципах проектирования пользовательских интерфейсов**, то можно назвать три основных положения:

1. Интерфейс должен помогать выполнить задачу, а не становиться этой задачей.

2. При работе с интерфейсом пользователь должен ощущать себя комфортно.

3. Интерфейс должен работать так, чтобы пользователь не считал компьютер дураком.

Одним из самых важных свойств интерфейса UI является его простота. Если интерфейс приложения сложен, то, скорее всего, и самим приложением трудно пользоваться. Если на форме много элементов управления, необходимо разбить их по вкладкам или связанным формам. Редко применяемые функции стоит выносить в отдельные формы. Для элементов интерфейса необходимо задавать значения по умолчанию. Сложные задачи можно упростить с помощью мастера.

Каждая функция программы должна иметь быстрый и интуитивно простой доступ к ней;

Программа должна учесть ситуации, в которых пользователю не обойтись без помощи, поэтому она должна включать в себя как встроенную справочную систему, так и печатную документацию. Кроме того, можно добавить всплывающие подсказки, статусные строки, подсказки и мастера.

Обобщенный алгоритм работы системы на базе ICP DAS представлен на рис. 2.

Работа системы начинается с процедуры проверки конфигурации, в которой определяется состав модулей и их адреса. По результатам проверки могут быть внесены соответствующие изменения.

В соответствии с особенностями решаемых задач определяется конфигурация каждого модуля, которая контролируется с помощью интерфейса пользователя.

В зависимости от условий работы системы принимается решение об использовании сторожевого таймера WD и настройки его параметров.

После настройки модулей разрабатывается программа сбора и обработки данных. При использовании WD в каждом цикле выполнения текущей команды формируется перезапуск сторожевого таймера командой «~**».

Обработка команды модулем включает последовательную проверку следующих параметров:

- скорость обмена по интерфейсу RS-485;
- код операции;
- адрес модуля;
- анализ контрольной суммы, если она предусмотрена;
- корректность выполнения операции;
- корректность завершения кадра.

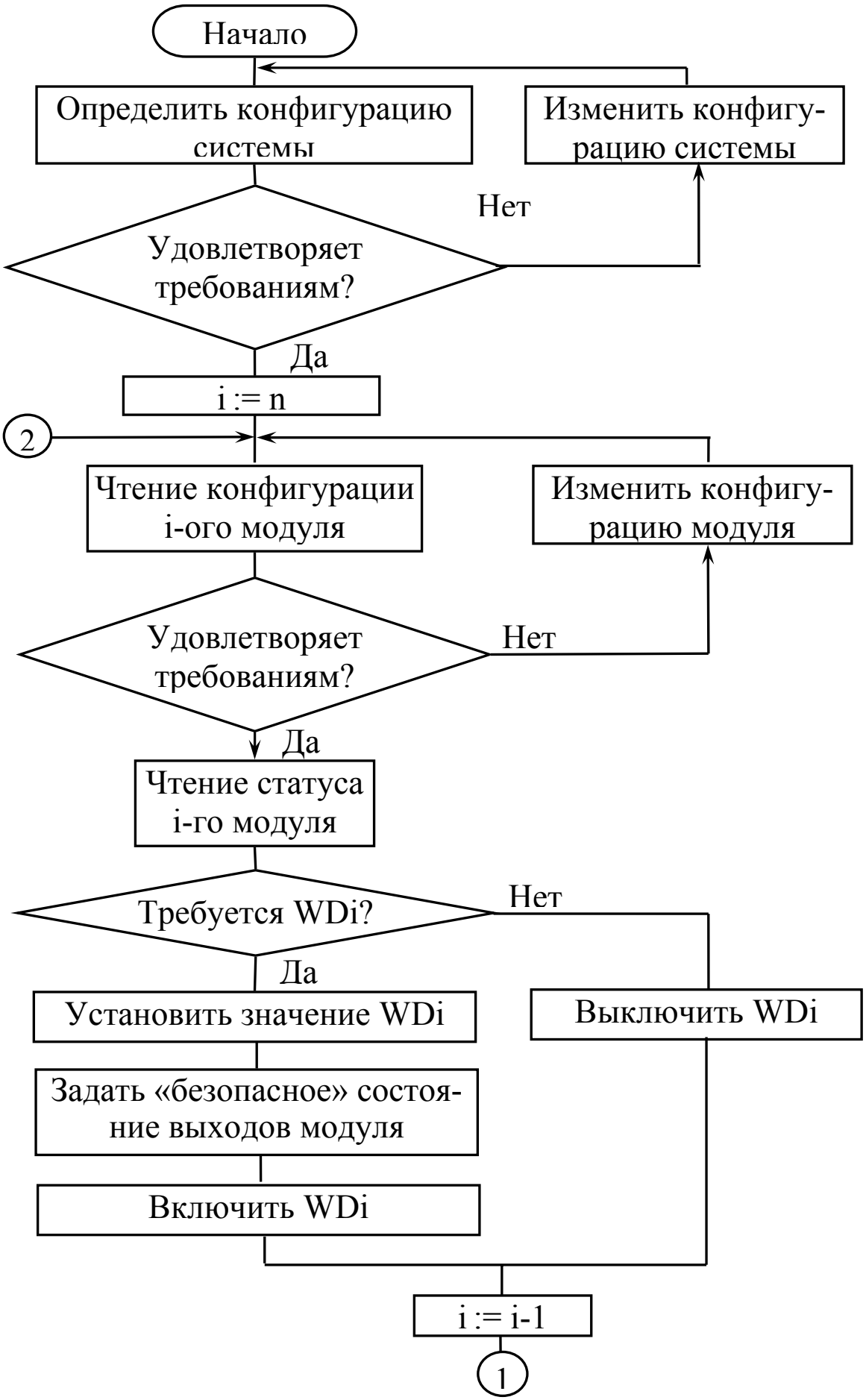
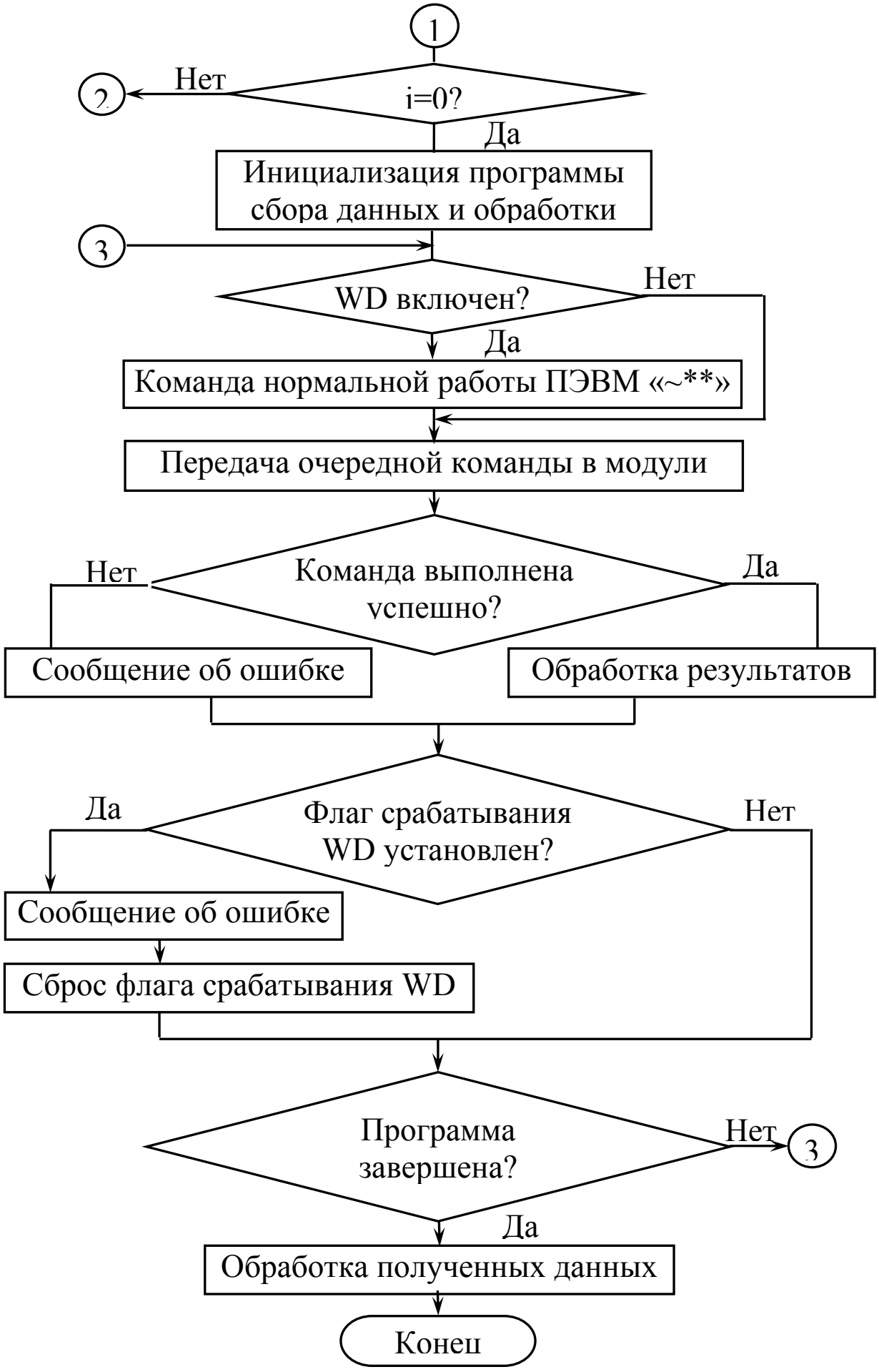


Рис. 2. Обобщенный алгоритм работы модуля,



совместимого с модулями ISP DAS

Если все условия выполнены, то формируется кадр подтверждения, первый байт которого содержит символ «!» или «>». Однако имеются исключения. Например, в команде #AA(данные) модуля I-7021 символ «!» используется как признак недопустимой команды. В случае обнаружения несоответствия между командой и внутренними настройками модуля формируется кадр недопустимой команды «?AA», где AA-адрес модуля.

Аналогичное сообщение может быть получено, если нарушен порядок следования команд. Например, при выполнении калибровки, если предварительно не была исполнена команда «разрешить калибровку».

Для повышения достоверности информации используют команды «задать значение», «проверить значение». Однако при повышенных требованиях по быстродействию подобного рода проверки можно минимизировать.

Контроль флага срабатывания WD позволяет предотвращать «зависание» системы.

В зависимости от особенностей алгоритма работы системы обработка данных может выполняться после выполнения очередной команды и/или в конце процедуры сбора информации (например, при построении графиков).

Считывание входной информации с модулей может быть выполнено в асинхронном или синхронном режимах.

Синхронный режим используется в том случае, когда необходимо зафиксировать значения входных параметров в нескольких модулях одновременно.

Для реализации синхронного режима следует использовать команду синхронизированной выборки «#**», по которой модули, поддерживающие данный режим, запоминают входную информацию во внутренних регистрах. Далее информация из этих регистров считывается асинхронно.

2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Современные средства автоматизации, как правило, поставляются с набором драйверов и компонентов, предоставляющих пользователю возможность выбирать инструментальные средства в соответствии со своими предпочтениями и наличием лицензионных программных продуктов.

В качестве технологии взаимодействия с компонентами системы автоматизации разработчик может выбирать один из методов: использование драйверов устройств, управление посредством динамически подключаемых библиотек или технологии OPC, SCADA (Supervisory Control And Data Acquisition) системы, разработка собственного интерфейса. Фирма ICP DAS обеспечивает необходимыми компонентами для разработки приложений в различных операционных системах: DOS, Windows, Linux.

В системах управления технологическими процессами широко применяются многочисленные программные решения (например, SCADA системы) разных производителей. Работа этих программных систем базируется на постоянном обмене данными с компонентами системы автоматизации (контроллерами, модулями УСО и т.д.). Возможность такого взаимодействия обеспечивается производителями программных решений путем разработки драйверов, которые интегрируются в вышеназванные программные пакеты. Однако такой подход имеет недостатки:

- увеличение затрат - должны разрабатываться отдельные драйверы для каждого поддерживаемого устройства;
- ограниченная функциональность драйверов - разработчиком драйверов поддерживаются не все функции соответствующего устройства;
- ограниченные возможности расширения и изменения состава компонент системы автоматизации - вследствие модернизации аппаратной платформы драйвер может работать нестабильно либо вообще перестать функционировать;
- конфликты доступа - различные программы не могут одновременно осуществлять доступ к одним и тем же компонентам системы автоматизации, т.к. обращение к данным осуществляется через собственные драйверы, работа одного из которых в каждый момент времени блокирует возможность работы остальных.

Динамически подключаемые библиотеки (DLL) - это модули, которые содержат определенный набор функций и данных и подгружаются в память при необходимости. Помимо экономии памяти DLL имеют два важных преимущества.

1) DLL загружаются в память один раз и позволяют использовать одни и те же функции и данные в различных приложениях (или копиях приложения) без их дублирования;

2) библиотеки DLL могут обновляться без перекомпиляции использующей их программы. Соответственно улучшение функциональности библиотек не затронет код программ.

Помимо указанных преимуществ динамически подключаемые библиотеки имеют ряд недостатков:

1) изменения в функциональности библиотек могут быть столь принципиальны, что использующая их программа перестанет работать;

2) сложная система получения информации об ошибках;

3) отсутствие единых стандартов на функциональность может привести к тому, что разные производители реализуют несовместимые библиотеки для одних и тех же устройств;

4) DLL библиотеки содержат «неуправляемый код», который может вывести приложение из строя.

Большое количество программ в области промышленной обработки данных реализуются в настоящее время на базе персональных компьютеров под управлением операционных систем семейства Windows (Windows 95/98/NT/2000/XP/Vista) фирмы Microsoft. Для решения коммуникационных проблем фирмой Microsoft была предложена технология OPC, ставшая в настоящее время промышленным стандартом.

OPC (OLE for Process Control) – это стандарт взаимодействия между программными компонентами системы сбора данных и управления (SCADA), основанный на объектной модели COM/DCOM. Технология OPC предназначена для обеспечения:

- универсального механизма обмена данными между датчиками, исполнительными механизмами, контроллерами, устройствами связи с объектом и системами представления технологической информации;
- оперативного диспетчерского управления;
- архивации данных СУБД.

Через интерфейсы OPC одни приложения могут читать или записывать данные в другие приложения, обмениваться событиями, оповещать друг друга о нештатных ситуациях (тревогах), осуществлять доступ к данным, зарегистрированным в архивах («исторические» данные). Эти приложения могут располагаться как на одном компьютере, так и быть распределенными по сети. При этом независимо от

фирмы-поставщика стандарт OPC, признанный и поддержанный всеми ведущими фирмами-производителями SCADA-систем и оборудования, обеспечит их совместное функционирование. Особый класс OPC-приложений представляют собой OPC-серверы конкретных аппаратных устройств – они поставляются многими производителями аппаратных средств. OPC-сервер создает своего рода абстракцию аппаратуры, позволяя любому OPC-клиенту записывать и считывать данные с устройства. Устройство, для которого есть OPC-сервер, может использоваться вместе с любой современной SCADA-системой.

Среди недостатков необходимо отметить следующие:

1) разработчику доступны только средства обмена данными. Доступ к параметрам модулей не реализован в сервере, что значительно ограничивает возможности применения данного продукта. Конфигурация модулей может производиться только дополнительной утилитой при обязательном условии остановки работы сервера;

2) структуры данных, описанные в NAPOPC, имеют обобщенный характер и не обеспечивают возможности конкретизации типов каналов передачи данных;

3) механизмы работы сервера с оборудованием скрыты от разработчика. Так, например, невозможно программно проверить использование сторожевого таймера.

Если перечисленные выше технологии из-за своих недостатков не могут быть использованы в автоматизированной системе, разработчик может создать собственный протокол взаимодействия, что позволит реализовать все необходимые функции работы с модулями.

Протокол взаимодействия – это программная конструкция, описывающая некоторую функциональность, не указывая при этом особенности ее реализации. Таким образом, интерфейс обобщает требования к классам, реализующим интерфейсы. В итоге уменьшается степень связанности отдельных компонентов системы, что позволяет изменять реализацию интерфейсов, не меняя при этом реализацию других компонентов.

В состав программного обеспечения, поставляемого фирмой ICP DAS, входит также платформа графического программирования LabVIEW, предназначенная для комплексной автоматизации промышленного производства.

Среда LabVIEW (Laboratory Virtual Instruments Engineering Workbench) – разработка компании National Instruments (NI) [12], позволяющая инженерам и ученым (не специалистам в области программирования) бы-

стро создавать собственные контрольно-измерительные системы (виртуальные приборы), максимально отвечающие их потребностям. Данная система разработки виртуальных приборов представляет собой среду прикладного графического программирования, которая используется в качестве стандартного инструмента для проведения измерений, анализа данных, управления приборами и исследуемыми объектами. В последнее время платформа LabVIEW получает все большее распространение в промышленности и образовании, при проведении научных исследований и выполнении проектных работ. Этому способствует ее несомненное преимущество – высокая производительность при разработке программ, широкий набор функциональных возможностей языка и среды программирования.

При разработке программных систем в текстовом виде все чаще используется компонентная архитектура, где программа представляется в виде совокупности компонент с простыми и четко специфицированными интерфейсами. Этот подход позволяет разрабатывать компоненты независимо, реализовывать их так, чтобы иметь возможность работы в распределенной среде, модифицировать одни компоненты программного обеспечения без изменения других, и т.д. Возможности компонентной архитектуры активно использует технология быстрой разработки приложений (RAD - Rapid Application Development), созданной на основе визуального объектно-ориентированного программирования. Она заключается в возможности наглядно конструировать пользовательский интерфейс с помощью манипулятора мышь, исключая написание объемных текстов программ. Среда интерфейса Visual Basic была первой, использующей данную технологию, затем появились Borland Delphi и Borland Builder, MS Visual C# .NET. Каждая среда ориентирована на свой язык программирования - Basic, Pascal, C++ или C#.

Наибольшую популярность в странах СНГ получила коммерческая среда разработки под названием Delphi. Реализация языка Delphi проектом Free Pascal (полное название Free Pascal Compiler) позволяет использовать его для создания приложений не только в Windows, но и Mac OS X, Windows CE и Linux. В настоящее время в рамках проекта разработана среда Lazarus – свободно распространяемый аналог среды программирования Delphi и Lazarus Components Library (LCL) - библиотека компонентов, аналогичная Delphi VCL [7].

Инструментальные средства Microsoft Visual C++, включая редакторы ресурсов, существенно облегчают работу по Win32-программированию. Имеется возможность разработки программ на C++ с использованием каркаса приложений – библиотеки MFC. Как правило, в них реализована архитектура «документ-вид», разделяющая данные и их представление, что позволяет представлять одни и те же данные по-разному.

С появлением платформы .NET, Microsoft предложила набор инструментов, облегчающих разработку программ для разных сред - Visual Studio .NET, а также новый язык программирования C# (C Sharp) адаптированный под .NET.

Язык C# создан для того, чтобы предоставить программисту простоту использования Visual Basic и, в случае необходимости, высокопроизводительный, низкоуровневый доступ к памяти по типу C++. На языке C# не рекомендуется разрабатывать программы, требующие повышенного быстродействия и высокой производительности.

Язык C++ останется в этой области лучшим из языков низкого уровня.

В C# отсутствуют средства, необходимые для создания высокопроизводительных приложений, в частности подставляемые функции и деструкторы, выполнение которых гарантируется в определенных точках кода. Тем не менее, число приложений, попадающих в эту категорию, невелико.

Далее в этом разделе приведены примеры создания пользовательских приложений, управляющих работой модулей серии I-7000. За основу взяты наиболее распространенные среды визуального программирования (Lazarus, Borland Delphi, Microsoft Visual C++, Microsoft Visual C# .NET) и графического- LabVIEW.

3. Методика настройки программного обеспечения I-7000

Для управления модулями семейства I-7000 из программ пользователя необходимо воспользоваться CD-ROM, поставляемым с модулями. На этом диске находятся все необходимые драйверы устройств, утилиты, а также руководство пользователя. При запуске диска открывается окно программы автозагрузки, представленное на рис. 3.

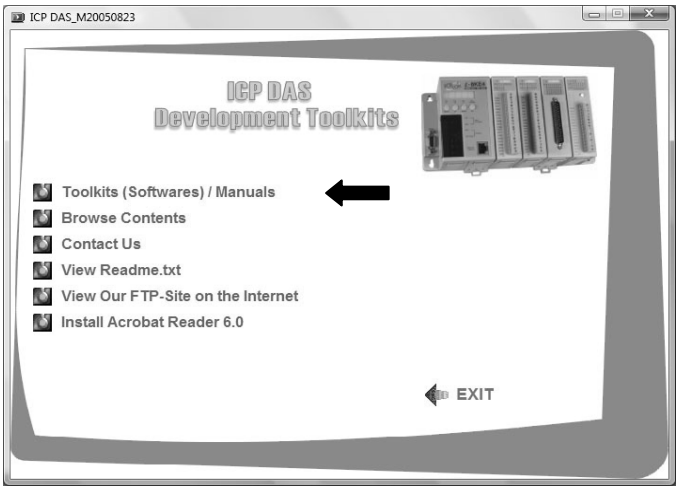


Рис. 3. Окно программы автозагрузки

При выборе первого раздела «Toolkits (Softwares) / Manuals» открывается окно, показанное на рис. 4. Здесь пользователь имеет возможность получить информацию по модулям различных серий, а также установить утилиты и драйвера.

Рис. 4. Содержимое раздела «Toolkits (Softwares) Manuals»

Так после выбора раздела «7000/8000/87K Series Toolkits (with DCON protocol)» открывается список утилит (см. рис. 26).

На рис. 5 показано меню, с помощью которого можно установить драйверы интерфейсных преобразователей (I-7560, I-7561, I-7563), утилиты взаимодействия с модулями из различных операционных систем (Windows, DOS, LINUX), с использованием различных механизмов доступа (DDE, DLL, ActiveX), а также управления на уровне виртуального прибора

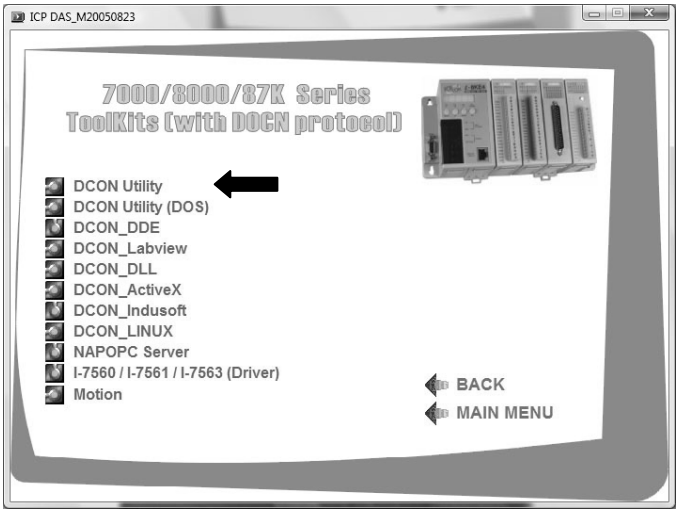


Рис. 5. Список утилит раздела «7000/8000/87K Series Toolkits (with DCON protocol)»

LabView.

Для подключения модулей I-7000 к компьютеру через универсальную последовательную шину (USB) необходимо установить драйвер используемого преобразователя интерфейса. Например, если в качестве преобразователя интерфейса используется модуль I-7561, установка драйвера производится в следующей последовательности:

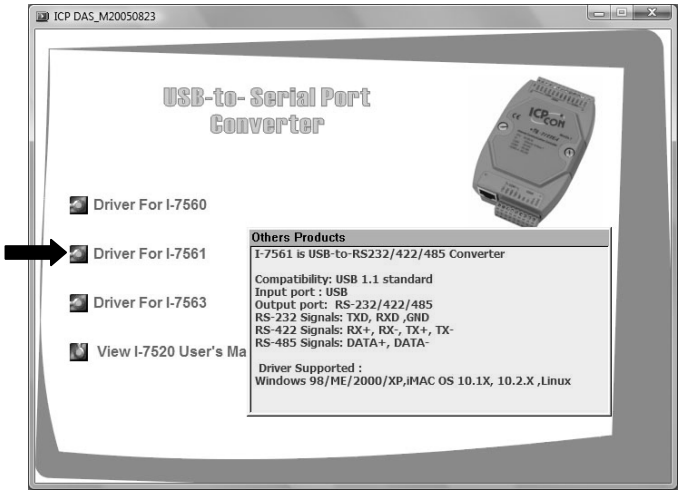


Рис. 6. Выбор драйвера для преобразователя интерфейса

1) Выбрать ссылку «I-7560/I-7561/I-7563(Driver)» (рис. 27).

2) При выборе ссылки «Driver For I-7561» (рис. 6), открывается окно проводника (рис. 7) с программой конфигурации порта, к которому подключается устройство I-7561, и драйвером устройства.

Рис. 7. Программа конфигурации COM-порта

Теперь можно подключить интерфейсный преобразователь I-7561 к порту USB. Появится предупреждение о найденном оборудовании, в окне установки новых устройств необходимо указать путь к драйверу устройства и установить его.

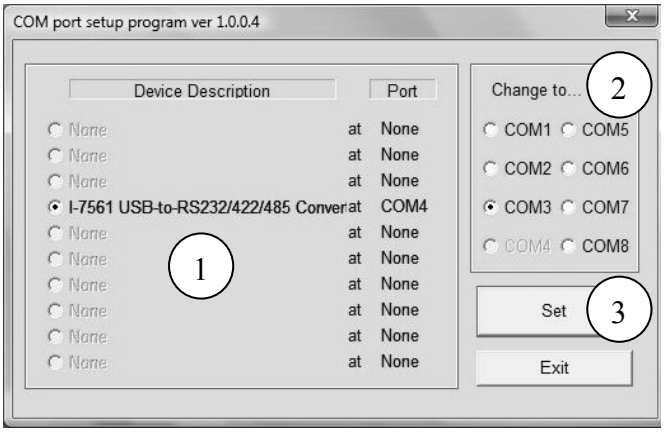


Рис. 8. Интерфейс программы настройки COM-порта.

3) Для конфигурации порта необходимо запустить приложение «SetCOM.exe» (данное действие не является обязательным и выполняется лишь при необходимости смены номера виртуального порта, к которому подключен преобразователь интерфейсов). После запуска в системном трее (system tray) появится соответсвующий значок, выбрав который, можно запустить программу

«COM port setup program» (рис. 8). Цифрами показана последовательность действий при настройке порта. Как видно из рис. 8, устройство «I-7561 USB-to-RS232/422/485 Converter», подключенное к виртуальному последовательному порту COM4, настраивается на взаимодействие с виртуальным портом COM3. Для успешного изменения параметров подключения устройства необходимо нажать кнопку «Set» и перезагрузить компьютер.

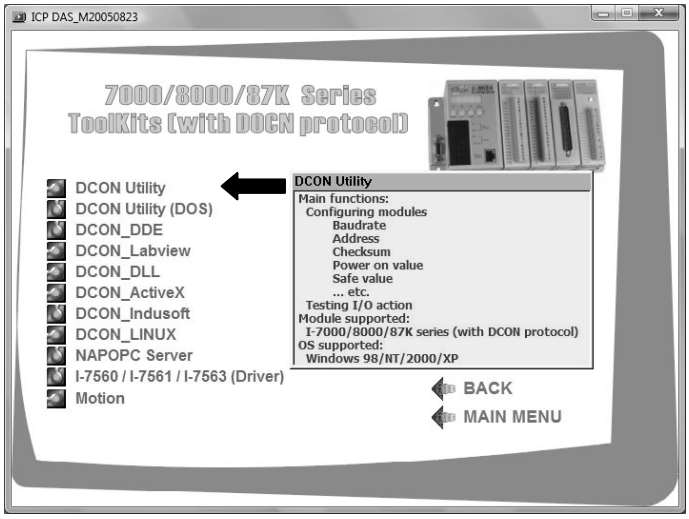


Рис. 9. Меню выбора утилиты DCON

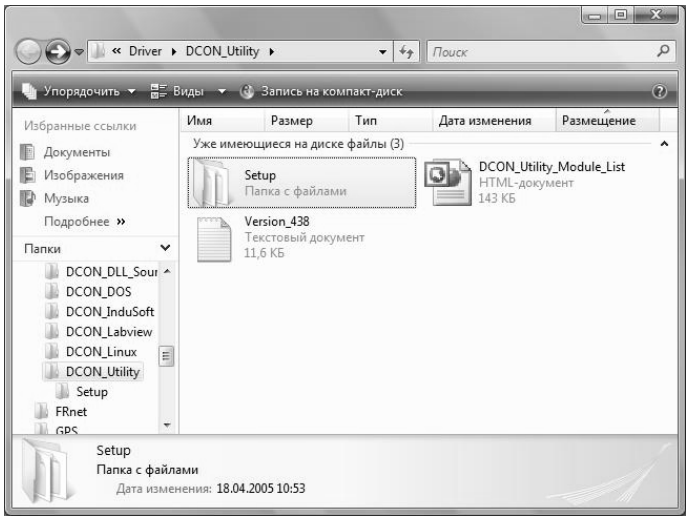


Рис. 10. Каталог утилиты DCON

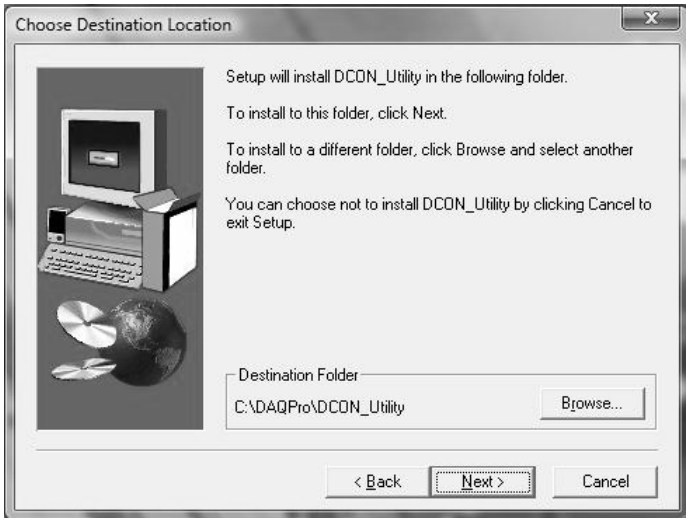


Рис. 11. Страница выбора каталога назначения

Примечание: При использовании преобразователя интерфейса I-7520AR установка дополнительных драйверов не требуется, управление преобразователем осуществляется через COM-порт, к которому оно подключено.

Проверка работоспособности модулей осуществляется утилитой «DCON Utility». Для ее установки необходимо выбрать первую ссылку (см. рис. 9).

В открывшемся окне проводника находится подкаталог с дистрибутивом утилиты, описанием ее версии и списком поддерживаемых устройств (рис. 10).

Запустив программу – инсталлятор «Setup.exe» из каталога «Setup», необходимо руководствоваться командами мастера установки. Мастер требует указать путь к каталогу назначения «Destination Folder» - ка-

талог, в который устанавливается программа (рис. 11), название каталога программы в меню ОС Windows «Пуск/Программы» (рис. 12), а также предоставляет информацию мастера установки (рис. 13).

На последней вкладке (рис. 14) мастер установки предлагает загрузить файл с описанием возможностей утилиты «Yes, I want to view the Readme file now.», а также загрузить утилиту «Yes, I want to launch DCON_Utility now.». Окно утилиты представлено на рис. 15.

