ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

Кафедра «Радиоэлектроника»

Учебно-методическое пособие

«Моделирование и анализ процессов обмена данными по промышленным протоколам в телекоммуникационных системах»
по дисциплине

«Компьютерные технологии в телекоммуникационных системах»

Авторы
Назарова О. Ю.,
Бурнашев И. Я.

Ростов-на-Дону, 2019

Аннотация

Учебно-методическое пособие предназначено для студентов очной, заочной форм обучения направления 11.04.02 «Инфокоммуникационные технологии и системы связи».

Предназначено для успешного освоения теоретической части дисциплины, а также для эффективного выполнения лабораторных работ. Приведены теоретические сведения и методика моделирования процесса обмена данными в телекоммуникационных системах на основе современных компьютерных технологий и алгоритмов.

Авторы

к.т.н., доцент кафедры «Радиоэлектроника» Назарова О.Ю.,
к.т.н., доцент кафедры «Радиоэлектроника» Бурнашев И.Я.

Оглавление

ВВЕДЕНИЕ

Появление новых технологий, как правило, охватывает широкий спектр областей, где возможно их практическое использование. Это происходит в машиностроении, строительстве, медицине, разных направлениях производственной сферы и т. д. Однако не каждая область выступает двигателем прогресса и стимулирует к переходу на новый этап развития. В этом смысле компьютерные технологии можно рассматривать как универсальный инструмент генерации новых идей, которые в дальнейшем переходят и в другие сферы применения.

Развитие технологической базы подразумевает совершенствование цифровых и аналоговых систем, обеспечивающих взаимодействие посредством сетей связи. В этом процессе основная роль отводится компьютерным системам и каналам связи. В связи с этим появляется необходимость изучать телекоммуникационные системы с точки зрения применения в них компьютерных технологий. Современная телекоммуникационная инфраструктура характеризуется высоким уровнем надежности, безопасности и скорости передачи данных.

Реализация телекоммуникационных задач предусматривает использование нескольких категорий средств. В частности, основу инфраструктуры составляют аппаратные инструменты, среди которых – каналы связи и компьютерные узлы. Невозможны телекоммуникационные системы без упомянутых каналов связи. Это обширная группа средств, в которую входят технические системы, реализующие электрическую, оптоволоконную, телефонную и радиосвязь – выбор конкурентного вида канала определяется требованиями к телекоммуникационному проекту.

Внедрение новых технологий и аппаратных средств в учебный процесс, как показывают исследования, способствует развитию творческого воображения и мыслительных процессов учащихся.

1. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ CAN

1.1. Общие сведения о протоколе CAN

CAN (Controller Area Network) протокол был разработан компанией Robert Bosch GmbH в середине 1980-х и в настоящее время широко распространен в промышленной автоматизации, технологиях «умного дома», автомобильной промышленности и многих

других областях. Различные сообщения , передающиеся по сети, имеют идентификатор. Каждая станция, основываясь на этом идентификаторе, решает, получать это сообщение или нет. Этот идентификатор определен в поле идентификатора CAN фрейма. При этом адрес приемника устанавливается в самом приемнике путем настройки входных фильтров соответствующих микросхем.

CAN контроллеры соединяются с помощью дифференциальной шины, имеющей две линии – CAN_H (can-high) и CAN_L (can-low), по которым передаются сигналы. Логический ноль регистрируется, когда на линии CAN_H сигнал выше, чем на линии CAN_L. Логическая единица – в случае, когда сигналы CAN_H и CAN_L одинаковы (отличаются менее чем на 0.5 В). Использование такой дифференциальной схемы передачи делает возможным работу CAN сети в очень сложных внешних условиях. Логический ноль называется доминантным битом, а логическая единица – рецессивным. Эти названия отражают приоритет логической единицы и нуля на шине CAN. При одновременной передаче в шину логического нуля и единицы на шине будет зарегистрирован только логический ноль (доминантный сигнал), а логическая единица будет подавлена (рецессивный сигнал).

CAN протокол состоит из следующих уровней:

1. Объектный уровень обеспечивает фильтрацию сообщений и обработку сообщений.

2. Транспортный уровень представляет собой ядро CAN протокола. Он отвечает за синхронизацию, арбитраж, доступ к шине, разделение посылок на фреймы, определение и передачу ошибок и минимизацию неисправностей.

3. Физический уровень определяет, как именно будут передаваться сигналы, их электрические уровни и скорость передачи. Типичные значения при напряжении питания +5В приведены на рис. 1.1, причем доминирующим уровнем является нижний уровень, а рецессивным, соответственно, верхний.

Рис. 1.1. Сигнальные уровни на шине

Максимальное расстояние между узлами составляет до 1 км. Скорость обмена – до 1 Мбит/с при длине линии 60 м. Возможность применения гальванической развязки, причем гальваническая развязка может устанавливаться либо между приемопередающим буфером и микросхемой, обеспечивающей функции CAN, либо между микросхемой и остальной системой.

В CAN протоколе определены следующие типы фреймов:

• фрейм данных **(Data Frame)** перемещает данные с передатчика на приемник (приемники);

• удаленный фрейм **(Remote Frame)** запрашивает передачу фрейма данных,связанного с определенным идентификатором;

• фрейм ошибки **(Error Frame)** выражает, какой узел обнаружил ошибку шины/сети;

• фрейм перегрузки **(Overload Frame)** обеспечивает задержку между передачей фреймов, чтобы управлять потоком данных.

Фрейм данных

Рассмотрим подробнее фрейм данных (рис. 1.2).

Рис. 1.2. Стандартный фрейм

Стандартный фрейм состоит из следующих полей:

- **Начало фрейма (Start of Frame (SOF))**. Поле SOF находится в начале фрейма данных и удаленного фрейма и содержит один доминирующий бит.

- **Поле арбитража (Arbitration Field)**. Поле арбитража содержит 11-битовый идентификатор и RTR-бит, показывающий, является данный фрейм фреймом данных или удаленным фреймом. Для фрейма данных бит RTR всегда выставлен в логический ноль (доминантный сигнал). Идентификатор предназначен для адресации сообщений, используется механизмом арбитража. Для стандарта CAN-2.0A используется 11-битный идентификатор, а для стандарта CAN-2.0B 29-битный идентификатор.

- **Управляющее поле (Control Field)**. Управляющее поле (рис. 1.3) содержит 6 битов, из которых 4 бита (DLC0 – DLC4) составляют поле Data Length Code, показывающее количество байтов данных, которое будет передаваться в поле данных; два других бита зарезервированы для следующих редакций протокола.

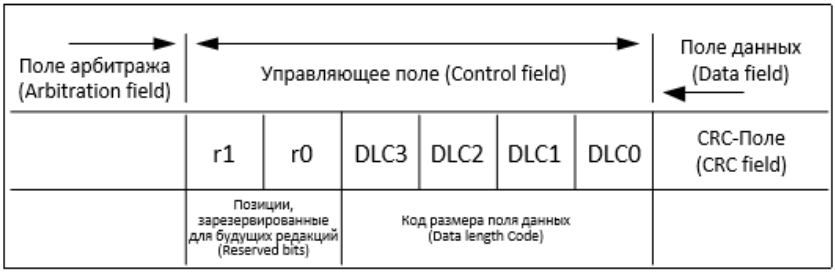

Рис. 1.3. Управляющее поле фрейма данных

Кодирование числа байтов данных по коду размера поля данных показано в виде табл. 1.1.

Таблица 1.1

Кодирование числа байтов данных

Число байтов данных	Код размера поля данных			
	DCL3	DCL2	DCL1	DCL0
0	дом.	дом.	дом.	дом.
1	дом.	дом.	дом.	рец.
2	дом.	дом.	рец.	дом.
3	дом.	дом.	рец.	рец.
4	дом.	рец.	дом.	дом.
5	дом.	рец.	дом.	рец.
6	дом.	рец.	рец.	дом.
7	дом.	рец.	рец.	рец.
8	рец.	дом.	дом.	дом.

– **Поле данных (Data Field)**. Поле данных содержит передаваемые данные, причем количество передаваемых байтов указывается в поле Control Field и не может превышать 8.

– **Поле контрольной суммы (Cyclic redundancy check (CRC))**. Поле CRC (рис. 1.4) содержит циклический избыточный код, служащий для обнаружения ошибок во всех предшествующих ему полях фрейма, включая бит начала фрейма. Каждый принимающий узел подсчитывает значение CRC для каждого полученного сообщения. Если подсчитанное значение CRC суммы не совпадает со значением CRC в теле сообщения, принимающий узел генерирует ошибку CRC Error.

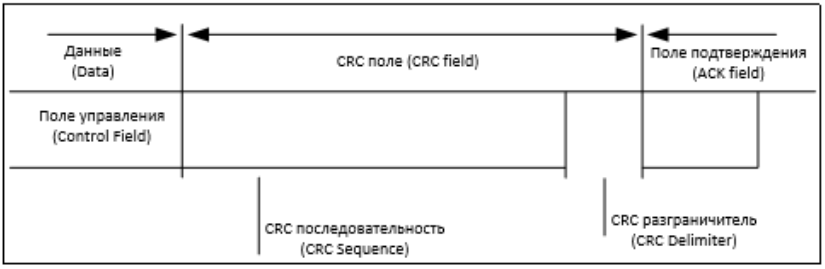

Рис. 1.4. Поле контрольной суммы фрейма данных

Для вычисления CRC полинома, полином, коэффициенты которого задаются потоком, состоящим из значений бит полей: стартового поля, поля арбитража, управляющего поля, поля данных (если имеется) (самые младшие 15 коэффициентов полинома=0), должен быть разделен на полином следующего вида:

$$x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$$

Остаток этого полиномиального деления и есть CRC-последовательность, передаваемая по шине. Поле CRC оканчивается CRC-разделителем (1 рецессивный бит).

– Поле подтверждения (ACK Field). Поле подтверждения (рис. 1.5) содержит участки ACK Slot и ACK Delimiter и выполняет следующую функцию: передающий узел посылает по одному рецессивному биту на каждом из участков, а приемник, если он принял сообщение без сбоев, устанавливает на линии доминирующий бит в поле ACK Slot. При наложении рецессивного и доминирующего уровней на линии устанавливается доминирующий, и это событие сигнализирует передающему узлу о том, что передача прошла нормально и повтор не требуется. Второй бит (ACK Delimiter) этого поля всегда является рецессивным.

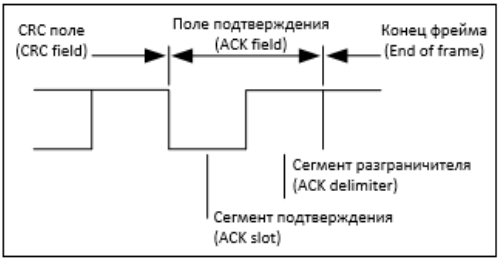

Рис. 1.5. Поле подтверждения фрейма данных

– Поле конца фрейма (End of Frame (EOF)). Поле конца фрейма содержится в фрейме данных и удаленном фрейме и состоит из семи рецессивных битов.

Удаленный фрейм

Удаленный фрейм аналогичен по структуре фрейму данных, но не имеет поля данных. Удаленный фрейм состоит из стартового поля, поля арбитража, управляющего поля, поля контрольной суммы, поля подтверждения и поля конца фрейма (рис. 1.6). В случае удаленного фрейма RTR бит является рецессивом. Полярность RTR бита указывает, является ли переданный фрейм фреймом данных (доминирующий RTR бит) или удаленным фреймом (рецессивный RTR бит).

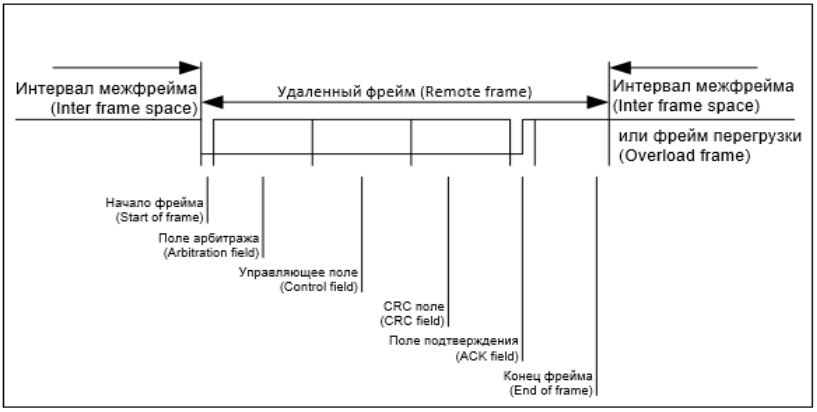

Рис. 1.6. Удаленный фрейм

Фрейм ошибки

Фрейм ошибок используется любым принимающим узлом, чтобы сообщить всем участникам сети о том, что передаваемое в

данный момент по сети сообщение содержит ошибку. Сообщение об ошибке имеет наивысший в системе приоритет, поэтому передается сразу после обнаружения ошибки и принимается всеми устройствами одновременно. Все устройства также одновременно удаляют из своей памяти сообщение, содержащее ошибку.

Фрейм ошибки состоит из двух различных полей (рис. 1.7). Первое поле задается суперпозицией флагов ошибки (Error Flags) (6 доминирующих битов при активном флаге ошибок/6 рецессивных битов при пассивном флаге ошибок), внесенных двумя различными станциями. Второе поле – поле разграничителя (Error Delimiter) (8 рецессивных бита).

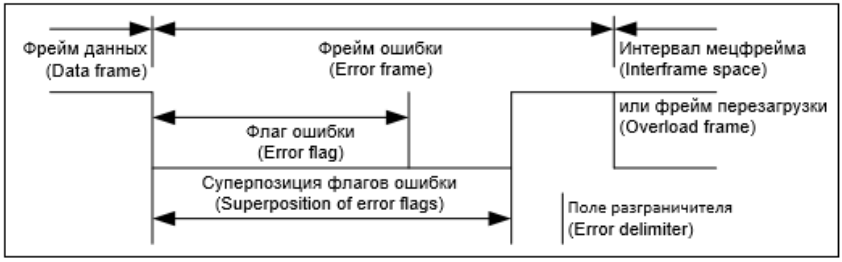

Рис. 1.7. Фрейм ошибки

Фрейм перегрузки

Фрейм перегрузки состоит из двух полей: флага перегрузки и поля разделителя. Существуют следующие условия, при наступлении которых начинается передача фрейма перегрузки:

– перегрузка приемника, которая требует увеличить паузу между принимаемыми им фреймами;

– обнаружение доминантного бита на месте первого и второго бита в поле перерыва паузы между фреймами.

Между фреймами данных, фреймом вызова и любыми другими фреймами устанавливается пауза (Interframe Space (IFS)). В отличие от этого, перед фреймами перегрузки и ошибок паузы нет, это ускоряет их доставку.

Пауза содержит поле перерыва (3 рецессивных бита) и поле простоя (произвольной длины) и, для пассивных к ошибке устройств, которые выполняли передачу предыдущего сообщения, поле приостановленной передачи.

Обнаружение ошибок в протоколе CAN

CAN протокол определяет пять способов обнаружения ошибок в сети:

– Текущий контроль логического уровня битов (Bit monitoring).

– Контроль заполнения битов (Bit stuffing).

– Контроль передаваемого поля битов (Frame check).

– Контроль сигнала «Подтверждение приема» (ACKnowledge-ment Check).

– Циклический контроль по избыточности (CRC Check).

Текущий контроль логического уровня битов (Bit monitoring) – каждый узел во время передачи битов в сеть сравнивает значение передаваемого им бита со значением бита, которое появляется на шине. Если эти значения не совпадают, то узел генерирует ошибку Bit Error. Естественно, что во время арбитража на шине (передача поля арбитража в шину) этот механизм проверки ошибок отключается.

Контроль заполнения битов (Bit stuffing) – После передачи пяти последовательных битов с одинаковым уровнем передатчик автоматически вводит в разрядный поток бит противоположной полярности. Приемники сообщения автоматически удаляют такие биты перед обработкой сообщения. Если обнаруживается шестой бит одинаковой полярности, то помечается ошибка заполнения битов. В случае если обнаружена ошибка, узел, обнаруживший ошибку, прерывает передачу посылкой флага ошибки. При этом передатчик автоматически производит повторную передачу сообщения, что предохраняет все узлы от возникновения ошибок и гарантирует непротиворечивость данных в сети.

