Dipl.-Ing. Jürgen Hulzebosch

USB in der Elektronik

FRANZIS
PC+ELEKTRONIK

Dipl.-Ing. Jürgen Hulzebosch

USB in der Elektronik

Die USB-Schnittstelle für praktische Anwendungen am PC einsetzen

Юрген Хульцебош

USB в электронике

2-е издание, исправленное

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.06
ББК 32.973
Х98

Хульцебош Ю.

Х98 USB в электронике: Пер. с нем. — 2-е изд., испр. — СПб.: БХВ-Петербург, 2009. — 224 с.: ил. + CD-ROM — (Электроника)

ISBN 978-5-9775-0658-8

В книге показано, как с помощью специализированных микросхем USB без интегрированного микроконтроллера создавать различные системы управления и устройства. Рассмотрены основы USB, аппаратное обеспечение (микросхемы, флэш-модули и др.), установка драйверов и разработка программ на Visual Basic. Приведены практические примеры различных устройств от простых (светофор, аварийная сигнализация, устройство для наблюдения за уровнем воды в аквариуме и др.) до более сложных (тестер дистанционного управления, устройство записи EEPROM-памяти, аналого-цифровой преобразователь и др.). Показано, как разработать универсальный интерфейс USB-I^2C, осуществить Flash-программирование AT89LP-микроконтроллера через SPI/ISP-интерфейсы и многое другое. На компакт-диске находятся примеры программ на языке Visual Basic, описания и спецификация электронных компонентов, а также специализированные драйверы и утилиты.

Для профессиональных инженеров-электронщиков и радиолюбителей

УДК 681.3.06
ББК 32.973

ISBN 978-3-7723-4089-5 (нем.)
ISBN 978-5-9775-0658-8 (рус.)

Оглавление

Предисловие

В последние годы USB стал универсальным интерфейсом. Наряду с компьютерной индустрией вряд ли осталась какая-либо область электроники, не затронутая USB, будь то автомобильный радиоприемник с USB-входом для подключения MP3-плеера, цифровой спутниковый приемник с обновлением через USB или современный сотовый телефон с MP3 и интегрированной камерой, которая имеет USB-порт.

USB — это сокращение Universal Serial Bus — универсальная последовательная шина. Интерфейс USB версии 1.0 был разработан еще в 1995 г. консорциумом нескольких больших предприятий по электронике.

В сравнении с последовательным COM-портом или параллельным LPT-портом персонального компьютера (ПК) USB обладает достаточно высокой скоростью передачи данных. Кроме того, USB поддерживает "горячее" подключение и отключение устройств.

При более детальном рассмотрении USB сложность этого интерфейса сначала может слегка испугать. Однако если раньше для управления собственной электроникой имелась возможность использования только параллельного или последовательного интерфейсов ПК, то сейчас нужно обязательно разбираться в USB, поскольку ПК новейших поколений оборудованы только этим интерфейсом.

Следует отметить, что большое количество литературы по USB можно найти в Интернете, у различных издательских компаний, а также у производителей микроконтроллеров. В большинстве случаев описываемые USB-контроллеры имеют микропроцессорное ядро, вследствие чего кажущийся простым USB-интерфейс снова становится достаточно сложным.

В книге мы не будем очень подробно останавливаться на функционировании USB. Используемая здесь USB-микросхема, произведенная компанией FTDI, не имеет интегрированного микроконтроллера. Однако, несмотря на это, в книге показано, как и без применения микроконтроллера могут быть созданы достаточно интересные USB-системы управления, сбора (и регистрации) данных.

Изначальный мотив изготовителей FTDI-микросхемы заключался в разработке преобразователя интерфейса USB в последовательный и параллельный интерфейсы. С помощью такого преобразователя, при необходимости, современные ПК можно дооборудовать и этими уже устаревшими интерфейсами. Все это послужило

основанием для автора лично разобраться с USB- и FTDI-микросхемами, поскольку новейшие ПК более не имеют последовательного интерфейса, а микроконтроллеры программируются именно через последовательный интерфейс.

Однако скоро было установлено, что хотя простая последовательная передача данных с помощью преобразователя функционировала, переключение отдельных сигналов последовательного интерфейса все же было слишком медленным по сравнению с обычным последовательным интерфейсом (RS-232) устаревшего ПК. Автора заинтересовал вопрос: не является ли будто бы быстрый USB-интерфейс на самом деле достаточно медленным. Из этой постановки вопроса, в конце концов, и родилась эта книга с большим количеством практических примеров.

В повседневной жизни встречается множество ситуаций, в которых можно использовать различные электронные устройства, управляемые при помощи компьютера. Книга начинается с простых экспериментов со светодиодом и светофором, подключенных к ПК с помощью интерфейса USB. Далее рассмотрены примеры схем аварийной сигнализации, устройства для наблюдения за уровнем воды в аквариуме, а также и другие устройства.

К концу книги представленные примеры становятся все сложнее и сложнее. Измерения яркости или температуры при помощи собственноручно построенного аналого-цифрового преобразователя способствуют закладке основ знаний, которые необходимы для создания как программного, так и аппаратного обеспечения для USB. Кое-кто, возможно, удивится, как легко с помощью нескольких строк в прикладном программном обеспечении можно управлять построенным собственными руками аналого-цифровым преобразователем. Тестер инфракрасного дистанционного управления на фотодиоде, вольтметр, запоминающее устройство на электрически программируемом ПЗУ (EEPROM-память) и использование USB-интерфейса для защиты программ от копирования являются дальнейшими практическими примерами.

Поскольку используемый USB-адаптер может служить в качестве преобразователя USB-интерфейса в последовательный интерфейс (COM-порт) микроконтроллера, приводятся дополнительные примеры программного обеспечения (ПО).

Достаточно опытный электронщик научится, каким образом можно использовать линии ввода/вывода (I/O) для более сложных интерфейсов, таких как I^2C (Inter-Integrated Circuit) или SPI (Serial Peripheral Interface). Работа с USB-I^2C-интерфейсом становится по сути "детской игрой", т. к. логический анализатор с помощью программного обеспечения фактически визуализирует сигнал данных и тактовый сигнал в виде генерируемой временной диаграммы. В книге рассмотрена поэтапная разработка интерфейса. Фактически за пять достаточно простых шагов вы сможете разработать универсальный USB-I^2C-интерфейс.

Последний пример в этой книге покажет вам основы того, как можно выполнить более сложные задания, например, флэш-программирование микроконтроллера Atmel AT89LPx052 посредством интерфейса SPI и внутрисистемного программирования ISP (In-System Programming).

Те, кто уже умеет разрабатывать, собирать и программировать электронику на микроконтроллерах, могут использовать описываемый в этой книге адаптер непосредственно с помощью преобразователя USB-интерфейса в последовательный интерфейс (COM-порт) или с помощью программируемого адаптера SPI/RS-232.

Отдельные примеры детально разъясняются и достаточно иллюстрируются, позволяя понять взаимосвязи программного обеспечения, выполненного на языке Visual Basic, с аппаратным обеспечением и USB-микросхемой FT232R. Эти взаимосвязи могут быть использованы и для собственных идей и разработок.

Те, кто любит экспериментировать и желает на практике воплотить некоторые приведенные примеры схем, должны иметь соответствующие навыки и уметь, по крайней мере, пользоваться паяльником.

Актуальную дополнительную информацию вы сможете найти на сайте: **www.minimikro.de**.

ГЛАВА 1

Измерение, управление и регулировка при помощи интерфейсов ПК

При использовании ПК для электронных экспериментов требуется хотя бы один интерфейс для внешней среды или для собственной созданной электроники.

Посредством этого интерфейса ПК и соответствующего программного обеспечения компьютер общается с подключаемой электроникой (рис. 1.1). Во многих случаях для этого использовались (и все еще иногда используются) последовательные или параллельные интерфейсы ПК.

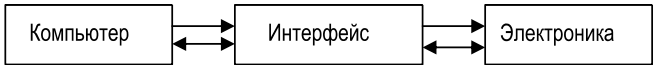


Рис. 1.1

Как можно догадаться из названия, при *последовательном интерфейсе* данные пересылаются последовательно, т. е. наименьшие единицы информации передаются по сигнальному проводу друг за другом. В последовательном интерфейсе ПК для отправки и приема передаваемых данных применяют соответствующие отдельные линии. Кроме того, при последовательном интерфейсе необходимы и другие линии, предназначенные для сигналов управления, с помощью которых осуществляется процедура передачи данных от компьютера к электронике (электронному устройству), или наоборот (рис. 1.2).

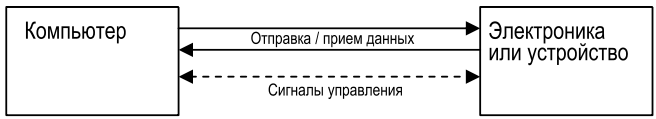


Рис. 1.2

Типичные устройства, которые подключаются к последовательному интерфейсу (COM-порту), — это модемы, измерительные устройства, а также мышь. Для соединения нужны 9- и 25-контактные электрические разъемы D-Sub.

Примечание

В USB используется последовательная передача данных.

В стандартном параллельном интерфейсе ПК одновременно может осуществляться передача восьми состояний на восьми информационных линиях. Как и в случае с последовательным интерфейсом, в параллельном интерфейсе также имеются дополнительные сигналы управления (рис. 1.3).

Рис. 1.3

Параллельный интерфейс использовался в основном для подключения принтеров. Для подсоединения применялись 25-контактные электрические разъемы D-Sub, которые устанавливались на ПК, и 36-контактные разъемы Centronics — на внешних периферийных устройствах.

Обычно к последовательному или параллельному интерфейсу ПК подключается только один прибор. Если один принтер уже подключили к параллельному интерфейсу, то для второго принтера нужен еще один параллельный интерфейс. Если нет в наличии или недостаточно свободных интерфейсов, то тогда ПК можно дооборудовать платами расширения, которые вставляются в различные внутренние слоты. Обусловленные прежними определениями аппаратного и программного обеспечений и операционной системы, начиная от DOS и Windows, вплоть до Windows XP в общем случае поддерживаются два параллельных и до 32 последовательных интерфейсов. Со специальной поддержкой изготовителя приборов возможны до 8 параллельных и 256 последовательных интерфейсов.

Как уже, наверняка, известно большинству читателей, для устройств, подключенных к последовательному или параллельному интерфейсу, необходимо установить соответствующее ПО — *драйвер*. Этот драйвер, как правило, предоставляется производителем устройства для соответствующей операционной системы. Без подобного драйвера устройство не может использоваться или может, но лишь ограниченно, т. к. операционная система ПК не имеет возможности распознать его.

В измерительной технике, напротив, электронщику для этого интерфейса драйвер может потребоваться лишь в очень редких случаях, потому что он имеет дело непосредственно с функциями аппаратного обеспечения компьютера (с адресами ввода и вывода), которое со времен введения их в IBM PC XT до сих пор поддерживается и в современных системах.

При разработке USB-интерфейса учитывался опыт, который уже был получен при подключении к компьютеру дополнительных устройств, что значительно облегчи-

ло пользователям применение этого интерфейса. USB означает Universal Serial Bus (универсальная последовательная шина) и является интерфейсом для многих современных устройств. USB (исходя от одного подключения USB-хоста) может опознать до 127 подключенных устройств, причем адрес 00 зарезервирован. Хост USB управляет всей активностью на USB-линиях и может целенаправленно обращаться к любому конкретному устройству по его адресу.

По сравнению с существующими последовательным и параллельным интерфейсами область применения USB-интерфейса была значительно расширена. USB-интерфейс находит применение не только в принтерах, модемах, мышках, сканерах и внешних жестких дисках, но он также активно используется и в других электронных приборах, таких как MP3-плееры, сотовые телефоны, видеоплееры, USB-карты памяти и т. д. Они зачастую поддерживаются непосредственно операционной системой без инсталляции драйвера. Шина USB имеет древовидную структуру. Если же ПК не обладает достаточным количеством USB-интерфейсов, то можно выполнить разветвление USB на нужное количество портов при помощи так называемых *хабов*.

Исходя от так называемого USB-хост-контроллера (в нашем примере USB-интерфейса компьютера), другие устройства могут быть включены звездообразно с помощью последовательно присоединенного 4-портового хаба. Хаб имеет один или несколько USB-контактов для подключения других USB-приборов. Хост-контроллер, установленный в ПК, управляет обменом с подключенными USB-устройствами и операционной системой.

Вместе с этим к 4-портовому хабу могут быть подключены и другие хабы. Так, например, USB-контакты, предназначенные для Прибора 4 (рис. 1.4), могут быть использованы для другого 7-портового хаба, благодаря чему количество USB-контактов структуры увеличивается на десять. В этом случае говорят о *каскадировании*.

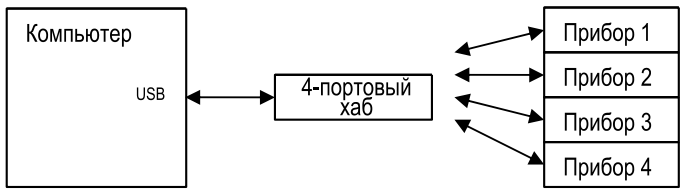


Рис. 1.4

В определенный момент времени только один подключенный USB-прибор может обмениваться с одним USB-хост-контроллером. Отправка сообщения одновременно на несколько подключенных приборов (*Broadcasting* — транслирование сообщений, режим передачи сообщения всем или нескольким абонентам сети одновременно) невозможна. Для повышения ширины полосы пропускания современные ПК имеют зачастую два, четыре или восемь хост-контроллеров. Если несколько USB-устройств должны обмениваться между собой данными, например, копировать данные с одного USB-жесткого диска на другой, и наоборот, имеет смысл подключить жесткие диски к разным USB-хост-контроллерам, а не к одному.

USB-разъем имеет только 4 соединительных контакта. Помимо двух контактов, предназначенных для подключения к источнику постоянного тока, которое не свойственно для последовательного или параллельного интерфейсов, имеется всего лишь две линии для передачи данных, по которым и осуществляется обмен с хост-контроллером. Вместе с тем, USB-разъемы не очень дороги и могут быть миниатюризированы.

Подключенные к USB устройства очень часто автоматически распознаются операционными системами, например, Windows 2000, Windows XP и Vista. Если же устройство подключается впервые, то почти всегда происходит запрос на инсталляцию драйвера. После успешной установки и интеграции драйвера операционная система распознает прибор автоматически.

Иногда, однако, не удается заметить интеграцию драйвера, т. к. операционная система (например, для USB-карты памяти или USB-жестких дисков) уже содержит драйвер в списке совместимости оборудования и автоматически интегрирует его.

1.1. Понятие о данных и единицах информации

Что в компьютерных технологиях или информатике подразумевается под понятием *данные*? Как выглядят данные, которые используются для обработки в компьютерах в соответствии с микроконтроллерами, а также передаются по различным интерфейсам — от электронных устройств к компьютеру или в самом компьютере. В Интернете можно обнаружить примерно следующий ответ:

Данные — это логически сгруппированные единицы информации, которые передаются между (компьютерными) системами. Они состоят из двоичного кода. Требуется наличие кодирующих инструкций, с помощью которых данные могут быть преобразованы.

В информатике и электронной обработке данных под понятием "данные" понимается читаемое (машиной) и обрабатываемое, как правило, цифровое представление информации. Информация для этого сначала кодируется в знаки (или цепочки знаков), структура которых подчиняется строгим правилам.

Данные — это множественное число термина "данное", т. е. наименьшей неделимой единицы информации, которая передается в сообщении. *Сообщение* — это форма представления информации, т. е. совокупность знаков или первичных сигналов, содержащих информацию, которая имеет признаки начала и конца и предназначена для передачи через определенную среду связи. Сообщение имеет определенное значение для получателя и подчиняется фиксированным правилам. В компьютере, работающем в двоичном коде, наименьшими *единицами информации* являются значения 0 и 1. Благодаря расположению в ряд нулей и единиц образуются цепочки знаков и чисел для дальнейшей обработки в программном и аппаратном обеспечении компьютера.

Примечание

Компьютер — это производное от англ. "to compute" (считать или вычислять).

Данные могут быть представлены в аналоговой или цифровой форме и быть соответственно аналоговыми или цифровыми.

Наиболее наглядным примером передачи *аналоговых данных*, наверное, является телефон: если говорить в микрофон телефонной трубки, то звуковые волны речи преобразуются в электрические колебания; эти электрические сигналы (данные) передаются по проводам, а в динамике телефонной трубки получателя они снова преобразуются в звуковые волны.

В противоположность двоичным *цифровым данным* с дискретными значениями 0 или 1 аналоговый сигнал может принять любое произвольное значение внутри установленных предельных значений, например: 0,1; 0,2; 0,2345...

1.2. Двоичный счет (логические 0 и 1)

Сигналы или сигнальные линии в цифровом интерфейсе закодированы в двоичной системе счисления и могут принимать только значения 0 или 1.

Все это становится довольно просто, если представить себе то, как мы считаем при помощи десяти чисел от 0 до 9 в нашей обычной десятичной системе счисления (табл. 1.1).

Таблица 1.1

Десятичное основание 10 (0–9)	Двоичное основание 2 (0,1)	Шестнадцатеричное основание 16 (0–F)
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

При всех системах счисления соответствующие дискретные единицы информации располагаются в ряд, и от этого зависит соответствующее значение позиции внутри информации (данных). Например, десятичное число 14_{10} в разных системах счисления можно представить следующим образом:

14_{10} в десятичной системе $\rightarrow 14 = 1\times10^1 + 4\times10^0 = 10 + 4$

14_{10} в двоичной системе $\rightarrow 1110 = 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 8 + 4 + 2 + 0$

14_{10} в шестнадцатеричной системе $\rightarrow E = 14\times16^0 = 14$

В приведенных системах счисления различны только основания или количество возможных дискретных значений единиц информации. В десятичной системе ими являются числа от 0 до 9 (до основания 10, десять возможных дискретных значений), а в двоичной системе — два значения 0 и 1 (до основания 2).

1.3. "Счет на пальцах"

Очень просто производить деление двоичного числа на два: нужно сдвинуть все биты на одну позицию направо, а последний бит отбросить. Например, при делении числа 10_{10} (двоичное число 1010_2) на 2_{10} получается двоичное число 101_2 (десятичное 5_{10}). Также и из десятичного числа 11 получится число 5. Умножение на два производится со сдвигом битов налево.

Читатель может проверить следующее умножение двух чисел и попытаться вникнуть в суть дела. Автору этот вид умножения известен под названием "счет на пальцах". Рассмотрим пример умножения 17 на 12 (17×12).

Левый столбец (17) каждый раз делится на 2, правый столбец (12) каждый раз умножается на 2. Остатки при этом не берутся в расчет (табл. 1.2).

Таблица 1.2

17	12
8	24
4	48
2	96
1	192

Сложим только те числа в столбце справа, которые в левом столбце имеют нечетное значение: т. е. 12 + 192 = 204. Ответ подошел, но всегда ли так происходит?

1.4. Биты и байты

Бит (binary digit, двоичный разряд) может принимать логические значения 0 и 1. Если расположить в ряд 8 битов, как в предыдущем примере, то тогда с 8 битами можно досчитать до 255 значений:

$$\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1} = 1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 =$$
$$= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

Итак, 8 битов образуют 1 байт. Байт может принимать 256 значений (от 0 до 255).

Итак, *байт* — это единица информации данных, которая состоит из 8 битов. Первый бит (самый левый, 128) обозначается также MSB (Most Significant Bit, наиболее значащий бит, старший разряд), последний бит (самый правый) — как LSB (Least Significant Bit, наименее значащий бит, младший разряд).

Слово данных (Word) — это единица информации, состоящая из 16 битов, или 2 байтов, которая может принимать 65 536 значений (от 0 до 65 535).

1 Мбайт (1 000 000 байт) равен 10^6 байтов, однако в вычислительной технике под *мегабайтом* принято понимать 2^{20} байт (1 048 576), что приводит к почти 5%-й разнице с первым значением, а под *килобайтом* принято понимать 2^{10} байт.

Иначе: 1 Мбайт = 1024 Кбайт = 2^{10} Кбайт,

1 Кбайт = 1024 байт = 2^{10} байт,

1 Мбайт = 1024×1024 байт = $2^{10}\times2^{10}$ байт = 2^{20} байт.

ГЛАВА 2

Основы USB

2.1. USB 1.0, 1.1, 2.0, On-The-Go

USB 1.0

Известная на сегодняшний момент под названием USB 1.0 последовательная шина была разработана компанией Intel. Спецификации USB 1.0 были обнародованы еще в январе 1996 г.

Интерфейс USB 1.0 имел максимальную скорость передачи, равную 1,5 Мбит/с (низкая скорость — Low-Speed), и поэтому не мог быть полезным для запоминающих устройств сверхбольшой емкости (например, жестких дисков). Однако такие устройства уже поддерживались. Первыми же используемыми устройствами были, например, принтер, сканер, мышь и клавиатура.

USB 1.1

С интерфейсом USB 1.1 стало возможным достигнуть скорости передачи данных, равной 12 Мбит/с (полная скорость — Full-Speed).

USB 2.0

В 2000 г. была обнародована спецификация USB 2.0 с высокоскоростной передачей данных (480 Мбит/с), а также созданы предпосылки для того, чтобы USB стал более привлекателен для периферийных устройств, таких как принтер, жесткий диск, карты памяти и т. д.

USB 2.0 со скоростью передачи данных, большей в 40 раз, чем при высокоскоростном режиме, остался совместим с версиями USB 1.X.

USB On-The-Go

Благодаря USB On-The-Go (OTG), дальнейшего расширения спецификации USB 2.0, оборудованные им USB-устройства могут непосредственно сообщаться между собой. Поэтому от компьютера, который принимает на себя функцию хоста,

можно отказаться. Два USB-устройства могут обмениваться сообщениями без хоста. Это имеет смысл при использовании цифровых камер с USB-разъемом для управления USB-принтерами, чтобы иметь возможность распечатать фотографии непосредственно на принтере, без подключения к компьютеру.

Wireless USB (беспроводная USB)

При упоминании этой спецификации USB речь идет о беспроводном USB. Данная спецификация была выпущена в мае 2005 г. Она делает возможным беспроводную связь между USB-устройствами при полной скорости передачи данных максимум 480 Мбит/с.

2.2. Идентификация устройств USB-интерфейса. Ток и напряжение линий электропитания

Физически USB-разъем имеет в общей сложности 4 линии (рис. 2.1), две для электропитания (земля и +5 В) и две линии для передачи данных (D+ и D–).

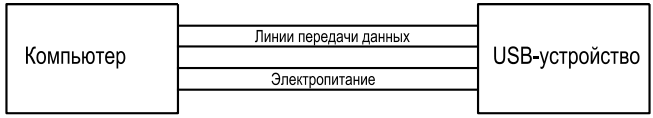


Рис. 2.1

Потребление электроэнергии подключенного USB-устройства не должно превышать 100 мА, если USB-устройство не было сконфигурировано. Максимальное потребление электроэнергии при сконфигурированном USB-устройстве составляет 500 мА. Если необходимо более 100 мА, то этого должно потребовать USB-устройство. Устройства, которые должны питаться непосредственно от шины USB, снабжены опознавательным признаком *bus powered* (питаемый от шины). Само собой разумеется, что USB-устройства также могут быть оснащены своим собственным электропитанием. Такие устройства обладают признаком *self powered* (с собственным источником питания).

Линии передачи данных (D+ и D–) рассчитаны для оптимизации скорости передачи данных. Отдельные состояния сигнала этих линий здесь рассматривать не будем, т. к. они были бы важны только при разработке USB-чипа.

Если USB-устройство подключается к USB-контроллеру или извлекается из него, то линии передачи данных используются по технологии *Plug-and-Play* (досл. англ. — подключи и работай). Если линия передачи данных D+ подключенного USB-устройства относительно напряжения питания имеет сопротивление 1,5 кОм, то устройство в этом случае определяется в качестве полноскоростного или иначе высокоскоростного устройства. Если же то же самое можно сказать для линии D–, то тогда это низкоскоростное устройство.

Все USB 2.0-совместимые устройства при подключении к шине с полной скоростью (Full-Speed) определяются как устройства интерфейса USB 1.1. Только когда хост-контроллер обнаружит, что речь идет об устройстве, приспособленном к интерфейсу USB 2.0, происходит переключение скорости через хост.

Если обе линии для передачи данных D+ и D– остаются на низком логическом уровне дольше 2,5 мкс, то тогда это состояние определяется как *disconnect* (разъединение), и устройство можно отсоединить.

Длина USB-кабеля не должна превышать 5 м. Если потребуется кабель большей длины, то тогда на каждые 5 м кабеля нужно использовать USB-хаб.

2.3. Последовательная передача данных в интерфейсах USB, RS-232, SPI и I^2C

Краткое сопоставление наиболее часто используемых последовательных интерфейсов приведено в табл. 2.1.

Таблица 2.1

Параметр	Интерфейс			
	USB 2.0	**RS-232**	**SPI**	**I^2C**
Количество устройств	127	2	8	40
Скорость передачи данных	до 480 Мбит/с	до 115 Кбит/с	до 2 Мбит/с	до 2 Мбит/с
Максимальное расстояние	5 м, макс. 30 м	до 30 м	до 3 м	до 5 м

USB служит для подключения самых разных внешних устройств. Интерфейс RS-232 можно встретить, например, в модемах, мышах и измерительных приборах.

Интерфейсы SPI (Serial Peripheral Interface) и I^2C (Inter-IC bus) служат преимущественно для внешнего соединения микроконтроллера с другими периферийными микросхемами (узлами) и поэтому поддерживают работу лишь на небольших расстояниях.

При последовательной передаче сообщение расчленяется на минимальные элементы. Таким образом, 1 байт посылки при передаче снова разделяется на 8 отдельных битов. Кроме данных к сообщению добавляется и другая служебная информация: как, например, признак начала и признак конца одной единицы посылаемой информации.

Скорость передачи данных — это скорость, выраженная в количестве бит, передаваемых за единицу времени, которая, по сути, соответствует частоте передаваемых битов. Частота, как известно, вычисляется по формуле: $f = 1/t$, где t — длительность передачи одного бита. Для интерфейса USB 1.1 в режиме Full-Speed скорость пере-

дачи данных составляет 12 Мбит/с. Время передачи одного бита таким образом будет равно: $t = 1/f = 1/12 \cdot 10^6 = 83$ нс. Что же касается реальной скорости передачи исходных данных, то ее значение, равное 12 Мбит/с, в действительности не достигается. Эффективные ее значения для интерфейса USB 1.1 находятся между 6,5 и 9,5 Мбит/с, в зависимости от того, какой из типов передачи данных по USB используется.

2.4. Кодировка NRZI

При USB-передаче применяется способ кодировки *NRZI* (Non Return to Zero Inverted, кодирование без возвращения к нулю с инверсией). Принцип кодирования NRZI заключается в следующем. Если на вход схемы кодирования поступает логическая 1, то уровень выходного напряжения остается без изменений, а если логический 0 — изменяется на противоположный. Для осуществления хорошей синхронизации с минимальными затратами на получателе кодовой последовательности применяется так называемый *формирователь битов — Bit Stuffer* (рис. 2.2), который осуществляет принудительную вставку логического 0 после приема шести следующих подряд друг за другом единичных битов двоичной последовательности.

Рис. 2.2

Этот бит должен быть вставлен также и в том случае, если в потоке данных после единичных повторяющихся данных следует бит со значением логического 0. Этот шаг обязателен для того, чтобы на стороне получателя можно было распознать и снова удалить этот дополнительный бит.

Кроме того, для обеспечения точной синхронизации в процессе длительной передачи сообщения перед каждым пакетом данных формируется специальная синхронизирующая последовательность, состоящая из 8 битов.

2.5. Последовательная шина USB

В Интернете на некоторых немецкоязычных сайтах аббревиатуру USB иногда представляют не как универсальную последовательную шину (Universeller Serieller Bus), а как неизвестную последовательную шину (Unbekannter Serieller Bus). Действительно разобраться в деталях интерфейса USB, процессах в протоколе USB-передачи, с дескрипторами, кодированием и декодированием данных, синхронизацией, скоростями передачи данных, необходимыми (а также и необязательными) драйверами на уровне операционной системы, со спецификациями OHCI, UHCI, EHCI или идентификатором производителя — не так-то это просто для электронщика-любителя.

В противоположность приложениям для последовательного или параллельного интерфейса, USB-приложения не могут просто писать данные по различным адресам ввода/вывода или читать с этих адресов. Для того чтобы приступить к работе с USB-устройством, приложения должны общаться с драйвером (класса устройств или устройства), который, в свою очередь, на более низком уровне общается с USB-драйверами управления сообщениями по линиям передачи данных USB. В устройстве должны быть реализованы протоколы, с помощью которых компьютер может распознать и идентифицировать устройство, а также обмениваться с ним.

К счастью, многие производители USB-контроллеров уже сталкивались с этой проблемой. Следует отметить, что в этой книге FTDI-адаптер представлен в двух вариантах: с разводкой выходных контактов для последовательных сигнальных линий, как на рис. 2.3, или для параллельных данных, как на рис. 2.4.

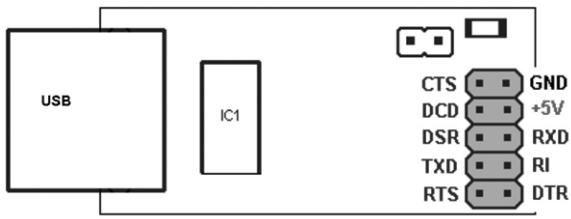


Рис. 2.3

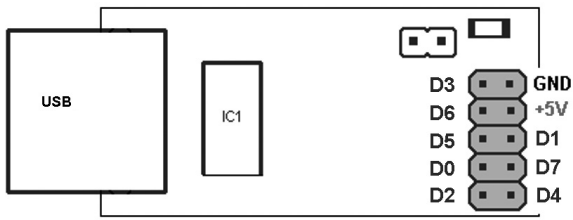


Рис. 2.4

Для работы с адаптером надо, прежде всего, выбрать USB-адаптер с последовательным или параллельным выходом, затем подключить его, установить FTDI-драйвер и запустить программы, написанные, например, на Visual Basic или Паскале (для профессионалов подойдет и язык Си).

Интересно, что выходные сигналы последовательного интерфейса FTDI-контроллера можно использовать также и для 8-битовой параллельной передачи данных. Кроме того, могут быть смоделированы и другие последовательные интерфейсы, такие как SPI или I^2C.

Используемый в данном случае USB-контроллер, выполненный на микросхеме FT232R и изготовленный компанией FTDI, рассчитан на интерфейс USB 2.0.

Передача данных по USB происходит в пределах кадров размером в 1 мс, которые можно легко представить как периодическое временное окно, причем тактирование осуществляется USB-хост-контроллером.

Сообщения передаются в виде пакетов данных внутри этого временного окна.

Передача данных внутри миллисекундных кадров различна. Для USB имеется три режима работы: Low-Speed — скорость передачи данных составляет 1,5 Мбит/с, при Full-Speed — до 12 Мбит/с, при High-Speed (только для USB 2.0) — до 480 Мбит/с. При USB High-Speed для достижения высокой скорости передачи данных каждый 1-миллисекундный кадр разделяется на 8 микрокадров размером по 125 мкс.

2.6. Типы USB-передач

USB-устройства могут по-разному обмениваться данными с USB-хост-контроллером. Для пересылки сообщений имеется четыре типа передач.

Управляющая передача (Control-Transfer)

Управляющие посылки отсылаются обычно в оба направления, так что и отправитель (источник) и получатель (приемник) всегда могут быть уверены, что данные прибыли. Каждое USB-устройство должно обязательно поддерживать управляющие передачи (Control-Transfer).

Управляющие передачи очень важны для начала обмена. Они служат для конфигурирования USB-устройств во время их подключения и в процессе работы для управления ими, а также, помимо всего прочего, для того, чтобы получить информацию о совместимости USB-устройства.

Передача по прерыванию (Interrupt-Transfer)

Передачи с использованием прерываний (Interrupt-Transfer) работают с проверкой ошибок и задуманы для коротких пакетов данных. В этом случае не гарантируется точно установленная скорость передачи. Прерывания имеют спонтанный характер. Подключенное USB-устройство периодически должно выполнять запрос на обслуживание. Предел времени обслуживания для низкоскоростного режима передачи данных (Low-Speed) устанавливается между 10 и 255 мс, для полноскоростного режима (Full-Speed) — между 1 и 255 мс, а для высокоскоростного (High-Speed) может быть 125 мкс.

При передаче с использованием прерывания и Low-Speed за один запрос передается до 8 байтов, при Full-Speed — до 64 байтов, а при High-Speed — до 1024 байтов. Отсюда максимальная скорость передачи данных при Low-Speed будет 800 байт/с, при Full-Speed — 64 Кбайт/с и при High-Speed — до 24 Мбайт/с. Несмотря на схожесть в названии, такое прерывание при USB-передаче не имеет ничего общего с общепринятым прерыванием процессора.

Поточная или групповая передача (Bulk-Transfer)

Поточная или иначе групповая передача (Bulk-Transfer) задумана для больших по размеру наборов данных, нуждается в проверке ошибок, но не имеет жестких тре-

бований к времени передачи по интерфейсу USB. Эти передачи обладают меньшим приоритетом. Они проводятся контроллером, когда другие изохронные передачи и передачи с использованием прерываний отключены. Применяемый FTDI-чип микросхемы FT232R поддерживает этот тип передачи.

Изохронная передача (Isochron Transfer)

Изохронные передачи являются важными для USB-устройств, которые обрабатывают большие наборы данных (файлы). Они имеют гарантированную скорость передачи данных. При изохронной передаче (Isochron Transfer) не происходит корректирование ошибок. Изохронная передача в режиме Low-Speed не используется.

2.7. USB-драйвер

Вообще, *драйвер* — это компьютерная программа, которая является связующим звеном между аппаратным и программным обеспечением компьютера. Для того чтобы обратиться к USB-устройству, прикладная программа должна связаться с драйвером, который в свою очередь на более низком уровне общается с USB-драйвером, управляющим устройством с помощью сообщений, передаваемых по линиям передачи данных. Было бы конечно хорошо обратиться к более низкому уровню непосредственно при помощи собственной программы, но в этом случае пришлось бы более детально разбираться со свойствами USB-устройства, специфичными для того или иного изготовителя. У новых операционных систем Windows драйверы на сегодняшний день имеют лишь необходимые права доступа, для того чтобы можно было обратиться к аппаратному обеспечению компьютера.

Компания FTDI для выпускаемых USB-контроллеров и ОС Windows свободно распространяет два различных драйвера. Первый драйвер — это так называемый VCP-драйвер (Virtual Com Port — виртуальный COM-порт). Этот драйвер заботится о том, чтобы операционная система интегрировала последовательный COM-порт, когда FTDI-USB-контроллер связывается с компьютером через USB. Благодаря этому обычно эмулируется последовательный интерфейс, представляясь для операционной системой другим COM-портом. Так удается как можно быстрее дооборудовать более новые компьютеры виртуальным последовательным интерфейсом и добиться высокой совместимости с существующими прикладными программами.

Второй драйвер — D2XX. Здесь имеется в виду драйвер FTD2XX.SYS, включающий драйвер WIN32 Driver Model (WDM — тип драйвера Win32), который осуществляет связь с устройством посредством Windows USB Stack и библиотеки DLL.

Для двух драйверов FTDI в Интернете на сайте **www.ftdichip.com** имеется соответствующая программная документация.

Сначала при установке драйвера нужно было выбирать один из двух драйверов. Причем если кто-то хотел использовать второй драйвер, то первый драйвер обязательно нужно было деинсталлировать. С выходом версии драйвера 2.0 с 2007 г. деинсталляция больше не требуется: драйвер для двух вариантов (D2XX и VCP) из-

бавляет от надоедливых установок и удалений, так что можно продолжать экспериментировать.

Компания FTDI поддерживает не только драйверы для всех распространенных вариантов Windows, но также и для операционных систем Linux, Mac OS и др.

2.8. Идентификация USB-устройства

Чтобы USB-устройство могло быть предложено на рынок, ему нужен так называемый *идентификатор производителя — VID* (Vendor ID) и *идентификатор изделия (продукта) — PID* (Product ID). Идентификатор производителя (VID) для FTDI-контроллера компания FTDI предоставляет бесплатно. Если же FTDI-контроллер используется для других целей, то от компании FTDI может быть получен другой номер идентификатора изделия — PID (рис. 2.5).

Номера VID и PID сохраняются внутри FTDI-контроллера в *EEPROM-памяти* (электрически стираемом программируемом постоянном запоминающем устройстве). Они могут быть даже изменены, но в этом случае потребуется процедура переустановки драйвера. После внесения каких-либо изменений требуется новая установка.

Рис. 2.5

В книге использован идентификатор производителя FTDI-VID и идентификатор изделия FTDI-PID, в то время как новый драйвер в любой момент может быть получен с сайта **www.ftdichip.com**.

ГЛАВА 3

Аппаратное обеспечение

Почти все изготовители чипов производят элементы с интегрированным USB, поскольку USB-подключение можно найти почти в любой области электроники. Соответственно и выбор чипов многообразен, если не сказать — необозрим. В связи с этим не так-то легко отдать предпочтение одной микросхеме. Несомненно, для этого необходимо интересоваться и заниматься микроконтроллерами (ядром многих микросхем USB), что раньше, при последовательном или параллельном интерфейсе персональных компьютеров, было необходимо лишь условно. К тому же существует еще множество приложений для последовательных и параллельных интерфейсов ПК, которые в будущем должны будут заменены USB и нуждаются в простом соединении.

Если кто-то хочет иметь USB-соединение без (явно) интегрированного микроконтроллера, то для собственных экспериментов в расчет могут быть взяты лишь микросхемы USB тех изготовителей, которые предлагают USB-преобразователь для последовательного или параллельного интерфейса:

- Profilic Technologie (технология Profilic) PL2303X — эта микросхема является преобразователем интерфейса USB в последовательный интерфейс для USB 1.1 в корпусе SSOP-28. Дескрипторы прочно вмонтированы во внутреннем *ПЗУ* (*ROM* — Read Only Memory). Для персонального идентификатора изделия PID или идентификатора производителя VID, конечно же, понадобится внешняя EEPROM-память, которая может быть подключена с помощью двухпроводной линии связи. Приложения ограничены сигнальными линиями последовательного интерфейса. Версия микросхемы PL2303HX RevD имеет однократно программируемую постоянную память *OTP-ROM* (One Time Programmable ROM) и предоставляет возможность использования последовательных сигналов, например, в виде 8 бит ввода/вывода (I/O);
- Silicon Labs CP2102/103 — в противоположность микросхеме Profilic Silicon Labs предлагает микросхему с интегрированной EEPROM-памятью емкостью 1024 байтов в QFN-корпусе и 23-штырьковыми выводами для USB 2.0 Full-Speed;
- MosChip MSC7820 — микросхема MCS7820, выпускаемая компанией MosChip, находящаяся в LQFP-корпусе с 48-штырьковыми выводами, поддерживает два последовательных интерфейса и в то же время инфракрасную передачу данных

SIR-IrDA, а также USB 2.0. Внешняя EEPROM-память может быть подключена по двум линиям интерфейса I^2C;

- FTDI — компания FTDI предлагает множество микросхем USB для эмуляции последовательного и параллельного интерфейсов. Так называемый режим Bit Bang (Bit Bang Mode), который делает микросхему довольно интересной для других приложений, в своих первых версиях задумывался как простое дополнение программирования/соединения FPGA. Компания FTDI свела воедино возможности преобразования из USB в другие последовательные интерфейсы (как последовательных интерфейсов ПК) в так называемом режиме *MPSSE* (Multi-Protocol Synchronous Serial Engine). При помощи режима Bit Bang микросхема FT232R дала простор другим приложениям без дополнительного микроконтроллера;
- MAXIM — фирма Maxim предлагает две интересные микросхемы USB Max3421 и Max3420, которые имеют два прямых соединения микроконтроллера с преобразователем от USB 2.0 (Full-Speed) в *SPI* (Serial Peripheral Interface, последовательный интерфейс периферийных устройств). Микросхема Max3421 поддерживает изохронную передачу данных. Контроллер находится в TQFP-корпусе с 32-штырьковыми выводами. Помимо интерфейса SPI он имеет восемь других входов и выходов. Микросхема подготовлена в качестве USB-хоста для USB-OTG. Поскольку последовательный интерфейс ПК не поддерживается, то микросхемы могут использоваться без дополнительного драйвера USB непосредственно через Windows USB Stack.