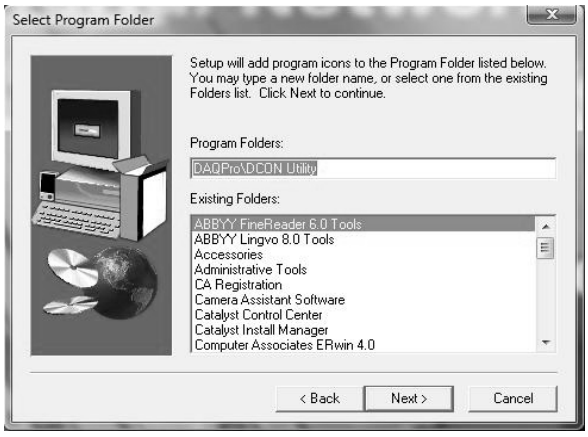


Рис. 12. Страница выбора каталога программы

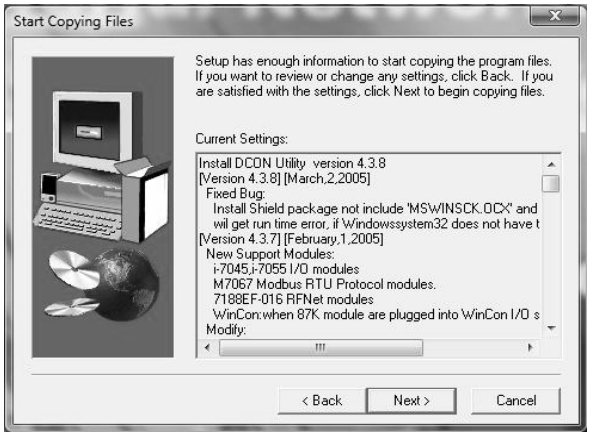


Рис. 13. Страница текущих настроек мастера установки

Рис. 14. Последняя страница мастера установки

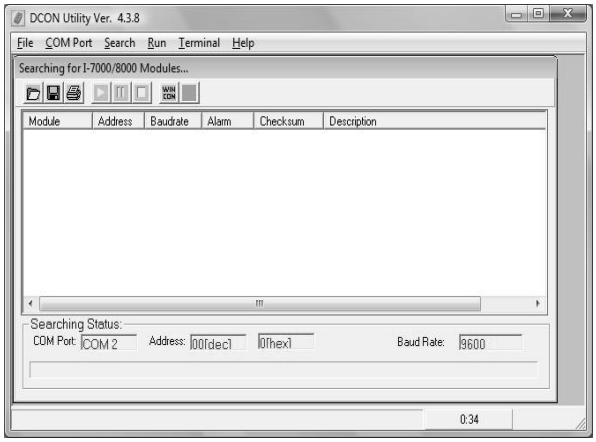


Рис. 15. Интерфейс утилиты DCON

Работа с утилитой начинается с задания номера COM-порта (Com to search), к которому подключен преобразователь интерфейса, указания скорости передачи (Baud rate to search), в некоторых случаях требуется указать время таймаута (Time Out Setting) и необходимость проверки контрольной суммы (To Search Checksum Enabled Module?). Перечисленные настройки задаются в диалоговом окне, доступном через пункт меню «COM Port» (рис. 16).

Как видно из рис. 16, утилита сконфигурирована на сканирование модулей, подключенных посредством преобразователя интерфейса к последовательному порту COM5 на скоростях 9600 и 115200, при этом проверяется контрольная сумма, время таймаута составляет 200 мс.

Нажав кнопку «Начать сканирование (Start Search)» утилита опрашивает подключенные модули на указанных скоростях, в результате чего формируется список модулей (рис. 17).

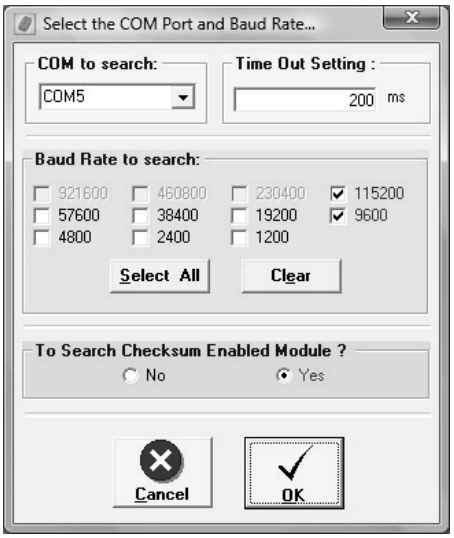


Рис. 16. Диалоговое окно настройки последовательного порта

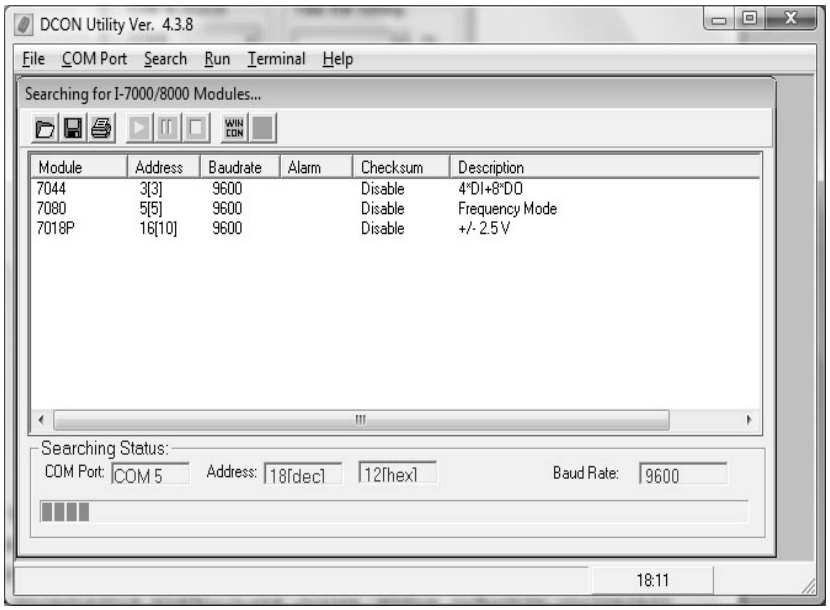


Рис. 17. Результат поиска подключенных к компьютеру модулей

Таким образом, с помощью утилиты сформирован список модулей, подключенных к системе, произведена настройка портов для управления устройствами с помощью компьютера. Теперь можно приступать к разработке пользовательских программ управления модулями.

4. Сбор данных и управление модулями

Управление модулями на уровне ПЭВМ сводится к асинхронному обмену информацией через COM-порт (при использовании преобразователя интерфейсов I-7520AR) и/или виртуальный COM-порт, если используется преобразователь интерфейсов I-7561.

Рассмотрим пример взаимодействия с произвольным модулем ICP DAS через COM-порт в режиме командной строки. В программе должна обеспечиваться возможность задания параметров COM-порта: номер, скорость обмена, формат посылки. При передаче команды необходимо обеспечить возможность указания флага проверки контрольной суммы, а также установки величины таймаута. После выполнения команды ответное сообщение должно отображаться на интерфейсе пользователя.

В следующих разделах рассматриваются вопросы управления модулями из программ, разработанных с использованием различных сред проектирования: LabVIEW, Lazarus, Borland Delphi, MS Visual C++, MS Visual C#.

4.1. Управление в SCADA системе LabVIEW

Для обеспечения доступа к модулям ICP DAS в виртуальном приборе LabVIEW необходимо установить утилиту DCON_LabVIEW (рис.18).

После окончания установки создается каталог, по умолчанию C:\DAQPro\DCON_LabVIEW, содержащий: описание принципов управления модулями из LabVIEW; библиотеки файлов LabVIEW, обеспечивающие взаимодействие через *.dll с модулями различных серий, а также COM-портом; динамически подключаемые библиотеки (I7000.dll, Uart.dll), обеспечивающие программный интерфейс COM-порта и модулей I-7000; подкаталоги с демонстрационными виртуальными приборами (рис.19).

При разработке виртуального прибора за основу взят пример «Send_Receive_command.vi» из каталога «..\DAQPro\DCON_LabVIEW\others». Интерфейс этого прибора доработан и переведен на русский язык (рис. 20).

Основой виртуального прибора является последовательность (Stacked Sequence Structure), содержащая четыре страницы (рис. 21-23, 27).

На первой вкладке определяется статус порта (рис. 21). На второй COM-порт открывается для управления из виртуального прибора (рис. 22).

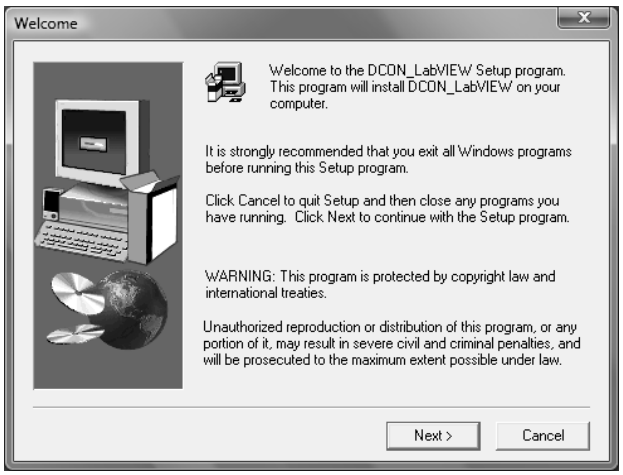


Рис. 18. Окно мастера установки «DCON_LabVIEW»

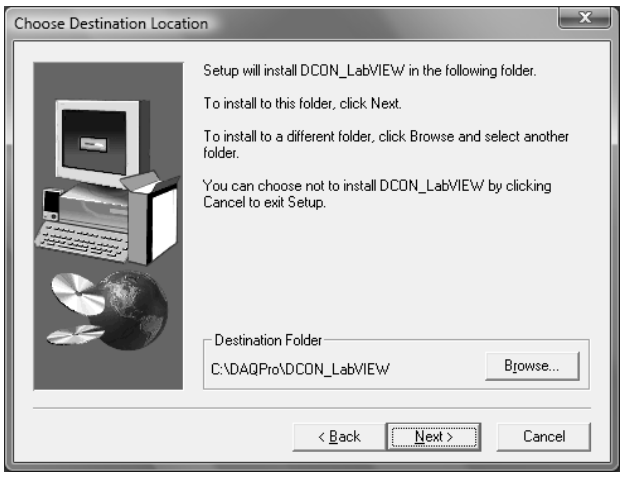


Рис. 19. Окно настройки каталога назначения

На следующей вкладке вызывается субвиртуальный прибор посылки команды в модуль и приема ответного сообщения (см. рис. 23).

Основой указанного субвиртуального прибора является вызов команды «Send_Receive_Cmd» из динамически подключаемой библиотеки «Uart.dll» (см. рис. 24).

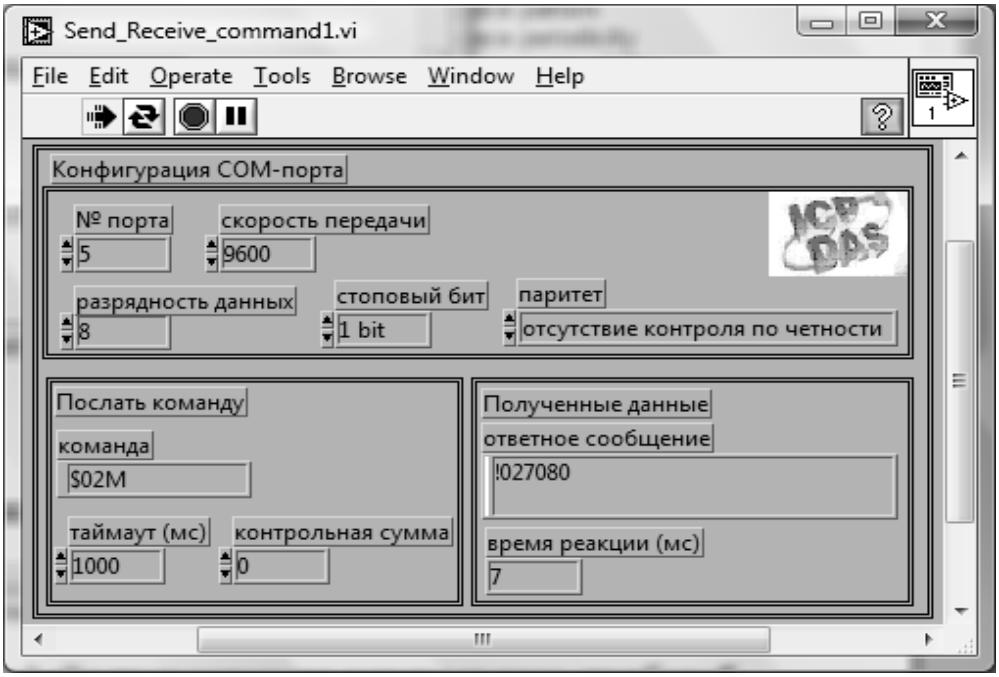


Рис. 20. Интерфейс пользователя виртуального прибора

Примечание: Следует заметить, что для функционирования виртуального прибора, содержащего узел вызова функции библиотеки «Call Library Function Node», необходимо задать путь к используемому файлу библиотеки *.dll. Для этого в контекстном меню функционального узла выбирается пункт «Configure…» (рис. 25), в открывшемся окне необходимо задать имя используемой библиотеки, название вызываемой функции, входные параметры функции и тип возвращаемого параметра (рис. 26).

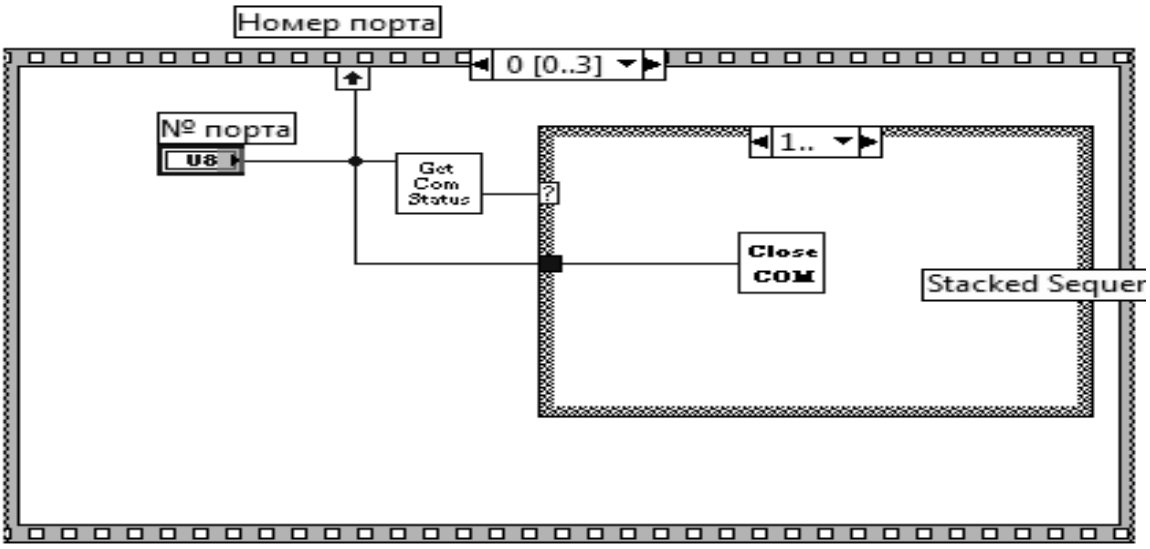


Рис. 21. Первая страница структуры «последовательность»

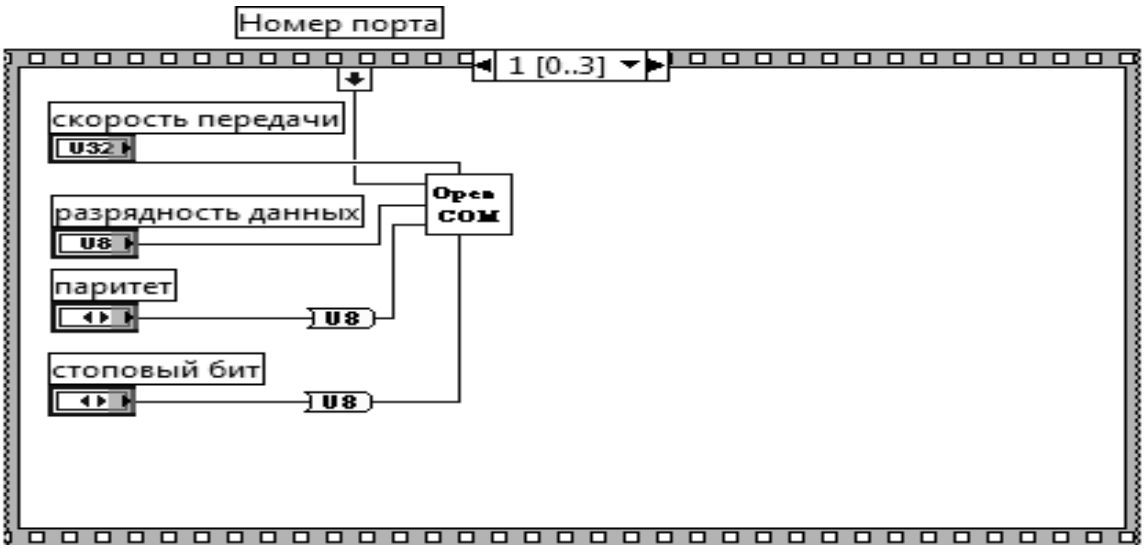


Рис. 22. Вторая страница структуры «последовательность»

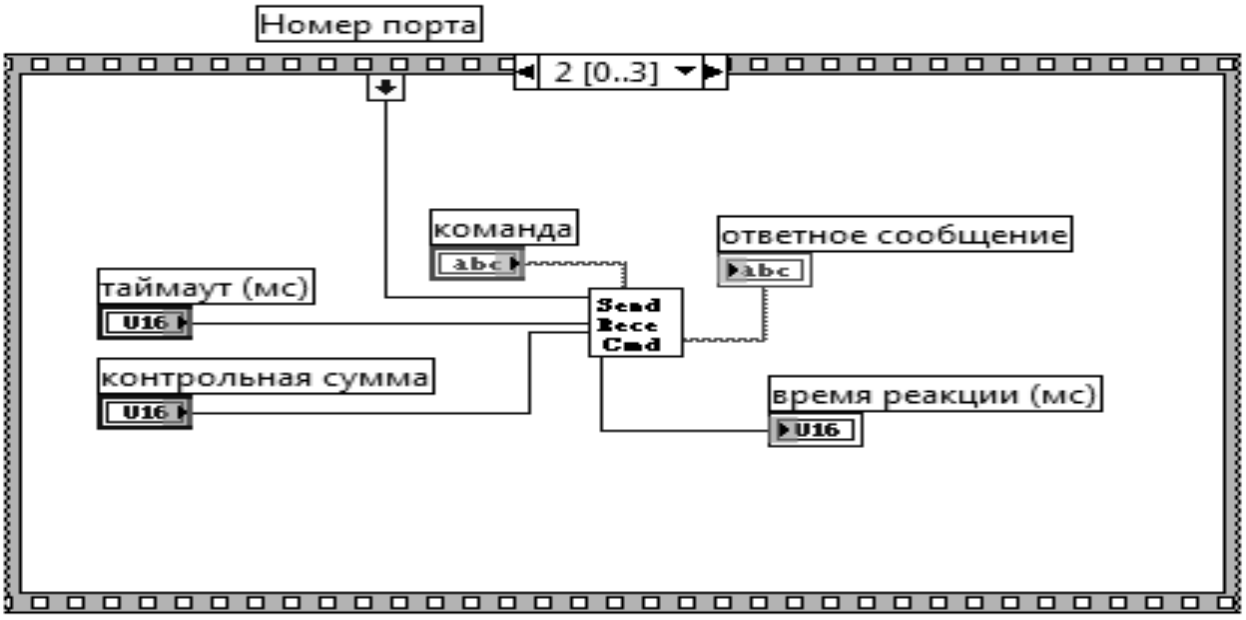


Рис. 23. Третья страница структуры «последовательность»

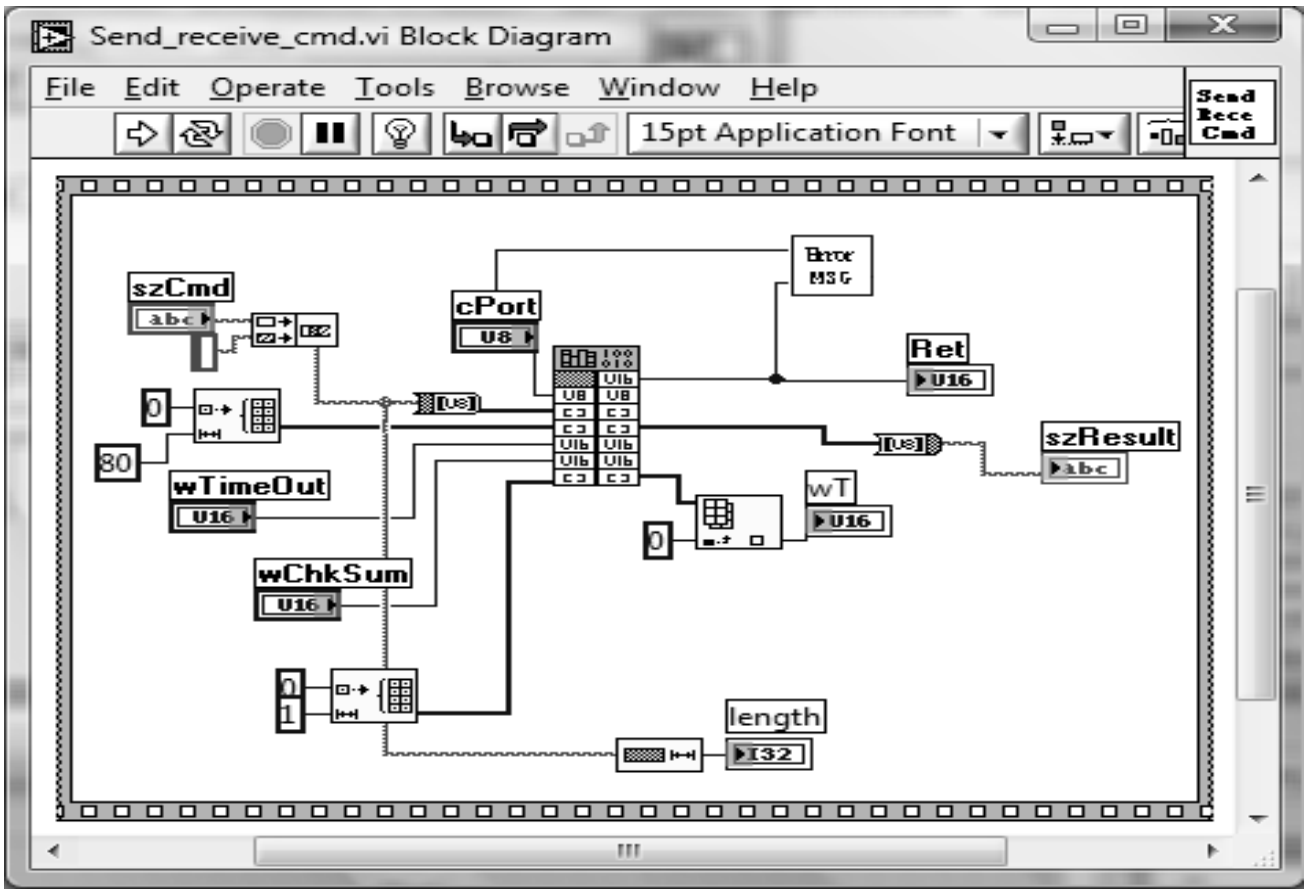


Рис. 24. Блок диаграмм субвиртуального прибора «Send_receive_cmd.vi»

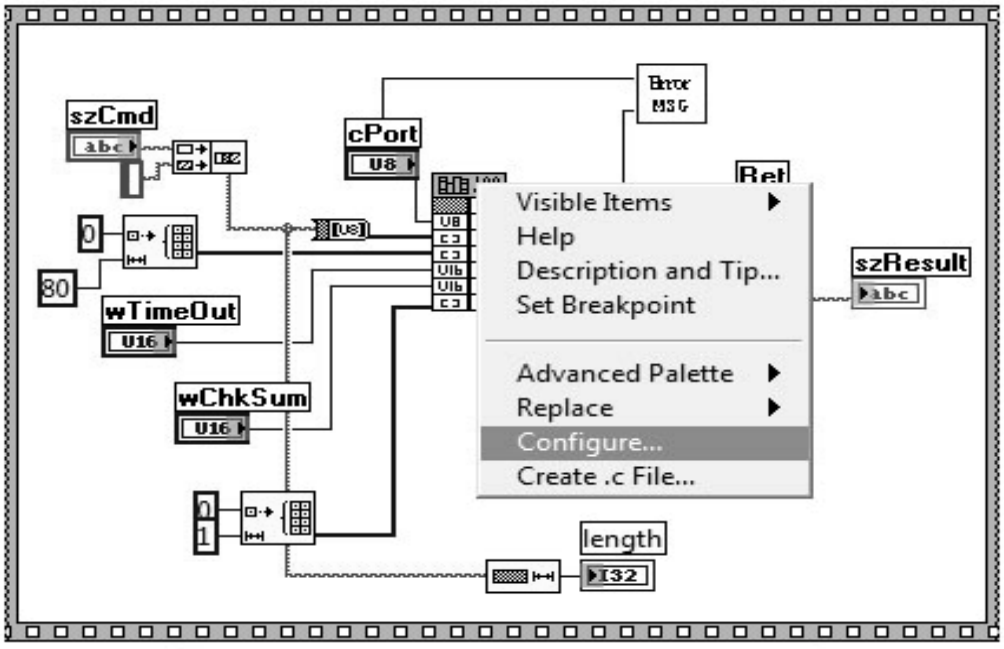


Рис. 25. Элемент меню «Configure…» для задания параметров вызываемой функции библиотеки

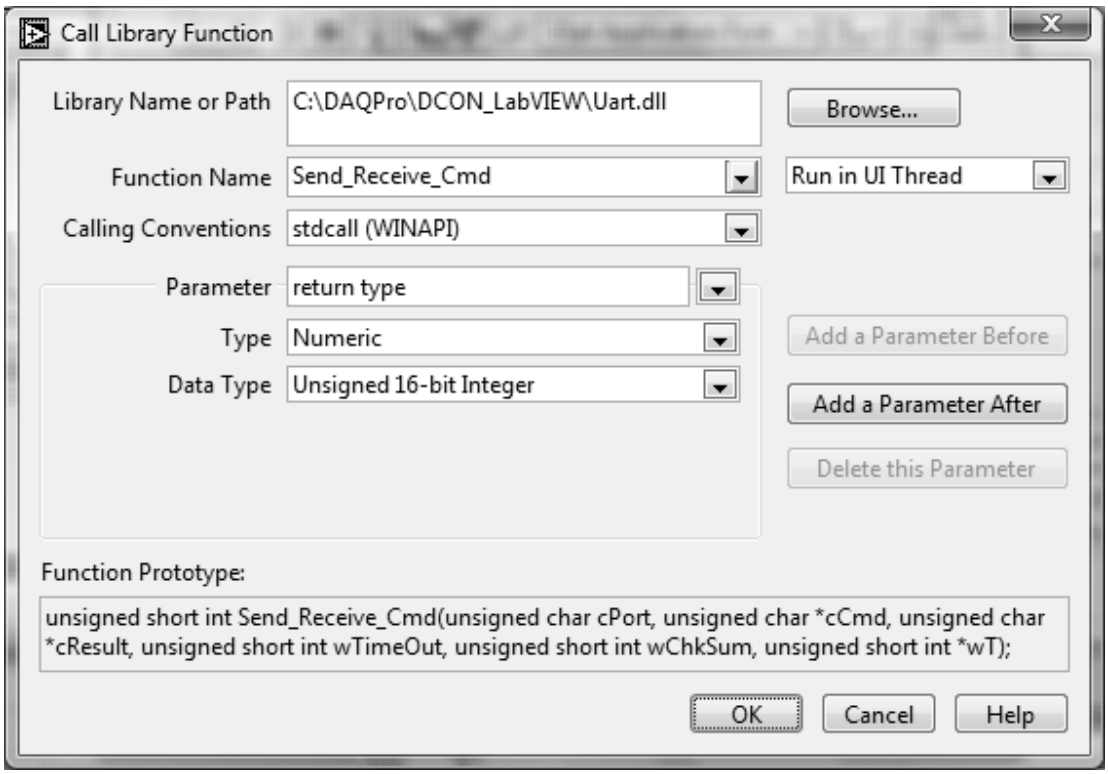


Рис. 26. Окно настройки параметров функции
динамически подключаемой библиотеки

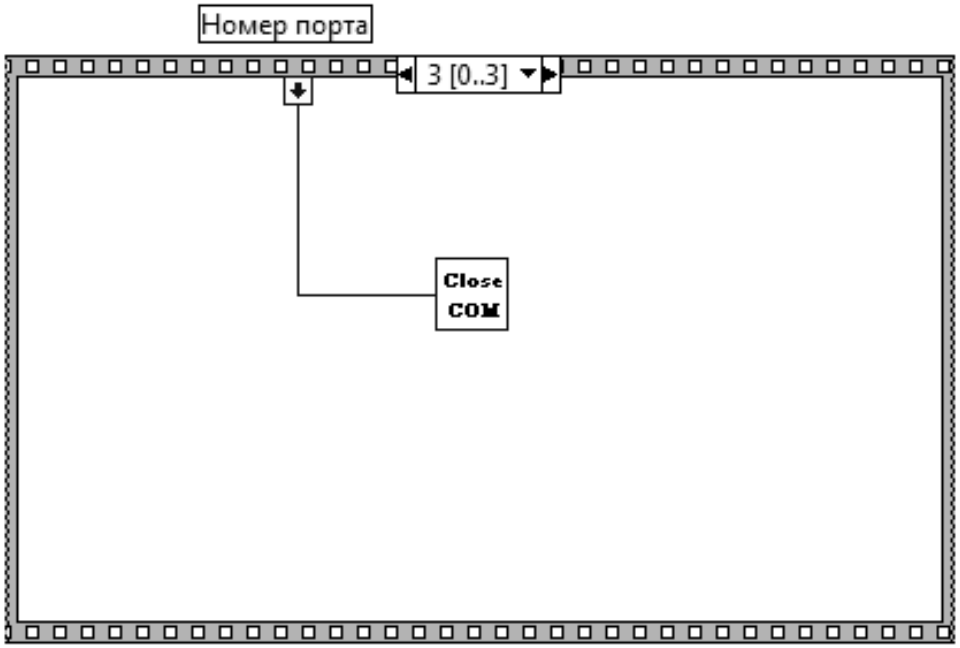


Рис. 27. Последняя страница структуры
«последовательность»

На последней вкладке структуры «последовательность» располагается субвиртуальный прибор, закрывающий COM-порт.

Для проверки работы виртуального прибора необходимо задать конфигурацию порта, команду модуля, установить время таймаута и указать необходимость проверки контрольной суммы, после чего за-

пустить виртуальный прибор. Полученное ответное сообщение модуля должно быть выведено в одноименном поле.

4.2. Управление в Lazarus

Для создания пользовательского приложения, осуществляющего управление модулями серии I-7000 в среде Lazarus, необходимо выполнить следующую последовательность:

1. Создать проект приложения. Запустите интегрированную среду разработки Lazarus из меню Windows «Пуск/Программы/Lazarus» (рис. 28).

Рис. 28. Меню «Пуск/Программы/Lazarus»

Выбрав пункт меню «Файл/Создать» (рис. 29), создайте новый проект. Для этого в древовидной структуре открывшегося диалогового окна «Создать» укажите раздел «Project/Application» (рис. 30) и нажмите кнопку «ОК», после чего будет создан новый проект, который необходимо сохранить под названием LazarusI7000 (рис. 31).

