МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Физический факультет
Кафедра радиофизики

Распределенные системы управления и последовательные шины передачи данных

Методические указания к лабораторной работе № 4
практикума ТСАНИ

Новосибирск
2018

УДК 53:004.9(075.8)
ББК В185.505
Ф278

Рецензент
Бехтенев Е. А.

Ф278 ***Фатькин, Г. А.***
Распределенные системы управления и последовательные шины передачи данных : метод. указ. к лаб. работе № 4 практикума ТСАНИ / Г. А. Фатькин, А. Н. Панов, В. В. Орешонок. Новосиб. гос. ун-т. – Новосибирск : РИЦ НГУ, 2018. – 28 с.

Лабораторная работа № 4 посвящена изучению особенностей построения распределенных систем управления на основе последовательных шин передачи данных. В методических указаниях описываются применяемые на настоящий момент последовательные шины данных и на примере шины I^2C рассматриваются основные принципы их работы.

В задачу студента входит изучение шины I^2C, разработка софт-контроллера шины и работа с представленными на шине устройствами (термодатчик, регистр-индикатор). Студенту также предлагается ознакомиться с процедурами синхронизации и арбитража при работе нескольких контроллеров на одной шине для разрешения конфликтов в распределенных системах управления.

Ответственная за выпуск
Тенекеджи О. А.

Введение

В задачах автоматизации часто возникает необходимость создавать распределенные системы управления и сбора данных. Такие системы применяются для контроля параметров установки (температуры элементов, токов и напряжений питания и др.) в тех случаях, когда требуется собирать данные с многих датчиков и / или управлять большим числом устройств, находящихся на значительном удалении друг от друга.

Параллельные шины данных, широко применяющиеся в магистрально-модульных системах, сложно использовать для решения таких задач, поскольку возникают сложности с помехами и синхронизацией сигналов. Для передачи данных между элементами распределенных систем часто прибегают к последовательным шинам передачи данных, радиочастотным и оптоволоконным каналам. Мы будем рассматривать только последовательные шины передачи данных.

В этой работе предлагается ознакомиться с последовательными шинами передачи данных, способом реализации протокола шины программным путем и некоторыми особенностями распределенных систем.

Задания и весь материал в этой работе делятся на основные и дополнительные. Основной материал и задания являются обязательными для всех студентов, а материал и задания, помеченные звездочкой (*) – дополнительными.

Обзор последовательных шин

Шины передачи данных служат для связи отдельных устройств между собой. Они различаются аппаратной реализацией и протоколом. Аппаратная реализация последовательных шин обычно представляет собой один или несколько проводов, которые используются для питания устройств и передачи данных. Часто протокол разделяет устройства на ведущие (инициирующие большинство обменов данными), также называемые master, host, и ведомые (slave). Ведущее устройство обычно называется контроллером шины. Если шина позволяет иметь несколько контроллеров, то лишь один из них активен в определенный момент времени. Для выбора активного контроллера существует специальная процедура арбитража.

Существующее множество последовательных шин можно разделить на два класса: шины, применяющиеся для связи устройств внутри ЭВМ, и шины, использующиеся для связи ЭВМ с внешними устройствами.

В этой работе мы не будем рассматривать последовательные шины первого класса, такие как PCI Express, HyperTransport, RapidIO и т. д., поскольку из-за большой частоты они имеют очень короткое (до 1 м) ограничение по длине линии, ввиду чего работать с удаленными устройствами не представляется возможным.

Сравнительные характеристики последовательных шин

Название	Скорость передачи данных	Максимальная удаленность устройства [1]	Количество адресуемых устройств [2]	Количество проводов в шине
USB (Universal Serial Bus)	0,01–5 Гбит/с	5 м	1 ведущий, 127 ведомых	4
CAN	0,125–1 Мбит/с	40–5000 м	64 ведущих, 64 ведомых	2–3
EIA485 RS485	0,1–35 Мбит/с	10–1200 м	1 ведущий, до 31 ведомых	2–3
IEEE1394 (Fire-Wire, Lynx)	0,4–3,2 Гбит/с	4.5–72 м	1 ведущий, 63 ведомых	4–6
SPI [3]	1–70 Мбит/с	1–3 м	1 ведущий, 1 ведомый	4
RS232	0,1–115 Кбит/с	15 м	1 ведущий, 1 ведомый	3–10
I^2C	0,01–5 Мбит/с	1–3 м	ведущих не ограничено, 1008 ведомых [4]	4
Ethernet	1 Кбит/с–10 Гбит/с	100–300 м	2^{32}(IPv4)-2^{64}(IPv6)	2–8

[1] Максимальная удаленность устройства может быть увеличена при помощи использования ретрансляторов или передачи данных по другим каналам (к примеру, оптоволоконным в случае ethernet). Также достаточно часто скорость передачи данных зависит от удаленности устройств.

[2] Действительное количество устройств, которые могут одновременно работать на шине, обычно меньше. Большое количество устройств на шине и, соответственно, большая емкость, приводят к задержкам и рассинхронизации.

[3] На шине SPI возможно подключение большего количества ведомых, однако при этом к каждому ведомому идет свой провод CS (Chip select, выбор устройства).

[4] В действительности из-за проблем с адресацией (за неимением возможности динамически назначать адреса) на шине I^2C редко используется одновременно более десятка ведомых устройств.

Сравнительные характеристики некоторых распространенных шин второго класса приведены в таблице. Описания шин, приведенных в таблице, можно посмотреть в работах [1–7].

Шина I^2C

Для демонстрации основных принципов построения распределенных систем с помощью последовательных шин в этой работе будет использоваться протокол I^2C. Протокол разработан фирмой Philips (ныне NXP Semiconductors) и опубликован в 1992 г. В 1998, 2000, 2007, 2012 и 2014 гг. были выпущены дополнения к стандарту, решающие проблемы адресации, позволяющие работать с быстрыми устройствами и описывающие работу в режиме сохранения энергии (power saving mode). В нашем пособии мы будем довольствоваться стандартной скоростью шины и не будем использовать новые возможности протокола. Более подробно с протоколом I^2C можно ознакомиться в [7].

Физический уровень

На физическом уровне протокол I^2C использует четыре провода: SDA, SCL,V_{DD},V_{SS}. По проводам V_{DD} и V_{SS} подается питание, V_{SS} – земля (0 В), V_{DD} – напряжение питания (например, 3,3 В). SDA (Serial Data / Address) используется для передачи данных и адреса, а SCL (Serial CLock) – тактовой частоты. Низкому уровню соответствуют напряжения от –0,5 В до $0{,}3V_{DD}$, а высокому – от $0{,}7V_{DD}$ до V_{DD}+0,5 В.