3.1. USB-адаптер и описание

Этот адаптер изначально был разработан для последовательного SPI-программирования семейства микроконтроллеров Atmel-AT89LP (рис. 3.1). Литературу и информацию для заказа вы сможете найти на сайте **www.minimikro.de**[1].

Адаптер со стороны выхода имеет 10-штырьковый соединительный разъем, который, в зависимости от использования, разведен для последовательных или параллельных сигнальных линий (см. рис. 2.3 и 2.4).

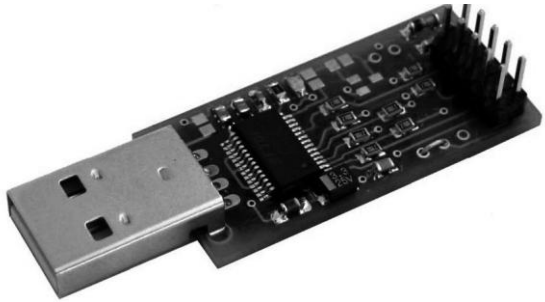


Рис. 3.1

[1] http://www.atmel.ru, www.chip-dip.ru. — *Прим. пер.*

Если смотреть на плату адаптера сверху и направить USB-разъем от себя, то выводы 1 и 2 (GND) штырькового соединителя будут с правой стороны: штырьковый вывод 1 ближе к чипу USB-контроллера IC1, а штырьковый вывод 2 — ближе к нижнему краю платы. Схема адаптера представлена на рис. 3.2.

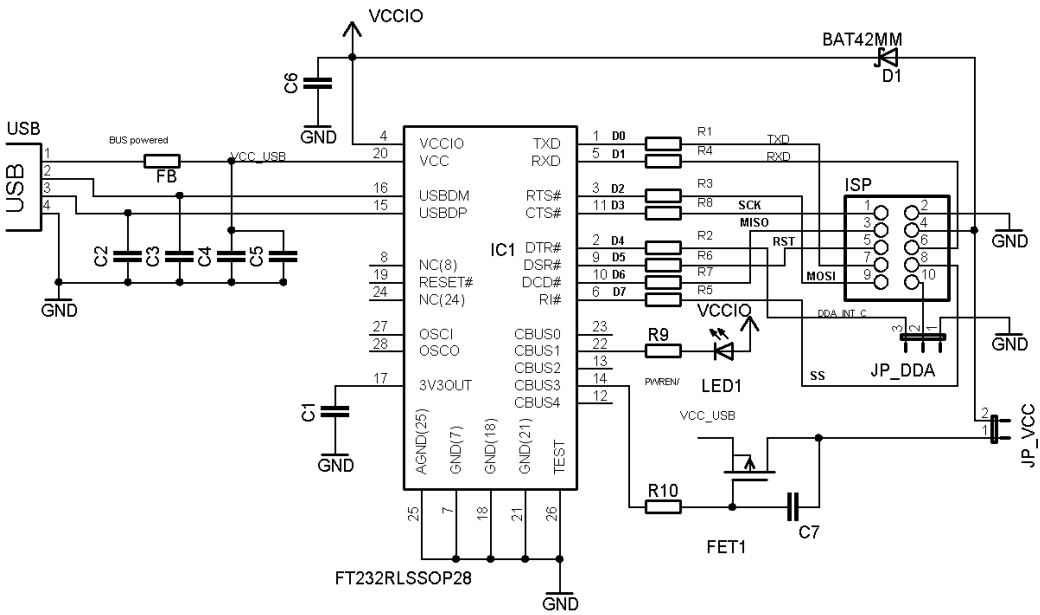


Рис. 3.2

Микросхема FT232R имеет достаточно простую схему включения. Резисторы от R1 до R8, подсоединенные к сигнальным линиям, имеют сопротивление 256 Ом и предназначены для уменьшения тока при возможном коротком замыкании линии.

Посредством МОП-транзистора FET1 и джампера (перемычки) JP_Vcc на штырьковый вывод 4 выходного 10-контактного разъема поступает напряжение 5 В. Транзистор необходим для управления питанием внешних логических схем. Как только подключенный компьютер переключается в режим Standby (спящий режим), соответствующим образом реагирует транзистор FET1, что также приводит к уменьшению потребления тока от USB. Резистор R10 и конденсатор C7 предназначены для плавного включения питания.

Конденсаторы C4 и C6 имеют емкость по 100 мкФ, а конденсатор C5 — 3,3 мкФ. Конденсаторы C2 и C3 (каждый по 22 пФ) служат для уменьшения возможных помех от микросхемы FT232.

Дроссель на ферритовом сердечнике (FB — Ferrite Bead) в цепи напряжения питания Vcc_USB — это рекомендация компании FTDI для уменьшения радиопомех от микросхемы FT232R.

Вывод 10-штырькового двухрядного выходного соединительного 10-контактного разъема (обозначенного ISP) подключен к средней точке штырьковых контактов

переключателя JP_DDA, который с помощью перемычки соединен с сигналом D4 (выводом 2) микросхемы IC1. Перечень элементов USB-адаптера приведен в табл. 3.1.

Таблица 3.1

Элемент	Значение (тип)	Описание корпуса
C1	Конденсатор 100 нф	C0805
C2	Конденсатор 22 пф	C0805
C3	Конденсатор 22 пф	C0805
C4	Конденсатор 3,3 мкф	C1206K
C5	Конденсатор 10 нф	C0805
C6	Конденсатор 100 нф	C0805
C7	Конденсатор 100 нф	C0805
C8	Конденсатор 22 пф	C0805
D1	Диод Шоттки BAT42MM	MINIMELF
FB	Дроссель на ферритовом сердечнике FB (Ferrite Bead)	R1206
FET1	МОП-транзистор IRLML6402	SOT23
IC1	Микросхема FT232RL	SSOP28
ISP	Штырьковый соединительный 10-контактный разъем на плате 2×05 (Pin header)	—
JP_DDA	Штырьковые контакты (Pin header) переключателя	—
JP_VCC	Штырьковые контакты (Pin header)	—
LED1	Светодиод в корпусе ChipLED	_0805
R1	Резистор 256 Ом	R0805
R2	Резистор 256 Ом	R0805
R3	Резистор 256 Ом	R0805
R4	Резистор 256 Ом	R0805
R5	Резистор 256 Ом	R0805
R6	Резистор 256 Ом	R0805
R7	Резистор 256 Ом	R0805
R8	Резистор 256 Ом	R0805
R9	Резистор 390 Ом	R0805
R10	Резистор 1 кОм	R0805
USB	Разъем USB (PN87520)	—

ВНИМАНИЕ!

При экспериментировании к порту USB 2.0 без использования USB-хаба должен быть подключен только USB-адаптер.

Постарайтесь по возможности браться только за края USB-платы. Иначе, при определенных условиях, вы можете повредить USB-адаптер за счет только одних статических разрядов! Когда вы будете вставлять адаптер, вы заметите короткую вспышку светодиода LED1.

3.2. Дополнительная плата

На дополнительной плате размещено большинство элементов, предназначенных для выполнения дальнейших экспериментов. Плата напрямую соединена с USB-адаптером (рис. 3.3). В качестве коммутационной панели для 8 сигнальных линий ввода/вывода служит 20-контактная панелька для интегральных микросхем (ИС).

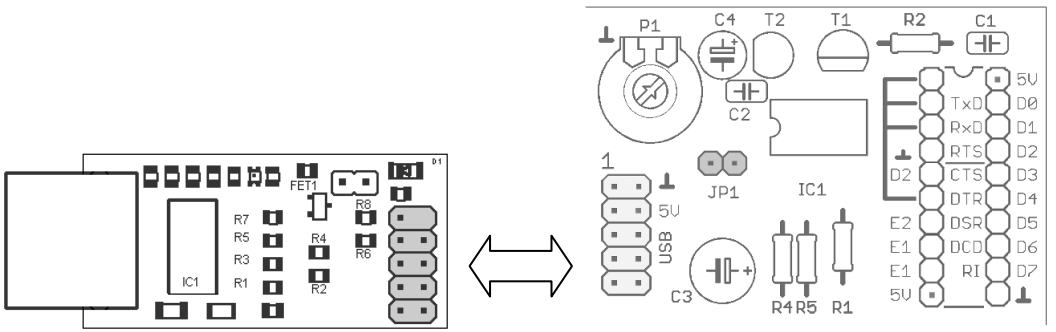


Рис. 3.3

На плате (рис. 3.4) находится операционный усилитель (ОУ) и конструктивные элементы для простого аналого-цифрового преобразователя (АЦП). При помощи перемычки JP1 на плате вы можете отключить питающее напряжение для аналоговой части. Выводы 1, 2, 3 и 6 20-контактной панельки для ИС непосредственно соединены между собой.

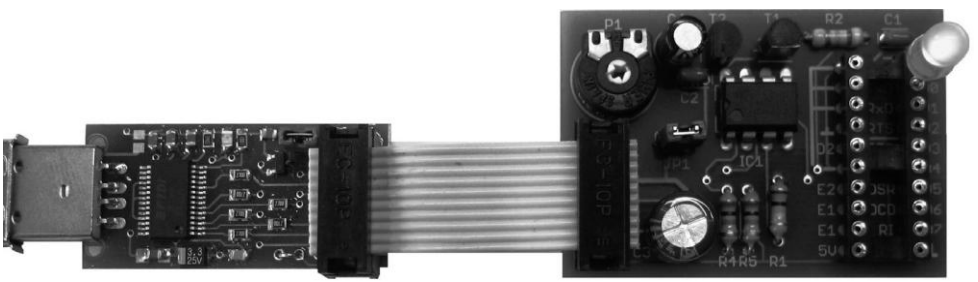


Рис. 3.4

Далее приведен перечень электрических элементов, размещенных на дополнительной плате (рис. 3.5):

- Конденсатор C1 100 нф C-EU025-024X044
- Конденсатор C2 100 нф C-EU025-024X044

- ❒ Конденсатор C3 100 мкф CPOL-EUE2.5-7
- ❒ Конденсатор C4 0,47 мкф CPOL-EUE2.5-5
- ❒ Микросхема IC1 LM358N DIP08
- ❒ Перемычка JP1 1×02 (штырьковые контакты + джампер)
- ❒ Потенциометр P1 100 кОм с линейной характеристикой
- ❒ Резистор R1 100 кОм R-EU_0204/7
- ❒ Резистор R2 39 кОм R-EU_0204/7
- ❒ Резистор R4 10 кОм R-EU_0204/7
- ❒ Резистор R5 10 кОм R-EU_0204/7
- ❒ Панелька для ИС S1 20Pin, DIP20
- ❒ Транзистор T1 BC557A TO92
- ❒ Транзистор T2 BS170 TO
- ❒ Штырьковый соединительный разъем USB 2×05

Информацию о заказе заготовки для дополнительной платы вы можете найти в Интернете по ссылке **www.minimikro.de**.

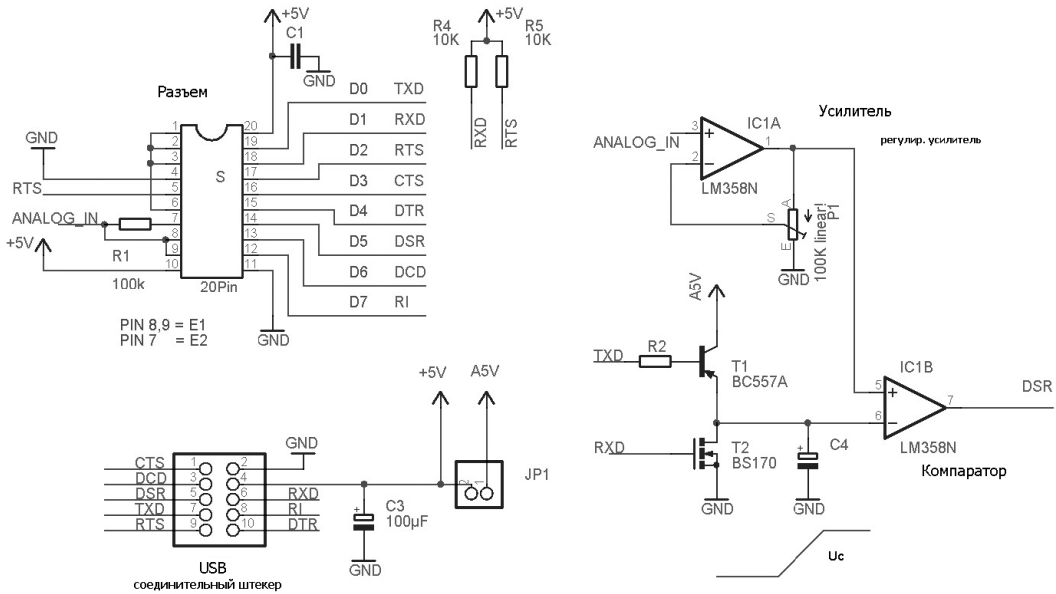


Рис. 3.5

Далее показаны другие используемые в работе с книгой электрические элементы:

- ❒ двухцветный светодиод (Duo-LED) 3 или 5 мм — (2-контактный);
- ❒ три светодиода 5 мм (красный, зеленый, желтый);
- ❒ светочувствительный фоторезистор LDR (Light Dependent Resistor);

- ❒ фототранзистор (фотодиод) BPW-40;
- ❒ терморезистор с отрицательным ТКС типа NTC 0,2-4K7 (4700);
- ❒ резистор 1 кОм;
- ❒ резистор 100 кОм;
- ❒ резистор 27 Ом;
- ❒ электролитический конденсатор (Elko) 100 мкф;
- ❒ микросхема EEPROM-памяти 24C16;
- ❒ диод BAT42;
- ❒ соединительный провод;
- ❒ соединительный плоский кабель с двумя разъемами FC-10P для подключения дополнительной платы к USB-адаптеру.

3.3. UM232R-модуль от компании FTDI

Компания FTDI предлагает собранный модуль с DIP-панелькой (панелька с двухрядным расположением штырьковых выводов) с 24-штырьковыми выводами, которая помимо прочего может быть использована как коммутационная панель. Этот модуль кроме FTDI может быть приобретен и у других компаний:

- ❒ AK Modul Bus на **http: // www.ak-modul-bus.de**;
- ❒ электронный магазин на **http: //www.elmikro.com/de/ft232r.html**;
- ❒ Farnell на **http: //at.farnell.com/jsp/search/productdetail.jsp?sku=1146036**;
- ❒ AO Unitronik на **http: // p5676.typo3server.info/ftdi-r.0.html**[1].

Модуль UM232R (рис. 3.6) предназначен для подключения к DIP-панельке и делает доступными все последовательные линии данных и управляющие линии с подтверждением микросхемы FT232R. Наряду с питанием от USB эти модули могут быть с собственным питанием (Self-Powered), а также с внешним напряжением питания 5 и 3,3 В.

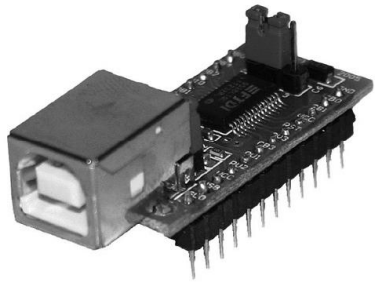
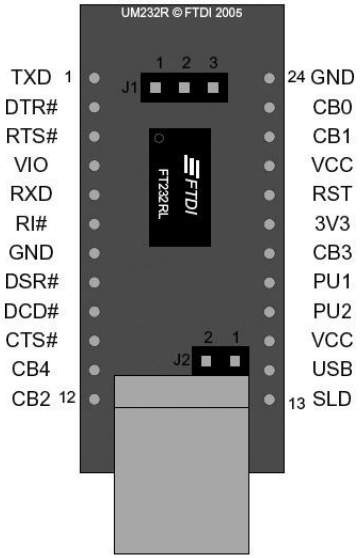
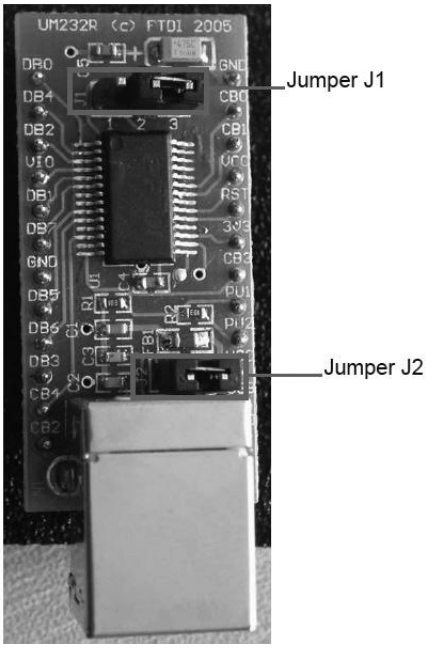


Рис. 3.6

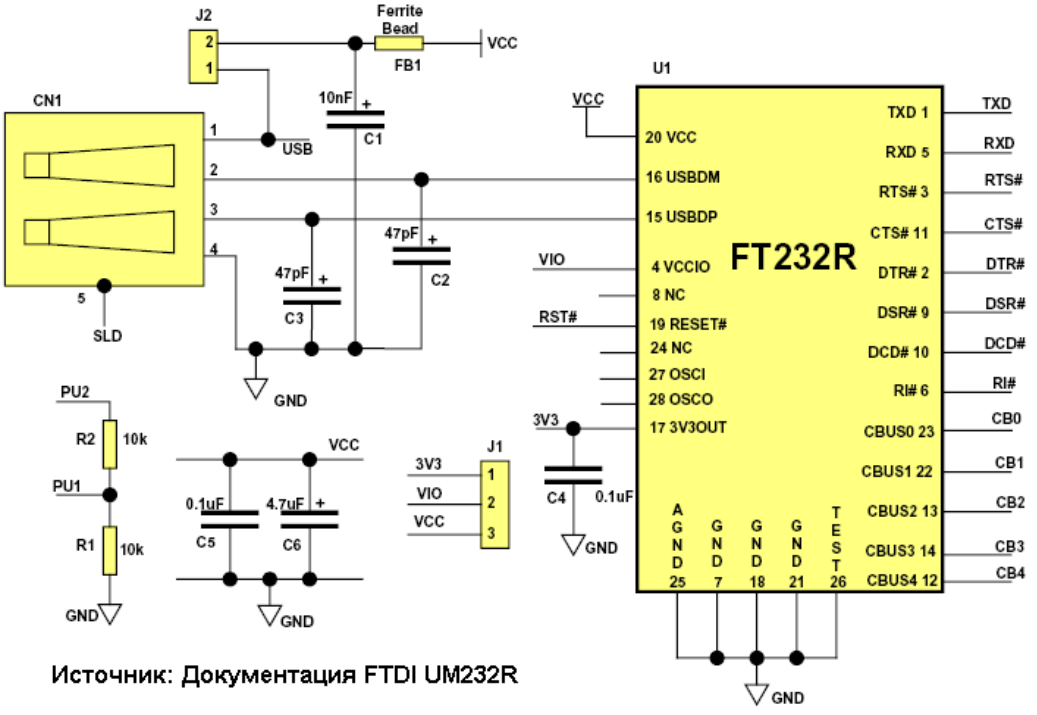
[1] Для России — http://www.mitracon.ru/, http://www.chip-dip.ru.





Источник: Документация FTDI UM232R

Рис. 3.7



Источник: Документация FTDI UM232R

Рис. 3.8

Примечание

На плате нет преобразователя уровня интерфейса RS-232.

Имеются и другие модули на микросхеме FT232R, как, например, модуль MM232R для последовательного интерфейса, модуль UC232R (ChiPi) с разъемом DB9 или кабельный адаптер TTL232R на микросхеме FT232R и с USB-разъемом. Вид сверху на модуль UM232R приведен на рис. 3.7, а электрическая схема модуля UM232R представлена на рис. 3.8 (рисунки взяты из документации компании FTDI).

3.4. Внутренняя структура микросхемы FT232R от FTDI

Функциональные блоки микросхемы FT232R представлены на рис. 3.9.

Как уже упоминалось ранее, используемый в книге FT232R-контроллер, изготовленный компанией FTDI, эмулирует последовательный интерфейс для USB-подключения.

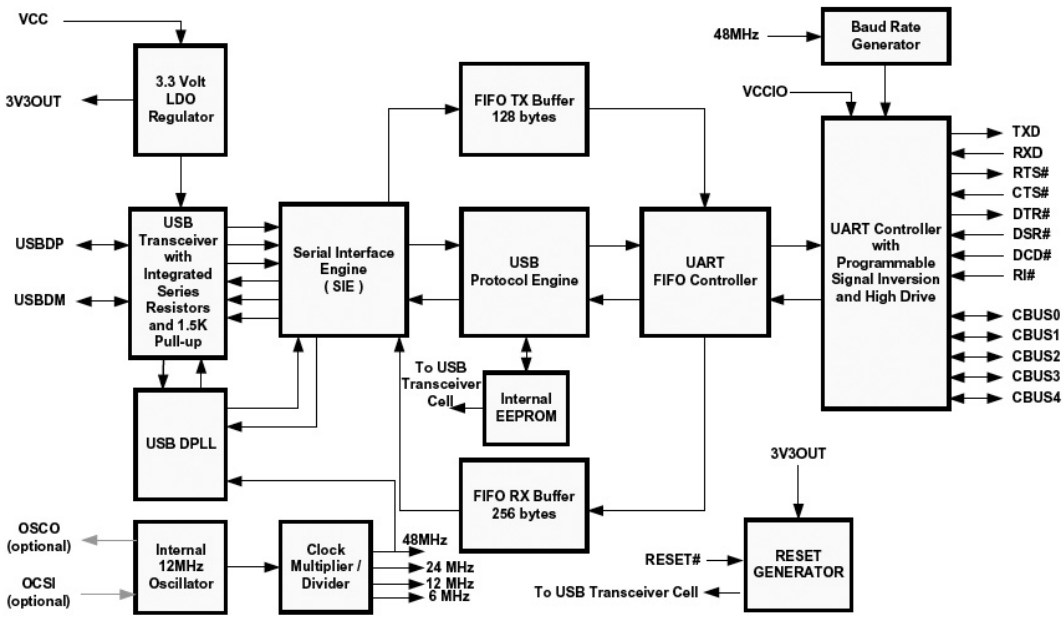


Рис. 3.9

Сигнальные линии USB

Сигнальные линии D+ и D– находятся на левой стороне блок-схемы микросхемы FT232R и именуются соответственно USBDP (USB Data Plus) и USBSM (USB Data Minus).

Микросхема FT232R имеет внутренний стабилизатор напряжения 3,3 В, который напрямую соединен с линией напряжения питания USB Vcc и общей линией GND.

Сигнальные линии последовательного интерфейса

Сигнальные линии на правой стороне блок-схемы являются сигналами последовательного интерфейса (табл. 3.2).

Таблица 3.2

Последовательный сигнал	Управление в качестве		Bit Bang	
TxD	Выход	Передача данных	D0	Вход-выход
RxD	Вход	Прием данных	D1	Вход-выход
RTS#	Выход		D2	Вход-выход
CTS#	Вход		D3	Вход-выход
DTR#	Выход		D4	Вход-выход
DSR#	Вход		D5	Вход-выход
DCD#	Вход		D6	Вход-выход
RI	Вход		D7	Вход-выход

Эти восемь сигнальных линий выведены на штырьковый соединитель USB-адаптера. Сигналы могут быть использованы также в так называемом режиме Bit Bang микросхемы FTDI-FT232R, тогда их следует пронумеровать от D0 до D7 (см. рис. 2.3 и 2.4).

Указание для электронщиков

Входы и выходы имеют внутренние подтягивающие резисторы (pull-up) с сопротивлением 200 кОм, подключенные к напряжению питания, каждый из которых при использовании в качестве входа становится активным.

Имеется также еще пять других сигналов CBUS от CBUS0 до CBUS4. CBUS 1 управляет работой светодиодов на плате USB, CBUS 3 — внешним напряжением питания.

3.5. Функции микросхемы FT232R

На блок-схеме, приведенной на рис. 3.9, можно увидеть множество функциональных блоков:

- **USB Transceiver** (USB-приемопередатчик) — осуществляет соединение с дифференциальными линиями данных USB D+ и D–. Этот функциональный блок имеет сопротивления для того, чтобы идентифицировать микросхему FT232R в качестве полноскоростного устройства (Full-Speed).

Из дифференциальных данных подготавливаются данные для блоков USB-DPLL и SIE (Serial Interface Engine — механизм последовательного интерфейса);

- **USB DPLL** (цифровая система фазовой автоподстройки частоты) — синхронизирует поток данных USB с блоком SIE;
- **Serial Interface Engine (SIE)** (механизм последовательного интерфейса) — выполняет преобразование последовательного потока данных USB в параллельные данные для дальнейшей обработки. Если передается сообщение от микросхемы FT232R на шину USB, то происходит обратное кодирование, из параллельных данных возникает поток последовательных данных USB;
- **USB Protocol Engine** (механизм протокола USB) — в этом блоке подготавливаются команды для последовательного интерфейса, а также обрабатываются запросы команд USB от USB-хост-контроллера;
- **FIFO TX/RX Buffer** — FIFO-буферы управляются FIFO-контроллером. В буфере FIFO (сокращение от First In — First Out, первым вошел — первым вышел) запоминаются передаваемые (FIFO TX) и соответственно принимаемые данные (FIFO RX), благодаря этому можно увеличить скорость приема и передачи данных;
- **UART Controller** (Universal Asynchronous Receiver and Transmitter, контроллер универсального асинхронного приемопередатчика, УАПП) — это блок, предназначенный для непосредственного вывода последовательных выходных сигналов. Работая совместно с блоком Baud Rate Generator и UART FIFO Controller, подготавливает сигналы для последовательного интерфейса, а также приходящие сигналы от последовательного интерфейса;
- **Baud Rate Generator** (генератор скорости передачи данных) — генерирует тактирующий сигнал для соответствующей скорости передачи в бодах последовательного интерфейса от 300 бод до 3 Мбод. В режиме Bit Bang параллельные данные выдаются согласно тактовой частоте при выбранной скорости передачи;
- **Internal 12 MHz Oscillator** (внутренний генератор) — генерирует все необходимые тактовые импульсы для внутренних функциональных блоков. Совместно с блоком умножителя/делителя частоты может использоваться в качестве задающего генератора для внешних устройств, при этом значение выходной частоты может быть 6, 12, 24 и 48 МГц;
- **Internal EEPROM** — внутренняя EEPROM-память на 1024 бит сохраняет конфигурационные данные для USB, CBUS-конфигурации и другие сведения, такие как PID, VID или серийный номер. От компании FTDI можно получить специальную программу под названием MPROG, с помощью которой можно изменить конфигурацию микросхемы FT232R, а также данные в EEPROM-памяти.

3.6. Пример последовательного подключения микроконтроллера к USB

При помощи FTDI-контроллера можно довольно быстро и без особых затрат выполнить адаптер между последовательным интерфейсом и USB. Для самой простой

передачи сообщений микроконтроллера к USB требуются только сигналы TxD, RxD и заземляющая шина.

Для этого можно использовать и готовый USB-адаптер (см. рис. 3.2). Сигнал TxD USB-адаптера (вывод 7 штырькового соединительного разъема) нужно подать на вход RxD микроконтроллера, а сигнал RxD USB-адаптера (вывод 6 штырькового соединительного разъема) на выход TxD микроконтроллера. После этого останется выполнить только лишь соединение общей линии (соединения с "землей") контроллера с выводом 2-штырькового соединительного разъема (табл. 3.3).

Таблица 3.3

Последовательный сигнал	**Выводы штырькового разъема USB-адаптера**		**Микроконтроллер**	
TxD	Вывод 7 (выход)	Передача данных →	RxD	Вход
RxD	Вывод 6 (вход)	← Прием данных	TxD	Выход
"земля"	Вывод 2			"земля"

После установки FTDI-драйвера через (виртуальный) COM-порт может осуществляться соединение с микроконтроллером.

Схема, представленная на рис. 3.10, показывает не только простое последовательное соединение, но также и то, как микроконтроллер одновременно тактируется

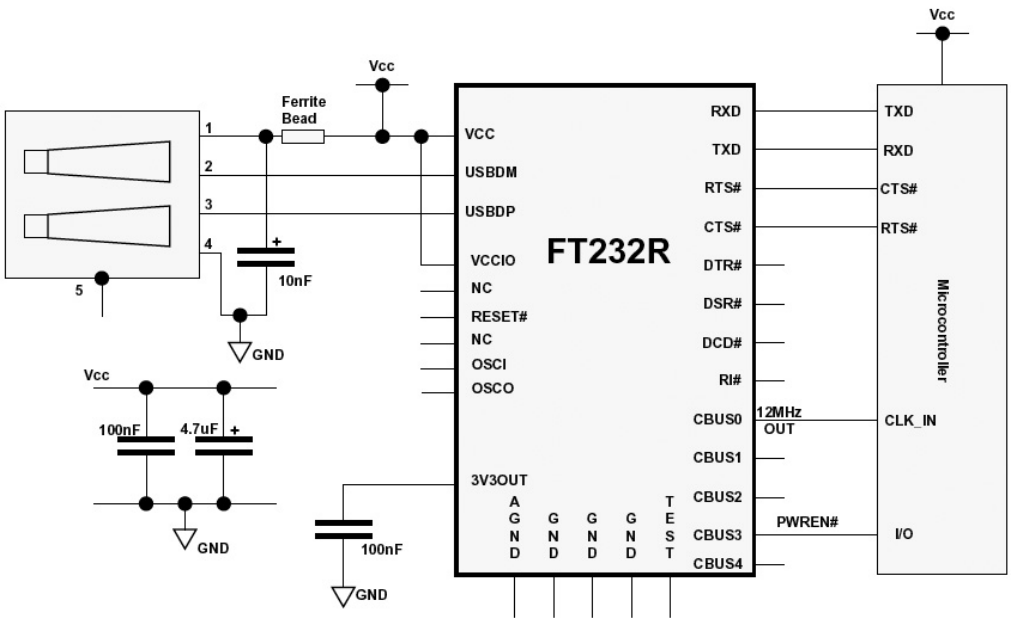


Рис. 3.10

частотой 12 МГц, полученной на выходе CBUS0 микросхемы FT232R, и управляется сигналом PWREN# в режиме низкого потребления (Suspend mode).

3.7. Согласование уровней напряжения RS-232/485

Если вы хотите использовать FTDI-чип в качестве преобразователя сигналов USB в сигналы последовательного интерфейса RS-232 или RS-485, то следует обязательно выполнить согласование уровней напряжения. Микросхемы серии 213 (Sipex SP213ECA, Maxim MAX213CAI или аналоговые устройства ADM213E) содержат необходимые аппаратные усилители-формирователи для четырех передаваемых и пяти принимаемых сигналов интерфейса RS-232. Эти элементы дополнительно поддерживают режим отключения (Shut Down Mode), таким образом для сохранения USB-соглашений они могут переводиться в экономичный режим ожидания ПК (рис. 3.11).

Если требуется только 4 сигнальных линии RXD, TXD, RTS и CTS (и не нужна поддержка режима Suspend), то вы можете использовать преобразователь уровней Max232 или ST232.

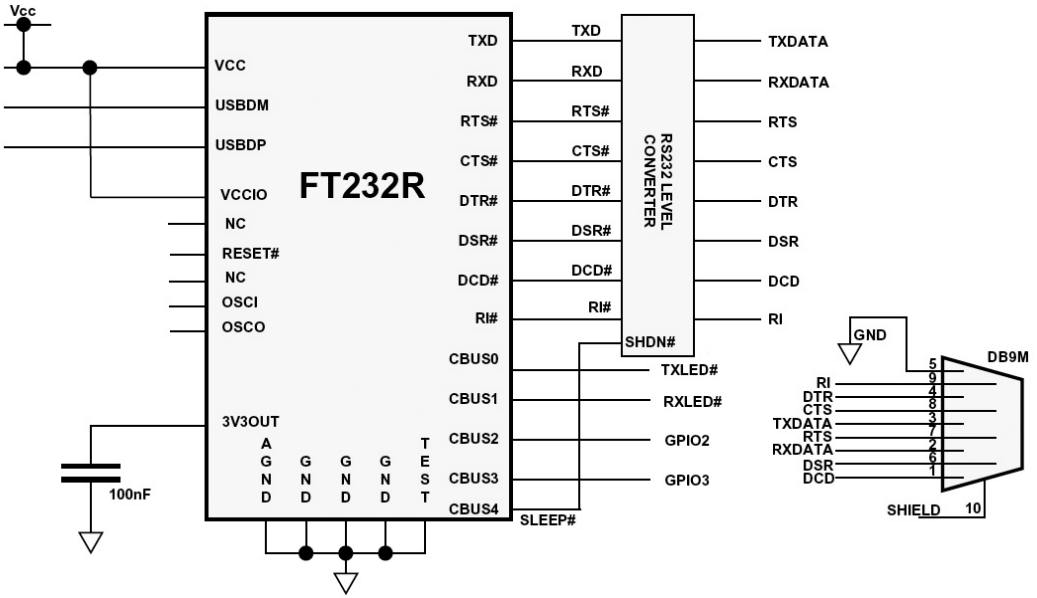


Рис. 3.11

Схема с Max232 довольно простая (рис. 3.12). Изображенные электролитические конденсаторы имеют емкость 1 мкФ.

В интерфейсе RS-485 сигналы TxD и RXD передаются как дифференциальные сигналы. Сигнал CBUS2 (TXDEN) активирует *усилитель-формирователь передатчи-*

ка, а сигнал CBUS3 (PWREN) — усилитель-формирователь приемника (рис. 3.13). Микросхема SP481 здесь приведена только для примера, такие же элементы имеются и в устройствах компании Maxim или Analog Devices.

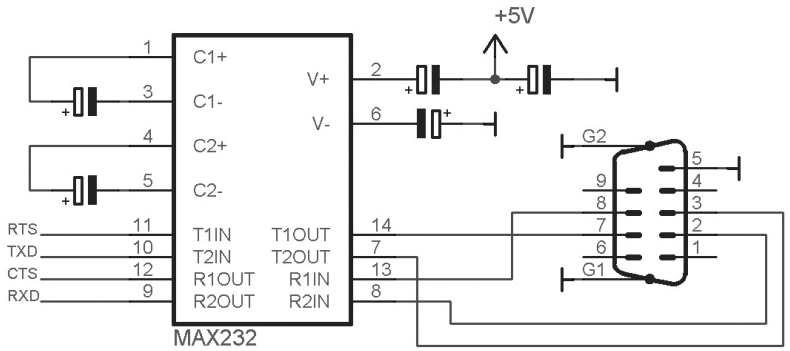


Рис. 3.12

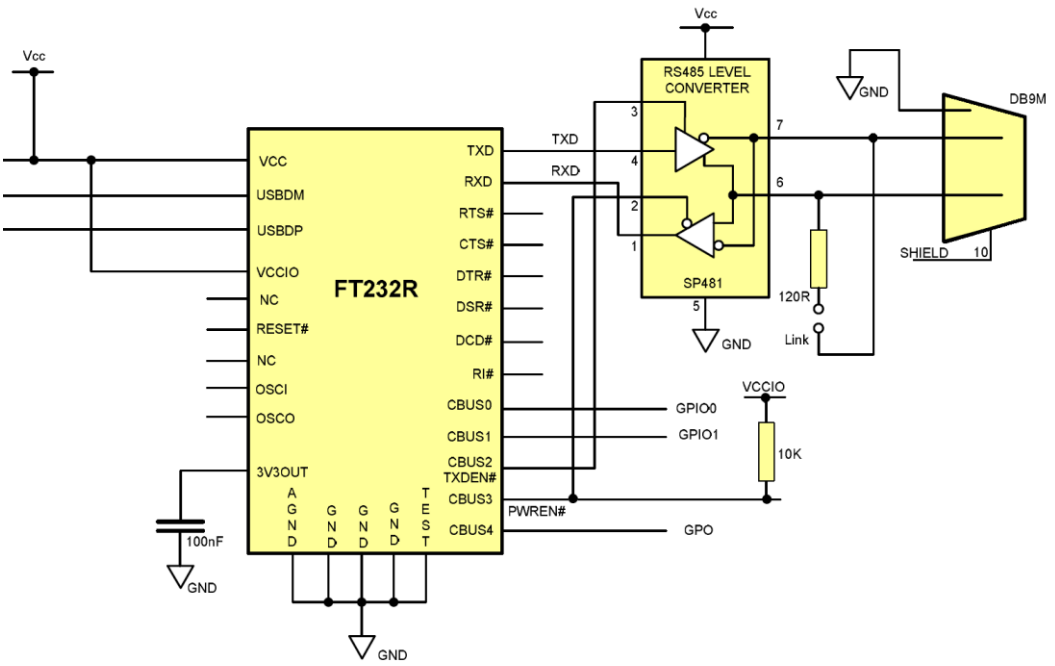


Рис. 3.13

3.8. Генератор скорости передачи данных

Для передачи данных в последовательном USB-адаптере особенно важным является настройка скорости передачи в бодах. Компания FTDI поддерживает все стандартные скорости передачи в бодах (300, 600, 1200, 2400, 4800, 9600, 14 400, 19 200, 38 400, 115 200, 230 400, 460 800 и т. д.). Скорости передачи в бодах обычно устанавливаются в прикладной программе или в ОС. Скорости передачи данных можно установить при помощи вызова функции `FT_SetBaudRate`.

Если раньше хотели использовать нестандартные скорости в бодах, то прежде всего нужно было детально разобраться в процессе генерирования скорости передачи данных и возможных коэффициентах деления. Коэффициенты деления после пересылки скорости в бодах также должны были передаваться при помощи вызова функции `FT_SetDivisor`.

В новых версиях драйверов компании FTDI этого больше не требуется: достаточно просто ввести скорость передачи (например, 49 000), и драйвер автоматически определяет необходимые оптимальные настройки.

Чтобы обойти процедуру настройки скорости в бодах (допустим, нужно перенастроить скорость передачи с 115 на 512 K), потребуется файл FTDIPort.inf. Там вы найдете запись `[FtdiPort232.NT.HW.AddReg]`. В записи

```
HKR ConfigData ,1,01,00,3F,3F,10,27,88,13
```

каждой скорости передачи сопоставляются два последовательных байта. Пара `10,27` соответствует скорости передачи 300, а `88,13` — скорости передачи 600 бод, и т. д.

Все стандартные значения скоростей указаны в руководстве компании FTDI (в файле FTDI_AN232B-05_BaudRates.pdf на компакт-диске) в разделе "Aliasing Using the Original Sub-Integer Divisors".

> ***Указание для "микроэлектронщика"***
>
> Существует несколько программ, которые используют последовательный порт для программирования микроконтроллеров. При этом выходные сигналы TxD, RTS и DTR управляются непосредственно флэш-программой. Поскольку передача каждого кадра в USB обычно осуществляется в цикле продолжительностью в 1 мс, то сигнальные линии USB могут быть изменены "только" каждую миллисекунду, тогда как при нормальном последовательном интерфейсе это выполняется значительно быстрее.

В этом случае в принципе можно использовать преобразователь интерфейса USB в последовательный интерфейс, но передача данных при этом будет со скоростью несколько байтов в секунду.

ГЛАВА 4

Установка драйвера FTDI версии 2.x

Условием выполнения ваших экспериментов и функционирования имеющихся тестовых программ является успешная установка драйвера.

Прежде чем вы подсоедините USB-адаптер компании FTDI, в диспетчере устройств Windows следует проверить, не используете ли вы уже USB-устройство с микросхемой FTDI.

Если используете, то в этом случае вы обязательно должны удалить существующий FTDI-драйвер, чтобы иметь возможность установить прилагаемый драйвер.

Обычно FTDI-драйвер в операционных системах Windows можно удалить, воспользовавшись меню **Пуск | Настройка | Панель управления | Установка и удаление программ**.

Иногда все же деинсталляцию провести не удается, тогда при необходимости можете попробовать воспользоваться утилитой FTClean, которую можно найти на сайте **www.ftdichip.com**. Наконец, вы можете также еще раз запустить диспетчер устройств Windows и проверить, можно ли удалить драйвер для соответствующего устройства. В завершение попытайтесь снова удалить драйвер из меню **Пуск | Настройка | Панель управления | Установка и удаление программ**.