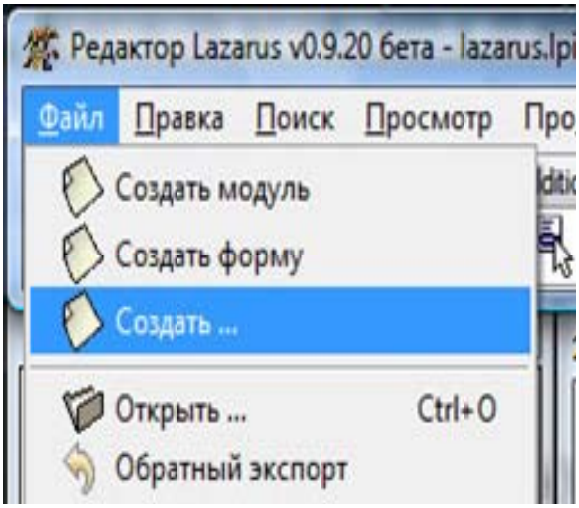


Рис. 29. Пункт меню Lazarus, создание нового проекта

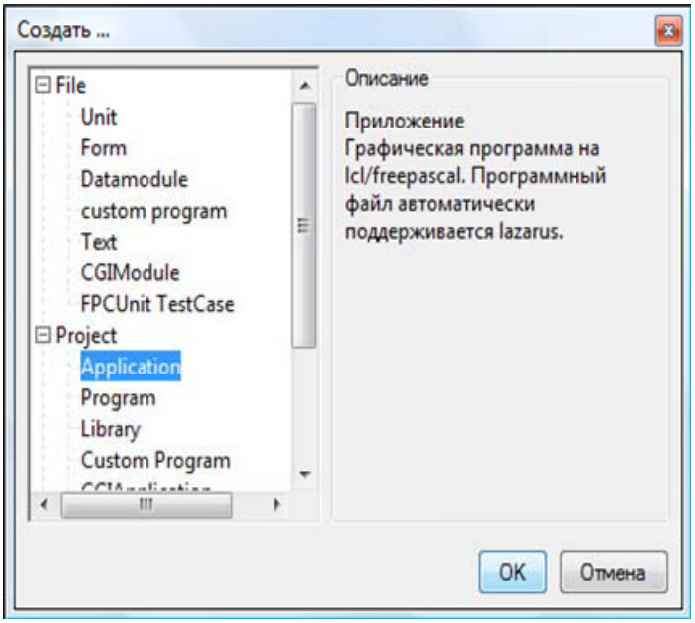


Рис. 30. Окно «Создать»

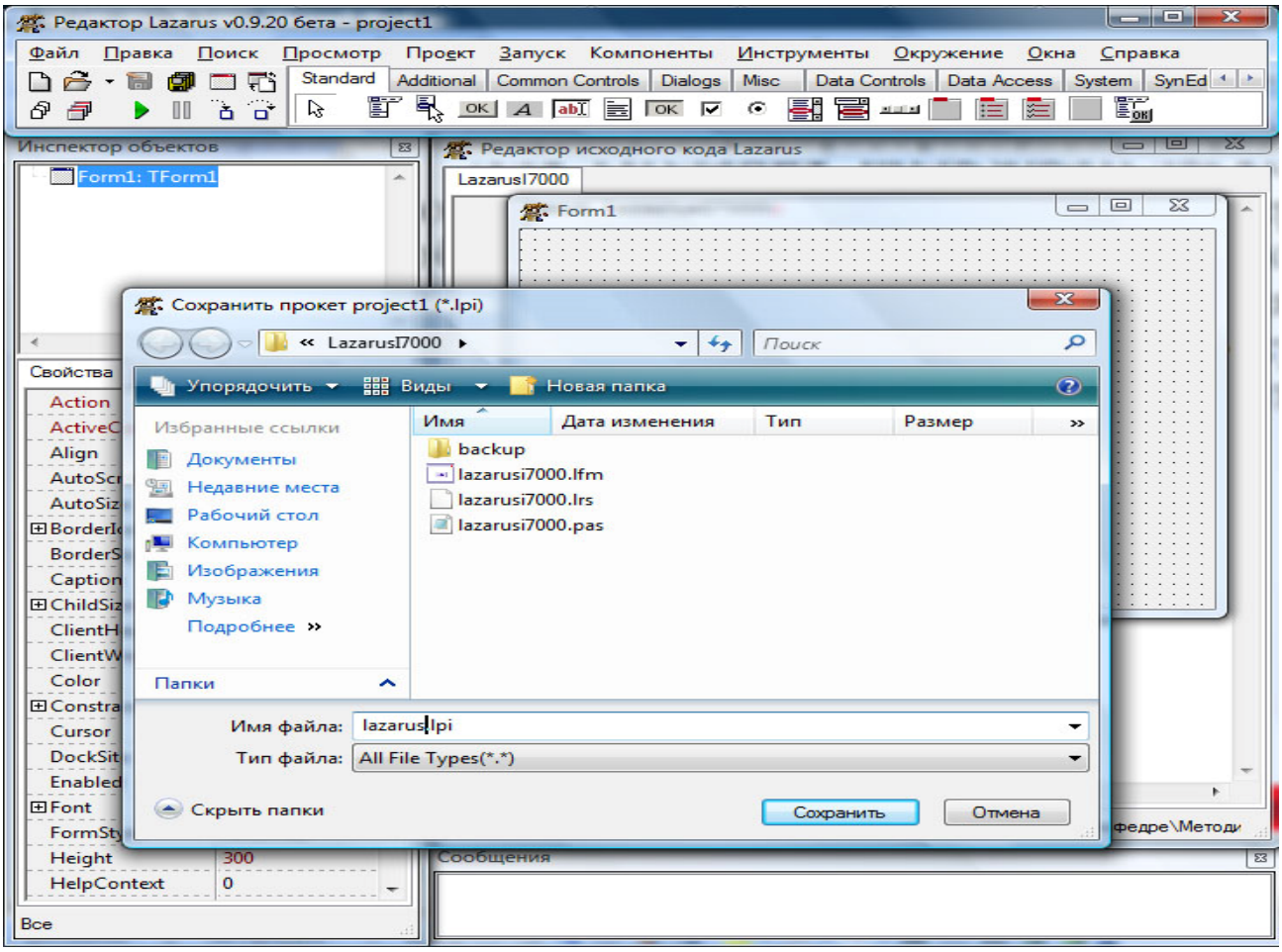


Рис. 31. Сохранение нового проекта в Lazarus

2. Измените внешний вид экранной формы приложения (Form1), описанной в модуле «LazarusI7000.pas», так, как показано на рис. 32. Присвойте новые названия компонентам, располагающимся на форме: «COM-порт» - editNumPort; «Скорость обмена» - editSpeed; «Команда» - editCommand; «Ответ» - editAnswer; «Открыть порт» - buttonOpen; «Записать в порт» - buttonWrite; «Закрыть порт» - buttonClose; «История выполненных команд» - memoHistory.

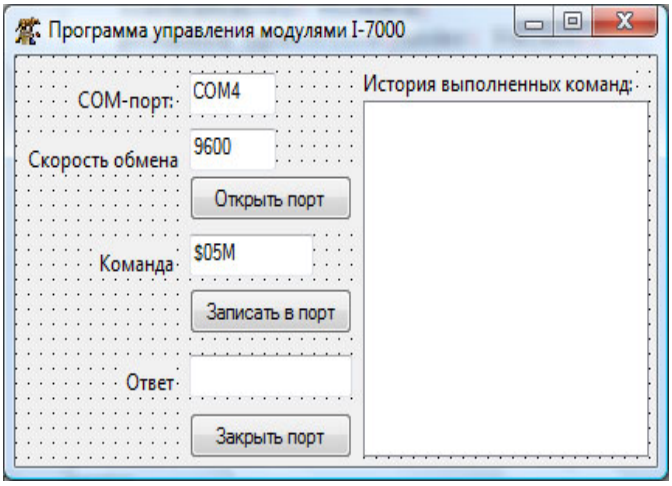


Рис. 32. Экранная форма приложения

3. **Скопируйте файлы synaser.pas**, **synafpc.pas**, **synau-til.pas в каталог вашего проекта**. Подключите к проекту модули Synaser.pas и ExtCtrl.pas. Для этого добавьте в раздел Units модуля «LazarusI7000.pas» (рис. 33).

```
uses
  Classes, SysUtils, LResources,
  Buttons, Synaser, ExtCtrls;
```

Рис. 33. Подключение модулей к проекту

4. Создайте класс, унаследованный от TThread, который будет реализовывать параллельный поток (нить) опроса COM-порта и отображения ответа модуля в полях «Ответ» и «История выполненных команд» (рис. 34).

Новый тип класса называется TReadThread и имеет ряд свойств: m_buffer – буфер-накопитель символов ответа модуля; m_countReceive – число полученных от модуля символов; m_receive – буфер чтения ответа модулей. Также в классе реализованы два метода: Execute – алгоритм опроса модулей; Showstatus – используется для отображения результатов на экранной форме программы.

```
type
  {Threads }
    TReadThread = class(TThread)
    private
      m_buffer: string;
      m_countReceive: integer;
      m_receive: array[0..100] of byte; // буфер для чтения ответа модулей
    procedure Showstatus(); // отобразить ответ модуля
    procedure Execute; override; // алгоритм параллельного потока
    end;
```

Рис. 34. Класс TReadThread, унаследованный от TThread

Следует напомнить, что вывод информации на элементы интерфейса программы из параллельного потока должен производиться в функциях, вызываемых как параметр Synchronize. Далее приводится код, реализующий методы класса TReadThread:

```
{ TReadThread }
procedure TReadThread.Showstatus();
var
  i: Integer;
begin
  // отобразить полученные данные
  for i := 0 to m_countReceive-1 do
  begin
    if m_receive[i] = $0D then
```

```
    begin
      MainForm.Receive.Text := m_buffer;
      MainForm.History.Lines.Add(MainForm.Receive.Text);
      m_buffer := '';
    end
    else
      m_buffer := m_buffer+char(m_receive[i]);
    end;
end;
procedure TReadThread.Execute;
begin
  repeat
    // определить число полученных байт
    m_countReceive := SerialPort.WaitingDataEx();
    if m_countReceive > 0 then
    begin
    // через COM-порт получена информация
      // прочитать полученные данные
      SerialPort.RecvBuffer(@m_receive, m_countReceive);
      // отобразить данные на интерфейсе пользователя
      // ВНИМАНИЕ !!!
      // Вывод на элементы интерфейса должен осуществлять-
ся в
      // процедуре, вызываемой как параметр Synchronize!!!
      Synchronize(@Showstatus);
    end;
  until Terminated;
end;
```

5. Объявите переменные COM-порта (SerialPort) типа TBlockSerial и параллельного потока для асинхронного чтения ответа модулей (ReadThread) типа TReadThread (рис. 35).

```
var
  Form1: TForm1;
  SerialPort: TBlockSerial; // объект COM-порта
  ReadThread: TReadThread; // объект параллельного потока
```

Рис. 35. Объявление переменных COM-порта и параллельного потока

6. Создайте конструктор формы. В конструкторе очищается поле «Ответ», а также история выполненных команд (поле «История выполненных команд») (рис. 36).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  // подготовить программу к работе
  memoHistory.Lines.Clear; // история выполнения команд
  editAnswer.Text := ''; // ответ модуля
end;
```

Рис. 36. Конструктор экранной формы приложения

7. Создайте обработчик нажатия кнопки «Открыть порт» (рис. 37).

```
procedure TForm1.buttonOpenClick(Sender: TObject);
var
  Speed: Integer;
begin
  // очистить историю и ответ модуля
  History.Lines.Clear;
```

Рис. 37. Обработчик нажатия кнопки «Открыть порт»

Ниже приводится полный текст обработчика кнопки «Открыть порт»:

```
procedure TForm1.buttonOpenClick(Sender: TObject);
var
  Speed: Integer;
begin
  // очистить историю и ответ модуля
  History.Lines.Clear;
  Receive.Text := '';
  Speed := StrToInt(PortSpeed.Text);
  // создать объект COM-порта
  SerialPort:=TBlockserial.Create;
  SerialPort.RaiseExcept:=true;
  // подключиться к заданному порту
  SerialPort.Connect(PortNumber.Text);
  // проверить ошибки подключения и вывести сообщения о
  // них в историю
  History.Lines.Add(IntToStr(SerialPort.LastError)+'-
'+SerialPort.LastErrorDesc);
  if SerialPort.LastError<>0 then
  begin
  // завершить попытку подключения, т.к. произошла
ошибка
    Exit;
```

```
    end;
    // конфигурирование порта (задается скорость, число бит,
    // паритет, старт/стоп биты ...)
    SerialPort.EnableRTSToggle(true);
    SerialPort.Config(Speed, 8,'N',0,false,false);
    // сообщение в историю выполненных команд
    History.Lines.Add('Порт '+SerialPort.Device+' - открыт.');

    ReadThread:= TReadThread.Create(true);
    ReadThread.Priority:=tpLowest;
    ReadThread.m_buffer := '';
    // запустить параллельный поток
    ReadThread.Resume;
  end;
```

8. Создайте обработчик нажатия кнопки «Записать в порт» (рис. 38) для записи команды в модуль семейства I-7000 через COM-порт и получения ответного сообщения в параллельном потоке.

```
procedure TForm1.buttonWriteClick(Sender: TObject);
begin
  // записать в COM-порт команду
  SerialPort.SendString(MainForm.Command.Text+chr($0D));
end;
```

Рис. 38. Обработчик нажатия кнопки «Записать в порт»

9. Создайте обработчик нажатия кнопки «Закрыть порт» (рис. 39).

```
procedure TForm1.buttonCloseClick(Sender: TObject);
begin
  // остановить параллельный поток
  ReadThread.Suspend;
  // освободить COM-порт
  SerialPort.Free;
  //
  History.Lines.Add('Порт закрыт.');
end;
```

Рис. 39. Обработчик нажатия кнопки «Закрыть порт»

После создания всех обработчиков получим класс приложения TForm1, представленный на рис. 40.

```
  TForm1 = class(TForm)
    buttonOpen: TButton;
    buttonWrite: TButton;
    buttonClose: TButton;
    editNumPort: TEdit;
    editSpeed: TEdit;
    editCommand: TEdit;
    editAnswer: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    memoHistory: TMemo;
    procedure buttonCloseClick(Sender: TObject);
    procedure buttonOpenClick(Sender: TObject);
    procedure buttonWriteClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;
  SerialPort: TBlockSerial; // объект COM-порта
```

Рис. 40. Класс TForm1

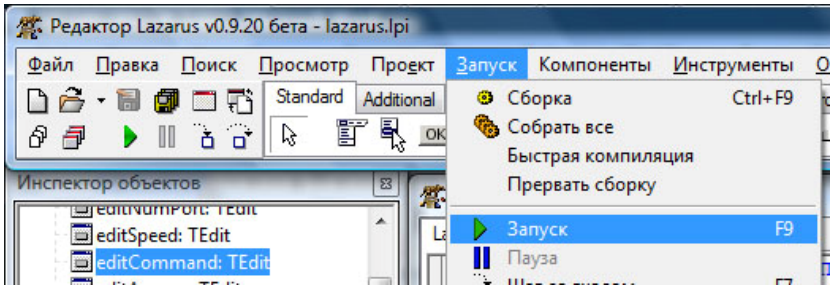


Рис. 41. Запуск приложения

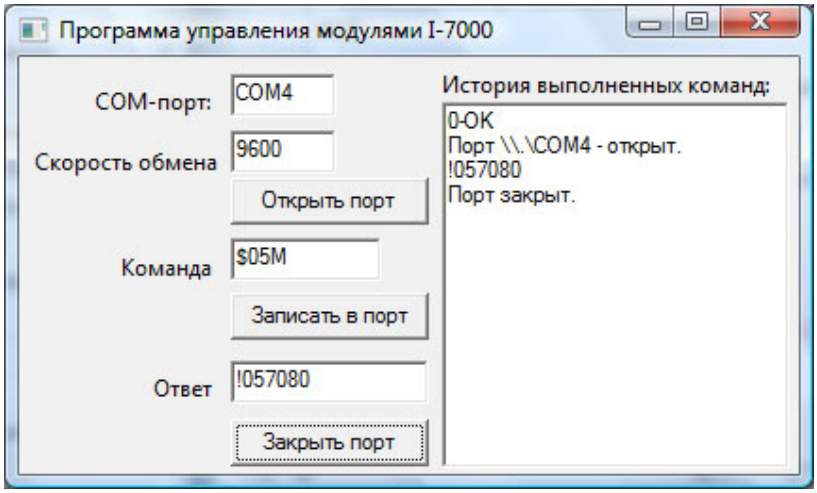


Рис. 42. Проверка работоспособности программы

10. Приложение, позволяющее управлять модулями, создано. Для его компиляции и запуска необходимо выбрать пункт меню «Запуск/Запуск F9» или нажать горячую клавишу **F9** (рис. 41).

Проверим работоспособность программы для устройства **I-7080,** имеющего адрес **05** и подключеного к **COM-порту №4,** работающему на скорости **9600 бод/с** . После запуска приложения необходимо произвести настройку порта **«COM-порт» - COM4**,

задать **«Скорость обмена» - 9600**, указать команду чтения названия модуля **«Команда» - $05M**. Затем необходимо последовательно нажать кнопки: «Открыть порт», «Записать в порт», «Закрыть порт». Результат взаимодействия программы с модулем можно оценить по истории выполненных команд, отображаемой в одноименном списке (рис. 42).

4.3. Управление в Borland Delphi

Для создания пользовательского приложения, осуществляющего управление модулями серии I-7000, в среде Borland Delphi необходимо выполнить следующую последовательность действий:

1. Создать проект приложения. Запустить интегрированную среду разработки Borland Delphi из меню Windows «Пуск/Программы/Borland Delphi 7/Delphi 7» (рис. 43). Выбрав пункт меню «File/New/Application» (рис. 44), создайте новый проект и сохраните его под названием DelphiI7000 (рис. 43). Для сохранения файла используйте пункт меню «File/Save Project As…».

2. Измените внешний вид экранной формы приложения (Form1), описанной в модуле «Unit1.pas», так, как показано на рис. 46. Присвойте новые названия компонентам, располагающимся на форме: «COM-порт №» - comboBoxNumPort;

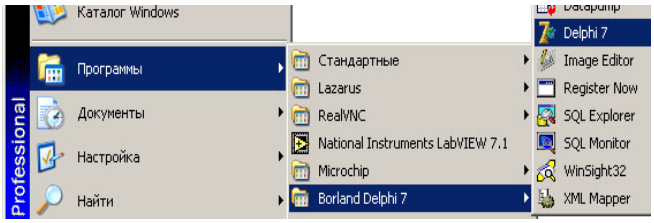


Рис. 43. Меню «Пуск/Программы/ Borland Delphi 7/Delphi 7»

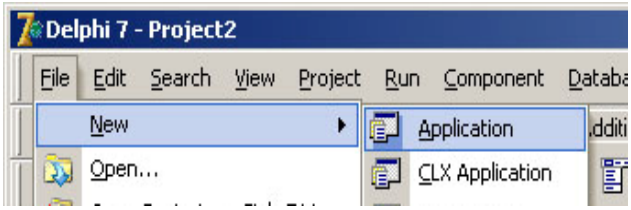


Рис. 44. Пункт меню Delphi, создание нового проекта

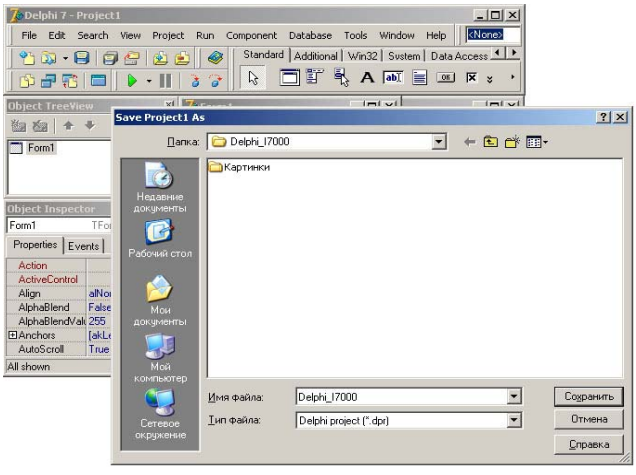


Рис. 45. Сохранение нового проекта в Delphi

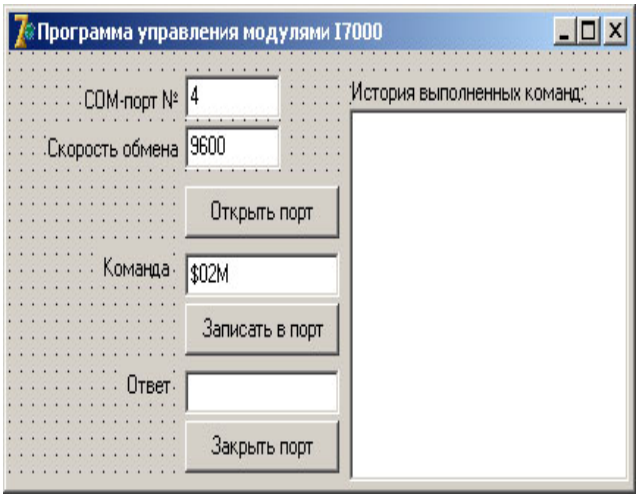


Рис. 46. Экранная форма приложения

«Скорость обмена» - editSpeed; «Команда» - editCommand; «Ответ» - editAnswer; «Открыть порт» - buttonOpen; «Записать в порт» - buttonWrite; «Закрыть порт» - buttonClose; «История выполненных команд» - memoHistory.

```
uses
  Windows, Messages, SysUtils, Variar
  Dialogs, StdCtrls, ComPort;
```

Рис. 47. Подключение модуля к проекту

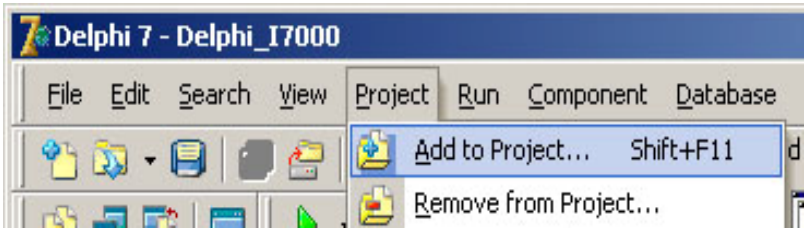


Рис. 48. Добавление модуля в проект

3. **Скопируйте файл comport.pas в каталог вашего проекта**. Подключите к проекту модуль **comport.pas**. Для этого добавьте описание в раздел Units модуля «Unit1.pas» (рис. 47). **Добавьте модуль comport.pas в проект, выбрав пункт меню «Project/Add to Project…»,** или нажмите клавиши **Shift+F11** (рис. 48). В появившемся окне «Add to Project» выделите файл comport.pas и нажмите кнопку «Открыть».

```
private
  { Private declarations }
  Port: TComPort; // Объект COM-порта
public
```

Рис. 49. Объявление переменной COM-порта

4. Объявите переменную COM-порта (Port), типа TComPort (рис. 49).

```
procedure TForm1.FormCreate(Sender: TObject);
var
  i : integer;
begin
  // подготовить программу к работе
  // очистить историю выполенения команд
  memoHistory.Lines.Clear;
  // очистить ответ модуля
  editAnswer.Text := '';
  // сформировать список COM-портов
  comboBoxNumPort.Items.Clear;
  for i := 1 to 256 do begin
    comboBoxNumPort.Items.Add('COM'+IntToStr(i));
  end;
  comboBoxNumPort.ItemIndex := 0;
end;
```

Рис. 50. Конструктор экранной формы приложения

5. Создайте конструктор формы. В конструкторе очищается поле «Ответ», история выполненных команд (поле «История выполненных команд»), а также формируется список COM-портов (поле «COM-порт №») (рис. 50).

В примере список портов формируется в широком диапазоне от COM1 до COM256. Однако для того, чтобы сформировать список из названий портов, существующих в данной конфигурации системы, конструктор необходимо отредактировать, а также подключить модуль, позволяющий работать с реестром Windows – Registry. Конструктор формы приложения, в котором определяется список существующих в данной конфигурации системы COM-портов, приведен на рис. 51.

1. Создайте обработчик нажатия кнопки «Открыть порт» (рис. 52). В обработчике создается экземпляр объекта, позволяющего взаимодействовать с COM-портом.

Свойству OnRead созданного объекта присваивается процедура, которая будет вызываться каждый раз по завершению команды чтения данных из COM-порта (пересылки данных из модуля в компьютер). В качестве одного из параметров процедуре передается байтовый буфер с ответом модуля (ReadBytes). Содержимое буфера сканируется до появления символа окончания строки $0D, а его содержимое преобразуется в последовательность ASCII кодов. Результат преобразования отображается в интерфейсе пользователя. Текст процедуры показан на рис. 53.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  reg : TRegistry; // переменная для работы с реестром
  ts : TStrings; // содержимое переменной реестра
  i : integer;
begin
  reg := TRegistry.Create;
  reg.RootKey := HKEY_LOCAL_MACHINE;
  reg.OpenKey('hardware\devicemap\serialcomm', false);
  ts := TStringList.Create;
  reg.GetValueNames(ts);
  comboBoxNumPort.Items.Clear;
  for i := 0 to ts.Count -1 do begin
    comboBoxNumPort.Items.Add(reg.ReadString(ts.Strings[i]));
  end;
  comboBoxNumPort.ItemIndex := 0;
  ts.Free;
  reg.CloseKey;
  reg.free;
end;
```

Рис. 51. Конструктор экранной формы приложения с определением списка существующих COM-портов

7. Создайте обработчик нажатия кнопки «Записать в порт» (рис. 54), по которой команда через COM-порт записывается в модуль и получает ответное сообщение.

```
procedure TForm1.buttonOpenClick(Sender: TObject);
begin
  // создать экземпляр объекта COM-порт
  // параметры: 1-номер COM-порта;
  // 2-скорость обмена (в данном случае скорость 9600)
  Port := TComPort.Create(comboBoxNumPort.ItemIndex+1, br9600);
  // назначение функции-обработчика чтения данных
  Port.OnRead := OnRead;
  memoHistory.Lines.Add('Порт открыт.');
end;
```

Рис. 52. Обработчик нажатия кнопки «Открыть порт»

```
procedure TForm1.OnRead(
  Sender: TObject;
  ReadBytes: array of Byte); // полученная поледовательность
var
  i: Integer;
  TmpStr : string;
begin
  TmpStr := '';
  i := Low(ReadBytes);
  while (i < High(ReadBytes)) and (ReadBytes[i] <> $0D) do
  begin
    TmpStr := TmpStr + chr(ReadBytes[i]);
    inc(i);
  end;
  editAnswer.Text := TmpStr;
  memoHistory.Lines.Add(TmpStr);
end;
```

Рис. 53. Листинг процедуры OnRead

```
procedure TForm1.buttonWriteClick(Sender: TObject);
var
  strWrite: string; // временная переменная ASCII последовательности
  arrBytes: array of Byte; // массиб байт команды
  i: Integer;
begin
  // чтение команды из поля "Команда"
  strWrite := editCommand.Text;
  // преобразование последовательности ASCII в массив байт
  SetLength(arrBytes, Length(strWrite)+1);
  for i := Low(arrBytes) to High(arrBytes) do
    arrBytes[i] := Ord(strWrite[i + 1]);
  // формирование признака конца команды
  arrBytes[High(arrBytes)]:=$0D;
  // запись сформированной команды в COM-порт
  Port.Write(arrBytes);
  arrBytes := nil;
end;
```

Рис. 54. Обработчик нажатия кнопки «Записать в порт»

8. Создайте обработчик нажатия кнопки «Закрыть порт» (рис. 55).

```
procedure TForm1.buttonCloseClick(Sender: TObject);
begin
  // закрыть COM-порт
  Port.Free;
  memoHistory.Lines.Add('Порт закрыт.');
end;
```

Рис. 44. Обработчик нажатия кнопки «Закрыть порт»

После создания всех обработчиков получим класс приложения TForm1, представленный на рис. 56.

10. Приложение, позволяющее управлять модулями, создано. Для его компиляции и запуска необходимо выбрать пункт меню «Run/Run F9» или нажать горячую клавишу **F9** (рис. 57). Проверим работоспособность программы для устройства **I-7080**, имеющего адрес **02 и** подключенного к **COM-порту №4,** настроенного на скорость **9600 бит/с**.

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Dialogs, StdCtrls, ComPort, Registry;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    editCommand: TEdit;
    editAnswer: TEdit;
    buttonOpen: TButton;
    buttonWrite: TButton;
    buttonClose: TButton;
    editSpeed: TEdit;
    Label5: TLabel;
    memoHistory: TMemo;
    comboBoxNumPort: TComboBox;
    procedure OnRead(Sender: TObject; ReadBytes: array of Byte);
    procedure FormCreate(Sender: TObject);
    procedure buttonOpenClick(Sender: TObject);
    procedure buttonWriteClick(Sender: TObject);
    procedure buttonCloseClick(Sender: TObject);
  private
    { Private declarations }
    Port: TComPort; // Объект COM-порта
  public
    { Public declarations }
  end;
```

Рис. 56. Класс TForm1

После запуска приложения необходимо произвести настройку порта **«COM-порт» - COM4**, задать **«Скорость обмена» - 9600**, указать команду чтения названия модуля **«Команда» - $02M**. После проведенных настроек необходимо последовательно нажать кнопки: «Открыть порт», «Записать в порт», «Закрыть порт». Результат взаимодействия программы с модулем можно оценить по истории выполненных команд, отображаемой в одноименном списке (рис. 58).

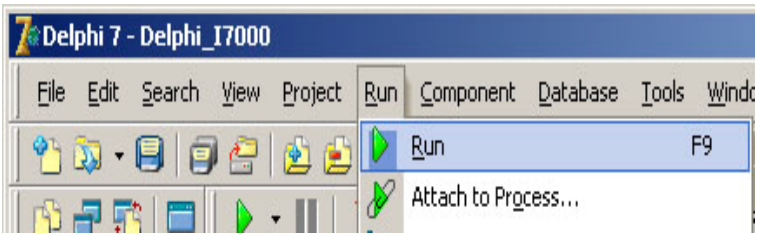


Рис. 57. Запуск приложения

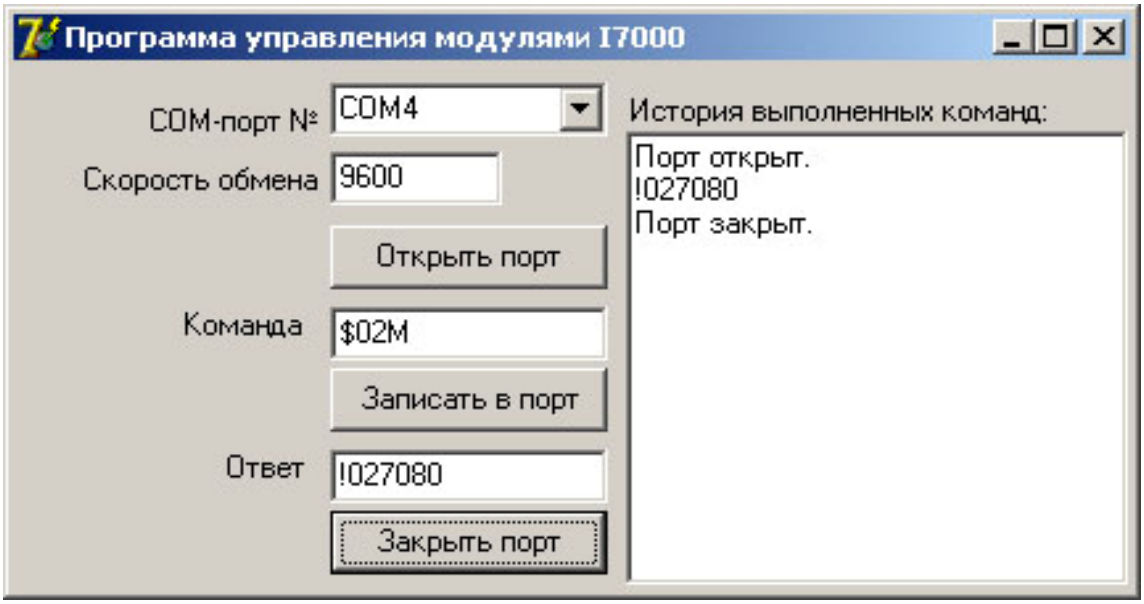


Рис. 58. Проверка работоспособности программы

4.4. Управление в Visual C++

В данном разделе приводится описание программы управления модулями в среде Visual C++ на основе библиотеки SerialGate.dll.

SerialGate.dll - динамически подключаемая библиотека, содержащая функции для работы с COM портами на языке C++. В основе библиотеки лежит класс SerialGate, используя который, можно выполнять наиболее часто востребованные действия управления COM портом: прием/передача данных, управление линиями взаимодействия, определение доступных портов в системе и т.д. Ниже приводится описание функций класса SerialGate.

bool Open(int port, int baud); - функция открывает доступ к COM порту с номером **port** на скорости **baud** бит/с. Если указанный порт существует и не занят другим приложением в данный момент,

функция вернет true, иначе false. Если, например, параметр port был указан как 3, то функция попытается открыть доступ к COM порту с именем COM3.

Пример использования:

```
SerialGate sg;
bool b = sg.Open(1, 9600);
if(b == true)
{
  //порт открыт успешно
}
else
{
  //порт открыт с ошибкой
}
```

int Send(char* buff, int szBuff); - функция записывает в ранее открытый порт szBuff байт данных из буфера buff. Возвращает число успешно записанных байт данных в порт.

Пример использования:

```
char buff[256];
for(int i=0; i < sizeof(buff); i++)
{
  buff[i] = i;
}
int SendCounter = sg.Send(buff, sizeof(buff));

if(SendCounter != sizeof(buff))
{
  //не все данные были записаны в порт
}
```

int Recv(char* buff, int szBuff); - читает из ранее открытого порта szBuff байт данных и помещает их в буфер buff. Возвращает число реально прочитанных байт данных.

Пример использования:

```
char buff[256];
int RcvCounter = sg.Recv(buff, sizeof(buff));
if(RcvCounter != sizeof(buff))
{
  //прочли меньше чем заказывали
```

}

void SetLine(OUT_LINES_NAME ln, bool state); - функция устанавливает одну из выходных сигнальных линий (DTR или RTS) в логическую единицу или ноль. Имя линии задается через перечисление OUT_LINES_NAME. Вторым параметром передается состояние (true – 1, false - 0), в которое необходимо перевести линию.

Пример использования:

```
// установит на линии RTS лог. 1
sg.SetLine(sg.RTS, true);
```

bool GetLine(IN_LINES_NAME ln); - функция возвращает состояние одной из входных сигнальных линий (CTS, DSR, RING или RLSD). Имя линии задается через перечисление IN_LINES_NAME.

Пример использования:

```
// чтение состояния линии DSR
boll b = sg.GetLine(sg.DSR);
```

void GetPortsInfo(PortInfo* pi); - функция заполняет переданную ей структуру PortInfo информацией об установленных в системе COM портах.

Пример использования:

```
#include <iostream>
#include <conio.h>
#include <windows.h>
#include "SerialGate.h"
void main()
{
  printf("Определить все COM-порты системы:\n");
  SerialGate sg;
  PortInfo pi;
  sg.GetPortsInfo(&pi);
  printf("Общее количество COM-портов: %d\n\n", pi.koll);
  for(int i = 0; i < pi.koll; i++)
  {
        if(pi.p[i].Availbl == true)
              printf("COM%d - свободен\n", pi.p[i].Id);
        else
              printf("COM%d - недоступен\n", pi.p[i].Id);
  }
  getch();
}
```

void Clean(); - очищает входной и выходной буфер данных COM порта.

Пример использования:

sg.Clean();

void Close(); - закрывает ранее установленное соединение с COM портом.

Пример использования:

sg.Close();

Для создания пользовательского приложения, осуществляющего управление модулями серии I-7000 в среде MS Visual C++ 6.0 на основе MFC необходимо выполнить следующую последовательность действий:

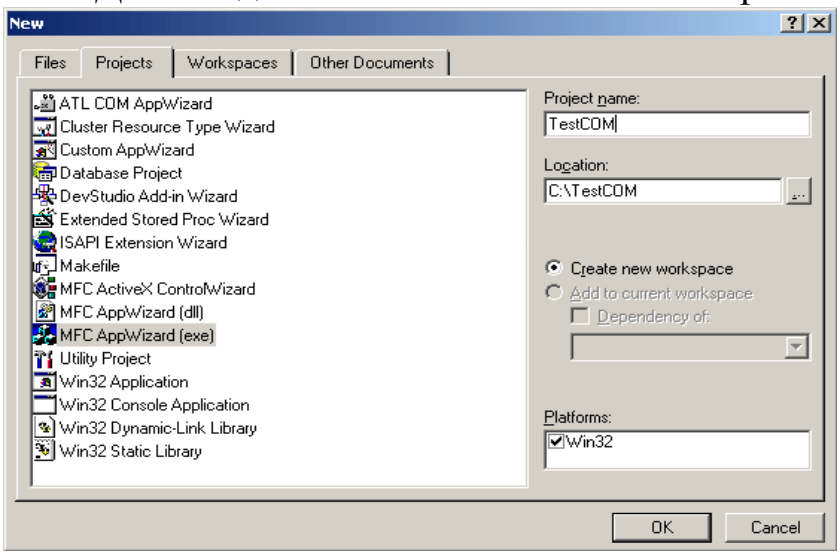


Рис. 59. Новый проект на основе диалогового окна

1. Создать новый проект MFC и указать его имя, TestCom (рис. 59). Выбрать вариант построения на основе диалогового окна.

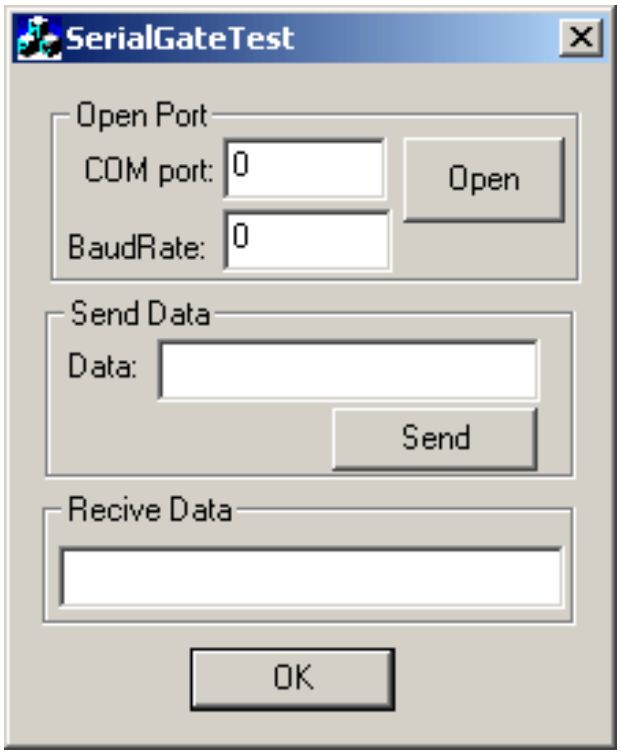


Рис. 60. Экранная форма приложения

2. Изменить внешний вид экранной формы приложения так, как показано на рис. 60.

С помощью этой экранной формы можно задать номер порта и скорость, на которой он будет работать. COM-порт открывается и настраивается нажатием кнопки «Open». Кнопка «Send» предназначена для отправки содержимого поля «Data» в COM-порт. В программе организован таймер, срабатывающий каждую секунду. В обработчике таймера производится чтение

информации из порта. Если данные готовы, то они помещаются в поле «Recive Data».

3. В заголовочном файле проекта (например, SerialGateTest.h) необходимо описать класс **sg** типа **SerialGate**.

```
class CSerialGateTestDlg : public CDialog
{
// Construction
public:
    // standard constructor
    CSerialGateTestDlg(CWnd* pParent = NULL);
    SerialGate sg;
```

4. Создать обработчик нажатия кнопки «Open». В обработчике производится проверка правильности введенных данных и попытка открыть порт методом **SerialGate.Open()**. Если порт открыть не удалось, выводится предупреждающее сообщение. После чего запускается таймер с идентификатором 1 и временем срабатывания 1000 мс.

```
void CSerialGateTestDlg::OnOpen()
    {
      UpdateData(true);
      if(m_port == 0 || m_rate == 0)
      {
            MessageBox("Данные введены неправильно.","Error",
MB_ICONERROR);
            return;;
      }
      bool b = sg.Open(m_port, m_rate);
      if(b == false)
      {
            MessageBox("Не могу открыть порт","Error",
MB_ICONERROR);
            return;
      }
      else
      {
            MessageBox("Порт открыт.","Info",
MB_ICONINFORMATION);
```

```
  }
  SetTimer(1, 1000, NULL);
}
```

5. Создать обработчик таймера. В обработчике производится попытка чтения данных из порта с помощью функции Recv(). Если данные были прочитаны (их число в байтах > 0), помещаем результат в поле «Recive Data».

```
void CSerialGateTestDlg::OnTimer(UINT nIDEvent)
{
  char buff[128];

  int rcv = sg.Recv(buff, sizeof(buff));
  if(rcv > 0)
  {
        for(int i = 0; i< rcv; i++)
              this->m_recieve += buff[i];
        UpdateData(false);
  }
  CDialog::OnTimer(nIDEvent);
}
```

6. Создать обработчик кнопки «Send». Вначале проверяется на ноль длина посылаемого сообщения. Если строка не пуста, то получается адрес на буфер типа Char, содержащий сообщение. Затем сообщение и длина сообщения передается в функцию Send(…), которая отправляет его в COM-порт.

```
void CSerialGateTestDlg::OnSend()
{
  UpdateData(true);
  int len = this->m_send.GetLength();
  if(len > 0)
  {
```

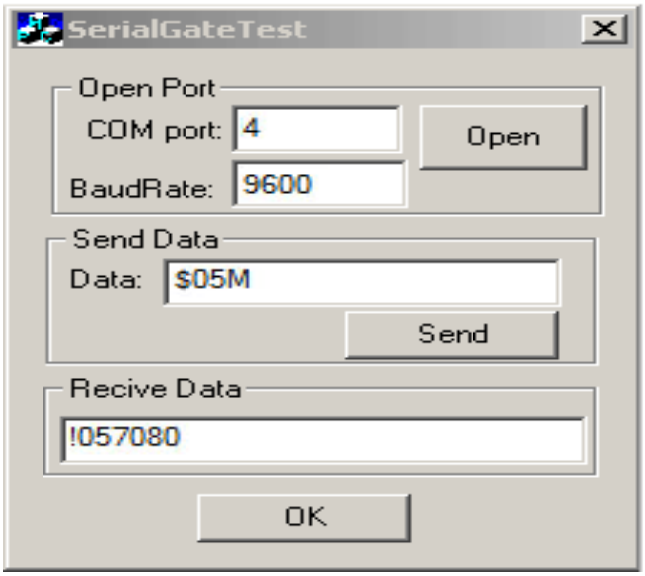


Рис. 61. Проверка работоспособности программы

```
            char* LocBuf =

                    sg.Send(LocBuf, len);

            m_send.ReleaseBuffer();
            }
}
```

Проверим работоспособность программы для случая, когда устройство **I-7080** имеет адрес **05** и подключено к **COM-порту №4,** работающему на скорости **9600 бит/с**. После запуска приложения необходимо произвести настройку порта **«COM port» - COM4**, задать **«BaudRate» - 9600**, указать команду чтения названия модуля **«Data» - $05M**. Затем необходимо последовательно нажать кнопки: «Open», «Send». Результат взаимодействия программы с модулем можно оценить по ответному сообщению поля Recive Data (рис. 61).

4.5. Управление в Visual C#

Для создания пользовательского приложения, осуществляющего управление модулями серии I-7000 в среде MS Visual C#, необходимо выполнить следующую последовательность действий:

1. Создать проект приложения. Запустите интегрированную среду разработки Microsoft Visual C# 2008 Express Edition (данную версию среды можно скачать из сети интернет с официального сайта компании Microsoft) из меню Windows «Пуск/Программы» (рис. 62).

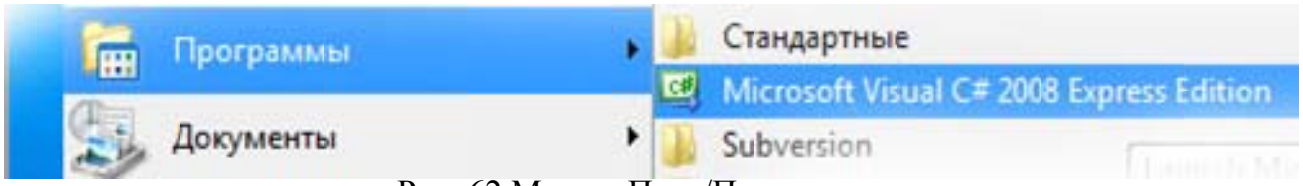


Рис. 62 Меню «Пуск/Программы»

Выбрав пункт меню «File/New Project…» (рис. 63), создайте новый проект. Для этого в появившемся окне «New Project» необходимо указать тип проекта - «Windows Forms Application» и задать его имя –

«CSharpI7000» (рис. 64). Нажмите кнопку «OK». После чего будет создан новый проект (рис. 65).

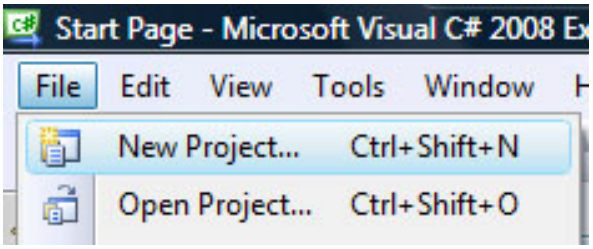


Рис. 63. Создание нового проекта в меню MS Visual C#

Сохраните проект, выбрав пункт «File/Save All» (рис. 66). В открывшемся окне «Save Project» укажите название проекта и его местоположение (рис. 67).

2. Измените внешний вид экранной формы приложения (Form1), описанной в модуле «Form1.cs» так, как показано на рис. 68.

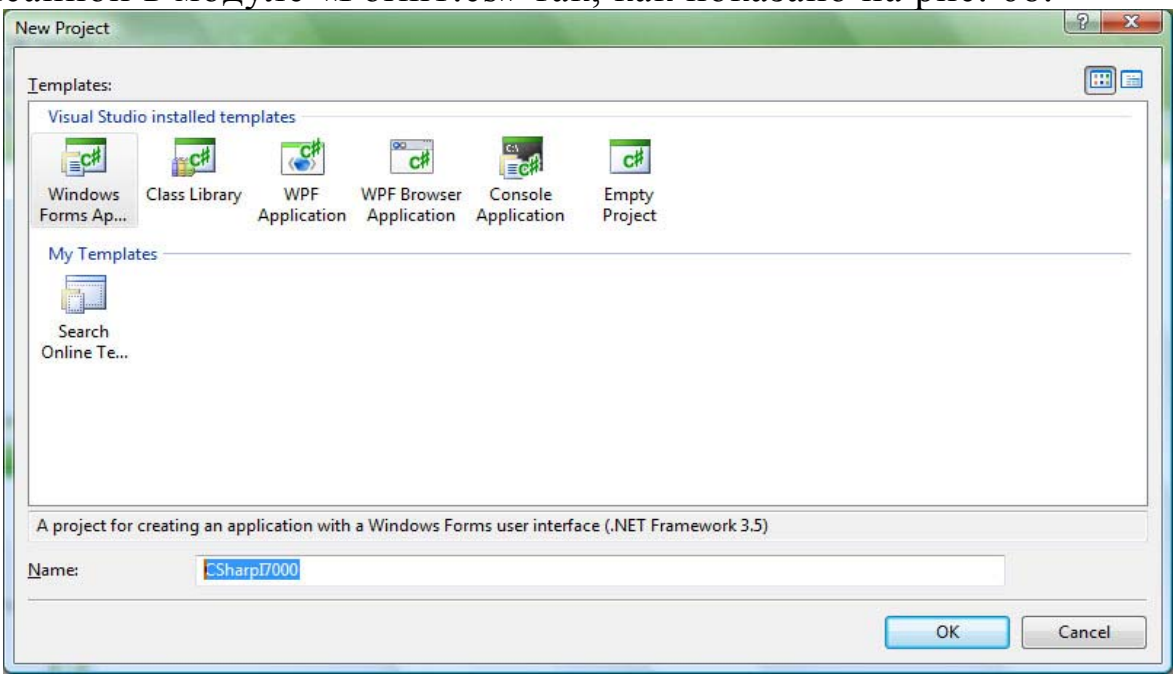


Рис. 64. Окно «New Project»

Присвойте новые названия компонентам: «COM-порт» - textBox-NumPort; «Скорость обмена» - textBoxSpeed; «Команда» - textBoxCommand; «Ответ» - textBoxAnswer; «Открыть порт» - buttonOpen; «Записать в порт» - buttonWrite; «Закрыть порт» - buttonClose; «История выполненных команд» - richTextBoxHistory.

3. Подключите пространство имен «System.IO.Ports», обеспечивающее доступ к COM-портам. Для этого в начало файла «Form1.cs» необходимо добавить строку «**using System.IO.Ports;**» (рис. 69).

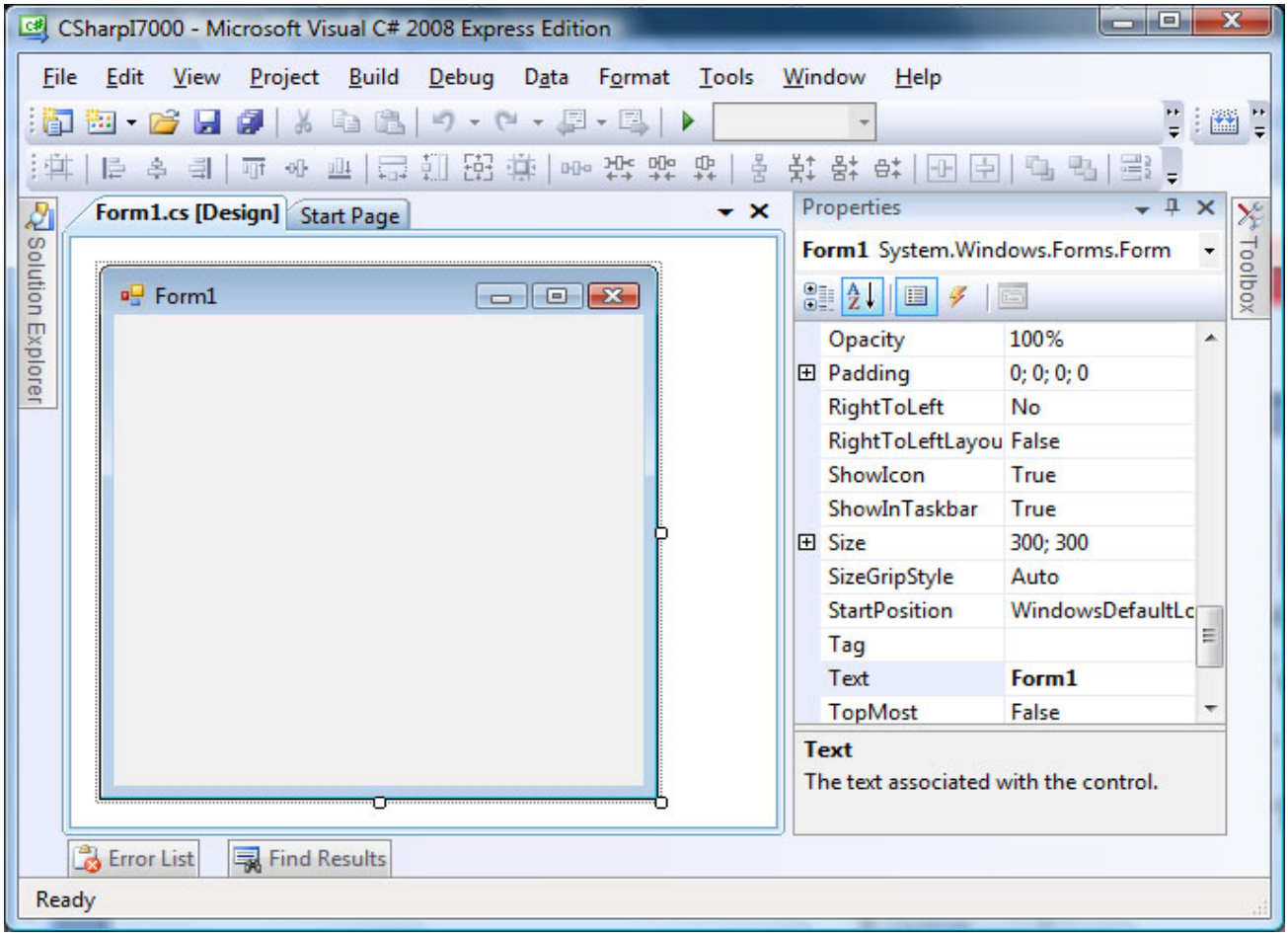


Рис. 65. Интегрированная среда разработки проектов на языке C#

Присвойте новые названия компонентам: «COM-порт» - textBoxNumPort; «Скорость обмена» - textBoxSpeed; «Команда» - textBoxCommand; «Ответ» - textBoxAnswer; «Открыть порт» - buttonOpen; «Записать в порт» - buttonWrite; «Закрыть порт» - buttonClose; «История выполненных команд» - richTextBoxHistory.

3. Подключите пространство имен «System.IO.Ports», обеспечивающее доступ к COM-портам. Для этого в начало файла «Form1.cs» необходимо добавить строку «**using System.IO.Ports;**» (рис. 69).

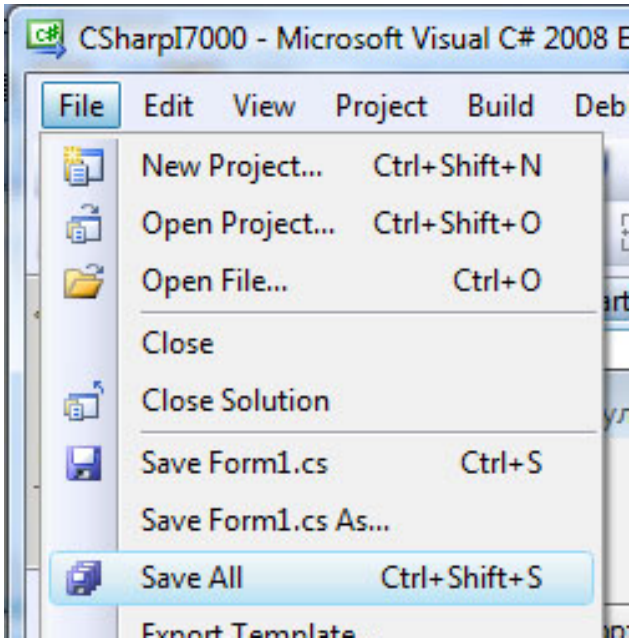


Рис. 66. Сохранение проекта в меню MS Visual C#.

3. Объявите создание экземпляра класса, обеспечивающего взаимодействие с COM-портом (comport). Тип экземпляра - SerialPort (рис. 92).

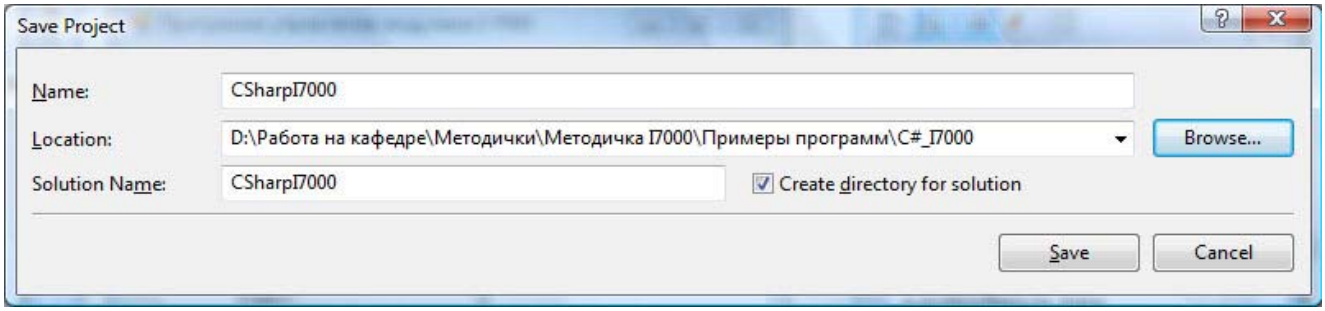


Рис. 67. Окно «Save Project»

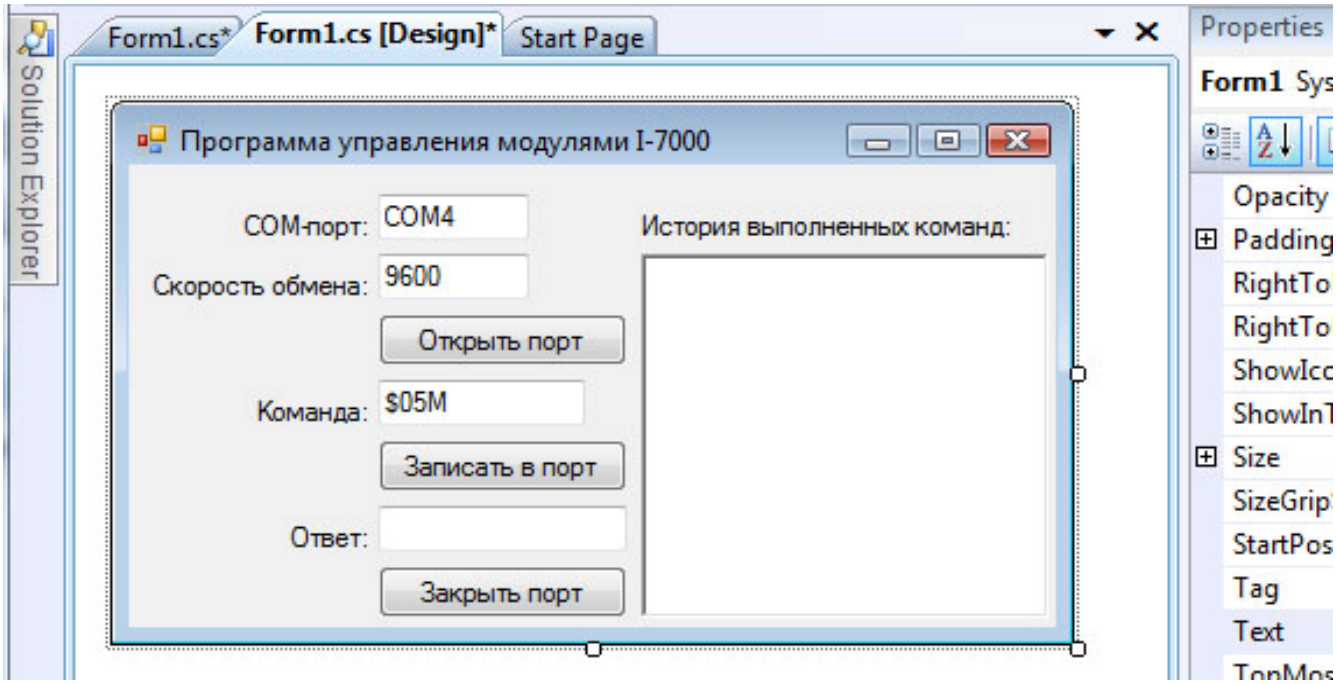


Рис. 68. Экранная форма приложения

```
// пространство имен для работы с COM-портом
using System.IO.Ports;
```

Рис. 69. Подключение пространства имен обеспечивающего доступ к COM-порту

```
// класс, обеспечивающий взаимодействие с COM-портом
private SerialPort comport = new SerialPort();
```

Рис. 70. Объявление переменных COM-порта

5. Укажите в классе Form1 метод добавления строки в историю выполненных команд:

```
private void AddHistoryMessage(string msg)
{/// добавить сообщение в историю выполненных команд
   richTextBoxHistory.Invoke(
      new EventHandler( delegate{
         // добавить текст в историю
```

```
            richTextBoxHistory.AppendText(msg);
            // в списке истории выделить последнюю запись
            richTextBoxHistory.ScrollToCaret();
        })
    );
}
```

6. Создайте обработчик нажатия кнопки «Открыть порт». Полный текст обработчика кнопки «Открыть порт» приводится ниже:

```
private void buttonOpen_Click(object sender, EventArgs e)
{
    // если порт был ранее открыт, закрыть его
    if (comport.IsOpen) comport.Close();
    else
    {// порт ранее открыт не был
        // название COM-порта
        comport.PortName = textBoxNumPort.Text;
        // скорость работы COM-порта
        int baudRate = 9600;
        int.TryParse(textBoxSpeed.Text, out baudRate);
        comport.BaudRate = baudRate;
        // число бит данных
        comport.DataBits = 8;
        // число стоповых бит - один
        comport.StopBits = StopBits.One;
        // бит паритета - нет
        comport.Parity = Parity.None;
        // квитировать установление связи - нет
        comport.Handshake = Handshake.None;
        // число принимаемых бит
        comport.ReceivedBytesThreshold = 8;
        // размер буфера для записи
        comport.WriteBufferSize = 20;
        // размер буфера для чтения
        comport.ReadBufferSize = 20;
        // время таймаута чтения - по умолчанию
        comport.ReadTimeout = -1;
        // время таймаута записи - по умолчанию
        comport.WriteTimeout = -1;
```

```
            // сигнал готовности терминала к передаче данных - не
            // установлен
            comport.DtrEnable = false;
            // открыть порт
            comport.Open();
            // запрос передатчика - установлен
            comport.RtsEnable = true;
            // задержка
            System.Threading.Thread.Sleep(100);
            AddHistoryMessage("Порт открыт. \n");
        }
    }
```

7. Создайте обработчик нажатия кнопки «Записать в порт»:

```
    private void buttonWrite_Click(object sender, EventArgs e)
    {
        // записать команду в COM-порт (символ окончания
команды –
        // 0x0D)
        comport.WriteLine(textBoxCommand.Text + (char)0x0D);
        // выдать сообщение в историю
        AddHistoryMessage("Записана команда:" +
textBoxCommand.Text + "\n");
    }
```

8. В конструкторе формы назначьте обработчик для события «данные COM-порта получены»:

```
    public Form1()
    {
        InitializeComponent();
        // назначить обработчик для события "данные COM-порта
        // получены"
        comport.DataReceived += new
SerialDataReceivedEventHandler( port_DataReceived);
    }
```

Обработчик **port_DataReceived** имеет вид:

```
    private void port_DataReceived(object sender,
SerialDataReceivedEventArgs e)
    {
        this.Invoke(new EventHandler(AddRecieve));
```

```
}
```

Листинг метода **AddRecieve** приводится ниже:

```
private void AddRecieve(object s, EventArgs e)
{
   // задержка
   System.Threading.Thread.Sleep(100);
   // буфер для чтения данных из COM-порта
   byte[] dataR = new byte[comport.BytesToRead];
   // прочитать данные
   comport.Read(dataR, 0, dataR.Length);
   // добавить ответ в историю команд
   AddHistoryMessage("Получен ответ:");
   for (int i = 0; i < dataR.Length; i += 1)
      AddHistoryMessage(((char)dataR[i]).ToString());
   AddHistoryMessage("\n");
   comport.DiscardInBuffer();
}
```

9. Создайте обработчик нажатия кнопки «Закрыть порт»:

```
private void buttonClose_Click(object sender, EventArgs e)
{
   // освободить выходной буфер
   comport.DiscardOutBuffer();
   // освободить входной буфер
   comport.DiscardInBuffer();
   // закрыть порт
   if (comport.IsOpen) comport.Close();
   AddHistoryMessage("Порт закрыт. \n");
}
```

После создания всех обработчиков получим класс приложения Form1, представленный на рис. 71.

10. Приложение, позволяющее управлять модулями, создано. Для его компиляции и запуска необходимо выбрать пункт меню «Debug/Start Debugging F5» или нажать горячую клавишу **F5** (см. рис. 72).

Проверим работоспособность программы для устройства **I-7080,** которое имеет адрес **05** и подключено к **COM-порту №4,** работающему на скорости **9600 бит/с**.

```
namespace CSharpI7000
{
    partial class Form1
    {
        /// <summary> ...
        private System.ComponentModel.IContainer components = null;

        /// <summary> ...
        protected override void Dispose(bool disposing)...