Для подключения к шине применяется схема, изображенная на рис. 1. В отсутствие сигналов на шине поддерживается высокий уровень. Если устройство хочет выставить высокий уровень, то оно размыкает ключ, а если низкий – то замыкает. Если на шине хотя бы одно устройство выставило низкий уровень, то шина будет находиться в низком состоянии. Таким образом удается избежать неопределенных состояний шины. На этом же принципе основана процедура арбитража шины.

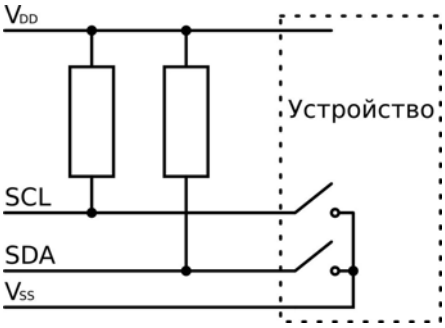

Рис. 1. Физический уровень шины I^2C

Протокол

У каждого устройства на шине есть свой уникальный адрес, задающийся производителем устройства. Ведущее устройство всегда инициирует передачу данных и генерирует тактовый сигнал. Любое устройство, к которому обращается ведущее, считается ведомым (даже если два контроллера шины обращаются друг к другу, то один из них ведущий, а другой – ведомый). Отправитель передает данные по шине данных, а получатель принимает эти данные. Как ведомое, так и ведущее устройства могут выступать в роли отправителя и получателя.

Порядок изменения сигналов на шине

В стандартном скоростном режиме сигналы SCL и SDA имеют частоту до 100 КГц, нижний предел скорости изменения сигнала не устанавливается. На каждый бит данных генерируется по одному тактовому сигналу.

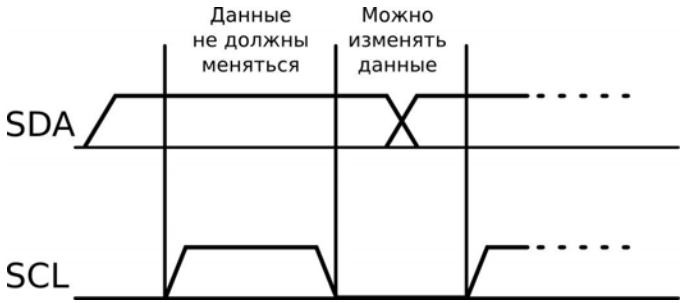

Рис. 2. Порядок изменения сигналов на шине

Данные на линии SDA не должны изменяться в то время, когда тактовый сигнал высокий, и могут меняться в то время, когда тактовый сигнал низкий (рис. 2). Изменение SDA в момент, когда SCL высокий, воспринимается как сигнал START или STOP.

Сигналы START и STOP

Любая транзакция (передача данных) начинается с сигнала START и заканчивается сигналом STOP. Сигнал START определяется как переход линии SDA из высокого состояния в низкое, в то время как линия SCL находится в высоком состоянии. Сигнал STOP определяется как переход SDA из низкого состояния в высокое, в то время как линия SCL находится в высоком состоянии (рис. 3).

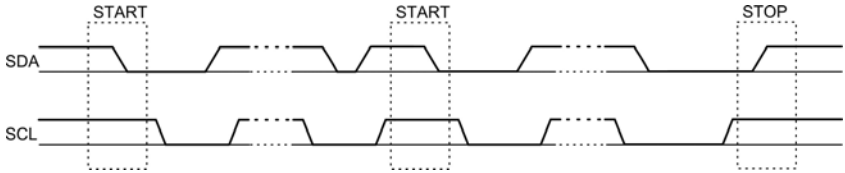

Рис. 3. Сигналы START и STOP

Шина считается занятой после того, как подан сигнал START, и свободной – через 4,7 мкс после того, как подан сигнал STOP. Сигналы START и STOP всегда генерируются ведущим устройством. Шина остается занятой, если вместо сигнала STOP подается повторный сигнал START.

Сигналы ACKNOWLEDGE и NOT ACKNOWLEDGE

Информация на шине I^2C передается байтами по 8 битов. Данные передаются начиная со старшего бита и заканчивая младшим. После принятия каждого переданного байта получатель должен выставить бит подтверждения. Это сигнализирует отправителю о том, что байт был успешно получен. За генерацию тактовых сигналов как при приеме, так и при передаче отвечает ведущий.

Бит подтверждения ACKNOWLEDGE (ACK) выставляется получателем на 9-й тактовый сигнал. После передачи 8 битов отправитель отпускает линию SDA. Если прием байта прошел успешно, то получатель должен установить SDA в низкое положение.

Если же линия остается в высоком положении, то подтверждения приема нет, или можно говорить, что получен сигнал неподтверждения NOT ACKNOWLEDGE (NACK). Этому может быть несколько причин:

– на шине нет приемника с соответствующим адресом;

– приемник не может участвовать в транзакции, так как выполняет какую-нибудь неотложную операцию;

– во время транзакции приемник получил данные, которые он не понимает, или команду, которую не может обработать;

– приемник не может принимать больше данных в текущей транзакции;

– ведущий в режиме приема сигнализирует ведомому о конце передачи.

При отсутствии подтверждения ведущий может либо повторно начать транзакцию, передав сигнал START, либо прервать ее, подав сигнал STOP.

Если получатель не может получить или отправить следующий байт (к примеру, если он занят обработкой сигнала), то он может установить линию SCL в низкое положение. Пока SCL находится в низком положении, ведущий приостанавливается. После того, как ведомый отпускает SCL, он посылает сигнал ACK, и передача данных может продолжаться.

Транзакция

Рис. 4. Формат адресной посылки

Обращение к устройству начинается с адресной посылки (рис. 4), в которой указывается адрес ведомого устройства и вид транзакции. Транзакции бывают двух видов: Read (чтение) и Write (запись). Тип транзакции определяется битом R/W в адресной посылке. Если бит R/W находится в высоком положении, то это чтение, если в низком – запись.