Пожалуйста, не применяйте для своих экспериментов USB-хаб, а подсоединяйте USB-адаптер непосредственно к свободному USB-порту компьютера!

Если подключить USB-адаптер к свободному разъему USB-интерфейса, то светодиод на адаптере ненадолго засветится. Мастер установки уведомит вас, что было найдено новое оборудование с именем FT232 USB UART, и в следующем окне вам будет предложено установить драйвер нового устройства (рис. 4.1—4.3).

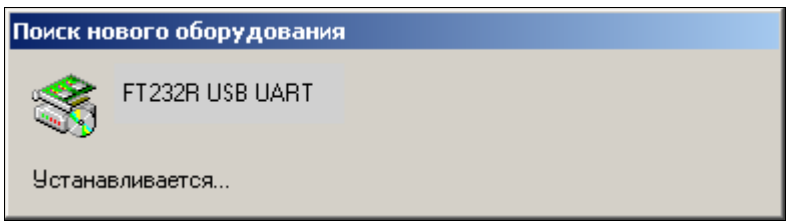


Рис. 4.1

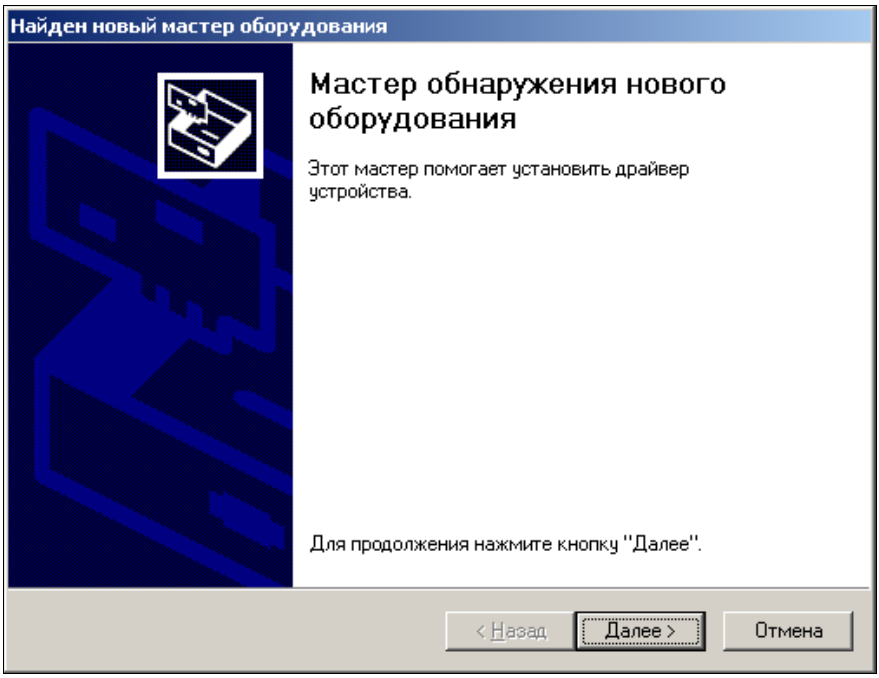

Найден новый мастер оборудования
Мастер обнаружения нового оборудования
Этот мастер помогает установить драйвер устройства.
Для продолжения нажмите кнопку "Далее".
< Назад
Далее >
Отмена

Рис. 4.2

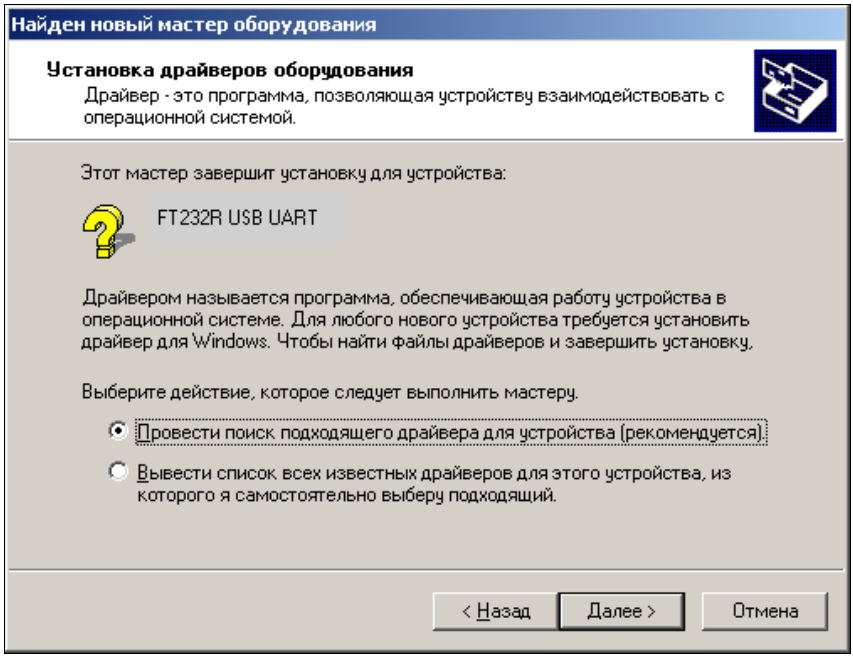

Найден новый мастер оборудования
Установка драйверов оборудования
Драйвер - это программа, позволяющая устройству взаимодействовать с операционной системой.
Этот мастер завершит установку для устройства:
FT232R USB UART
Драйвером называется программа, обеспечивающая работу устройства в операционной системе. Для любого нового устройства требуется установить драйвер для Windows. Чтобы найти файлы драйверов и завершить установку,
Выберите действие, которое следует выполнить мастеру.
Провести поиск подходящего драйвера для устройства (рекомендуется)
Вывести список всех известных драйверов для этого устройства, из которого я самостоятельно выберу подходящий.
< Назад
Далее >
Отмена

Рис. 4.3

Выберите, пожалуйста, переключатель **Провести поиск подходящего драйвера для устройства** и в следующих диалоговых окнах — каталог с драйвером. Каталог вы можете либо выбрать из обзора, либо задать вручную. Если ваш компакт-диск, прилагаемый к книге, имеет, например, логическое имя `D:`, то следует указать следующий путь `D:\FTDI Treiber` (рис. 4.4—4.6).

Для новых драйверов FTDI версий 2.x эту программу установки придется запустить дважды.

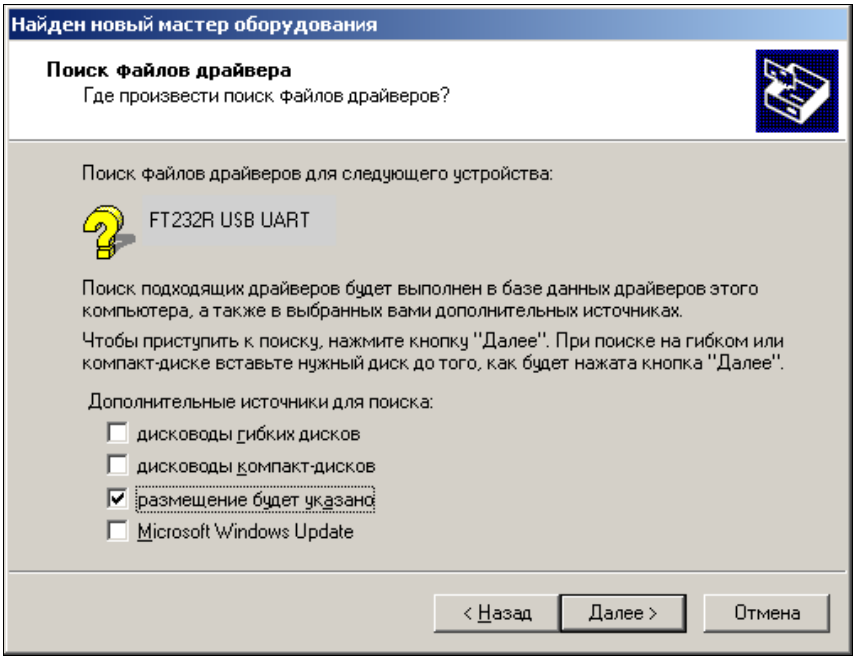


Рис. 4.4

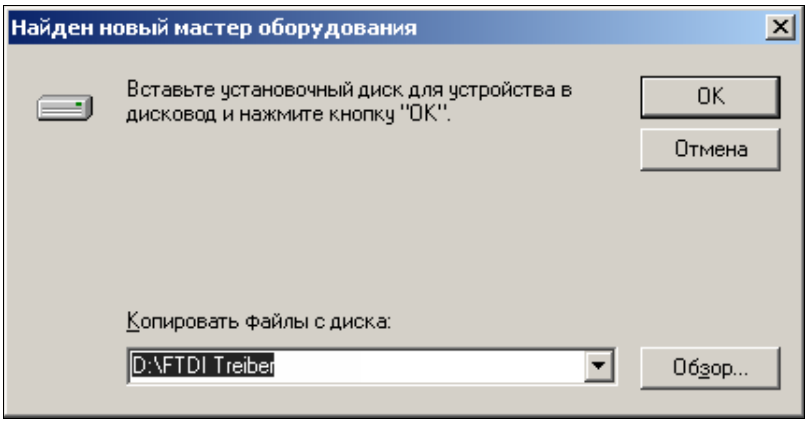


Рис. 4.5

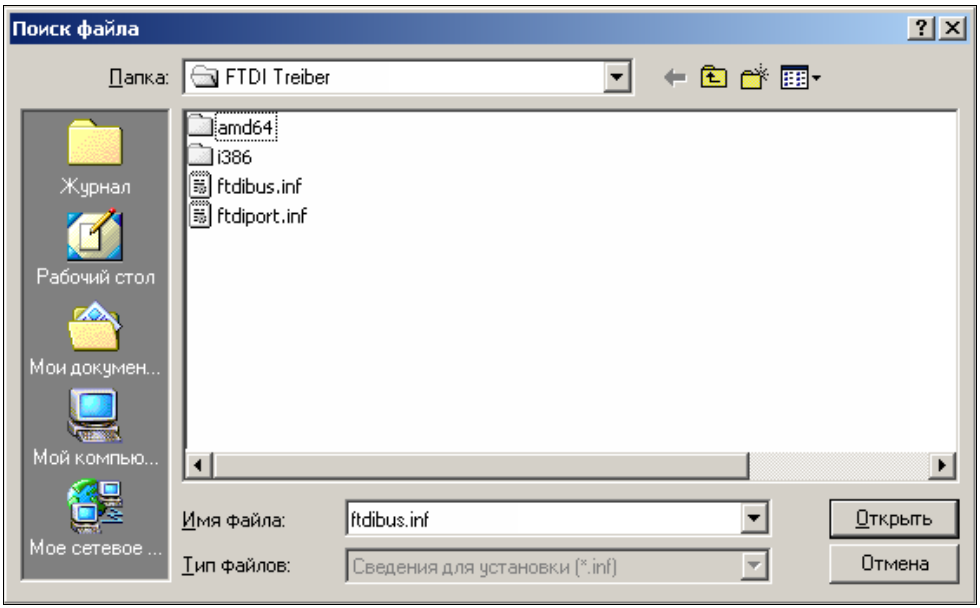


Рис. 4.6

После успешной установки драйвера FTDI можно проверить наличие установленного устройства в диспетчере устройств: в строке **Порты COM и LPT** вы должны увидеть (VCP)-устройство **USB Serial Port (COMx)** (рис. 4.7). Теперь можно обратиться к COM-порту через USB.

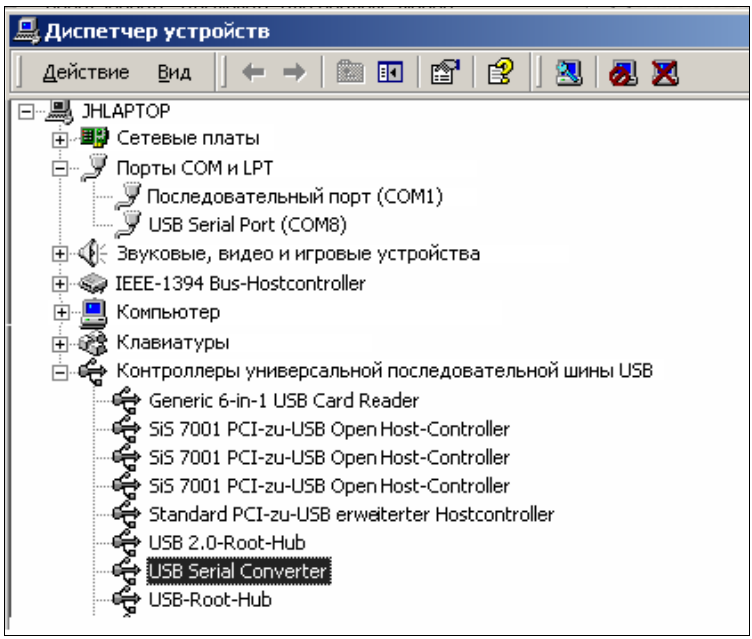


Рис. 4.7

В списке контроллеров универсальной последовательной шины USB вы должны найти запись: **USB Serial Converter**.

Указание

Если в строке **Порты COM и LPT** вы не увидите устройство **USB Serial Port (COMx)**, то в окне свойств для USB Serial Converter на вкладке **Advanced** (Дополнительно) можно установить флажок **Load VCP** (Загрузить VCP).

4.1. Программа для отображения USB-устройств (утилита USB View)

Из Интернета можно загрузить бесплатную утилиту фирмы Microsoft для отображения устройств, подключенных к шине USB. Программа находится на компакт-диске, прилагаемом к книге, в папке \USBView.

С помощью этой программы вы можете увидеть отображение не только подключенных USB-устройств, но также и основные их идентификаторы (рис. 4.8).

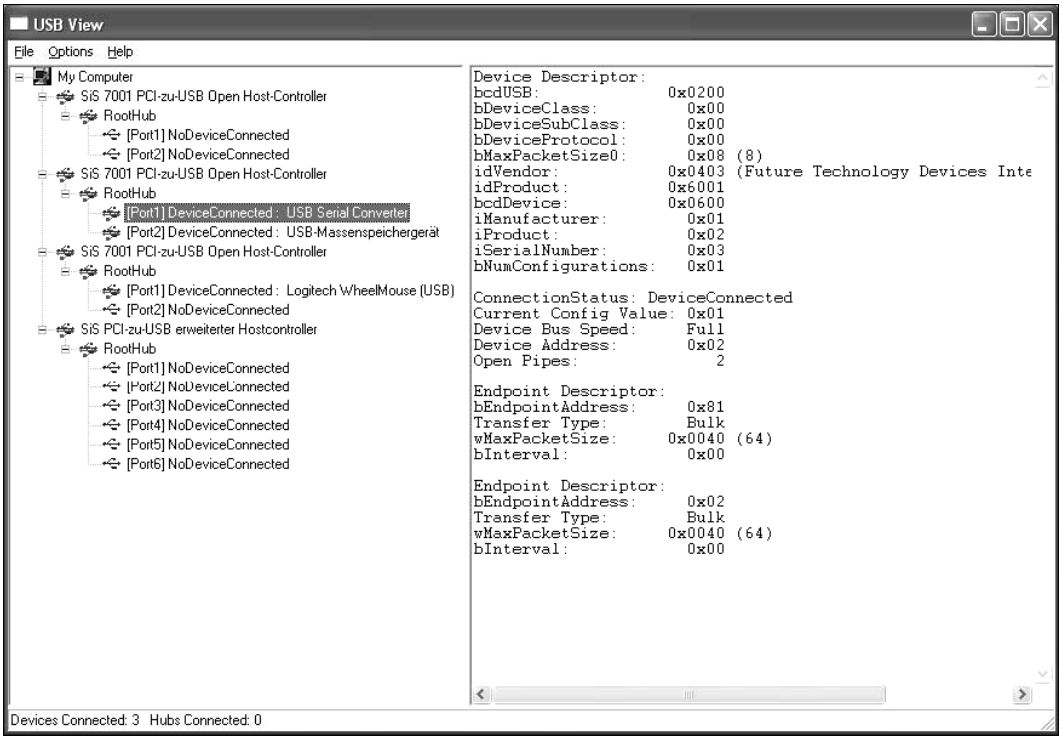


Рис. 4.8

Если в левой части окна вы выберете устройство USB Serial Converter, то в правой части окна вы получите подробные данные конфигурации FTDI-микросхемы. Например, для `bcdUSB` значение `0x200` соответствует USB 2.0, FTDI-USB идентифика-

тор производителя (idVendor) — 0x0403 и FTDI-USB идентификатор изделия (idProduct) — 0x6001, а также два коммуникационных канала с передачей типа Bulk (wMaxPacketSize) и максимальной величиной пакета 64 байта. Значения bEndpointAddress различаются при сообщении в бите данных D7, который указывает направление 0 = OUT или 1 = IN.

4.2. Удаление FTDI-драйвера

Обычно FTDI-драйвер в операционных системах Windows можно удалить, воспользовавшись меню **Пуск | Настройка | Панель управления | Установка и удаление программ**.

Подробную информацию (на английском языке) вы сможете найти также в документе Windows_XP_Installation_Guide.pdf в папке \FTDI_Treiber с драйвером на компакт-диске.

> ***Указание***
>
> При необходимости используйте в своей работе утилиту FTClean (**www.ftdchip.com**), если удаления почему-либо не происходит.

Когда вы подготавливаете свои эксперименты, вы обязательно заранее должны отсоединить USB-адаптер от USB-порта вашего компьютера, чтобы избежать возможного короткого замыкания!

Когда после подготовки эксперимента вы снова подключите USB-адаптер, то драйвер заново будет интегрирован операционной системой, и после этого вы сможете непосредственно запустить ваши программы.

ГЛАВА 5

Начало работы

Если установка драйвера прошла успешно, то вы можете отважиться на первый тест.

Все демонстрационные программы содержатся в виде исполняемых exe-файлов на прилагаемом к книге компакт-диске. Рекомендуется скопировать каталог \Beispielprogramme с демонстрационными программами на жесткий диск.

После копирования не забудьте при необходимости снять у отдельных файлов защиту от записи.

В качестве языков программирования (создание простых прикладных программ для USB-адаптера) могут быть приняты в расчет, например, Visual Basic или Pascal. Для программирования на C потребуются солидные навыки программирования. Зачастую, в особенности у начинающих, в распоряжении нет C-компилятора. Предлагаемые демонстрационные программы на Visual Basic (VB) можно относительно легко преобразовать на другие языки программирования.

Для разработки собственных прикладных программ для относительно простого управления микросхемой FT232R все необходимые подпрограммы должны находиться в программной библиотеке FTD2XX.DLL. Процесс обмена информацией между приложением и USB-устройством (FT232RL) выглядит примерно так, как показано на рис. 5.1.

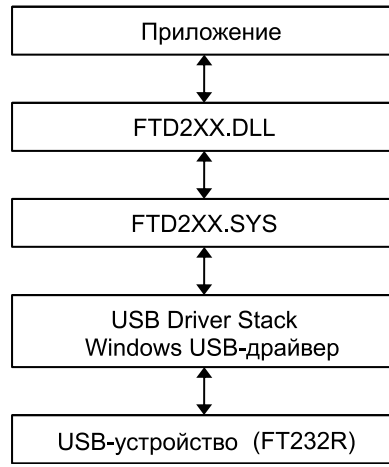


Рис. 5.1

Файл WDM-драйвера FTD2XX.SYS и файл библиотеки FTD2XX.DLL были установлены вместе с FTDI-драйверами. WDM-драйвер связывается, с одной стороны, со стеком драйвера Windows USB Stack, чтобы иметь доступ к физическому устройству USB, а с другой стороны — с FTDI-библиотекой FTD2XX.DLL.

Важно!

Классические функции Win32 API (Application Programming Interface, интерфейс для программирования прикладных программ) ни в коем случае не должны быть спутаны с API-функциями FTDI-драйвера!

5.1. Вызов первой демонстрационной программы на Visual Basic (VB)

Вы подключили USB-адаптер? Тогда запустите программу beispiel_1.exe, находящуюся в папке демонстрационных программ \Beispielprogramme\Bsp_1.

Указание

Если при запуске программы возникает ошибка времени выполнения, то сначала следует выполнить программу setup.exe в каталоге \Beispielprogramme\BSp_1, чтобы установить модуль программной среды VB.

Нажмите кнопку **Test** — вы должны получить результат, представленный на рис. 5.2.

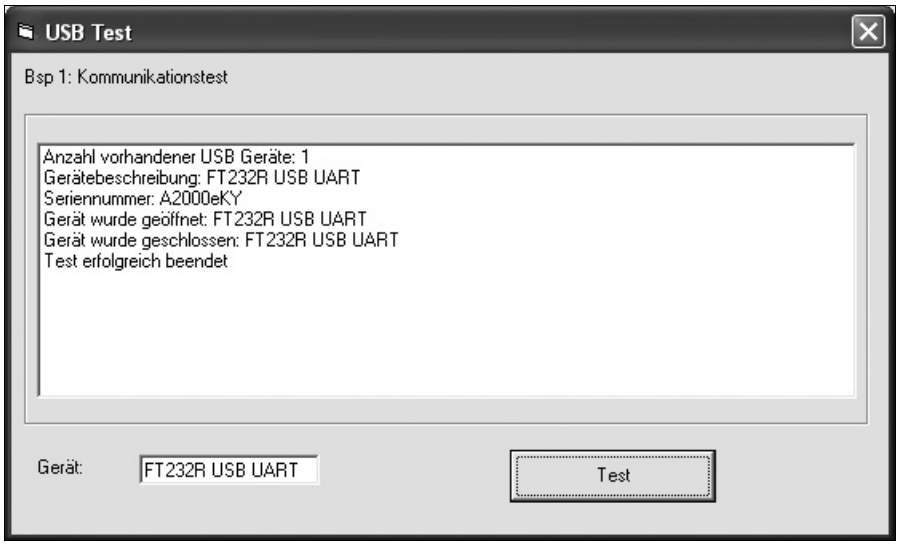


Рис. 5.2

Результаты работы первой демонстрационной программы будут представлены в небольшой области окна **USB Test**.

Программа сначала определяет количество FTDI-устройств, имеющихся в наличии, затем следуют описания устройства и серийные номера. Далее программа пытается открыть канал связи для USB-адаптера (чтобы читать и отсылать сообщения). В заключение канал связи снова будет закрыт.

Строка **Gerätebeschreibung: FT232R USB UART** (описание устройства) может быть изменена. Если USB-устройство подсоединено не как отдельное (FTDI)-USB-

устройство (и поэтому не появляется в качестве первого USB-устройства в списке), то в программе для тестирования связи с устройством используется описание (наименование) устройства FT232R USB UART.

Если же вы используете драйвер с другим описанием устройства, нежели FT232R USB UART, то тогда вы соответственно должны изменить это указание в программах для примеров в поле **Gerät** (устройство).

5.2. Первые обращения программы к FTD2XX.DLL-библиотеке

Как было описано в начале главы, для процесса передачи информации используются функции FTDI-библиотеки FTD2XX.DLL.

Все вызовы функций описаны в руководстве "D2XX Programmer's Guide". Соответствующий PDF-документ имеется на компакт-диске в файле \FTDI_Manuals \D2XXPG34.pdf.

В первой демонстрационной программе на Visual Basic (VB) используются вызовы функций библиотеки, представленные в табл. 5.1.

Таблица 5.1

Вызов функции	Краткое описание
`FT_GetNumDevices`	Позволяет получить количество USB-устройств (вызывает `FT_ListDevices`)
`FT_ListDevices`	Позволяет получить количество USB-устройств, описаний устройств и серийных номеров подключенных устройств
`FT_OpenEx`	Открывает канал связи для данного USB-устройства
`FT_Close`	Закрывает канал связи

5.3. Пример программы на Visual Basic

Все VB-программы записаны на компакт-диске в исходном коде. Если вы установили VB, то просто вызовите файл проекта на Visual Basic с расширением VBP, и тогда проект откроется: в данном случае это файл Bsp.VBP в папке \Beispielprogramme\Bsp_1 (рис. 5.3). Чтобы открыть в VB исходную программу первого примера, нужно дважды щелкнуть на файле формы Beispiel 1 в окне проводника проекта (в правом верхнем окне на рис. 5.4 обведен овалом), исходный код будет показан в отдельном окне.

Имя кнопки **Test** — `bt_test`. Если пользователь приводит в действие кнопку, то вызывается функция `bt_test_Click()`.

Если окно проводника проекта не открылось в верхнем правом углу, то это окно вы можете вызвать при помощи команды из меню **View | Project Explorer** или после нажатия комбинации клавиш <Ctrl>+<R>.

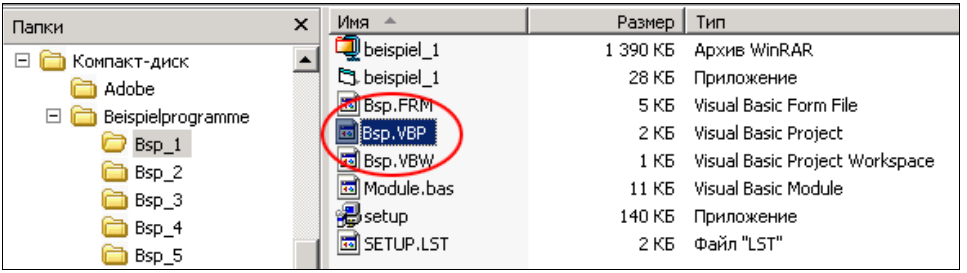


Рис. 5.3

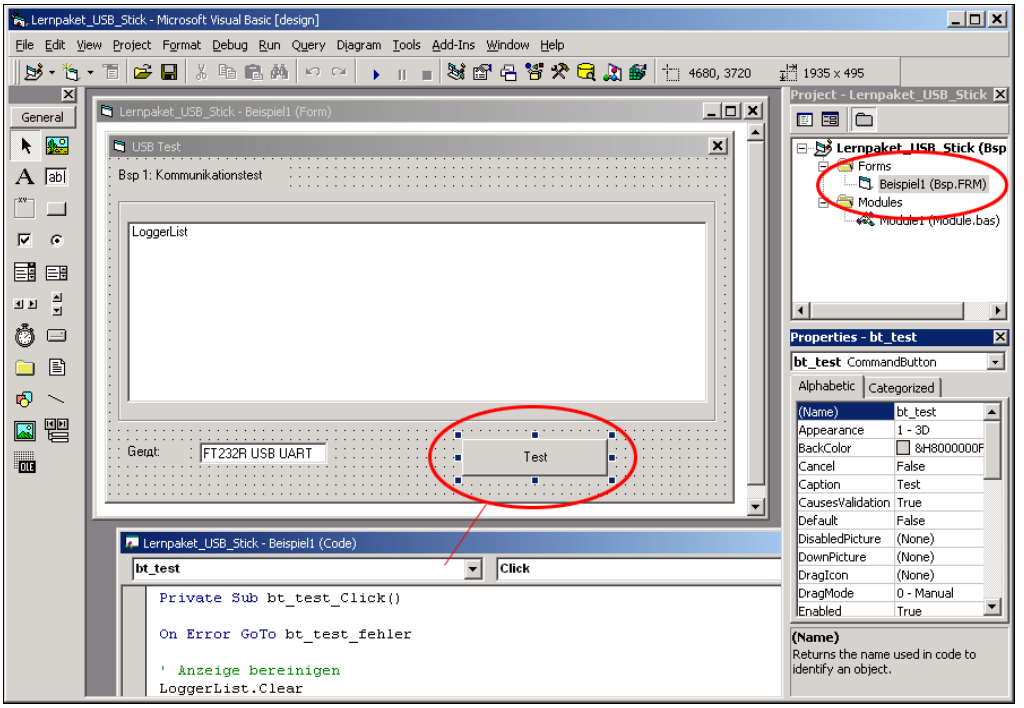


Рис. 5.4

5.4. Объявление функций драйвера FTD2XX для Visual Basic

Чтобы сообщить Visual Basic, что должны быть вызваны функции во внешней библиотеке, их нужно объявить. Объявления содержатся в `Modul1` (в области объекта `Module` (модули) в окне проводника проекта VB). Здесь должны быть произведены два объявления:

```
Public Declare Function FT_ListDevices Lib "FTD2XX.DLL" ( _
       ByVal arg1 As Long, _
       ByVal arg2 As String, _
       ByVal dwFlags As Long) As Long
```

```
Public Declare Function FT_GetNumDevices Lib "FTD2XX.DLL" Alias
"FT_ListDevices" ( _
       ByRef arg1 As Long, _
       ByVal arg2 As String, _
       ByVal dwFlags As Long) As Long
```

Обращение к функции `FT_GetNumDevices` вызывает функцию `FT_ListDevices` в FTDI-библиотеке FTD2XX.DLL.

Затем зачастую объявляются используемые в модуле константы, например:

```
' для FT_OpenEx
Public Const FT_OPEN_BY_SERIAL_NUMBER = 1
Public Const FT_OPEN_BY_DESCRIPTION = 2
```

Имена констант соответствуют в большинстве своем описанию в руководстве "D2XX Programmer's Guide" от компании FTDI (на компакт-диске в файле \FTDI_Manuals\D2XXPG34.pdf).

5.5. Исходный код функций *FT_ListDevices* и *FT_OpenEx*

Следующая подпрограмма вызывается в случае, если нажата кнопка **Test** в форме VB во время выполнения программы (листинг 5.1).

Листинг 5.1

```
Private Sub bt_test_Click()
On Error GoTo bt_test_fehler
LoggerList.Clear
If FT_GetNumDevices(lngNumDevices,vbNullString,FT_LIST_BY_NUMBER
     _ONLY)<>FT_OK Then LoggerList.AddItem ("ошибка вызова:
     FT_GetNumDevices не работает")
     Exit sub
   Else
     LoggerList.AddItem ("Anzahl vorhandener USB-Geräte
             (количество устройств): " & lngNum Devices)
End If
If FT_ListDevices(0,strBeschreibung,FT_LIST_BY_INDEX Or FT_OPEN_
      BY_DESCRIPTION)_
      <> FT_OK Then
     LoggerList.AddItem ("ошибка вызова:
     FT_ListDevices_ не работает")
     Exit Sub
  Else
     LoggerList.AddItem ("Gerätebeschreibung (описание устройства):
                                                  " & strBeschreibung)
End If
```

```
If FT_ListDevices(0,strSerienNummer,FT_LIST_BY_INDEX Or FT_OPEN_
     BY_SERIAL_NUMBER) _<> FT_OK Then
     LoggerList.AddItem ("ошибка определения серийного номера:
     FT_ListDevices не работает")
     Exit Sub
  Else
     LoggerList.AddItem ("Seriennummer (серийный номер):
                                          " & strSerienNummer)
End If
If Left$(strBeschreibung, 15) <> "FT232R USB UART" Then
     strBeschreibung = Trim(Me.DeviceName.Text) & Chr(0)
     LoggerList.AddItem "Alternative Gerätebeschreibung verwendet
     (альтернативное описание устройства)"
End If
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle) <>
       FT_OK Then
     LoggerList.AddItem "ошибка вызова: FT_OpenEx"
     Exit Sub
  Else
     LoggerList.AddItem "Gerät wurde geöffnet
     (канал для связи с устройством открыт): " & Trim(strBeschreibung)
End If
If FT_Close(lngHandle) <> FT_OK Then
     LoggerList.AddItem "Ошибка вызова: FT_Close"
     Exit Sub
  Else
     LoggerList.AddItem ("Gerät wurde geschlossen
     (канал для связи с устройством закрыт): " & Trim(strBeschreibung)
End If
     LoggerList.AddItem "Test erfolgreich beendet
                         (тест успешно завершен)"
     bt_test_fehler_ende:
     Exit Sub
bt_test_fehler:
     MsgBox Err.Description
     Resume bt_test_fehler_ende
End Sub
```

К отдельным шагам программы ведут ссылки:

```
On Error GoTo bt_test_fehler
```

Программа обработки ошибок начинается строкой:

```
LoggerList.Clear
```

и очищает окно вывода `LoggerList`.

```
If FT_GetNumDevices(lngNumDevices,vbNullString,
   FT_LIST_BY_NUMBER_ONLY)<>FT_OK Then
   LoggerList.AddItem ("Ошибка вызова: FT_GetNumDevices
   funktionierte nicht")
```

```
Else
   LoggerList.AddItem ("Anzahl vorhandener USB-Geräte: " & IngNumDevices)
End If
```

Функция `FT_GetNumDevices` вызывает функцию `FT_ListDevices` из библиотеки FTD2XX.DLL, чтобы сообщить число USB-устройств. Должны быть введены три значения параметров:

- `lngNumDevices` — был определен в заголовке программы и после успешного вызова функции содержит число USB-устройств;
- `vbNullString` — требуется при других возможных вызовах функции `FT_ListDevices`. Так как третий параметр должен сообщить количество устройств, то второй параметр должен быть установлен в 0;
- `FT_LIST_BY_NUMBER_ONLY` — третий параметр указывает, какие значения должны сообщаться и возвращаться функцией. `FT_LIST_BY_NUMBER_ONLY` объявлен в `Modul1` как константа и имеет значение `&H80000000`, которое в этой форме не является особенно понятным.

Функция `FT_GetNumDevices` после вызова — в соответствии с объявлением `Public Declare Funktion FT_GetNumDevices Lib "FTD2XX.DLL" Alias "FT_ListDevices" (....) As Long` может возвратить значение. В случае успеха вернется 0. Константа `FT_OK` объявлена в `Modul1`.

Условный оператор `IF` опрашивает, успешен ли вызов, и если функция после вызова возвращает 0 (`FT_OK`), в окне вывода отображается количество устройств. Если значение не равно 0, то вызов был ошибочным, выдается сообщение об ошибке и тестовая программа завершается командой `Exit Sub`.

5.6. Другие вызовы функции *FT_ListDevices*

```
If FT_ListDevices(0,strBeschreibung,FT_LIST_BY_INDEX Or FT_OPEN_
BY_DESCRIPTION) <> FT_OK
......
If FT_ListDevices(0,strSerienNummer,FT_LIST_BY_INDEX Or FT_OPEN_
BY_SERIAL_NUMBER) <> FT_OK
......
```

Помимо описанного ранее вызова функции, который сообщает о количестве USB-устройств, функция `FT_ListDevices` позволяет получить описание устройства или его серийный номер:

```
FT_ListDevices(0,strBeschreibung,FT_LIST_BY_INDEX Or_
                FT_OPEN_BY_DESCRIPTION)
```

Параметр `FT_OPEN_BY_SERIAL_NUMBER` называет серийный номер, который после вызова находится в строке `strSerienNummer`:

```
FT_ListDevices(0,strSerienNummer,FT_LIST_BY_INDEX Or_
                FT_OPEN_BY_SERIAL_NUMBER)
```

Строковые переменные `strBeschreibung` и `strSerienNummer` длиною по 256 символов содержат сведения о всех найденных USB-устройствах.

Приведенный пример программирования не вполне корректен. В качестве первого параметра функции (здесь 0 в качестве ID-устройства) он обращается только к первому USB-устройству. Если бы было подключено несколько USB-устройств компании FTDI, то тогда нужно было бы сначала определить их количество и после этого привести отдельные описания:

```
FT_ListDevices(0,strBeschreibung0,FT_LIST_BY_INDEX Or_
                 FT_OPEN_BY_DESCRIPTION)
FT_ListDevices(1,strBeschreibung1,FT_LIST_BY_INDEX Or_
                 FT_OPEN_BY_DESCRIPTION)
```

Код можно было бы также обобщить и в отдельном вызове функции: в качестве первого параметра нужно было бы задать указатель (на `Buffer Array`), во втором параметре — установленное число USB-устройств. Это следует принимать во внимание в случае, если вы разрабатываете собственные программы для сложной среды с несколькими USB-устройствами.

Если первым не стоит USB-адаптер с микросхемой FT232R от компании FTDI, перед тем как открыть канал для связи с USB-устройством, просто наберите:

```
If Left$(strBeschreibung, 15) <> "FT232R USB UART" Then
       strBeschreibung = Trim(Me.DeviceName.Text) & Chr(0)
   LoggerList.AddItem "альтернативное устройство"
End If
```

Описание "FT232R USB UART" получено из (изменяемого пользователем в тексте) поля формы `DeviceName`. Описание должно заканчиваться символом 0. Этот 0 присоединяется с `Chr (0)` к строке описания `strBeschreibung`.

Если же вы работаете с типом микросхемы FT232R от компании FTDI или если даже меняете описание в EEPROM-памяти, то нужно изменить описание устройства во время выполнения функции, чтобы можно было проверить эту тестовую программу.

5.7. Вызовы функций *FT_OpenEx* и *FT_Close*

Если прикладной программе необходимо обменяться информацией с USB-устройством, то для начала нужно открыть канал связи (листинг 5.2).

Листинг 5.2

```
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle) <>
                                                          FT_OK Then
     LoggerList.AddItem "Ошибка вызова: FT_OpenEx"
     Exit Sub
```

```
  Else
     LoggerList.AddItem "канал для связи с устройством открыт: " &
                                                    Trim(strBeschreibung)
End If
If FT_Close(lngHandle) <> FT_OK Then
     LoggerList.AddItem "Ошибка вызова: FT_Close"
     Exit Sub
  Else
     LoggerList.AddItem "канал для связи с устройством закрыт: " &
                                                      Trim(strBeschreibung)
End If
```

Канал для связи с устройством открывается при помощи функции `FT_OpenEx (strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle)` на основе описания в `strBeschreibung`. От операционной системы, если вызов функции был успешен, обратно получаем дескриптор. Это не что иное, как однозначный номер, который назначается операционной системой для связи. Этот номер находится в `lngHandle`, глобальной переменной, которая была объявлена в `Modul1`.

При каждом последующем сообщении используется значение переменной `lngHandle`.

Однако вы можете открыть канал с USB-устройством и при помощи серийного номера, если вторым параметром будет `FT_OPEN_BY_SERIAL_NUMBER`, а первый содержит серийный номер. В качестве альтернативы можно использовать функцию `FT_OPEN (ID, lngHandle)`.

Для освобождения зарезервированных (операционной системой) ресурсов при помощи функции `FT_Close(lngHandle)` связь с устройством завершается:

```
LoggerList.AddItem "Test erfolgreich beendet"
```

Теперь, когда все части первой демонстрационной программы были успешно рассмотрены, в окне вывода появится сообщение: `Test erfolgreich beendet` (тест успешно завершен).

ГЛАВА 6

"Игры" со светом

Как еще можно использовать стандартную конфигурацию микросхемы FT232R, кроме как для последовательного сообщения с другими микроконтроллерами?

Так как последовательные сигналы на соединительном штекере адаптера могут быть опрошены, или же выходы могут быть изменены, то существуют следующие возможности (рис. 6.1):

- Сигналы TxD, RTS и DTR — это выходы, программируемые вызовом функций из библиотеки FTD2XX.DLL. Состояния входов также можно опросить при помощи вызовов функций. Обзор последовательных сигналов интерфейса представлен в табл. 6.1.

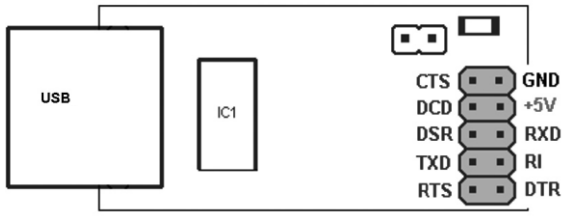


Рис. 6.1

Таблица 6.1

Последовательный сигнал	
TxD	Выход
RxD	Вход
RTS#	Выход
CTS#	Вход
DTR#	Выход
DSR#	Вход
DCD#	Вход
RI	Вход

- Подключив выходные сигналы к ярким светодиодам, к примеру, можно построить собственный USB-осветитель для чтения и управлять им при помощи вашей программы.

6.1. Включение светодиода

Попытаемся включить светодиод LED (англ. Light Emitting Diode) к выходу DTR.

> ***Важно!***
>
> Перед каждым экспериментом отсоединяйте USB-адаптер от порта, чтобы не вызвать короткого замыкания питания USB!