        Windows Form Designer generated code

        private System.Windows.Forms.Button buttonOpen;
        private System.Windows.Forms.Button buttonWrite;
        private System.Windows.Forms.Button buttonClose;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.RichTextBox richTextBoxHistory;
        private System.Windows.Forms.TextBox textBoxNumPort;
        private System.Windows.Forms.TextBox textBoxSpeed;
        private System.Windows.Forms.TextBox textBoxCommand;
        private System.Windows.Forms.TextBox textBoxAnswer;
    }
}
```

Рис. 71. Класс Form1

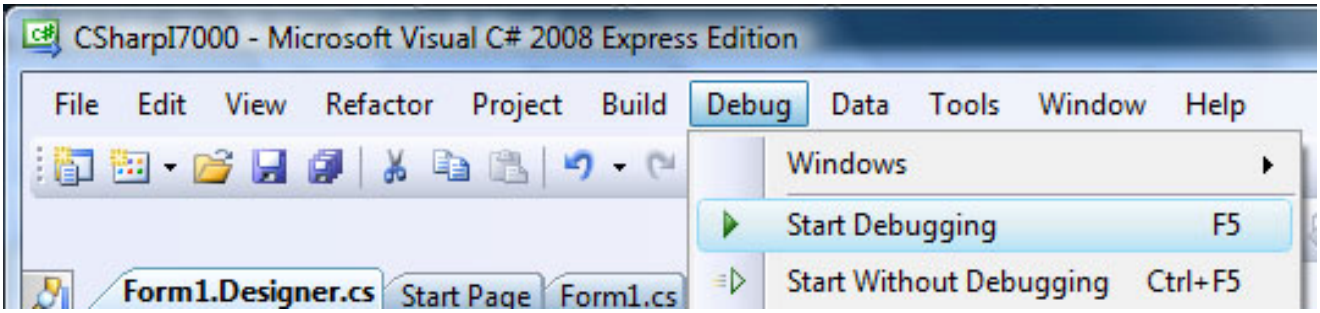


Рис. 72. Запуск приложения

После запуска приложения необходимо произвести настройку порта **«COM-порт» - COM4**, задать **«Скорость обмена» - 9600**, указать команду чтения названия модуля **«Команда» - $05M**. Затем не-

обходимо последовательно нажать кнопки «Открыть порт», «Записать в порт», «Закрыть порт».

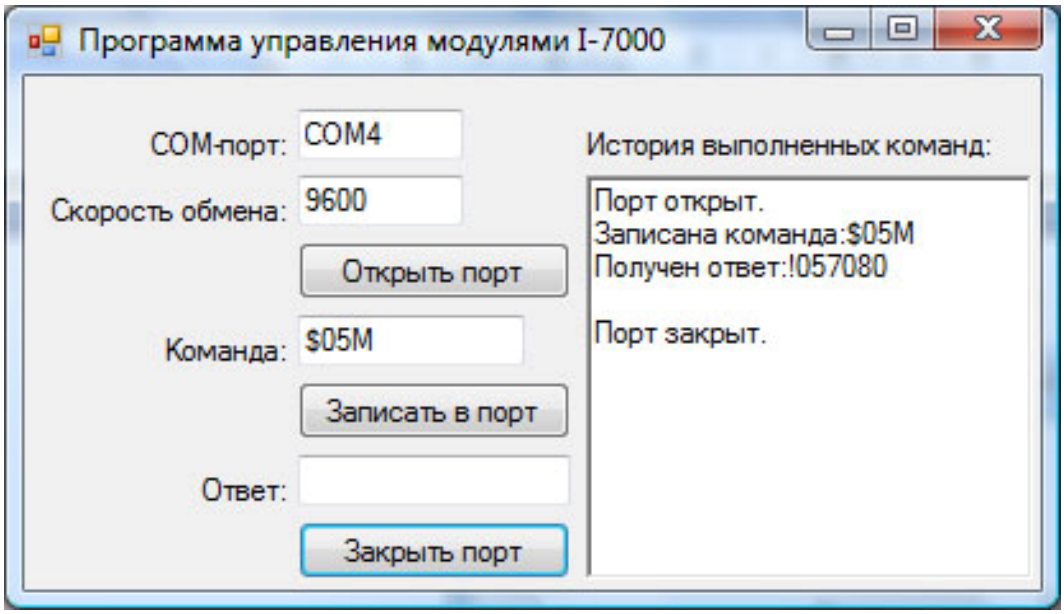


Рис. 73. Проверка работоспособности программы

Результат взаимодействия программы с модулем можно оценить по истории выполненных команд, отображаемой в одноименном списке (рис. 73).

ЗАКЛЮЧЕНИЕ

Особенностью современного производства является тесное взаимодействие технологических (производственных) и информационных потоков. В этих условиях большую роль играют данные, регистрируемые на всех уровнях автоматизированной системы управления технологическими процессами. Требования, предъявляемые со стороны потребителей этой информации, все более ужесточаются в части объема, скорости и надежности получения данных, поэтому вопросы обеспечения коммуникаций становятся высокоприоритетными. В качестве элементной базы таких систем используется класс совместимых модулей удаленного сбора данных I-7000, ADAM-4000, NuDAM-6000, взаимодействующих через интерфейс RS-485.

В монографии проведен анализ технических характеристик наиболее часто используемых модулей удаленного ввода-вывода серии I-7000 и даны рекомендации по их применению в системах сбора и обработки данных.

Модули удаленного ввода-вывода серии I-7000 фирмы ICP DAS обеспечивают недорогое, гибкое и эффективное решение для самого широкого спектра индустриальных и лабораторных задач. Изделия этой серии предназначены для управления технологическим процессом, встраивания в технологическое оборудование, удаленного сбора и обработки информации, могут использоваться в качестве коммуникационных устройств и т.п. Контроллеры включены в Государственный реестр средств измерений и допущены к применению в Российской Федерации.

Разработка программного обеспечения иллюстрируется многочисленными примерами с использованием как визуальной среды программирования (Lasarus, Borland Delphi, Visual C++, C#), так и графической (LabVIEW).

СПИСОК ЛИТЕРАТУРЫ

1. Бень, Е.А. RS-485 для чайников [Электронный ресурс] – www.mayak-bit.narod.ru.
2. Вольфган Эйзенбард. Промышленные шины для систем автоматизации [Электронный ресурс] - www.asutp.ru.
3. Графкин, А.В. Архитектура автоматизированных систем на основе модулей ICP DAS серии I-7000 [Текст]: учебное пособие / А.В. Графкин, В.Г. Иоффе – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2009. – 160 с.: ил.
4. Технические средства автоматизации. Программно-технические комплексы и контроллеры [Текст]: учеб. пособие/ И.А. Елизаров, Ю.Ф. Мартемьянов, А.Г. Схиртладзе [и др.] - М.: Машиностроение-1, 2004. - 180 с.
5. Климентьев, К.Е. Графическое программирование в среде LabVIEW / К. Климентьев. - Самара: Самар. гос. аэрокосм. ун-т, 2003. - 76 с.
6. Кругляк, К.М. Промышленные сети: цели и средства [Текст] / К.М. Кругляк // Современные технологии автоматизации. - 2002. -№4. – С. 6 – 17.
7. Локотков, А. Интерфейсы последовательной передачи данных. Стандарты EIA RS-422/RS-485 // Современные технологии автоматизации. – 1997. – №3. – С. 110-119.
8. Любавин, С.А. Программирование на Delphi Win32 / С.А. Любавин. – М.: НТ Пресс, 2008. – 576с.: ил.
9. Любашин, А.Н. Промышленные сети [Электронный ресурс] – www.asutp.ru.
10. Модули удаленного ввода-вывода серии I-7000. [Электронный ресурс] - http://www.ipc2u.ru./I7000rus.pdf, I7018rus.pdf, I7021rus.pdf, I7013rus.pdf, I7044rus.pdf.
11. Основы построения интерфейсов [Электронный ресурс] – www. info-system.ru/interface.
12. Синенко, О. Стандарты де-факто и де-юре, регламентирующие средства промышленной автоматизации [Электронный ресурс] - www.archive.expert.ru.
13. Суранов, А.Я. LabVIEW 8.20: справочник по функциям / А.Я. Суранов. – М.: ДМК, 2007. – 536 с.: ил.
14. Хетагуров, Я.А. Практические методы построения надежных цифровых систем. Проектирование, производство, эксплуатация / Я.А. Хетагуров. – М.: Высшая школа, 2008 – 156 с.: ил.

Интернет-сайты:

icpdas-usa.com, phyton.ru, microchip.ru, cec-mc.ru, intel.ru, ni.com, icos.ru, ipc2u.ru

ПРИЛОЖЕНИЕ 1

Методика выполнения лабораторной работы

Цель работы: изучение архитектуры автоматизированных систем на основе модулей серии I-7000.

Задание на самостоятельную работу

1. Получить задание у преподавателя.
2. В соответствии с заданием выбрать необходимые модули, изучить их структуру, особенности применения и состав команд.
3. Выписать команды, необходимые для реализации задания. Общие команды модулей приведены в приложении П.6, а индивидуальные – в приложениях П.1-П.5. Требуемые команды оформить в виде таблицы.
4. Выполнить соединение модулей в соответствии с заданием.
5. Подключить модули к компьютеру. **В случае использования преобразователя интерфейсов RS-232 в RS-485 (модуль I-7520AR), подключение модулей производить при выключенном компьютере!**
6. Проверить функционирование модулей с помощью DCON Utility:
   - 6.1. Запустить утилиту.
   - 6.2 Установить номер используемого COM-порта (при работе с USB задать номер виртуального COM-порта).
   - 6.3. Установить допустимое время ожидания ответа модулей.
   - 6.4. Установить скорость, на которой будет выполняться обмен информацией. Если скорость обмена, установленная в модулях неизвестна – выполнить поиск в полном диапазоне.
   - 6.5. В случае необходимости включить проверку контрольной суммы.
   - 6.6. Запустить поиск модулей. Результаты поиска отображаются в окне DCON Utility. Двойной клик по имени модуля открывает окно параметров и тестирования.
7. Если требуется изменить состав модулей, производится реконфигурация системы и повторный запуск DCON Utility.
8. Изменение параметров модулей и их тестирование выполняется в окне «Terminal», в котором можно задать и выполнить требуемую команду.

9. Разработать программное обеспечение. В состав программного обеспечения обязательно должен входить интерфейс пользователя, обеспечивающий функционирование автоматизированной системы. При отладке программы, в случае необходимости, следует использовать DCON Utility.

10. Показать преподавателю работающую в соответствии с заданием программу.

11. Оформить отчет.

Содержание отчета

1. Задание.
2. Цель работы.
3. Схема соединения модулей системы.
4. Таблица с используемыми командами.
5. Листинг программы с комментариями, интерфейс пользователя с экранными формами.

Пример выбора команд

Задание:

Определить частоту, поступающую на вход модуля 7080, и преобразовать ее в пропорциональное значение выходного напряжения на выходе ЦАП 7021. Частота изменяется в диапазоне от 0 до 100 кГц, а напряжение на выходе - (0÷10) В.

Для настройки модуля **I-7080** необходимо:

1. Задать параметры конфигурации: установить режим измерения частоты (51), время измерения (например, 0,1 с), скорость обмена, включить или выключить контрольную сумму, задать требуемый адрес. Сравнить параметры конфигурации модуля с требуемыми значениями. Параметры конфигурации могут быть прочитаны DCON Utility и, по возможности, их изменять не рекомендуется, т.к. число циклов программирования EEPROM ограничено.

2. Выбрать номер используемого счетчика и задать тип входа (с оптической развязкой или неизолированный). Например, Сч1, Сч0, неизолированный (команда $AAB0, где AA – адрес модуля).

3. Установить уровни входных напряжений. В лабораторной работе рекомендуется использовать уровни, устанавливаемые по умолчанию логический «0» - 0,8 В, логическая «1» - 2,4 В. Уровни могут быть установлены программно $AA1H – для верхнего уровня, $AA1L – для нижнего.

4. Принять решение о включении или отключении сторожевого таймера WD. Если таймер WD не используется, выполнить команду

~AA3O00, а затем проверить состояние модуля ~AAO. Если результат выполнения этой команды равен «!AAO», то модуль готов к функционированию. В противном случае состояние модуля следует сбросить командой ~AA1.

5. Команда #AAN читает значение частоты в Гц, представленное в 16-ричном коде, где N – номер канала измерителя частоты 0 или 1.

Настройка **I-7021** состоит в следующем:

1. Задать параметры конфигурации аналогично предыдущему. Особенностью ЦАП является необходимость указания скорости нарастания входного сигнала В/с или А/с и формата представления данных: технические единицы (В, мА) в % от диапазона, в дополнительном 16-ричном коде. Для сформулированной задачи можно использовать по умолчанию заводские настройки, адрес изменять только по необходимости, а данные представлять в 16-ричном коде.

2. Принять решение о необходимости калибровки. Ввиду отсутствия эталонного оборудования в лабораторной работе калибровка не производится.

3. Настроить модуль на работу без сторожевого таймера. Для этого необходимо произвести сброс статуса модуля ~AA1.

4. Перед началом работы запросить статус сброса $AA5. Если в результате ответа установлен бит 1, то модуль приведен в исходное состояние. В противном случае произвести повторный сброс модуля.

5. Задать значение, устанавливаемое на выходе ЦАП, при включении питания равное нулю. Для этого следует выполнить команду #AA0000, а затем команду $AA4.

Для повышения достоверности аналогового вывода можно использовать команды $AA6 (чтение последнего значения, переданного на аналоговый выход) и $AA8 (эхоконтроль выхода). Команда $AA8 производит считывание информации непосредственно с выхода ЦАП. Поэтому с ее помощью можно определить недопустимую нагрузку, неправильное подключение проводов, короткое замыкание и т.д.

Настройка модулей I-7080 и I-7021закончена.

Измерение частоты выполняется командой #AAN, где N – номер канала измерения. При установленном времени измерения 0,1с и максимальной входной частоте в счетчике будет накоплено 10000 (2710h), разрядность ЦАП – 12 бит. Поэтому значение счетчика необходимо масштабировать путем сдвига вправо на 2 разряда.

Полученное значение записать командой #AA (данные) в ЦАП и измерить цифровым мультиметром или модулем I-7018P.

При программировании необходимо помнить, что в работе с модулями для повышения достоверности используются пакеты подтверждения

различных форматов. Кроме того, многие команды позволяют производить проверки значений записанных данных.

Для увеличения быстродействия в процессе работы системы часть проверок можно исключить.

Заключительным этапом выбора команд является составление таблицы команд, используемых для реализации задания (см. табл. 6, 7).

Таблица 6

Таблица команд модуля I-7080

Команда	Ответное сообщение	Содержание команды
$AA2	!AATTCCFF	Чтение конфигурации
%AANNTTCCFF	!AA	Запись конфигурации
$AABS	!AA	Установка типа входных линий
~AA3ETT	!AA	Включение сторожевого таймера
~AA0	!AASS	Чтение состояния модуля
~AA1	!!AA	Сброс состояния модуля
#AAN	>(данные)	Чтение содержимого счетчика

Таблица 7.

Таблица команд модуля I-7021

Команда	Ответное сообщение	Содержание команды
$AA2	!AATTCCFF	Чтение конфигурации
%AANNTTCCFF	!AA	Запись конфигурации
~AA1	!AA	Сброс статуса модуля
$AA5	!AAS	Запросить статус сброса
#AA(данные)	>	Задать значение на выходе

Окончание табл. 7

$AA4	!AA	Задать значение на выходе при включении питания
$AA6	!AA(данные)	Считать значение, переданное на аналоговый выход последним
$AA8	!AA(данные)	Эхоконтроль выхода

Пример интерфейса пользователя модуля I-7044 приведен на рис. 74.

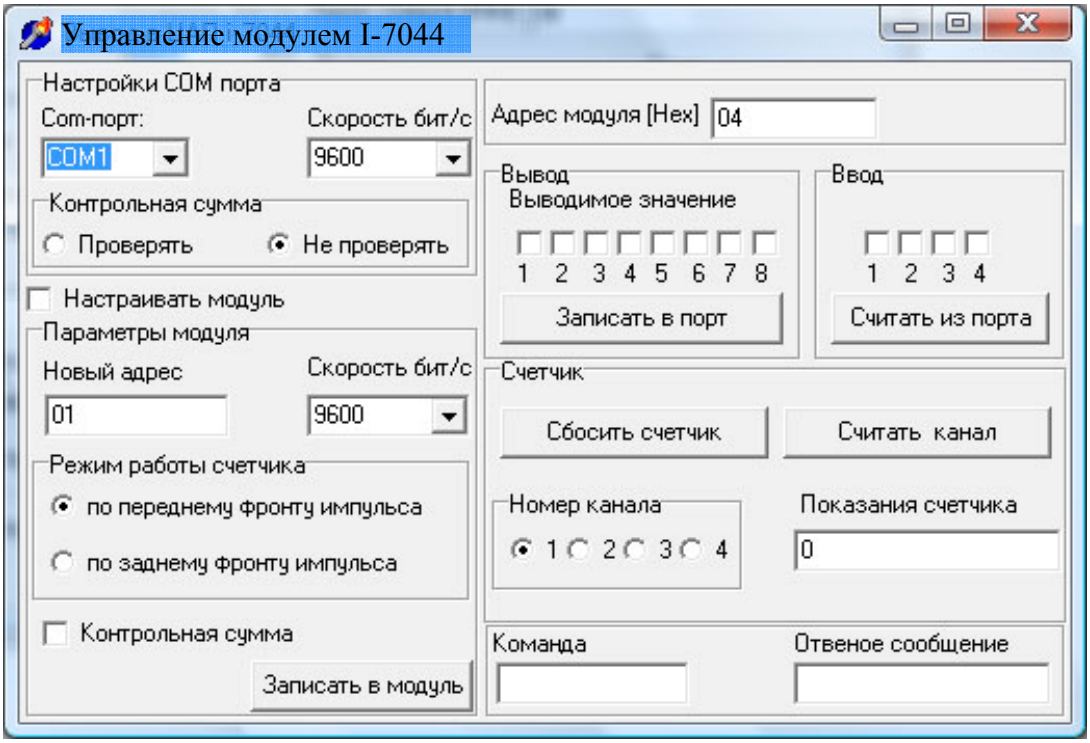


Рис. 74. Интерфейс пользователя программы

Контрольные вопросы

Общие сведения

1. Какие преимущества обеспечивают децентрализованные автоматизированные системы по сравнению с централизованными?

2. Почему передача данных в цифровом коде обладает большей помехозащищенностью и надежностью по сравнению с аналоговыми каналами связи?

3. Какие технические решения необходимо использовать для обеспечения живучести систем?

4. Какие преимущества обеспечивает применение иерархических промышленных сетей?

5. В чем особенности передачи информации на уровне планирования? уровне управления? цеховом уровне? уровне периферийного оборудования?

6. Какие преимущества при проектировании обеспечивает использование семиуровневой модели открытых систем?

7. Принципы обмена информацией в семиуровневой модели открытых систем.

8. В чем особенности физического уровня?

9. Какие характеристики физического уровня следует анализировать при использовании интерфейса?

Интерфейс RS-485/RS-422

10. Каковы основные достоинства интерфейса RS-485?

11. Какими средствами обеспечивается повышенная помехозащищенность RS-485?

12. Сколько устройств может быть подключено к RS-485? Как увеличить число устройств?

13. Какие протоколы обмена используются в RS-485?

14. Сравнительная характеристика RS-485 и RS-422.

Модули удаленного ввода-вывода серии I-7000

15. В чем основные достоинства модулей I-7000? Какие у них недостатки?

16. В чем особенности сети на базе модулей I-7000?

17. Какие каналы передачи данных могут использоваться в системе на основе модулей I-7000?

18. Принцип обмена информацией в системе на основе модулей I-7000.

Использование модулей в измерениях и управлении технологическими процессами

19. Какие функции реализуют модули I-7000?

20. Для каких целей служит ЭППЗУ? Какая информация в нем хранится?

21. Какие параметры модулей могут быть изменены только при заземленном входе INIT?

22. Как считать номер версии программного обеспечения модуля? название модуля?

23. Как изменить название модуля?

24. Какие функции выполняет сторожевой таймер?

25. Какие действия реализуются в модуле при сбросе или переполнении сторожевого таймера?

26. С помощью каких команд выполняется настройка сторожевого таймера? установка безопасного состояния выходов модулей?

Модуль аналогового ввода I-7018P

27. Какие функции выполняет модуль аналогового вывода I-7018P?

28. Как выполнить калибровку модуля I-7018P?

29. Как реализовать циклический или выборочный опрос каналов?

30. Как организовать преобразование сигнала термопары с учетом компенсации холодного спая, если данный модуль не содержит её градуировку?

31. Как организовать с помощью одного модуля измерение сигналов нескольких термопар различных градуировок?

Модуль аналогового вывода I-7021

32. Какие функции выполняет модуль аналогового вывода I-7021?

33. Как выполнить калибровку модуля I-7021?

34. Как с помощью модуля I-7021 реализовать генератор заданной формы? преобразователь напряжения в ток?

35. Какие методы обеспечения достоверности выходного сигнала используются в модуле?

Модуль аналогового ввода параметров терморезисторов I-7033/7013

36. Какие функции выполняет модуль аналогового ввода параметров терморезисторов I-7033/7013?

37. Как выполнить калибровку модуля I-7033/7013?

38. Как использовать модуль для измерения параметров терморезисторов произвольной градуировки? потенциометрических датчиков?

39. В каких случаях целесообразно использовать двухпроводную схему? трехпроводную? четырехпроводную?

40. Почему в схеме рис. 11,д влияние линии связи меньше, чем в схеме рис. 11,б,в?

Модуль счетчика-частотомера I-7080

41. Какие функции выполняет модуль I-7080 в режиме счетчика? частотомера?

42. В каких случаях следует использовать дифференциальные входные сигналы? Как настроить счетчик на работу с дифференциальными сигналами?

43. Для каких целей используется программно-управляемый фильтр? С помощью каких команд можно управлять фильтром?

44. Для каких целей используется вход GATE? Какие команды управляют этим входом?

45. Какие особенности функционирования счетчика в режиме сигнализации 0? режиме 1?

46. В чем особенности подключения нагрузки к дискретным выходам?

47. С помощью каких команд можно управлять дискретными выходами в режиме измерителя частоты? счетчика?

48. Как задать допустимые значения в режиме сигнализации 0? режиме сигнализации 1?

49. В чем недостатки измерителя частоты? Как устранить эти недостатки?

50. Как на базе модуля построить формирователь импульсов заданной длительности? широтно-импульсный модулятор?

Модуль дискретного ввода- вывода I-7044

51. Какие функции выполняет модуль дискретного ввода- вывода I-7044?

52. Как подключить внешнюю нагрузку ко входу модуля I-7044? к выходу модуля I-7044?

53. Как активизировать выходные линии модуля I-7044?

54. Как прочитать состояние линий ввода-вывода I-7044?

55. Как подсчитать количество импульсов, приходящих на вход требуемого канала I-7044?

56. Как установить исходное значение выходных линий I-7044 при включении питания и срабатывании сторожевого таймера?

Алгоритм управления модулями

57. Как изменяется длина кадра в зависимости от особенностей команды?

58. Какой код используется для кодирования информации? В чем его преимущества и недостатки?

59. В чем особенности режима «Master-Slave»?

60. Как реализовать синхронное чтение входных данных из нескольких модулей?

61. Какие методы повышения достоверности информации используются в автоматизированных системах на базе модулей I-7000?

62. В чем преимущества и недостатки контрольной суммы по модулю 256?

63. Как задать режим контрольной суммы? Как она формируется?

64. Какие требования должны предъявляться к интерфейсу пользователя?

65. Как реализовать процедуру обмена информацией между ПЭВМ и модулями системы?

66. За счет каких средств можно повысить быстродействие системы на базе модулей I-7000?

Разработка программного обеспечения

67. Какие функции реализует утилита «SetCom»?

68. Какие действия необходимо выполнить, чтобы назначить преобразователю интерфейсов новый адрес виртуального последовательного порта?

69. Для каких целей используется утилита «DCON Utility»? Какие функции реализованы в ней?

70. К чему сводится управление модулями на уровне ПЭВМ?

71. Перечислите параметры Com-порта, используемые при его настройке.

Разработка программного обеспечения в LabVIEW

72. Какую утилиту необходимо установить, чтобы обеспечить доступ к модулям ICP DAS из LabVIEW?

73. Перечислите динамически подключаемые библиотеки (*.dll), обеспечивающие программный интерфейс com-порта и модулей I-7000.

74. Чем обусловлено использование последовательности (Stacked Sequence Structure) при управлении модулями? Какие действия выполняются на вкладках этой структуры?

75. Как настраивается узел вызова функции библиотеки «Call Library Function Node»? Какие параметры настраиваются?

Разработка программного обеспечения в Lazarus

76. Какие модули необходимо подключить к проекту, чтобы обеспечить доступ к COM-порту?

77. Какой класс обеспечивает возможность реализации параллельного потока и для каких целей он используется?

78. Что следует помнить при отображении информации на элементах интерфейса приложения в параллельном процессе?

79. Назовите тип объекта COM-порта и перечислите основные действия, которые необходимо выполнить при настройке.

80. Какой метод класса TBlockSerial обеспечивает запись команд в COM-порте?

Разработка программного обеспечения в Borland Delphi

81. Какие модули необходимо добавить в проект пользователя, чтобы обеспечить доступ к COM-порту?

82. Какой класс обеспечивает взаимодействие с COM-портом?

83. Содержимое какой переменной реестра используется для получения списка существующих в данной конфигурации COM-портов?

84. Каким образом назначается функция-обработчик чтения данных из COM-порта?

85. Какую функцию выполняет процедура OnRead?

86. Какой метод класса TComPort обеспечивает запись команд в COM-порте?

Разработка программного обеспечения в Visual C++

87. Какая динамически подключаемая библиотека используется для работы с COM-портом?

88. Какой класс реализует функции управления COM-портом?

89. В каком случае метод Open класса SerialGate возвращает true?

90. Какую функцию реализует метод Send класса SerialGate? Опишите его параметры и результат.

91. Какой метод класса SerialGate реализует функцию чтения данных из COM-порта?

92. Какие методы класса SerialGate управляют состоянием сигнальных линий COM-порта и возвращают их значение?

93. Как получить информацию об установленных в системе COM-портах?

94. Для каких целей в программе на C++ используется таймер?

Разработка программного обеспечения в Visual C#

95. Какое пространство имен обеспечивает доступ к COM-портам?

96. Каким образом в C# реализуется подключение пространства имен к проекту?

97. Какой класс используется для взаимодействия с COM-портом?

98. Какие параметры COM-порта задаются при его открытии с помощью метода Open класса SerialPort?

99. Какие методы обеспечивают взаимодействие с COM-портом?

100. В каком режиме (синхронном или асинхронном) производится чтение данных из COM-порта? Какие классы обеспечивают этот режим?

ПРИЛОЖЕНИЕ 2

СИСТЕМА КОМАНД МОДУЛЯ I-7018P

Таблицы параметров настройки:

Настройка скорости передачи (CC)

Код	03	04	05	06	07	08	09	0A
Скорость передачи	1200	2400	4800	9600	19200	38400	57600	115200

Настройка типа аналогового входа (TT)

Код типа входа	00	01	02	03	04	05	06
Минимальное значение входного сигнала	-15 мВ	-50 мВ	-100 мВ	-500 мВ	-1 В	-2,5 В	-20 мА
Максимальное значение входного сигнала	+15 мВ	+50 мВ	+100 мВ	+500 мВ	+1 В	+2,5 В	+20 мА

Код типа входа	0E	0	10	11	12	13	14	15	16	17	18
Тип термопары	J	K	T	E	R	S	B	N	C	L	M
Минимальная температура	-210	-270	-270	-270	0	0	0	-270	0	-200	-200
Максимальная температура	760	1372	400	1000	1768	1768	1820	1300	2320	800	100
Значения температуры приведены в градусах Цельсия											

Настройка формата данных (FF)

7	6	5	4	3	2	1	0
*1	*2	0	0	0	0	*3	

*1: Бит выбора режекторного фильтра: 0 = подавление частоты 60Гц; 1 = подавление частоты 50Гц.

*2: Бит контроля суммы: 0 = контроль суммы запрещен; 1 = контроль суммы разрешен.

*3: Биты формата данных: 00 = в технических единицах; 01 = в процентах от полного диапазона (ПД).

10 = в дополнительном (дополнение до 2) шестнадцатеричном коде.

Таблица типов аналогового входа и форматов данных

Код типа входа	Входной диапазон	Формат данных	+ПД	Ноль	-ПД
00	-15 мВ ÷ +15 мВ	Технические единицы	+15.000	+00.000	-15.000
		% от полного диапазона	+100.00	+000.00	-100.00

Продолжение табл.

		Доп. шестнадцатеричный код	7FFF	0000	8000
01	-50 мВ ÷ +50 мВ	Технические единицы	+50.000	+00.000	-50.000
		% от полного диапазона	+100.00	+000.00	-100.00
		Доп. шестнадцатеричный код	7FFF	0000	8000
02	-100мВ÷ +100 мВ	Технические единицы	+100.00	+000.00	-100.00
		% от полного диапазона	+100.00	+000.00	-100.00
		Доп. шестнадцатеричный код	7FFF	0000	8000
03	-500 мВ ÷ +500 мВ	Технические единицы	+500.00	+000.00	-500.00
		% от полного диапазона	+100.00	+000.00	-100.00
		Доп. шестнадцатеричный код	7FFF	0000	8000
04	-1 В ÷ +1 В	Технические единицы	+1.0000	+0.0000	-1.0000
		% от полного диапазона	+100.00	+000.00	-100.00
		Доп. шестнадцатеричный код	7FFF	0000	8000
05	-2,5 В ÷ +2,5 В	Технические единицы	+2.5000	+0.0000	-2.5000
		% от полного диапазона	+100.00	+000.00	-100.00
		Доп. шестнадцатеричный код	7FFF	0000	8000
06	-20 мА ÷ +20 мА	Технические единицы	+20.000	+00.000	-20.000
		% от полного диапазона	+100.00	+000.00	-100.00
		Доп. шестнадцатеричный код	7FFF	0000	8000
0E	Термопара J-типа -210 °C ÷ +760 °C	Технические единицы	+760.00	+00.000	-210.00
		% от полного диапазона	+100.00	+000.00	-027.63
		Доп. шестнадцатеричный код	7FFF	0000	DCA2
0	Термопара K-типа -270 °C ÷ +1372 °C	Технические единицы	+1372.0	+00.000	-0270.0
		% от полного диапазона	+100.00	+000.00	-019.68
		Доп. шестнадцатеричный код	7FFF	0000	E6D0
10	Термопара T-типа -270 °C ÷ +400 °C	Технические единицы	+400.00	+000.00	-270.00
		% от полного диапазона	+100.00	+000.00	-067.50
		Доп. шестнадцатеричный код	7FFF	0000	A99A

Окончание табл.

11	Термопара E-типа -270 °C ÷ +1000 °C	Технические единицы	+1000.0	+000.00	-0270.0
		% от полного диапазона	+100.00	+000.00	-027.00
		Доп. шестнадцатеричный код	7FFF	0000	DD71
12	Термопара R-типа 0 °C ÷ +1768 °C	Технические единицы	+1768.0	+0000.0	+0000.0
		% от полного диапазона	+100.00	+0000.0	+0000.0
		Доп. шестнадцатеричный код	7FFF	0000	0000
13	Термопара S-типа 0 °C ÷ +1768 °C	Технические единицы	+1768.0	+0.0000	+0000.0
		% от полного диапазона	+100.00	+000.00	+0000.0
		Доп. шестнадцатеричный код	7FFF	0000	0000
14	Термопара B-типа 0 °C ÷ +1820 °C	Технические единицы	+1820.0	+00.000	+0000.0
		% от полного диапазона	+100.00	+000.00	+0000.0
		Доп. шестнадцатеричный код	7FFF	0000	0000
15	Термопара N-типа -270 °C ÷ +1300 °C	Технические единицы	+1300.0	+00.000	-0270.0
		% от полного диапазона	+100.00	+000.00	-20.77
		Доп. шестнадцатеричный код	7FFF	0000	E56B
16	Термопара C-типа 0 °C ÷ +2320 °C	Технические единицы	+2320.0	+00.000	+00.000
		% от полного диапазона	+100.00	+000.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000	0000

Код типа входа	Входной диапазон	Формат данных	+ПД	Ноль	-ПД
17	Термопара L-типа -200 °C ÷ +800 °C	Технические единицы	+800.00	+00.000	-200.00
		% от полного диапазона	+100.00	+000.00	-025.00
		Доп. шестнадцатеричный код	7FFF	0000	E000
18	Термопара M-типа -200 °C ÷ +100 °C	Технические единицы	+100.00	+000.00	-200.00
		% от полного диапазона	+050.00	+000.00	-100.00
		Доп. шестнадцатеричный код	4000	0000	8000

ПД – полный диапазон

Система команд:

Команда	Ответное сообщение	Описание	Раздел
%AANNTTCCFF	!AA	Настроить параметры конфигурации модуля	Раздел П.1
#AA	> (Данные)	Считать значение сигналов на аналоговых входах	Раздел П.2
#AAN	> (Данные)	Считать значение сигнала по каналу «N» аналогового входа	Раздел П.3
$AA0	!AA	Выполнить калибровку диапазона	Раздел П.4
$AA1	!AA	Выполнить калибровку нуля	Раздел П.5
$AA2	!AATTCCFF	Считать параметры конфигурации модуля	Раздел П.6
$AA3	> (Данные)	Считать значение температуры «холодного» спая	Раздел П.7
$AA5VV	!AA	Включить определенные каналы аналогового ввода	Раздел П.8
$AA6	!AAVV	Считать состояние каналов аналогового ввода	Раздел П.9
$AA9(Данные)	!AA	Задать величину смещения для схемы компенсации «холодного» спая	Раздел П.10
~AAEV	!AA	Разрешить/Запретить выполнение калибровки	Раздел П.11

П.1 %AANNTTCCFF

Назначение команды: Настроить параметры конфигурации модуля
Формат команды: %AANNTTCCFF[CHK](cr)
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)

Для изменения настроек скорости передачи или контрольной суммы необходимо замкнуть контакт INIT* на землю.

В случае попытки изменения настроек скорости передачи или контрольной суммы при незамкнутом на землю контакте INIT* модуль выдаст ответное сообщение о недопустимой команде **?**

Пример:
Команда: %0102050600 Ответное сообщение: !02

Изменяется адрес модуля с «01» на «02». Успешное выполнение.
См. также команды:
Раздел А. 6 Команда $AA2

П.2 #AA

Назначение команды: Считать значения сигналов на аналоговых входах.
Формат команды: #AA[CHK](cr)
Ответное сообщение: Допустимая команда: >(Данные)[CHK](cr)
Для модулей I-7018/18P эти данные представляют собой комбинацию значений для каждого из каналов аналогового ввода соответственно.
Пример:
Команда: #04 Ответное сообщение: >+05.123+04.153+07.234-02.356+10.000-05.133+02.345+08.234.
По адресу 04 находится модуль I-7018. В результате считывания информации с этого модуля получены данные о значениях сигналов по 8 каналам аналогового ввода.
См. также команды:
Раздел П.1 Команда %AANNTTCCFF, Раздел П.6 Команда $AA2

П.3 #AAN

Назначение команды: Считать значение сигнала по каналу “N” аналогового входа
Формат команды: #AAN[CHK](cr)
N - номер канала, по которому считывается значение аналогового сигнала (от 0 до 7).
Ответное сообщение: Допустимая команда: >(Данные) [CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: #032 Ответное сообщение: >+02.513
Считывается значение аналогового сигнала по каналу 2 модуля с адресом 03. Данные получены успешно.
Команда: #029 Ответное сообщение: ?02
При считывании значения аналогового сигнала по каналу 9 модуля с адресом 02 принято ответное сообщение о недопустимой команде (ошибка в номере канала).
См. также команды:
Раздел П.1 Команда %AANNTTCCFF, Раздел П.6 Команда $AA2

П.4 $AA0

Назначение команды: Выполнить калибровку диапазона
Формат команды: $AA0[CHK](cr)

0 - команда на выполнение калибровки диапазона.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Символ ? формируется, если выполнение калибровки предварительно не было разрешено.
Пример:
Команда: $010 Ответное сообщение: !01
Выполняется калибровка диапазона аналогового ввода модуля с адресом 01. Успешное выполнение.
Команда: $020 Ответное сообщение: ?02
При попытке выполнения калибровки диапазона аналогового ввода модуля с адресом 02 принято ответное сообщение о недопустимой команде (перед тем как подать команду калибровки необходимо разрешить выполнение такой операции).
См. также команды:
Раздел П.5 Команда $AA1, Раздел П.11 Команда ~AAEV

П.5 $AA1
Назначение команды: Выполнить калибровку нуля
Формат команды: $AA1[CHK](cr)
1 - команда на выполнение калибровки нуля.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Символ ? формируется, если выполнение калибровки предварительно не было разрешено
Пример:
Команда: $011 Ответное сообщение: !01