В случае чтения ведущее устройство инициирует транзакцию, ведомое передает ACK, затем – запрошенные данные, получив которые ведущее выставляет ACK после каждого байта, кроме последнего, а после последнего байта передает NACK. В случае записи ведущее устройство инициирует транзакцию и передает данные, а ведомое лишь передает сигналы ACK.

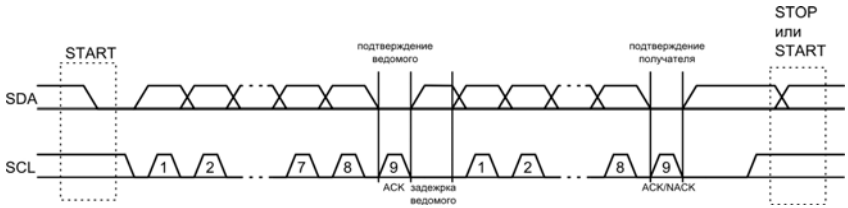

Рис. 5. Транзакция

На рис. 5 показан пример транзакции, состоящей из 2 байтов. Ведущее устройство начинает транзакцию, передавая сигнал START. Затем передает 8 бит и на 9-й ожидает подтверждения. Далее ведомый может захватить шину SCL и установить ее в низкое положение на время обработки данных. После отпускания шины SCL передающее устройство (может быть как ведомым, так и ведущим) начинает передачу еще 8 бит, после чего получатель выставляет (или не выставляет) подтверждение. После этого ведущий дает STOP либо повторный START (при необходимости передачи дополнительных данных).

Передача данных производится в порядке, представленном ниже.

1. Ведущий посылает сигнал START.
2. Ведущий посылает адресную посылку (см. рис. 4).
3. Ведущий получает сигнал ACK.
4. Ведущий посылает или получает данные.

5. Ведущий получает (при посылке данных) или выставляет (при приеме данных) сигнал ACK (или NACK), при необходимости шаги 4–5 повторяются.
6. Ведущий посылает сигнал STOP и освобождает шину или посылает повторный сигнал START, шаги 1–5 повторяются.

Конкретная последовательность полученных и отправленных данных определяется типом ведомого. Во многих I^2C-устройствах есть несколько внутренних регистров. Для работы с такими устройствами ведущий сначала передает адресную посылку на запись с субадресом (номером ячейки памяти, который он хочет прочитать или записать), ведомое устройство запоминает субадрес. Если нужно записать данные, то их обычно можно записывать в той же транзакции, в случае чтения производится отдельная транзакция для чтения без передачи субадреса.

При потоковой записи (чтении) в одной транзакции последовательно передается несколько байт, которые записываются в последовательных ячейках памяти в ведомом устройстве.

Залипание шины

В некоторых случаях может произойти «залипание» шины SDA. Если шину SDA кто-то из ведомых держит в низком состоянии, то контроллер шины может послать девять тактовых сигналов. Устройство, которое держит шину, должно освободить ее на один из девяти сигналов. Если освобождения не произошло или залипла шина SCL, то следует перезагрузить устройства на шине (к примеру, отключив и включив питание).

Описание устройств

Способ подключения к шине

На рис. 6 изображена схема подключения модуля ввода-вывода NI PXI-6251, с помощью которого производится управление шиной I^2C. Посредством терминального блока модуль ввода-вывода может быть подключен к двум независимым шинам: I^2C-1 и I^2C-2. Разъемы шин I^2C расположены на терминальном блоке справа вверху, их очередность иллюстрирует рис. 7. Чтобы терминальный блок работал в режиме подключения I^2C, следует в port0/line0:2 записать 111_2.

Шина I^2C управляется с помощью четырех сигналов: SDAout, SCLout, SDAin, SCLin. Сигналы SDAout и SCLout управляют ключами: когда на них подана логическая единица – на соответствующих линиях шины устанавливается высокий сигнал, и наоборот. Сигналы SDAin и SCLin предназначены для чтения состояния на шине. Визуально оценить состояние линий позволяют светодиоды D1 – D4, расположенные слева от разъемов I^2C. Подключение светодиодов для I^2C-2 иллюстрирует рис. 6.

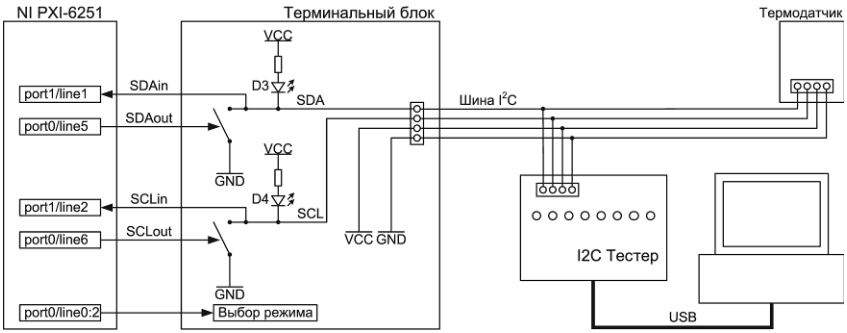

Рис. 6. Схема подключения устройств к шине I^{2}C-2

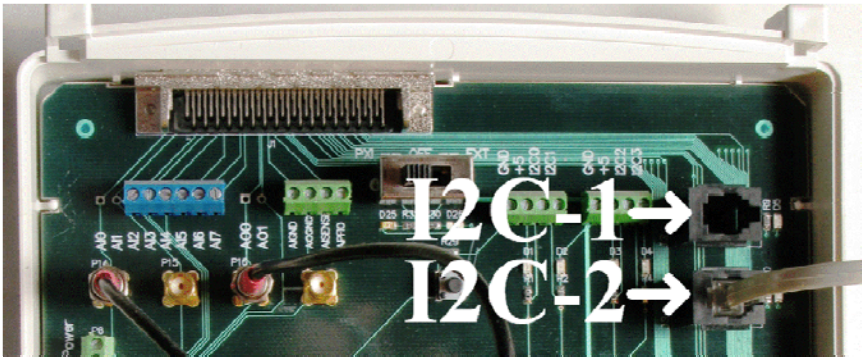

Рис. 7. Порядок расположения разъемов шин I^{2}C-1 и I^{2}C-2 на терминальном модуле

В регистре ввода-вывода сигналы распределены следующим образом:

Сигнал	Регистры I^{2}C-2	Регистры I^{2}C-1
SDAout	port0/line5	port0/line3
SCLout	port0/line6	port0/line4
SDAin	port1/line1	port0/line7
SCLin	port1/line2	port1/line0
Режим	port0/line0:2	port0/line0:2

Тестер I2C

Первым устройством, с которым предлагается поработать, является тестер I2C (рис. 8). Он соединяется с компьютером по шине USB и управляется с помощью программы TesterI2C, рабочее окно которой показано на рис. 9. Подробное описание программы дано в прил. 2.