Анод светодиода должен быть соединен с напряжением 5 В USB, а катод светодиода — с выходом DTR. Светодиод горит только тогда, когда используется в прямом направлении, когда на анод подается более положительный потенциал, чем на катод. Рабочее напряжение 3—5-миллиметровых светодиодов лежит в границах от 2 до 3,5 В. Таким образом, светодиод нельзя напрямую подсоединять к напряжению 5 В!

У светодиода вывод анода обычно длиннее, чем катод. Кроме того, анод можно распознать по тому, что он обычно тоньше катода (рис. 6.2).

В данном случае дополнительного гасящего резистора не требуется, поскольку он уже содержится в цепи сигнала DTR на USB-адаптере.

Рис. 6.2

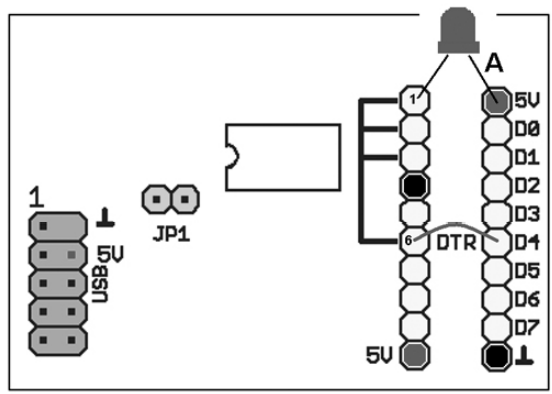


Рис. 6.3

На рис. 6.3 показано, как подключается светодиод к 20-контактной панельке для ИС на дополнительной плате: анод светодиода — к 5 В (вывод 20), а катод — к выводу 1.

Кроме того, для подачи на катод сигнала DTR (вывод 15) нужно вставить проволочную перемычку между выводами 15 и 6.

Подсоедините USB-адаптер вместе с дополнительной платой к вашему компьютеру. Если светодиод подключен правильно, то при инициализации (новой установке драйвера) светодиод многократно вспыхивает. Далее светодиод затухает.

Запустите программу bespiel_2.exe, находящуюся в каталоге демонстрационных программ \Beispielprogramme\Bsp_2. Откроется окно программы. Если вы щелкнете на переключателе **LED AN** (светодиод вкл.), то светодиод должен загореться, а если на переключателе **LED AUS** (светодиод выкл.) — погаснуть (рис. 6.4).

Исходный текст программы для второго примера можно найти в файле Bsp.VBP каталога Bsp_2. Следующие далее подпрограммы вызываются после выбора соответствующего переключателя **LED AN** или **LED AUS**:

```
Private Sub Option1_Click(Index As Integer) 'при LED AN
     LED_Click (False) 'вызов LED_Click с параметром False
End Sub

Private Sub Option2_Click(Index As Integer) 'при LED AUS
     LED_Click (True) 'вызов LED_Click с параметром True
End Sub
```

Собственно выполнение описано в листинге 6.1.

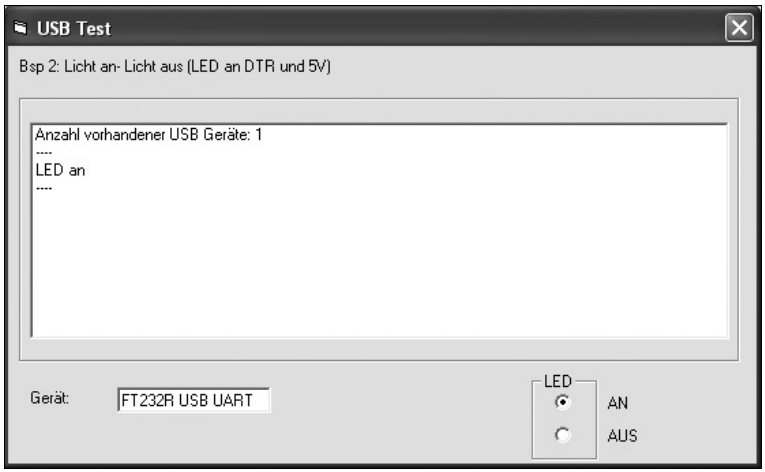


Рис. 6.4

Листинг 6.1

```
Private Function LED_Click(AnAus As Boolean)
On Error GoTo LED_Click_fehler
LoggerList.Clear
If FT_GetNumDevices(lngNumDevices,vbNullString,FT_LIST_
   BY_NUMBER_ONLY)<>FT_OK Then
   LoggerList.AddItem ("Ошибка: FT_GetNumDevices Не функционирует")
   Exit Function
Else
  LoggerList.AddItem ("Количество USB-устройств: " & lngNumDevices)
End If
```

```
strBeschreibung = Trim(Me.DeviceName.Text) & Chr(0)
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle) <>
                                                        FT_OK Then
   LoggerList.AddItem "Ошибка вызова: FT_OpenEx"
   Exit Function
Else
   LoggerList.AddItem "----"
End If
If AnAus Then
   If FT_ClrDtr(lngHandle) = FT_OK Then 'Подать 5 В на DTR => LED выкл
      LoggerList.AddItem "LED выкл"
   End If
Else
   If FT_SetDtr(lngHandle) = FT_OK Then 'Подать 0 В на DTR => LED вкл
      LoggerList.AddItem "LED вкл"
   End If
End If
If FT_Close(lngHandle) <> FT_OK Then
   LoggerList.AddItem " Ошибка вызова: FT_Close"
   Exit Function
 Else
   LoggerList.AddItem "----"
End If
LED_Click_fehler_ende:
Exit Function
LED_Click_fehler:
 MsgBox Err.Description
 Resume LED_Click_fehler_ende
End Function
```

В зависимости от того, какая опция была выбрана, функция `LED_Click` после вызова может иметь значение "Истина" (True) или "Ложь" (False). Полученное значение сохраняется в переменной `AnAus`.

Большинство вызовов должны быть вам уже знакомы из прошлого примера. В отличие от первого примера при открытии канала для связи с устройством используется описание устройства FT232R USB UART непосредственно из поля формы `DeviceName`.

Если переменная `AnAus` имеет состояние True, то происходит вызов функции `FT_ClrDtr (lngHandle)`. При этом на выходе DTR получается уровень напряжения USB — 5 В, так что светодиод гаснет. Не дайте сбить себя с толку названием `Clr` (от англ. Clear): в этом случае напряжение не всегда равно 0 В!

Если переменная `AnAus` имеет состояние False, то вызывается команда `FT_SetDtr (lngHandle)`. При этом к выходу DTR подводится напряжение 0 В. Сейчас через светодиод (и дополнительный резистор в линии DTR на USB-адаптере) проходит ток достаточный для того, чтобы он начал светиться.

6.2. Переключение светодиода

Если в предыдущем примере анод светодиода был подключен к напряжению питания 5 В, а катод — к сигналу DTR, теперь вы можете поменять схему подключения светодиода на инверсную. Для этого подсоедините анод светодиода к сигнальному проводу DTR, а катод к "земле".

Светодиод подсоединяется к 20-контактной панельке для ИС на дополнительной плате (рис. 6.5).

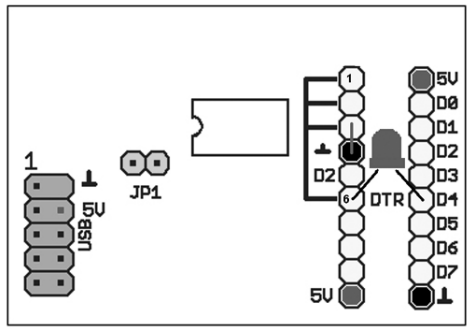


Рис. 6.5

Анод светодиода подводится к сигналу DTR (вывод 15), а катод к выводу 6. Кроме того, необходимо использовать проволочную перемычку между выводами 3 и 4, подсоединив тем самым катод светодиода на "землю" (GND). Только после этого подключите USB-адаптер с ПК.

Если вновь запустить программу beispiel_2.exe, то вы обнаружите обратную картину: при вызове функции `FT_ClrDtr (lngHandle)` сигнал DTR будет соответствовать уровню напряжения в 5 В и поэтому светодиод будет светиться.

6.3. Еще вариант переключения

Сигнал RTS, так же, как и DTR, является сигналом последовательного интерфейса. Для того чтобы включить светодиод с помощью сигнала RTS, нужно лишь вместо функций `FT_ClrDtr` и `FT_SetDtr` вызвать функции `FT_ClrRts` и `FT_SetRts`. При этом, разумеется, не забудьте подсоединить анод светодиода к сигналу RTS (вывод 17 панельки).

6.4. Вспышка светодиода

Третий пример программы — это лишь небольшое развитие первого примера. Оно основывается на длительности кадра USB (1 мс).

Если от прикладной программы к USB-устройству посылается какая-либо команда, то следующая команда может быть передана только через 1 мс (рис. 6.6).

Внутри каждого из миллисекундных временных окон отправляется один из пакетов данных:

- `FtStatus = FT_SetDtr(lngHandle);`
- `FtStatus = FT_SetDtr(lngHandle);`
- `FtStatus = FT_ClrDtr(lngHandle).`

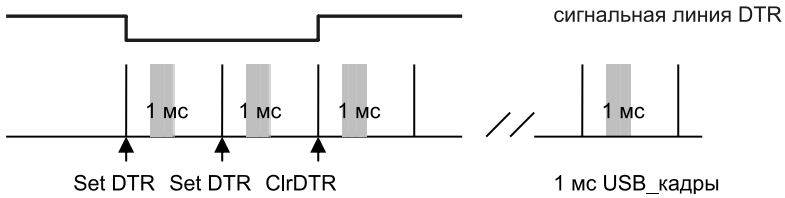


Рис. 6.6

Тип поточной USB-передачи (Bulk-Transfer) не гарантирует, что следом за кадром данных устройства передается кадр данных того же устройства, что порой может привести к продлению длительности свечения подсоединенного светодиода. Подробнее с этим режимом, который играет лишь второстепенную роль, вы познакомитесь в другом примере.

На рис. 6.7 светодиод подключен к 20-контактной панельке для микросхем к сигналу DTR и к 5 В: анод светодиода подсоединен к напряжению 5 В (вывод 20), а катод к выводу 1. Для подключения катода к линии DTR вы должны добавить проволочную перемычку между выводами 15 и 6 панельки.

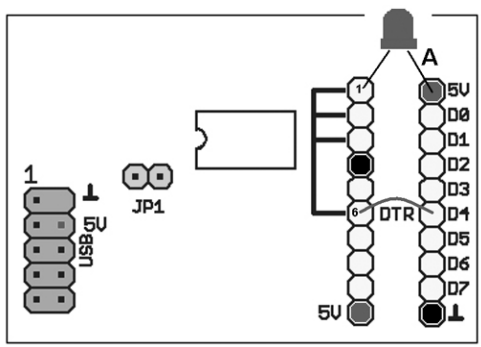


Рис. 6.7

Только после этого подключите USB-адаптер с дополнительной платой, а затем и компьютером.

Запустите программу beispiel_3.exe, расположенную на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_3 (рис. 6.8).

Ползунок, показанный в нижней части диалогового окна предварительно, установлен на 20 мс. Диапазон настройки составляет от 1 до 500 мс. Если нажата кнопка **LED Blitz** (вспышка светодиода), то выполняется 5 циклов (прогонов) с 500-милли-

секундными паузами, каждый раз с одной вспышкой заданной продолжительности свечения. Чем правее ползунок, тем больше будет длительность свечения светодиода. В крайнем правом положении отношение между включенным и выключенным состоянием светодиода будет 1:1 (500 мс светится — 500 мс погашен).

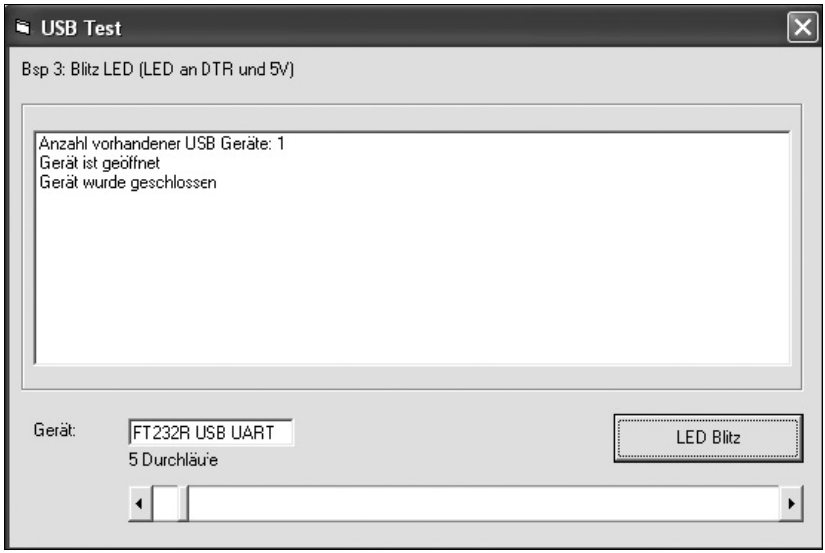


Рис. 6.8

Если вспышки в несколько миллисекунд покажутся вам немного продолжительнее, чем должны быть на самом деле, то причина здесь в инерции человеческого глаза.

Исходный код программы для третьего примера содержится в файле Bsp.VBP, находящемся в каталоге \Beispielprogramme\Bsp_3 прилагаемого компакт-диска.

Следующий фрагмент кода программы отражает процесс ее выполнения:

```
For i = 1 To 5
   For il = 1 To Me.HScroll_PWM.Value
     FtStatus = FT_SetDtr(lngHandle) ' dauert immer 1 Frame = 1ms
   Next il
     FtStatus = FT_ClrDtr(lngHandle) ' LED aus
   Sleep (500) ' 500ms Sekunden warten
Next i
```

Внутри цикла `For` с переменной `i`, который будет выполняться 5 раз, находится второй цикл с переменной `il`. Число прогонов цикла `il` зависит от настроенного значения ползунка. Каждый вызов функции `FT_SetDtr(lngHandle)` происходит в течение 1 мс (1 кадр).

После цикла путем вызова функции `FT_ClrDtr(lngHandle)` выполняется выключение светодиода, а с помощью функции `Sleep (500)` добавляется время ожидания, равное 500 мс.

Можно было бы также применить программный таймер, который бы каждую миллисекунду проверял соответствующие состояния и действовал в соответствии с ними, имея в виду, что этот таймер срабатывает достаточно быстро.

6.5. Управление яркостью светодиода

Четвертый пример основан на предыдущем, однако немного с другой точки зрения.

ШИМ — это широтно-импульсная модуляция (от англ. *PWM* — Pulse Width Modulation). ШИМ называют модуляцией сигнала прямоугольной формы при постоянной частоте (рис. 6.9).

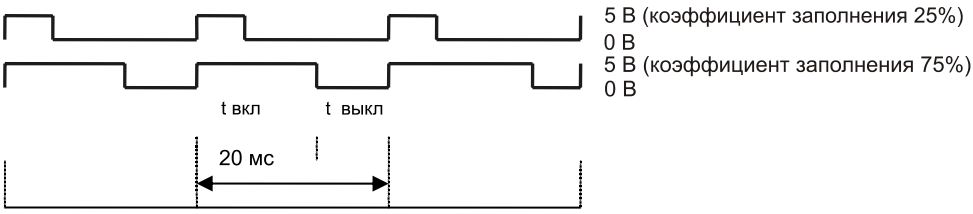


Рис. 6.9

При постоянной частоте (здесь 50 Гц — период 20 мс) изменяется отношение времени включения ко времени выключения и, следовательно, среднее значение постоянного напряжения выходного сигнала. Если время включения $t_{вкл}$ одинаково по продолжительности со временем выключения $t_{выкл}$, то *коэффициент заполнения*[1] равен 50%, а среднее постоянное напряжение составляет ровно половину приложенного напряжения. Если же коэффициент заполнения меньше, как показано на верхней временной диаграмме (см. рис. 6.9), то средняя мощность меньше. Если коэффициент заполнения больше, как это показано на нижней временной диаграмме, то средняя мощность соответственно больше.

Таким образом, с помощью ШИМ можно управлять яркостью светодиода, регулировать частоту вращения двигателя постоянного напряжения или так же смоделировать простой ЦАП.

ШИМ можно легко выполнить программно. Для начала проведем один опыт.

Подключим светодиод к 20-контактной панельке для микросхем (рис. 6.10): анод светодиода подсоединим к 5 В (выводу 20), а катод к выводу 1. Кроме того, для подключения катода к сигналу DTR вы должны вставить проволочную перемычку между выводами 6 и 15.

Запустите программу beispiel_4.exe, находящуюся в каталоге \Beispielprogramme\Bsp_4 на прилагаемом компакт-диске.

---

[1] *Скважность импульсов* — это отношение периода следования импульсов T к длительности импульса $t_и$. Величина, обратная скважности и часто используемая в англоязычной литературе, называется *коэффициентом заполнения импульсов*. *— Ред.*

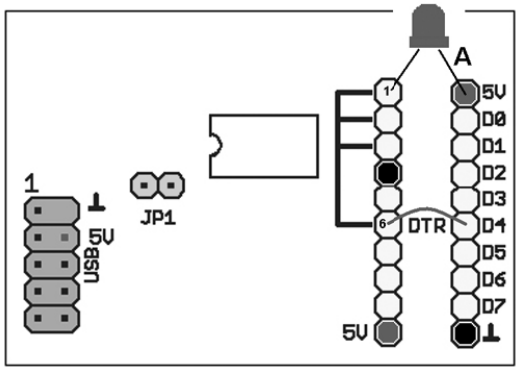


Рис. 6.10

Ползунок, показанный в нижней части диалогового окна (рис. 6.11), предварительно настроен на 1 мс. Диапазон настройки составляет от 1 до 19 мс. Если при этом нажать кнопку **LED PWM**, то светодиод будет светить, но с малой яркостью. Чем дальше вы будете передвигать ползунок вправо, тем ярче будет свечение светодиода. В крайнем правом положении ползунка светодиод будет светиться постоянно. Вы можете заметить легкие колебания яркости: это объясняется тем, что передача данных по USB время от времени работает с перебоями, следовательно, при поточной USB-передаче (в режиме передачи типа Bulk) возникают промежутки между USB-кадрами.

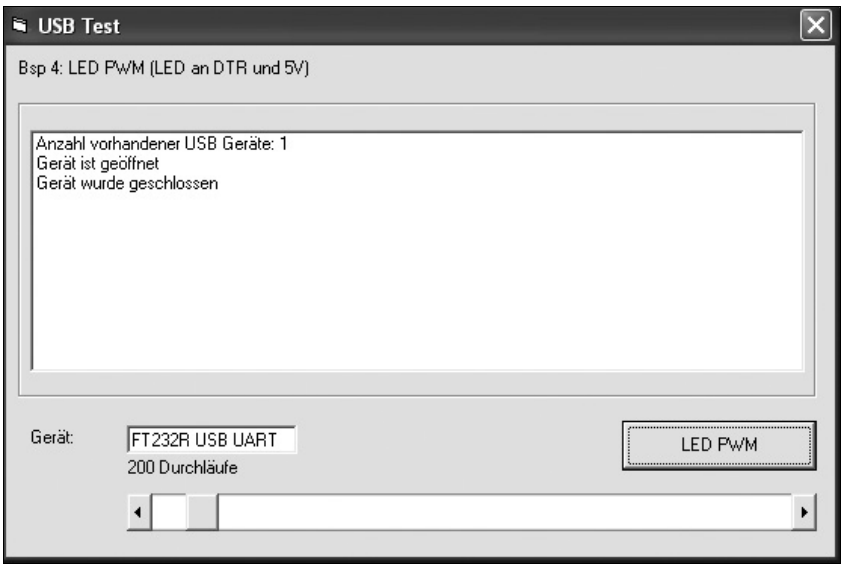


Рис. 6.11

Исходный код для четвертой демонстрационной программы содержится в файле Bsp.VBP, находящимся в каталоге \Beispielprogramme\Bsp_4 на компакт-диске.

Следующий фрагмент кода программы отображает процесс ее выполнения:

```
For i = 1 To 200
   ' максимально 20 мс для одного общего цикла
     ' Цикл 1
   For il = 1 To Me.HScroll_PWM.Value
     FtStatus = FT_SetDtr(lngHandle) ' длительность кадра = 1 мс
   Next il
   ' Цикл 2
   For il = 1 To (Me.HScroll_PWM.Max — Me.HScroll_PWM.Value)
     FtStatus = FT_ClrDtr(lngHandle) ' длительность кадра = 1 мс
   Next il
Next i
```

Внутри цикла `For` с переменной `i`, который будет выполняться 200 раз, находятся два других цикла с переменной `il`. Число прогонов первого цикла `il` зависит от настроенного значения ползунка регулятора. Если значение было настроено на 1, то при подстановке получится:

```
For il = 1 to 1
```

Светодиод включен. В следующем цикле светодиод выключается. Значение `Me.HScroll_PWM.Max` составляет 19. Второй цикл будет выполняться 19 раз, если ползунок настроен на 1. В итоге общий цикл длится 20 мс. Если же подвинуть ползунок вправо, то увеличится число прогонов первого цикла, при этом лампочка будет светить дольше. Число прогонов второго цикла пропорционально сократится. Однако длительность общего цикла остается по-прежнему 20 мс, следовательно, и частота остается постоянной.

6.6. Управление двухцветным светодиодом

Светодиоды различных типов, цветов и мощностей могут иметь интегрированные функции, например, мигающий светодиод или светодиод с интегрированным добавочным резистором. Двухцветные светодиоды имеют два цвета. Такие светодиоды могут быть как с тремя, так и с двумя выводами. Двухцветные светодиоды с тремя выводами (имеющие два кристалла) зачастую имеют один общий вывод катодов. При этом если подать напряжение питания на один анод, то светодиод, к примеру, будет светить красным цветом, а если на другой, то зеленым. Если оба анода подсоединены одновременно, то красный и зеленый цвета смешиваются в оранжевый оттенок.

У двухцветных светодиодов, имеющих только два вывода, светодиоды включаются в корпусе встречно-параллельно. Нужно поменять местами выводы светодиода или напряжение на его выводах, чтобы получить свечение другого цвета.

Поскольку для изменения цвета светодиода нужно постоянно переключать питающее его напряжение (вы же не хотите для получения второго цвета постоянно ме-

нять местами выводы светодиода), можно использовать для эксперимента сигналы DTR и RTS.

Подключение светодиода к панельке на дополнительной плате показано на рис. 6.12. Подсоедините двухцветный двухвыводной светодиод к сигналам RTS и DTR. На этот раз не нужно учитывать полярность светодиода. Соедините USB-адаптер с компьютером.

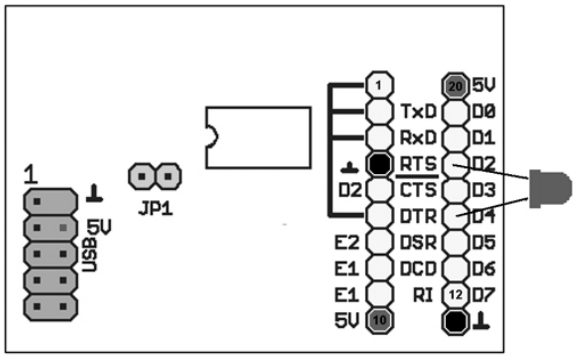


Рис. 6.12

Запустите программу beispiel_5.exe, находящуюся в каталоге \Beispielprogramme\ Bsp_5 на прилагаемом компакт-диске.

Соответствующие сигналы вы можете включить или отключить при помощи флажков DTR или RTS (рис. 6.13). Если оба флажка имеют одинаковое состояние, то светодиод остается темным.

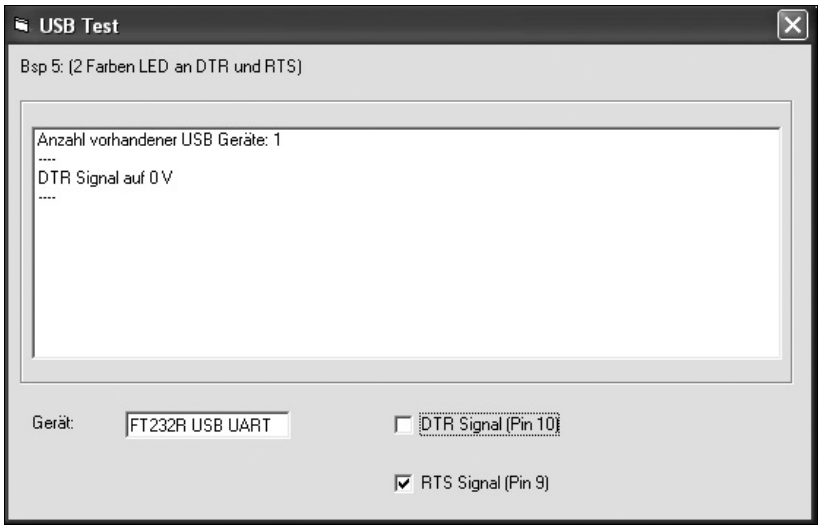


Рис. 6.13

Если же установить только флажок DTR, то светодиод светится одним цветом, а если флажок RTS — то другим цветом.

Исходный код для пятой демонстрационной программы содержится в файле Bsp.VBP в каталоге \Beispielprogramme\Bsp_5.

Следующий фрагмент кода программы (листинг 6.2) отображает процесс ее выполнения.

Листинг 6.2

```
Private Sub CbDtr_Click()
       LEDS_Click Me.CbDtr.Value, True
End Sub

Private Sub CbRts_Click()
       LEDS_Click Me.CbRts.Value, False
End Sub

 Private Function LEDS_Click(AnAus As Boolean, DTRoderRTS As Boolean)
' DTRoderRTS = True DTR или RTS
' ANAUS = True сигнал на 5B
......
......
......
If DTRoderRTS Then
       If AnAus Then
              If FT_ClrDtr(lngHandle) = FT_OK Then
                     LoggerList.AddItem "DTR сигнал на 5B"
              End If
          Else
              If FT_SetDtr(lngHandle) = FT_OK Then
                     LoggerList.AddItem "DTR сигнал на 0B"
              End If
       End If
   Else
       If AnAus Then
              If FT_ClrRts(lngHandle) = FT_OK Then
                     LoggerList.AddItem "RTS сигнал на 5B"
              End If
          Else
              If FT_SetRts(lngHandle) = FT_OK Then
                     LoggerList.AddItem "RTS сигнал на 0B"
              End If
       End If
End If
......
End Function
```

Функция `LEDS_Click (AnAus As Boolean, DTRoderRTS As Boolean)` вызывается с различными параметрами, в зависимости от того, какой флажок выбрал пользователь. При установке флажка DTR активируется функция `FT_ClrDtr(lngHandle)` или `FT_SetDtr(lngHandle)`, а при установке RTS активируется функция `FT_ClrRts(lngHandle)` или `FT_SetRts(lngHandle)`. Если был установлен тот или иной флажок, значение `AnAus` истинно, и соответствующий сигнал устанавливается на 5 В.

В зависимости от состояния сигналов DTR и RTS меняется напряжение, подаваемое на светодиод, который будет светиться соответствующим цветом.

6.7. Мигающее светосигнальное устройство

В пятом примере подключите двухцветный светодиод, так же как в предыдущем примере (рис. 6.14), а затем соедините USB-адаптер с подключенной дополнительной платой и компьютером.

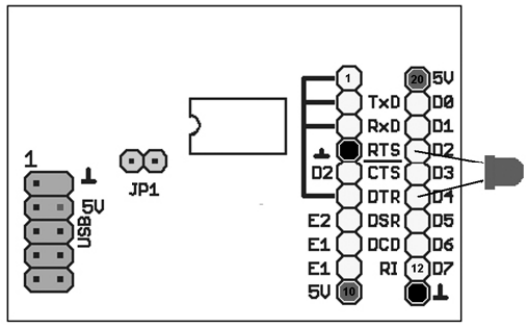


Рис. 6.14

Итак, двухцветный светодиод подсоединен к сигналам RTS или DTR. На этот раз также не нужно учитывать полярность светодиода.

Если после запуска программы, находящейся в каталоге \Beispielprogramme\Bsp_5, нажать кнопку **Wechselblinker** (переменный блинкер), то светодиод 5 раз мигнет, попеременно меняя два цвета. Исходный код этого примера очень прост, и поэтому здесь приведен лишь его небольшой фрагмент:

```
For i% = 1 To 5
       FT_ClrDtr (lngHandle)
       FT_SetRts (lngHandle)
       Sleep 500
       FT_ClrRts (lngHandle)
       FT_SetDtr (lngHandle)
       Sleep 500
Next i
```

Сначала к линии DTR будет подведено напряжение 5 В, а к линии RTS — 0 В. Светодиод в этом случае будет светить одним цветом на протяжении 500 мс. Затем к линии RTS подсоединяется 5 В, а к DTR — 0 В, и светодиод будет светить 500 мс, но уже другим цветом. Этот процесс повторится пять раз.

6.8. Включение выхода TxD

В основной конфигурации микросхемы FT232R выходы представлены не только сигналами DTR и RTS, но и сигналом TxD последовательного интерфейса на рис. 6.15.

В программной документации компании FTDI вы не найдете вызов функции как `FT_SetTxd`, по аналогии с вызовами функций `FT_SetDtr` или `FT_SetRts`, которые приведены в предыдущих примерах. Для управления сигналом TxD вводятся вызовы функций `FT_SetBreakOn`, а также `FT_SetBreakOff`.

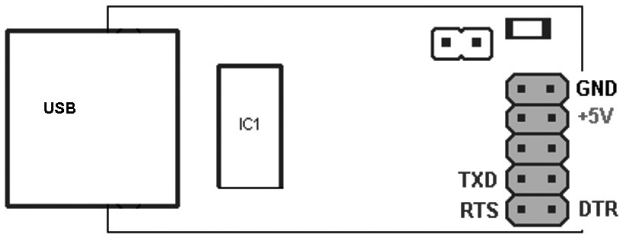


Рис. 6.15

Светодиод подключается к 20-контактной панельке, как показано на рис. 6.16: анод светодиода — к выводу 20 (5 В), а катод — к выводу 19 (TxD).

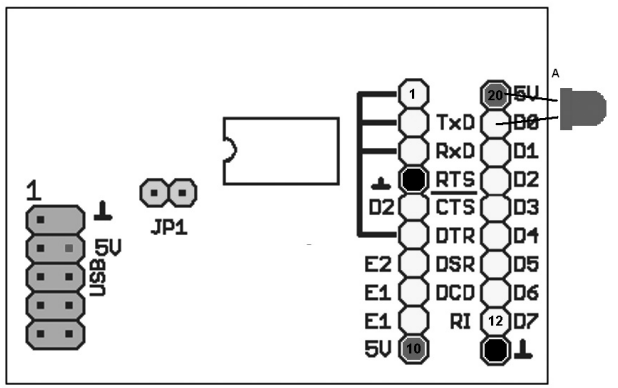


Рис. 6.16

Запустите программу beispiel_6.exe, расположенную на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_6. Если установить флажок **TxD schalten**

(переключить TxD), то светодиод должен загореться, а при сбросе флажка **TxD schalten** он снова должен погаснуть.

Исходный код для шестой демонстрационной программы можно найти в файле Bsp.VBP, который находится в каталоге \Beispielprogramme\Bsp_6.

Следующий фрагмент кода программы отображает вызовы `Break`-функций (прерываний):

```
If Me.CbTxD.Value Then
       If FT_SetBreakOn(lngHandle) = FT_OK Then
              LoggerList.AddItem "Break ON – LED an?"
       End If
   Else
       If FT_SetBreakOff(lngHandle) = FT_OK Then
              LoggerList.AddItem "Break OFF"
       End If
End If
```

От состояния флажка (`CbTxD` — Check Box TxD) зависит, вызывается ли функция `FT_SetBreakOn` или же функция `FT_SetBreakOff`, и, следовательно, включается или отключается светодиод.

6.9. Пример схемы светофора с тремя светодиодами

Предлагается при помощи трех выходных сигналов последовательного интерфейса — DTR, RTS и TxD — создать небольшую схему светофора с тремя разноцветными светодиодами, которые имеют красный, желтый и зеленый цвет. В восьмом примере вы найдете "программный светофор". Немного сноровки — и теперь вам на основании предыдущих примеров удастся создать собственное ПО.

Катод красного светодиода подключен к сигналу TxD на 20-контактной панельке (вывод 19), желтого — к сигналу RTS (вывод 17), а зеленого — к DTR (вывод 15) (рис. 6.17). Все аноды светодиодов подсоединены вместе к USB-напряжению питания, т. е. напряжению +5 В. Кроме того, вы должны вставить проволочную перемычку между выводами 20 и 1.

Присоедините USB-адаптер с подключенной дополнительной платой к ПК и запустите программу. Первым исходным состоянием является свечение красного светодиода светофора: красный светодиод должен быть включен, а все остальные — выключены. Последовательность выполнения программы легче всего представить в виде таблицы (табл. 6.2).

Между отдельными фазами нет времени ожидания, которое может быть получено при помощи уже известной функции `Sleep` или же таймера.

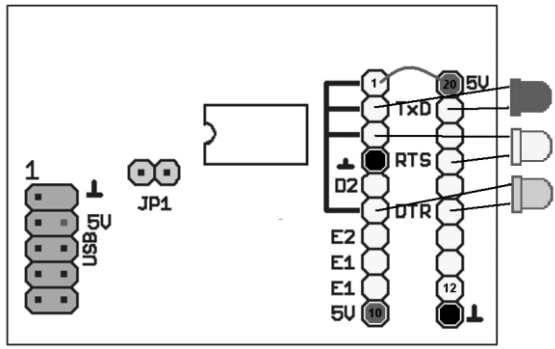


Рис. 6.17

Таблица 6.2

Фаза	Сигнал	Процесс	Вызовы функций
1	● ○ ○	Красный светодиод — включить (желтый и зеленый светодиоды выключены)	`FT_ClrRts` `FT_ClrDtr` `FT_SetBreakOn`
2	● ● ○	Дополнительно включить желтый светодиод	`FT_SetRts`
3	○ ○ ●	Выключить красный и желтый светодиоды и включить зеленый светодиод	`FT_ClrRts` `FT_SetBreakOff` `FT_SetDtr`
4	○ ● ○	Выключить зеленый и включить желтый светодиод	`FT_ClrDtr` `FT_SetRts`
1	● ○ ○	Желтый светодиод выключить, а красный — включить	`FT_ClrRts` `FT_SetBreakOn`

6.10. Пример схемы USB-осветителя для чтения

Точно так же, при помощи трех выходных сигналов DTR, RTS и TxD, можно выполнить USB-осветитель для чтения. Включайте или выключайте светодиоды при помощи разработанных вами программ. В данном случае на компакт-диске нет демонстрационного ПО, но поскольку процедура вызовов функций одинакова с приведенными для схемы светофора, поэтому вы довольно быстро сможете сами их воспроизвести.

Кроме того, вы можете подключить катоды всех светодиодов к "земле" (подсоединив их к штырьковому выводу 2 соединительного разъема), а аноды подсоединить к соответствующим выходным сигналам. Поскольку выходные сигналы USB-

адаптера обладают сопротивлением, то возможно прямое присоединение светодиодов на рис. 6.18.

> ***Совет***
>
> Во многих магазинах радиоэлектронных товаров и магазинах, продающих товары по сниженным ценам, можно приобрести относительно недорогие светодиодные сборки с двумя или тремя светодиодами, которые достаточно просто подключить.

Если вы интересуетесь светодиодами, то далее найдете небольшой пример схемы для измерения яркости при помощи светодиода в качестве элемента устройства слежения за солнцем.

Можно разработать увлекательные схемы, например схему слежения за солнцем (solar tracker), или выполнить достаточно точные измерения яркости. Благодаря двум последовательно подключенным светодиодам можно определить направление источника света.

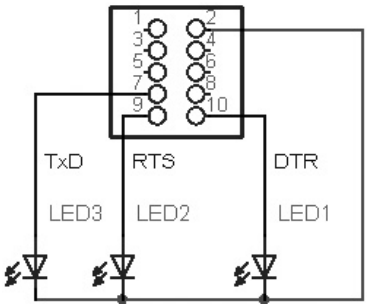


Рис. 6.18

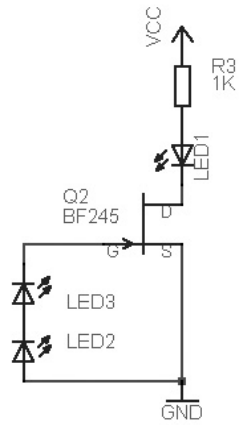


Рис. 6.19

В соединении с микроконтроллером светодиод может использоваться в качестве источника света и одновременно в качестве датчика света.

Если затвор G (англ. *Gate*) полевого транзистора BF245 (рис. 6.19) не присоединен к "земле", то тогда светодиод LED1 будет излучать свет. Чем больше будет напряжение между затвором G и истоком S (англ. *Source*), тем больше полевой транзистор будет переходить в состояние насыщения. Отрицательное напряжение затвор-исток (GS) будет генерироваться под воздействием света светодиодами "солнечных элементов" LED2 и LED3. В зависимости от положения солнца напряжение затвор-исток (GS) будет либо уменьшаться, либо увеличиваться. При этом светодиод LED1 будет либо гаснуть на свету, либо светить в темноте. С помощью конденсатора, подключенного параллельно светодиодам солнечных элементов, можно настроить время задержки включения и выключения индикаторного светодиода.

На самом деле, эта книга с представленной микросхемой FT232R должна была обладать такой возможностью. Однако прямое преобразование (без других элементов) все же потерпит неудачу из-за слишком низкого входного сопротивления сигнальных проводов.

ГЛАВА 7

Опрос входов

Сигналы DCD, DSR, RI и CTS (рис. 7.1) являются цифровыми входами, которые опрашиваются программно благодаря вызову функций библиотеки FTD2XX.DLL.

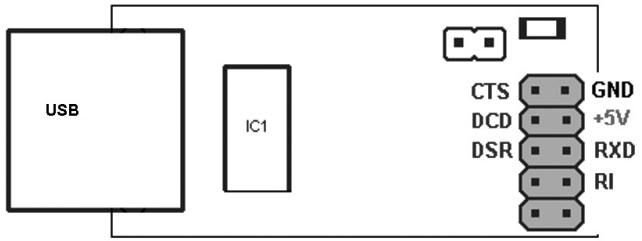


Рис. 7.1

Эти входные сигналы поступают в микросхему FT232R, которая на соответствующих входах имеет внутренние резисторы с сопротивлением 200 кОм, подключенные к напряжению питания USB. Поэтому если эти входы микросхемы никуда не подключены, то на них будет присутствовать напряжение, равное 5 В.

Исходя из этого, напрашивается достаточно простая схема системы аварийной сигнализации. Обычно системы аварийной сигнализации контролируют не замыкание, а размыкание некоторой электрической цепи. Этого же можно достичь и при помощи микросхемы FT232R, используя простую проводную линию, связывающую один из входов с "землей", и таким образом подключая соответствующий вход к напряжению 0 В. Если линия связи размыкается, то на входе микросхемы вновь будет 5 В, и прикладная программа, реагируя на это, вырабатывает сигнал тревоги.

Первый эксперимент связан с опросом входного сигнала CTS. Для этого отсоедините USB-адаптер от вашего компьютера. Линию CTS при помощи соединительного провода подключите к "земле". Этот соединительный провод должен представлять собой замкнутый переключатель, т. е. электрическую цепочку, соединяющую сигнальную линию с "землей".

На 20-контактной панельке дополнительной платы установите проволочную перемычку между выводами 4 и 16, подключающую сигнал CTS к "земле" (GND) (рис. 7.2).

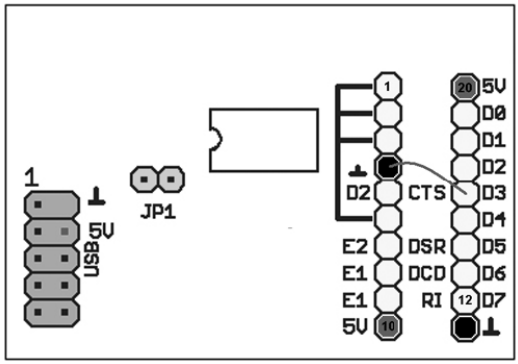


Рис. 7.2

После этого подключите USB-адаптер с подсоединенной дополнительной платой к ПК. Затем запустите программу beispiel_7.exe, находящуюся на компакт-диске в каталоге \Beispielprogramme\Bsp_7.