Контроль передаваемого поля битов (Frame check) – некоторые части CAN-сообщения имеют одинаковое значение во всех типах сообщений, т.е. протокол CAN точно определяет, какие уровни напряжения и когда должны появляться на шине. Если формат сообщений нарушается, то узлы генерируют ошибку Form Error.

Контроль сигнала «Подтверждение приема» (ACKnowledgement Check) – каждый узел, получив правильное сообщение по сети, посылает в сеть доминантный (0) бит. Если же этого не происходит, то передающий узел регистрирует ошибку Acknowledgement Error.

Циклический контроль по избыточности (CRC Check) – каждое сообщение CAN содержит CRC сумму, и каждый принимающий узел подсчитывает значение CRC для каждого полученного сообщения. Если подсчитанное значение CRC суммы не совпадает со

значением CRC в теле сообщения, принимающий узел генерирует ошибку CRC Error.

Контроль доступа (побитовый арбитраж). Передающий узел в CAN протоколе слышат все другие узлы в сети и подтверждают это. Всякий раз, когда шина свободна от передачи, узел может начинать передавать. Если узел передает, эта передача должна быть завершена прежде, чем другой узел может пытаться передавать. Если два или больше узла начинают передавать в одно и то же время, конфликт решается при помощи неразрушающего (non-destructive) поразрядного алгоритма арбитража, использующего поле арбитража. 11-битовое поле идентификатора передается от старшего к младшему значащему биту. Доминирующий уровень – логический 0. Одновременная передача бита с доминирующим уровнем (логический 0) и бита с рецессивным уровнем (логическая 1) дает в результате уровень логического 0.

В течение передачи поля арбитража каждый передатчик контролирует текущий уровень на шине и сравнивает это с битом, который он должен передавать. Если значения равны, узел способен затем продолжить передачу. Если бит с пассивным уровнем (логическая 1) был передан, а активный бит (логический 0) обнаружен на шине, то данный узел теряет право передачи и должен прекратить передачу последующих данных (рис. 1.8). Узел, который потерял шину, может сделать попытку передачи снова, когда текущая передача завершена.

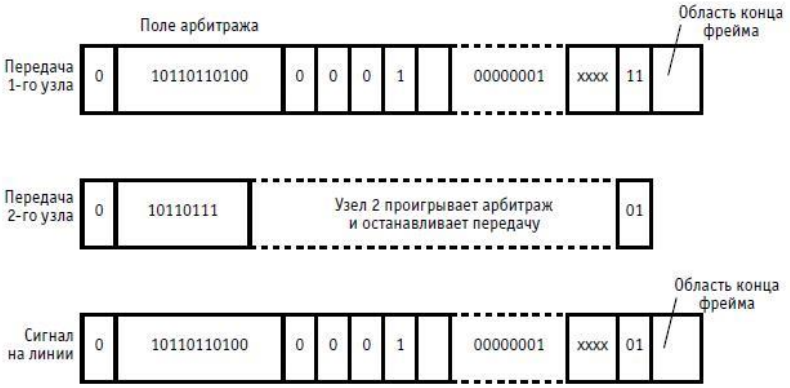

Рис. 1.8. Пример поразрядного арбитража

Из сказанного можно сделать следующие выводы.

Приоритетным является не передающий или приемный узел, а сообщение, имеющее меньшее значение идентификатора. Если в

сети один из узлов (сервер) будет ответственным за принятие решений, то он должен иметь наименьший адрес из задействованных.

Вторая возможность, которую дает механизм арбитража, использована в сети верхнего уровня DeviceNet. В этой сети количество узлов ограничено 64 и для адресации отведены младшие разряды идентификатора, а старшие разряды предназначены для кодирования видов сообщений. Естественно, что сообщение, имеющее 0 в старшем бите, захватит шину первым, независимо от адреса узла приемника. Это, в свою очередь, обеспечивает передачу сообщений первого вида, например об аварии, по сети первыми, независимо от адресов приемных и передающих узлов.

1.2. Моделирование передачи данных по протоколу CAN

Данные передаются по 8-битному формату. Биты чисел должны быть отправлены, начиная со старшего разряда (MSB).

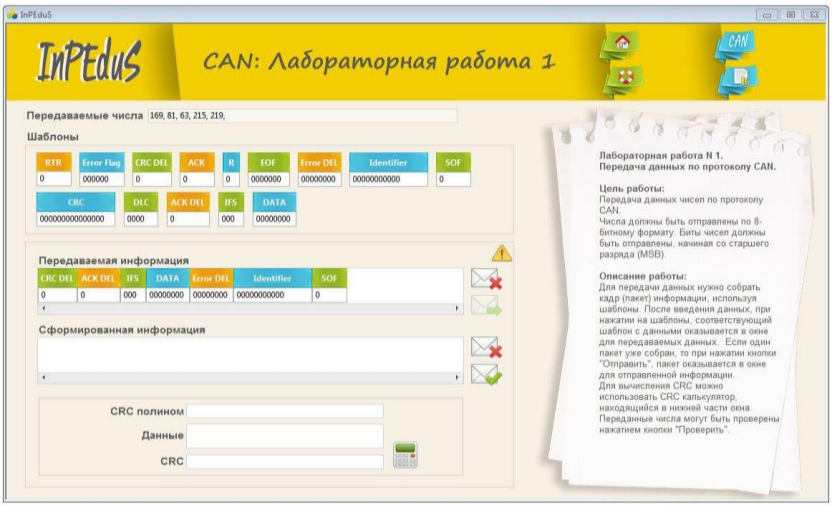

Рис. 1.9. Рабочее окно

Программа автоматически генерирует числа, которые необходимо отправить. Эти числа выводятся в поле Передаваемые числа в верхней части окна.

В правой части окна представлена цель лабораторной работы, а также краткое описание по выполнению (рис.1.9).

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них, после чего шаблон с данными окажется в поле **Передаваемая информация**. Нажмите кнопку **«Отправить».**

Индикатор ошибки загорится красным светом в случае ошибки при сборе кадра. Нажмите кнопку **«Очистить»** и соберите кадр заново. Вы можете отправить сразу все числа в одном кадре, либо отправить каждое число в отдельном кадре.

Для вычисления CRC используйте калькулятор в нижней части окна. Запишите полином в поле **Полином CRC**. Заполните поле Данные, после чего нажмите кнопку **«Вычислить CRC»**.

Если кадр собран правильно, то после нажатия кнопки «Отправить», он переместится в поле **Отправленная информация**. Переданные числа могут быть проверены нажатием кнопки «Проверить». Если числа введены правильно, программа выведет диалоговое окно с надписью «Правильно!». Если числа введены неправильно, программа выведет диалоговое окно с надписью «Неправильно!». В этом случае очистите поля Передаваемая информация и Отправленная информация и повторите необходимые шаги. Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку **«Отчет».**

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена»**.

Если лабораторная работа выполнена правильно, то после нажатия кнопки **«Отчет»** появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите ОК. Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку. Для перехода к меню протокола CAN нажмите кнопку **«CAN»** в верхнем правом углу окна.

1.3. Моделирование запроса данных по протоколу CAN

Для запроса данных необходимо отправить удаленный фрейм с данным идентификатором. Для этого необходимо во вкладке «Протокол CAN» открыть лабораторную работу 2.

Программа автоматически генерирует код идентификатора, который выводятся в поле **Идентификатор** в верхней части окна.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для сбора удаленного фрейма необходимо записать данные в **Шаблоны** и нажать на них**,** после чего шаблон с данными окажется в поле **Передаваемая информация.** Нажмите кнопку «Отправить».

Индикатор ошибки загорится красным светом в случае ошибки при сборе кадра. Нажмите кнопку «Очистить» и соберите кадр заново.

Для вычисления CRC используйте калькулятор в нижней части окна. Запишите полином в поле **Полином CRC.** Заполните поле **Данные**, после чего нажмите кнопку «Вычислить CRC».

Если кадр собран правильно, то после нажатия кнопки «Отправить», передаваемая информация переместится в поле **Отправленная информация,** а в поле **Полученная информация** появится полученный кадр с данными.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола CAN нажмите кнопку «CAN» в верхнем правом углу окна.

1.4. Моделирование проверки полученных данных по протоколу CAN и отправка фрейма ошибки

Полученный пакет данных должен быть проверен, и при обнаружении ошибок активный флаг ошибок должен быть отправлен.

Открыть вкладку лабораторная работа 3 в опции протокол CAN.

В поле **Полученная информация** показан кадр, который необходимо проверить.

В правой части окна представлена цель лабораторной работы, а также краткое описание по выполнению. Для того чтобы проверить кадр на ошибку, вычислите CRC код с помощью калькулятора в нижней части окна. Проверьте, соответствует ли CRC код данным кадра. В случае ошибки необходимо отправить фрейм ошибки.

Для передачи фрейма ошибки в поле Передаваемая информация необходимо правильно собрать начало кадра**.** Для этого нажмите на **Шаблоны**, и соответствующий шаблон с данными окажется в поле **Передаваемая информация.** Индикатор ошибки загорится красным светом в случае ошибки при сборе кадра. Нажмите кнопку «Очистить» и соберите кадр заново.

Если кадр собран правильно, то после нажатия кнопки «Отправить», он переместится в поле **Отправленная информация.**

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью «**Лабораторная работа не закончена».**

Сформировать отчет, сделать выводы об особенностях работы протокола в различных режимах.

Контрольные вопросы

1. Каково назначение протокола?
2. Каков алгоритм сборки кадра передачи?
3. Каков принцип обнаружения ошибок в протоколе CAN?

2. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ SPI

2.1. Общие сведения о протоколе SPI

SPI (Serial Peripheral Bus) **–** интерфейс для последовательного обмена данными между микросхемами, разработанный компанией Motorola, в настоящее время используется в продукции многих производителей. SPI иногда также называют четырехпроходным (four-wire) интерфейсом. Шина SPI организована по принципу master-slave (мастер-подчиненный). В качестве мастера обычно выступает микроконтроллер, а в качестве подчиненного выступают различного рода микросхемы, запоминающие устройства, специализированные контроллеры и т.д.

Протокол SPI правильнее назвать протоколом обмена данными между двумя сдвиговыми регистрами, каждый из которых одновременно выполняет функцию и приемника, и передатчика. Непременным условием передачи данных по шине SPI является генерация сигнала синхронизации шины. Этот сигнал может генерировать только мастер.

При передаче данных по шине SPI участвуют четыре сигнала:

– MOSI или SI – выход мастера, вход подчиненного (Master Out Slave In). Служит для передачи данных от мастера подчиненному.

– MISO или SO – вход мастера, выход подчиненного (Master In Slave Out). Служит для передачи данных от подчиненного мастеру.

– SCLK или SCK – последовательный тактовый сигнал (Serial Clock). Служит для передачи тактового сигнала для подчиненных.

– CS или SS – выбор микросхемы, выбор подчиненного (Chip Select, Slave Select).

Существует три типа подключения к шине SPI. Самое простое подключение, в котором участвуют только две микросхемы, показано на рис. 2.1. Здесь мастер передает данные по линии MOSI синхронно с генерируемым им же сигналом SCLK, а подчиненный получает переданные биты данных по фронтам принятого сигнала синхронизации. Одновременно с этим подчиненный отправляет свою посылку данных. Если ответная передача данных не предусматривается или не требуется, представленную схему можно упростить, исключив линию MISO.

Рис. 2.1. Простейшее подключение
к шине SPI

При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение, либо каскадное (последовательное). В случае каскадного подключения несколько подчиненных устройств могут быть подключены к одной линии SS.

Стандартный алгоритм работы SPI :

1. Мастер устанавливает низкий уровень на той линии SS, к которой подключено подчиненное устройство, с которым устанавливается связь.
2. Мастер задаёт тактовый сигнал, переключая уровень сигнала на SCLK между «0» и «1». Одновременно с каждым изменением уровня сигнала SCLK мастер выставляет нужный уровень сигнала MOSI, передавая подчиненному по одному биту за такт.
3. Подчиненное устройство при каждом изменении уровня SCLK выставляет нужный уровень сигнала MISO, передавая мастеру по одному биту за такт.
4. Для завершения передачи мастер устанавливает высокий уровень на линии SS.

SPI является полнодуплексной шиной – данные передаются одновременно в обе стороны (рис. 2.2). Скорость работы шины обычно находится в пределах 1- 50 МГц. Благодаря исключительной простоте алгоритма передачи SPI получил широчайшее распространение в самых различных электронных устройствах – в датчиках, чипах памяти и т.д.

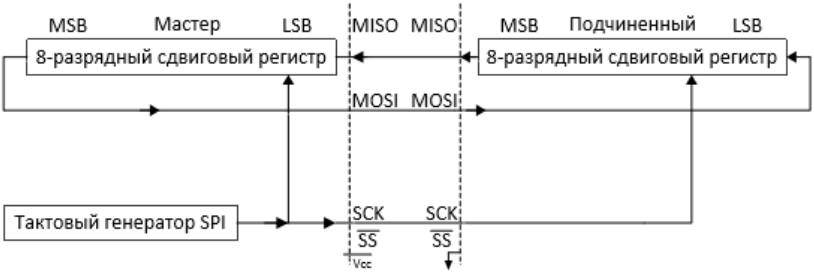

Рис. 2.2. Полнодуплексная передача данных

Последовательный интерфейс SPI использует два регистра для установления параметров передачи данных (частота синхросигнала, режим передачи и т.д.): регистр управления (SPCR) и регистр статуса (SPSR).

Регистры SPI

• SPIE: разрешение прерывания SPI. Если установлен флаг SPIF в регистре статуса SPI, то установка (SPIE=1) данного бита приведет к исполнению процедуры обработки прерывания по SPI.

• SPE: разрешение SPI. Если флаг SPE установлен (SPE=1), то разрешается работа SPI. Данный бит должен быть установлен, если необходимо использовать SPI независимо от того, в каком режиме он будет работать.

• DORD: порядок сдвига данных. Если DORD=1, то при передаче данных первым передается младший разряд (LSB). Если же DORD=0, то первым передается старший разряд (MSB).

• MSTR: выбор ведущего/подчиненного. Если флаг MSTR установлен (MSTR=1), то SPI работает как ведущий (мастер), иначе (MSTR=0) как подчиненный. Если SS настроен как вход и к нему приложен низкий уровень, когда MSTR был равен 1, то бит MSTR автоматически сбрасывается и устанавливается флаг прерывания SPIF в регистре SPSR. Для возобновления ведущего режима SPI пользователь должен предусмотреть программную установку бита MSTR.

• CPOL: полярность синхронизации. Если CPOL=1, то SCK имеет высокий уровень в состоянии ожидания. Если CPOL=0, то SCK имеет низкий уровень в состоянии ожидания.

• CPHA: фаза синхронизации. Значение бита фазы синхронизации (CPHA) определяет, по какому фронту SCK происходит выборка данных: по переднему или заднему.

SPR1, SPR0: биты 1 и 0 выбора частоты синхронизации SPI. Данные биты совместно с флагом SPI2X, который устанавливается

в регистре статуса задают частоту синхронизации на выводе SCK в режиме мастера. SPR1 и SPR0 не оказывают никакого влияния в режиме подчиненного.

• SPIF: флаг прерывания по SPI. Флаг SPIF устанавливается по завершении последовательной передачи. Прерывание генерируется в том случае, если установлен бит SPIE в регистре управления SPI и разрешены общие прерывания. Если SS настроен как вход и к нему приложен низкий уровень, то, если SPI находился в режиме мастера, также установится флаг SPIF. SPIF сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно бит SPIF сбрасывается при первом чтении регистра статуса SPI с установленным флагом SPIF, а также во время доступа к регистру данных SPI (SPDR).

• WCOL: флаг повторной записи. Бит WCOL устанавливается, если выполнена запись в регистр данных SPI (SPDR) во время передачи данных. Бит WCOL (а также бит SPIF) сбрасывается при первом чтении регистра статуса SPI с установленным WCOL, а также во время доступа к регистру данных SPI.