Выполняется калибровка нуля аналогового ввода модуля с адресом 01. Успешное выполнение.
Команда: $021 Ответное сообщение: ?02
При попытке выполнения калибровки нуля аналогового ввода модуля с адресом 02 принято ответное сообщение о недопустимой команде (перед тем, как подать команду калибровки необходимо разрешить выполнение такой операции).
См. также команды:
Раздел П.4 Команда $AA0, Раздел П.11 Команда ~AAEV

П.6 $AA2
Назначение команды: Считать параметры конфигурации модуля
Формат команды: $AA2[CHK](cr)
2 - команда считывания параметров конфигурации

Ответное сообщение: Допустимая команда: !AATTCCFF[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $012 Ответное сообщение: !01050600
Считываются параметры конфигурации модуля с адресом 01. Успешное выполнение.
Команда: $022 Ответное сообщение: !02030602
Считываются параметры конфигурации модуля с адресом 02. Успешное выполнение.
См. также команды:
Раздел П.1 Команда %AANNTTCCFF

П.7 $AA3

Назначение команды: Считать значение температуры «холодного» спая.
Формат команды: $AA3[CHK](cr)
3 - команда считывания значения температуры «холодного» спая
Ответное сообщение: Допустимая команда:>(Данные) [CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
(Данные) значение температуры «холодного» спая в градусах Цельсия.
Пример:
Команда: $033 Ответное сообщение: >+0025.4
При считывании температуры «холодного» спая в модуле с адресом 03 получено значение +25,4°C.
См. также команды:
Раздел П.10 Команда $AA9(Данные)

П.8 $AA5VV

Назначение команды: Включить определенные каналы аналогового ввода
Формат команды: $AA5VV[CHK](cr)
5 - команда включения определенных каналов аналогового ввода
VV – параметр, определяющий комбинацию включенных и отключенных каналов ввода. При значении «00» этого параметра все каналы аналогового ввода отключены, а при значении «FF» - все каналы включены.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $0155A Ответное сообщение: !01
В модуле с адресом 01 включаются каналы 1, 3, 4, 6 и отключаются каналы 0, 2, 5, 7. Успешное выполнение.
Команда: $016 Ответное сообщение: !015A

При считывании состояния каналов аналогового ввода модуля с адресом 01 получено ответное сообщение о том, что каналы 1, 3, 4, 6 включены, а каналы 0, 2, 5, 7 выключены.
См. также команды:
Раздел П.9 Команда $AA6

П.9 $AA6
Назначение команды: Считать состояние каналов аналогового ввода
Формат команды: $AA6[CHK](cr)
6 - команда считывания состояния каналов аналогового ввода
Ответное сообщение: Допустимая команда: !AAVV[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
VV – параметр, содержащий информацию о комбинации включенных и отключенных каналов. При значении «00» этого параметра все каналы аналогового ввода отключены, а при значении «FF» - все каналы включены.
Пример:
Команда: $015A5 Ответное сообщение: !01
В модуле с адресом 01 включаются каналы 0, 2, 5, 7 и отключаются каналы 1, 3, 4, 6. Успешное выполнение.
Команда: $016 Ответное сообщение: !01A5
При считывании состояния каналов аналогового ввода модуля с адресом 01 получено ответное сообщение о том, что каналы 0, 2, 5, 7 включены, а каналы 1, 3, 4, 6 - выключены.
См. также команды:
Раздел П.8 Команда $AA5VV

П.10 $AA9(Данные)
Назначение команды: Задать величину смещения для схемы компенсации «холодного» спая
Формат команды: $AA9(Данные)[CHK](cr)
9 - команда настройки смещения в схеме компенсации «холодного» спая
(Данные) значение смещения в схеме компенсации «холодного» спая включает в себя знак смещения и четырехразрядное шестнадцатеричное число величины смещения от -1000 до +1000, каждая единица которого соответствует 0,01°C.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $019+0010 Ответное сообщение: !01

Установить для схемы компенсации холодного спая в модуле с адресом 01 положительное смещение величиной 16 единиц (+0,16°C). Успешное выполнение.
См. также команды:
Раздел П.7 Команда $AA3

П.11 ~AAEV

Назначение команды: Разрешить/Запретить выполнение калибровки.
Формат команды: ~AAEV[CHK](cr)
E - команда разрешения или запрещения выполнения калибровки
V 1 = разрешить калибровку; 0 = запретить калибровку.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $010 Ответное сообщение: ?01
При попытке выполнения калибровки диапазона аналогового ввода модуля с адресом 01 принято ответное сообщение о том, что модуль не готов к калибровке.
Команда: ~01E1 Ответное сообщение: !01
Команда разрешения калибровки модуля с адресом 01. Успешное выполнение.
Команда: $010 Ответное сообщение: !01
Выполняется калибровка диапазона аналогового ввода модуля с адресом 01. Успешное выполнение.
См. также команды:
Раздел П.4 Команда $AA0, Раздел П.5 Команда $AA1

ПРИЛОЖЕНИЕ 3

СИСТЕМА КОМАНД МОДУЛЯ I-7080

Таблицы параметров настройки:

Таблица кодов настройки скорости передачи: CC

CC	03	04	05	06	07	08	09	0A
Скорость передачи, бит/с	1200	2400	4800	9600	19200	38400	57600	115200

Код настройки: FF, 2 символа (для всех)

7	6	5	4	3	2	1	0
0	*1	0			*2	0	

*1: Бит контроля суммы: 0 = контроль суммы запрещен; 1 = контроль суммы разрешен

*2: Бит длительности эталонного временного интервала: 0 = длительность интервала равна 0,1 с; 1 = длительность интервала равна 1,0 с.

Таблица кодов настройки режима работы: TT

TT	50	51
Режим работы	Счетчик	Частотомер

Общие команды модуля

Команда	Ответ	Описание	Раздел
%AANNTTCCFF	!AA	Установить параметры конфигурации	П.12
#AAN	>(data)	Считать показания счетчик или таймера	П.13
$AA2	!AATTCCFF	Считать параметры конфигурации	П.23
$AAI	!AAS	Считать значение INIT*	П.36

Команды измерителя частоты

Команда	Ответ	Описание	Ссылка
$AAB	!AAS	Считать режим ввода	П.43
$AABS	!AA	Установить режим ввода	П.44
$AA1H	!AA(data)	Считать значение уровня входного сигнала, соответствующего логической «1»	П.25
$AA1H(data)	!AA	Установить значение уровня входного сигнала, соответствующего логической «1»	П.26

Продолжение табл.

$AA1L	!AA(data)	Считать значение уровня входного сигнала, соответствующего логическому «0»	П.27
$AA1L(data)	!AA	Установить значение уровня входного сигнала, соответствующего логическому «0»	П.28

Команды счётчика

~AAAS	!AA	Установить режим сигнализации	П.14
$AA0H	!AA(data)	Считать минимальную длительность входного сигнала уровня «1»	П.15
$AA0H(data)	!AA	Установить минимальную длительность входного сигнала уровня «1»	П.16
$AA0L	!AA(data)	Считать минимальную длительность входного сигнала уровня «0»	П.17
$AA0L(data)	!AA	Установить минимальную длительность входного сигнала уровня «0»	П.18
$AA1H	!AA(data)	Считать значение входного сигнала уровня «1»	П.19
$AA1H(data)	!AA	Установить значение входного сигнала уровня «1»	П.20
$AA1L	!AA(data)	Считать значение входного сигнала уровня «0»	П.21
$AA1L(data)	!AA	Установить значение входного сигнала уровня «0»	П.22
$AA3N	!AA(data)	Считать максимальное значение счётчика	П.24
$AA3N(data)	!AA	Установить максимальное значение счётчика	П.25
$AA4	!AAS	Считать состояние фильтра	П.26
$AA4S	!AA	Установить состояние фильтра	П.27
$AA5N	!AAS	Считать состояние счётчика	П.28
$AA5NS	!AA	Установить состояние счётчика	П.29
$AA6N	!AA	Сбросить показания счётчика	П.30
$AA7N	!AAS	Считать флаг переполнения	П.31
$AAA	!AAG	Считать режим управления внешним запуском	П.32
$AAAG	!AA	Установить режим управления внешним запуском	П.33
$AAB	!AAS	Считать режим ввода	П.34
$AABS	!AA	Установить режим ввода	П.35
@AADI	!AAS0D00	Считать состояние дискретных выходов и сигналов управления	П.37

Окончание табл.

@AADO0D	!AA	Установить дискретные выходы	П.38
@AAGN	!AA(data)	Считать исходное значение счетчика	П.44
@AAPN(data)	!AA	Установить исходное значение счетчика	П.45

Команды сигнального режима 0

@AAEAN	!AA	Включить режим	П.39
@AADAN	!AA	Отключить режим	П.43
@AAPA(data)	!AA	Установить порог срабатывания счётчика 0	П.46
@AASA(data)	!AA	Установить порог срабатывания счётчика1	П.48
@AARP	!AA	Считать значение порога срабатывания счетчика 0	П.50
@AARA	!AA	Считать значение порога срабатывания счетчика 1	П.52

Команды сигнального режима 1

@AAEAT	!AA	Включить сигнализацию	П.40
@AACA	!AA	Выполнить сброс сигнала состояния тревоги	П.41
@AADA	!AA	Отключить сигнализацию	П.42
@AAPA(data)	!AA	Установить нижний порог сигнализации счетчика 0	П.47
@AASA(data)	!AA	Установить верхний порог сигнализации счетчика 0	П.49
@AARP	!AA	Считать значение нижнего порога сигнализации счетчика 0	П.51
@AARA	!AA	Считать значение верхнего порога сигнализации счетчика 0	П.53

П.12 %AANNTTCCFF

Назначение команды: Настроить параметры конфигурации модуля
Формат команды: %AANNTTCCFF[chk](cr)
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда?AA[chk](cr)
Пример:
Команда: %0102500600(cr) Ответное сообщение: !02(cr)
Изменяется адрес модуля с «01» на «02», и модуль переводится в режим счетчика.
Команда: %0202510600(cr) Ответное сообщение: !02(cr)
Модуль переводится в режим частотомера.

Примечание:
Если подать команду %AANNTTCCFF для изменения параметров настройки (конфигурации) модуля, то новый код конфигурации будет немедленно занесен в ЭСППЗУ. Код конфигурации включает в себя адрес модуля, код типа модуля, код скорости передачи данных, код разрешения/запрета контроля суммы, код калибровки, а также значения, определяющие состояние выходов модуля по включении питания и в случае приведения его в безопасный режим работы.
Хранящиеся в ЭСППЗУ модулей серии I-7000 данные могут быть считаны оттуда неограниченное количество раз и записаны туда примерно 100000 раз максимум.
Таким образом, пользователю не следует слишком часто изменять код конфигурации, только лишь в целях тестирования.
Команда $AA2 применяется только для считывания данных из ЭСППЗУ, поэтому пользователь может подавать эту команду модулям серии I-7000 неограниченное количество раз.

П.13 #AAN
Назначение команды: Считать показания счетчика или частотомера.
Формат команды: #AAN[chk](cr)
N параметр номера канала
N=0: канал 0 счетчика или частотомера
N=1: канал 1 счетчика или частотомера
Ответное сообщение:
Допустимая команда: >(Данные)(cr)
Недопустимая команда: Ответное сообщение не передается
(Данные) 8 символов (в шестнадцатеричной форме)
Пример:
Команда: $012(cr) Ответное сообщение: !01500600(cr)
Команда: #010(cr) Ответное сообщение: >0000001E(cr)
Считываются показания по каналу 0 счетчика: счетчик 0 = 0x1E = 30 (в десятичной форме)
Команда: $022(cr) Ответное сообщение: !02510600(cr)
Команда: #021(cr) Ответное сообщение: >0000001E(cr)
Считываются показания по каналу 1 частотомера: частота = 0x1E Гц = = 30 Гц (в десятичной форме)

П.14 ~AAAS
Назначение команды: Задать режим работы устройства сигнализации о состоянии счетчика.
Формат команды: ~AAAS[chk](cr)

S параметр, определяющий режим работы устройства сигнализации:
S=0: режим 0; S=1: режим 1.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: ~01A0(cr) Ответное сообщение: !01(cr)
Установить устройство сигнализации в режим 0.
Команда: ~02A1(cr) Ответное сообщение: !02(cr)
Установить устройство сигнализации в режим 1.

П.15 $AA0H

Назначение команды: Считать значение минимальной длительности входного сигнала высокого логического уровня.
Формат команды: $AA0H[chk](cr)
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
(Данные) пятизначное десятичное число, соответствующее значению минимальной длительности сигнала высокого логического уровня в микросекундах. Возможный диапазон значений: от 2 мкс до 65535 мкс.
Пример:
Команда: $010H(cr) Ответное сообщение: !0100010(cr)
Минимальная длительность равна 10 мкс.
Команда: $020H(cr) Ответное сообщение: !0201000(cr)
Минимальная длительность равна 1000 мкс или 1 мс.

П.16 $AA0H(Данные)

Назначение команды: Задать значение минимальной длительности входного сигнала высокого логического уровня.
Формат команды: $AA0H(Данные)[chk](cr)
(Данные) пятизначное десятичное число, соответствующее значению минимальной длительности сигнала высокого логического уровня в микросекундах. Возможный диапазон значений: от 2 мкс до 65535 мкс.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $010H00010(cr) Ответное сообщение: !01(cr)
Минимальная длительность сигнала устанавливается равной10 мкс.
Команда: $020H01000(cr) Ответное сообщение: !02(cr)
Минимальная длительность сигнала устанавливается равной 1000 мкс или 1 мс.

П.17 $AA0L

Назначение команды: Считать значение минимальной длительности входного сигнала низкого логического уровня.
Формат команды: $AA0L[chk](cr)
Ответное сообщение: Допустимая команда:!AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
(Данные) пятизначное десятичное число, соответствующее значению минимальной длительности сигнала низкого логического уровня в микросекундах. Возможный диапазон значений: от 2 мкс до 65535 мкс.
Пример:
Команда: $010L(cr) Ответное сообщение: !0100020(cr)
Минимальная длительность равна 20 мкс.
Команда: $020L(cr) Ответное сообщение: !0202000(cr)
Минимальная длительность равна 2000 мкс или 2 мс.

П.18 $AA0L(Данные)
Назначение команды: Задать значение минимальной длительности входного сигнала низкого логического уровня.
Формат команды: $AA0L(Данные)[chk](cr)
(Данные) пятизначное десятичное число, соответствующее значению минимальной длительности сигнала низкого логического уровня в микросекундах. Возможный диапазон значений: от 2 мкс до 65535 мкс.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $010L00020(cr) Ответное сообщение: !01(cr)
Минимальная длительность сигнала устанавливается равной 20 мкс.
Команда: $020L02000(cr) Ответное сообщение: !02(cr)
Минимальная длительность сигнала устанавливается равной 2000 мкс или 2 мс.

П.19 $AA1H
Назначение команды: Считать пороговое значение уровня сигнала логической единицы на неизолированном входе.
Формат команды: $AA1H[chk](cr)
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
(Данные) двузначное десятичное число, соответствующее пороговому значению входного сигнала высокого логического уровня в десятых долях вольта. Возможный диапазон значений: от 0.0 В до 5.0 В.
Пример:

Команда: $011H(cr) Ответное сообщение: !0124(cr)
Значение порогового уровня сигнала логической единицы равно 2.4 В.
Команда: $021H(cr) Ответное сообщение: !0230(cr)
Значение порогового уровня сигнала логической единицы равно 3.0 В.

П.20 $AA1H(Данные)
Назначение команды: Задать пороговое значение уровня сигнала логической единицы на неизолированном входе.
Формат команды: $AA1H(Данные)[chk](cr)
(Данные) двузначное десятичное число, соответствующее пороговому значению входного сигнала высокого логического уровня в десятых долях вольта. Возможный диапазон значений: от 0.0 В до 5.0 В.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $011H24(cr) Ответное сообщение: !01(cr)
Значение порогового уровня входного сигнала логической единицы задается равным 2.4 В.
Команда: $021H30(cr) Ответное сообщение: !02(cr)
Значение порогового уровня входного сигнала логической единицы задается равным 3.0 В.
Примечание:
По умолчанию пороговое значение устанавливается равным 2.4 В.

П.21 $AA1L
Назначение команды: Считать пороговое значение уровня сигнала логического нуля на неизолированном входе.
Формат команды: $AA1L[chk](cr)
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
(Данные) двузначное десятичное число, соответствующее пороговому значению входного сигнала низкого логического уровня в десятых долях вольта. Возможный диапазон значений: от 0.0 В до 5.0 В.
Пример:
Команда: $011L(cr) Ответное сообщение: !0108(cr)
Значение порогового уровня сигнала логического нуля равно 0.8 В
Команда: $021L(cr) Ответное сообщение: !0210(cr)
Значение порогового уровня сигнала логического нуля равно 1.0 В
П.22 $AA1L(Данные)

Назначение команды: Задать пороговое значение уровня сигнала логического нуля на неизолированном входе.
Формат команды: $AA1L(Данные)[chk](cr)
(Данные) двузначное десятичное число, соответствующее пороговому значению входного сигнала низкого логического уровня в десятых долях вольта. Возможный диапазон значений: от 0.0 В до 5.0 В.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $011L08(cr) Ответное сообщение: !01(cr)
Значение порогового уровня входного сигнала логического нуля задается равным 0.8 В.
Команда: $021L10(cr) Ответное сообщение: !02(cr)
Значение порогового уровня входного сигнала логической единицы задается равным 1.0 В.
Примечание:
По умолчанию пороговое значение устанавливается равным 0.8 В.

П.23 $AA2
Назначение команды: Считать параметры конфигурации модуля
Формат команды: $AA2[chk](cr)
Ответное сообщение: Допустимая команда: !AATTCCFF[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $012(cr) Ответное сообщение: !01500600(cr)
Считываются параметры настройки: адрес модуля - 01; режим работы - счетчик; скорость передачи – 9600 бит/с; контроль суммы не производится.
Команда: $022(cr) Ответное сообщение: !02510700(cr)
Считываются параметры настройки: адрес модуля - 02; режим работы - частотомер; скорость передачи - 19200 бит/с; контроль суммы не производится.
Примечание:
Если подать команду %AANNTTCCFF для изменения параметров настройки (конфигурации) модуля, то новый код конфигурации будет немедленно занесен в ЭСППЗУ. Код конфигурации включает в себя адрес модуля, код типа модуля, код скорости передачи данных, код разрешения/запрета контроля суммы, код калибровки, а также значения, определяющие состояние выходов модуля по включении питания и в случае приведения его в безопасный режим работы.
Хранящиеся в ЭСППЗУ модулей серии I-7000 данные могут быть считаны

оттуда неограниченное количество раз и записаны туда примерно 100000 раз максимум. Таким образом, пользователю не следует слишком часто изменять код конфигурации, только лишь в целях тестирования.
Команда $AA2 применяется только для считывания данных из ЭСППЗУ, поэтому пользователь может подавать эту команду модулям серии I-7000 неограниченное количество раз.

П.24 $AA3N
Назначение команды: Запросить максимальное показание счетчика.
Формат команды: $AA3N[chk](cr)
N – параметр номера канала: N=0: канал 0 счетчика или частотомера; N=1: канал 1 счетчика или частотомера.
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $0130(cr) Ответное сообщение: !010000FFFF(cr)
Диапазон показаний счетчика 0: от предварительно заданного исходного значения до числа «FFFF».
Команда: $0131(cr) Ответное сообщение: !01FFFFFFFF(cr)
Диапазон показаний счетчика 1: от предварительно заданного исходного значения до числа «FFFFFFFF».

П.25 $AA3N(Данные)
Назначение команды: Задать максимальное показание счетчика.
Формат команды: $AA3N(Данные)[chk](cr)
N – параметр номера канала: N=0: канал 0 счетчика или частотомера; N=1: канал 1 счетчика или частотомера.
(Данные) восьмизначное шестнадцатеричное число.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $01300000FFFF(cr) Ответное сообщение: !01 (cr)
Задать диапазон показаний счетчика 0: от предварительно заданного исходного значения до числа «FFFF».
Команда: $0131FFFFFFFF(cr) Ответное сообщение: !01 (cr)
Задать диапазон показаний счетчика 1: от предварительно заданного исходного значения до числа «FFFFFFFF».

П.26 $AA4
Назначение команды: Запросить информацию о состоянии цифрового фильтра.
Формат команды: $AA4[chk](cr)

Ответное сообщение: Допустимая команда: !AAS[chk](cr)
Недопустимая команда: ?AA[chk](cr)
S – параметр состояния цифрового фильтра:
S=0: цифровой фильтр выключен; S=1: цифровой фильтр включен.
Пример:
Команда: $014(cr) Ответное сообщение: !010(cr)
Цифровой фильтр выключен.
Команда: $024(cr) Ответное сообщение: !021(cr)
Цифровой фильтр включен.

П.27 $AA4S
Назначение команды: Задать состояние цифрового фильтра.
Формат команды: $AA4S[chk](cr)
S – параметр, определяющий состояние цифрового фильтра: S=0: выключить цифровой фильтр; S=1: включить цифровой фильтр.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $0140(cr) Ответное сообщение: !01(cr)
Цифровой фильтр выключен.
Команда: $0241(cr) Ответное сообщение: !02(cr)
Цифровой фильтр включен.

П.28 $AA5N
Назначение команды: Запросить информацию о состоянии счетчика.
Формат команды: $AA5N[chk](cr)
N – параметр номера канала: N=0: счетчик 0; N=1: счетчик 1.
Ответное сообщение: Допустимая команда: !AAS[chk](cr)
Недопустимая команда: ?AA[chk](cr)
S параметр состояния счетчика: S=0: счетчик остановлен (выключен); S=1: счетчик запущен (включен).
Пример:
Команда: $0150(cr) Ответное сообщение: !010(cr)
В данный момент счетчик 0 остановлен.
Команда: $0151(cr) Ответное сообщение: !011(cr)
В данный момент счетчик 1 запущен.

П.29 $AA5NS
Назначение команды: Задать состояние счетчика.
Формат команды: $AA5NS[chk](cr)
N – параметр номера канала:N=0: счетчик 0; N=1: счетчик 1.

S – параметр, определяющий состояние счетчика: S=0: остановить счетчик; S=1: запустить счетчик.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $01500(cr) Ответное сообщение: !01(cr)
Остановить счетчик 0.
Команда: $01511(cr) Ответное сообщение: !02(cr)
Запустить счетчик 1.

П.30 $AA6N
Назначение команды: Произвести сброс показаний счетчика 0 или счетчика 1 к предварительно заданному исходному значению и снять флаг переполнения.
Формат команды: $AA6N[chk](cr)
N – параметр номера канала: N=0: счетчик 0; N=1: счетчик 1.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: @01G0(cr) Ответное сообщение: !0100000000(cr)
Для данного счетчика предварительно задано исходное значение «0».
Команда: $0160(cr) Ответное сообщение: !01(cr)
Произвести сброс показаний счетчика 0 к предварительно заданному исходному значению «0».
Команда: @01G1(cr) Ответное сообщение: !010000ABCD(cr)
Для данного счетчика предварительно задано исходное значение 0xABCD.
Команда: $0161(cr) Ответное сообщение: !01(cr)
Произвести сброс показаний счетчика 1 к предварительно заданному исходному значению 0xABCD.

П.31 $AA7N
Назначение команды: Запросить статус переполнения счетчика.
Для приведения счетчика в исходное состояние и удаления флага переполнения можно воспользоваться командой $AA6N.
Формат команды: $AA7N[chk](cr)
N – параметр номера канала: N=0: Счетчик 0; N=1: Счетчик 1.
Ответное сообщение: Допустимая команда: !AAS[chk](cr)
Недопустимая команда: ?AA[chk](cr)
S – параметр переполнения: S=0: нет переполнения; S=1: есть переполнение.
Пример:

Команда: $0170(cr) Ответное сообщение: !011(cr)
Имеет место переполнение счетчика 0.
Команда: $0160(cr) Ответное сообщение: !01(cr)
Снять флаг переполнения.
Команда: $0171(cr) Ответное сообщение: !010(cr)
Счетчик 1 работает нормально.

П.32 $AAA

Назначение команды: Запросить режим управления внешним запуском.
Формат команды: $AAA[chk](cr)
Ответное сообщение: Допустимая команда: !AAG[chk](cr)
Недопустимая команда: ?AA[chk](cr)
G – параметр режима работы временного селектора:
G=0: счет разрешается при наличии на входе «GATE» строб-импульса низкого логического уровня
G=1: счет разрешается при наличии на входе «GATE» строб-импульса высокого логического уровня
G=2: стробирование выключено.
Пример:
Команда: $01A(cr) Ответное сообщение: !010(cr)
Стробирование импульсом низкого логического уровня.
Команда: $02A(cr) Ответное сообщение: !021(cr)
Стробирование импульсом высокого логического уровня.
Команда: $03A(cr) Ответное сообщение: !032(cr) Стробирование выключено (счетчик работает постоянно).

П.33 $AAAG

Назначение команды: Установить режим управления внешним запуском
Формат команды: $AAAG[chk](cr)
G – параметр режима работы схемы внешнего запуска:
G=0: счет разрешается при наличии на входе «GATE» строб-импульса низкого логического уровня.
G=1: счет разрешается при наличии на входе «GATE» строб-импульса высокого логического уровня.
G=2: стробирование выключено.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $01A0(cr) Ответное сообщение: !01(cr)
Стробирование импульсом низкого логического уровня.
Команда: $02A1(cr) Ответное сообщение: !02(cr) Стробирование импуль-

сом высокого логического уровня.
Команда: $03A2(cr) Ответное сообщение: !03(cr)
Стробирование выключено (счетчик работает постоянно).

П.34 $AAB

Назначение команды: Запросить режим ввода
Формат команды: $AAB[chk](cr)
Ответное сообщение: Допустимая команда: !AAS[chk](cr)
Недопустимая команда: ?AA[chk](cr)
S – параметр режима ввода:
S=0: Канал 0 - неизолированный; Канал 1 – неизолированный.
S=1: Канал 0 - изолированный; Канал 1 - изолированный S=2: Канал 0 - неизолированный; Канал 1 - изолированный S=3: Канал 0 - изолированный; Канал 1 – неизолированный.
Пример:
Команда: $01B(cr) Ответное сообщение: !010(cr)
Каналы 0 и 1 счетчика или частотомера неизолированные.
Команда: $02B(cr) Ответное сообщение: !021(cr)
Каналы 0 и 1 счетчика или частотомера изолированные.
Команда: $03B(cr) Ответное сообщение: !032(cr)
Канал 0 счетчика или частотомера неизолированный, а Канал 1 - изолированный.

П.35 $AABS

Назначение команды: Установить режим ввода
Формат команды: $AABS[chk](cr)
S – параметр, определяющий режима ввода:
S=0: Канал 0 - неизолированный; Канал 1 – неизолированный;
S=1: Канал 0 - изолированный; Канал 1 – изолированный;
S=2: Канал 0 - неизолированный; Канал 1 – изолированный;
S=3: Канал 0 - изолированный; Канал 1 – неизолированный;
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: $01B0(cr) Ответное сообщение: !01(cr).
Каналы 0 и 1 счетчика или частотомера неизолированные.
Команда: $02B1(cr) Ответное сообщение: !02(cr) Каналы 0 и 1 счетчика или частотомера изолированные.
Команда: $03B2(cr) Ответное сообщение: !03(cr).
Канал 0 счетчика или частотомера неизолированный, а Канал 1 - изолированный.

П.36 $AAI

Назначение команды: Запросить состояние контакта *INIT
Формат команды: $AAI[chk](cr)
Ответное сообщение: Допустимая команда: !AAS[chk](cr)
Недопустимая команда: ?AA[chk](cr)
S – параметр состояния контакта *INIT:
S=0: контакт *INIT соединен с контактом GND; S=1: контакт *INIT не замкнут на землю.
Пример:
Команда: $01I(cr) Ответное сообщение: !010(cr).
Контакт *INIT соединен с контактом GND.
Команда: $02I(cr) Ответное сообщение: !021(cr).
Контакт *INIT не замкнут на землю.

П.37 @AADI

Назначение команды: Запросить состояние дискретных выходов и сигнализации.
Формат команды: @AADI[chk](cr)
Ответное сообщение: Допустимая команда: !AAS0D00[chk](cr)
Недопустимая команда: ?AA[chk](cr)
S – параметр состояния устройства сигнализации.
В случае работы устройства сигнализации в Режиме 0:
S=0: сигнализация по счетчику 0 выключена; сигнализация по счетчику 1 выключена;
S=1: сигнализация по счетчику 0 включена; сигнализация по счетчику 1 выключена;
S=2: сигнализация по счетчику 0 выключена; сигнализация по счетчику 1 включена;
S=3: сигнализация по счетчику 0 выключена; сигнализация по счетчику 1 выключена;
В случае работы устройства сигнализации в Режиме 1:
S=0: сигнализация по счетчику 0 выключена;
S=1: сигнализация по счетчику 0 включена и действует в режиме кратковременного срабатывания;
S=2: сигнализация по счетчику 0 включена и действует в режиме с фиксацией факта срабатывания;
D – параметр состояния выходов устройства дискретного вывода: D=0: выходы DO0 и DO1 находятся в состоянии «выключено»;
D=1: выход DO0 находится в состоянии «включено», а выход DO1 - в состоянии «выключено»;
D=2: выход DO0 находится в состоянии «выключено», а выход DO1 - в со-

стоянии «включено»;
D=3: выходы DO0 и DO1 находятся в состоянии «включено».
Пример:
Команда: @01DI(cr) Ответное сообщение: !0100000(cr)
Устройство сигнализации выключено. Дискретные выходы DO0 и DO1 находятся в состоянии «выключено».
Команда: @02DI(cr) Ответное сообщение: !0230100(cr)
Устройство сигнализации включено. Дискретный выход DO0 находится в состоянии «включено», а выход
DO1 - в состоянии «выключено».

П.38 @AADO0D
Назначение команды: Установить дискретные выходы.
Формат команды: @AADO0D[chk](cr)
D – параметр, определяющий состояние дискретных выходов:
D=0: выходы DO0 и DO1 устанавливаются в состояние «выключено»;
D=1: выход DO0 устанавливается в состояние «включено», а выход DO1 - в состояние «выключено»;
D=2: выход DO0 устанавливается в состояние «выключено», а выход DO1 - в состояние «включено»;
D=3: выходы DO0 и DO1 устанавливаются в состояние «включено».
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Включена сигнализация: ?AA[chk](cr)
Пример:
Команда: @01DO00(cr) Ответное сообщение: !01(cr)
Привести все дискретные выходы «выключено».
Команда: @02DO01(cr) Ответное сообщение: !02(cr)
Привести дискретный выход DO0 в состояние «включено», а выход DO1 - в состояние «выключено».
Примечание:
Если устройство сигнализации включено, то дискретные выходы DO0 и DO1 будут постоянно находиться под управлением самого модуля. Таким образом, последующие команды дискретного вывода будут проигнорированы. Состояние выходов, устанавливаемое по включении питания, немедленно изменяется на состояние высокого или низкого уровня. Команда @AADO0D игнорируется.

П.39 @AAEAN
Назначение команды: Включить сигнализацию результата действия счетчика (в случае работы устройства сигнализации в Режиме 0).
Формат команды: @AAEAN[chk](cr)

N – параметр, определяющий канал счетчика, по которому работает устройство сигнализации:
N=0: включить сигнализацию по каналу 0 счетчика;
N=1: включить сигнализацию по каналу 1 счетчика.
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: @01EA0(cr) Ответное сообщение: !01(cr).
Включить сигнализацию по каналу 0 счетчика.
Команда: @02EA1(cr) Ответное сообщение: !02(cr).
Включить сигнализацию по каналу 1 счетчика.

П.40 @AAEAT

Назначение команды: Включить сигнализацию о результате действия счетчика (в случае работы устройства сигнализации в режиме 1).
Формат команды: @AAEAT[chk](cr)
T – параметр, определяющий тип (режим работы) сигнализации: T=M: кратковременная сигнализация;
T=L: сигнализация с фиксацией факта срабатывания (тревоги).
Ответное сообщение: Допустимая команда: !AA[chk](cr)
Недопустимая команда:?AA[chk](cr)