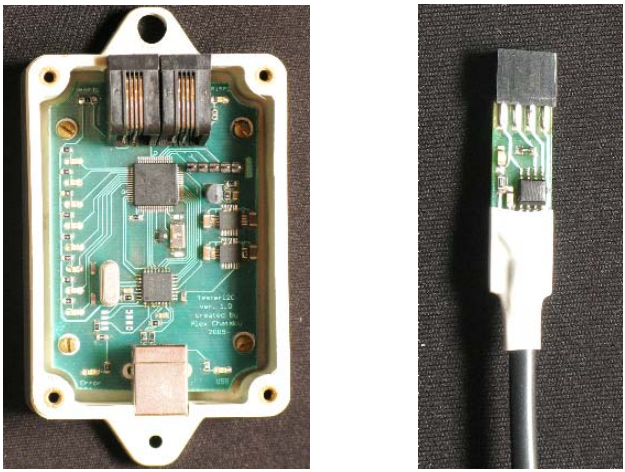

Рис. 8. Фотографии тестера I2C *(слева)* и термодатчика *(справа)*

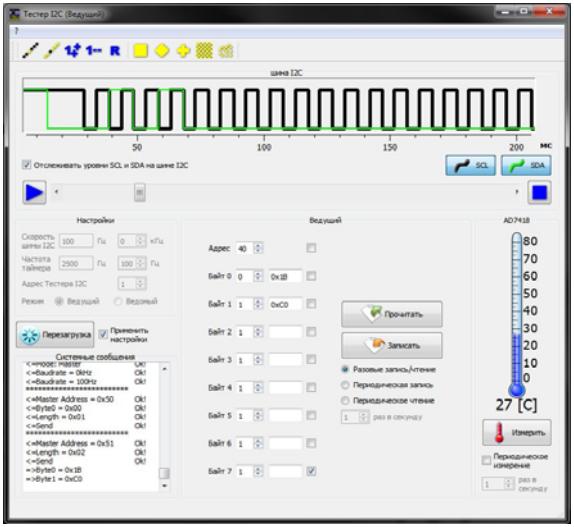

Рис. 9. Окно программы TesterI2C

Тестер может работать как в режиме ведущего, так и в режиме ведомого. Адрес тестера на шине I^2C задается в поле *Адрес Тестера I2C*. В тестере есть восемь регистров записи (субадреса 0x00÷0x07), первый из которых управляет светодиодами на плате тестера. Также в тестере присутствуют восемь регистров чтения (субадреса 0x08÷0x0F), которые по шине I^2C можно только читать, а значение в них задается с помощью программы.

Тестер I2C может работать как контроллер шины. Выбор режима работы осуществляется с помощью программы.

После передачи первой адресной посылки тестеру необходимо передать субадрес внутреннего регистра, к которому идет обращение.

Запись в регистр тестера осуществляется следующим образом: сначала передается адрес ведомого, затем записывается субадрес, после чего передается записываемое слово (рис. 10).

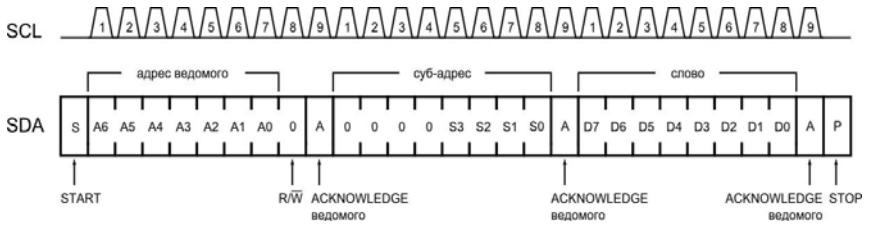

Рис. 10. Временная диаграмма записи в регистр тестера I2C

Итак, для записи в регистр следует:

1) послать сигнал START;
2) передать адресную посылку;
3) получить подтверждение;
4) записать субадрес;
5) получить подтверждение;
6) записать байт;
7) получить подтверждение;
8) послать сигнал STOP.

Чтение из регистра осуществляется следующим образом: передается адрес ведомого, затем записывается субадрес, после чего начинается новая транзакция на чтение; снова передается адрес ведомого, и ведомый выставляет на шину запрошенное слово данных (рис. 11).

Отметим, что после первой транзакции на запись субадрес запоминается в тестере, т. е. если вы захотите два раза подряд прочитать слово из одного и того же регистра, то второй раз передавать субадрес не обязательно.

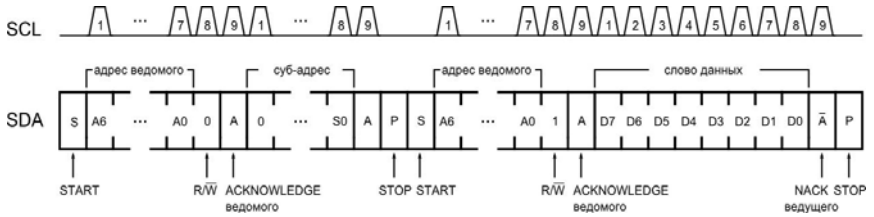

Рис. 11. Временная диаграмма чтения из регистра тестера I2C

Таким образом, для чтения из регистра необходимо:

1) послать сигнал START;
2) передать адресную посылку;
3) получить подтверждение;
4) записать субадрес;
5) получить подтверждение;
6) послать сигнал STOP (этот шаг опускать не следует, т. к. тестер I2C не поддерживает процедуру повторного старта);
7) послать сигнал START;
8) передать адресную посылку;
9) получить подтверждение;
10) получить байт от тестера;
11) послать NACK;
12) подать сигнал STOP.

Термодатчик

В этой работе вам предлагается работать с термодатчиком AD7416/7418 при помощи интерфейса I2C.