• SPI2X: бит удвоения скорости SPI. Если SPI2X=1, то скорость работы SPI (частота SCK) удвоится, если SPI находится в режиме мастера. Если SPI работает как подчиненный, то работа SPI гарантирована только на частоте fosc /4 или менее.

Для данных в SPI существует регистр данных (SPDR). Регистр данных SPI имеет доступ на чтение и запись и предназначен для обмена данными между файлом регистров и сдвиговым регистром SPI. Запись в данный регистр инициирует передачу данных. При чтении данного регистра фактически считывается содержимое приемного буфера сдвигового регистра.

Режимы передачи

У SPI есть четыре режима передачи, которые основаны на комбинации «полярности» тактового сигнала (clock polarity, CPOL) и фазы синхронизации (clock phase, CPHA). Проще говоря, CPOL – это уровень на тактовой линии до начала и после окончания передачи: низкий (0) или высокий (1). А фаза определяет, на фронте или спаде тактового сигнала передавать биты (рис. 2.3):

– Режим 0: CPOL=0, CPHA=0. Чтение бита происходит на фронте тактового сигнала (переход 0 -> 1), а запись – на спаде (1 -> 0).

– Режим 1: CPOL=0, CPHA=1. Чтение – на спаде, запись – на фронте.

– Режим 2: CPOL=1, CPHA=0. Чтение – на спаде, запись – на фронте.

– Режим 3: CPOL=1, CPHA=1. Чтение – на фронте, запись – на спаде.

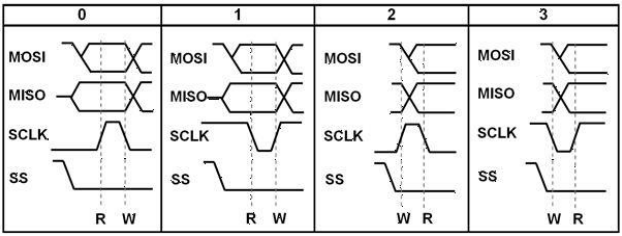

Рис. 2.3. Режимы передачи SPI

Данные по SPI можно передавать либо старшим битом вперед, либо младшим. Обычно используется первый вариант, но перед началом работы с устройством следует уточнить это в документации.

2.2. Моделирование передачи данных по протоколу SPI

Данные передаются по 8-битному формату. Не принимать во внимание прерывания по SPI. Биты данных должны быть отправлены, начиная со старшего разряда (рис.2.4).

Рис. 2.4. Рабочее окно

Программа автоматически генерирует произвольные числа для отправки. Эти числа выводятся в поле **Передаваемое число** в верхней части окна. В поле **Режим передачи** выводится один из четырех режимов передачи, а в поле **Частота синхросигнала** показана частота SCK.

В правой части окна представлена цель лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо нажать на **Шаблоны,** предварительно записав в них данные.

Перед тем, как отправить данные по протоколу SPI, необходимо заполнить поля **Заполненный регистр SPCR** и **Заполненный регистр SPSR**. Выберите «Регистр управления SPI» для того, чтобы заполнить **регистр SPCR.** После чего соберите кадр и нажмите кнопку «Загрузить». Если кадр собран правильно, то он переместится в поле **Заполненный регистр SPCR**.

Если кадр собран неправильно, индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Для того чтобы заполнить **регистр SPSR**, выберите «Регистр статуса SPI» и проделайте ту же последовательность шагов.

После правильного заполнения регистров можно начать отправку данных. Для этого выберите «Регистр данных SPI» и нажмите на шаблон данных. Шаблон с данными переместится в выбранное поле.

Нажмите кнопку «Отправить», и шаблон с данными переместится в поле **Отправленная информация.** Переданное число может быть проверено нажатием кнопки «Проверить».

Если число введено правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если число введено неправильно, программа выведет диалоговое окно с надписью «**Неправильно!»**. В этом случае очистите все поля и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола SPI нажмите кнопку «SPI» в верхнем правом углу окна.

2.3. Моделирование получения данных по протоколу SPI

Данные передаются по 8-битному формату. Не принимать во внимание прерывания по SPI. Биты данных должны быть отправлены, начиная со старшего разряда.

Открыть вкладку Лабораторная работа 2 в опции «Протокол SPI».

В поле **Режим передачи** выводится один из четырех режимов передачи, а в поле **Частота синхросигнала** показана частота SCK.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо нажать на **Шаблоны,** предварительно записав в них данные.

Перед тем, как получить данные по протоколу SPI, необходимо заполнить поля **Заполненный регистр SPCR** и **Заполненный регистр SPSR**. Выберите "Регистр управления SPI" для того, чтобы заполнить **регистр SPCR.** После чего соберите кадр и нажмите кнопку «Загрузить».

Если кадр собран правильно, то он переместится в поле **Заполненный регистр SPCR**.

Если кадр собран неправильно, индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Для того чтобы заполнить **регистр SPSR**, выберите «Регистр статуса SPI» и проделайте ту же последовательность шагов.

Для получения данных необходимо отправить какую-нибудь информацию. Для этого выберите «Регистр данных SPI» и нажмите на шаблон данных. Шаблон с данными переместится в выбранное поле. Нажмите кнопку «Отправить», и шаблон с данными переместится в поле **Отправленная информация**.

Нажмите кнопку «Получить», чтобы получить данные. Данные в поле **Полученный пакет** выводятся в двоичном виде. Переведите двоичный код в десятичный, введите полученное число в поле **Полученные числа** и нажмите кнопку "Проверить".

Если число введено правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если число введено неправильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае исправьте значение числа и снова нажмите кнопку «Проверить».

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

Для перехода к меню протокола SPI нажмите кнопку «SPI» в верхнем правом углу окна. Оформите отчет, сделайте выводы об особенностях работы протокола SPI, структуре кадра и режимах работы данного протокола.

Контрольные вопросы

1. Каково назначение и применение протокола SPI?
2. Какова структура кадра в режиме передачи?
3. Какие особенности приема кадра SPI?

4. Какова очередность передачи битов?

3. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ I2C

3.1. Общие сведения о протоколе 12C

I2C (Inter **–** Integrated Circuit) **–** последовательная шина данных для связи интегральных схем. Она была разработана фирмой Philips в начале 1980-х в качестве простой шины внутренней связи для создания управляющей электроники. Используется для соединения низкоскоростных периферийных компонентов с материнской платой, получения данных от разнообразных датчиков, связи между различными компонентами встраиваемых систем и т.д.

Физически I2C представляет собой две двунаправленные линии – линия последовательных данных (Serial DAta Line – SDA) и линия синхронизации (Serial CLock Input SCL), подтянутые к напряжению питания и подключаемые по схеме с открытым коллектором или открытым стоком (рис. 3.1).

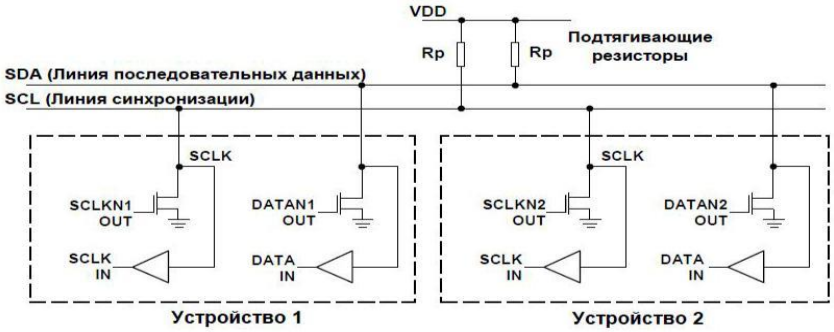

Рис. 3.1. Соединение I2C устройств

Любой элемент, инициирующий передачу, является мастером (master), любой адресуемый элемент является подчиненным (slave). В системах с несколькими мастерами один и тот же элемент может в разное время выступать и как мастер, и как подчиненный.

Когда шина свободна, обе линии находятся в состоянии «1». Данные могут передаваться по шине I2C со скоростью до 100 kbit/s в стандартном режиме или до 400 kbit/s в быстром режиме. Число подключаемых к шине интерфейсов зависит

исключительно от емкости шины, максимальное значение – 400pF.

Существуют два особых состояния шины I2C: START и STOP, которые служат для индикации начала и конца передачи и, соответственно, перехода шины в неактивное состояние. Следует отметить, что до тех пор, пока не установлено состояние START, сигналы на линиях SDA и SCL могут быть совершенно произвольными (рис. 3.2).

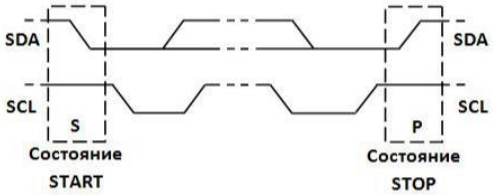

Рис. 3.2. Состояния START и STOP

Состояние START – переход от «1» к «0» на линии SDA при «1» на линии SCL.

Состояние STOP – переход от «0» к «1» на линии SDA при «1» на линии SCL. Эти два состояния всегда генерируются мастером.

Поскольку линии шины подтянуты к источнику питания, то устройствам необходимо притягивать линии к земле для того, чтобы послать «0», и отпустить для того, чтобы послать «1». Подключение для совместной работы устройств с таким включением называется монтажное «И».

Все передачи производятся побайтно. Число байтов, которые могут быть переданы за одну передачу, не ограничено. Каждый байт должен сопровождаться битом подтверждения ACK (ACKnowledge). Данные передаются, начиная со старшего бита MSB (Most Significant Bit) (рис. 3.3). Если приемник не может получить полный байт данных, он не выдает сигнала ACK, который используется передатчиком для синхронизации и сигнализации о неисправности приемника (или его отсутствии).

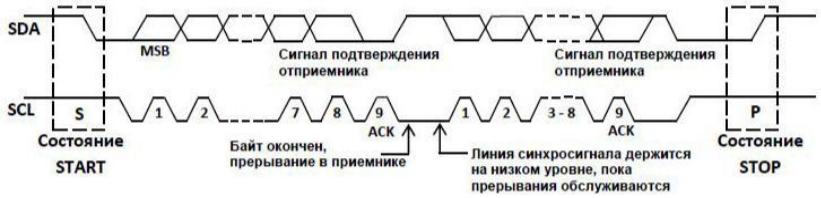

Рис. 3.3. Передача данных по шине I2C

Для подтверждения передачи байта передатчик устанавливает линию SDA в «1» в течение синхронизирующего импульса. Приемник при этом должен выставить «0» на SDA (рис. 3.4).

Когда подчиненный приемник не подтверждает подчиненный адрес (NACK), линия данных SDA должна оставаться в "1". После этого мастер может выдать состояние STOP и повторить передачу.

Если подчиненный приемник подтверждает подчиненный адрес, но через некоторое время не может больше получать байты данных, мастер должен приостановить передачу. При приеме последнего байта в серии вместо сигнала ACK мастер может выставить состояние STOP, при этом подчиненный передатчик должен освободить линию данных.

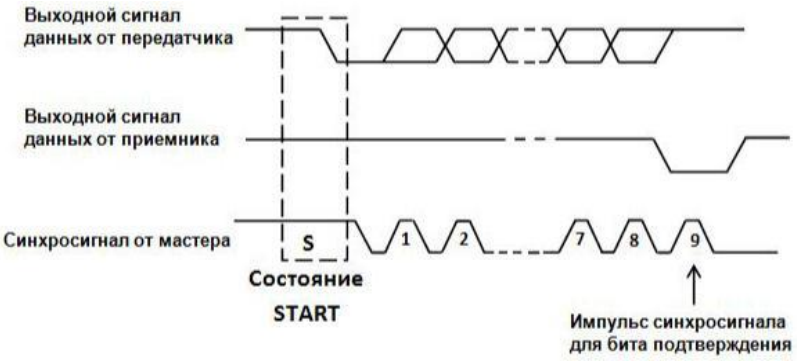

Рис. 3.4. Подтверждение передачи

Передача с 7 – битной адресацией показана на рис. 3.5. После выдачи состояния START следует передача адресного байта, при этом 8-й байт адреса определяет направление передачи данных («0» – запись данных от мастера к подчиненному, «1» – чтение данных из подчиненного к мастеру). Передача данных всегда завершается сгенерированным мастером состоянием STOP.

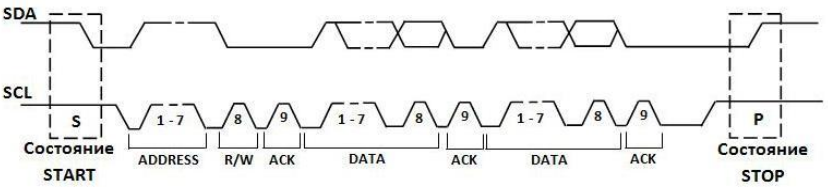

Рис. 3.5. Полная схема передачи данных

Ниже приводятся примеры передачи данных от мастера к подчиненному, а также чтение мастером данных с подчиненного устройства (рис. 3.6, 3.7).

Рис. 3.6. Передача от мастера к подчиненному

Рис. 3.7. Чтение мастером данных с подчиненного устройства

3.2. Моделирование процесса передачи данных по протоколу I2C

Данные должны быть переданы мастером подчиненному с данным адресом. Числа необходимо передать по 8- битному формату. Биты чисел должны быть отправлены, начиная со старшего разряда (MSB).

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению (рис.3.8).

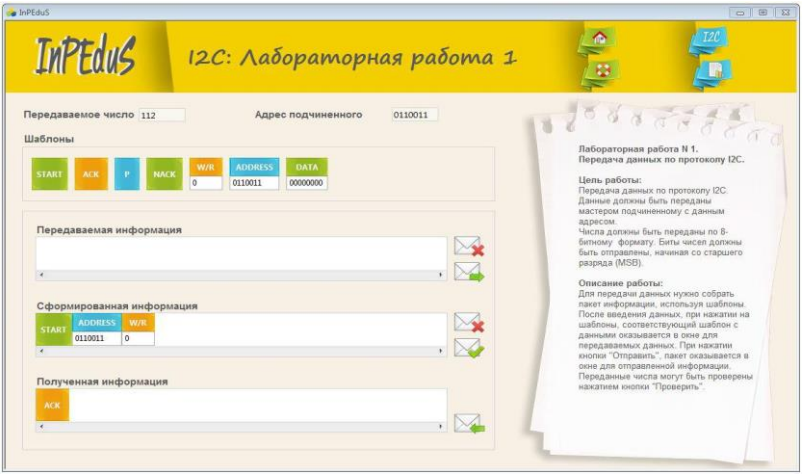

Рис. 3.8. Рабочее окно

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них**,** после чего шаблон с данными окажется в поле **Передаваемая информация.** Необходимо указать начало кадра, адрес подчиненного устройства, а также определить направление передачи данных. Адрес подчиненного устройства автоматически генерируется в поле **Адрес подчиненного**. Нажмите кнопку «Отправить».

В случае ошибки индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если начало кадра собрано правильно, после нажатия кнопки «Отправить» данные переместятся в поле **Сформированная информация,** а в поле **Полученная информация** появится бит подтверждения ACK. Нажмите кнопку «Получить» для того, чтобы бит подтверждения переместился в поле **Сформированная информация.**

В поле **Передаваемое число** появится число для отправки.

Вы можете отправить несколько чисел, после чего укажите конец кадра и нажмите кнопку «Проверить».

Если числа введены правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если числа введены непра-

вильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае очистите поле **Сформированная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола I2C нажмите кнопку «I2C» в верхнем правом углу окна.

3.3. Моделирование процесса получения данных по протоколу I2C

Данные передаются по 8-битному формату. Биты передаются, начиная со старшего бита. Для этого необходимо открыть окно Inpedus, выбрать папку «Протокол I2C», нажать кнопку Лабораторная работа 2.

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них**,** после чего шаблон с данными окажется в поле **Передаваемая информация.** Необходимо указать начало кадра, адрес подчиненного устройства, а также определить направление передачи данных. Адрес подчиненного устройства автоматически генерируется в поле **Адрес подчиненного**. Нажмите кнопку «Отправить».