Если вы нажмете кнопку **Signale abfragen** (опросить сигналы) (рис. 7.3), то должен установиться только флажок **Alarm Leitung 1 - CTS** (линия сигнала 1). Другие флажки остаются сброшенными, т. к. на остальных линиях имеется напряжение 5 В. Если же вы удалите проволочную перемычку и вновь нажмете кнопку **Signale abfragen** (опросить сигналы), то флажок для сигнала CTS снова сбросится, что равнозначно срабатыванию сигнала тревоги.

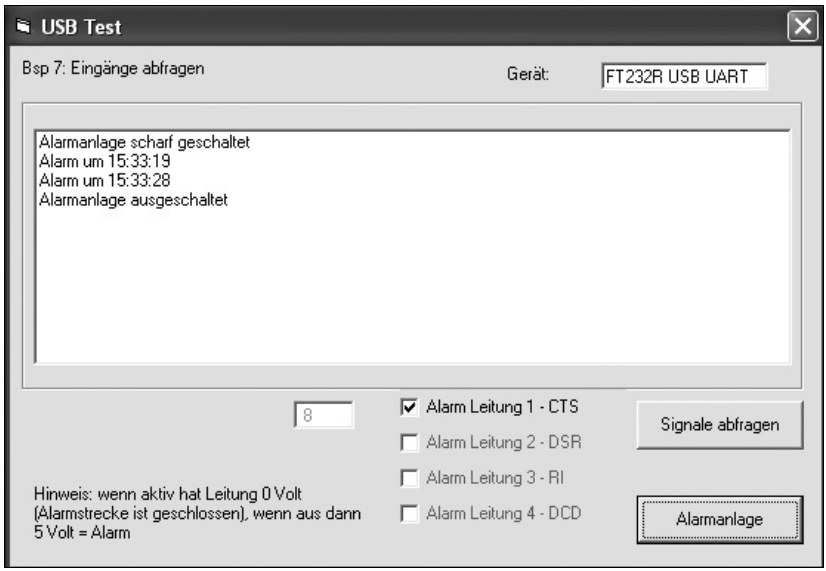


Рис. 7.3

При помощи других входных сигналов DSR, DCD и RI вы можете проводить свои дальнейшие эксперименты. Однако при этом будьте осторожны: не сделайте короткое замыкание напряжения питания USB (вывод 10 панельки) на землю!

Исходный код седьмой демонстрационной программы находится в файле Bsp.VBP каталога \Beispielprogramme\Bsp_7. Следующая подпрограмма вызывается в том случае, когда нажимается кнопка **Signale abfragen** в VB-форме (листинг 7.1):

Листинг 7.1

```
Private Sub bt_Eingang_abfragen_Click()
..
IF FT_GetModemStatus(lngHandle, ModemStatus) <> FT_OK Then
       LoggerList.AddItem "ошибка вызова: FT_GetModemStatus"
     Else
       Me.Cb_cts.Value = (ModemStatus And &H10) / &H10
       Me.cb_dsr.Value = (ModemStatus And &H20) / &H20
       Me.cb_ri.Value = (ModemStatus And &H40) / &H40
       Me.Cb_dcd.Value = (ModemStatus And &H80) / &H80
End If
..
End Sub
```

Вызов функции `FT_GetModemStatus` показывает состояние соответствующих сигнальных проводов в возвращенном значении `ModemStatus` в отдельных битах. Они должны быть опрошены по одному и далее обработаны (рис. 7.4).

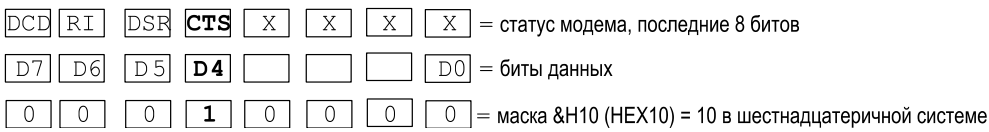


Рис. 7.4

В качестве примера мы рассмотрим обработку сигнала CTS немного подробнее. CTS управляется битом данных D4. При помощи `(ModemStatus And &H10)` бит данных D4 в `ModemStatus` посредством логической операции И логически связывается с маской `&H10`. При логической операции И, чтобы результат был равен 1, оба бита должны иметь значение 1. Если же CTS равняется 0, то результат также остается равен 0, при этом флажок **Alarm Leitung 1 - CTS** (`Cb_cts`) будет сброшен (рис. 7.5).

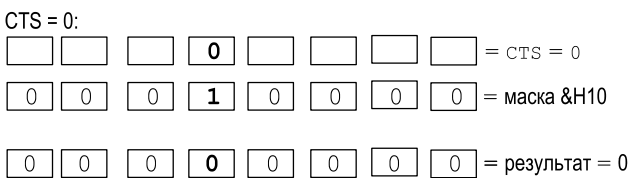


Рис. 7.5

Только когда сигнал CTS имеет значение 1, в качестве результата также получают 1. Результатом из `(ModemStatus And & H10)` было бы в данном случае число 16

(10 в шестнадцатеричной системе). Для того чтобы флажок `Cb_cts` мог быть использован так же, как и значение 1, нужно выполнить его выделение с помощью маски `&H10` (рис. 7.6).

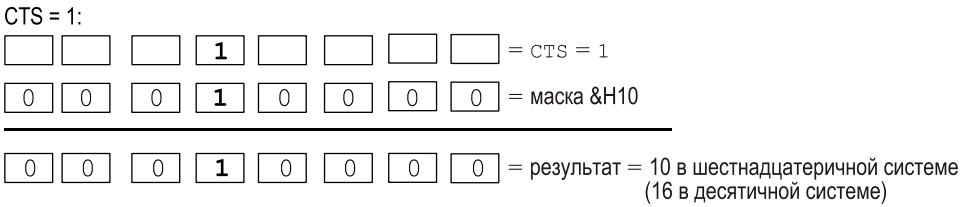


Рис. 7.6

Другие сигналы DCD, RI и DSR в соответствии с положением бита в `ModemStatus` маскируются соответственно `&H80`, `&H40` и `&H20`.

7.1. Система сигнализации

По щелчку по кнопке **Alarmanlage** (сигнал тревоги) входные сигналы будут периодически опрашиваться. Если сигнальные линии CTS, DCD, RI или DSR соединены с "землей", то соответствующая электрическая цепь будет замкнута, а флажок установлен. При размыкании цепи обозначение соответствующего флажка отображается красным цветом, а сам флажок при этом будет сброшен. Сигнальные линии опрашиваются через 500 мс таймером 1 (Timer1), который запускается после нажатия кнопки **Alarmanlage**. Фрагмент исходного кода примера 7 приведен в листинге 7.2.

Листинг 7.2

```
Private Sub Timer1_Timer()
..
If FT_GetModemStatus(lngHandle, ModemStatus) <> FT_OK Then
        LoggerList.AddItem "Ошибка вызова: FT_GetModemStatus"
   Else
        Me.Cb_cts.Value = (ModemStatus And &H10) / &H10
               Me.Cb_cts.ForeColor = 0
               ' по тревоге красный
               If Me.Cb_cts.Value = False Then Cb_cts.ForeColor = 255
        Me.cb_dsr.Value = (ModemStatus And &H20) / &H20
               Me.cb_dsr.ForeColor = 0
               If Me.cb_dsr.Value = False Then cb_dsr.ForeColor = 255
        Me.cb_ri.Value = (ModemStatus And &H40) / &H40
               Me.cb_ri.ForeColor = 0
               If Me.cb_ri.Value = False Then cb_ri.ForeColor = 255
```

```
       Me.Cb_dcd.Value = (ModemStatus And &H80) / &H80
              Me.Cb_dcd.ForeColor = 0
              If Me.Cb_dcd.Value = False Then Cb_dcd.ForeColor = 255
End If
.. End Sub
```

Цикл опроса линий осуществляется с помощью таймера 1 (Timer1) каждые 500 мс. Чтобы можно было воспроизвести состояние системы сигнализации в цвете, анализируются соответствующие уровни сигнальных линий, и в случае тревожной ситуации в линии меняется цвет отображения для нее с черного (`0`) на красный (`255`).

Таймер 1 (Timer1) запускается или останавливается при нажатии кнопки **Alarmanlage** (листинг 7.3).

Листинг 7.3

```
Private Sub bt_alarm_Click()
.
If TimerIstAn Then
       Me.bt_alarm.Caption = "Alarmanlage"
       Me.Timer1.Interval = 0
       TimerIstAn = False
       LoggerList.AddItem ("Alarmanlage ausgeschaltet")
       Me.lb_anzeige.Visible = False
       Me.lb_impulse.Visible = False
   Else
       Me.bt_alarm.Caption = "Alarm aus"
       Me.Timer1.Interval = 500
       TimerIstAn = True
       LoggerList.Clear
       LoggerList.AddItem ("Alarmanlage scharf geschaltet")
       Me.lb_anzeige.Visible = True
       Me.lb_impulse.Visible = True
End If
.
End Sub
```

При первом запуске программы логическая переменная `TimerIstAn` принимает значение True, а интервал времени для таймера равен 500 мс. Одновременно с этим высвечивается другая дополнительная информация для пользователя (`lb_anzeige` и `lb_imbulse`). После запуска кнопка переименовывается в **Alarm aus** (сигнал тревоги выключить).

Новое нажатие кнопки останавливает таймер и обнуляет переменную `TimerIstAn`.

7.2. Счетчик сигналов тревоги

При каждом сигнале тревоги на линии CTS значение счетчика тревожных ситуаций увеличивается на 1. Для этого предыдущее состояние сигнальной линии сохраняется в логической переменной `bcts_vorher` и сравнивается с текущим состоянием сигнальной линии CTS (флажок `Cb_cts`). Это сравнение требуется тогда, когда должно быть подсчитано число срабатываний сигнала тревоги и, как в данном случае, должно быть задокументировано время срабатывания. Соответствующий фрагмент исходного кода примера 7 приведен в листинге 7.4.

Листинг 7.4

```
Private Sub Timer1_Timer()
..
. Me.Cb_cts.Value = (ModemStatus And &H10) / &H10
Me.Cb_cts.ForeColor = 0
'по тревоге поменять на красный
If Me.Cb_cts.Value = False Then Cb_cts.ForeColor = 255
'подсчет сигналов тревоги только при изменении состояния
If bcts_vorher = True And Me.Cb_cts.Value = False Then
Me.tb_impulse.Text = Str(Val(Me.tb_impulse.Text) + 1)
Me.LoggerList.AddItem "Alarm um " & Time()
End If
bcts_vorher = Me.Cb_cts.Value
Me.cb_dsr.Value = (ModemStatus And &H20) / &H20
Me.cb_dsr.ForeColor = 0
If Me.cb_dsr.Value = False Then cb_dsr.ForeColor = 255
..
End Sub
```

Каждый раз, когда вы удаляете проволочную перемычку сигнальной линии CTS, замечается изменение состояния и документируется время изменения. Если же теперь вы вернете проволочную перемычку на прежнее место, то ничего это не изменит.

7.3. Пример схемы системы охранной сигнализации

На практике в линиях в качестве датчиков тревоги используются *герконы* (герметизированные магнитоуправляемые контакты). В магнитном поле контакты геркона замкнуты. Если же магнитное поле удаляется, то контакты геркона размыкаются. Постоянный магнит целесообразно крепить к подвижному элементу окон и дверей (рис. 7.7).

Геркон должен быть установлен, конечно же, в непосредственной близости от постоянного магнита, чтобы электрическая цепь при закрытом окне была замкнута.

Постоянные магниты и герконы представлены на рис. 7.8 (геркон и магниты), рис. 7.9 (геркон в непрозрачном корпусе для установки) и рис. 7.10 (геркон в стеклянном корпусе).

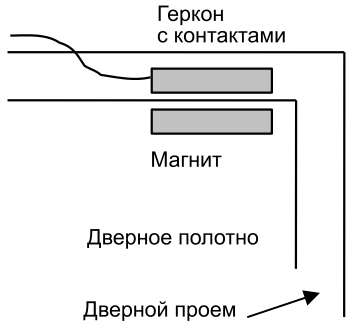


Рис. 7.7

Рис. 7.8

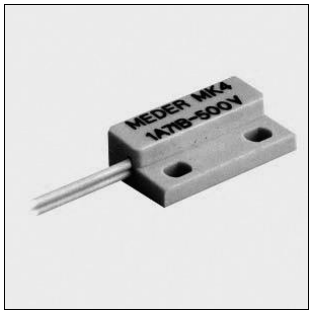


Рис. 7.9

Рис. 7.10

Совет

Во многих магазинах, торгующих по сниженным ценам, для окон и дверей предлагаются довольно недорогие мини-системы охранной сигнализации, которые зачастую можно легко приспособить для данной цели.

Примерно так могла бы выглядеть электрическая схема вашей самостоятельно собранной системы сигнализации для вашей компьютерной комнаты (рис. 7.11):

- 4 геркона для двери, окна, выдвижного ящика стола, шкафа с компакт-дисками или замка-выключателя, который активирует сигнализацию;
- 2 реле (5 В, 20 мА), например для передачи сигнала тревоги по телефонной линии, управления дверью вашего шкафа с компакт-дисками или звукового сигнала тревоги;
- яркий светодиод, для того чтобы взломщик не остался незамеченным в темноте.

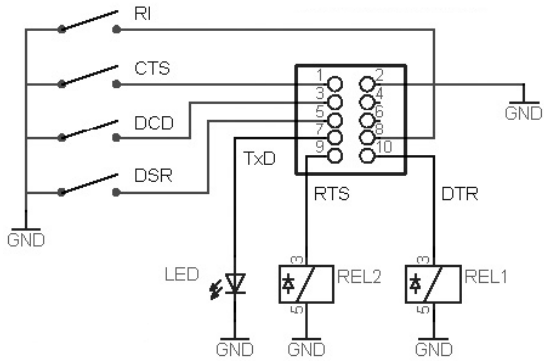


Рис. 7.11

Собранная вашими руками охранная сигнализация вашей компьютерной комнаты при использовании дополнительной платы могла бы выглядеть так, как это показано на рис. 7.12.

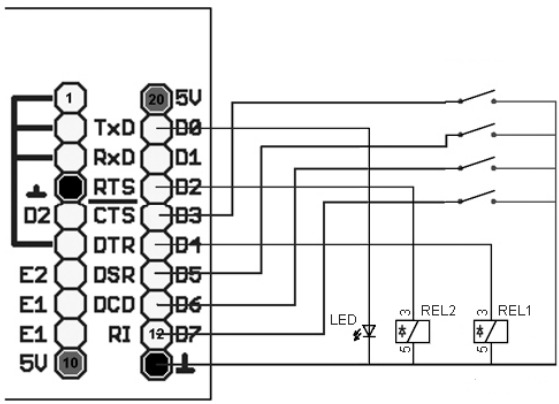


Рис. 7.12

Уже при помощи первых рассмотренных в этой книге примеров вы можете самостоятельно создавать программы для обработки, оценки и выполнения соответствующих действий в случае тревожной ситуации.

7.4. Здесь ли кошка?

Что вы думаете об электронной проверке откидывающейся кошачьей дверцы, установленной на входной двери квартиры, которая позволит узнать привычки вашей кошки?

Для данного эксперимента вам понадобится по одному геркону как внутри, так и снаружи откидной дверцы. Контакты герконов изначально должны быть разомкнуты и замыкаться лишь тогда, когда откидная дверца будет двигаться. Для ваших экспериментов вы можете использовать пример 12 (периодическое чтение входов в

режиме Bit Bang). Если же вы хотите немного поэкспериментировать, то можете сами внести необходимые изменения в имеющийся пример.

ВНИМАНИЕ!

Если вы работаете с реле, у которого напряжение на выходных контактах более 24 В, то в случае ошибки речь может пойти об опасности для жизни или опасности возникновения пожара!

7.5. Осторожно, вода

При цифровой обработке логических 0 или 1 сигналы на входе не всегда имеют уровни напряжений, равные 0 или 5 В. Имеются так называемые пороговые напряжения, которые входным сигналом могут быть превышены или же не превышены. Это пороговое напряжение для логической 1 для микросхемы FT232R при напряжении ее питания 5 В лежит между 1,3 и 1,9 В, при стандартном значении 1,6 В. Таким образом, если входное напряжение будет соответствовать этой пороговой области, то сигнал может быть интерпретирован в качестве логического 0 или же 1. При входном напряжении ниже 1,3 В гарантированно обнаруживается низкий логический уровень (LOW), а при напряжении выше 1,9 В гарантированно обнаруживается высокий логический уровень (HIGH).

На цифровых выходах высокий логический уровень микросхемы FT232R имеет стандартное напряжение 4,1 В (3,2—4,9 В, при токе потребления 6 мА), а низкий уровень обычно составляет около 0,4 В (0,3—0,6 В).

Если на входе, как в примере с системой сигнализации, переключатель подключен к "земле", то входное напряжение точно будет лежать ниже 0,3 В. Так как входы микросхемы FT232R имеют внутренние резисторы с сопротивлением 200 кОм, подключенные к напряжению питания (рис. 7.13), то можно определить сопротивление, чтобы на входе был обнаружен низкий уровень.

$$I = U / R$$

$$I = U_{Rx} / Rx = U_{Ri} / Ri$$

$$Rx = (U_{Rx} / U_R) \cdot Ri = (1,3 / 3,7) \cdot 200 \text{ кОм} = 70,27 \text{ кОм}$$

$$Rx <= 70,2 \text{ кОм}$$

Таким образом, сопротивление на входе должно быть меньше 70 кОм, при пренебрежении других факторов, чтобы микросхемой FT232R был распознан низкий логический уровень. На практике обнаруживается, что сопротивление около 65 кОм создает низкий уровень.

Для проверки системы сигнализации подходит программное обеспечение примера 7.

Присоедините USB-адаптер с подключенной дополнительной платой к ПК (рис. 7.14).

После запуска программы beispiel_7.exe из каталога \Beispielprogramme\Bsp_7 щелкните кнопку **Alarmanlage**. Когда вы накоротко замкнете концы проводов, то

вы увидите, что в программе будет установлен флажок **Alarm Leitung 1 – CTS**, соответствующий нулевому логическому уровню сигнала CTS, т. е. активному его значению.

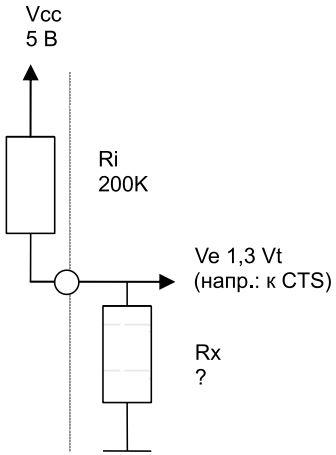


Рис. 7.13

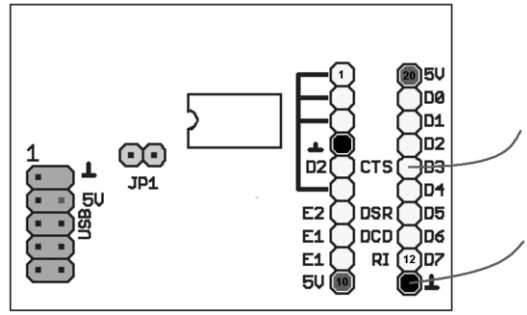


Рис. 7.14

Если вы подержите оба изолированных конца проводов на расстоянии нескольких миллиметров между большим и указательным пальцами, то CTS-сигнал все еще будет активным. Сопротивление в этом случае будет менее 70 кОм. При необходимости, если система не будет функционировать, увлажните пальцы.

Хотя из медицинских измерений известно, что значение сопротивления поверхностного слоя кожи человека составляет от 600 кОм до 1 МОм, в данном случае относительно низкое значение сопротивления будет объясняться только влажностью кожи.

Как известно, вода тоже проводит: если вы погрузите два изолированных конца провода в наполненный стакан, то сигнал CTS все равно останется активным. Теперь вы знаете, как при помощи вашего USB-компьютера можно наблюдать за состоянием воды в вашем аквариуме, бассейне или стиральной машине!

7.6. Светло или темно?

Фоторезистор (англ. *LDR — Light Dependent Resistor*) — это полупроводниковый светочувствительный элемент. Фоторезисторы, например, используются в автоматах уличного освещения в качестве датчиков света. Чем больше света падает на светочувствительный фоторезистор, тем меньше будет его сопротивление и тем больше будет электрический ток при неизменном приложенном к нему напряжении. В темноте фоторезистор обладает достаточно высоким сопротивлением в несколько мегаом, а на свету его сопротивление составляет несколько килоом. Это изменение сопротивления вы можете использовать, как и в предыдущем примере.

Подключите светочувствительный фоторезистор к 20-контактной панельке дополнительной платы между выводами 16 и 11 (рис. 7.15).

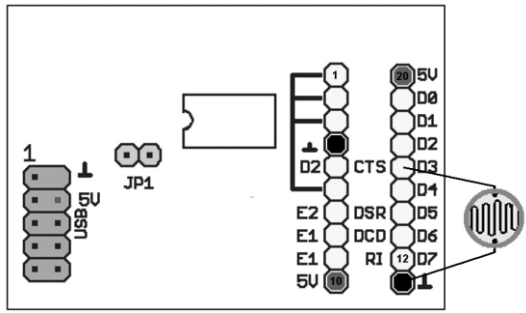


Рис. 7.15

Подключите USB-адаптер с подсоединенной дополнительной платой к ПК.

Запустите программу beispiel_7.exe, находящуюся в каталоге \Beispielprogramme\ Bsp_7 на прилагаемом компакт-диске, и затем нажмите кнопку **Alarmanlage**. Как увидите, при достаточном освещении в программе будет установлен флажок **Alarm Leitung 1 - CTS**, соответствующий нулевому логическому значению сигнала CTS, т. е. активному его значению, поскольку сопротивление фоторезистора в этом случае будет менее 70 кОм. Если вы в достаточной степени затемните фоторезистор, флажок будет сброшен, т. к. при уменьшении света увеличивается значение сопротивления светочувствительного элемента.

Если вы хотите уменьшить светочувствительность, то перед фоторезистором расположите картонку с отверстиями, размеры которых при необходимости можно изменять. Применив фоторезистор в схеме вместе с USB-лампой для чтения *(см. главу 6)*, появляется возможность управлять светодиодами лампы автоматически.

7.7. Применение оптического фотоэлемента в аварийной сигнализации

Каждый геркон, используемый в аварийной сигнализации, рассмотренной в *разд. 7.1*, вы также можете заменить фоторезистором или фотодиодом. При этом

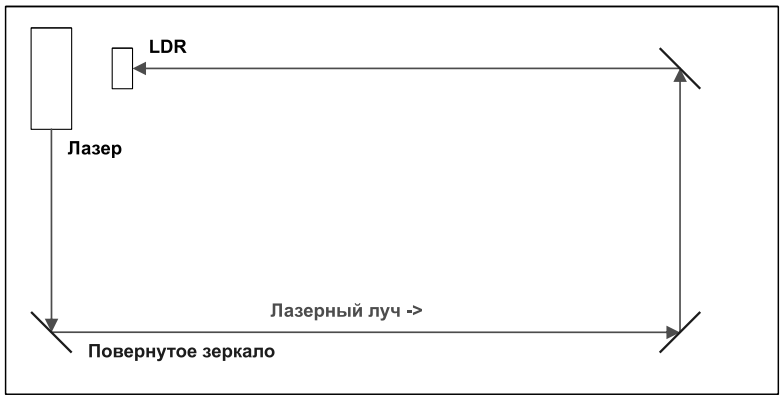


Рис. 7.16

фотоэлемент можно осветить светодиодом или недорогой лазерной указкой (рис. 7.16).

Пока луч света не будет прерван, фотоэлемент имеет незначительное сопротивление. Если же луч света прерывается (или лазерная указка выходит из строя), вы получите аварийную ситуацию.

7.8. Более точное определение сопротивления фоторезистора

Для автомата уличного освещения или фотореле может быть использован предыдущий пример. Однако что нужно сделать, если в наступающей темноте хочется включить первую лампу, а при еще большем затемнении и вторую лампу? Для такого подключения потребуется лучшая обработка изменений сопротивления светочувствительного фоторезистора. Можно было бы, к примеру, использовать аналого-цифровой преобразователь, который из соответствующего аналогового значения яркости генерировал бы эквивалентное цифровое значение. Но зачем же сразу стрелять из пушек по воробьям? Существуют и более простые решения, о которых речь пойдет далее.

При подключении конденсатора C к напряжению питания V через резистор R напряжение на конденсаторе будет экспоненциально повышаться от некоторого начального до напряжения питания (рис. 7.17).

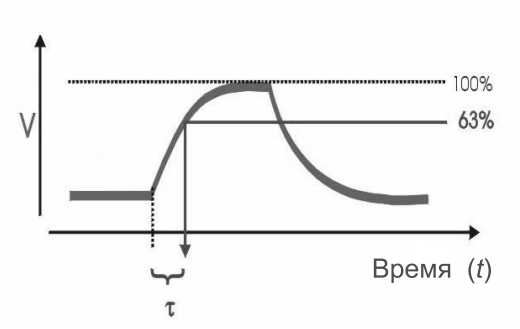


Рис. 7.17

Время заряда зависит лишь от величины емкости конденсатора C и сопротивления резистора R. Время заряда будет тем больше, чем больше емкость конденсатора и сопротивление резистора. Произведение емкости конденсатора C и сопротивления резистора R определяется как *постоянная времени* τ.

За время постоянной времени τ напряжение каждый раз увеличивается в 0,63 раза относительно разности между конечным и начальным значением напряжения. Так после первого отрезка времени, равного τ, напряжение на конденсаторе составляет 63,2% от напряжения питания, после 2τ — 86%, после 5τ — на конденсаторе будет практически полное напряжение питания.

Кривая напряжения V_C на конденсаторе соответствует уравнению:

$$V_C = V \cdot (1 - e^{-(t/RC)}).$$

V — это входное напряжение (напряжение питания) в вольтах, t — время в секундах, R — сопротивление резистора в омах, C — емкость конденсатора в фарадах. Проще говоря: когда подводишь напряжение через резистор к конденсатору, то конденсатор заряжается дольше при больших значениях сопротивления резистора и быстрее при малых значениях. Для лучшего понимания вы можете запустить на выполнение программу RC_E_Funktion.exe, находящуюся в каталоге \Beispielprogramme\e-funktion на прилагаемом к книге компакт-диске. Это, по сути, "интерактивная временная диаграмма", разработанная профессором Рюдигером Хартвигом, которую он любезно предоставил в качестве небольшой забавной игры. Здесь речь идет о наглядности описанной до этого функции V(t) с заданными значениями R и C. Значения сопротивления R и емкости C вы должны предварительно ввести в оба соответствующие поля, показанные в левой части диалогового окна программы (рис. 7.18). После ввода значений щелкните кнопку **Neuzeich** (перерисовка), и в правой части диалогового окна будет изображена соответствующая кривая функции (временная диаграмма). При помощи курсора мыши вы можете в желтых полях выбрать значения для времени T или напряжения V.

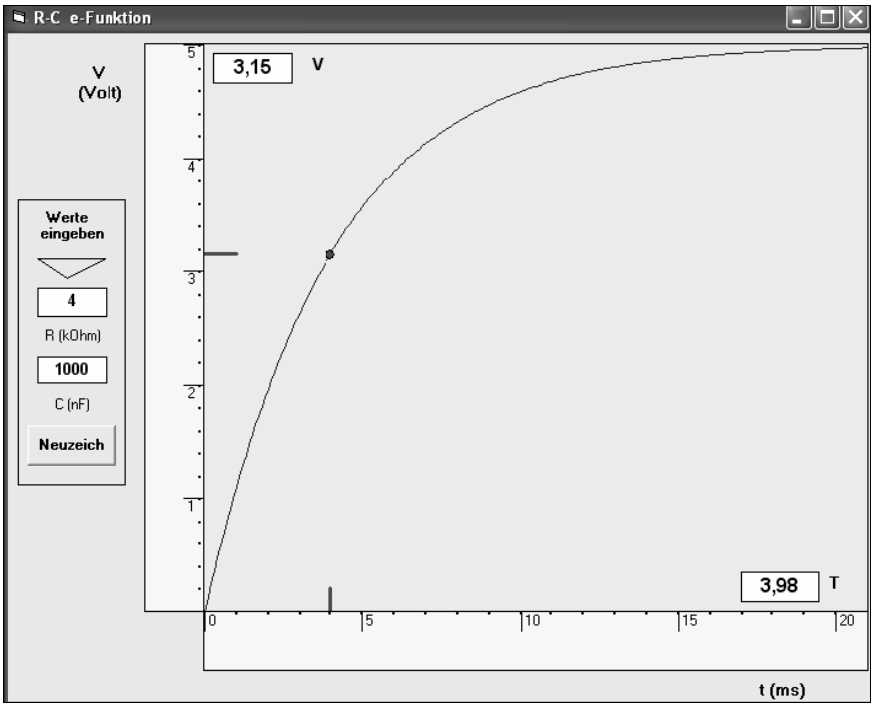


Рис. 7.18

Значения рассчитываются программно и представляются в небольших полях рядом с соответствующей осью диаграммы. Таким образом можно практически мгновенно определить, сколько миллисекунд пройдет до того момента, как — при задан-

ных значениях R и C — напряжение достигнет определенной величины. Путем экспериментирования с различными заданными величинами R или C можно быстро получить представление о их влиянии на временную диаграмму.

В нашем эксперименте вместо резистора будем использовать фоторезистор (рис. 7.19). В этом случае при свете через фоторезистор LDR конденсатор C будет заряжаться относительно быстро, а в темноте — медленно. Для включения процесса заряда требуется выходной сигнал USB-адаптера. Для этого используется сигнал TxD. Измерение времени происходит при помощи входного сигнала CTS. Если конденсатор C1 (100 мкФ) на выходе TxD, который напрямую соединен с входом CTS, в исходном состоянии не заряжен, то сигнал CTS имеет низкий логический уровень (LOW).

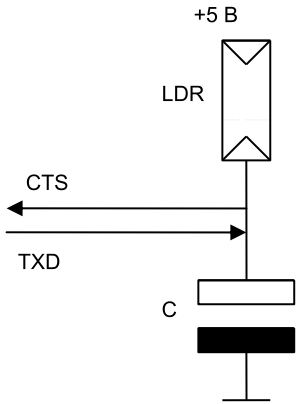


Рис. 7.19

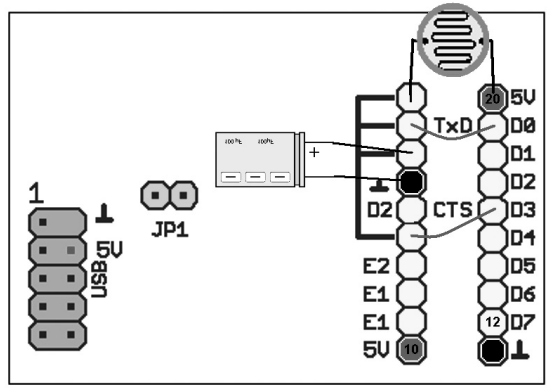


Рис. 7.20

Если напряжение на конденсаторе в какой-нибудь момент времени превысит порог переключения на входе CTS, то сигнал CTS поменяет свое состояние с низкого логического уровня (LOW) на высокий (HIGH).

В качестве величины при определении мгновенной освещенности в данном случае фактически необходимо измерять время.

> ***Примечание***
>
> Надо учитывать, что конденсатор заряжается не только через светочувствительный фоторезистор LDR, но и через параллельно включенный с ним 200-килоомный внутренний резистор вывода CTS и TxD.

Процесс выполнения программы очень прост. Заряд конденсатора и измерение времени запускаются циклически при помощи таймера.

Подключите фоторезистор к выводам 1 и 20 панельки для микросхем, установленной на дополнительной плате (рис. 7.20). Отрицательный вывод электролитический конденсатора емкостью 100 мкФ нужно подключить к выводу 4, а положительный к выводу 3 панельки. Кроме того, необходимо установить две проволочные перемычки для сигналов CTS и TXD между выводами 2 и 19, а также 6 и 16. После этого подключите USB-адаптер с присоединенной дополнительной платой к ПК.

Запустите программу beispiel_10.exe, находящуюся в каталоге \Beispielprogramme \Bsp_10 прилагаемого к книге компакт-диска. После нажатия кнопки **Start** (рис. 7.21) начинается циклическое измерение в интервале времени 500 мс. Нажатием кнопки **Stop** измерение можно завершить. Если во время измерения вы немного затемните рукой фоторезистор, то ползунок, расположенный в нижней части диалогового окна программы, сдвинется вправо, а при увеличении освещенности фоторезистора — влево.

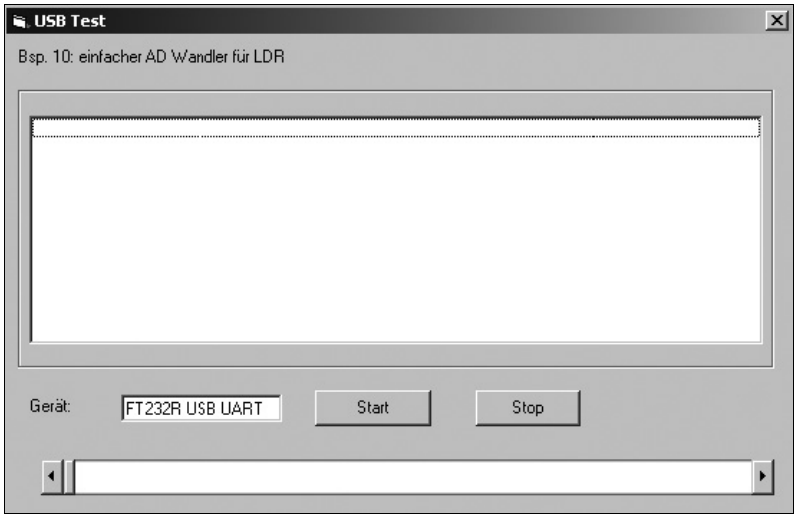


Рис. 7.21

Исходный код для десятого примера находится в файле Bsp.VBP в каталоге \Beispielprogramme\Bsp_10.

Подпрограмма (представлена в листинге 7.5) вызывается в том случае, если нажимается кнопка **Test** в форме на VB.

Листинг 7.5

```
Private Sub bt_test_Click()
       Me.Timer1.Interval = 1000
End Sub
```

Каждую секунду осуществляется опрос подпрограммы таймера 1 (Timer1) (листинг 7.6).

Листинг 7.6

```
Private Sub Timer1_Timer()
On Error GoTo Timer1_fehler
CtsMask = &H10
MeinZaehler = 0
..
```

```
FT_SetBreakOff lngHandle ' зарядить электролитический конденсатор
WartenAufCts:
       FtStatus = FT_GetModemStatus(lngHandle, ModemStatus)
        If (ModemStatus And CtsMask) Then ' то единица ?
              MeinZaehler = MeinZaehler + 1
              GoTo WartenAufCts
        End If
   If MeinZaehler > Me.HScroll1.Max Then MeinZaehler = Me.HScroll1.Max
   Me.HScroll1.Value = MeinZaehler
FT_SetBreakOn lngHandle ' электролитический конденсатор разрядить
Sleep 20
..
End Sub
```

Путем вызова функции `FT_SetBreakOff` уровень выходного сигнала TxD становится равным напряжению 5 В, и в результате начинается процесс заряда электролитического конденсатора емкостью 100 мкФ. Затем счетчик `MeinZaehler` увеличивается на 1 так долго, пока вызов функции `FT_GetModemStatus(lngHandle, ModemStatus)` не возвращает высокое логическое состояние входного сигнала CTS.

Показания счетчика передаются для визуализации в виде положения ползунка `HScroll1`. Вызов функции `FT_SetBreakOn` разряжает конденсатор для следующего измерения. Иногда ползунок резко двигается влево: налицо замедление при передаче данных USB. USB-передача приостанавливается, конденсатор заряжается дальше, а счетчик не считает заряд высоким, т. к. замедляется вызов функции `FT_GetModemStatus`.

Поточные передачи данных (Bulk-Transfer) микросхемы FT232R не задуманы для критических во времени передач. Однако этот режим работы можно ускорить, если сократить периодические опросы, например, с 1000 до 200 мс.

В программе это происходит или благодаря образованию среднего значения, или, как в данном случае, благодаря простому сравнению (листинг 7.7).

Листинг 7.7

```
If ((MeinZaehler < Me.HScroll1.Value · 0.9) And _
                                          (bNichtgezählt = False)) Then
       bNichtgezählt = True
    Else
       Me.HScroll1.Value = MeinZaehler
       bNichtgezählt = False
End If
```

ГЛАВА 8

Управление кварцевыми часовыми механизмами

Есть ли у вас аналоговый (стрелочный) кварцевый часовой механизм или недорогой механический кварцевый будильник?

В этой главе рассказано, как в кварцевом часовом механизме после небольшой его доработки можно управлять катушкой механизма, которая создает магнитное поле для хода часов. В собственных ваших экспериментах используемый часовой механизм мог бы идти в 15—20 раз быстрее. А как насчет электронной игры для вашего ребенка с секундной стрелкой кварцевого часового механизма или вокзальных часов в качестве индивидуального счетчика (числа) часов работы?

Будьте осторожны при разборке часового механизма. Очень часто при удалении задней стенки механизма выпадают ведущие шестерни. В этом случае будет очень сложно снова правильно собрать часовой механизм.

В механическом кварцевом будильнике помимо различных электронных частей можно найти магнит, который поворачивается под действием магнитного поля катушки и тем самым приводит в действие ведущие шестерни часов. У кварцевых механизмов с так называемым "плавающим ходом" другим является только передаточное отношение. В этом случае за одну секунду секундная стрелка перемещается на 5—6 позиций, вместо одной позиции за одну секунду у механизмов с "тикающим ходом". Достоинство плавности хода первого типа механизма довольно скоро становится его недостатком, поскольку потребление тока у него значительно больше, и батарея питания разряжается намного быстрее.

8.1. Подключение катушки

В кварцевом часовом механизме (рис. 8.1) нужно припаять два провода напрямую к двум контактам катушки, которая создает соединение с USB-адаптером (рис. 8.2).

Новые провода напрямую соединяются с катушкой и через дополнительно сделанное небольшое отверстие в корпусе выводятся наружу. Если провода не подсоединены к USB-адаптеру, то кварцевый часовой механизм — после переделки остается работоспособным — может и далее питаться при помощи батареи. В этом эксперименте батарея для кварцевого часового механизма не требуется!

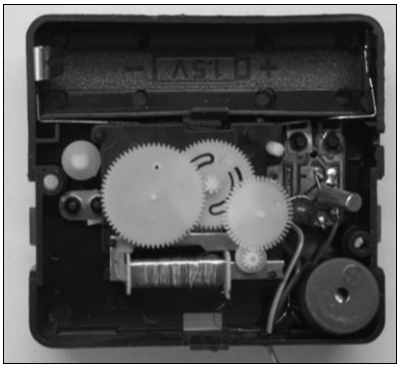


Рис. 8.1

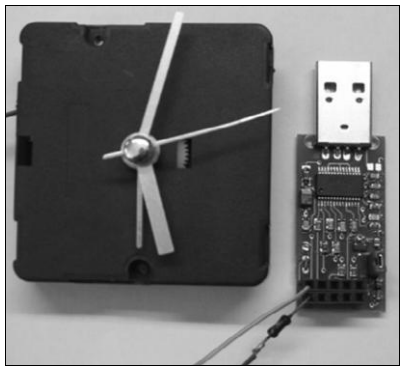


Рис. 8.2

Благодаря подведению напряжения в воздушном зазоре катушки создается магнитное поле. При этом подвижный постоянный магнит в воздушном зазоре ориентируется в магнитном поле катушки. Такое перемещение хорошо только при первом такте, но на следующем шаге магнит должен продолжить вращение. Чтобы добиться этого к катушке часового механизма можно подвести реверсированное напряжение *(см. разд. 6.6)*.

С помощью соединительных проводов катушку часового механизма подключите к выводам 17 (RTS) и 15 (DTR) 20-контактной панельки, размещенной на дополнительной плате (рис. 8.3).