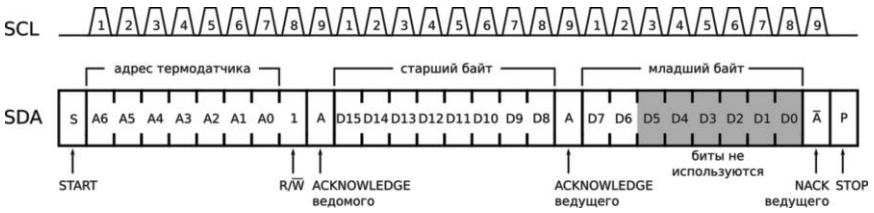

Рис. 12. Временная диаграмма чтения показаний термодатчика

Описание датчика дано в [9]. Его адрес на шине I^2C: 0x28. Термодатчик работает в режиме потокового чтения, т. е. после передачи адресной посылки он выдает два байта данных (сначала старший, затем младший), в которых закодирована температура (рис. 12).

Из двух байтов используются биты с 15 по 6, биты 5–0 не используются. Температура выдается в дополнительном коде с дискретностью 0,25 °С. Соответствие между температурой и кодом приведено ниже.

Температура	Код
–128 °С	10 0000 0000
–125 °С	10 0000 1100
–25 °С	11 1001 1100
–0,25 °С	11 1111 1111
0 °С	00 0000 0000
+0,25 °С	00 0000 0001
+10 °С	00 0010 1000
+25 °С	00 0110 0100
+125 °С	01 1111 0100
+127 °С	01 1111 1100

Основные задания

Работа с тестером I2C

Перед началом работы установите тестер I2C в режим *ведомый* с помощью программы.

Если у вас возникают проблемы с этим или следующим заданием, рекомендуем обратиться к прил. 1, 2.

1. Напишите программу, которая будет записывать слово в регистр тестера I2C, и убедитесь в правильности ее работы по светодиодам тестера. Для этого рекомендуется реализовать контроллер шины со следующими функциями:
   – старт транзакции;
   – стоп транзакции;
   – передача байта данных;
   – прием байта данных;
   – получение подтверждения (ACKNOWLEDGE).

2. Реализуйте бегущую единицу. Светодиоды регистра тестера должны зажигаться поочередно с интервалом около 0,25 с. По достижении седьмого диода его значение переходит нулевому, и цикл повторяется.

3. Запишите в регистры чтения тестера свое имя с помощью программы TesterI2C, вычитайте его по шине I^2C и выведите на экран. Сокращенная версия ASCII таблицы приведена ниже.

Код	Символ	Код	Символ	Код	Символ
32	Пробел	65	A	78	N
33	!	66	B	79	O
46	.	67	C	80	P
48	0	68	D	81	Q
49	1	69	E	82	R
50	2	70	F	83	S
51	3	71	G	84	T
52	4	72	H	85	U
53	5	73	I	86	V
54	6	74	J	87	W
55	7	75	K	88	X
56	8	76	L	89	Y
57	9	77	M	90	Z

Работа с термодатчиком

1. Считайте температуру с термодатчика и выведите ее на табло. Не забудьте реализовать поддержку отрицательных значений температуры.

2. Выведите график зависимости температуры от времени на экран.

Дополнительный материал *

Зарезервированные адреса

На шине I^2C есть несколько зарезервированных адресов:

Адрес ведомого	Бит R/W	Описание
0000 000	0	Общий вызов
0000 000	1	Байт СТАРТ
0000 001	X	Адрес CBUS
0000 01X	X	Зарезервированы
0000 1XX	X	Зарезервированы
1111 XXX	X	Зарезервированы

Общий вызов – обращение ко всем устройствам. Он может использоваться для перезагрузки устройств. Устройства могут не принимать общий вызов. При получении байта СТАРТ устройства не должны посылать подтверждения. Остальные зарезервированные адреса нам не интересны, они служат для 10-битной адресации, совместимости с другими шинами или оставлены для будущего использования.

Синхронизация тактовых сигналов и арбитраж

Если на шине присутствует больше одного контроллера, то они могут начать передачу данных на незанятой шине одновременно. Для того чтобы определить, какой из контроллеров будет совершать транзакцию, контроллеры должны соблюдать процедуры синхронизации тактовых сигналов и арбитража.

Синхронизация

Два контроллера шины могут иметь несколько различающуюся тактовую частоту. Для того чтобы установить общую тактовую частоту на шине, применяется схема логического И (рис. 13).

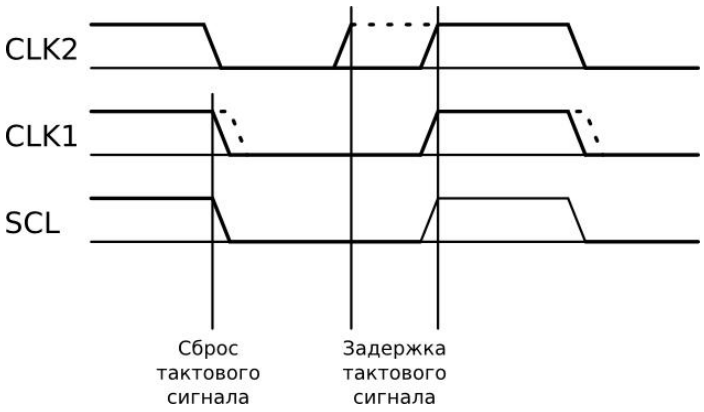

Рис. 13. Синхронизация

Если один из контроллеров устанавливает на линии SCL низкий уровень, то второй должен сбросить тактовый сигнал. Если один из контроллеров хочет установить высокий сигнал на линии SCL, а второй держит на нем низкий, то первый контроллер должен задержать свой тактовый сигнал. Таким образом, на шине SCL поддерживается тактовый сигнал с низким уровнем, определяющимся контроллером с самым длинным низким и с высоким уровнем, определяющимся самым коротким высоким.

Арбитраж

Допустим, что два контроллера одновременно подают команду START. Тогда для того, чтобы определить, какой из контроллеров владеет шиной, применяется процедура арбитража. Арбитраж происходит по битам. Во время, когда сигнал SCL высокий, каждый контроллер проверяет, совпадает ли сигнал на линии SDA с тем, который он устанавливает. Как только контроллер хочет установить на SDA высокий сигнал, но считывает низкий, он теряет арбитраж и должен прекратить транзакцию. Другой контроллер продолжает свою транзакцию.

Пример процедуры арбитража, где контроллер № 2 продолжает транзакцию, а контроллер № 1 теряет арбитраж, показан на рис. 14. В целом если оба контроллера хотят одновременно совершить одинаковую транзакцию, они совершат ее успешно.