В случае ошибки индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если начало кадра собрано правильно, после нажатия кнопки «Отправить» данные переместятся в поле **Сформированная информация,** а в поле **Полученная информация** появится бит подтверждения ACK и шаблон с данными. Нажмите кнопку «Получить» для того, чтобы бит подтверждения и шаблон с данными переместились в поле **Сформированная информация.**

Для того чтобы проверить данные, переведите двоичный код в десятичный, введите полученное число в поле **Полученные числа** и нажмите кнопку «Проверить».

Если число введено правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если число введено неправильно, программа выведет диалоговое окно с надписью «**Неправильно!»**. В этом случае исправьте значение числа и нажмите кнопку «Проверить».

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Контрольные вопросы

1. Что представляет собой интерфейс I2C?
2. Поясните временную диаграмму сигналов на рис.3.5.
3. Какова логическая структура протокола I2C?

4. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ RS-232

4.1. Общие сведения о протоколе RS-232

RS-232 (Recommended Standard) **–** популярный протокол, применяемый для связи компьютеров с модемами и другими периферийными устройствами. RS-232 был введен в 1962 году. Оборудование, соединяемое по RS-232 протоколу, разделяют на два типа: DCE (Data Communication Equipment /оборудование передачи данных/) и DTE (Data Terminal Equipment /терминальное оборудование/).

При использовании RS-232 информация между двумя устройствами может передаваться на расстояние до 20 м, так как после прохождения по кабелю сигналы ослабляются и искажаются. Для обеспечения большей устойчивости к помехам при передаче информации по проводам используются более высокие, чем стандартные 5В, уровни сигналов.

В RS-232 используются два уровня сигналов: логическая 1 (mark) и 0 (space). Логической единице соответствует отрицательный уровень напряжения, а логическому нулю – положительный. Значения напряжений, используемых при передаче данных по этому протоколу, представлены в табл. 4.1 и 4.2.

Таблица 4.1

Уровни сигналов данных

Уровень	Передатчик	Приемник	
Логический ноль	От +5 В до +15 В	От +3	В до +25 В
Логическая единица	От-5 В до -15 В	От -3	В до -25 В
Не определен	От -3 В до +3 В		

Таблица 4.2

Уровни управляющих сигналов

Сигнал	На выходе устройства (Driver)		На входе устройствам (Terminator)	
«Off»	От -5	В до -15 В	от -3	В до -25 В
«On»	От 5	В до 15 В	от 3	В до 25 В

В табл. 4.3 представлена максимальная длина кабеля в зависимости от скорости передачи информации.

Таблица 4.3

Длина кабеля в зависимости
от скорости передачи информации

Скорость [бод]	Макс. длина [футы]	Макс. длина [метры]
19 200	50	15
9 600	500	150
4 800	1000	300
2 400	3000	900

Скорость передачи информации по RS-232 измеряется в Бодах (бит в секунду). Максимальная скорость согласно стандарту составляет 20000 Бод. Однако современное оборудование может работать значительно быстрее.

Устройства для связи по последовательному каналу соединяются кабелями с 9- или 25-контактными разъемами типа D. Обычно они обозначаются DB-9, CANNON 9, CANNON 25 и т.д. Каждый вывод обозначен и пронумерован. На рис. 4.1 показан контактный разъем кабеля DB-9.

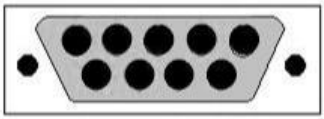

Рис. 4.1. Разъем кабеля DB-9

В табл. 4.4 представлены обозначения контактов разъема DB-9, а также направление и краткое описание сигнала.

Таблица 4.4

Обозначения контактов

Контакт	Обозначение	Направление	Описание
1	DCD	Вход	Наличие несущей
2	RXD	Вход	Прием данных
3	TXD	Выход	Передача данных
4	DTR	Выход	Готовность приемника данных
5	GND	---	Земля
6	DSR	Вход	Готовность источника данных
7	RTS	Выход	Запрос на передачу
8	CTS	Вход	Готовность передачи
9	RI	Вход	Сигнал вызова

Контроль четности (Parity)

Для обеспечения контроля четности компьютер и устройство должны производить подсчет битов четности по одинаковому алгоритму (необходимо определиться, будет ли бит четности устанавливаться при четном (even) или нечетном (odd) числе единиц). При контроле четности биты данных и бит четности всегда должны содержать четное число единиц. Противоположный случай соответствует для контроля на нечетность.

Проверка на четность – это простейший способ обнаружения ошибок. Он может определить возникновение ошибок в одном бите, но при наличии ошибок одновременно в двух битах может пропускать ошибки. Кроме того, такой контроль не определяет, какой именно бит является ошибочным.

К другим механизмам проверки верности передачи сигнала относятся включение в передачу старт- и стоп-битов, контроль с помощью циклического избыточного кода и т.д.

Сигнальная линия может находиться в двух состояниях: включена и выключена (логическая единица или ноль). Линия, находящаяся в состоянии ожидания всегда включена. Когда устройство или компьютер хотят передать данные, они переводят линию в состояние выключено – это установка старт-бита.

Стоп-бит позволяет устройству или компьютеру произвести синхронизацию при возникновении сбоев. Например, помеха на линии может скрыть старт-бит. Период между старт- и стоп-битами постоянен и зависит от скорости обмена, числа бит данных в посылке и наличия бита четности. Стоп-бит всегда включен. Если

приемник определяет выключенное состояние (логический ноль на линии) в то время, когда должен присутствовать стоп-бит, фиксируется появление ошибки.

На компьютерах обычно стоп-бит эквивалентен 1 или 2 битам. Наиболее распространен выбор 1 бита, выбор 2 бит немного замедлит передачу.

Иногда устройство не может обработать принимаемые от компьютера или другого устройства данные. Тогда устройство использует возможности управления потоком для того, чтобы приостановить передачу данных. Может быть использовано как аппаратное, так и программное управление потоком.

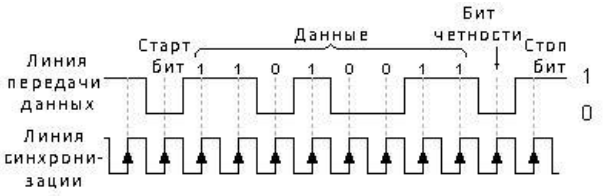

Рис. 4.2. Временная диаграмма

На рис. 4.2 показана структура передаваемых данных с синхронизирующим тактовым сигналом. В этом примере используется 8 бит данных, бит четности и стоп-бит. Такая структура также обозначается 8E1.

4.2. Моделирование передачи данных по протоколу RS-232

Данные передаются по 8-битному формату. При передаче биты четности не применяются, а данные сопровождаются одним стоп-битом.

Программа автоматически генерирует числа, которые необходимо отправить. Эти числа выводятся в поле **Передаваемое число**.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению (рис.4.3).

Рис. 4.3. Рабочее окно

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них**,** после чего шаблон с данными окажется в поле **Передаваемая информация.** Выберите правильные состояния для сигналов RS-232 и нажмите кнопку «Отправить».

В случае, если кадр собран неверно и/или выбраны неправильные состояния для сигналов, индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Отправленная информация**. Переданное число может быть проверено нажатием кнопки «Проверить». Если число введено правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если число введено неправильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае очистите поля **Передаваемая информация** и **Отправленная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

Для перехода к меню протокола RS-232 нажмите кнопку «RS-232» в верхнем правом углу окна.

4.3. Моделирование процесса получения данных по протоколу RS-232

Данные передаются по 8-битному формату. Бит четности не применяется, а данные сопровождаются одним стоп-битом. Для этого необходимо открыть окно Inpedus, нажать кнопку Лабораторная работа 2.

Для получения данных необходимо отправить запрос на передачу и известить о состоянии устройства путем применения соответствующих уровней сигналов RS-232. Если логические уровни всех сигналов RS-232 выбраны правильно, в окне **Полученный пакет информации** появится первый пакет данных. Нажмите кнопку «Получить», и пакет автоматически переместится в поле **Полученная информация**, а в окне **Полученный пакет информации** появится следующий пакет данных.

В случае если логические уровни сигналов RS-232 выбраны неправильно, индикатор ошибки загорится красным светом. Исправьте логические уровни сигналов и нажмите кнопку «Получить».

Введите полученное число в поле **Полученные числа** и нажмите кнопку «Проверить».

Если число введено правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если число введено неправильно, программа выведет диалоговое окно с надписью «**Неправильно!»**. В этом случае исправьте значение числа и снова нажмите кнопку «Проверить».

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

Для перехода к меню протокола RS-232 нажмите кнопку «RS-232» в верхнем правом углу окна.

Контрольные вопросы

1. Дайте определение интерфейсу RS-232.
2. Поясните временную диаграмму сигналов на рис.3.2.
3. Поясните назначение разъемов на рис. 3.3.
4. В чем отличие интерфейсов RS-232 и I2C?

5. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ RS-485/RS-422

5.1. Общие сведения о протоколах RS-485 и RS-422

Интерфейсы являются одними из наиболее распространенных стандартов связи физического уровня.

Стандарт RS-485 совместно разработан двумя ассоциациями: _ссоциациией электронной промышленности (EIA – Electronics Industries Association) и Ассоциацией промышленности средств связи (TIA – Telecommunications Industry Association).

RS-422 – американский стандарт, его международный эквивалент ITU-T Recommendation V.11.

Сети, построенные на интерфейсах RS-485/422, представляют собой приемопередатчики, соединенные при помощи витой пары – двух скрученных проводов. В основе интерфейсов RS-485/422 лежит принцип дифференциальной (балансной) передачи данных. Такой способ передачи обеспечивает высокую устойчивость к синфазной помехе. Синфазной называют помеху, действующую на оба провода линии одинаково.

Передатчик должен обеспечивать уровень сигнала 1.5В при максимальной нагрузке и не более 6В на холостом ходу. Уровни напряжений измеряются дифференциально: напряжение на одном сигнальном проводе относительно другого. На стороне приемника минимальный уровень принимаемого сигнала должен быть не менее 200 мВ.

Максимальная скорость связи по спецификации RS-485/422 может достигать 10 Мбод. Максимальное расстояние – 1200 м. Если необходимо организовать связь на расстоянии более 1200 м или подключить больше число устройств, чем допускает нагрузочная способность передатчика, применяют специальные повторители (репитеры).

Стандарт RS-485 предусматривает только 32 пары передатчик/приемник, но производители расширили возможности RS-485 протокола, так что теперь он поддерживает от 128 до 255 устройств на одной линии, а используя репитеры, можно удлинять RS-485/RS-422 практически до бесконечности. При использовании RS-485 можно, и в случае длинного провода и/или большого количества устройств необходимо, использовать терминаторы, которые, впрочем, обычно встроены в устройства с RS-485 протоколом

(хотя при коротком проводе может наблюдаться даже ухудшение связи при использовании терминаторов). Стандарт RS-485 также предусматривает использование двухжильной экранированной витой пары (так называемый 2-wire RS-485), но возможно использование и четырехпроводной витой пары (4-wire RS-485), при этом можно получить полный дуплекс. В этом случае необходимо, чтобы одно из устройств было сконфигурировано как ведущее (Master), а остальные как ведомые (Slave). Тогда все ведомые устройства общаются только с ведущим устройством и никогда не передают ничего напрямую друг другу. В таких случаях обычно RS-422 драйвер используется как ведущее устройство, так как RS-422 допускает подключения только как master/slave, а RS-485 устройства как ведомые, для общего удешевления системы.

Стандарт на RS-422 изначально предусматривает использование четырехжильной экранированной витой пары, но допускает соединения только от одного устройства к другим (до пяти драйверов и до десяти ресиверов на каждый драйвер).

RS-422 является полнодуплексным интерфейсом. Прием и передача идут по двум отдельным парам проводов. На каждой паре проводов может быть только по одному передатчику.

RS-485 является полудуплексным интерфейсом. Прием и передача идут по одной паре проводов с разделением по времени. В сети может быть много передатчиков, так как они могут отключаться в режиме приема.

В табл. 5.1 приведены основные технические параметры протоколов RS 485/422.

Таблица 5.1

Основные технические параметры

Параметры интерфейсов	RS-422	RS-485
Допустимое число передатчиков / приемников	1 / 10	32 / 32
Максимальная длина кабеля	1200 м	1200 м
Максимальная скорость связи	10 Мбит/с	10 Мбит/с
Диапазон напряжений «1» передатчика	+2…+10 В	+1.5…+6 В
Диапазон напряжений «0» передатчика	-2…-10 В	-1.5…-6 В
Диапазон синфазного напряжения передатчика	-3…+3 В	-1…+3 В

Допустимый диапазон напряжений приемника	-7…+7 В	-7…+12 В
Пороговый диапазон чувствительности приемника	±200 мВ	±200 мВ
Максимальный ток короткого замыкания драйвера	150 мА	250 мА
Допустимое сопротивление нагрузки передатчика	100 Ом	54 Ом
Входное сопротивление приемника	4 кОм	12 кОм
Максимальное время нарастания сигнала пере датчика	10% бита	30% бита

RS-422 использует строго разделенные две (или более) пары проводов: одну пару для приема, одну для передачи, и еще по одной на каждый сигнал контроля/подтверждения (control/handshake).

Устройства для связи по последовательному каналу соединяются кабелями с 9- или 25-контактными разъемами типа D. Обычно они обозначаются DB-9, CANNON 9, CANNON 25 и т.д. Каждый вывод обозначен и пронумерован.

В табл. 5.2 представлены обозначения контактов разъема DB-9, а также направления и краткие описания сигналов.

Таблица 5.2

Обозначения контактов

Контакт	Обозначение	Направление	Описание
1	TXD-	Выход	Передача данных
2	TXD+	Выход	Передача данных
3	RTS-	Выход	Запрос на передачу
4	RTS+	Выход	Запрос на передачу
5	GND	---	Земля
6	RXD-	Вход	Прием данных
7	RXD+	Вход	Прием данных
8	CTS-	Вход	Готовность передачи
9	CTS+	Вход	Готовность передачи

Структура данных, передаваемых по протоколам RS-485/422, идентична протоколу RS-232. На рис. 5.1 показана структура передаваемых данных со синхронизирующим тактовым сигналом. В

этом примере используется 8 битов данных, бит четности и стоп-бит. Такая структура также обозначается 8E1.

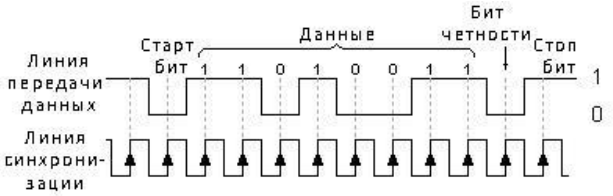

Рис. 5.1. Временная диаграмма передаваемых данных

Иногда устройство не может обработать принимаемые от компьютера или другого устройства данные. Тогда устройство использует возможности управления потоком для того, чтобы приостановить передачу данных. Может быть использовано как аппаратное, так и программное управление потоком.

5.2. Моделирование передачи данных по протоколу RS-485/RS-422

Данные необходимо передать по 8-битному формату. При передаче биты четности не применяются, а данные сопровождаются одним стоп-битом.

Для запуска лабораторного стенда необходимо открыть вкладку в папке «Протокол RS-485/RS-422», «Лабораторная работа 1».

Программа автоматически генерирует числа, которые необходимо отправить. Эти числа выводятся в поле **Передаваемое число**.

В правой части окна представлена цель лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них**,** после чего шаблон с данными окажется в поле **Передаваемая информация.** Выберите правильные состояния для сигналов RS-422/485 и нажмите кнопку «Отправить».

В случае если кадр собран неверно и/или выбраны неправильные состояния для сигналов, индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Отправленная информация**. Переданные числа могут быть проверены нажатием кнопки «Проверить». Если числа введены правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если числа введены неправильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае очистите поля **Передаваемая информация** и **Отправленная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола RS-485/RS-422 нажмите кнопку «RS-485/RS-422» в верхнем правом углу окна.

5.3. Моделирование процесса получения данных по протоколу RS-485/RS-422

Данные передаются по 8-битному формату. Бит четности не применяется, а данные сопровождаются одним стоп-битом. Для этого необходимо открыть окно Inpedus, нажать кнопку Лабораторная работа 2.