Пример:
Команда: @01EAL(cr) Ответное сообщение: !01(cr).
Сигнализация с фиксацией факта срабатывания.
Команда: @02EAM(cr) Ответное сообщение: !02(cr).
Кратковременная сигнализация.
Примечание:
Если устройство сигнализации включено, то дискретные выходы DO0 и DO1 будут постоянно находиться под управлением самого модуля. Таким образом, последующие команды дискретного вывода будут проигнорированы. Состояние выходов, устанавливаемое по включении питания, немедленно изменяется на состояние высокого или низкого уровня. Команда @AADO0D игнорируется.

П.41 @AACA

Назначение команды: Произвести сброс зафиксированного устройством сигнализации состояния тревоги (в случае работы устройства сигнализации в режиме 1).
Формат команды: @AACA[chk](cr)
Ответное сообщение:Допустимая команда: !AA[chk](cr)

Недопустимая команда:?AA[chk](cr)

Пример:

Команда: @01CA(cr) Ответное сообщение: !01(cr)

Производится сброс состояния тревоги, зафиксированного устройством сигнализации.

Команда: @02CA(cr) Ответное сообщение: !02(cr)

Производится сброс состояния тревоги, зафиксированного устройством сигнализации.

П.42 @AADA

Назначение команды: Отключить сигнализацию (в случае работы устройства сигнализации в режиме 1)

Формат команды: @AADA[chk](cr)

Ответное сообщение:Допустимая команда: !AA[chk](cr)

Недопустимая команда:?AA[chk](cr)

Пример:

Команда: @01DA(cr) Ответное сообщение: !01(cr).

Производится отключение сигнализации.

Команда: @02DA(cr) Ответное сообщение: !02(cr).

Производится отключение сигнализации.

П.43 @AADAN

Назначение команды: Отключить сигнализацию (в случае работы устройства сигнализации в режиме 0).

Формат команды: @AADAN[chk](cr)

N – параметр номера канала:

N=0: отключить сигнализацию по каналу 0 счетчика;

N=1: отключить сигнализацию по каналу 1 счетчика.

Ответное сообщение:Допустимая команда: !AA[chk](cr)

Недопустимая команда:?AA[chk](cr)

Пример:

Команда: @01DA0(cr) Ответное сообщение: !01(cr).

Производится отключение сигнализации по каналу 0 счетчика.

Команда: @02DA1(cr) Ответное сообщение: !02(cr).

Производится отключение сигнализации по каналу 1 счетчика.

П.44 @AAGN

Назначение команды: Считать предварительно заданное значение счетчика. Для выполнения сброса счетчика к этому предварительно задаваемому значению можно воспользоваться командой $AA6.

Формат команды: @AAGN[chk](cr)

N – параметр номера канала:
N=0: считывается предварительно заданное значение для канала 0 счетчика;
N=1: считывается предварительно заданное значение для канала 1 счетчика.
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
Пример:
Команда: @01G0(cr) Ответное сообщение: !010000FFFF(cr).
Для канала 0 счетчика предварительно задано значение «0000FFFF».
Команда: @02G1(cr) Ответное сообщение: !0200000000(cr)
Для канала 1 счетчика предварительно задано значение «00000000».

П.45 @AAPN(Данные)
Назначение команды: Задать предварительно устанавливаемое значение счетчика.
Для выполнения сброса счетчика к этому предварительно задаваемому значению можно использовать команду $AA6.
Формат команды: @AAPN(Данные)[chk](cr)
N – параметр номера канала:
N=0: задается значение для канала 0 счетчика;
N=1: задается значение для канала 1 счетчика.
(Данные) восьмизначное шестнадцатеричное число.
Ответное сообщение:Допустимая команда: !AA[chk](cr)
Недопустимая команда:?AA[chk](cr)
Пример:
Команда: @01P0FFFF0000(cr) Ответное сообщение: !01(cr)
Для канала 0 счетчика предварительно задается значение «FFFF0000».
Команда: @02P10000FFFF(cr) Ответное сообщение: !02(cr)
Для канала 1 счетчика предварительно задается значение «0000FFFF».

П.46 @AAPA(Данные)
Назначение команды: Установить порог срабатывания сигнализации по каналу 0 счетчика (в случае работы устройства сигнализации в режиме 0).
Формат команды: @AAPA(Данные)[chk](cr).
(Данные) восьмизначное шестнадцатеричное число.
Ответное сообщение:Допустимая команда: !AA[chk](cr)
Недопустимая команда:?AA[chk](cr)
Пример:
Команда: @01PAFFFF0000(cr) Ответное сообщение: !01(cr)
Для канала 0 счетчика устанавливается пороговое значение срабатывания

сигнализации «FFFF0000».
Команда: @02PA0000FFFF(cr) Ответное сообщение: !02(cr)
Для канала 0 счетчика устанавливается пороговое значение срабатывания сигнализации «0000FFFF».

П.47 @AAPA(Данные)
Назначение команды: Установить нижний порог срабатывания сигнализации по каналу 0 счетчика (в случае работы устройства сигнализации в режиме 1).
Формат команды: @AAPA(Данные)[chk](cr)
(Данные) восьмизначное шестнадцатеричное число.
Ответное сообщение:Допустимая команда: !AA[chk](cr)
Недопустимая команда:?AA[chk](cr)
Пример:
Команда: @01PAFFFF0000(cr) Ответное сообщение: !01(cr)
Для канала 0 счетчика устанавливается значение «FFFF0000» нижнего порога срабатывания сигнализации.
Команда: @02PA0000FFFF(cr) Ответное сообщение: !02(cr)
Для канала 0 счетчика устанавливается значение «0000FFFF» нижнего порога срабатывания сигнализации.

П.48 @AASA(Данные)
Назначение команды: Установить порог срабатывания сигнализации по каналу 1 счетчика (в случае работы устройства сигнализации в режиме 0).
Формат команды: @AASA(Данные)[chk](cr)
(Данные) восьмизначное шестнадцатеричное число.
Ответное сообщение:Допустимая команда: !AA[chk](cr)
Недопустимая команда:?AA[chk](cr)
Пример:
Команда: @01SAFFFF0000(cr) Ответное сообщение: !01(cr)
Для канала 1 счетчика устанавливается пороговое значение срабатывания сигнализации «FFFF0000».
Команда: @02SA0000FFFF(cr) Ответное сообщение: !02(cr)
Для канала 1 счетчика устанавливается пороговое значение срабатывания сигнализации «0000FFFF».

П.49 @AASA(Данные)
Назначение команды: Установить верхний порог срабатывания сигнализации по каналу 0 счетчика (в случае работы устройства сигнализации в режиме 1).
Формат команды: @AASA(Данные)[chk](cr)

(Данные) восьмизначное шестнадцатеричное число.
Ответное сообщение:Допустимая команда: !AA[chk](cr)
Недопустимая команда:?AA[chk](cr)
Пример:
Команда: @01SAFFFF0000(cr) Ответное сообщение: !01(cr)
Для канала 0 счетчика устанавливается значение «FFFF0000» верхнего порога срабатывания сигнализации.
Команда: @02SA0000FFFF(cr) Ответное сообщение: !02(cr)
Для канала 0 счетчика устанавливается значение «0000FFFF» верхнего порога срабатывания сигнализации.

П.50 @AARP
Назначение команды: Считать пороговое значение срабатывания сигнализации по каналу 0 счетчика (в случае работы устройства сигнализации в режиме 0).
Формат команды: @AARP[chk](cr)
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
(Данные) восьмизначное шестнадцатеричное число
Пример:
Команда: @01RP(cr) Ответное сообщение: !01FFFF0000(cr)
Для канала 0 счетчика установлено пороговое значение срабатывания сигнализации «FFFF0000».
Команда: @02RP(cr) Ответное сообщение: !020000FFFF(cr)
Для канала 0 счетчика установлено пороговое значение срабатывания сигнализации «0000FFFF».

П.51 @AARP
Назначение команды: Считать значение нижнего порога срабатывания сигнализации по каналу 0 счетчика (в случае работы устройства сигнализации в режиме 1).
Формат команды: @AARP[chk](cr)
Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)
(Данные) восьмизначное шестнадцатеричное число
Пример:
Команда: @01RP(cr) Ответное сообщение: !01FFFF0000(cr)
Для канала 0 счетчика установлено значение «FFFF0000» нижнего порога срабатывания сигнализации.
Команда: @02RP(cr) Ответное сообщение: !020000FFFF(cr)

Для канала 0 счетчика установлено значение «0000FFFF» нижнего порога срабатывания сигнализации.

П.52 @AARA

Назначение команды: Считать значение порога срабатывания сигнализации по каналу 1 счетчика (в случае работы устройства сигнализации в режиме 0).

Формат команды: @AARA[chk](cr)

Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr)
Недопустимая команда: ?AA[chk](cr)

(Данные) восьмизначное шестнадцатеричное число

Пример:

Команда: @01RA(cr) Ответное сообщение: !01FFFF0000(cr)

Для канала 1 счетчика установлено пороговое значение срабатывания сигнализации «FFFF0000».

Команда: @02RA(cr) Ответное сообщение: !020000FFFF(cr)

Для канала 1 счетчика установлено пороговое значение срабатывания сигнализации «0000FFFF».

П.53 @AARA

Назначение команды: Считать значение верхнего порога срабатывания сигнализации по каналу 0 счетчика (в случае работы устройства сигнализации в режиме 1).

Ответное сообщение: Допустимая команда: !AA(Данные)[chk](cr);
Недопустимая команда: ?AA[chk](cr).

(Данные) восьмизначное шестнадцатеричное число.

Пример:

Команда: @01RA(cr) Ответное сообщение: !01FFFF0000(cr)

Для канала 0 счетчика установлено значение «FFFF0000» верхнего порога срабатывания сигнализации.

Команда: @02RA(cr) Ответное сообщение: !020000FFFF(cr)

Для канала 0 счетчика установлено значение «0000FFFF» верхнего порога срабатывания сигнализации.

ПРИЛОЖЕНИЕ 4

СИСТЕМА КОМАНД МОДУЛЯ I-7021

Таблицы параметров настройки:

Настройка скорости передачи (CC)

Код	03	04	05	06	07	08	09	0A
Скорость передачи	1200	2400	4800	9600	19200	38400	57600	115200

Настройка типа аналогового входа (TT)

Код типа входа	30	31	32
Минимальное значение выходного сигнала	0 мА	4 мА	0 В
Максимальное значение выходного сигнала	20 мА	20 мА	+10 В

Настройка формата данных (FF)

7	6	5	4	3	2	1	0
0	*1	*2				*3	

*1: Бит контроля суммы: 0 = контроль суммы запрещен; 1 = контроль суммы разрешен.

*2: Контроль скорости изменения выходного сигнала.

*3: Биты формата данных: 00 = в технических единицах; 01 = в процентах от полного диапазона; 10 = в дополнительном (дополнение до 2) шестнадцатеричном коде.

Скорость изменения выходного сигнала					
	В/с	мА /с		В/с	мА /с
0000	Мгновенно		1000	8.0	16.0
0001	0.0625	0.125	1001	16.0	32.0
0010	0.125	0.25	1010	32.0	64.0
0011	0.25	0.5	1011	64.0	128.0
0100	0.5	1.0	1100	128.0	256.0
0101	1.0	2.0	1101	256.0	512.0
0110	2.0	4.0	1110	512.0	1024.0
0111	4.0	8.0			

Тип аналогового выхода и формат данных				
Код типа	Выходной диапазон	Формат данных	Макс.	Мин.
30	0…20 мА	Технические единицы	20.000	00.000
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	FFF	0000

Окончание табл.

31	4…20 мА	Технические единицы	20.000	04.000
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	FFF	0000
32	0…+10 В	Технические единицы	10.000	00.000
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	FFF	0000

Система команд

Команда	Ответное сообщение	Описание	Раздел
%AANNTTCCFF	!AA	Настроить параметры конфигурации модуля	Раздел П.54
$AA2	!AANNTTCCFF	Считать параметры конфигурации модуля	Раздел П.55
$AA5	!AAS	Запросить статус сброса	Раздел П.56
#AA(Данные)	>	Задать значение, устанавливаемое на аналоговом выходе модуля	Раздел П.57
$AA0	!AA	Произвести калибровку значения 4 мА	Раздел П.58
$AA1	!AA	Произвести калибровку значения 20 мА	Раздел П.59
$AA3VV	!AA	Точная подстройка выходного значения при калибровке	Раздел П.60
$AA4	!AA	Задать значение, устанавливаемое на аналоговом выходе модуля по включении питания	Раздел П.61
$AA6	!AA(Данные)	Считать последнее значение, переданное на аналоговый выход	Раздел П.62
$AA7	!AA	Произвести калибровку значения 10 В	Раздел П.63
$AA8	!AA(Данные)	Эхоконтроль выхода	Раздел П.64
~AA4	!AA(Данные)	Считать значение, устанавливаемое на выходе модуля в случае приведения его в безопасный режим работы	Раздел П.65

Окончание табл.

~AA5	!AA	Задать значение, устанавливаемое на выходе модуля в случае приведения его в безопасный режим работы	Раздел П.66

П.54 %AANNTTCCFF

Назначение команды: Настроить параметры конфигурации модуля

Формат команды: %AANNTTCCFF[CHK](cr)

Для изменения значений скорости передачи или контрольной суммы необходимо замкнуть контакт INIT* на землю.

Ответное сообщение: Допустимая команда: !AA[CHK](cr)

Недопустимая команда: ?AA[CHK](cr)

В случае попытки изменения настроек скорости передачи или контрольной суммы при незамкнутом на землю контакте INIT* модуль выдаст ответное сообщение о недопустимой команде ?.

Пример:

Команда: %0102300600 Ответное сообщение: !02

Изменяется адрес модуля с «01» на «02». Успешное выполнение.

Команда: %0202050602 Ответное сообщение: !02

Изменяется параметр формата данных с «00» на «02». Успешное выполнение.

См. также команды: Раздел П.55 Команда $AA2

П.55 $AA2

Назначение команды: Считать параметры конфигурации модуля

Формат команды: $AA2[CHK](cr)

2 - команда считывания параметров конфигурации.

Ответное сообщение: Допустимая команда: !AATTCCFF[CHK](cr)

Недопустимая команда: ?AA[CHK](cr)

Пример:

Команда: $012 Ответное сообщение: !01300600

Считываются параметры конфигурации модуля с адресом 01. Тип выхода «30», диапазон 0…20 мА, формат данных – технические единицы, мгновенное изменение выходного значения. Успешное выполнение.

Команда: $022 Ответное сообщение: !02310602

Считываются параметры конфигурации модуля с адресом 02. Успешное выполнение.

См. также команды: Раздел П.54 Команда %AANNTTCCFF

П.56 $AA5
Назначение команды: Запросить статус сброса
Формат команды: $AA5[CHK](cr)
5 - команда считывания статуса сброса
Ответное сообщение: Допустимая команда: !AAS[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
S – статус сброса: 1 = модуль приведен в исходное состояние;
0 = модуль не приводился в исходное состояние.
Пример:
Команда: $015 Ответное сообщение: !011
При запросе статуса сброса модуля с адресом 01 принимается ответное сообщение, что модуль приведен в исходное состояние.
Команда: $015 Ответное сообщение: !010
При запросе статуса сброса модуля с адресом 01 принимается ответное сообщение, что модуль не приводился в исходное состояние.

П.57 #AA(Данные)
Назначение команды: Задать значение, устанавливаемое на аналоговом выходе модуля.
Формат команды: #AA(Данные)[CHK](cr)
(Данные) значение, устанавливаемое на аналоговом выходе модуля.
Ответное сообщение: Допустимая команда: >[CHK](cr)
Выход за пределы диапазона: ?AA[CHK](cr)
Недопустимая команда: ![CHK](cr)
? - разделитель в случае, когда данные выходят за пределы заданного диапазона. В этом случае на выходе устанавливается максимальное значение выбранного диапазона.
! - разделитель в случае недопустимой команды
Пример:
Команда: $012 Ответное сообщение: !01300600
Считываются параметры конфигурации модуля с адресом 01. Тип выхода «30», диапазон 0…20 мА, формат данных – технические единицы, мгновенное изменение выходного значения. Успешное выполнение.
Команда: #0105.000 Ответное сообщение: >
Для модуля с адресом 01 на выходе устанавливается значение 5.00 мА.
Успешное выполнение.
Команда: #0125.000 Ответное сообщение: ?01.
Для модуля с адресом 01 попытка установить на выходе значение 25.00 мА. На выходе устанавливается значение, равное верхнему пределу рабочего диапазона 20 мА. Возвращается сообщение о выходе за пределы рабочего диапазона.

Команда: $022 Ответное сообщение: !02300601
Считываются параметры конфигурации модуля с адресом 02. Тип выхода «30», диапазон 0…20 мА, формат данных – % от полного диапазона, мгновенное изменение выходного значения. Успешное выполнение.
Команда: #02+050.00 Ответное сообщение: >
Для модуля с адресом 02 на выходе устанавливается значение 50% от диапазона (10мА). Успешное выполнение.
Команда: $032 Ответное сообщение: !02300602
Считываются параметры конфигурации модуля с адресом 03. Тип выхода «30», диапазон 0…20 мА, формат данных – дополнительный шестнадцатеричный код, мгновенное изменение выходного значения. Успешное выполнение.
Команда: #03800 Ответное сообщение: >
Для модуля с адресом 03 на выходе устанавливается значение 0x800 (10мА). Успешное выполнение.
См. также команды: Раздел П.54 Команда %AANNTTCCFF, Раздел П.55 Команда $AA2

П.58 $AA0
Назначение команды: Произвести калибровку значения 4 мА.
Формат команды: $AA0[CHK](cr)
0 - команда калибровки значения 4 мА.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $010 Ответное сообщение: !01
Калибровка значения 4 мА для аналогового выхода модуля с адресом 01. Успешное выполнение.
См. также команды:
Раздел П.59 Команда $AA1 Раздел П.60 Команда $AA3VV

П.59 $AA1
Назначение команды: Произвести калибровку значения 20 мА.
Формат команды: $AA1[CHK](cr)
1 - команда калибровки значения 20 мА.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $011 Ответное сообщение: !01
Калибровка значения 20 мА для аналогового выхода модуля с адресом 01. Успешное выполнение.

См. также команды:
Раздел П.58 Команда $AA0 Раздел П.60 Команда $AA3VV

П.60 $AA3VV

Назначение команды: Точная подстройка выходного значения при калибровке.
Формат команды: $AA3VV[CHK](cr)
3 - команда точная подстройки выходного значения при калибровке.
VV – дополнительный шестнадцатеричный код для точной подстройки выходного значения. Величины от 00 до 5F – для увеличения на 0…95 единиц. Величины от FF до 1A – для уменьшения на 1 до 95 единиц. Каждая единица соответствует 4.88 мкА для токового выхода и 2.44 мВ для потенциального выхода.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $0131F Ответное сообщение: !01
Точная подстройка выходного значения при калибровке модуля с адресом 01. Увеличение выходного значения на 31 единицу. Успешное выполнение.
См. также команды:
Раздел П.58 Команда $AA0, Раздел П.59 Команда $AA1, Раздел П.63 Команда $AA7

П.61 $AA4

Назначение команды: Задать значение, устанавливаемое на аналоговом выходе модуля по включении питания
Формат команды: $AA4[CHK](cr)
4 - команда задания значения, устанавливаемого на аналоговом выходе модуля по включении питания.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: #0100.000 Ответное сообщение: >
Для модуля с адресом 01 на выходе устанавливается значение 0.00
Успешное выполнение.
Команда: $014 Ответное сообщение: !01
Для модуля с адресом 01 задается значение, устанавливаемое на аналоговом выходе модуля по включении питания.
Оно равно значению, присутствующему на выходе модуля **до выполнения команды $AA4.**
Для приведенного примера на выходе было установлено значение 0.00 Успешное выполнение.

После включения питания на выходе данного модуля будет установлено значение 0.00 В.
Успешное выполнение.
См. также команды: Раздел П.57 Команда #AA(Данные)

П.62 $AA6
Назначение команды: Считать последнее значение, переданное на аналоговый выход командой #AA(Данные)
Формат команды: $AA6[CHK](cr)
6 - команда чтения последнего значения, переданного на аналоговый выход.
Ответное сообщение: Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr)
(Данные) последнее значение, переданное на аналоговый выход командой #AA(Данные)
Пример:
Команда: #0110.000 Ответное сообщение: >
Для модуля с адресом 01 на выходе устанавливается значение 10.00.
Успешное выполнение.
Команда: $016 Ответное сообщение: ! 0110.000
Чтение последнего значения, переданного на аналоговый выход модуля с адресом 01. Принято значение 10.00 Успешное выполнение.
См. также команды:
Раздел П.57 Команда #AA(Данные), Раздел П.64 Команда $AA8

П.63 $AA7
Назначение команды: Произвести калибровку значения 10 В.
Формат команды: $AA7[CHK](cr)
7 - команда калибровки значения 10 В.
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
Пример:
Команда: $017 Ответное сообщение: !01
Калибровка значения 10 В для аналогового выхода модуля с адресом 01.
Успешное выполнение.
См. также команды: Раздел П.60 Команда $AA3VV

П.64 $AA8
Назначение команды: Эхоконтроль выхода
Формат команды: $AA8[CHK](cr)
8 - команда эхоконтроля выхода.

Ответное сообщение: Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
(Данные) текущее значение на выходе.
Пример:
Команда: $012 Ответное сообщение: !01320614.
Считываются параметры конфигурации модуля с адресом 01. Тип выхода «20», диапазон 0…10 В, формат данных – технические единицы, скорость изменения выходного значения 1.0 В/с. Успешное выполнение.
Команда: #0110.000 Ответное сообщение: >
Для модуля с адресом 01 на выходе устанавливается значение 10.000.
Успешное выполнение.
Команда: $016 Ответное сообщение: !0110.000.
Чтение последнего значения, переданного на аналоговый выход модуля с адресом 01.
Принято значение 10.00. Успешное выполнение.
Команда: $018 Ответное сообщение: !0101.000.
Чтение текущего значения на аналоговом выходе модуля с адресом 01.
Принято значение 1.0 В. Успешное выполнение.
Команда: $018 Ответное сообщение: !0101.500.
Чтение текущего значения на аналоговом выходе модуля с адресом 01.
Принято значение 1.5 В. Успешное выполнение.
См. также команды:
Раздел П.57 Команда #AA(Данные), Раздел П.62 Команда $AA6

П.65 ~AA4

Назначение команды: Считать значение, устанавливаемое на выходе модуля в случае приведения его в безопасный режим работы
Формат команды: ~AA4[CHK](cr)
4 - команда считывания значения, устанавливаемого на аналоговом выходе модуля в случае приведения модуля в безопасный режим работы.
Ответное сообщение: Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
(Данные) значение, устанавливаемое на выходе модуля в случае приведения его в безопасный режим работы.
Пример:
Команда: ~014 Ответное сообщение: !0105.00.
При опросе модуля с адресом 01 принято сообщение о том, что значение, устанавливаемое на выходе модуля в случае приведения его в безопасный режим работы, равно 5.00 Успешное выполнение.

См. также команды:
Раздел П.94 Команда ~**, Раздел П.95 Команда ~AA0, Раздел П.96 Команда ~AA1, Раздел П.97 Команда ~AA2, Раздел П.98 Команда ~AA3EVV, Раздел П.66 Команда ~AA5.

П.66~AA5

Назначение команды: Задать значение, устанавливаемое на выходе модуля в случае приведения его в безопасный режим работы.
Формат команды: ~AA5[CHK](cr)
5 - команда задания значения, устанавливаемого на аналоговом выходе модуля в случае приведения модуля в безопасный режим работы.
Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
Пример:
Команда: #0100.000 Ответное сообщение: !01.
Для модуля с адресом 01 на выходе устанавливается значение 0.00
Успешное выполнение.
Команда: ~015 Ответное сообщение: !0105.00.
Для модуля с адресом 01 на выходе задается значение, устанавливаемое в случае приведения его в безопасный режим работы. Оно равно значению, присутствующему на выходе модуля **до выполнения команды~AA5**. Для приведенного примера на выходе было установлено значение 0.00 Успешное выполнение.
См. также команды: Раздел П.94 Команда ~**, Раздел П.95 Команда ~AA0, Раздел П.96 Команда ~AA1, Раздел П.97 Команда ~AA2, Раздел П.98 Команда ~AA3EVV, Раздел П.65 Команда ~AA4.

ПРИЛОЖЕНИЕ 5

СИСТЕМА КОМАНД МОДУЛЯ I-7044

Таблицы параметров настройки:
Настройка скорости передачи (CC)

Код	03	04	05	06	07	08	09	0A
Скорость передачи	1200	2400	4800	9600	19200	38400	57600	115200

Настройка типа входа (TT)
Тип входа = 40 для режима дискретного ввода-вывода.

Настройка формата данных (FF)

7	6	5	4	3	2	1	0
*1	*2	0	0	0	0		

*1: Бит режима работы счетчика: 0 = по заднему фронту импульса; 1 = по переднему фронту импульса.
*2: Бит контроля суммы: 0 = контроль суммы запрещен
1 = контроль суммы разрешен

Формат данных при считывании состояния модулей дискретного ввода-вывода.
Данные, получаемые по командам $AA6, $AA4, $AALS: (Первый байт данных)(Второй байт данных)00.
Данные, получаемые по команде @AA: (Первый байт данных)(Второй байт данных).

Первый байт данных		Второй байт данных	
DO (1-8)	00 FF	DI (1-4)	00 0F

Система команд

Команда	Ответное сообщение	Описание	Раздел
%AANNTTCCFF	!AA	Настроить параметры конфигурации модуля	Раздел П.67
#**	Не переда-ется	Команда синхронизированной выборки	Раздел П.68
#AABBDD	>	Установить дискретные выходы	Раздел П.69
#AAN	!AA(Данные)	Считать показания счетчика по каналу «N» дискретного ввода	Раздел П.70
$AA2	!AATTCCFF	Считать параметры конфигурации модуля	Раздел П.71
$AA4	!S(Данные)	Считать синхронизированные данные	Раздел П.72
$AA5	!AAS	Запросить статус сброса	Раздел П.73

Окончание табл.

$AA6	!(Данные)	Считать состояние каналов дискретного ввода-вывода	Раздел П.74
$AAC	!AA	Произвести сброс данных о сигналах, зафиксированных на дискретных входах	Раздел П.75
$AACN	!AA	Произвести сброс показаний счетчика сигналов на дискретном входе	Раздел П.76
$AALS	!(Данные)	Считать данные о сигналах, зафиксированных на дискретных входах	Раздел П.77
@AA	>(Данные)	Считать данные с дискретных входов	Раздел П.78
@AA(Данные)	>	Установить дискретные выходы	Раздел П.79
~AA4V	!AA(Данные)	Считать значения, устанавливаемые на дискретных выходах модуля по включении питания и в случае приведения модуля в безопасный режим работы	Раздел П.80
~AA5V	!AA	Задать значения, устанавливаемые на дискретных выходах модуля по включении питания и в случае приведения его в безопасное состояние	Раздел П.81

П.67 %AANNTTCCFF

Назначение команды: Настроить параметры конфигурации модуля
Формат команды: %AANNTTCCFF[CHK](cr)
Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)

В случае попытки изменения настроек скорости передачи или контрольной суммы при незамкнутом на землю контакте INIT* модуль выдаст ответное сообщение о недопустимой команде ?.

Пример:
Команда: %0102400600 Ответное сообщение: !02.
Изменяется адрес модуля с «01» на «02». Успешное выполнение.
См. также команды:
Раздел П.71 Команда $AA2.

П.68 #**

Назначение команды: Команда выполнения синхронизированной выборки.
Формат команды: #**[CHK](cr)

** команда синхронизированной выборки. Все модули, поддерживающие данную команду, немедленно считывают значение сигнала на дискретном входе и сохраняют его во внутреннем регистре. Эти данные могут быть считаны из каждого модуля командой $AA4.
Ответное сообщение: Не передается
Пример:
Команда: #** Ответное сообщение: Не передается.
На все модули передается команда синхронизированной выборки.
Команда: $014 Ответное сообщение: !10F0000.
Считываются синхронизированные данные в модуле с адресом 01. В ответном сообщении содержится значение статуса синхронизированных данных S=1 (первое чтение) и собственно данные.
Команда: $014 Ответное сообщение: !00F0000.
Считываются синхронизированные данные в модуле с адресом 01. В ответном сообщении содержится значение статуса синхронизированных данных S=0 (данные уже считывались) и собственно данные.
См. также команды:
Раздел П.72 Команда $AA4.

П.69 #AABBDD

Назначение команды: Установить дискретные выходы.
Формат команды: #AABBDD[CHK](cr)
BBDD команда и параметр дискретного вывода.
Для режима многоканального вывода параметр «BB» определяет группу выходных каналов и может иметь значения «00», «0A» или «0B». Параметр «DD» задает значение, определяющее состояние дискретных выходов.

Параметр для режима многоканального вывода					
	Количество выходных каналов	Параметр DD для команды #AABBDD			
		BB=00/0A		BB=0B	
I-7044/44D	8	00 ÷ FF	DO (1-8)	Отсутствует	Отсутствует

Для режима одноканального вывода параметр «BB» может иметь значения «1C», «AC» или «BC», где «C» определяет номер выбранного выходного канала. Параметр «DD» должен иметь значение «00» для того, чтобы привести выбранный канал в состояние «выключено», или же значение «01» для того, чтобы привести его в состояние «включено».

Параметр для режима одноканального вывода				
	Команда одноканального вывода #AABBDD			
	Значение «C», если BB=1C/AC		Значение «C», если BB=BC	
I-7044/44D	0 ÷ 7	DO (1-8)	Отсутствует	Отсутствует

Ответное сообщение: Допустимая команда: >[CHK](cr); Недопустимая команда: ?AA[CHK](cr)
Проигнорированная команда: !AA[CHK](cr)
! - разделитель в том случае, если команда проигнорирована. Такая ситуация имеет место, если установлен флаг срабатывания сторожевого таймера главного ПК, а выходы модуля установлены в состояние, соответствующее «безопасному» значению (Safe Value).
Пример:
Команда: #01000F Ответное сообщение: >
Для модуля, адрес которого равен 01, задается значение 0F. Успешное выполнение.
Команда : #0300FF Ответное е сообщение: !
Предпринимается попытка задать для модуля с адресом «03» выходное значение «FF». Принимается ответное сообщение, что данная команда проигнорирована. В данном модуле установлен флаг срабатывания сторожевого таймера главного ПК, а его выходы приведены в состояние, соответствующее «безопасному» значению.
См. также команды:
Раздел П.79 Команда @AA(Данные)

П.70 #AAN
Назначение команды: Считать показания счетчика по каналу «N» дискретного ввода.
Формат команды: #AAN[CHK](cr)
N - номер опрашиваемого канала.
Ответное сообщение:
Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
(Данные) показания счетчика сигналов на дискретном входе (десятичное число в диапазоне от 00000 до 65535).
Пример:
Команда: #032 Ответное сообщение: !00103
При считывании показаний счетчика по каналу 2 дискретного ввода модуля по адресу 03 получено значение «103».
Команда: #025 Ответное сообщение: ?02
При считывании показаний счетчика по каналу 5 дискретного ввода модуля по адресу 02 получено ответное сообщение о недопустимой команде (ошибка в номере канала).
См. также команды:
Раздел П.76 Команда $AACN

П.71 $AA2
Назначение команды: Считать параметры конфигурации модуля

Формат команды: $AA2[CHK](cr)
2 - команда считывания параметров настройки.
Ответное сообщение: Допустимая команда: !AATTCCFF[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
Пример:
Команда: $012 Ответное сообщение: !01400600
При считывании параметров настройки модуля с адресом 01 принимается ответное сообщение, что модуль работает в режиме дискретного ввода-вывода, скорость передачи 9600 бит/с, контроль суммы не производится.
См. также команды:
Раздел П.67 Команда %AANNTTCCFF

П.72 $AA4
Назначение команды: Считать синхронизированные данные.
Формат команды: $AA4[CHK](cr)
4 - команда считывания синхронизированных данных, сохраненных во внутреннем регистре модуля командой #**
Ответное сообщение: Допустимая команда: !S(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
S – статус синхронизированных данных:
1 = первое чтение; 0 = данные уже считывались.
(Данные) синхронизированные данные о состоянии дискретных входов/выходов.
Пример:
Команда: $014 Ответное сообщение: ?01.
При попытке считывания синхронизированных данных в модуле с адресом 01 принимается ответное сообщение о том, что таковые данные отсутствуют.
Команда: #** Ответное сообщение: Не передается.
На все модули передается команда на выполнение синхронизированной выборки.
Команда: $014 Ответное сообщение: !1000F00.
Считываются синхронизированные данные в модуле с адресом 01. В ответном сообщении содержится значение S=1 статуса синхронизированных данных (первое чтение) и собственно синхронизированные данные «0F00».
См. также команды:
Раздел П.68 Команда #**

П.73 $AA5
Назначение команды: Запросить статус сброса.
Формат команды: $AA5[CHK](cr)
5 - команда считывания статуса сброса.
Ответное сообщение: Допустимая команда: !AAS[CHK](cr)

Недопустимая команда: ?AA[CHK](cr)
S – статус сброса:
1 = модуль приведен в исходное состояние;
0 = модуль не приводился в исходное состояние.
Пример:
Команда: $015 Ответное сообщение: !011
При запросе статуса сброса модуля с адресом 01 принимается ответное сообщение, что модуль приведен в исходное состояние. Команда: $015 Ответное сообщение: !010
При запросе статуса сброса модуля с адресом 01 принимается ответное сообщение, что модуль не приводился в исходное состояние.

П.74 $AA6

Назначение команды: Считать состояние каналов дискретного ввода-вывода.
Формат команды: $AA6[CHK](cr)
6 - команда считывания состояния каналов дискретного ввода-вывода.
Ответное сообщение: Допустимая команда: !(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr)
(Данные) значение, содержащее информацию о состоянии входных и выходных каналов.
Пример:
Команда: $016 Ответное сообщение: !010F00
Предположим, что по адресу «01» находится модуль I-7044 и при считывании в нем состояния каналов дискретного ввода-вывода принимается значение «010F». Это означает, что дискретный выход DO1 и дискретные входы находятся в состоянии «лог.1»
См. также команды:
Раздел П.78 Команда @AA

П.75 $AAC

Назначение команды: Произвести сброс данных о сигналах, зафиксированных на дискретных входах.
Формат команды: $AAC[CHK](cr)
C - команда сброса сигналов, зафиксированных на дискретных входах.
Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
Пример:
Команда: $01L0 Ответное сообщение: !01FFFF00
При считывании данных о сигналах низкого логического уровня, зафиксированных на дискретных входах модуля по адресу «01», получено значение «FFFF».
Команда: $01C Ответное сообщение: !01

Выполняется сброс данных о сигналах, зафиксированных на дискретных входах модуля по адресу «01».
Успешное выполнение.
Команда: $01L0 Ответное сообщение: !01000000
При считывании данных о сигналах низкого логического уровня, зафиксированных на дискретных входах модуля по адресу «01», получено значение «0000».
См. также команды: Раздел П.77 Команда $AALS

П.76 $AACN
Назначение команды: Произвести сброс показаний счетчика сигналов на дискретном входе.
Формат команды: $AACN[CHK](cr)
C - команда сброса показаний счетчика сигналов на дискретном входе.
N - номер «N» канала дискретного ввода, по которому требуется произвести сброс счетчика.
Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
Пример:
Команда: #010 Ответное сообщение: !0100123
При считывании показаний счетчика по каналу «0» дискретного ввода модуля с адресом «01» получено значение «123».
Команда: $01C0 Ответное сообщение: !01
Производится сброс показаний счетчика по каналу «0» дискретного ввода модуля с адресом «01». Успешное выполнение.
Команда: #010 Ответное сообщение: !0100000
При считывании показаний счетчика по каналу «0» дискретного ввода модуля с адресом «01» получено значение «000».
См. также команды:
Раздел П.70 Команда #AAN

П.77 $AALS
Назначение команды: Считать данные о сигналах, зафиксированных на дискретных входах.
Формат команды: $AALS[CHK](cr)
L - команда считывания данных о сигналах, зафиксированных на дискретных входах.
S - параметр выбора критерия фиксации:
1 = фиксировать сигналы высокого логического уровня;
0 = фиксировать сигналы низкого логического уровня.
Ответное сообщение: Допустимая команда: !(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr)

(Данные) считанный статус дискретных входов:
1 = на данном канале ввода сигнал зафиксирован;
0 = на данном канале ввода сигнал не зафиксирован.
Пример:
Команда: $01L1 Ответное сообщение: !012300
При считывании данных о сигналах высокого логического уровня, зафиксированных на дискретных входах модуля по адресу «01», получено значение «0123».
Команда: $01C Ответное сообщение: !01
Выполняется сброс данных о сигналах, зафиксированных на дискретных входах модуля по адресу «01».
Успешное выполнение.
Команда: $01L1 Ответное сообщение: !000000
При считывании данных о сигналах высокого логического уровня, зафиксированных на дискретных входах модуля по адресу «01», получено значение «0».
См. также команды:
Раздел П.75 Команда $AAC

П.78 @AA

Назначение команды: Считать данные с дискретных входов.
Формат команды: @AA[CHK](cr)
Ответное сообщение: Допустимая команда: >(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
(Данные) считанные данные о состоянии дискретных входов и выходов.
Пример:
Команда: @01 Ответное сообщение: >0F00
При считывании данных с дискретных входов модуля по адресу «01» получено значение «0F00».
См. также команды:
Раздел П.74 Команда $AA6.

П.79 @AA(Данные)

Назначение команды: Установить дискретные выходы.
Формат команды: @AA(Данные)[CHK](cr)
(Данные) значение, определяющее состояние дискретных выходов. Для модулей I-7044 параметр (Данные) содержит два символа: от 00 до FF.
Ответное сообщение: Допустимая команда: >[CHK](cr); Недопустимая команда: ?AA[CHK](cr)
Проигнорированная команда: !AA[CHK](cr)
! - разделитель в том случае, если команда проигнорирована. Такая ситуация имеет место, если установлен флаг срабатывания сторожевого таймера

главного ПК, а выходы модуля установлены в состояние, соответствующее «безопасному» значению.
Пример:
Команда: @0200 Ответное сообщение: >
Дискретные выходы модуля по адресу «02» устанавливаются в состояние, соответствующее значению «00». Успешное выполнение.
См. также команды:
Раздел П.69 Команда #AABBDD

П.80 ~AA4V
Назначение команды: Считать значения, определяющие состояние дискретных выходов модуля по включении питания или в случае приведения модуля в безопасный режим работы.
Формат команды: ~AA4V[CHK](cr)
4 - команда считывания значений, определяющих состояние дискретных выходов модуля по включении питания или в случае приведения модуля в безопасный режим работы.
При V=P считать значение по включении питания; V= S считать «безопасные» значения.
Ответное сообщение: Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
(Данные) значение, определяющее состояние дискретных выходов модуля по включении питания или в случае приведения модуля в безопасный режим работы.
Параметр (Данные) имеет формат VV00, где «VV» - значение, определяющее состояние дискретных выходов модуля по включении питания (или в случае приведения модуля в безопасный режим работы).
Пример:
Команда: @010000 Ответное сообщение: >
Дискретные выходы модуля с адресом «01» устанавливаются в состояние, соответствующее значению «0000». Успешное выполнение.
Команда: ~015S Ответное сообщение: !01
Задается «безопасное» значение для модуля с адресом «01». Успешное выполнение.
Команда: @01FFFF Ответное сообщение: >
Дискретные выходы модуля с адресом «01» устанавливаются в состояние, соответствующее значению «FFFF». Успешное выполнение.
Команда: ~015P Ответное сообщение: !01
Задается значение по включении питания для модуля с адресом «01». Успешное выполнение.
Команда: ~014S Ответное сообщение: !010000

При считывании значения, определяющего состояние дискретных выходов модуля с адресом 01 в случае приведения модуля в безопасный режим работы, получено значение «0000».
Команда: ~014P Ответное сообщение: !01FFFF
При считывании значения, определяющего состояние дискретных выходов модуля с адресом 01 по включении питания, получено значение «FFFF».
См. также команды: Раздел П.81 Команда ~AA5V.

П.81~AA5V
Назначение команды: Задать значения, определяющие состояние дискретных выходов модуля по включении питания или в случае приведения модуля в безопасный режим работы.
Формат команды: ~AA5V[CHK](cr)
5 - команда установки значений, определяющих состояние дискретных выходов модуля по включении питания или в случае приведения модуля в безопасный режим работы: V=P принять текущее состояние дискретных выходов модуля в качестве значения, устанавливаемого по включении питания; V =S принять текущее состояние дискретных выходов модуля в качестве значения, устанавливаемого в случае приведения модуля в безопасный режим работы.
Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
Пример:
Команда: @01AA Ответное сообщение: >
Дискретные выходы модуля с адресом «01» устанавливаются в состояние, соответствующее значению «AA». Успешное выполнение.
Команда: ~015P Ответное сообщение: !01
Задается значение по включении питания для модуля с адресом «01». Успешное выполнение. Команда: @0155 Ответное сообщение: >
Дискретные выходы модуля с адресом «01» устанавливаются в состояние, соответствующее значению «AA». Успешное выполнение.
Команда: ~015S Ответное сообщение: !01
Задается «безопасное» значение для модуля с адресом «01». Успешное выполнение.
Команда: ~014P Ответное сообщение: !01AA00
При считывании значения, определяющего состояние дискретных выходов модуля с адресом 01 по включении питания, получено значение «AA». Команда: ~014S Ответное сообщение: !015500
При считывании значения, определяющего состояние дискретных выходов модуля с адресом 01 в случае приведения модуля в безопасный режим работы, получено значение «55».
См. также команды: Раздел П.80 Команда ~AA4V

ПРИЛОЖЕНИЕ 6
СИСТЕМА КОМАНД МОДУЛЯ I-7033/7013

Таблицы параметров настройки:

Настройка скорости передачи (CC)

Код	03	04	05	06	07	08	09	0A
Скорость передачи	1200	2400	4800	9600	19200	38400	57600	115200

Настройка типа аналогового входа (TT)

Код типа входа	Тип термометра сопротивления	Диапазон измеряемых температур
20	Pt100, α=0.00385	–100…+100°C
21	Pt100, α=0.00385	0…+100°C
22	Pt100, α=0.00385	0…+200°C
23	Pt100, α=0.00385	0…+600°C
24	Pt100, α=0.003916	–100…+100°C
25	Pt100, α=0.003916	0…+100°C
26	Pt100, α=0.003916	0…+200°C
27	Pt100, α=0.003916	0…+600°C
28	Ni 120	–80…+100°C
29	Ni 120	0…+100°C
2A	Pt1000, α=0.00385	–200…+600°C

Настройка формата данных (FF)

7	6	5	4	3	2	1	0
*1	*2	0	0	0	0	*3	

*1: Бит выбора режекторного фильтра: 0 = подавление частоты 60Гц;1= подавление частоты 50Гц.

*2: Бит контроля суммы: 0 = контроль суммы запрещен; 1 = контроль суммы разрешен.

*3: Биты формата данных: 00 = в технических единицах; 01 = в процентах от полного диапазона (ПД); 10 = в дополнительном (дополнение до 2) шестнадцатеричном коде; 11 = в омах

Замечание

Установка формата данных в омах позволяет измерять показания и других типов термометров сопротивления, не поддерживаемых данными модулями, например медных TCM-10, TCM-50, TCM-100. В компьютер или контроллер, к которому подключен модуль, в этом случае будут передаваться показания термометра сопротивления в омах. Показания затем легко могут быть пересчитаны в значения температуры, выраженные в градусах.

Таблица типов аналогового входа и форматов данных:

Код типа входа	Входной диапазон	Формат данных	+ПД	-ПД
20	Pt100, α=0.00385 –100…+100°C	Технические единицы	+100.00	-100.00
		% от полного диапазона	+100.00	-100.00
		Доп. шестнадцатеричный код	7FFF	8000
		омы	+138.50	+060.60
21	Pt100, α=0.00385 0…+100°C	Технические единицы	+100.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+138.50	+100.00
22	Pt100, α=0.00385 0…+200°C	Технические единицы	+200.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+175.84	+100.00
23	Pt100, α=0.00385 0…+600°C	Технические единицы	+600.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+313.59	+100.00
24	Pt100, α=0.003916 –100…+100°C	Технические единицы	+100.00	-100.00
		% от полного диапазона	+100.00	-100.00
		Доп. шестнадцатеричный код	7FFF	8000
		омы	+139.16	+060.60
25	Pt100, α=0.003916 0…+100°C	Технические единицы	+100.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+139.16	+100.00
26	Pt100, α=0.003916 0…+200°C	Технические единицы	+200.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+177.13	+100.00
27	Pt100, α=0.003916 0…+600°C	Технические единицы	+600.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+317.28	+100.00
28	Ni 120, -80…+100°C	Технические единицы	+100.00	-080.00
		% от полного диапазона	+100.00	-080.00
		Доп. шестнадцатеричный код	7FFF	999A
		омы	+200.64	+066.60

Окончание табл.

29	Ni 120, 0…+100°C	Технические единицы	+100.00	+000.00
		% от полного диапазона	+100.00	+000.00
		Доп. шестнадцатеричный код	7FFF	0000
		омы	+200.64	+120.60
2A	Pt1000, α=0.00385 -200…+600°C	Технические единицы	+600.00	-200.00
		% от полного диапазона	+100.00	-033.33
		Доп. шестнадцатеричный код	7FFF	AAAA
		омы	+3137.1	+185.20

ПД – полный диапазон

Значения при выходе за границы диапазона измерения

Формат данных	Выход за верхнюю границу	Выход за нижнюю границу
Технические единицы	+9999	-0000
% от полного диапазона	+9999	-0000
Доп. шестнадцатеричный код	7FFF	8000

Команды модуля

Команда	Ответное сообщение	Описание	Раздел
%AANNTTCCFF	!AA	Настроить параметры конфигурации модуля	Раздел П.82
#**	Не передается	Команда синхронизированной выборки	Раздел П.83
#AA	>(Данные)	Считать значение сигнала на аналоговом входе	Раздел П.84
#AAN	>(Данные)	Считать значение сигнала по каналу «N» аналогового ввода	Раздел П.85
$AA0	!AA	Выполнить калибровку диапазона	Раздел П.86
$AA1	!AA	Выполнить калибровку нуля	Раздел П.87
$AA2	!AANNTTCCFF	Считать параметры конфигурации модуля	Раздел П.88
$AA4	>AAS(Данные)	Считать синхронизированные данные	Раздел П.89
~AAEV	!AA	Разрешить/Запретить выполнение калибровки	Раздел П.90

П.82 %AANNTTCCFF

Назначение команды: Настроить параметры конфигурации модуля

Формат команды: %AANNTTCCFF[CHK](cr)

Для изменения значений скорости передачи или контрольной суммы необходимо замкнуть контакт INIT* на землю.

Ответное сообщение: Допустимая команда: !AA[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)

В случае попытки изменения настроек скорости передачи или контрольной суммы при незамкнутом на землю контакте INIT* модуль выдаст ответное сообщение о недопустимой команде **?**.

Пример:

Команда: %0102050600 Ответное сообщение: !02

Изменяется адрес модуля с «01» на «02». Успешное выполнение.

Команда: %0202050602 Ответное сообщение: !02

Изменяется параметр формата данных с «00» на «02». Успешное выполнение.

См. также команды: Раздел П.88 Команда $AA2

П.83 #**

Назначение команды: Команда синхронизированной выборки

Формат команды: #**[CHK](cr)

символ разделителя.

** команда синхронизированной выборки. Все модули, поддерживающие данную команду, немедленно считывают значение сигнала на аналоговом входе и сохраняют его во внутреннем регистре. Эти данные могут быть считаны из каждого модуля командой $AA4.

Ответное сообщение: Ответное сообщение не передается

Пример:

Команда: #** Ответное сообщение: Не передается

Передается команда синхронизированной выборки.

Команда: $014 Ответное сообщение: >011+025.123

Первое чтение. Получен статус синхронизированных данных = 1.

Команда: $014 Ответное сообщение: >010+025.123

Повторное чтение. Получен статус синхронизированных данных = 0.

См. также команды: Раздел П.89 Команда $AA4

Примечание: **Данная команда является допустимой только для модуля I-7013**

П.84 #AA

Назначение команды: Считать значение сигнала на аналоговом входе.

Формат команды: #AA[CHK](cr)

Ответное сообщение: Допустимая команда: >(Данные)[CHK](cr)

(Данные) значение сигнала на аналоговом входе. Формат данных определяется параметром TT команды конфигурации. В случае передачи команды #AA для модулей I-7033 эти данные представляют собой комбинацию значений для каждого из каналов аналогового ввода соответственно.
Пример:
Команда: #01 Ответное сообщение: >+026.35
Считывается значение сигнала на аналоговом входе модуля с адресом 01. Успешное выполнение.
Команда: #02 Ответное сообщение: >4C53
Считывается значение сигнала на аналоговом входе модуля с адресом 02. Успешно получены требуемые данные в шестнадцатеричном формате.
Команда: #03 Ответное сообщение: >-0000
Считывается значение сигнала на аналоговом входе модуля с адресом 03. Входное значение находится за пределами нижней границы диапазона измерений.
Команда: #04
Ответное сообщение: >+025.12+054.12+150.12
По адресу 04 находится модуль I-7033. В результате считывания информации с этого модуля получены данные о значениях сигналов по всем 3 каналам аналогового ввода.
См. также команды: Раздел П.82 Команда %AANNTTCCFF, Раздел П.88 Команда $AA2

П.85 #AAN
Назначение команды: Считать значение сигнала по каналу «N» аналогового ввода
Формат команды: #AAN[CHK](cr)
N - номер канала, по которому считывается значение аналогового сигнала (от 0 до 7).
Ответное сообщение: Допустимая команда: (Данные)[CHK](cr)
Недопустимая команда: ?AA[CHK](cr)
(Данные) значение сигнала на аналоговом входе. Формат данных определяется полем TT команды конфигурации.
Пример:
Команда: #032 Ответное сообщение: >+025.13
Считывается значение аналогового сигнала по каналу 2 модуля с адресом 03. Данные получены успешно.
Команда: #024 Ответное сообщение: ?02
При считывании значения аналогового сигнала по каналу 4 модуля с адресом 02 принято ответное сообщение о недопустимой команде (ошибка в номере канала).

См. также команды: Раздел П.82 Команда %AANNTTCCFF, Раздел П.88 Команда $AA2
Примечание: **Данная команда является допустимой только для модулей I-7033.**

П.86 $AA0
Назначение команды: Выполнить калибровку диапазона
Формат команды: $AA0[CHK](cr)
0 - команда выполнения калибровки диапазона.
Ответное сообщение:Допустимая команда: !AA[CHK](cr)
Недопустимая команда:?AA[CHK](cr)
Пример:
Команда: $010 Ответное сообщение: !01
Выполняется калибровка диапазона аналогового ввода модуля с адресом 01. Успешное выполнение.
Команда: $020 Ответное сообщение: ?02
При попытке выполнения калибровки диапазона аналогового ввода модуля с адресом 02 принято ответное сообщение о недопустимой команде, так как перед тем, как подать команду калибровки, необходимо разрешить выполнение такой операции.
См. также команды: Раздел П.87 Команда $AA1, Раздел П.90 Команда ~AAEV.

П.87 $AA1
Назначение команды: Выполнить калибровку нуля
Формат команды: $AA1[CHK](cr)
1 - команда на выполнение калибровки нуля.
Ответное сообщение:Допустимая команда: !AA[CHK](cr)
Недопустимая команда:?AA[CHK](cr)
В случае синтаксической или коммуникационной ошибки может быть не принято никакого ответного сообщения.
Пример:
Команда: $011 Ответное сообщение: !01
Выполняется калибровка нуля аналогового ввода модуля с адресом 01. Успешное выполнение.
Команда: $021 Ответное сообщение: ?02
При попытке выполнения калибровки нуля аналогового ввода модуля с адресом 02 принято ответное сообщение о недопустимой команде, так как перед тем, как подать команду калибровки, необходимо разрешить выполнение такой операции.
См. также команды: Раздел П.86 Команда $AA0, Раздел П.90 Команда ~AAEV.

П.88 $AA2

Назначение команды: Считать параметры конфигурации модуля.

Формат команды: $AA2[CHK](cr)

2 - команда считывания параметров конфигурации.

Ответное сообщение: Допустимая команда: !AATTCCFF[CHK](cr); Недопустимая команда: ?AA[CHK](cr).

Пример:

Команда: $012 Ответное сообщение: !01200600

Считываются параметры конфигурации модуля с адресом 01. Успешное выполнение.

Команда: $022 Ответное сообщение: !02230602

Считываются параметры конфигурации модуля с адресом 02. Успешное выполнение.

См. также команды: Раздел П.82 Команда %AANNTTCCFF.

П.89 $AA4

Назначение команды: Считать синхронизированные данные.

Формат команды: $AA4[CHK](cr)

4 - команда считывания синхронизированных данных, сохраненных во внутреннем регистре модуля командой #**.

Ответное сообщение: Допустимая команда: >AAS(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).

S – статус синхронизированных данных: 1 = первое чтение; 0 = данные уже считывались.

(Данные) синхронизированные данные. Формат данных определяется параметром TT команды конфигурации.

Пример:

Команда: $014 Ответное сообщение: ?01

При попытке считывания синхронизированных данных в модуле с адресом 01 принимается ответное сообщение о том, что таковые данные отсутствуют.

Команда: #** Ответное сообщение: Не передается.

Передается команда на выполнение синхронизированной выборки.

Команда: $014 Ответное сообщение: >011+025.56

Считываются синхронизированные данные в модуле с адресом 01. В ответном сообщении содержится значение «1» статуса синхронизированных данных и собственно данные.

Команда: $014 Ответное сообщение: >010+025.56

Считываются синхронизированные данные в модуле с адресом 01. В ответном сообщении содержится значение «0» статуса синхронизированных данных и собственно данные.

См. также команды: Раздел П.83 Команда #**
Примечание: **Данная команда является допустимой только для модулей I-7013.**

П.90 ~AAEV
Назначение команды: Разрешить или запретить выполнение калибровки.
Формат команды: ~AAEV[CHK](cr)
E - команда разрешения или запрещения выполнения калибровки
V 1 = разрешить калибровку; 0 = запретить калибровку
Ответное сообщение:Допустимая команда: !AA[CHK](cr)
Недопустимая команда:?AA[CHK](cr)
Пример:
Команда: $010 Ответное сообщение: ?01
При попытке выполнения калибровки диапазона аналогового ввода модуля с адресом 01 принято ответное сообщение о том, что данная команда является недопустимой, пока калибровка не будет разрешена.
Команда: ~01E1 Ответное сообщение: !01
Разрешается выполнение калибровки модуля с адресом 01. Успешное выполнение.
Команда: $010 Ответное сообщение: !01
Выполняется калибровка диапазона аналогового ввода модуля с адресом 01. Успешное выполнение.
См. также команды: Раздел П.86 Команда $AA0, Раздел П.87 Команда $AA1.

ПРИЛОЖЕНИЕ 7

ОБЩИЕ КОМАНДЫ МОДУЛЕЙ

Команда	Ответное сообщение	Описание	Раздел
$AAF	!AA(Данные)	Считать номер версии микропрограммного обеспечения	Раздел П.91
$AAM	!AA(Данные)	Запросить название модуля	Раздел П.92
~AAO(Данные)	!AA	Присвоить модулю название	Раздел П.93
~**	Не передается	Главный ПК работает нормально	Раздел П.94
~AA0	!AASS	Считать статус модуля	Раздел П.95
~AA1	!AA	Произвести сброс статуса модуля	Раздел П.96
~AA2	!AAVV	Считать значение временного интервала сторожевого таймера главного ПК	Раздел П.97
~AA3EVV	!AA	Задать значение временного интервала сторожевого таймера главного ПК	Раздел П.98

П.91 $AAF

Назначение команды: Считать номер версии микропрограммного обеспечения

Формат команды: $AAF[CHK](cr)

F - команда чтения номера версии микропрограммного обеспечения.

Ответное сообщение: Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).

(Данные) номер версии микропрограммного обеспечения данного модуля.

Пример:

Команда: $01F Ответное сообщение: !01A2.0

При запросе версии микропрограммного обеспечения модуля с адресом 01 получен номер версии A2.0.

Команда: $02F Ответное сообщение: !02B1.1

При запросе версии микропрограммного обеспечения модуля с адресом 02 получен номер версии B1.1.

П.92 $AAM
Назначение команды: Запросить название модуля.
Формат команды: $AAM[CHK](cr)
M - команда считывания названия модуля.
Ответное сообщение: Допустимая команда: !AA(Данные)[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
(Данные) название модуля.
Пример:
Команда: $01M Ответное сообщение: !017021
При запросе названия модуля с адресом 01 получен ответ: 7021.
Команда: $03M Ответное сообщение: !037021P
При запросе названия модуля с адресом 03 получен ответ: 7021P.

П.93 ~AAO(Данные)
Назначение команды: Присвоить модулю название
Формат команды: ~AAO(Данные)[CHK](cr)
O - команда присвоения модулю названия.
(Данные) новое имя модуля длиной до 6 символов.
Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
Пример:
Команда: ~01O7021 Ответное сообщение: !01
Присвоить модулю с адресом 01 название «7021». Успешное выполнение.
Команда: $01M Ответное сообщение: !017021
При считывании названия модуля с адресом 01 получено ответное сообщение: 7021.
См. также команды: Раздел П.92 Команда $AAM.

П.94 ~**
Назначение команды: Главный ПК работает нормально.
Главный ПК (или контроллер), к которому подключены модули, передает эту команду для того, чтобы сообщить всем модулям информацию: «Главный ПК работает нормально».
Формат команды: ~**[CHK](cr)
** - команда для всех модулей.
Ответное сообщение: Не передается.
Пример:
Команда: ~** Ответное сообщение: Не передается.
На все модули передается информация о том, что главный ПК работает нормально.
См. также команды: Раздел П.95 Команда ~AA0, Раздел П.96 Команда ~AA1, Раздел П.97 Команда ~AA2, Раздел П.98 Команда ~AA3EVV.

П.95~AA0

Назначение команды: Считать статус модуля

Формат команды: ~AA0[CHK](cr)

0 - команда считывания статуса модуля.

Ответное сообщение: Допустимая команда: !AASS[CHK](cr); Недопустимая команда: ?AA[CHK](cr).

SS - статус модуля. Значение статуса модуля заносится в ЭСППЗУ и может быть сброшено только при помощи команды ~AA1.

7	6 5 4 3	2	1 0
*1	Зарезервировано	*2	Зарезервировано

*1: Статус сторожевого таймера главного ПК: 0= Выключен; 1= Включен.

*2: Флаг срабатывания сторожевого таймера главного ПК: 0= Снят; 1= Установлен.

Пример:

Команда: ~010 Ответное сообщение: !0100

При считывании статуса модуля с адресом 01 принято значение «00», свидетельствующее о том, что флаг срабатывания сторожевого таймера главного ПК снят.

Команда: ~010 Ответное сообщение: !0104

При считывании статуса модуля с адресом 01 принято значение «04», свидетельствующее о том, что в модуле установлен флаг срабатывания сторожевого таймера главного ПК.

См. также команды: Раздел П.94 Команда ~**, Раздел П.96 Команда ~AA1, Раздел П.97 Команда ~AA2, Раздел П.98 Команда ~AA3EVV.

П.96 ~AA1

Назначение команды: Произвести сброс статуса модуля.

Формат команды: ~AA1[CHK](cr)

1 - команда сброса статуса модуля.

Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).

Пример:

Команда: ~010 Ответное сообщение: !0104

При считывании статуса модуля с адресом 01 принято значение «04», свидетельствующее о том, что в модуле установлен флаг срабатывания сторожевого таймера главного ПК.

Команда: ~011 Ответное сообщение: !01

Модуль с адресом 01 приводится в исходное состояние. Успешное выполнение. Светодиодный индикатор на этом модуле перестает мигать.

Команда: ~010 Ответное сообщение: !0100
При считывании статуса модуля с адресом 01 принято значение «00», свидетельствующее о том, что флаг срабатывания сторожевого таймера главного ПК снят.
См. также команды: Раздел П.94 Команда ~**, Раздел П.95 Команда ~AA0, Раздел П.97 Команда ~AA2, Раздел П.98 Команда ~AA3EVV.

П.97 ~AA2
Назначение команды: Считать значение временного интервала сторожевого таймера главного ПК.
Формат команды: ~AA2[CHK](cr)
2 - команда считывания значения временного интервала сторожевого таймера главного ПК.
Ответное сообщение: Допустимая команда: !AAEVV[CHK](cr); Недопустимая команда: ?AA[CHK](cr).
E - статус сторожевого таймера главного ПК: 0= Выключен; 1= Включен.
VV - значение временного интервала в шестнадцатеричном формате, каждая единица которого соответствует длительности 0,1 секунды (01 = 0,1 с, а FF = 25,5 с).
Пример:
Команда: ~012 Ответное сообщение: !010FF
При считывании длительности временного интервала сторожевого таймера главного ПК в модуле с адресом 01 принято значение «FF», что соответствует длительности 25,5 с, сторожевой таймер выключен.
См. также команды: Раздел П.94 Команда ~**, Раздел П.95 Команда ~AA0, Раздел П.96 Команда ~AA1, Раздел П.97 Команда ~AA2, Раздел П.98 Команда ~AA3EVV.

П.98 ~AA3EVV
Назначение команды: Задать значение временного интервала сторожевого таймера главного ПК.
Формат команды: ~AA3EVV[CHK](cr)
3 - команда установки временного интервала сторожевого таймера главного ПК.
E 1 = Включить сторожевой таймер главного ПК; 0 = Отключить сторожевой таймер главного ПК.
VV - значение временного интервала от 01 до FF, каждая единица которого соответствует длительности 0.1 секунды.
Ответное сообщение: Допустимая команда: !AA[CHK](cr); Недопустимая команда: ?AA[CHK](cr).

Пример:
Команда: ~010 Ответное сообщение: !0100
При считывании статуса модуля с адресом 01 принято значение «00», свидетельствующее о том, что флаг срабатывания сторожевого таймера главного ПК снят.
Команда: ~013164 Ответное сообщение: !01
Включается сторожевой таймер главного ПК в модуле с адресом 01 и для него устанавливается значение «64» (10.0 секунд). Успешное выполнение.
Команда: ~012 Ответное сообщение: !01164
При считывании длительности временного интервала сторожевого таймера главного ПК в модуле с адресом 01 принято значение «64», что соответствует длительности 10,0 с, сторожевой таймер включен.
Команда: ~** Ответное сообщение: Не передается.
Происходит сброс сторожевого таймера главного ПК. Выдержите паузу длительностью 10 с и не подавайте команду ~**.
После этого на модуле начнет мигать светодиодный индикатор с частотой примерно 1 раз в секунду.
Команда: ~010 Ответное сообщение: !0104
При считывании статуса модуля с адресом 01 принято значение «04», свидетельствующее о том, что в модуле установлен флаг срабатывания сторожевого таймера главного ПК.
Команда: ~011 Ответное сообщение: !01
Модуль с адресом 01 приводится в исходное состояние. Успешное выполнение. Светодиодный индикатор на этом модуле перестает мигать.
См. также команды: Раздел П.94 Команда ~**, Раздел П.95 Команда ~AA0, Раздел П.96 Команда ~AA1, Раздел П.97 Команда ~AA2.