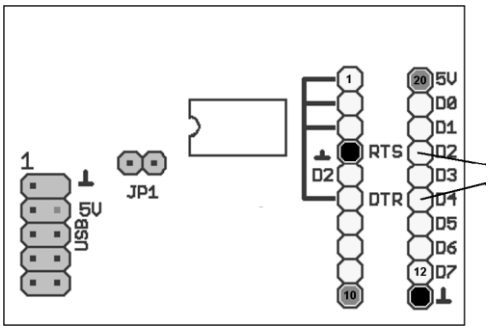


Рис. 8.3

Для продолжительной работы в одну из двух линий нужно добавить резистор сопротивлением от 100 до 500 Ом, необходимый для исключения перегрузки катушки. Однако при кратковременном экспериментировании это сопротивление не потребуется. После этого USB-адаптер с дополнительной платой можно подключить к ПК.

8.2. Программное обеспечение

Запустите программу beispiel_9.exe, находящуюся на компакт-диске в каталоге \Beispielprogramme\Bsp_9. Код программы похож на код примера 5 (\Beispielprogramme\Bsp_5) для двухцветных светодиодов, однако имеются некоторые дополнения для выполнения регулировки временных параметров.

В поле **Impulsdauer in ms** задается длительность импульса в миллисекундах, в поле **Impulsabstand in** — интервал между импульсами в миллисекундах, а в **Anzahl Durchläufe** — количество прогонов (проходов) циклов. Десять прогонов соответствуют примерно 20 с, когда продолжительность импульса настроена на 100 мс и интервал между импульсами 400 мс, т. к. оба этих значения складываются.

Если нажата кнопка **Test**, то ваш часовой механизм должен выполнить 20 секундных тактов (рис. 8.4).

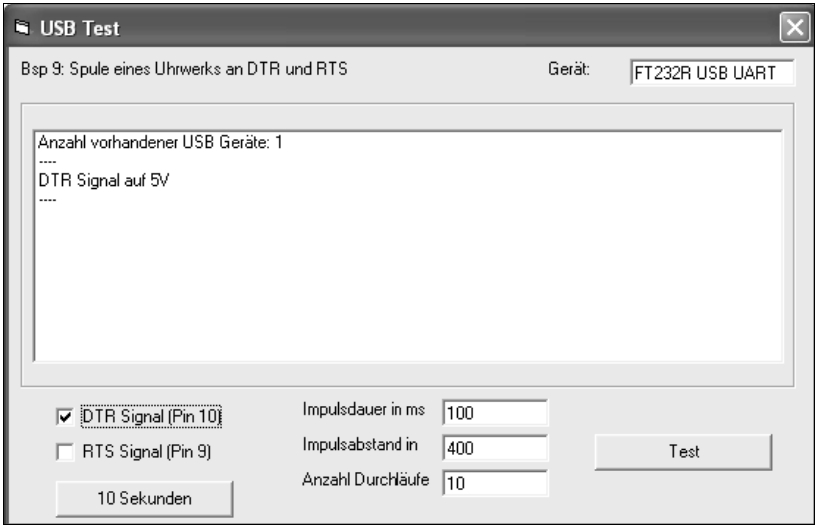


Рис. 8.4

Как можно использовать данную программу, например, для электронной игры? Если вы сократите длительность импульсов и выберите ее в интервале 10—40 мс, а интервал между импульсами сделаете равным нулю, то ваши часы будут идти нестабильно, иногда даже в обратном направлении. Если повысить число прогонов, то может случиться, что секундная стрелка будет несколько раз произвольно менять свое направление.

Если удалить минутную и часовую стрелки из вашего часового механизма, то при каждом нажатии кнопки **Test** секундная стрелка будет останавливаться в случайном (произвольном) положении.

Хрипение или звучание?

Вместо катушки из кварцевого часового механизма подключите "пищалку" (миниатюрный динамик, если он у вас имеется). Получится ли у вас с помощью USB сгенерировать сигнал тревоги будильника?

ГЛАВА 9

Режим Bit Bang

До этого вы познакомились с возможностями последовательных сигналов интерфейса микросхемы FT232R. В распоряжении имелись только предопределенные (стандартные) входы и выходы последовательного порта. Однако в так называемом режиме Bit Bang все происходит несколько иначе. Этот режим уже встречался в некоторых предыдущих версиях микросхем компании FTDI. Режим Bit Bang разрешает двунаправленно использовать восемь линий ввода/вывода (I/O) микросхемы FT232R. Каждая линия ввода/вывода (I/O) может быть настроена отдельно как на вывод (Output), так и на ввод (Input). Дополнительно имеются и другие служебные CBUS-линии в режиме Bit Bang, которые также могут быть использованы двусторонне. Сигналы CBUS в USB-адаптере не используются. Передача байтов (Byte-Transfer) с помощью сигналов CBUS ограничена USB-кадрами, и соответственно, медленная.

Скорость передачи данных в режиме Bit Bang определяется тактовой частотой генератора и равна 16-кратной скорости передачи в бодах.

Так при скорости передачи, равной 9600 бод, теоретическая передача данных составляет 9600×16 байт/с = 153 600 байт/с, причем один байт передается за 6,5 мкс.

В режиме Bit Bang микросхемы FT232R может быть использован асинхронный режим, известный из предшествующих версий микросхем компании FTDI. В противоположность асинхронному режиму в синхронном режиме состояние линий ввода/вывода (I/O) сначала считывается и сохраняется в буфере чтения, прежде чем оно будет записано. Данные поэтому записываются только тогда, когда имеется в наличии соответствующее место для чтения, что при неправильном управлении непреднамеренно может привести к потере данных.

USB-адаптер имеет на выходе двухрядный 10-штырьковый соединительный разъем, который в режиме Bit Bang имеет разводку, приведенную на рис. 9.1.

На дополнительной плате сигнальные линии от D0 до D7 находятся на правой стороне 20-контактной панельки для микросхем (рис. 9.2).

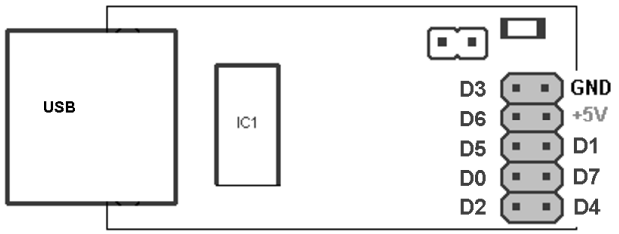


Рис. 9.1

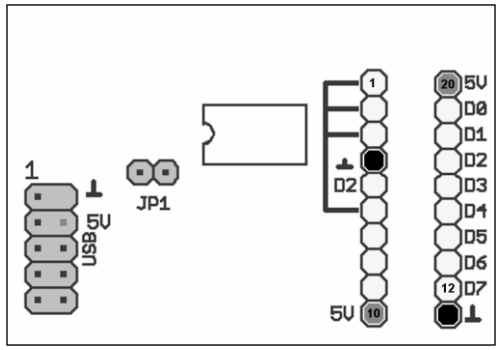


Рис. 9.2

9.1. Синхронный режим Bit Bang

Следующий пример отображает процесс включения режима Bit Bang, конфигурирования, параллельной отправки данных и считывания данных из внутреннего буфера чтения микросхемы FT232R. При синхронном режиме предыдущее состояние линий ввода/вывода (I/O) сначала читается и сохраняется в буфере чтения, прежде чем записывается первый байт. При этом важно, чтобы всегда имелось свободное место в буфере чтения. Если в буфер записывается только один байт, то нужно записать еще второй байт, чтобы, как в нашем случае, микросхема FT232R прочитала первый байт.

Для выполнения примера 11 подключите USB-адаптер к вашему компьютеру и запустите программу beispiel_11.exe, находящуюся на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_11. При нажатии кнопки **Test synchronous BitBang** в области для вывода событий вы должны получить сообщения, аналогичные приведенным на рис. 9.3.

Сначала задается скорость передачи в бодах `300` и устанавливается режим Bit Bang, затем записываются данные `1234567 ... LMNOP`. Следующая строка будет содержать данные, полученные из буфера чтения: `P1234567 ... LMNO`. Скорее всего, после нового вызова в качестве первого знака в буфере чтения вы получите заглавную букву `P`.

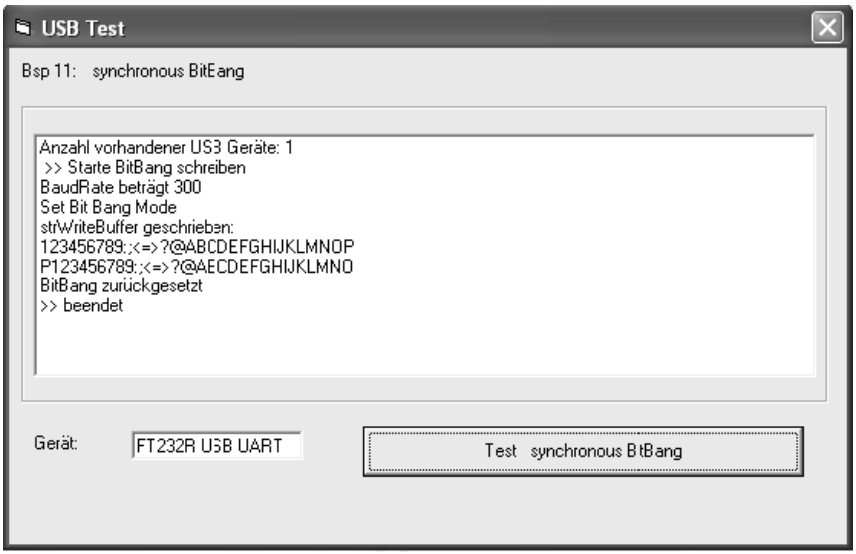


Рис. 9.3

Соотношение данных для записи и данных чтения таково:

```
1234567  ... LMNOP
P1234567 ... LMNO
```

Первый знак в буфере чтения содержит последнее состояние предыдущей записи! Только со второго знака в буфере чтения результат будет идентичен записанным значениям. Чтобы прочитать записанный последний символ `P`, необходимо прочитать первый считанный байт.

Как теперь можно определить, были ли входы на самом деле прочитаны? Сигнальные линии D0 до D7 были определены в демонстрационном ПО в качестве входов (рис. 9.4). Результат в буфере чтения подтверждает, что на входе не происходило никаких изменений в процессе записи.

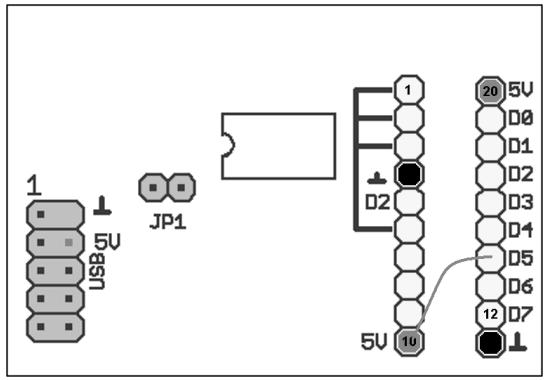


Рис. 9.4

Удалите USB-адаптер из вашего компьютера и подключите к USB-адаптеру дополнительную плату. С помощью провода соедините линию передачи данных D5 (вывод 14) с напряжением питания +5 В (выводом 10 20-контактной панельки для микросхем).

Теперь снова подключите USB-адаптер к дополнительной плате и заново запустите тест в режиме Bit Bang. После этого придется немного подождать, пока операционная система вновь свяжется с USB-драйвером. Результат выглядит сейчас иначе — из прописных (заглавных) букв получились строчные буквы:

```
 1234567 ... LMNOP
p1234567 ... lmno
```

С помощью проволочной перемычки изменяется сигнал на входе D5. Для того чтобы понять, как из прописных букв получились строчные, нужно рассмотреть коды букв в виде комбинаций двоичных последовательностей.

Заглавной букве L в коде ASCII соответствует такая двоичная комбинация: 01001100_2 (в шестнадцатеричной системе — $4C_{16}$), строчной латинской букве l — 01101100_2 ($6C_{16}$). Вход D5 (это шестой бит, если смотреть справа налево, начиная с 0) благодаря проволочной перемычке устанавливается на 1, следовательно, прописная буква L становится строчной буквой l. Однако почему же цифры 1, 2 или 3 и т. д. остаются без изменений? Так, например, цифра 1 в коде ASCII имеет следующую двоичную последовательность $00110000 1_2$ (в шестнадцатеричной системе — 31_{16}), при этом бит D5 уже установлен, и потому не происходит никаких изменений после добавления проволочной перемычки.

Исходный код для одиннадцатого примера находится на компакт-диске в файле Bsp.VBP каталога \Beispielprogramme\Bsp_11. Следующие подпрограммы, представленные далее в сокращенном виде (листинг 9.1), вызываются, когда нажата кнопка **Test asynchronous BitBang** во время выполнения программы в Visual Basic.

Листинг 9.1

```
Private Sub bt_TestBitBang_Click()
Dim i As Long
' очистить дисплей
LoggerList.Clear
.
' сообщение о количестве имеющихся USB-устройствах
.
' установка скорости
FtStatus = FT_SetBaudRate(lngHandle, 300)
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler SetBaudRate"
     GoTo CloseHandle
   Else
     LoggerList.AddItem "BaudRate beträgt 300"
End If
```

```
intMask = &HFF ' D0-D7 установить в качестве выходов
intMode = 4 ' синхронизировать Bit Bang
' установить режим Bit Bang
FtStatus = FT_SetBitMode(lngHandle, intMask, intMode)
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler bei FT_SetBitMode"
     GoTo CloseHandle
   Else
     LoggerList.AddItem "Set Bit Bang Mode"
End If
' заполнить строку символами
strWriteBuffer = ""
For i = 49 To 80
     strWriteBuffer = strWriteBuffer & Chr$(i)
Next i
...
```

После инициализации устройства при помощи вызова функции `FT_SetBaudRate` сначала следует установить скорость передачи в бодах. Для этого требуется глобальная переменная `lngHandle`, однозначный номер для открытого канала связи и значение для скорости передачи данных.

Вызов функции `FT_SetBitMode` активирует и деактивирует режим Bit Bang в значении переменной `intMode`. При вызове функции одновременно в переменной `intMask` будет пересылаться то, как должны использоваться входы.

Значения для `intMode`:

- 0 — выключить режим Bit Bang;
- 1 — асинхронный Bit Bang;
- 4 — синхронный Bit Bang.

При помощи переменной `intMask` каждая отдельная сигнальная линия может быть настроена на вход или выход, а значение `&HFF` означает, что все сигнальные линии от D0 до D7 используются в качестве выхода. Бит D0 в переменной `intMask` служит в данном случае для сигнальной линии D0: 1 — это использование сигнальной линии в качестве выхода, а 0 — в качестве входа.

Затем следует строка данных `strWriteBuffer`, предназначенная для вывода, которая заполняется ASCII-символами с кодом от 49 до 80 (в шестнадцатеричной системе 31—50, т. е. от знака `1` до знака `P`).

```
If FT_Write(lngHandle, strWriteBuffer, Len(strWriteBuffer), _
                              lngBytesWritten) <> FT_OK Then
      LoggerList.AddItem "Write Failed"
      GoTo CloseHandle
Else
      LoggerList.AddItem "strWriteBuffer geschrieben:"
End If
LoggerList.AddItem strWriteBuffer
```

Вызов функции FT_Write помимо параметра lngHandle нуждается еще в трех других. В качестве второго значения должен передаваться указатель на выходной буфер (программисты C должны были подумать о префиксе указателя на адрес). В данном случае выдается содержание strWriteBuffer. Следующий параметр указывает число передаваемых байтов, т. е. передается длина цепочки знаков strWriteBuffer. В четвертом и последнем параметре функция отражает число записанных байтов. Поэтому параметр lngBytesWritten заранее устанавливается в 0.

```
Do
FtStatus = FT_GetStatus(lngHandle, FT_RxBytes, FT_TxBytes, _
                                                    lngevent_get_stat)
      If FtStatus <> FT_OK Then
          If FtStatus = FT_IO_ERROR Then
             LoggerList.AddItem "FT_GetStatus IO Error aufgetreten"
             GoTo CloseHandle
          End If
      End If
Loop Until FT_RxBytes = lngBytesWritten
```

Перед чтением при помощи функции FT_GetStatus нужно проверить, как много байтов содержит (Rx)-буфер чтения. В этом Do-Loop-Until-цикле осуществляется ожидание до тех пор, пока в буфере чтения не будет находиться столько FT_RxBytes (байтов), сколько их было записано ранее при вызове функции FT_Write.

```
flTimedout = False
flFatalError = False
lngTotalBytesRead = 0
strReadBuffer = ""
Do
     lngBytesRead = 0
     FtStatus = FT_Read(lngHandle, strReadBuffer, FT_RxBytes, _
                                                      lngBytesRead)
     If (FtStatus = FT_OK) Or (FtStatus = FT_IO_ERROR) Then
         If lngBytesRead > 0 Then
            lngTotalBytesRead = lngTotalBytesRead + lngBytesRead
            Else
             flTimedout = True
        End If
       Else
        flFatalError = True
     End If
Loop Until (lngTotalBytesRead = FT_RxBytes) Or _
           (flTimedout = True) Or _
           (flFatalError = True)
```

Прямое чтение из буфера чтения микросхемы FT232R происходит при помощи вызова функции FT_Read. В этом случае цикл выполняется, пока все байты (количест-

во в `FT_RxBytes`) не будут прочитаны. В параметре `lngTotalBytesRead` содержится общее число прочитанных байтов. Прочитанные данные записываются в `strReadBuffer`. Внутри цикла обрабатываются такие возможные случаи ошибок, как `Timedout` или `FatalError`:

```
If (flTimedout = False) And (flFatalError = False) Then
   LoggerList.AddItem strReadBuffer
ElseIf flTimedout = True Then
   LoggerList.AddItem "FT_Read timeout Fehler ftStatus = " & FtStatus
Else
   LoggerList.AddItem "FT_Read Fehler ftStatus = " & FtStatus
End If
```

Затем пользователю в окне вывода показывается результат в `strReadBuffer`, а также возможно возникшие ошибки в `LoggerList`:

```
intMode = 0
FtStatus = FT_SetBitMode(lngHandle, intMask, intMode)
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler bei FT_SetBitMode"
     GoTo CloseHandle
   Else
     LoggerList.AddItem "BitBang zurückgesetzt"
End If
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
      LoggerList.AddItem "Fehler bei Aufruf: FT_Close"
      Exit Sub
   Else
      LoggerList.AddItem ">> beendet"
End If
End Sub
```

В конце с помощью функции `FT_SetBitMode` при `intMask = 0` вызывается программа для деактивации режима Bit Bang. После этого канал связи `lngHandle` снова становится закрытым.

9.2. Опрос входных сигналов от D0 до D7 при помощи режима Bit Bang

Цифровые входные сигналы могут опрашиваться в режиме Bit Bang при помощи вызова функции `FT_GetBitMode`. Для этого соответствующие линии сигналов должны находиться на входах. Если к входу ничего не подключено, то на выходе будет находиться напряжение 5 В, т. к. входы в микросхеме FT232R имеют резистор сопротивлением от 200 кОм, подключенные к напряжению 5 В.

Следующий пример 12 поможет вам в экспериментировании с режимом Bit Bang. Подключите USB-адаптер с дополнительной платой к вашему компьютеру и запустите программу beispiel_12.exe, находящуюся в каталоге \Beispielprogramme\Bsp_12 прилагаемого компакт-диска (рис. 9.5).

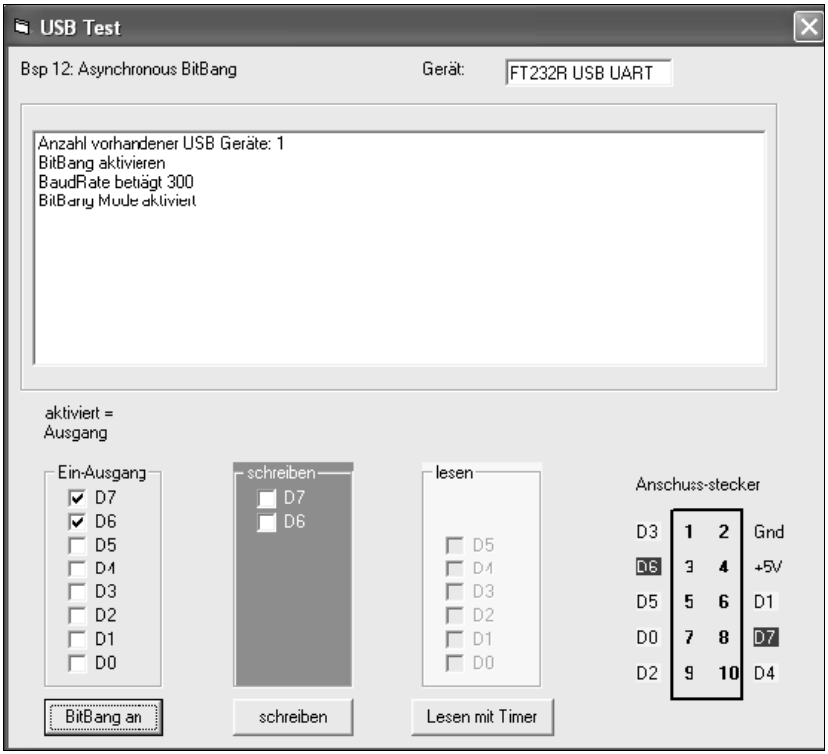


Рис. 9.5

При подключенном USB-адаптере нажатием кнопки **BitBang an** вы можете активировать режим Bit Bang на время ваших экспериментов. В группе флажков **Ein-Ausgang** (вход-выход) определите для каждой сигнальной линии, должна ли она использоваться в качестве цифрового входа или цифрового выхода. Ваш выбор будет графически изображен справа, на разводке двухрядного 10-контактного соединительного штырькового разъема USB-адаптера.

Активация или деактивация выходного и входного сигналов

После того как вы выбрали, какие выходы хотите использовать, во втором слева поле, окрашенном синим цветом, вы можете установить имеющуюся в распоряжении сигнальную линию в 0 или 1. При нажатии кнопки **schreiben** (запись) сигнальные линии будут активированы (если включен режим Bit Bang). Вы можете легко испытать функции при помощи светодиода, который подключен на выбранном выходе.

Опрос входов

Входы могут периодически опрашиваться в предварительно настроенном 500-миллисекундном цикле при помощи кнопки: **Lesen mit Timer** (чтение с таймером).

Каким образом вы можете теперь тестировать вход? Либо вы снова установите проволочную перемычку от соответствующего сигнального входа к "земле" (вывод 11 на 20-контактной панельке дополнительной платы), либо попытаетесь проделать это при помощи фоторезистора (LDR), как в одном из предыдущих примеров: в общем, подведите один контакт ко входу, а другой — к "земле". При затенении фоторезистора и выполнении операции чтения на соответствующем входе должно отображаться измененное состояние.

9.3. Исходный код для режима Bit Bang

Исходный код для примера режима Bit Bang содержится в файле Bsp.VBP на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_12. При выполнении вызываются следующие подпрограммы (листинги 9.2—9.4).

Листинг 9.2. Активация режима Bit Bang

```
Private Sub bt_BitBangAn_Click()
.
' установить скорость передачи данных
FtStatus = FT_SetBaudRate(lngHandle, 300)
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler SetBaudRate"
     GoTo CloseHandle
   Else
     LoggerList.AddItem "BaudRate beträgt 300"
End If
intMode = 4 ' синхронный Bit Bang
' установить режим Bit Bang
FtStatus = FT_SetBitMode(lngHandle, intMask, intMode)
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler bei FT_SetBitMode"
     GoTo CloseHandle
   Else
     LoggerList.AddItem "BitBang Mode aktiviert"
End If
CloseHandle:
.
End Sub
```

Вызовы функций известны из предыдущего примера. Параметр `intMask` для соответствующих сигнальных линий осуществляет выбор между входом или выходом.

Листинг 9.3. Запись одного байта

```
Private Sub bt_wr_Click(Index As Integer)
.
' заполнить строку введением в поле выбора
Databyte = 0
For i = 0 To 7
           If Me("cb_wr" & Trim(Str(i))).Value = 1 Then
            Databyte = Databyte Or 2 ^ i
           End If
Next i
strWriteBuffer = Chr$(Databyte)
If FT_Write(lngHandle, strWriteBuffer, Len(strWriteBuffer), _
                                  lngBytesWritten) <> FT_OK Then
     LoggerList.AddItem "Write Failed"
     GoTo CloseHandle
Else
     LoggerList.AddItem "Datenbyte geschrieben:" & strWriteBuffer
End If
.
End Sub
```

Байт для вывода должен быть собран из отдельных битов (в окрашенном голубым цветом поле выбора). Отдельные поля выбора битов, которые должны записаться, именуются последовательно: `cb_wr0`, `cb_wr1`, `cb_wr2` до `cb_wr7`. `cb_wr0` заменяет собой LSB (самый младший бит), а `cb_wr7` — MSB (самый старший бит).

В зависимости от соответствующей позиции бита в байте данных записываемый байт данных собирается в цикле от 0 до 7.

Если активирован D0, то `cb_wr0` имеет значение 1. При 1 последняя (правая) позиция устанавливается в `2^0` (1). Если активирован D1, то `cb_wr1` имеет значение 1. При 1 вторая с конца позиция устанавливается в `2^1` (2). Аналогично продолжая последовательность, если активирован D7, то `cb_wr7` имеет значение 1. При 1 первая позиция устанавливается на `2^7` (128). По окончании при помощи функции `FT_Write` выводится байт данных.

Листинг 9.4. Чтение и обработка входных сигналов

```
Private Sub Timer1_Timer()
.
' статус ввода/вывода (I/O)
FtStatus = FT_GetBitMode(lngHandle, Databyte)
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler bei FT_GetBitMode"
      GoTo CloseHandle
```

```
   Else
' и теперь побитно показать байт данных
For i = 0 To 7
   Me("cb_rd" & Trim(Str(i))).Value = ((Databyte And 2 ^ i) / 2 ^ i)
  Next i
End If
.
End Sub
```

Чтение происходит периодически при помощи таймера 1 (Timer1) и функции `FT_GetBitMode`. `FT_GetBitMode` в байте данных поставляет актуальное состояние входных сигналов. Также здесь положение бита в байте данных определяет соответствующий сигнал: Bit 0 (LSB) — для сигнала D0, Bit 7 (MSB) — для входного сигнала D7.

Для изображения отдельных битов нужно дальше анализировать байт данных. Обнаружение отдельных битов в прочитанном байте данных происходит в цикле `i` от 0 до 7 (в зависимости от соответствующей позиции бита в байте данных):

```
((Databyte And 2 ^ i) / 2 ^ i)
```

При `i = 0` все равно получается 0, если соответствующий входной сигнал D0 является 0 (Low), в других случаях — 1. Одновременно с результатом активируется флажок:

```
Me("cb_rd" & Trim(Str(i)))
```

При `i = 0` соответствует флажок `cb_rd0` для младшего бита LSB. Отдельные флажки обозначаются от `cb_rd0` до `cb_rd7`.

9.4. Режим Bit Bang и эмуляция других портов

Благодаря режиму Bit Bang микросхемы FT232R многие системы сбора (регистрации) данных и измерений могут полностью обходиться без микроконтроллера. На рис. 9.6 представлены лишь некоторые из применений расширения.

Для синхронного режима Bit Bang используются следующие вызовы функций в соответствующей библиотеке FTD2XX.DLL:

- `FT_SetBitMode` — значение 0x04 активирует синхронный режим Bit Bang, 0 вызывает сброс в стандартный режим;
- `AE_SetBaudRate` — определяет скорость передачи данных. Скорость передачи данных равняется 16-кратной настроенной скорости передачи в бодах;
- `FT_Write` — запись данных;

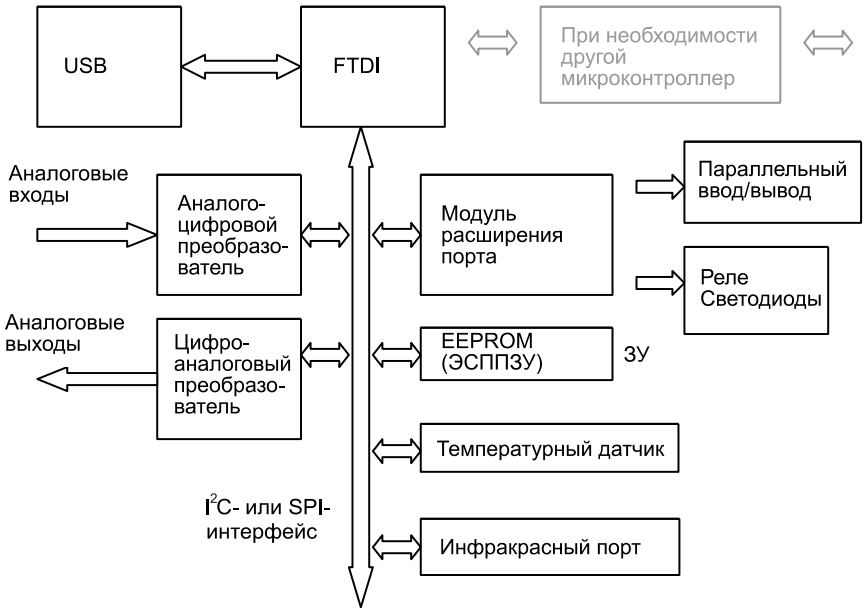


Рис. 9.6

- `FT_GetBitMode` — состояние ввода/вывода (I/O) на данный момент;
- `FT_Read` — чтение данных, которые были прочитаны микросхемой FT232R при синхронной записи.

В качестве примеров в этой книге приводится чтение и запись памяти EEPROM и более сложная демонстрационная программа для программирования семейств микроконтроллеров AT89LP производства компании Atmel.

Также показаны примеры электрических схем для I²C-ЦАП (рис. 9.7) и модуля расширения порта I²C (рис. 9.8).

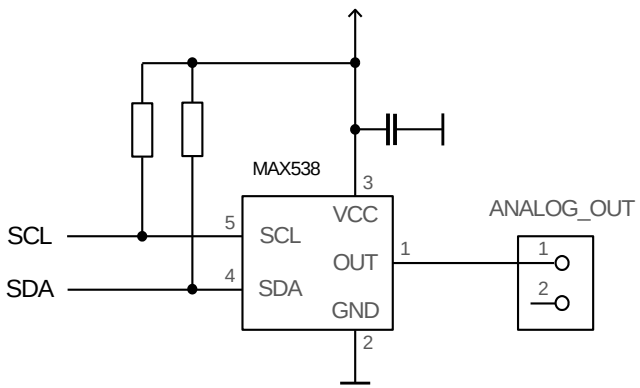


Рис. 9.7

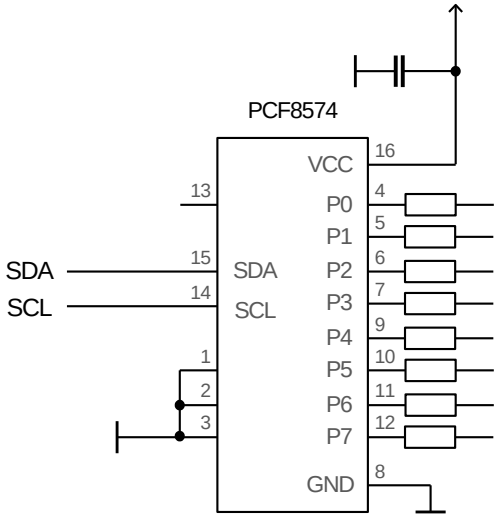

PCF8574
VCC
16
13
P0
4
P1
5
SDA
15
SDA
P2
6
SCL
14
SCL
P3
7
P4
9
1
P5
10
2
P6
11
3
P7
12
GND
8

Рис. 9.8

ГЛАВА 10

Простой АЦП с использованием режима Bit Bang

При помощи АЦП (аналого-цифрового преобразователя) можно оцифровывать аналоговые напряжения. В противоположность цифровым данным с (дискретными) значениями 0 или 1 аналоговый сигнал в установленном диапазоне может принимать любое значение.

АЦП бывают 8-, 10-, 12-, 16- или 24-битные. При 8-битном преобразовании аналоговое значение представляется с помощью 256 дискретных значений (8 бит = = 1 байт – $2^8 = 256_{10}$), при 24-битном преобразовании — 16,7 млн дискретных значений (24 бита = 4 байта – $2^{24} = 16777216_{10}$).

Если аналоговый сигнал с напряжением 5 В при помощи 8-битного преобразователя превращается в цифровые значения, то преобразователь выдает 256 дискретных значений (ступеней), каждое по 19,6 мВ. В случае 10-битного преобразователя (1024 значения) каждое дискретное значение составляет около 5 мВ, что примерно в 4 раза меньше по сравнению с 8-битным АЦП.

Интерес представляют также однобитные преобразователи, которые могут быть использованы в наверняка знакомых вам оптических приводах для компакт-дисков. Оцифровываются лишь изменения уровня сигнала. В наиболее простом случае цифровая логическая "1" обозначает повышающееся напряжение, а "0" — понижающееся. Каждый бит представляется при этом фиксированным дискретным значением. Если напряжение остается постоянным, то происходит постоянное чередование "1" и "0". Такой процесс подходит, в частности, для звуковых сигналов.

10.1. Понятие аналого-цифрового преобразователя (АЦП)

Существуют различные методы аналого-цифрового преобразования, как, например, *прямого преобразования* или *параллельного преобразования* (АЦП:прямого преобразования), метод *последовательного приближения* или *с поразрядным уравновешиванием* и метод *однократного* и *двойного интегрирования*, которые здесь не будут рассмотрены. Сами методы вообще различаются по скорости преобразования, времени преобразования, точности или линейности. Чем больше разрешающая способность, чем выше частота дискретизации и меньше возможных источников ошибок. Короче говоря, чем более дорогостоящая электроника, тем дороже и АЦП.

В АЦП аналоговый сигнал считывается с определенной фиксированной частотой. Довольно простым является АЦП, основанный на методе однократного интегрирования (Single-Slope), который здесь был еще более упрощен для преобразования положительных напряжений. Измеряемое напряжение преобразуется во время, пропорциональное напряжению. Измерение времени осуществляется путем подсчета тактовых импульсов в течение этого времени.

Если конденсатор с емкостью C заряжается *постоянным* током I, то напряжение U на нем линейно возрастает за время t. Это напряжение постоянно сравнивается с входным напряжением U_e (в индексе "e" от нем. *eingang* — вход). Сравнение происходит при помощи интегральной схемы — компаратора (нем. *komparator*), выполненного на *операционном усилителе*, кратко называемого ОУ. Как только линейно возрастающее напряжение превысит входное напряжение U_e к моменту времени tx, на выходе ОУ внезапно меняется выходное напряжение компаратора U_k, т. е. он срабатывает. Промежуток времени от момента t0 до момента tx является, таким образом, характеристикой, соответствующей величине входного напряжения U_e. Чем больше проходит времени до того момента, когда срабатывает компаратор, тем больше оказывается входное напряжение.

К моменту времени t0 запускается линейно возрастающее напряжение, после чего периодически опрашивается выходной сигнал компаратора U_k. С каждым тактом счетчик увеличивает свое значение. Счетчик остановится, как только изменится выходное напряжение на компараторе U_k. При этом состояние счетчика будет соответствовать измеряемому времени. На рис. 10.1 счетчик останавливается после момента времени tx, в данном случае после завершения двенадцати периодов тактовых импульсов. Если удвоить количество тактовых импульсов — на рисунке они показаны между основными тактовыми импульсами пунктирными линиями — то удвоится также и разрешающая способность напряжения U_k.

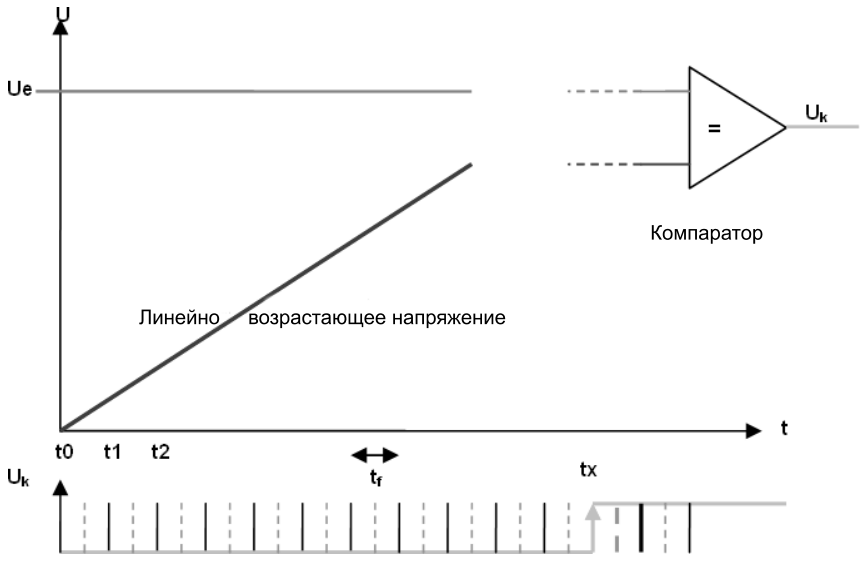


Рис. 10.1

Период счетных импульсов t_f вместе с тем представляет достигаемую разрешающую способность. Если за 256 тактов при измерении достигается напряжение 5 В, то это соответствует разрешающей способности для входного напряжения U_e порядка 20 мВ, при удвоении количества импульсов (до 512 тактов) — около 10 мВ.

АЦП, оборудованный последовательным интерфейсом I^2C для цифрового вывода аналоговых входных сигналов, был бы, разумеется, идеальным, но это немного дороже, чем представленное здесь решение. Для медленно изменяемых входных напряжений зачастую хватает даже простой экономичной схемы с небольшой точностью.

Для работы схемы потребуются: элементы для генератора линейно возрастающего напряжения, ОУ в качестве компаратора напряжения, по возможности постоянное сканирование выходов ОУ и задание определенного момента времени для начала работы схемы.

10.2. Электрическая схема АЦП с компаратором

Схема построена на основе элементов электрической схемы (рис. 10.2), размещенных на дополнительной плате.

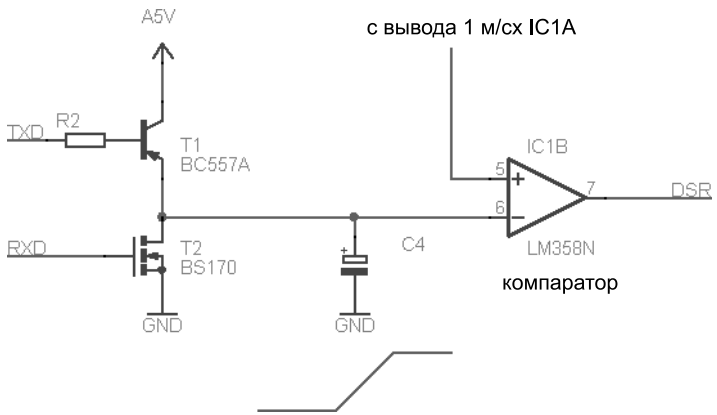


Рис. 10.2

Линейно возрастающее напряжение на конденсаторе C4 получается из постоянного напряжения 5 В (5V), поступающего на конденсатор при помощи транзистора T1. Сопротивление резистора R2, емкость конденсатора C4 и значение коэффициента усиления транзистора T1 влияют на длительность наклонного сигнала (линейно возрастающего напряжения на конденсаторе или на одном из входов ОУ).

Температурный и длительный дрейф параметров компонентов схемы влияет на постоянную времени схемы и, следовательно, на точность измерения.

При помощи сигнальной линии TxD USB-адаптера можно управлять транзистором T1, таким образом может быть определен момент времени начала заряда конденса-

тора C4. Для каждого следующего измерения конденсатор C4 должен быть разряжен. Для разряда конденсатора он замыкается накоротко на "землю" с помощью полевого транзистора T2, которым можно управлять сигналом RxD адаптера USB.

Состояние выхода компаратора опрашивается по сигнальной линии DSR. Что касается USB, то ранее вы уже научились, как управлять входными и выходными сигналами на USB-адаптере. В программе нужно лишь все правильно связать.

10.3. Первое тестирование ПО для АЦП

Программа примера 13 показывает возможности соединения простого АЦП с USB-адаптером. В данном случае оцифровываются значения яркости фотоэлемента.