Для получения данных необходимо отправить запрос на передачу и известить о состоянии устройства путем применения соответствующих уровней сигналов RS-485/RS-422. Если логические уровни всех сигналов RS-485/RS-422 выбраны правильно, в окне **Полученный пакет информации** появится первый пакет данных. Нажмите кнопку «Получить» и пакет автоматически переместится в поле **Полученная информация**, а в окне **Полученный пакет информации** появится следующий пакет данных.

В случае если логические уровни сигналов RS-485/RS-422 выбраны неправильно, индикатор ошибки загорится красным светом. Исправьте логические уровни сигналов и нажмите кнопку «Получить».

Введите полученное число в поле **Полученные числа** и нажмите кнопку «Проверить».

Если числа введены правильно, программа выведет диалоговое окно с надписью «**Правильно!»**. Если числа введены неправильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае исправьте значения чисел и снова нажмите кнопку «Проверить».

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола RS-485/RS-422 нажмите кнопку «RS-485/RS-422» в верхнем правом углу окна.

Сформировать отчет, сделать выводы о режимах работы, особенностях передачи и приема сигналов по протоколу RS-485/RS-422.

Контрольные вопросы

1. Поясните назначение интерфейсов RS-485/RS-422.
2. Поясните временную диаграмму сигналов рис. 5.1.
3. Есть ли особенности протоколов RS-485 и RS-422? Чем они отличаются от RS-232?
4. Поясните назначение контактов интерфейсов RS-485/RS-422.
5. Какова дальность связи при использовании RS-485/RS-422, от чего она зависит?

6. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ MODBUS

6.1. Общие сведения о протоколе Modbus

Modbus – коммуникационный протокол, основанный на архитектуре клиент-сервер. Широко применяется в промышленности для организации *связи между электронными устройствами.* Может использоваться для передачи данных через последовательные линии связи *RS-485, RS-422, RS-232,* а также сети *TCP/IP (Modbus TCP).* Modbus был разработан компанией Modicon. Впервые спецификация протокола была опубликована в 1979 году.

Разновидностями Modbus являются протоколы Modbus Plus – многомастерный протокол с кольцевой передачей маркера и Modbus TCP, рассчитанный на использование в сетях Ethernet и Интернет. Сам же протокол Modbus имеет два режима передачи: RTU (Remote Terminal Unit – «удаленное терминальное устройство») и ASCII. Стандарт предусматривает, что режим RTU в протоколе Modbus должен присутствовать обязательно, а режим ASCII является опционным.

Основной особенностью протокола является наличие в сети одного ведущего устройства – мастера (master). Протокол Modbus позволяет создать промышленную сеть из одного ведущего устройства (мастера) и до 247 ведомых. Только мастер может опрашивать остальные устройства сети, которые являются подчиненными (slave). Подчиненное устройство не может самостоятельно инициировать передачу данных или запрашивать какие- либо данные у других устройств. Мастер может также выдать широковещательный запрос, адресованный всем устройствам в сети, в таком случае ответное сообщение не посылается.

Режимы передачи

Режим передачи (табл. 6.1) определяет структуру отдельных блоков информации в сообщении и системы счисления , используемую для передачи данных. В системе Modbus существуют два режима передачи. Оба режима обеспечивают одинаковую совместимость при связи с подчиненными. Режим выбирается в зависимости от оборудования, используемого как Master Modbus. Для каждой системы Modbus должен использоваться только один режим. Режимы делятся на ASCII и RTU (Remote Terminal Unit).

Таблица 6.1

Характеристики режимов ASCII и RTU

Характеристика	ASCII (7-бит)	RTU (8-бит)
Система кодирования	Используются ASCII символы 0-9. A- F	8-битовая двоичная система
Число бит на символ:		
стартовые биты	1	1
биты данных (LSB вперед)	7	8
Четность	Вкл./Выкл.	Вкл./Выкл.
Стоповые биты	1 или 2	1 или 2
Контрольная сумма	LRC (Longitudinal Redundancy Check). LRC	CRC (Cyclical Redundancy Check). CRC_16

Символы ASCII удобнее использовать при отладке , поэтому этот режим удобен для компьютеров, программируемых на языке высокого уровня. Режим RTU подходит для компьютеров, программируемых на машинных языках. В режиме RTU данные передаются в виде 8-разрядных двоичных символов. В режиме ASCII каждый RTU символ сначала делится на две 4-разрядные части (старший и младший), переводится в свой шестнадцатеричный эквивалент и затем используется в создании сообщения. ASCII режим использует в два раза больше символов, чем RTU режим, но декодирование и управление данными – легче. К тому же в режиме RTU символы сообщения должны передаваться непрерывным потоком. В режиме ASCII допустима задержка до 1 секунды между двумя соседними символами.

Для передачи данных по сетям Ethernet поверх TCP/IP используется **Modbus TCP**. Для Modbus TCP ADU выглядит следующим образом:

ID транзак-ции	ID прото-кола	Длина пакета	Адрес ведомого устрой-ства	Код функции	Данные

– ID транзакции – два байта, обычно нули.

– ID протокола – два байта, нули.

– Длина пакета – два байта, старший, затем младший, длина следующей за этим полем части пакета.

– Адрес ведомого устройства – адрес подчиненного устройства, к которому адресован запрос. Обычно игнорируется, если соединение установлено с конкретным устройством.

Поле контрольной суммы в Modbus TCP отсутствует.

Сообщения **Modbus RTU** передаются в виде кадров, для каждого из которых известно начало и конец. Признаком начала кадра является пауза (тишина) продолжительностью не менее 3,5 шестнадцатеричных символов (14 бит). Кадр должен передаваться непрерывно. Если при передаче кадра обнаруживается пауза продолжительностью более 1,5 шестнадцатеричных символов (6 бит), то считается, что кадр содержит ошибку и должен быть отклонен принимающим модулем. Эти величины пауз должны строго соблюдаться при скоростях ниже 19200 бит/с, однако при больших скоростях рекомендуется использовать фиксированные значения пауз, 1,75 мс и 750 мкс соответственно.

В протоколе Modbus RTU сообщения передаются в виде пакетов PDU (Protocol Data Unit):

Код функции	Данные

Но для передачи пакета по физическим линиям связи PDU (Protocol Data Unit) помещается в другой пакет, содержащий дополнительные поля. Этот пакет носит название ADU (Application Data Unit). Формат ADU зависит от типа линии связи.

Общая структура ADU следующая (в зависимости от реализации некоторые из полей могут отсутствовать):

Адрес ведомого устройства	Код функции	Данные	Блок обнаружения ошибок

Адрес ведомого устройства – это адрес подчиненного устройства, к которому адресован запрос. Ведомые устройства отвечают только на запросы, поступившие по их адресу. Ответ также начинается с адреса отвечающего ведомого устройства, который может изменяться от 1 до 247. Адрес 0 используется для широковещательной передачи, его распознает каждое устройство. Адреса в диапазоне 248…255 – зарезервированы.

Код функции – это следующее однобайтное поле кадра . Оно говорит ведомому устройству, какие данные или выполнение какого действия требует от него ведущее устройство.

Данные – это поле содержит информацию, необходимую ведомому устройству для выполнения заданной мастером функции, или содержит данные, передаваемые ведомым устройством в ответ на запрос ведущего. Длина и формат поля зависит от номера функции.

Блок обнаружения ошибок (CRC) – это контрольная сумма для проверки отсутствия ошибок в кадре . Максимальный размер ADU для последовательных сетей RS232/RS485 – 256 байт, для сетей TCP – 260 байт.

В протоколе Modbus RTU данные передаются байтами младшими разрядами вперед, с добавлением контрольных битов (стартовый, стоповый биты и бит четности).

По умолчанию в RTU режиме бит четности (паритета) устанавливают равным 1, если количество двоичных единиц в байте нечетное, и равным 0, если оно четное. Такой паритет называют четным (even parity), а метод контроля называют контролем четности. При четном количестве двоичных единиц в байте бит паритета может быть равен 1. В этом случае говорят, что паритет является нечетным (odd parity).

Контроль четности может отсутствовать, в этом случае вместо бита паритета должен использоваться второй стоповый бит.

Контроль ошибок в протоколе Modbus RTU

Во время обмена данными могут возникать ошибки двух типов:

– ошибки, связанные с искажениями при передаче данных;

– логические ошибки.

Ошибки первого типа обнаруживаются при помощи фреймов символов, контроля четности и циклической контрольной суммы *CRC*-16-IBM (используется число-полином = = 0×A001).

При этом младший байт передается первым, в отличие от байтов адреса и значения регистра в PDU.

В RTU режиме сообщение должно начинаться и заканчиваться интервалом тишины – временем передачи не менее 3,5 символов при данной скорости в сети. Затем передается адрес устройства. Вслед за последним передаваемым символом также следует интервал тишины продолжительностью не менее 3,5 символов. Новое сообщение может начинаться после этого интервала.

Фрейм сообщения передается непрерывно. Если интервал тишины продолжительностью 1,5 возник во время передачи фрейма, принимающее устройство должно игнорировать этот фрейм как неполный.

Таким образом, новое сообщение должно начинаться не раньше 3,5 интервала, так как в этом случае устанавливается ошибка.

Интервалы (для Serial Modbus RTU): при скорости 9600 и 11 битах в кадре (стартовый бит + 8 бит данных + бит контроля четности + стоп-бит): $3.5 \times 11 / 9600 = 0{,}00401041(6)$, т. е. более 4 мс; $1.5 \times 11 / 9600 = 0{,}00171875$, т. е. более 1 мс. Для скоростей более 19200 Бод допускается использовать интервалы 1,75 и 0,75 мс соответственно.

CRC-16 (Cyclic Redundancy Check)

Сообщение (только биты данных, без учета старт/стоповых бит и бит четности) рассматривается как одно последовательное двоичное число, у которого старший значащий бит (MSB) передается первым. Сообщение умножается на X16 (сдвигается влево на 16 бит), а затем делится на полином X16+X15+X2+1, выражаемое как двоичное число (1100000000000101). Целая часть результата игнорируется, а 16-битный остаток (предварительно инициализированный единицами для предотвращения случая, когда все сообщение состоит из нулей) добавляется к сообщению (старшим битом вперед) как два байта контрольной суммы. Полученное сообщение, включающее CRC, затем в приемнике делится на тот же полином (X16+X15+X2+1). Если ошибок не было, остаток от деления должен получиться нулевым (приемное устройство может рассчитать CRC и сравнить ее с переданной). Вся арифметика выполняется по модулю 2 (без переноса).

Устройство, используемое для подготовки данных для передачи, посылает условно самый правый (LSB) бит каждого символа первым. При расчете CRC первый передаваемый бит определен как MSB делимого. Так как арифметика не использует перенос, для удобства расчета CRC можно предположить, что MSB расположен справа. Поэтому порядок бит при расчете полинома должен быть

реверсивным. MSB полинома опускается , так как он влияет только на делитель, а не на остаток.
В результате получается 1010 0000 0000 0001 (A001H). Заметьте, что эта реверсивность порядка бит в любом случае не влияет на интерпретацию или порядок бит байт данных при вычислении CRC.

Пошаговая процедура расчета CRC-16:

1. Загрузить 16-разрядный регистр числом FFFFH.
2. Выполнить операцию XOR над первым байтом данных и старшим байтом регистра.
3. Поместить результат в регистр.
4. Сдвинуть регистр на один разряд вправо.
5. Если выдвинутый вправо бит единица, выполнить операцию XOR между регистром и полиномом 1010 0000 0000 0001 (A001H).
6. Если выдвинутый бит ноль, вернуться в шагу 4.
7. Повторять шаги 4 – 6 до тех пор, пока не будут выполнены 8 сдвигов регистра.
8. Выполнить операцию XOR над следующим байтом данных и регистром.
9. Повторять шаги 4-8 до тех пор, пока не будет выполнена операция XOR над всеми байтами данных и регистром.
10. Содержимое регистра представляет собой два байта CRC и добавляется к исходному сообщению старшим битом вперед.

Типы данных и стандартные функции Modbus

Modbus специфицирует 4 типа данных:

– Discrete Inputs – однобитовый тип, доступен только на чтение.

– Coils – однобитовый тип, доступен на чтение и на запись.

– Input Registers – 16-битовый знаковый или беззнаковый тип, доступен только на чтение.

– Holding Registers – 16-битовый знаковый или беззнаковый тип, доступен на чтение и на запись.

Для чтения значений этих данных используются функции с кодами 1–4 (0×01–0×04):

• 1 (0×01) – чтение значений из нескольких регистров флагов (**Read Coil Status**).

• 2 (0×02) – чтение значений из нескольких дискретных входов (**Read Discrete Inputs**).

• 3 (0×03) – чтение значений из нескольких регистров хранения (**Read Holding Registers**).

• 4 (0×04) – чтение значений из нескольких регистров ввода (**Read Input Registers**).

Запрос состоит из адреса первого элемента таблицы (значение которого требуется прочитать), а также количества считываемых элементов. Адрес и количество данных задаются 16-битными числами, старший байт каждого из них передается первым.

В ответе передаются запрошенные данные. Количество байт данных зависит от количества запрошенных элементов. Перед данными передается один байт, значение которого равно количеству байт данных.

Запись одного значения происходит при помощи следующих функций:

• 5 (0×05) – запись значения одного флага (**Force Single Coil**);

• 6 (0×06) – запись значения в один регистр хранения (**Preset Single Register**).

Команда состоит из адреса элемента (2 байта) и устанавливаемого значения (2 байта). Если команда выполнена успешно, ведомое устройство возвращает копию запроса.

Запись нескольких значений задается функциями:

• 15 (0×0F) – запись значений в несколько регистров флагов (Force Multiple Coils);

• 16 (0×10) – запись значений в несколько регистров хранения (Preset Multiple Registers).

Команда состоит из адреса элемента, количества изменяемых элементов, количества передаваемых байт устанавливаемых значений и самих устанавливаемых значений. В ответе ведомый передает начальный адрес и количество измененных элементов.

Логические ошибки

Для сообщений об ошибках второго типа протокол Modbus RTU предусматривает, что устройства могут отсылать ответы, свидетельствующие об ошибочной ситуации. Признаком того, что от-

вет содержит сообщение об ошибке, является установленный старший бит кода команды. Пример кадра при выявлении ошибки ведомым устройством в ответ на запрос приведен в табл. 6.2.

Таблица 6.2

Кадр ответа (Slave→Master)
при возникновении ошибки modbus RTU

Направление передачи	Адрес подчиненного устройства	Номер функции	Данные (или код ошибки)	CRC
Запрос (Master→Slave)	0×01	0×77	0×DD	0×C7 0×A9
Ответ (Slave→Master)	0×01	0×F7	0×EE	0×E6 0×7C

1. Если подчиненный принимает корректный запрос и может его нормально обработать, то возвращает нормальный ответ.
2. Если подчиненный не принимает значения, то он не отправляет никакого ответа. Мастер диагностирует ошибку по тайм-ауту.
3. Если подчиненный принимает запрос с обнаруженной ошибкой (parity, LRC,CRC), то он не отправляет никакого ответа. Мастер диагностирует ошибку по тайм-ауту.
4. Если подчиненный принимает запрос, но не может его обработать (обращение к несуществующему регистру и т. д.), отправляется ответ, содержащий данные об ошибке.

В случае приема ведомым устройством запроса на чтение с верным адресом, но неверными последующими байтами (например, контрольной суммы или других) ведомое устройство должно сформировать ответ, сообщающий ведущему устройству об ошибке приема.

Для этого к байту кода функции при ответе добавляется 0x80, а третий байт равен 0×02 – что является стандартным для протокола Modbus сигналом об ошибке приема запроса.

Стандартные коды ошибок

01 – принятый код функции не может быть обработан на подчиненном.

02 – адрес данных, указанный в запросе, не доступен данному подчиненному.

03 – величина, содержащаяся в поле данных запроса, является недопустимой величиной для подчиненного.

04 – невосстанавливаемая ошибка имела место, пока подчиненный пытался выполнить затребованное действие.