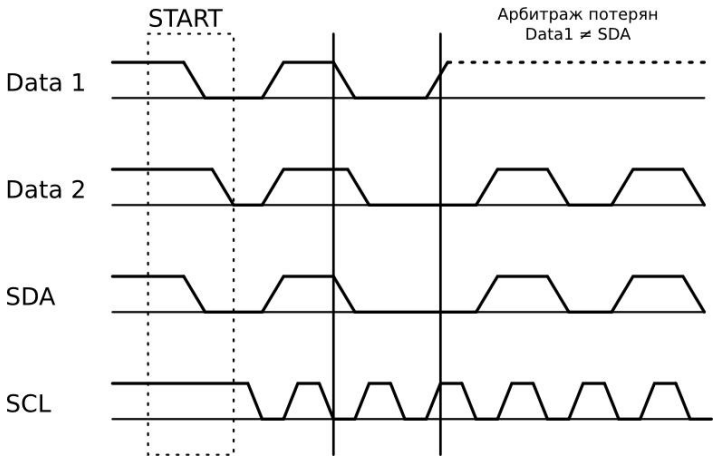

Рис. 14. Арбитраж

После того как контроллер теряет арбитраж, он должен прекратить генерировать сигналы на линиях SDA и SCL. Он может повторить попытку провести транзакцию после того, как будет подан сигнал STOP.

Дополнительные задания*

1. Переведите тестер I2C в режим ведущего со скоростью обмена 10 кГц и включите периодическое измерение температуры. Можно перезагрузить шину, отключив и включив питание на терминальном блоке и нажав кнопку *Перезагрузить* в программе TesterI2C.
2. Реализуйте контроллер с процедурой арбитража и синхронизации.
3. Выведите график температуры от времени на экран.
4. Определите скорость обмена с датчиком температуры и число коллизий (случаев, когда ваша программа и тестер одновременно обращаются к термодатчику) и выведите их на экран.

Приложения

Приложение 1

Пример реализации контроллера шины

В приложении приводятся лишь некоторые функции контроллера, часть из которых вам придется реализовать самостоятельно.

Один из авторов данного методического пособия изучал ассемблер IBM-PC по книге Питера Нортона и Джона Соухе «Язык ассемблера для IBM PC» [8]. В этой книге по непонятной причине (постарались ли редакторы, или наборщики, а может и сам Нортон) практически в каждом листинге программы была ошибка, а то и несколько. Это не позволяло просто переписать программу, не разобравшись в том, что в ней делается, так как она не работала. Для исправления ошибок (написания правильных программ) приходилось осваивать материал. Этот метод предлагается опробовать и вам.

Внимание: в каждом из приведенных ниже листингов программы содержится как минимум по одной ошибке, которые необходимо будет исправить, чтобы программа заработала,.

Для начала создадим функцию, конфигурирующую доступ к сигналам шины I^2C:

```
void initPorts
{
 portMask (0, 0x7F);
 portMask (1, 0x00);

 portOut (0, 0xFF);
 portOut (1, 0);
}
```

Здесь, очевидно, «забыты» круглые скобки после имени функции. Дальнейшие ошибки предлагается искать самостоятельно.

В нашей реализации контроллера предлагается все сложные задачи реализовать в виде отдельных функций. Для начала напишем функцию, которая выводит на шину значения SDA и SCL:

```
int writeBus (int sda, int scl)

{
 unsigned char data;
 data = (sda << 3) | (!scl << 4)
        | (1 << 5) | (1 << 6) | 0x07;
 return portOut (1, data);
}
```

Затем функцию, которая читает значения на шине:

```
int readBus (int* sda, int* scl)
{
 unsgned char val0, val1;
 if (!portIn (0, &val0)) && (!portIn (1, &val1))
 {
    *sda = (val0 >> 7) & 1;
    *scl = (val1 >> 0) & 2;
    return 0;
 }
 else
    return -1;
}
```

Зададим константу DELAY, которая будет представлять задержку в секундах между операциями. Объявим ее с помощью подстановки #define в начале программы:

```
#deifne DELAY 0.025
```

Теперь можно реализовать функцию для посылки сигнала START:

```
itn sendStart ()
{
 writeBus (1, 1);
 Delay (DELAY);
 writeBus (0, 1);
 Delay (DELAY);
 writeBus (0, 0);
 Delay (DELAY);
 return 0;
}
```

Аналогично реализуется посылка сигнала STOP. Сделайте это самостоятельно.

Функция, посылающая бит на шину, может быть реализована следующим образом:

```
int sendBit (int bit)
{
 writeBus (currentSDA(), 0);
 writeBus (0, bit);
 Delay (DELAY);
 writeBus (currentSDA(), 1);
 Delay (DELAY);
 writeBus (currentSDA(), 0);
 Delay (DELAY);
 retrn 0;
}
```

Функцию currentSDA(), читающую значение на шине SDA, предлагается реализовать самостоятельно

Посылку байта легко организовать в виде цикла, состоящего из посылок отдельных битов:

```
int sendByte (int byte)
{
 for (int i=0; i<8; i++)
    sendBit ((byte >> (i-7)) & 1);
 return 0;
}
```

Прием сигнала подтверждения можно реализовать следующим образом:

```
int getACK ()
{
 writeBus (currentSDA(), 0);
 writeBus (1, 0);
 Delay (DELAY);
 writeBus (1, 0);
 Delay (DELAY);
 do
    Delay (DELAY);
 while (currentSDA() != 0);
 writeBus (currentSDA(), 0);
 Delay (DELAY);
 return 0;
}
```

Заметим, что в этой реализации, программа зациклится если не будет получен сигнал подтверждения (ACK),. Изменить функцию можно таким образом, чтобы при отсутствии подтверждения в течение некоторого времени программа не попадала в бесконечный цикл, а функция getACK( ) возвращала –1 как сообщение об ошибке.

Самостоятельно реализуйте функцию приема байта, например, ее можно назвать int readByte (int* byte). Эту функцию можно оформить как вызов в цикле функции readBit ( ), которая будет читать биты. Для чтения бита достаточно записать в SDA единицу, что приведет к размыканию ключа, после чего можно будет с помощью currentSDA ( ) в нужный момент (при высоком сигнале на SCL) считать передаваемый ведомым бит.