Отсоедините USB-адаптер от компьютера и подключите к адаптеру дополнительную плату. На дополнительной плате (рис. 10.3) подсоедините фототранзистор (или светочувствительный фоторезистор — LDR) к входу E1 (вывод 9 20-контактной панельки) и к напряжению +5 В (вывод 10), а резистор с сопротивлением 1 кОм к второму входу E1 (вывод 8) и к "земле" (вывод 4, GND).

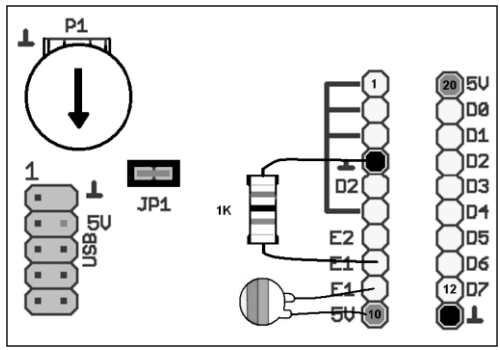


Рис. 10.3

Вставьте перемычку JP1. Коллектор *n-p-n* фототранзистора (обычно, наиболее короткий его вывод) должен быть подключен к напряжению +5 В (выводу 10 панельки). Установите потенциометр P1 в нейтральное положение. Соедините дополнительную плату с USB-адаптером и только после этого подключите адаптер к вашему компьютеру. Электрическая схема подключения фотоэлемента к операционному усилителю LM358N показана на рис. 10.4.

После этого из каталога \Beispielprogramme\Bsp_13 запустите программу beispiel_13.exe и нажмите кнопку **AD Lesen**, чтобы начать периодическое измерение (кнопка **A/D Test** — для выполнения однократного непериодического измерения).

Если вы нажали кнопку **AD Lesen**, то она меняет свое наименование на **AD stoppen** (рис. 10.5). Данные АЦП опрашиваются каждые 500 мс, результаты представляются справа около кнопки. Следует учитывать, что выводимые данные это только результаты счетчика АЦП, и не являются значениями напряжения!

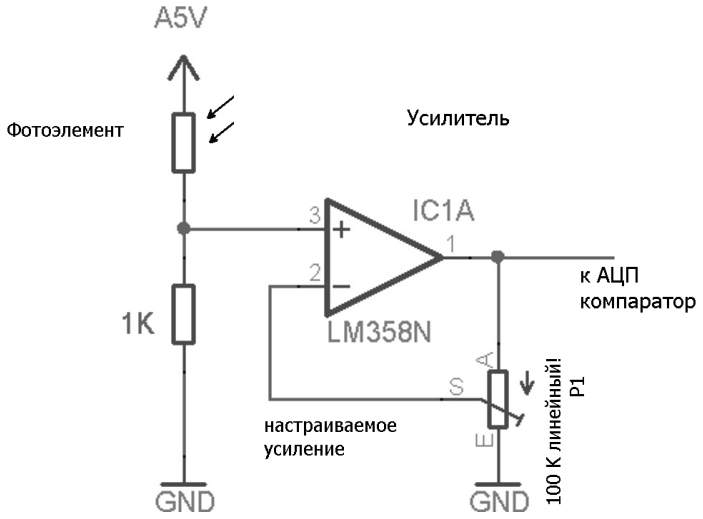


Рис. 10.4

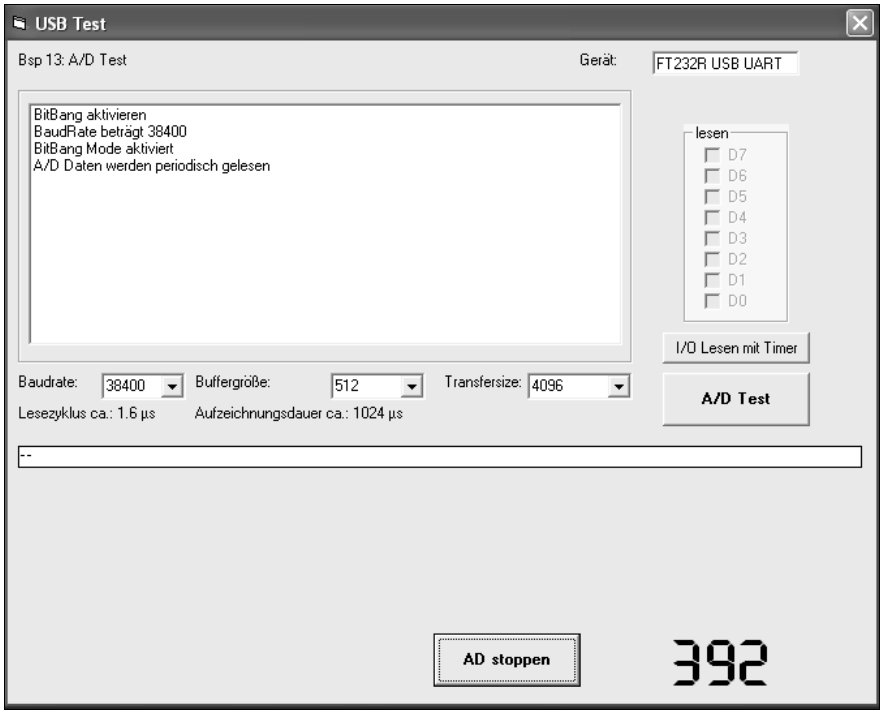


Рис. 10.5

Если вы затените фототранзистор или фоторезистор (LDR), то полученные значения станут меньше или соответственно больше, при освещении фотоэлемента настольной лампой. Переполнение в области показаний изображается знаками ">>>". При необходимости уменьшите скорость передачи в бодах, чтобы согласовать диа-

пазоны яркости со значениями счетчика. При помощи потенциометра P1 (см. рис. 10.4) вы можете настроить коэффициент усиления ОУ IC1A и согласовать схему.

Если же вы захотите работать с настраиваемыми значениями скорости передачи в бодах, заданной величиной буфера и количеством передаваемых данных, следует остановить все еще активную программу и после выполнения соответствующих изменений запустить ее снова.

Вы можете сразу заметить, что в данном примере есть большие изменения по сравнению с программой, приведенной в *примере 10*.

10.4. Согласование между программным и аппаратным обеспечением USB

С синхронным режимом Bit Bang вы уже познакомились в *примере 11*. Имеющиеся в распоряжении 8 сигнальных линий могут, в зависимости от использования каждой из них, быть сконфигурированы в качестве входа или выхода. Сигналы TxD и RxD конфигурируются как выходы, а все другие сигналы остаются входами.

В синхронном режиме, прежде чем записать следующий байт информации, сначала читается и сохраняется в буфере чтения предыдущее состояние сигнальных линий. При этом очень важно, чтобы в буфере чтения всегда имелось свободное место. Внутренний буфер записи (FIFO TX Buffer) микросхемы FT232R имеет размер 128 байтов, а буфер чтения (FIFO RX Buffer) — 256 байтов. 128 байтов соответствуют 7-разрядному АЦП. Одновременно с этим осуществляется передача по USB. Скорость передачи при полноскоростном режиме передачи по USB (Full-Speed) составляет 64 байта за 1 кадр, что соответствует 6-разрядному АЦП.

В режиме Bit Bang запись и чтение происходит с 16-кратной заданной скоростью передачи в бодах. Скорость передачи данных в бодах по сути указывает частоту выполнения операций. При скорости передачи 9600 бод длина цикла передачи одного байта в режиме Bit Bang составляет около 6,5 мкс. Если умножить ее на 128 байтов буфера чтения, то получим 833 мкс длительности считывания или 417 мкс для 64 байтов (рис. 10.6).

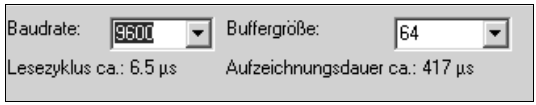


Рис. 10.6

Для использования в буфере записи, который заполняется только до заданного количества байтов, нужно подготовить выходы TxD и RxD, а состояние сигнальной линии DSR при записи в режиме Bit Bang будет считываться автоматически.

Демонстрационное программное обеспечение автоматически определяет длительность цикла для передачи байта и максимальную продолжительность записи, а

также позволяет задать скорость передачи и размер буфера. Так, при скорости передачи в 9600 бод продолжительность записи составляет около 417 мкс, а при скорости 4800 бод — около 833 мкс (рис. 10.7).

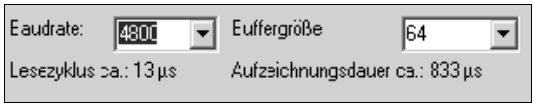


Рис. 10.7

Если вы измените размер буфера, то можете проверить, работает ли устройство при задании размера буфера более 64 байтов, является ли процесс критичным по времени, или эта возможность для вашего собственного использования вообще не может быть принята в расчет и, вероятно, вам необходим более дорогой АЦП. Для экспериментов и измерения яркости в данном случае вполне достаточно 64 уровней квантования.

Возвратимся к АЦП. Продолжительность времени для изменения линейно возрастающего напряжения от 0 до 5 В составляет 1 до 2 мс. При скорости передачи 9600 бод и 64-байтовом буфере продолжительность считывания составляет 417 мкс, т. е. значительно меньше. Иначе говоря, измеряемый диапазон напряжений может быть не от 0 до 5 В, а только от 0 до 1,5 В. Если же входное напряжение будет большим, то компаратор в это время не сработает, и не изменится сигнал DSR! Это соответствует переполнению в области показаний и изображается знаками ">>>".

Следующая временная диаграмма (рис. 10.8) поясняет некоторые взаимосвязи. Линейно изменяющееся напряжение U_{c2} достигает входного напряжения U_e в момент времени t2, в это же время срабатывает компаратор и изменяется напряжение U_k на его выходе. Если же уменьшить сопротивление резистора R2, подключенного к базе транзистора T1, или уменьшить значение емкости конденсатора C4, то линейно

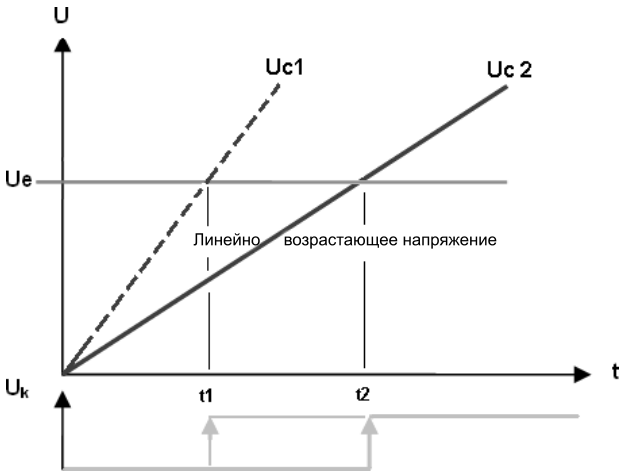


Рис. 10.8

изменяющееся напряжение на конденсаторе будет возрастать быстрее. Чтобы при помощи осциллографа исследовать поведение линейно возрастающего напряжения, можно заменить постоянный резистор R2 на потенциометр с изменяющимся сопротивлением от 0 до 200 кОм и при необходимости подключить конденсаторы с другой емкостью вместо конденсатора C4.

Линейно возрастающее напряжение U_{c1} показывает кривую напряжения при меньшем значении сопротивления резистора R2 (см. рис. 10.8). В этом случае такое же входное напряжение U_e достигается быстрее, соответственно компаратор срабатывает значительно раньше, т. е. в момент времени t1.

На следующей временной диаграмме (рис. 10.9) показаны результаты, которые были получены при изменении скорости передачи данных в бодах.

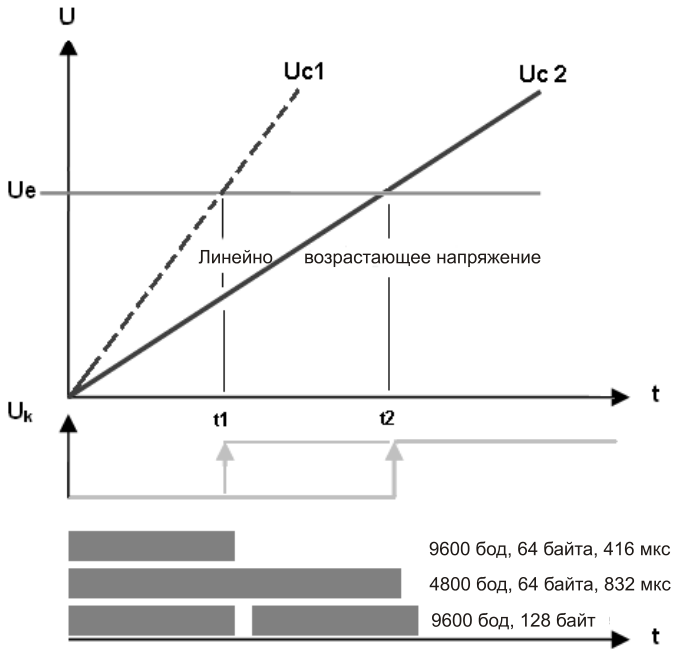


Рис. 10.9

Изменяя скорость передачи данных в бодах и размер буфера, можно согласовать время выборки данных (время преобразования) с линейно возрастающим напряжением (U_{c1} или U_{c2}). Несколько различных возможных вариантов согласования показаны под осью времени для диаграммы изменения напряжения U_k на выходе компаратора (см. рис. 10.9). Продолжительность выборки данных определяется скоростью передачи в бодах и размером буфера. При скорости передачи 9600 бод и размере буфера в 64 байта линейно возрастающее напряжение U_c за время 417 мкс обязательно должно достичь входного напряжения U_e, т. к. иначе не получить какой-либо результат измерения.

Если скорость передачи удваивается с 9600 до 19200 бод, при неизменном размере буфера, продолжительность времени выборки АЦП и соответственно измерения входного напряжения U_e сокращается в два раза.

Если за время преобразования вы не получите результата измерения, то можно уменьшить скорость передачи в бодах. Так при уменьшении скорости передачи в два раза в приведенном примере на рис. 10.9 с 9600 до 4800 бодов — продолжительность выборки данных примерно удваивается с 416 до 832 мкс.

Если же при более высокой скорости передачи в бодах хочется и увеличить продолжительность времени выборки, то размер буфера должен быть больше 64 байтов. Однако задание скорости передачи 9600 бод и размера буфера 128 байт могут привести, как показано в диаграмме (см. рис. 10.9), к прерываниям и замедлениям при выборке, т. к. микросхема FTDI поддерживает поточные передачи типа Bulk-Transfer максимум при 64-байтовой передаче данных. При очень медленном линейно возрастающем напряжении эта зависимость играет второстепенную роль, но значительно удлиняет продолжительность измерения.

Возможно, при экспериментировании вас удивит то, что счетчик иногда просто не превосходит определенного значения. Так, при размере буфера в 1024 байтов он каждый раз достигает значения только 750 байтов. Причиной является операционный усилитель, с помощью которого нельзя получить напряжение 5 В, а максимально лишь 4 В. В данном случае для регулировки диапазона измерения необходимо уменьшить коэффициент усиления входного усилителя при помощи потенциометра P1 (см. рис. 10.4).

В отличие от буфера записи и чтения микросхемы FT232R, с помощью буфера компьютера по USB может быть передано большее количество байтов. Этот "размер USB-передачи" также является настраиваемым, и он был принят во внимание в программном обеспечении данной книги.

Чем же можно объяснить колебания результатов измерения? Не только такие факторы, как буфер записи, поточная или групповая USB-передача типа Bulk-Transfer или 64 байта при полной скорости USB (USB Full-Speed), играют роль в данном случае, но и другие факторы, которые остались без внимания: колебания питающего напряжения для USB или частота 50 Гц настольной лампы — они также приводят к различным результатам при использовании фотоэлементов при измерении.

Для достижения более стабильных результатов вы можете проводить многократные измерения для медленно изменяющихся сигналов, таких как колебания дневного света.

10.5. Исходный код к АЦП

Исходный код для данного примера содержится в файле Bsp.VBP, находящегося на компакт-диске в каталоге \Beispielprogramme\Bsp_13. После запуска программы и нажатия кнопки **AD Lesen** будет вызвана функция `Sub bt_ad_read_Click` (лис-

тинг 10.1), циклически запускающая таймер 2, с помощью которого через каждые 500 мс будут опрашиваться данные.

Листинг 10.1

```
Private Sub bt_ad_read_Click()
' Timer 2 подготовить и запустить или остановить
If TimerIstAn Then
    Me.bt_ad_read.Caption = "AD Lesen"
    Me.Timer2.Interval = 0
    TimerIstAn = False
    LoggerList.AddItem "A/D Daten lesen gestoppt"
  Else
    intMask = &H0 Or 2 ^ TXD Or 2 ^ RXD
    Call BitBangAn
    Me.bt_ad_read.Caption = "AD stoppen"
    CLadebuffer = ""
    For il% = 0 To Val(Me.lb_bsize.Text) — 1
        CLadebuffer = CLadebuffer & Chr$(&H0)
    Next il
    LoggerList.AddItem "A/D Daten werden periodisch gelesen"
    Me.Timer2.Interval = 500
    TimerIstAn = True
End If
End Sub
```

Когда таймер 2 будет запущен, перед вызовом функции `Call BitBangAn`, должны быть определены входы и выходы для режима Bit Bang.

В программе константе `TXD` присваивается значение 0 (для D0), константе `RXD` — значение 1 (для D1), константе `RTS` — 2 (для D2), …, константе `RI` — 7 (для D7). С помощью логической операции ИЛИ сигнальные линии RxD и TxD объявляются выходами. Другие биты в `intMask` остаются в 0 — являются входами.

Чтобы каждый раз не приходилось заполнять буфер записи, когда работает таймер, область памяти, выделенная для переменной `CLadebuffer`, однократно заполняется нулевыми значениями.

После открытия USB-устройства передается байт со значением 1 (листинг 10.2). Первый бит с помощью выхода TXD управляет транзистором T1, который закрывают (переводят в состояние высокого сопротивления), а конденсатор C4 при этом медленно разряжается. Со следующим байтом (со значением 3) бит 1 остается в значении 1 (и транзистор T1 остается закрытым), а вот конденсатор C4 разряжается через открытый полевой транзистор (FET) T2, чем создается состояние старта.

Листинг 10.2

```
Private Sub Timer2_Timer()
.
strBeschreibung = Trim(Me.DeviceName.Text) & Chr(0)
```

```
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle)_
                                                        <> FT_OK Then
      LoggerList.AddItem "Fehler bei Aufruf: FT_OpenEx"
      Exit Sub
End If
strWriteBuffer = Chr$(&H1) ' с помощью вывода TXD
SendRead_FTDI_Bytes        ' закрыть транзистор T1
strWriteBuffer = Chr$(&H3) ' с помощью вывода RXD открыть T2
SendRead_FTDI_Bytes        ' и разрядить конденсатор C
strWriteBuffer = strWriteBuffer & CLadebuffer
SendRead_FTDI_Bytes        ' и писать (и читать)
For il% = 1 To Len(strReadBuffer)
            If ((Asc(Mid(strReadBuffer, il, 1))_
                  And 2 ^ DSR) / 2 ^ DSR) = 0 Then
                Me.lb_Anzahl.Caption = Str(il)
                Exit For
Next il
' выход за пределы области ?
If il >= Len(strWriteBuffer) Then Me.lb_Anzahl.Caption = ">>>"
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
      LoggerList.AddItem "Fehler bei Aufruf: FT_Close"
      Exit Sub
   Else
End If
.
End Sub
```

После того как буфер записи заполнится значением переменной `CLadebuffer`, стартует запись буфера `strWriteBuffer`. Начиная со второго байта в буфере записи имеются только лишь "нули", что соответствует состоянию старта. После этого транзистор T1 открывается, а полевой транзистор T2 закрывается на все время выполнения измерения, что приводит к формированию линейно возрастающего напряжения на конденсаторе C4 и соответственно на одном из входов компаратора.

Сразу после вызова функции `SendRead_FTDI_Bytes` в синхронном режиме Bit Bang выполняется чтение сигнальной линии DSR (отражающей состояние выхода компаратора), значение которой помещается в буфер чтения.

В небольшом цикле `For` последовательно проверяется состояние линии DSR (=D5) каждого отдельного байта в `strReadBuffer`, т. е. проверяется, изменилось ли состояние компаратора.

Если обнаруживается изменение, то программа в результате выведет состояние счетчика, и цикл будет прерван.

В табл. 10.1 изображен ход процесса измерения с указанием времени в микросекундах для скорости передачи 9600 бод.

Таблица 10.1

Сигнал	Байт	1	2	3	4	…	n–3	n–2	n–1	n
	D7	0	0							
	D6	0	0							
DSR	D5	0	1	1	1		1	0	0	0
	D4	0	0							
	D3	0	0							
	D2	0	0							
RxD	D1	1	0	0	0		0	0	0	0
TxD	D0	1	0	0	0		0	0	0	0
t в мкс		6,5	13	19,5	26					

При нажатии кнопки **A/D Test** вызывается функция `bt_ad_ test_Click`, которая использует те же самые подпрограммы, и результат затем обрабатывается графически. Информацию для графического анализа вы найдете в *примере 16* (испытание дистанционного управления).

Указание

Если вы планируете собственные разработки или хотите изменить это ПО для своих нужд, то, как только будете подключать USB-адаптер, вы каждый раз должны принимать во внимание исходное состояние (состояния на выходе) его сигнальных линий. Уже при подключении адаптера могут возникнуть конфликтные ситуации. В данном случае это имело бы место, если бы транзисторы T1 и T2 были открыты одновременно, что могло привести к короткому замыканию напряжения питания!

Кроме того, вы могли бы так ускорить ваше ПО, что транзисторы T1 и T2 или даже напряжение питания USB вашего компьютера могли бы выйти из строя.

10.6. Добавочный операционный усилитель

Операционный усилитель (ОУ) — это интегрированная схема, состоящая из транзисторов и представляющая собой дифференциальный усилитель постоянного тока с большим коэффициентом усиления с двумя входами и единственным выходом, и которая имеет различные режимы работы и схемы включения. Со схемой включения в качестве компаратора напряжения вы уже познакомились.

Используемая микросхема LM358 имеет два интегрированных операционных усилителя. Второй добавочный операционный усилитель применяется для усиления входных сигналов на входе E1 и на выходе соединен со входом компаратора АЦП.

Далее представлены три схемы включения операционного усилителя по схеме неинвертирующего усилителя, которые будут рассмотрены с точки зрения настроек потенциометра P1.

1. Потенциометр P1 находится в крайнем верхнем по схеме положении (рис. 10.10, слева). В этом случае выход ОУ соединен с инвертирующим входом (см. рис. 10.10, справа). Коэффициент усиления схемы с ОУ равен 1. Выходное напряжение U_a равно входному напряжению U_e.

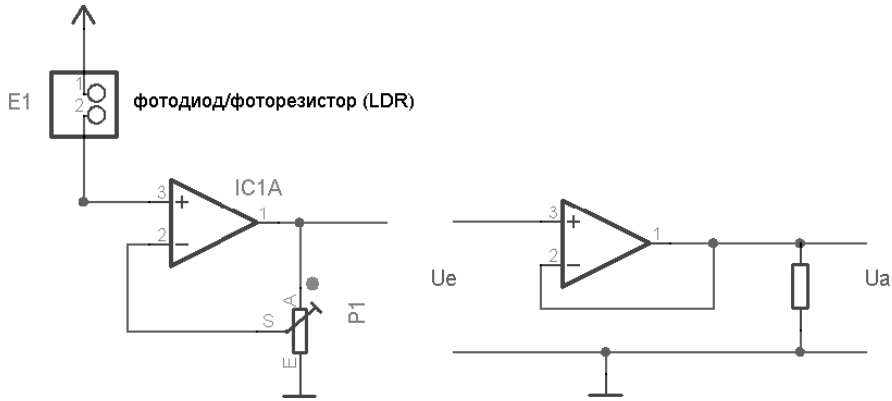


Рис. 10.10

2. Потенциометр P1 находится в среднем по схеме положении (рис. 10.11, слева). Усиление ОУ определяется из выражения: $U_a/U_e = 1 + (R1/R2)$ (см. рис. 10.11, справа). Поскольку R1 и R2 в среднем положении одинаковы, то коэффициент усиления схемы с ОУ будет равен 2. Если потенциометр P1 будет находиться в интервале от крайнего верхнего положения до среднего, то коэффициент усиления составит значения от 1 до 2.

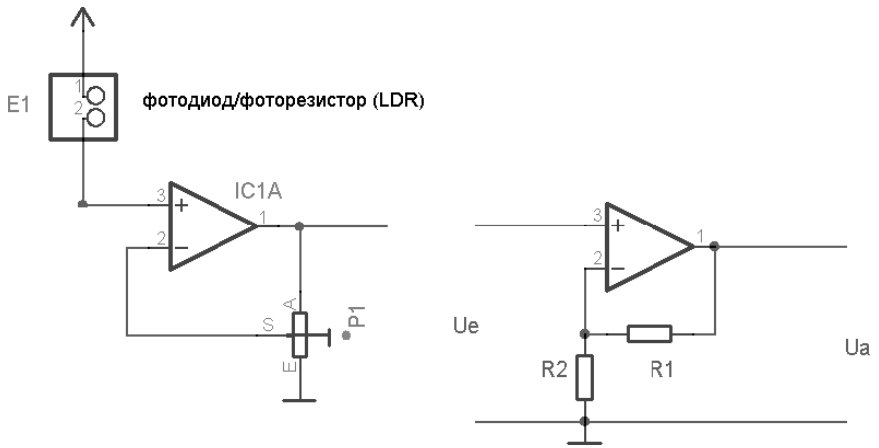


Рис. 10.11

3. Потенциометр P1 находится в крайнем нижнем по схеме положении (рис. 10.12, слева). Коэффициент усиления в этом случае будет иметь очень большое значение (в идеале оно считается бесконечно большим), определяющееся коэффициентом усиления самого ОУ (см. рис. 10.12, справа). Если потенциометр устано-

вить в промежуточное значение от среднего до крайнего нижнего по схеме положения, то коэффициент усиления при этом будет изменяться от 2 до коэффициента усиления самого ОУ.

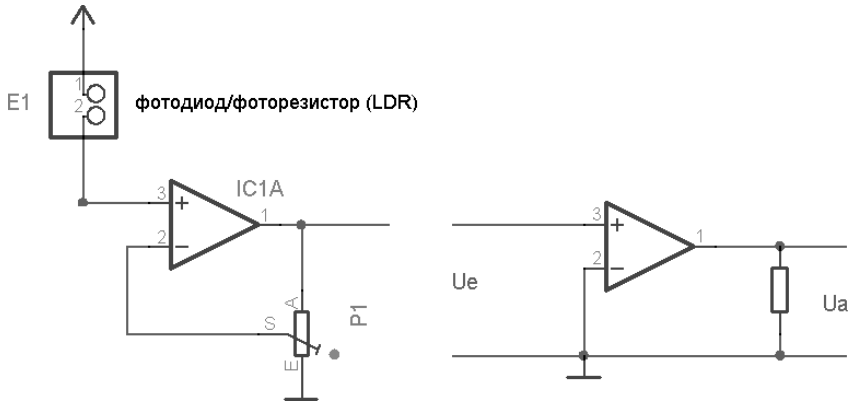


Рис. 10.12

При настройке вашего усилителя вы должны учитывать, что максимальное выходное напряжение ОУ, при рабочем питающем напряжении равным 5 В, будет немного меньше 4 В.

10.7. Измерение напряжения вольтметром на аналоговом входе E2

При помощи простого АЦП могут быть измерены аналоговые напряжения на входе E2. На первом этапе был предусмотрен диапазон измерения от 0 до 2 В. Программное обеспечение для этого примера предназначено для диапазона измерения от 0 до 4 В, которое отображается с шагом 0,1 В.

Промышленные АЦП имеют обычно очень точный источник опорного напряжения и опорные емкости. Самостоятельно собранный АЦП лишен этих преимуществ. Крутизна линейно возрастающей кривой на основании допусков микросхем будет различной, поэтому и результаты измерений могут быть неточными.

Калибровка вольтметра

Для калибровки вольтметра можно воспользоваться схемой, представленной на рис. 10.13. В ней вместе с резистором R1, подключенным между напряжением питания +5 В и входом E1, используется дополнительный резистор точно с таким же номиналом 100 кОм и подключенный между входом E1 и общим выводом источника питания — "землей" (GND). В этом случае напряжение питания в точке E1 будет делиться на 2.

Отсоедините USB-адаптер от компьютера и подключите к нему дополнительную плату. Установите резистор с сопротивлением 100 кОм между выводом 9 (E1) и

выводом 4 (GND) 20-контактной панельки (рис. 10.14). С помощью проволочной перемычки соедините вывод 7 (E2) с выводом 10 (+5 В). Вставьте перемычку JP1. Установите потенциометр P1 в положение, соответствующее коэффициенту усиления усилителя, равного 1 (см. рис. 10.10).

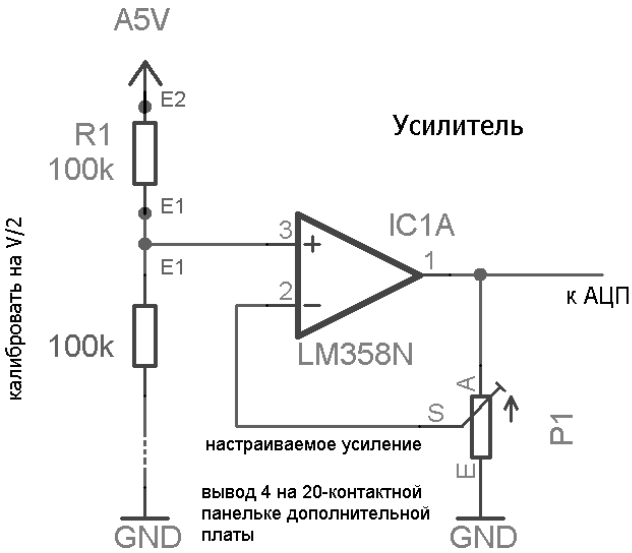


Рис. 10.13

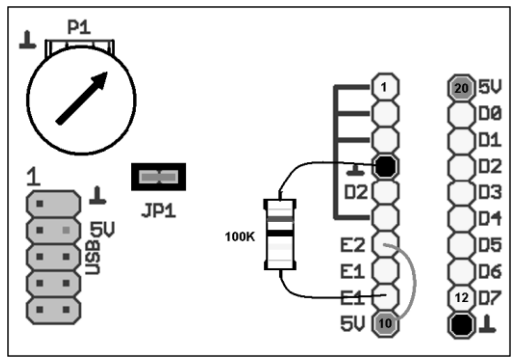


Рис. 10.14

Подключите дополнительную плату вместе с USB-адаптером к вашему компьютеру. Входное напряжение, поступающее на АЦП, должно быть равно половине напряжения питания USB (2,5 В) при соответствующей предварительной настройке потенциометра.

Запустите программу beispiel_Multimeter.exe, расположенную на компакт-диске в каталоге Bsp_Multimeter. Далее, для выполнения периодического измерения нажмите кнопку **Multimeter starten.**

Выведенное значение в левом нижнем углу диалогового окна программы под надписью **Zähler AD Wandler** (счетчик АЦП) представляет собой состояние счет-

чика, соответствующее напряжению около 2,5 В (рис. 10.15). В данном случае состояние счетчика колеблется между 594 и 604. В качестве среднего значения вносится 600. Это среднее значение вместе со значением 2,5 В входит в следующую формулу для расчета преобразования:

U = состояние счетчика × 2,5/среднее значение.

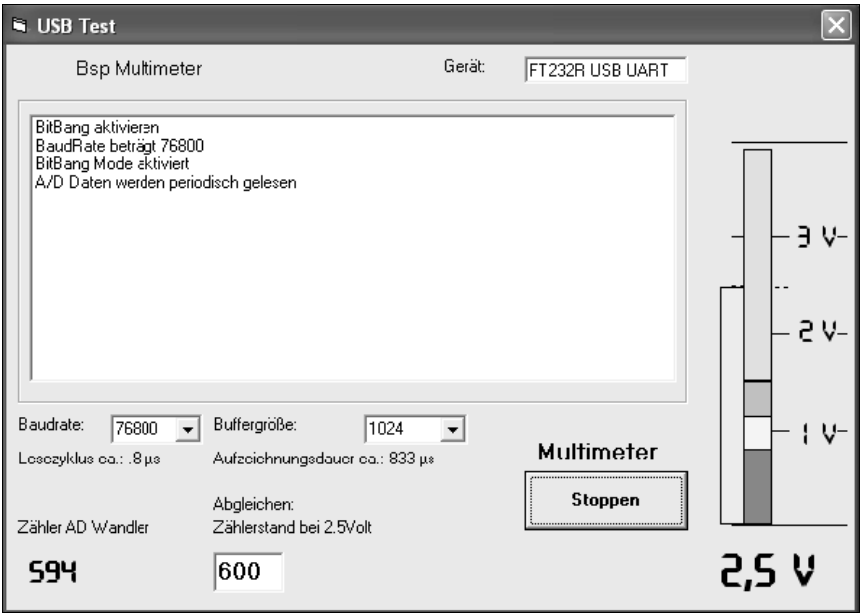


Рис. 10.15

После калибровки вольтметра надо удалить дополнительный резистор сопротивлением 100 кОм и, разумеется, также проволочную перемычку, если вы хотите измерить напряжение на входе E2.

При измерении входного напряжения U_e потенциометр P1 должен быть установлен в положение, аналогичное положению при калибровке вольтметра. Для питания ОУ должна быть вставлена перемычка JP1. Схема подключения для измерения напряжения выглядит так, как показано на рис. 10.16.

Эксперименты показали, что лучшие значения скорости передачи в бодах для стабильных результатов измерения лежат в интервале между 38400 и 76800 бод. Уменьшение размера буфера сокращает продолжительность измерения и сужает диапазон измерения. Изменение скорости с 76800 до 38400 бод делит пополам как разрешение, так и значение напряжения при индикации (что не было принято во внимание в программном обеспечении), а также расширяет вдвое диапазон измерения.

В показаниях счетчика при превышении диапазона измерения уже при входном напряжении 2,5 В вы увидите знаки ">>>". В этом случае вы можете уменьшить входное напряжение и скорость передачи в бодах, а также увеличить размер буфе-

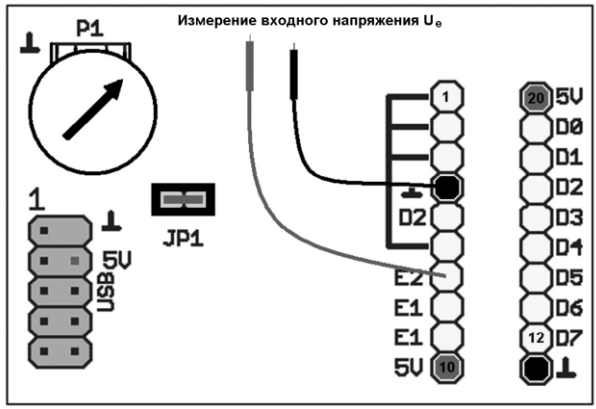


Рис. 10.16

ра. Благодаря этим мерам частота выборки согласуется с линейно возрастающим эталонным напряжением на входе компаратора.

Попробуйте поэкспериментировать с различными возможностями! Если у вас есть паяльник, то вы можете также изменить емкость конденсатора C4 на дополнительной плате и вместе с ней крутизну линейно возрастающего напряжения.

В качестве альтернативы можно также заменить сопротивление резистора R2, подключенного к базе транзистора T1, потенциометром с изменяющимся сопротивлением от 0 до 200 кОм, чтобы повысить или уменьшить крутизну линейно возрастающего эталонного напряжения.

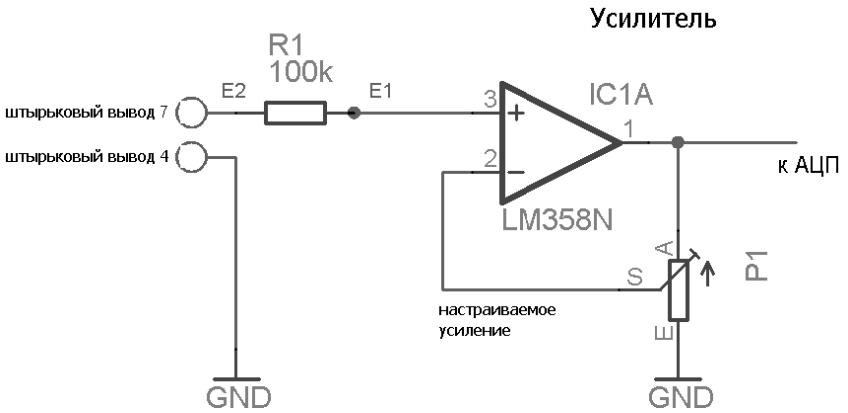


Рис. 10.17

Как уже говорилось, при больших скоростях передачи (в бод) обычно получают более стабильные результаты, чем при незначительных скоростях.

Поэтому лучше изменять крутизну линейно возрастающего эталонного напряжения при помощи изменения сопротивления резистора R2 и емкости конденсатора C4 для нужного диапазона измерения.

Конечно, вы также можете увеличить коэффициент усиления усилителя при помощи потенциометра P1 до тех пор, пока приложенное входное напряжение не будет соответствовать показанию. Необходимы лишь небольшие изменения в предыдущем примере с АЦП.

10.8. Тестер батареек питания

Емкость батареек питания типа AA или AAA с номинальным напряжением 1,5 В можно оценить с помощью тока 50 и 150 мА в нагрузке. Если напряжение под нагрузкой падает до значения ниже 0,8 В, то батарея считается негодной. Если же напряжение будет от 0,9 до 1,2 В — батарея слабая, а выше 1,2 В — батарея все еще хорошая.

Испытайте батарейку типа AA при помощи программы вольтметра из предыдущего примера. Нагрузочное сопротивление рассчитывается по формуле:

$$1{,}5\ \text{В}/0{,}05\ \text{А} = 30\ \text{Ом}.$$

Таким образом, в качестве нагрузки может подойти резистор со стандартным сопротивлением 27 Ом (рис. 10.18).

При измерении входного напряжения U_e установите потенциометр P1 в положение, соответствующее единичному усилению для усилителя. Для питания ОУ перемычка JP1 должна быть обязательно вставлена. Резистор с сопротивлением 27 Ом подключается на дополнительной плате параллельно входу между выводами 4 (GND) и 7 (E2) 20-контактной панельки (рис. 10.18).

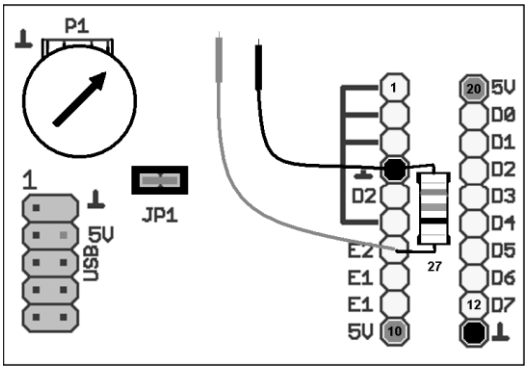


Рис. 10.18

ГЛАВА 11

Измерение температуры при помощи терморезистора с отрицательным ТКС

В этом примере вы узнаете, как использовать АЦП с резистивным датчиком, имеющим отрицательный *ТКС* (температурный коэффициент сопротивления), для измерения температуры в диапазоне примерно от –20 ºС до 40 ºС.

Резисторы с отрицательным ТКС (*NTC-resistor* — Negative Temperature Coefficient resistor) — это полупроводниковые терморезисторы с сопротивлением, зависимым от температуры. Датчики с отрицательным ТКС часто применяются в температурных переключателях, регуляторах и, конечно же, используются для измерения температуры.

Резисторы с отрицательным ТКС уменьшают значение своего сопротивления при повышении температуры. При снижении температуры значение сопротивления повышается. Иначе говоря, резисторы с отрицательным ТКС при высоких температурах проводят ток лучше, чем при низких. Следует заметить, что зависимость сопротивления полупроводникового датчика от температуры нелинейная.

Некоторые значения сопротивления 5-килоомного терморезистора с отрицательным ТКС в зависимости от температуры в градусах Цельсия представлены в табл. 11.1.

Таблица 11.1

Температура, ºС	–50	0	25	70
Сопротивление, Ом	100 000	10 000	5 000	1 000

Номинальное сопротивление терморезистора с отрицательным ТКС, как правило, указывается для температуры 20 или 25 ºС.