05 – подчиненный принял запрос и обрабатывает его, но это требует много времени. Этот ответ предохраняет главного от генерации ошибки тайм-аута.

06 – подчиненный занят обработкой команды. Главный должен повторить сообщение позже, когда подчиненный освободится.

07 – подчиненный не может выполнить программную функцию, принятую в запросе. Этот код возвращается для неудачного программного запроса, использующего функции с номерами 13 или 14. Главный должен запросить диагностическую информацию или информацию об ошибках с подчиненного.

08 – подчиненный пытается читать расширенную память, но обнаружил ошибку паритета. Главный может повторить запрос, но обычно в таких случаях требуется ремонт.

6.2. Моделирование процесса передачи данных по протоколу Modbus

Данные необходимо передать по 8-битному формату. Кадр должен включать в себя старт и стоп биты для каждого байта. При передаче бит четности не применяется (рис. 6.1).

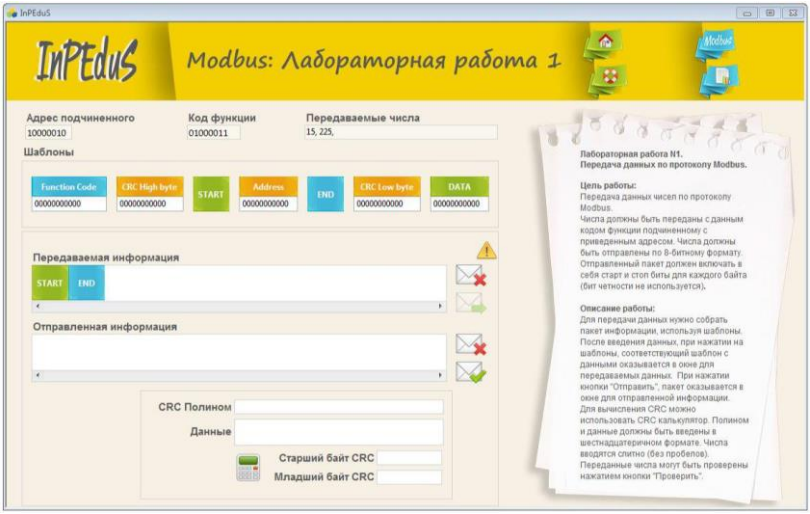

Рис.6.1. Рабочее окно

Программа автоматически генерирует числа, которые необходимо отправить. Эти числа выводятся в поле **Передаваемое число**. При передаче необходимо указать **Адрес подчиненного** и **Код функции.**

В правой части окна представлена цель лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них, после чего шаблон с данными окажется в поле **Передаваемая информация.** Нажмите кнопку «Отправить».

Индикатор ошибки загорится красным светом в случае ошибки при сборе кадра. Нажмите кнопку «Очистить» и соберите кадр заново. Вы можете отправить сразу все числа в одном кадре, либо отправить каждое число в отдельном кадре.

Для вычисления CRC используйте калькулятор в нижней части окна. В поля **Полином CRC** и **Данные** должны быть введены данные в шестнадцатеричном формате без пробелов. Нажмите кнопку «Вычислить CRC». В полях **Старший байт CRC** и **Младший байт CRC** данные также выдаются в шестнадцатеричном формате.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Отправленная информация**. Переданные числа могут быть проверены нажатием кнопки «Проверить». Если числа введены правильно, программа выведет диа-

логовое окно с надписью «**Правильно!»**. Если числа введены неправильно, программа выведет диалоговое окно с надписью «**Неправильно!»**. В этом случае очистите поля **Передаваемая информация** и **Отправленная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола Modbus нажмите кнопку «Modbus» в верхнем правом углу окна.

6.3. Моделирование проверки полученных данных по протоколу Modbus и уведомление об ошибке

Полученный кадр данных должен быть проверен, и при обнаружении ошибок ответный кадр с уведомлением об ошибке должен быть отправлен. Во вкладке «Протокол Modbus» открыть «Лабораторная работа 2».

Кадр должен включать в себя старт- и стоп-биты для каждого байта. При передаче бит четности не применяется.

В поле **Полученная информация** показан кадр, который необходимо проверить.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для того чтобы проверить кадр на ошибку, вычислите CRC-код с помощью калькулятора в нижней части окна. Проверьте, соответствует ли CRC -код данным кадра. В случае ошибки необходимо отправить ответный кадр с уведомлением об ошибке.

Для передачи ответного кадра в поле **Передаваемая информация** необходимо правильно собрать кадр. Для этого

нажмите на **Шаблоны**, и соответствующий шаблон с данными окажется в поле **Передаваемая информация.** Индикатор ошибки загорится красным светом в случае ошибки при сборе кадра. Нажмите кнопку «Очистить» и соберите кадр заново.

Для вычисления CRC используйте калькулятор в нижней части окна. В поля **Полином CRC** и **Данные** должны быть введены данные в шестнадцатеричном формате без пробелов. Нажмите кнопку «Вычислить CRC». В полях **Старший байт CRC** и **Младший байт CRC** данные также выдаются в шестнадцатеричном формате.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Отправленная информация.**

После выполнения лабораторной работы нажмите кнопку «**Сохранить отчет»**. В появившемся окне заполните поля **Имя** и **Фамилия** и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола Modbus нажмите кнопку «Modbus» в верхнем правом углу окна.

Контрольные вопросы

1. Дайте краткую характеристику применению протокола Modbus.
2. Какие режимы работы поддерживает протокол Modbus?
3. Какие функции поддерживает протокол Modbus?

4. Что означает код ошибки «05» в фрейме приема протокола?

5. Каков алгоритм формирования поля CRC-16? Какие задачи решает эта процедура

7. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ GPIB

7.1. Общие сведения о протоколе GPIB

GPIB (General Purpose Interface Bus) **–** интерфейсная шина общего назначения**.** В середине 1960-х годов компания Hewlett-Packard представила интерфейс Hewlett-Packard Interface Bus, HP-IB как многоцелевой контроллер. В 1970-х стандарт был преобразован в более общий GPIB, а также был принят как стандарт IEEE-488. В начале 1990-х годов были выпущены новые спецификации стандарта, а также выпущена спецификация языка программирования SCPI, позволяющего сравнительно удобно создавать программное обеспечение для измерительной техники.

Программные продукты для работы с шиной GPIB выпускаются в первую очередь компаниями National Instruments и Keithley Instruments, а также группой энтузиастов «Лабораторный проект для Линукс» (Linux Lab Project).

Характеристики

Каждое устройство на шине имеет уникальный пятибитный первичный адрес (от 0 до 30). Во избежание конфликтов адреса должны быть различными. Стандарт позволяет подключить до 15 устройств к одной двадцатиметровой физической шине, используя для наращивания соединители цепочечного типа.

К шине можно подключить три различных типа устройств (рис. 7.1):

– Слушатель (Listener).
– Спикер (Talker).
– Контроллер (Controller).

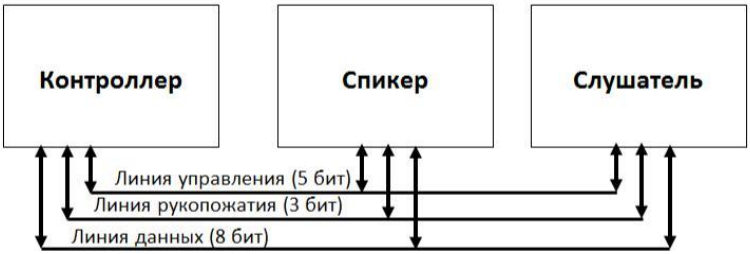

Рис. 7.1. GPIB система

Слушатель считывает сообщения с шины, а спикер посылает сообщения на шину. Спикером может быть только одно устройство, в то время как слушателей может быть произвольное количество устройств. Контроллер выполняет функции арбитра и определяет, какие из устройств в данный момент являются спикерами и слушателями. К шине может быть одновременно подключено несколько контроллеров. В этом случае один из контроллеров (как правило, расположенный на интерфейсной карте GPIB) является ответственным контроллером (Controller-in-Charge, CIC) и делегирует по мере надобности свои функции другим контроллерам.

Сигнальные линии шины относятся к одному из трех классов: линии данных, линии рукопожатия и линии управления интерфейсом. Для пересылки команд по шине используются восемь линий данных, причем старший бит (DIO8) в большинстве случаев игнорируется. Линии данных пронумерованы от 1 до 8, а не от 0 до 7, как в большинстве других стандартов.

Три линии рукопожатия управляют передачей данных и команд и обеспечивают гарантированный прием данных всеми слушателями в надлежащее время.

Ниже приведены примеры сигналов.

Сигнал Данные готовы (Data Valid) используется спикером для оповещения слушателей о том, что информация, подготовленная спикером, выставлена на линиях данных и готова к приему.

Сигнал Не готов к приему (Not Ready For Data) используется слушателями для того, чтобы сообщить спикеру о том, что они не готовы к приему данных. В этом случае спикер прекращает обмен информацией до того момента, когда все слушатели будут готовы к продолжению диалога. Сигнал реализован по принципу "монтажное ИЛИ", что позволяет каждому слушателю приостановить всю шину.

Сигнал Данные не приняты (Not Data Accepted) используется слушателями и сообщает спикеру, что данные приняты всеми адресатами. Когда этот сигнал не активен, спикер может быть уверен, что все клиенты успешно прочли данные с шины и можно приступать к передаче следующего байта данных. Процедура рукопожатия гарантирует, что скорость передачи данных по шине не превышает скорость их обработки самым медленным из клиентов.

Спикер помещает новые данные на шину только тогда, когда все слушатели готовы к приему.

Пять линий управления интерфейсом сообщают устройствам, присоединенным к шине, какие действия предпринимать, в каком режиме находиться и как реагировать на команды GPIB. Контроллер шины использует линию **Внимание (Attention)** для сообщения клиентам о том, что по шине идут команды, а не данные.

Сигнал Запрос обслуживания (Service Request) доступен любому клиенту шины. По этому сигналу контроллер переводит подавшее его устройство в состояние спикера и передает ему функции передачи данных.

Сигнал Очистка интерфейса (Interface Clear) используется для инициализации или реинициализации шины.

Сигнал Разрешить работу в удаленном режиме (Remote Enable) переводит устройство, подключенное к шине, в режим исполнения команд с шины (а не с контрольной панели) и обратно.

Сигнал Конец идентификации (End of Identify) используется спикером для идентификации конца сообщения. Контроллер выставляет этот сигнал для инициации параллельного опроса подключенных к шине устройств.

Электрически IEEE-488 восьмибитная параллельная шина, содержащая шестнадцать сигнальных линий (восемь двусторонних

используются для передачи данных, три – для установки соединения, пять – для управления шиной) плюс восемь – обратные провода для земли.

Все сигнальные линии используют отрицательную логику: наибольшее положительное напряжение интерпретируется как логический «0», а наибольшее отрицательное – как логическая «1». Линии данных (DIO) пронумерованы от 1 до 8, а линии данных (ЛД) в ГОСТ от 0 до 7.

Пять линий управления интерфейсом сообщают устройствам, присоединенным к шине, какие действия предпринимать, в каком режиме находиться и как реагировать на команды GPIB.

На рис. 7.2 изображено гнездо разъема IEEE-488, посредством которого осуществляется обмен сигналами.

Рис. 7.2. Разъемы

В табл. 7.1 указаны назначения разъемов интерфейса IEE-488.

Таблица 7.1

Назначение разъемов IEE-488

Номер контакта	наименование по IEEE		Назначение
1	Data input/output bit.	DIO1	Провод в КОП системы интерфейса, применяемый для передачи информации между соединенными устройствами
2	Data input/output bit.	DIO2	Провод в КОП системы интерфейса, применяемый для передачи информации между соединенными устройствами.
3	Data input/output bit.	DIO3	Провод в КОП системы интерфейса, применяемый для передачи информации между соединенными устройствами
4	Data input/output bit.	DIO4	Провод в КОП системы интерфейса, применяемый для передачи информации между соединенными устройствами
5	End-or-identify.	EOI	Используется спикером для идентификации конца сообщения. Контроллер выставляет этот сигнал для инициации параллельного опроса подключенных к шине устройств
6	Data valid.	DAV	Используется спикером для оповещения слушателей о том, что информация, подготовленная спикером, выставлена на линиях данных и достоверна
7	Not ready for	NRFD	Используется слушателями для того, чтобы сообщить командам, поступающим от контроллера
18	(wire twisted with DAV)	GND	Один из проводов «логической земли», скрученный с сигнальной линией, для минимизации взаимных помех между сигнальными линиями, восприимчивости сигнальных линий к внешним шумам и передачи интерфейсных сигналов во внешнюю среду
19	(wire twisted with NRFD)	GND	Аналогично
20	(wire twisted with NDAC)	GND	Аналогично
21	(wire twisted with IFC)	GND	Аналогично
22	(wire twisted with SRQ)	GND	Аналогично
23	(wire twisted with ATN)	GND	Аналогично
24	Logic ground		«Логическая земля»

Команды

Команды GPIB всегда передаются с использованием классического протокола IEEE-488.1. Стандарт задает формат команд, посылаемых инструментам, и формат и кодировку откликов. Команды, как правило, являются аббревиатурами соответствующих слов английского языка. Команды-запросы снабжаются на конце вопросительным знаком. Все обязательные команды префиксируются астериском «*». Стандарт определяет минимальный набор возможностей, которыми должен обладать каждый инструмент:

принимать и передавать данные, посылать запрос на обслуживание и реагировать на сигнал **Очистить интерфейс**. Все команды и большинство данных используют 7-битный набор ASCII, в котором не используется 8 бит, либо используется для четности.

Для получения информации от устройств, подключенных к шине, и переконфигурации шины контроллер посылает команды пяти классов:

– Однобитная (Uniline).

– Многобитная общего назначения (Universal Multiline).

– Многобитная адресная (Address Multiline).

– Многобитная групповая адресная передающая (Talk Address Group Multiline).

– Многобитная групповая адресная приемная (Listen Address Group Multiline).

Управляющие последовательности IEEE-488.2:

Описание	Управляющая последовательность	Требования IEEE-488.2
Посылка команда ATN-истинно	Send Command	Обязательно
Установка адреса для посылки данным	Send Setup	Обязательно
Посылка команд ATN-ложно	Send Data Bytes	Обязательно
Посылка программного сообщения	Send	Обязательно
Установка адреса для получения данных	Receive Setup	Обязательно
Получение данных ATN-ложно	Receive Response Message	Обязательно
Получение сообщения ответа	Receive	Обязательно
Активизация линии IFC	Send IFC	Обязательно
Очистка приборов	Device Clear	Обязательно
Установка приборов в автономное состояние	Enable Local Controls	Обязательно
Установка приборов в состояние удаленного управления	Enable Remote	Обязательно

Установка приборов в режим удаленного управления в состоянии локаута	Set RWLS	Обязательно
Установка приборов в автономное состояние в состоянии локаута	Send LLO	Обязательно
Чтение байта статуса 488.1	Read Status Byte	Обязательно
Посылка сообщения выполнения триггера группе (GET)	Trigger	Обязательно
Передача управления другому прибору	Pass Control	Обязательно
Параллельный опрос	Perform Parallel Poll	
Конфигурация приборов для параллельного опроса	Parallel Poll Configure	
Отмена возможности параллельного опроса	Parallel Poll Unconfigure	

Вторым компонентом системы команд является SCPI (Standard Commands for Programming Instruments), принятый в 1990 году. SCPI определяет стандартные правила сокращения ключевых слов, используемых в качестве команд. Ключевые слова могут быть использованы либо в длинной (например, MEASure – измерить), либо в короткой прописной форме (MEAS). Команды в формате SCPI префиксируются двоеточием. Аргументы команд разделяются запятой. Стандарт SCPI оперирует с моделью программируемого инструмента. Функциональные компоненты модели включают систему измерений (подсистемы «вход», «датчик» и «калькулятор»), систему генерации сигналов (подсистемы «калькулятор», «источник» и «выход») и подсистемы «формат», «показ», «память» и «триггер».