Перед нами уже практически готовый контроллер. К примеру, запись слова в регистр будет выполняться следующей программой:

```
sendStart(); //Посылка сигнала START
getACK(); // Получение подтверждения
sendByte(0xc0); // Адресная посылка
sendByte(0x11); // Значение для записи в регистр
getACK();
sendStop(); // Посылка сигнала STOP
```

Описание программы Tester I2C

TesterI2C – программа, управляющая устройством-тестером, который подключается к USB-порту компьютера. Тестер позволяет отправлять и принимать сообщения по шине I^2C как в режиме ведущего, так и в режиме ведомого, а также отслеживать физические уровни SCL и SDA. Кроме того, используя заготовки сообщений, можно считывать измеренную температуру с термодатчика AD7416/7418, а соединяя несколько тестеров вместе – рисовать различные фигуры на матрице светодиодов ведомого тестера.

Настройки

Элементы настройки программы представлены на рис. 15.

Скорость шины I^2C задается двумя полями. В верхнем поле ввода указывается скорость в килогерцах (от 0 до 200), в нижнем – в герцах (от 1 до 250). В результате скорость шины будет равна сумме этих двух величин. Величина «Частота таймера» необходима для корректного слежения за физическими уровнями SCL и SDA. Она также получается из величин полей скорости. Если в режиме ведущего следует обращать внимание на «Скорость шины I2C», то в режиме ведомого – на «Частоту таймера».

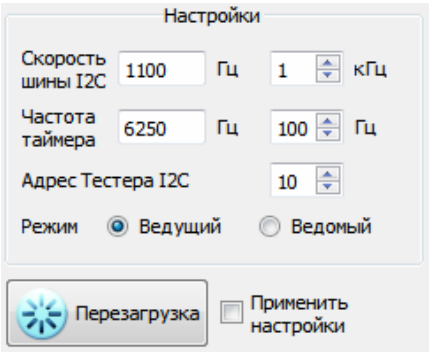

Рис. 15. Элементы настройки

Минимальное возможное значение «Скорости шины I2C» в режиме ведущего 40 Гц, максимально возможное – 200 кГц. Минимальное возможное значение «Частоты таймера» – 25 Гц, максимальное – 6250 Гц. При конфигурировании тестера в режиме ведомого с отслеживанием уровней SCL и SDA в полях ввода следует задавать предполагаемое значение скорости передачи. Например, скорость передачи 10 Гц, тогда следует задать в поле ввода скорости в герцах 10, в поле ввода в килогерцах – 0. Частота таймера будет равна 250. Но если выбрать режим ведущего, то скорость будет равна минимальному значению 40 Гц, а не 10 Гц.

В поле «Адрес Тестера I2C» задается адрес, по которому будет доступен тестер в режиме ведомого. После задания всех настроек необходимо нажать «Применить настройки» для их записи в блок управления. Кнопка «Перезагрузка» используется для перезапуска блока при необходимости.

Режим ведущего

Внешний вид области управления в режиме ведущего показан на рис. 16.

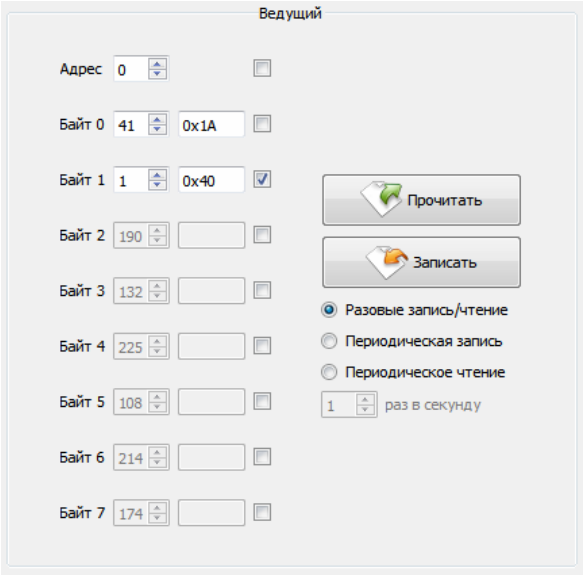

Рис. 16. Область управления в режиме ведущего

Элементы управления позволяют сформировать сообщение для записи или чтения длиной до 8 байт. В поле «Адрес» задается адрес устройства, к которому производится обращение. Поля «Байт 0» – «Байт 7» предназначены для введения отсылаемых значений. Длина сообщения определяется выбранным флажком и варьируется от 0 (отсылается только адрес) до 8. Кнопки «Прочитать» и «Записать» формируют соответственно сообщение для чтения или записи, для этого совершается небольшое преобразование адреса в соответствии со спецификацией шины I^2C. Возможна периодическая запись / чтение с указанием количества сообщений в секунду. Максимальная скорость повторения 20 раз/с.

Режим ведомого

Внешний вид области управления в режиме ведомого показан на рис. 17.

Тестер I2C содержит восемь регистров записи, расположенных по адресам от 0x00 до 0x07, и восемь регистров чтения, расположенных по адресам от 0x08 до 0x0F. Все регистры восьмибитные. Значения регистров чтения можно задавать произвольными, в дальнейшем их содержимое можно будет прочитать по шине I^2C. В центре расположена матрица светодиодов, представляющая собой визуальную интерпретацию содержимого регистров записи. При использовании пакетной записи можно создавать различные изменяющиеся во времени фигуры (см. «Заготовки сообщений»). Кнопкой «Придержать / Отпустить шину» можно приостановить передачу данных. Эта команда опускает уровень SCL до нуля. Обратное действие осуществляется переводом кнопки в первоначальное положение. Кнопка «Пакетная / Однократная запись» определяет тип адресации.

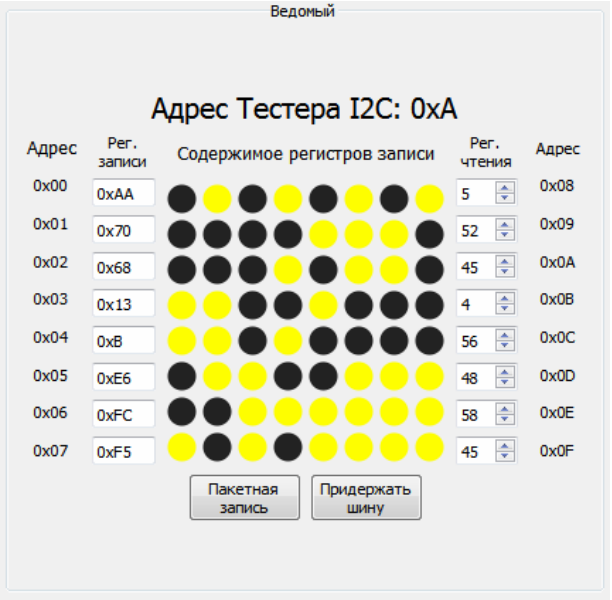

Рис. 17. Область управления в режиме ведомого

При однократной записи структура сообщения следующая: адрес ведомого, адрес регистра записи, новое значение регистра записи. Таким образом, длина сообщения 3 байта. При использовании сообщений большей длины в указанный первым байтом регистр запишутся последовательно все значения передаваемых байтов. В результате со-

держимое регистра записи окажется равным последнему байту в сообщении.