7.2. Моделирование передачи данных по протоколу GPIB

Данные должны быть отправлены слушателю с данным адресом по 8-битному формату. Для запуска лабораторного стенда необходимо открыть вкладку в папке «Протокол GPIB », «Лабораторная работа 1».

В верхней части окна представлен **Адрес слушателя** принимающего устройства. Программа автоматически генерирует числа, которые необходимо отправить. Эти числа выводятся в поле **Передаваемое число** в верхней части окна.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Перед тем как начать передачу, необходимо указать адрес слушателя. Переведите адрес из 10-ричного формата в двоичный и запишите двоичный код в полях **Передаваемая информация.** Выберите сигналы **ATN** и **DAV** и нажмите кнопку «Отправить».

Индикатор ошибки загорится красным светом в случае ошибки при вводе кода адреса слушателя и /или при выборе неправильного сигнала. Нажмите кнопку «Очистить» и введите правильный код.

Для передачи чисел отмените сигнал **ATN,** переведите передаваемое число из 10-ричного формата в двоичный и запишите двоичный код в полях **Передаваемая информация.** Вы можете отправить несколько чисел.

Для того чтобы указать конец передачи, выберите сигнал **EOI** вместе с последним передаваемым числом.

После нажатия кнопки «Отправить» передаваемые числа переместятся в поле **Отправленная информация**. Переданные числа могут быть проверены нажатием кнопки «Проверить».

Если числа введены правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если числа введены неправильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае очистите поля **Передаваемая информация** и **Отправленная информация** и повторите необходимые шаги.

Контрольные вопросы

1. Дайте определение протоколу GPIB.
2. Какие сигналы использует данный протокол?
3. Какие команды применяются для получения информации от подключенных устройств?

4. Поясните назначение входов/выходов интерфейса устройства подключения по протоколу GPIB.
5. Каково назначение сигналов ATN и DAV?

8. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ TCP

8.1. Общие сведения о протоколе TCP

TCP (Transmission Control Protocol) является одним из основных сетевых протоколов Интернета, предназначен для управления передачей данных в сетях и подсетях TCP/IP. Выполняет функции протокола транспортного уровня модели OSI. TCP – это транспортный механизм, предоставляющий поток данных, с предварительной установкой соединения. За счет этого протокол дает уверенность в достоверности получаемых данных, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета. В отличие от UDP, TCP гарантирует целостность передаваемых данных и уведомление отправителя о результатах передачи. На рис. 8.1 показан формат TCP сегмента.

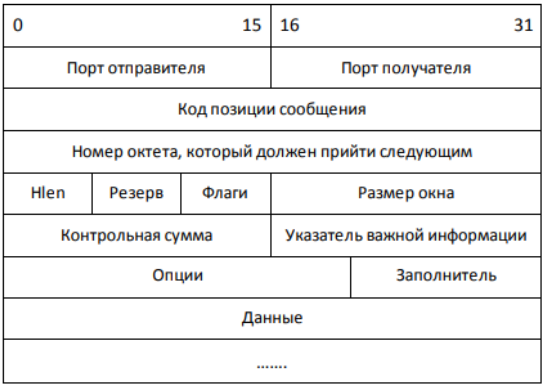

Рис. 8.1. Формат TCP сегмента

Порт источника (Source port) идентифицирует приложение клиента, с которого отправлены пакеты. По возвращении данные передаются клиенту на основании номера порта источника.

Порт назначения (Destination port) идентифицирует порт, на который отправлен пакет.

Существует набор служб (использующие для передачи данных TCP), за которыми закреплены определенные порты: 20/21 – FTP, 22 – SSH, 23 – Telnet, 25 – SMTP, 80 – HTTP, 110 – POP3, 194 – IRC (Internet Relay Chat), 443 – HTTPS (Secure HTTP), 1863 – MSN

Messenger, 2000 – Cisco SCCP (VoIP), 3389 – RDP, 8080 – альтернативный порт HTTP.

Номер последовательности (Sequence number) выполняет две задачи:

1. Если установлен флаг SYN (synchronize), то это начальное значение номера последовательности ISN (Initial Sequence Number). Первый байт данных, который будет передан в следующем пакете, будет иметь номер последовательности, равный ISN + 1.

2. Если SYN не установлен, первый байт данных, передаваемый в данном пакете, имеет этот номер последовательности.

Номер подтверждения (Acknowledgement number) – это следующий номер последовательности, который ожидает получить отправитель подтверждения. Это номер последовательности плюс 1 последнего успешно принятого байта данных. Это поле принимается в рассмотрение, только если флаг ACK взведен.

Смещение данных (Offset). Это поле определяет размер заголовка пакета TCP в 4-байтных словах. Минимальный размер составляет 5 слов, а максимальный – 15, что составляет 20 и 60 байт соответственно. Смещение считается от начала заголовка TCP.

Зарезервировано (Reserved). Зарезервировано (6 бит) для будущего использования и должно устанавливаться в ноль. Из них два (5-й и 6-й) уже определены:

– CWR (Congestion Window Reduced) **–** поле "Окно перегрузки уменьшено" **–** флаг установлен отправителем для того, чтобы указать, что получен пакет с установленным флагом ECE (RFC 3168).

– ECE (ECN-Echo) **–** поле "Эхо ECN" **–** указывает, что данный узел способен на ECN (явное уведомление перегрузки) и для указания отправителю о перегрузках в сети (RFC 3168).

Флаги (управляющие биты, control bits). Это поле содержит 6 битовых флагов:

– URG (Urgent pointer field is significant). Поле «Указатель важности" задействовано.

– ACK (Acknowledgement field is significant). Поле «Номер подтверждения» задействовано.

– PSH (Push function). Инструктирует получателя протолкнуть данные, накопившиеся в приемном буфере, в приложение пользователя.

– RST (Reset the connection). Оборвать соединения, сбросить буфер (очистка буфера).

– SYN (Synchronize sequence numbers). Синхронизация номеров последовательности.

– FIN (Final, бит). Флаг, будучи установленным, указывает на завершение соединения (FIN bit used for connection termination).

Окно (Window). В этом поле содержится число, определяющее в байтах размер данных, которые отправитель готов принять.

Псевдозаголовок (Pseudo header). TCP-заголовок не содержит информации об адресе отправителя и получателя, поэтому даже при совпадении порта получателя нельзя с точностью сказать, что сообщение пришло в нужное место. Поскольку назначением протокола TCP является надежная доставка сообщений, то этот момент имеет принципиальное значение. Эту задачу можно было решить разными способами. Самый очевидный добавить информацию об адресе назначения в заголовок TCP, однако это, во-первых, приводит к дублированию информации, что снижает долю полезной информации, переносимой TCP-сегментом, а во-вторых, нарушает принцип инкапсуляции модели OSI. Поэтому разработчики протокола решили использовать дополнительный псевдозаголовок (табл. 8.1, 8.2):

Таблица 8.1

TCP – псевдозаголовок IPv4

Биты	0-7	8-15	16-31
0-31	IP-адрес отправителя (Source address)		
32-63	IP-адрес получателя (Destination address)		
64-95	0 0 0 0 0 0 0 0	Протокол (Protocol)	Длина TCP-сегмента (TCP length)

Таблица 8.2

TCP – псевдозаголовок IPv6

Биты	0-23	24-31
0-96	IP-адрес отправителя (Source address)	
128-224	IP-адрес получателя (Destination address)	
256	Длина TCP-сегмента (TCP length)	
288	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Протокол верхнего уровня (Next header)

– Протокол (Protocol)/Протокол верхнего уровня (Next header) – содержит в себе значение 6 (000000110 в двоичном виде, 0x6 – в шестнадцатеричном) – идентификатор TCP-протокола.

– Длина TCP-сегмента (TCP length) – содержит в себе длину TCP-сегмента в байтах (TCP-заголовок + данные; длина псевдозаголовка не учитывается).

Псевдозаголовок не включается в TCP-сегмент. Он используется для расчета контрольной суммы перед отправлением сообщения и при его получении (получатель составляет свой псевдозаголовок, используя адрес хоста, с которого пришло сообщение, и собственный адрес, а затем считает контрольную сумму).

Контрольная сумма (Checksum). Поле контрольной суммы **–** это 16-битное дополнение к сумме всех 16-битных слов заголовка (включая псевдозаголовок) и данных. Если сегмент, по которому вычисляется контрольная сумма, имеет длину не кратную 16-ти битам, то длина сегмента должна увеличиться до кратной 16-ти, за счет дополнения справа нулевых битов заполнения (0). Биты заполнения не передаются в сообщении и служат только для расчета контрольной суммы. При расчете контрольной суммы значение самого поля контрольной суммы принимается равным 0.

Ниже приведен пример расчета контрольной суммы:

0×450000304422400080060000 8c7c19acae241e2b.

Для начала разделим 16-ричный заголовок на части, по 16 бит каждая и прибавим друг к другу:

4500 + 0030 + 4422 + 4000 + 8006 + 0000 + 8c7c + 19ac +
+ ae24 + 1e2b = 2BBCF BBD1 = 2 + BBCF = 1011101111010001

Контрольная сумма = дополнение (1011101111010001) =
=0100010000101110 = 442E

Указатель важности (Urgent pointer). 16- битовое значение положительного смещения от порядкового номера в данном сегменте. Это поле указывает порядковый номер октета, которым заканчиваются важные данные. Поле принимается во внимание только для пакетов с установленным флагом URG.

Опции (Options). Могут применяться в некоторых случаях для расширения протокола. Иногда используются для тестирования.

Механизм действия протокола

В отличие от традиционной альтернативы – UDP, который может сразу же начать передачу пакетов, TCP устанавливает соединения, которые должны быть созданы перед передачей данных. TCP соединение можно разделить на 3 стадии: **Установка соединения, Передача данных** и **Завершение соединения**.

• **Установка соединения.** Процесс начала сеанса TCP, обозначаемого как «рукопожатие», состоит из 3 шагов.

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN. Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет для обслуживания нового клиента.

В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK и переходит в состояние SYN-RECEIVED. В случае неудачи сервер посылает клиенту сегмент с флагом RST.

2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK. Если он одновременно получает и флаг ACK, то он переходит в состояние ESTABLISHED. Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться. Если клиент не получает ответа в течение 10 секунд, то он повторяет процесс соединения заново.

3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED. В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED. Процесс называется «трехэтапным согласованием» (three way handshake), так как, несмотря на то, что возможен процесс установления соединения с использованием 4 сегментов (SYN в сторону сервера, ACK в сторону клиента, SYN в сторону клиента, ACK в сторону сервера), на практике для экономии времени используются
3 сегмента.

• **Передача данных.** При обмене данными приемник использует номер последовательности, содержащийся в получаемых сегментах, для восстановления их исходного порядка. Приемник уведомляет передающую сторону о номере последовательности байт, до которой он успешно получил данные, включая его в поле Номер подтверждения. Все получаемые данные, относящиеся к промежутку подтвержденных последовательностей, игнорируются. Если полученный сегмент содержит номер последовательности больший, чем ожидаемый, то данные из сегмента буферизируются,

но номер подтвержденной последовательности не изменяется. Если впоследствии будет принят сегмент, относящийся к ожидаемому номеру последовательности, то порядок данных будет автоматически восстановлен исходя из номеров последовательностей в сегментах.

Для того чтобы передающая сторона не отправляла данные интенсивнее, чем их может обработать приемник, TCP содержит средства управления потоком. Для этого используется поле «окно». В сегментах, направляемых от приемника передающей стороне, в поле «окно» указывается текущий размер приемного буфера. Передающая сторона сохраняет размер окна и отправляет данных не более, чем указал приемник. Если приемник указал нулевой размер окна, то передача данных в направлении этого узла не происходит до тех пор, пока приемник не сообщит о большем размере окна.

В некоторых случаях передающее приложение может явно затребовать протолкнуть данные до некоторой последовательности принимающему приложению, не буферизируя их. Для этого используется флаг PSH. Если в полученном сегменте обнаруживается флаг PSH, то реализация TCP отдает все буферизированные на текущий момент данные принимающему приложению. «Проталкивание» используется, например, в интерактивных приложениях. В сетевых терминалах нет смысла ожидать ввода пользователя после того, как он закончил набирать команду. Поэтому последний сегмент, содержащий команду, обязан содержать флаг PSH, чтобы приложение на принимающей стороне смогло начать её выполнение.

• **Завершение соединения** можно рассмотреть в три этапа:

1. Посылка серверу от клиента флагов FIN и ACK на завершение соединения.

2. Сервер посылает клиенту флаги ответа ACK , FIN, что соединение закрыто.

После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.

8.2. Моделирование процесса установки соединения и передачи данных и завершения соединения по протоколу TCP

Для установки соединения отправитель должен установить соединение с получателем, указав **Порт отправителя** и **Порт получателя**.

Рис. 8.2. Рабочее окно

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению (рис. 8.2).

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них, после чего шаблон с данными окажется в поле **Передаваемая информация.** Все данные, кроме флагов, записываются в 16-ричном формате. Нажмите кнопку «Отправить».

В случае ошибки индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Сформированная информация,** а в поле **Полученная информация** появится ответный сегмент на отправленный запрос.

При успешном выполнении соединения появится соответствующее сообщение. В противном случае очистите поля **Передаваемая информация** и **Сформированная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени.

В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода ко второй части лабораторной работы нажмите кнопку «Часть 2». Для перехода к меню протокола TCP нажмите кнопку «TCP» в верхнем правом углу окна.

Для передачи данных по протоколу TCP отправитель должен отправить данные получателю, указав **Порт отправителя** и **Порт получателя**.

Программа автоматически генерирует числа, которые необходимо отправить. Эти числа в 10-ричном формате выводятся в поле **Передаваемые числа**.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них, после чего шаблон с данными окажется в поле **Передаваемая информация.** Все данные вводятся в 16-ричном формате. Нажмите кнопку «Отправить».

В случае ошибки индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Сформированная информация,** а в поле **Полученная информация** появится ответный сегмент на отправленный запрос.

Переданные числа могут быть проверены нажатием кнопки «Проверить».

При успешной передаче появится соответствующее сообщение. В противном случае очистите поля **Передаваемая информация** и **Сформированная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Для перехода к третьей части лабораторной работы нажмите кнопку «Часть 3».

Для перехода к меню протокола TCP нажмите кнопку «TCP» в верхнем правом углу окна.

Для завершения соединения по протоколу TCP отправитель должен завершить соединение с получателем, указав **Порт отправителя** и **Порт получателя.**

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо записать данные в **Шаблоны** и нажать на них, после чего шаблон с данными окажется в поле **Передаваемая информация.** Все данные, кроме флагов, записываются в 16-ричном формате. Нажмите кнопку «Отправить».

В случае ошибки индикатор ошибки загорится красным светом. Нажмите кнопку «Очистить» и соберите кадр заново.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле **Сформированная информация,** а в поле **Полученная информация** появится ответный сегмент на отправленный запрос.

При успешном выполнении завершения соединения появится соответствующее сообщение. В противном случае очистите поля **Передаваемая информация** и **Сформированная информация** и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Для перехода к меню протокола TCP нажмите кнопку «TCP» в верхнем правом углу окна.

8.3. Моделирование проверки данных по протоколу TCP

Полученный пакет должен быть проверен и в случае ошибки должен быть отклонен. Если пакет не содержит ошибок, он должен быть принят. Для запуска лабораторного стенда необходимо открыть вкладку в папке «Протокол TCP», «Лабораторная работа 2».

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для проверки полученных данных необходимо вычислить контрольную сумму **Checksum** для полученных данных, затем сравнить результат с переданной контрольной суммой. Для вычисления контрольной суммы воспользуйтесь калькулятором в правой части окна. В поля **IP Checksum** и **Данные** должны быть введены данные в 16-ричном формате без пробелов. Нажмите кнопку «Вычислить Checksum». В поле **Checksum** появится вычисленный код в 16-ричном формате.

Если правильный пакет был отправлен, то после нажатия кнопки «Получить» кадр переместится в поле **Полученная информация.** В противном случае нажмите кнопку «Отклонить», а в поле **Переданная информация** появится следующий кадр.

Если неправильный пакет был отправлен, то после нажатия кнопки «Получить» индикатор ошибки загорится красным светом.

Полученные данные можно проверить. Для этого переведите данные из 16-ричного формата в 10-ричный, введите полученные числа в поле **Полученные числа,** предварительно разделив их запятыми, и нажмите кнопку «Проверить».

Если числа введены правильно, программа выведет диалоговое окно с надписью «**Правильно!**». Если числа введены неправильно, программа выведет диалоговое окно с надписью «**Неправильно!**». В этом случае исправьте значения чисел и нажмите кнопку «Проверить».

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Если лабораторная работа не выполнена, то после нажатия этой кнопки появится диалоговое окно с надписью **«Лабораторная работа не закончена».**

Если лабораторная работа выполнена правильно, то после нажатия кнопки «Отчет» появится окно для внесения имени. В появившемся окне введите ваше имя и нажмите **ОК.** Данные отображаются в .doc файле, который включает в себя: название протокола, номер и название лабораторной работы, имя и фамилия студента, дата выполнения работы, а также будет приведено изображение кадра, в зависимости от цели работы. Сохраните файл и поместите его в любую папку.

Для перехода к меню протокола TCP нажмите кнопку «TCP» в верхнем правом углу окна.

Контрольные вопросы

1. На каком уровне OSI работает протокол TCP?
2. Какие функции выполняет протокол TCP?
3. Каков механизм действия данного протокола?
4. Поясните формат сегмента протокола TCP.
5. Каким образом протокол «гарантирует» безошибочную передачу данных получателю?

9. МОДЕЛИРОВАНИЕ И АНАЛИЗ ПРОЦЕССА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ UDP

9.1. Общие сведения о протоколе UDP

UDP (User Datagram Protocol) протокол был разработан Дэвидом П. Ридом в 1980 году и официально определен в RFC 768. Протокол предназначен для обмена дейтаграммами между процессами компьютеров, входящих в единую сеть с коммутацией пакетов, и является одним из ключевых элементов **Internet Protocol Suite** (более известного как **TCP/IP**). С помощью UDP компьютерные приложения могут посылать дейтаграммы (сообщения) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи.

UDP использует простую модель передачи и предоставляет ненадежный сервис, вследствие чего дейтаграммы могут исчезнуть, задержаться, дублироваться либо прийти не по порядку. UDP подразумевает, что проверка ошибок и исправление либо не необходимы, либо должны исполняться в приложении. Чувствительные ко времени приложения часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться невозможным в системах реального времени.

Природа UDP как протокола без сохранения состояния также полезна для серверов, отвечающих на небольшие запросы от огромного числа клиентов, например потоковые мультимедийные приложения (IPTV, VoIP и многие онлайн-игры).

Служебные порты

UDP-приложения используют дейтаграммные сокеты для установки соединения между хостами. Приложение связывает сокет с его конечной точкой передачи данных, которая является комбинацией IP-адреса и порта службы. Порт – это программная структура, определяемая номером порта – 16-битным целочисленным значением (от 0 до 65535). Порт 0 зарезервирован, хотя и является допустимым значением порта источника в случае, если процесс-отправитель не ожидает ответных сообщений.

IANA (Internet Assigned Numbers Authority) разбила номера портов на три группы:

– Порты с номерами от 0 до 1023 используются для обычных, хорошо известных служб.

– Порты с номерами от 1024 до 49151 предназначены для зарегистрированных IANA служб.

– Порты с 49152 по 65535 – динамические и могут быть использованы для любых целей, поскольку официально не разработаны для какой-то определенной службы.

Зарегистрирован ряд портов для стандартного применения, например:

Номер порта	Обозначение	Назначение
1397	Audio-activmail	Активная звуковая почта
1398	Video-activmail	Активная видеопочта
5002	RFE	Радио-Ethernet
6000-6063	X11	Система X Window
7008	AFS3-update	Сервер-сервер актуализация

Структура пакета

UDP не предоставляет никаких гарантий доставки сообщения для протокола верхнего уровня и не сохраняет состояния отправленных сообщений. По этой причине UDP иногда называют Unreliable Datagram Protocol.

UDP обеспечивает многоканальную передачу и проверку целостности заголовка и существенных данных. На рис. 9.1 показан формат UDP дейтаграмм.

Рис. 9.1. Формат UDP дейтаграмм

Заголовок UDP состоит из четырех полей, каждое по 2 байта (16 бит). Два из них необязательны к использованию в IPv4, в то время как в IPv6 необязателен только порт отправителя.

– Порт отправителя (Source Port, 16 бит). Данное поле в случае необходимости содержит номер порта, с которого был отправлен пакет (например, отправитель ожидает ответа). Если поле не используется, оно заполняется нулями.

– Порт получателя (Destination Port, 16 бит). Порт компьютера, на который был отправлен пакет.

– Длина дейтаграммы (Length, 16 бит). Длина (в байтах) данной дейтаграммы, включая заголовок и данные. Минимальная длина равна длине заголовка – 8 байт. Максимальный размер поля составляет 65535 байт (8 байт на заголовок и 65527 на данные). Фактический предел для длины данных при использовании IPv4 – 65507 (помимо 8 байт на UDP-заголовок требуется еще 20 на IP-заголовок). Максимальное значение для IPv6 составляет 4,294,967,295 байт (2^32 – 1), из которых 8 байт соответствуют заголовку, а остальные 4,294,967,287 байт – данным.

– Контрольная сумма (Checksum, 16 бит). Контрольная сумма UDP-пакета представляет собой побитное дополнение 16-битной суммы 16-битных слов (аналогично TCP).

Расчёт контрольной суммы

Перед расчетом контрольной суммы в конце UDP-сообщения добавляются нулевые биты до длины, кратной 16 битам (псевдозаголовок и добавочные нулевые биты не отправляются вместе с сообщением). Поле контрольной суммы в UDP-заголовке во время расчета контрольной суммы принимается за 0.

Для расчета контрольной суммы псевдозаголовок и UDP-сообщение разбивается на слова (1 слово = 2 байта = 16 бит). Затем рассчитывается поразрядное дополнение до единицы суммы всех слов с поразрядным дополнением. Результат записывается в соответствующее поле в UDP-заголовке.

Нулевое значение контрольной суммы зарезервировано и означает, что дейтаграмма не имеет контрольной суммы. В случае если вычисленная контрольная сумма получилась равной нулю, поле заполняют двоичными единицами.

При получении сообщения получатель считает контрольную сумму заново (уже учитывая поле контрольной суммы), и если в результате получится двоичное число из шестнадцати единиц, то контрольная сумма считается сошедшейся. Если сумма не сходится, дейтаграмма уничтожается.

Ниже приведен пример расчета контрольной суммы нескольких 16-битных слов: 0x398a, 0xf802, 0x14b2, 0xc281. Находим их сумму с поразрядным дополнением.

0×398a + 0×f802 = 0×1318c -> 0×318d
0×318d + 0×14b2 = 0×0463f -> 0×463f
0×463f + 0×c281 = 0×108c0 -> 0×08c1

Теперь находим поразрядное дополнение до единицы полученного результата:

0×08c1 = 0000 1000 1100 0001 -> 1111 0111 0011 1110 =
=0×f73e или, иначе – 0×ffff – 0×08c1 = 0×f73e.

Псевдозаголовок для IPv4

Если UDP работает над IPv4, контрольная сумма вычисляется при помощи псевдозаголовка, который содержит некоторую информацию из заголовка IPv4. Псевдозаголовок не является настоящим IPv4-заголовком, используемым для отправления IP-пакета. В табл. 9.1 приведен псевдозаголовок, используемый только для вычисления контрольной суммы.

Таблица 9.1

Псевдозаголовок, используемый
для вычисления контрольной суммы

Биты	0 – 7	8 – 15	16 – 23	24 – 31
0	Адрес источника			
32	Адрес получателя			
64	Нули	Протокол	Длина UDP	
96	Порт источника		Порт получателя	
128	Длина		Контрольная сумма	
160+	Данные			

Адреса источника и получателя берутся из IPv4-заголовка. Значения поля Протокол для UDP равно 17. Поле Длина UDP соответствует длине заголовка и данных.

Вычисление контрольной суммы для IPv4 необязательно. Если она не используется, то значение равно 0.

Псевдозаголовок для IPv6

При работе UDP над IPv6 контрольная сумма обязательна. При вычислении контрольной суммы также используется псевдозаголовок, имитирующий реальный IPv6-заголовок (табл. 9.2).

Таблица 9.2

Заголовок протокола IPv6

Биты	0 – 7	8 – 15	16 – 23	24 – 31
0	Адрес источника			
32				
64				
96				
128	Адрес получателя			
160				
192				
224				
256	Длина UDP			
288	Нули			Следующий заголовок
320	Порт источника		Порт получателя	
352	Длина		Контрольная сумма	
384+	Данные			

Адрес источника такой же, как и в IPv6-заголовке. Адрес получателя – финальный получатель; если в IPv6-пакете не содержится заголовка маршрутизации (Routing), то это будет адрес получателя из IPv6-заголовка, в противном случае, на начальном узле, это будет адрес последнего элемента заголовка маршрутизации, а на узле-получателе – адрес получателя из IPv6-заголовка. Значение Следующий заголовок равно значению протокола – 17 для UDP. Длина UDP – длина UDP-заголовка и данных.

9.2. Моделирование процесса передачи данных по протоколу UDP

Все данные в Шаблонах представлены в 16-ричном формате (рис. 9.2).

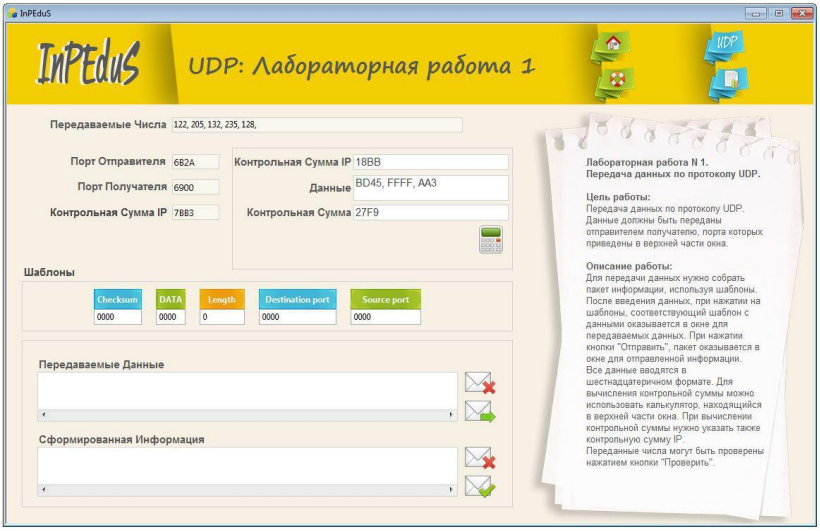

Рис. 9.2. Рабочее окно

Программа автоматически генерирует числа, которые необходимо отправить. Эти числа в 10-ричном формате выводятся в поле Передаваемое число. Каждому передаваемому числу соответствует 1 байт информации.

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для сбора кадра необходимо записать данные в Шаблоны и нажать на них, после чего шаблон с данными окажется в поле Передаваемая информация. Нажмите кнопку «Отправить».

Индикатор ошибки загорится красным светом в случае ошибки при сборе кадра и/или, если Порт отправителя и Порт получателя были указаны неверно. Нажмите кнопку «Очистить» и соберите кадр заново.

Для вычисления Checksum-кода используйте калькулятор в правой части окна. В поля IP Checksum и Данные должны быть введены данные в 16-ричном формате без пробелов. Нажмите кнопку «Вычислить Checksum». Checksum-код также выдается в 16-ричном формате.

Если кадр собран правильно, то после нажатия кнопки «Отправить» он переместится в поле Сформированная информация. Переданные числа могут быть проверены нажатием кнопки «Проверить». Если числа введены правильно, программа выведет диалоговое окно с надписью «Правильно!». Если числа введены не-

правильно, программа выведет диалоговое окно с надписью «Неправильно!». В этом случае очистите поля Передаваемая информация и Сформированная информация и повторите необходимые шаги.

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Для перехода к меню протокола UDP нажмите кнопку «UDP» в верхнем правом углу окна.

9.3. Моделирование проверки полученных данных по протоколу UDP

Полученный пакет должен быть проверен и в случае ошибки должен быть отклонен. Если пакет не содержит ошибок, он должен быть принят. Для запуска лабораторного стенда необходимо открыть вкладку в папке «Протокол UDP», «Лабораторная работа 2».

В правой части окна представлена цель данной лабораторной работы, а также краткое описание по выполнению.

Для проверки полученных данных необходимо вычислить контрольную сумму Checksum для полученных данных, затем сравнить результат с переданной контрольной суммой. Для вычисления контрольной суммы воспользуйтесь калькулятором в правой части окна. В поля IP Checksum и Данные должны быть введены данные в 16-ричном формате без пробелов. Нажмите кнопку «Вычислить Checksum». В поле Checksum появится вычисленный код в 16-ричном формате.

Если правильный пакет был отправлен, то после нажатия кнопки «Получить» кадр переместится в поле Полученная информация. В противном случае нажмите кнопку «Отклонить», а в поле Переданная информация появится следующий кадр.

Если неправильный пакет был отправлен, то после нажатия кнопки «Получить» индикатор ошибки загорится красным светом.

Полученные данные можно проверить. Для этого переведите данные из 16-ричного формата в 10-ричный, введите полученные числа в поле Полученные числа, предварительно разделив их запятыми, и нажмите кнопку «Проверить».

Если числа введены правильно, программа выведет диалоговое окно с надписью «Правильно!». Если числа введены неправильно, программа выведет диалоговое окно с надписью «Неправильно!». В этом случае исправьте значения чисел и нажмите кнопку «Проверить».

Программа имеет возможность сохранения данных выполненной работы. Для этого нажмите кнопку «Отчет».

Контрольные вопросы

1. Каково назначение протокола UDP?
2. Гарантирует ли этот протокол надежную передачу данных получателю?
3. Какова структура дейтаграммы UDP?
4. Что такое дейтаграммный сокет?
5. В чем отличия протокола UDP от протокола TCP?

ЛИТЕРАТУРА

1. Семенов Ю.А. Алгоритмы телекоммуникационных сетей. Ч.1. Алгоритмы и протоколы каналов и сетей передачи данных: учеб. пособие / Ю.А. Семенов. – М. : Интернет-университет информационных технологий (ИНТУИТ), 2016. – 448 с. [Электрон. ресурс]. – Режим доступа: http://www.iprbookshop.ru/62806.html

2. Семенов Ю.А. Протоколы и алгоритмы маршрутизации в Internet. Ч.1. Алгоритмы и протоколы каналов и сетей передачи данных: учеб. пособие / Ю.А. Семенов. – М.: Интернет-университет информационных технологий (ИНТУИТ), 2016. – 355 с. [Электрон. ресурс]. – Режим доступа: http://www.iprbookshop.ru/62826.html

3. Берлин А.Н. Основные протоколы Интернет: учеб. пособие / А.Н. Берлин. – М. : Интернет-университет информационных технологий (ИНТУИТ), 2016. – 265 с. [Электрон. ресурс]. – Режим доступа: http://www.iprbookshop.ru/52181.html

4. Ковган Н.М. Компьютерные сети: учеб. пособие / Н.М. Ковган. – Минск. Республиканский институт профессионального образования (РИПО), 2014. – 469 с. [Электрон. ресурс]. – Режим доступа: http://www.iprbookshop.ru/67638.html

5. Васин Н.Н. Построение сетей на базе коммутаторов и маршрутизаторов: учеб. пособие / Н.Н. Васин. – М. : Интернет-университет информационных технологий (ИНТУИТ), 2016. – 434 с. [Электрон. ресурс]. – Режим доступа: http://www.iprbookshop.ru/52162.html

6. Галушка В.В. Сети и системы передачи информации: учеб. пособие / В.В. Галушка. – Ростов-на-Дону: ДГТУ, 2016. –258 с.

7. Калинкина Т.И. Телекоммуникационные и вычислительные сети. Архитектура, стандарты и технологии: учеб. пособие / Т.И.

Калинкина, Б.В. Костров, В.Н. Ручкин. – СПб: БХВ-Петербург, 2010. – 234 с.