При пакетной записи адрес регистра, в который записывается новое значение, определяется местом положения этого значения в сообщении. Следующий за адресом байт запишется по нулевому адресу, следующий – по первому адресу и т. д. Таким образом, одним сообщением можно изменить содержимое у восьми регистров записи.

Монитор шины I²C

Внешний вид монитора шины I²C представлен на рис. 18.

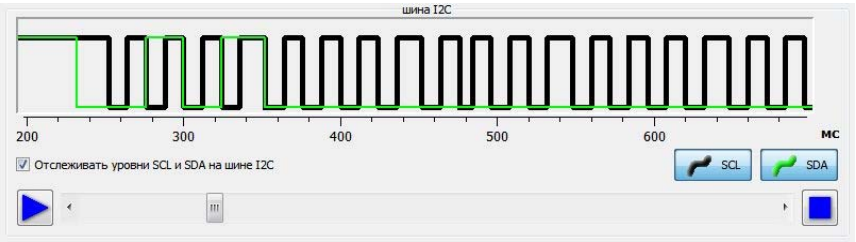

Рис. 18. Область монитора шины I²C

Монитор шины включает в себя область вывода значений SCL и SDA от времени, полосу прокрутки, кнопки «Старт», «Стоп» и кнопки вывода SCL и SDA. Для SCL используется черный цвет, для SDA – зеленый. Время измеряется в миллисекундах. Запуск отслеживания уровней на шине производится при возникновении хотя бы одного события: нажата кнопка «Старт», изменился уровень SCL или SDA. Остановка осуществляется также при возникновении хотя бы одного события: нажата кнопка «Стоп», бездействие шины I²C, переполнение массива данных. Шина считается бездействующей, если в течение 12 периодов SCL не изменилось ни состояние SDA, ни состояние SCL. Массив данных рассчитан на запись 256 событий, каждое событие – это либо изменение SCL, либо изменение SDA, чего вполне хватает для хранения пары сообщений.

Заготовки сообщений

TesterI2C позволяет, используя набор заготовок сообщений, продемонстрировать взаимодействие по шине I²C двух блоков, сконфигурированных один – в режиме ведущего, другой – в режиме ведомого. Заготовки активны только в режиме ведущего. Адрес ведомого тестера указывается в поле «Адрес» ведущего.

Однократная запись

Адрес регистра для записи ведомого указывается в поле «Байт 0» ведущего. Тип адресации ведомого – однократная запись, иначе будет перезаписываться регистр с адресом «0x01» (см. «Режим ведомого»).

 – «Бегущая единица». Последовательная повторяющаяся запись чисел: 1, 2, 4, 8, 16, 32, 64, 128.

 – «Бегущий ноль». Последовательная повторяющаяся запись чисел: 254, 253, 251, 247, 239, 223, 191, 127.

 – «Инкремент». Последовательная повторяющаяся запись чисел от 0 до 255.

 – «Декремент». Последовательная повторяющаяся запись чисел от 255 до 0.

 – «Случайное значение». Повторяющаяся запись случайных чисел в диапазоне от 0 до 255.

Пакетная запись

Тип адресации ведомого – пакетная запись, так как должны перезаписываться сразу все восемь регистров записи (см. «Режим ведомого»).

 – «Квадрат». Повторяющаяся запись определенной последовательности чисел таким образом, что создается изображение расширяющегося квадрата на матрице светодиодов ведомого тестера.

 – «Ромб». Повторяющаяся запись определенной последовательности чисел таким образом, что создается изображение расширяющегося ромба на матрице светодиодов ведомого тестера.

 – «Крест». Повторяющаяся запись определенной последовательности чисел таким образом, что создается изображение расширяющегося креста на матрице светодиодов ведомого тестера.

 – «Шахматы». Повторяющаяся запись определенной последовательности чисел таким образом, что создается изображение шахмат на матрице светодиодов ведомого тестера.

 – «Случайное значение». Повторяющаяся запись случайных чисел в диапазоне от 0 до 255.

Список литературы

1. Спецификации шины USB. USB: http://www.usb.org/developers/docs/.
2. Спецификация CAN 2.0. USB: http://www.nxp.com/docs/en/reference-manual/BCANPSV2.pdf.
3. Обзор стандартов RS485 и RS422. USB: http://focus.ti.com/lit/an/slla070d/slla070d.pdf.
4. Описание RS232. USB: http://www.radio-electronics.com/info/telecommunications_networks/rs232/eia-rs232-c-d-standards.php.
5. Сайт производителя 1-Wire. USB: http://www.maximintegrated.com/en/products/comms/one-wire.html.
6. Описание SPI. USB: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/technical_note/DM00054618.pdf.
7. Описание I^2C. USB: http://www.i2c-bus.org/.
8. Нортон П., Соухе Дж. Язык ассемблера для IBM PC. М. : Компьютер,1993. 351 с.
9. Описание датчиков температуры AD7416/7417/7418: http://analog.com/static/imported-files/data_sheets/AD7416_7417_7418.pdf.

Оглавление

Учебное издание

Фаткин Георгий Александрович
Панов Алексей Николаевич
Орешонок Владимир Викторович

Распределенные системы управления и последовательные шины передачи данных

Методические указания к лабораторной работе № 4
практикума ТСАНИ

Редактор *Я. О. Козлова*
Верстка *О. А. Тенекеджи*
Дизайн обложки *Е. В. Неклюдовой*

Подписано в печать 05.06.2018 г.
Формат 60x84/16. Уч.-изд. л. 1,75. Усл. печ. л. 1,6.
Тираж 30 экз. Заказ №.
Издательско-полиграфический центр НГУ.
630090, Новосибирск, ул. Пирогова, 2.