Номинальное сопротивление терморезистора R_{25} (обычно в Омах при температуре 25 ºС) в представленном ранее примере (см. табл. 11.1) составляет 5 кОм. На рынке этот терморезистор предлагается под наименованием NTC-5K. Общую формулу зависимости сопротивления от температуры для терморезистора с отрицательным ТКС можно представить следующим образом:

$$\frac{R_{25}}{R_T} = e^{B\left(\frac{1}{298K} - \frac{1}{T}\right)},$$

где R_T — это сопротивление терморезистора в Омах при данной температуре T; B — коэффициент, который определяется материалом терморезистора, характеризует ход кривой зависимости сопротивления от температуры и постоянен для данного экземпляра терморезистора (указывается в паспортных данных).

Для терморезистора 4.7K-NTC значение номинального сопротивления R_{25} составляет 4472 Ом, коэффициент B равен 3977. В исходном коде программы, при необходимости, эти значения вы можете поменять.

Выбор терморезистора с отрицательным ТКС осуществляют в зависимости от требуемой температуры или сопротивления, точности измерения, желаемого времени срабатывания и нужного размера.

Многие производители в своих технических паспортах для терморезисторов с отрицательным ТКС указывают немного другие формулы и таблицы, при помощи которых можно значительно улучшить точность измерений, а также далее математически линеаризовать их температурную характеристику.

Кроме того, следует уделить некоторое внимание выбору АЦП: чем точнее должно быть измерение температуры при помощи терморезистора с отрицательным ТКС, тем лучше должны быть разрешающая способность АЦП и необходимые математические вычисления. На примере вольтметра вы видели, что необходимая разрешающая способность в диапазоне измерения от 0 до 4 В легко достигается с помощью достаточно простого АЦП с 64 уровнями квантования.

Терморезистор с отрицательным ТКС одним контактом подключается непосредственно к напряжению питания, другой контакт через резистор с сопротивлением 1 кОм соединяется с "землей" (GND). Возможная область измерения от –20 ºС до 40 ºС требует двукратного усиления. Без использования усиления диапазон измерения будет больше, но при этом хуже разрешающая способность. Терморезистор с отрицательным ТКС создает на резисторе с сопротивлением 1 кОм некоторое падение напряжения, которое после измерения программно пересчитывается в сопротивление терморезистора, а затем по общей формуле для терморезисторов вычисляется и температура.

Программное обеспечение для вольтметра при измерении температуры должно быть немного подкорректировано.

11.1. Подключение терморезистора и запуск ПО для измерения температуры

Терморезистор с отрицательным ТКС должен подключаться к входу E1 на дополнительной плате. Сначала отсоедините от компьютера USB-адаптер и отключите от него дополнительную плату.

Присоедините один вывод терморезистора с отрицательным ТКС к входу E1 (вывод 9) на 20-контактной панельке дополнительной платы (рис. 11.1), а другой к напряжению питания +5 В (вывод 10). Резистор с сопротивлением 1 кОм подключают ко второму входу E1 (вывод 8) и к "земле" (вывод 4, GND) дополнительной платы.

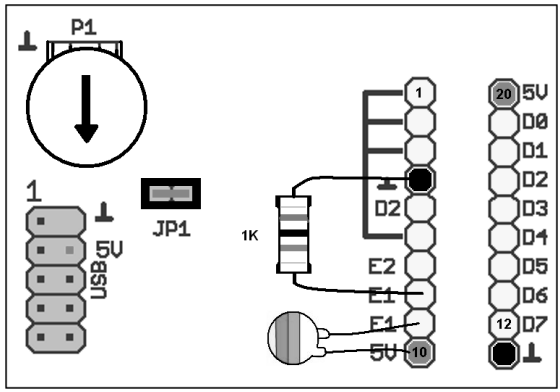


Рис. 11.1

Вставьте перемычку JP1. Установите потенциометр P1 в среднее положение. Соедините дополнительную плату с USB-адаптером и только после этого снова подключите USB-адаптер к компьютеру.

Для измерения температуры запустите программу Temperatur.exe (рис. 11.2), которую вы можете найти на прилагаемом компакт-диске в каталоге \Beispielprogramme\ Bsp_14. Поскольку ПО вольтметра используется в качестве основы, то необходимо

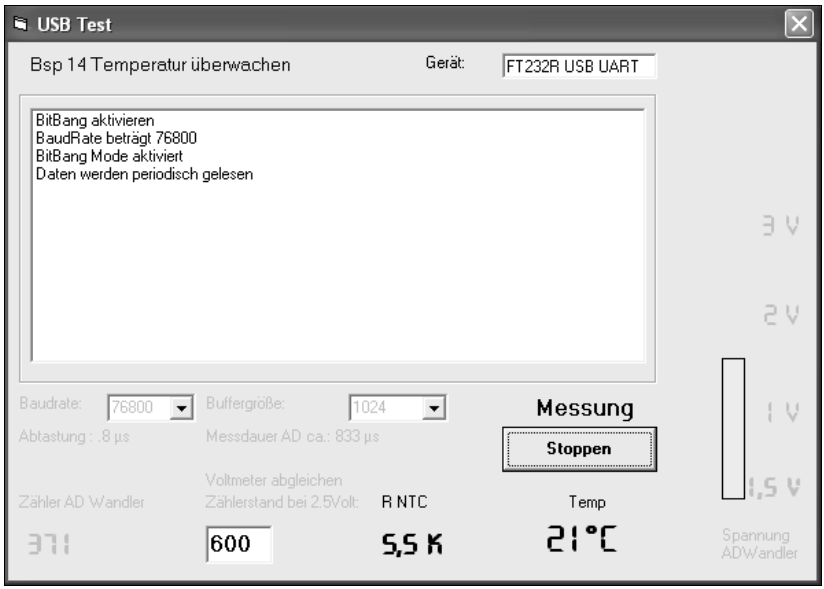


Рис. 11.2

скорректировать вольтметр, как в предыдущем примере на 2,5 В, и ввести соответствующее значение для счетчика.

Периодические измерения начинаются после нажатия кнопки **Messung starten** (начать измерения). Когда измерения будут завершены, можно остановить выполнение программы, нажав кнопку **Messung Stoppen** (остановить измерения).

В определенный момент времени на основании значения счетчика АЦП вычисляется и отображается значение сопротивления терморезистора (**R NTC**). Используя это значение и общую формулу для терморезисторов с отрицательным ТКС, рассчитывается и отображается измеренная текущая температура. Значения коэффициента B и номинального сопротивления терморезистора R_{25} вы можете изменить в исходном коде программы.

Указание

Само собой разумеется, вы можете изменить отображаемое значение текущей температуры и при помощи потенциометра, но вместе с тем также поменяется и линейность.

11.2. Исходный код программы для измерения температуры

Исходный код программы находится в файле Bsp.VBP на компакт-диске в каталоге \Beispielprogramme\Bsp_14. По сравнению с программой вольтметра в конце подпрограммы таймера 2 были проведены лишь небольшие дополнения, которые далее приведены в листинге 11.1.

Листинг 11.1

```
Private Sub Timer2_Timer()
.
' Timer2-программа дописана следующим кодом,
' чтобы определить сопротивление терморезистора и текущую температуру
Dim U_mess_1000 As Single
Dim R_NTC As Single
' двойное усиление!
U_mess_1000 = (il · 2.5 / Val(Me.lb_2_5_Volt.Text)) / 2
' сопротивление = 1000 Ом, напряжение U_mess_1000
' U_mess_1000/1000 = U_NTC/R_NTC
' U_NTC = 5 В — U_mess_1000
' R_NTC = U_NTC/U_mess_1000 · 10000 =
' R_NTC = (5B-U_mess_1000) /U_mess_1000 · 1000
R_NTC = 1000 · (5 — U_mess_1000) / U_mess_1000
' вывод показаний сопротивления терморезистора в килоомах
Me.lb_R_NTC.Caption = Format(R_NTC / 1000, "0.0") & " K"
```

```
' таблица B и R25 значений
' вычисление и вывод текущей температуры
Me.lb_temp.Caption = _
Format(1 / (Log(R_NTC / 4472) / 3977 + 1 / 298) – 273, «0») & «°C»
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
      LoggerList.AddItem "Fehler bei Aufruf: FT_Close"
      Exit Sub
  Else
End If
.
End Sub
```

Совместно с микросхемой FT232R в качестве альтернативы рассмотренного здесь терморезистора с отрицательном ТКС вместе с простым АЦП вы можете использовать готовые микросхемы термометра с высокой точностью и интерфейсом I^2C (например, DS1820 или LM75).

ГЛАВА 12

Генерирование сигналов различных частот и их применение

При помощи микросхемы FT232R можно генерировать сигналы различной частоты. Такие сигналы очень хорошо подходят для синхронизации различных устройств и задач управления.

12.1. Генератор частот для последовательного интерфейса

С этим применением генератора вы уже познакомились в *главе 6*. В этом случае вы почти всегда связаны с миллисекундным USB-кадром, кроме того, определенную роль играет и быстродействие компьютера. Чем меньше частота, тем точнее можно сформировать необходимую характеристику сигналов.

12.2. Генератор частот с использованием режима Bit Bang

В режиме Bit Bang можно генерировать более высокие частоты. Буфер чтения выдается на сигнальные линии с частотой, кратной 16, определяемой заданной скоростью передачи в бодах. При настроенной скорости 9600 бод тактовая частота составляет 153 600 Гц, а состояние сигнала может меняться каждые 6,5 мкс. Применение генератора из-за максимальной скорости передачи данных по USB в режиме полной скорости (Full-Speed) на уровне 64 байта/кадр для приложений реального времени ограничено.

Другие примеры использования генераторов тактовых сигналов вы найдете в *главе 13* для эмуляции последовательного интерфейса I^2C и в *главе 16*, где говорится об исследовании цифровых схем.

Кроме того, следует упомянуть цифровые счетчики, программирование запоминающих устройств, а также управление пьезодвигателями.

На рис. 12.1 показано, насколько просто формируются сигналы различных частот на выходных линиях D0—D7 в 8-разрядном двоичном счетчике.

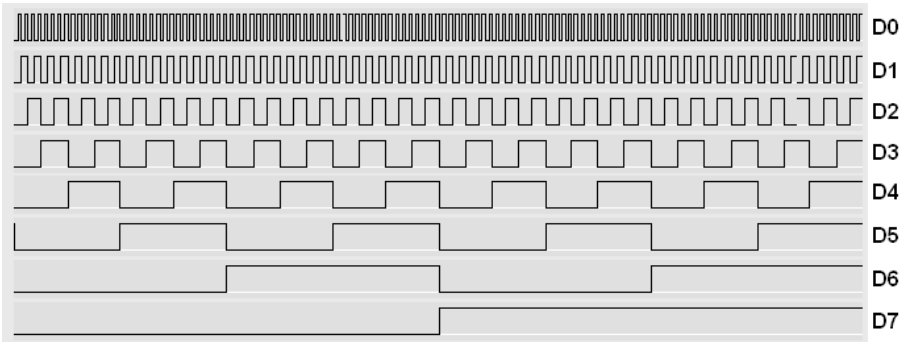


Рис. 12.1

Запустите программу Bsp_Frequengenerator.exe, которая находится на прилагаемом к книге компакт-диске в каталоге \Beispielprogramme\Bsp_15\Bsp_Frequenzgenerator.

После нажатия кнопки **START** (рис. 12.2) вы без осциллографа ничего собственно не увидите. На самом же деле на сигнальных линиях с D0 до D7 будут формироваться сигналы, которые приведены на рис. 12.1.

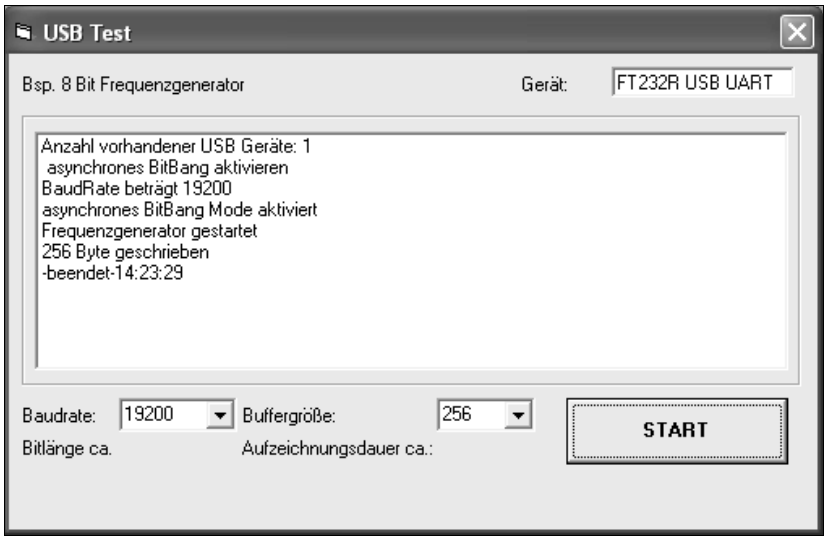


Рис. 12.2

Генератор частоты можно довольно просто реализовать на микросхеме FT232R, работающей в асинхронном режиме передачи Bit Bang (листинг 12.1).

Листинг 12.1

```
Private Sub bt_freq_Click ()
' включить асинхронный режим Bit Bang
' с выбором скорости передачи в бодах!
Call bt_BitBangAn
```

```
' наполнить буфер выбранными байтами от 0-255
strWriteBuffer = ""
For il% = 1 To Val(lb_bsize.Text) / 256
       For ib% = 0 To 255
          strWriteBuffer = strWriteBuffer & Chr$(ib)
       Next ib
Next il
LoggerList.AddItem "Frequenzgenerator gestartet"
' и записать в асинхронном режиме Bit Bang
If FT_Write(lngHandle, strWriteBuffer, Len(strWriteBuffer), _
                                  lngBytesWritten) <> FT_OK
Then
       LoggerList.AddItem "Write Failed"
       GoTo CloseHandle
Else
       LoggerList.AddItem Len(strWriteBuffer) & " Byte geschrieben"
End If
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
       LoggerList.AddItem "Fehler bei Aufruf: FT_Close"
       Exit Sub
   Else
       LoggerList.AddItem "-beendet-" & Time()
End If
End Sub.
```

Исходный код этой программы вы также найдете в каталоге \Beispielprogramme\Bsp_15\Bsp_Frequenzgenerator. В этом примере используется асинхронный режим Bit Bang. Буфер записи заполняется, прежде всего, показаниями счетчика 0 до 256, процесс при необходимости повторяется несколько раз, если заданный размер буфера составляет 512 или 1024 байтов. Для записи буфера применяется функция `FT_Write`.

12.3. Цифроаналоговый преобразователь с ШИМ

ШИМ (широтно-импульсная модуляция) называют модуляцию сигнала прямоугольной формы при постоянной частоте (рис. 12.3).

При постоянной частоте (в данном случае частота f = 50 Гц, что соответствует периоду сигнала T = 20 мс) изменяется отношение времени включения ко времени выключения. Если время *t вкл* такое же по величине, как и *t выкл*, то *коэффициент заполнения*[1] сформированного сигнала составляет 50 %, а среднее постоянное на-

[1] *Скважность импульсов* — это отношение периода следования импульсов T к длительности импульса $t_и$. Величина, обратная скважности и часто используемая в англоязычной литературе, называется *коэффициентом заполнения импульсов*. — *Ред.*

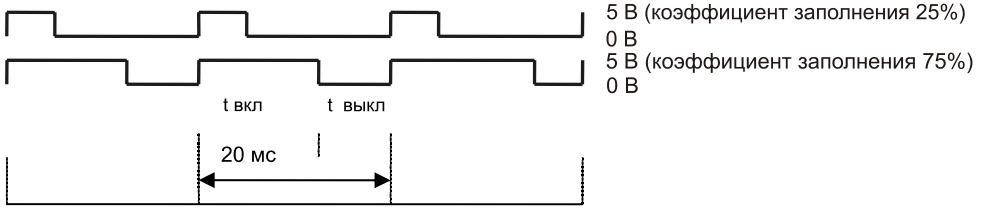


Рис. 12.3

пряжение будет равно половине приложенного напряжения. Если коэффициент заполнения меньше, как это показано на верхней временной диаграмме на рис. 6.9, то и среднее значение напряжения выходных сигналов ШИМ тоже меньше. Если же коэффициент заполнения больше, как это показано на нижней временной диаграмме, то среднее значение напряжения выходных сигналов будет соответственно больше.

В качестве сглаживающей интегрирующей цепи для выходного напряжения ШИМ может применяться обычная RC-цепочка, при условии, что постоянная времени $\tau = R \cdot C$ этой цепи будет больше длительности импульса.

При достаточно высокой тактовой частоте с помощью правильно выбранной RC-цепи можно получить постоянное напряжение, которое будет соответствовать коэффициенту заполнения. Так, при коэффициенте заполнения 50 % постоянное напряжение будет равно половине питающего напряжения USB, т. е. около 2,5 В.

ШИМ-сигналы могут быть достаточно легко программно сгенерированы для микросхемы FT232R в режиме Bit Bang. Исходный код программы (листинг 12.2) вы найдете на компакт-диске в каталоге \Beispielprogramme\Bsp_15\Bsp_DAWandler.

Листинг 12.2

```
Private Sub bt_da_test_Click()
' включить режим Bit Bang с заданной скоростью передачи
' установить нужную сигнальную линию в качестве выхода (TxD)
intMask = &H0 Or 2 ^ PWMAusgang
Call bt_BitBangAn
' заполнить буфер, 128 байтов для ШИМ
Dim ipwm As Integer
strWriteBuffer = ""
     For il% = 1 To Val(lb_bsize.Text) / 128
        ' сначала высокий (High)
        For ipwm = 0 To Me.HScroll_PWM.Value
          strWriteBuffer = strWriteBuffer & Chr$(&H0 Or 2 ^ PWMAusgang)
        Next ipwm
        ' затем низкий (low)
        For ipwm = Me.HScroll_PWM.Value To Me.HScroll_PWM.Max
          strWriteBuffer = strWriteBuffer & Chr$(&H0)
        Next ipwm
     Next il
```

```
strBeschreibung = Trim(Me.DeviceName.Text) & Chr(0)
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle)
<> _
FT_OK
Then
        LoggerList.AddItem "Fehler bei Aufruf: FT_OpenEx"
        Exit Sub
End If
' в цикле удлинить интервал
For ipwm = 1 To 10
     SendRead_FTDI_Bytes
Next ipwm
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
        LoggerList.AddItem "Fehler bei Aufruf: FT_Close"
        Exit Sub
     Else
        LoggerList.AddItem "-beendet-"
End If
End Sub
```

Для генерирования ШИМ-сигнала с коэффициентом заполнения 50 %, первые 64 бита должны формировать сигнал одного логического уровня, а следующие 64 бита — другого уровня. Период ШИМ-сигнала составляет число битов, т. е. 128, умноженных на период импульсов тактовой частоты или, иначе говоря, частота ШИМ-сигнала равна задающей частоте, деленной на 128. Используемый в программе ползунок имеет максимальное значение 127. В зависимости от настройки ползунка **Tastverhältnis** (коэффициент заполнения), который изначально установлен в среднее положение, изменяется и коэффициент заполнения генерируемого ШИМ-сигнала.

Для вывода буфера записи в цикле вызывается функция `SendRead_FTDI_Bytes`. Число проходов вы должны согласовать с вашими потребностями и выбранными RC-параметрами.

Сопротивление одним выводом подключается к сигналу TXD, а другим к аналоговому выходу, к нему же подсоединяют один вывод конденсатора, а другой его вывод подключают к "земле" (рис. 12.4). Постоянное напряжение следует измерять на конденсаторе. Конечно, в программе можно настроить и другие выходы, в качестве используемого в данном случае TXD.

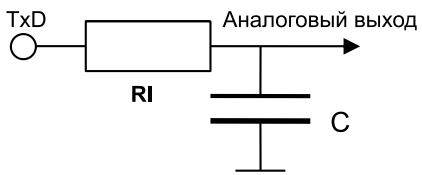


Рис. 12.4

При высоких частотах и больших значениях постоянной времени τ ($\tau = R \cdot C$) вывод сигнала из-за быстрого опустошения буфера записи будет многократно повторяться. Таким образом, при передаче данных по USB вы получите многочисленные колебания постоянного напряжения.

При сопротивлении резистора 10 кОм и емкости конденсатора 1 мкФ (1000 нФ) постоянная времени будет 10 мс, а приблизительно через 50 мс конденсатор будет полностью заряжен (продемонстрировать это на конкретном примере можно с помощью программы RC-E-Funktion.exe, находящейся на прилагаемом к книге компакт-диске в каталоге \Beispielprogramme\e-funktion). При скорости передачи 76800 бод вывод сигнала при размере буфера 1024 байта не длится и одной миллисекунды, и поэтому полное напряжение не может быть достигнуто!

При небольших значениях сопротивления резистора RC-цепи определенную роль играет также входное сопротивление сигнальной линии, равное 200 кОм.

Если для тестирования вам понадобится подключить динамик ПК (пьезогромкоговоритель или динамик с дополнительным 100-омным сопротивлением), то присоедините его через электролитический конденсатор емкостью 100 мкФ и тогда при разных скоростях передачи в бодах вы будете слышать звуки разной тональности.

При этом нужно учитывать, что ШИМ-сигналы синхронизируются с частотой, кратной 16 и определяемой заданной скоростью передачи данных, а также которая в программе делится на 128.

ГЛАВА 13

Хранение данных в EEPROM-памяти

В этой главе вы узнаете, как с помощью режима Bit Bang микросхемы FT232R посредством интерфейса USB можно общаться с EEPROM-памятью. Обратиться к памяти EEPROM можно используя последовательный интерфейс I^2C. При этом микросхема FT232R по сути становится преобразователем интерфейсов. Программное обеспечение позволяет довольно простым способом обрабатывать и определять передаваемые данные и временные соотношения сигналов в интерфейсе I^2C.

13.1. Основы EEPROM-памяти

Данные могут храниться в EEPROM-памяти (Electrically Erasable Programmable Read-Only Memory, электрически стираемое программируемое постоянно запоминающее устройство — ЭСППЗУ) десятками лет без подключенного напряжения питания.

В отличие от компакт-диска CD-R, который может быть записан лишь один раз, данные в EEPROM-память могут заново записываться в ее запоминающие ячейки многократно, аналогично многократно перезаписываемым компакт-дискам CD-RW.

Современные EEPROM в большинстве своем позволяют выполнять до миллиона циклов чтения/записи. Многие производители EEPROM поддерживают интерфейс I^2C. Описываемое запоминающее устройство без использования питания сохраняет свои данные на протяжении как минимум 40 лет.

13.2. Основы интерфейса I^2C

Микросхема EEPROM-памяти типа 24C16 управляется при помощи двух сигнальных линий шины интерфейса I^2C и соответствующих сигналов — SCL и SDA. Сигнал SCL (Serial Clock) — это сигнал синхронизации. Он обычно генерируется исключительно ведущим устройством (Master) на шине, а читается подключенными ведомыми устройствами (Slave), поэтому направление передачи устанавливается однозначно.

Сигнал SDA (Serial Data) предназначен для обмена данными. По этой линии как Master-, так и Slave-устройства передают свои данные.

Существует ряд различных точно определенных состояний, которые регулируют общий ход обмена между ведущим (Master) и подчиненным (Slave) устройствами, начиная от состояния покоя, затем старта (START), передачи данных, подтверждения данных и заканчивая состоянием останова (STOP), как это нам наглядно демонстрируют временные диаграммы, приведенные на рис. 13.1.

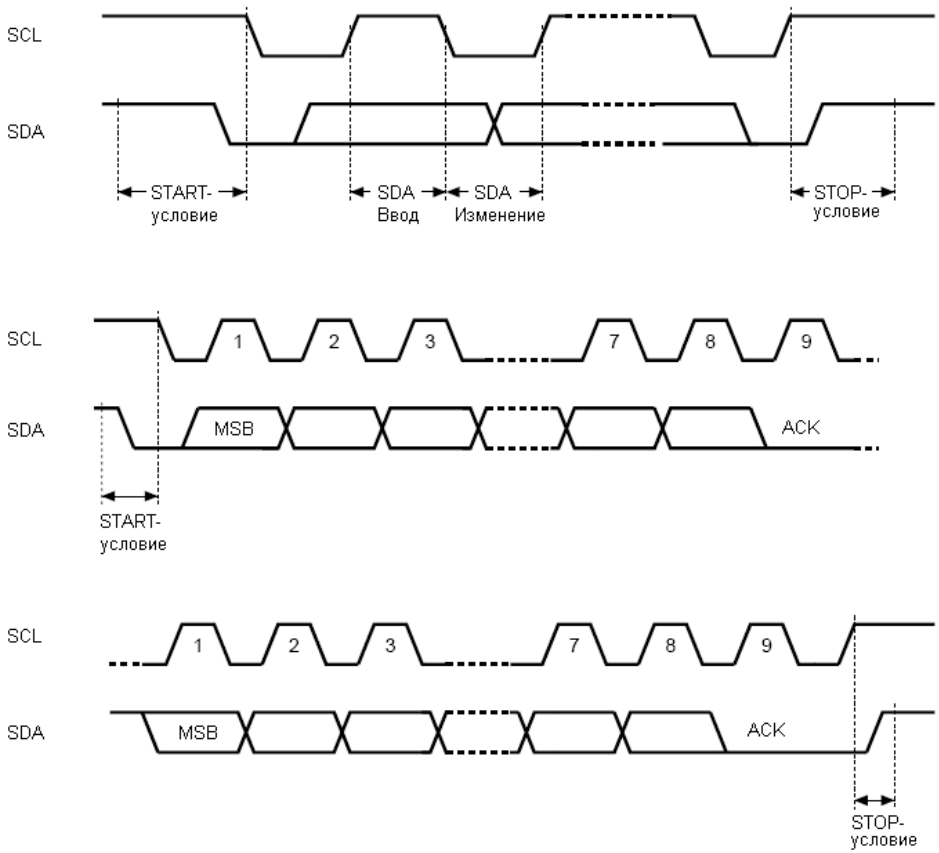


Рис. 13.1

Состояние выданного на линию SDA бита данных может изменяться только тогда, когда на линии синхронизации SCL имеется низкий логический уровень. Во время высокого логического уровня SCL выданный бит данных считается действительным. Данные передаются байтами, начиная со старшего бита MSB (Most Significant Bit — самый старший бит).

Любая передача начинается с условия START (старт), которое формирует ведущее устройство и постоянно проверяется подключенным подчиненным устройством. Условие START соответствует такому состоянию, когда сигнал SDA изменяет свой

логический уровень с высокого (HIGH) на низкий (LOW) при высоком уровне синхросигнала SCL. Завершается передача состоянием STOP (останов), которое также формируется ведущим устройством и соответствует переводу сигнала SDA с низкого логического уровня на высокий при высоком уровне сигнала синхронизации SCL. Девятый бит подтверждения ACK (acknowledge) добавляется после передачи каждого байта устройством-приемником и нулевым своим значением свидетельствует о нормальном получении очередного байта от передатчика (рис. 13.2).

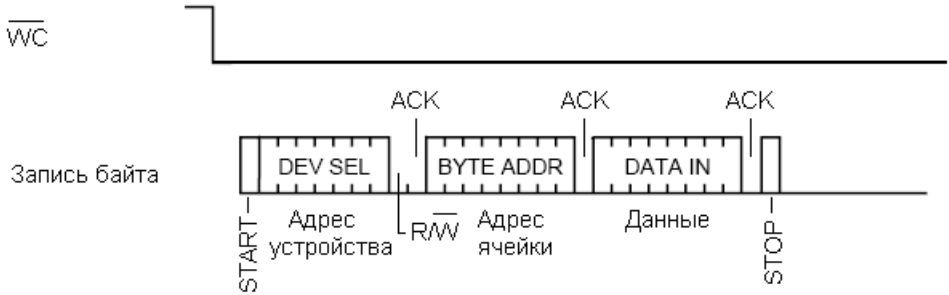


Рис. 13.2

После формирования условия START при обмене по интерфейсу I^2C передается адрес устройства DEV SEL (Device Select Byte, байт выбора устройства) и бит $R/\overline{W}$ признака операции (низкий логический уровень — "Запись", высокий логический уровень — "Чтение"). Старшие 4 бита адреса устройства A[7:4] несут информацию о типе устройства. Для микросхем EEPROM-памяти, и в частности для ST24Cxx, эти биты будут равны значению 1010. Младшие 3 бита адреса устройства A[3:1] могут определять номер конкретного устройства данного типа.

После байта адреса устройства передается адрес ячейки памяти, из которой должны читаться или в которую должны записываться данные. Для микросхемы 16-килобитной EEPROM-памяти 24C16, которая подключается посредством интерфейса I^2C и имеет 11-битный адрес ячеек памяти, 3 бита адреса устройства DEV SEL являются дополнительными битами адресов ячеек памяти — A10, A9 и A8.

Память может читаться и записываться по байтам или постранично. При постраничной записи до 16 байтов адрес следующей ячейки памяти будет определяться автоматически, каждый раз увеличиваясь на 1.

13.3. Подключение EEPROM-памяти

Электрическое подключение EEPROM-памяти с последовательным двухпроводным интерфейсом I^2C выглядит достаточно просто. На рис. 13.3 изображен пример для подключения микросхемы EEPROM типа ST24C16.

SDA — это вывод для передачи данных, SCL — вывод, предназначенный для временной синхронизации, выводы 1 по 3 — выводы для выбора кристалла для этой микросхемы не используются и поэтому подключены к "земле". WC — вывод для

управления записью (Write Control). Если на этот вывод подать напряжение +5 В, он может предотвратить случайную запись или стирание EEPROM-памяти.

Теперь вы, возможно, спросите себя, почему эта схема имеет три, а не два контакта для подключения, тем более что для микросхемы FT232R достаточно контактов RXD и RTS (или DCD)? Кроме того, зачем подключен дополнительный диод (стабилитрон)? Ответы на эти вопросы вы получите в следующем разделе.

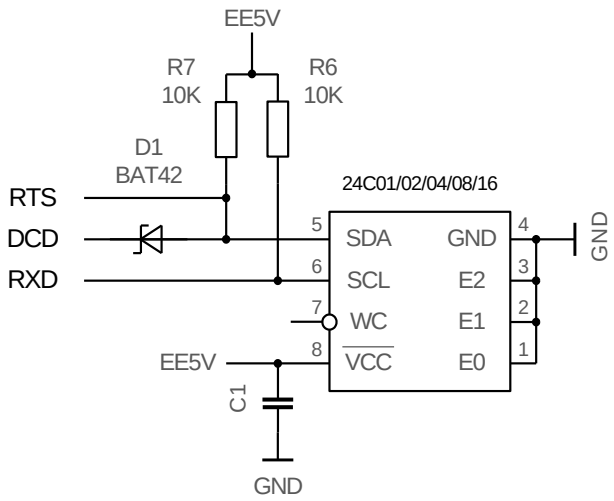


Рис. 13.3

13.4. Предварительные размышления

Для реализации последовательного интерфейса I^2C нужно иметь всего две сигнальные линии: SDA и SCL.

Многие электронщики испытывают большие трудности при работе с последовательными интерфейсами, потому что они, как правило, пользуются одним лишь осциллографом и не применяют логический анализатор. Осциллограф позволяет регистрировать периодические сигналы, а логический анализатор исследует последовательность битов непериодических, цифровых сигналов. Без анализатора можно проводить измерения в течение нескольких часов или дней. Отдельный неправильный бит может довести разработчика до отчаяния. Однако тем большей оказывается его радость, когда у него все, в конце концов, получится. Другие надеются на то, что кто-нибудь уже выполнил соответствующую разработку, и ищут готовые решения в Интернете. Кроме того, и компания FTDI, изготовитель микросхемы FT232R, предлагает свое решение интерфейса I^2C. Этим решением является готовый программный интерфейс, который удовлетворяет нескольким решениям аппаратного обеспечения. Если точно придерживаться предписаниям компании FTDI, то это законченное решение в любом случае будет проще, и вероятно, работающим быстрее. Однако если возникнет проблема (требуется устранить ошибку или вы

планируете использовать несколько другой последовательный интерфейс), то не следует ждать поддержки от производителя при планировании малого числа изделий.

Теперь, собственно, нам не хватает только логического анализатора. Вы еще, наверное, не забыли, что при синхронном режиме Bit Bang микросхемы FT232R прежде записи каждого байта сначала читается и сохраняется в буфере чтения предыдущее состояние сигнальных линий. Чтение предыдущего состояния сигнальных линий — это уже своего рода анализатор!

Для получения близкого соответствия сформированных в программе данных с (почти) действительным поведением сигналов на линии передачи данных нужно лишь представить результат буфера чтения в качестве временной диаграммы, как это сделано в данном примере.

Теперь ответ на вопрос о дополнительном диоде от линии RTS к линии DCD: микросхема FT232R в режиме Bit Bang может использовать сигнальную линию в качестве входа или выхода и лишь условно в качестве двунаправленной (одновременно в качестве входа и выхода). Линия SDA интерфейса I^2C предназначена как для передачи, так и приема данных. Чтобы прочитать данные из EEPROM-памяти, нужно сначала обратиться к ней по соответствующему адресу. Для этого сигнальная линия RTS должна находиться в состоянии выхода. После получения адреса EEPROM-память с помощью сигнала ACK подтверждает получение адреса и затем сама передает требуемые данные. Для этого сигнальная линия RTS должна находиться уже в состоянии входа. Переключение этой сигнальной линии из состояния входа в состояние выхода, и наоборот, осуществляется программно. На этом примере показан процесс функционирования двухпроводной линии связи. По причине переключения истинная временная характеристика не может быть представлена в цифровой форме на основании переключения, поскольку компьютер вынужден получать ее по программе. В этом примере с двухпроводной линией вы узнаете, как можно анализировать сигналы сначала при записи, а потом при чтении данных. Одновременная обработка невозможна.

Для получения почти истинной временной характеристики применяют дополнительный диод и другую сигнальную линию DCD, которые позволяют выполнять операцию чтения во время записи на линии передачи данных. При этом линия RTS используется в качестве входа, а DCD — в качестве выхода. Если сигнал DCD имеет высокий логический уровень, то по линии RTS можно выполнять чтение данных EEPROM-памяти.

Если сигнал DCD меняет свое состояние с высокого логического уровня на низкий, то это состояние, в свою очередь, также может быть считано с линии RTS. Таким образом, становится почти возможным в реальном времени выполнить исследование поведения сигналов интерфейса.

Вы можете точно узнать, что и как вы запрограммировали, и что отвечает присоединенная микросхема. Вы можете в режиме отладки в любой момент времени остановить ваше программное обеспечение и проверить состояние сигналов с помощью простого вольтметра, или даже одного светодиода. При этом базовые зна-

ния и это программное обеспечение будут вам также полезны для исследования других последовательных интерфейсов, таких как интерфейс SPI (Serial Peripheral Interface).

13.5. Пять шагов к успеху

Поэтапная разработка программного анализатора для интерфейса I²C

Отсоедините USB-адаптер от компьютера и затем дополнительную плату от адаптера. Вставьте микросхему EEPROM-памяти типа 24C16 в 20-контактную панельку дополнительной платы так, как показано на рис. 13.4.

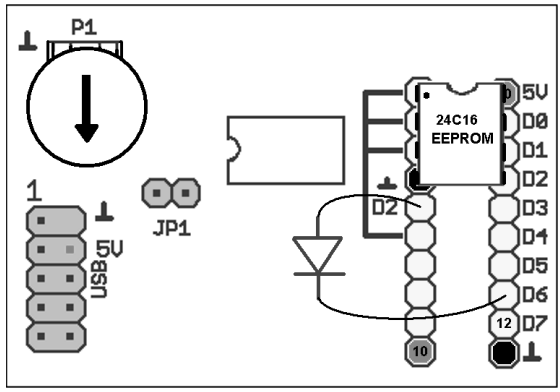


Рис. 13.4

Затем подключите анод диода BAT42 к выводу 5 (D2 = RTS), а катод к выводу 13 (D6 = DCD) этой панельки. Обратите особое внимание на правильное подключение диода, иначе в дальнейшем вы можете получить неверные результаты. Соедините дополнительную плату с USB-адаптером и после этого подключите его к вашему компьютеру. Программу для этого примера вы можете найти на прилагаемом к книге компакт-диске в каталоге \Beispielprogramme\Bsp_I2C_EEPROM\dreidraht_mitDiode. Запустите программу I2C_3.exe.

Шаг 1: генерирование условия старта для интерфейса I²C

В соответствии с изображением в техническом описании микросхемы EEPROM-памяти (рис. 13.5) условию старта соответствует условие, когда сигнал SDA меняет свое состояние с высокого логического уровня на низкий (сбрасывается) при высоком логическом уровне сигнала SCL, который с небольшой задержкой тоже становится низкого логического уровня, т. е. сбрасывается. Запрограммированный результат (после нажатия кнопки **1. I2S Start**) выглядит так, как показано на рис. 13.6.

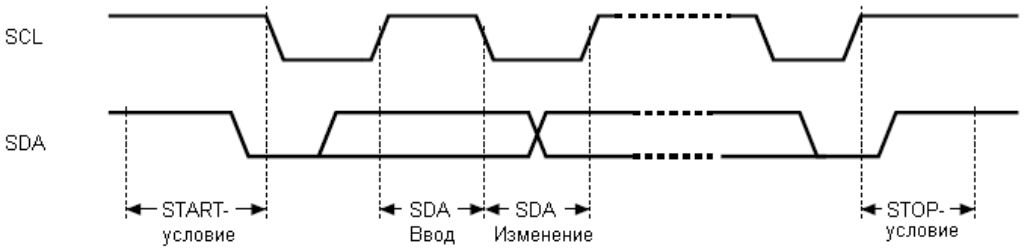


Рис. 13.5

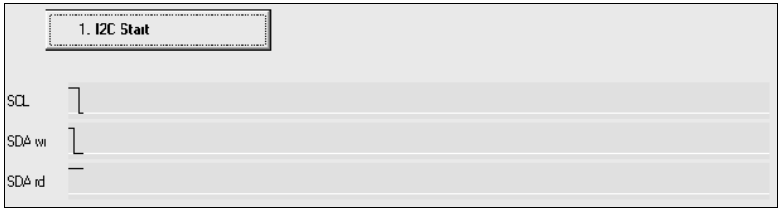


Рис. 13.6

Все последовательности данных должны быть предварительно подготовлены. Сначала данные подготавливаются как отдельные байты в буфере записи. После этого буфер записи записывается (и снова читается) в режиме Bit Bang, и наконец, результаты представляются в виде временной диаграммы для сигналов SDA и SCL.

```
Private Sub bt_I2CStart_Click()
   strWriteBuffer = ""
   I2CStart
   If FTWriteRead(True) = False Then MsgBox "FTWriteRead Error"
LoggerList.AddItem "Beendet " & Time
End Sub
```

Сначала очищается буфер записи `strWriteBuffer`. Вызывается функция `I2CStart`, после этого буфер записи записывается с помощью функции `FTWriteRead`. Задача функции `I2CStart` заключается лишь в том, чтобы записать сигналы для условия START в буфер записи `strWriteBuffer`:

```
Private Sub I2CStart()
Databyte = 0
Databyte = Databyte Or 2 ^ SCL Or 2 ^ SDA ' установить сигналы SDA и SCL
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
Databyte = Databyte And Not (2 ^ SDA) ' сбросить сигнал SDA
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
Databyte = Databyte And Not (2 ^ SCL) ' сбросить сигнал SCL
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
END Sub
```

Это очень легко запрограммировать при помощи функций `AND` (логическое И) и `OR` (логическое ИЛИ): благодаря функции `OR 2^SCL` на линии передачи данных SCL

устанавливается высокий логический уровень сигнала (5 В), на другие биты не оказывается влияния. При помощи `AND NOT (2^SCL)` на линии SCL устанавливается низкий логический уровень, другие биты в байте данных также остаются незатронутыми. Сигналы SCL и SDA при вызове программы были соотнесены со следующими сигнальными линиями: SCL = RXD, SDA = DCD и SDAREAD = RTS.

На рис. 13.7 представлено содержание буфера с временными диаграммами для I^2C-сигналов (с указанием времени для скорости передачи 9600 бод).

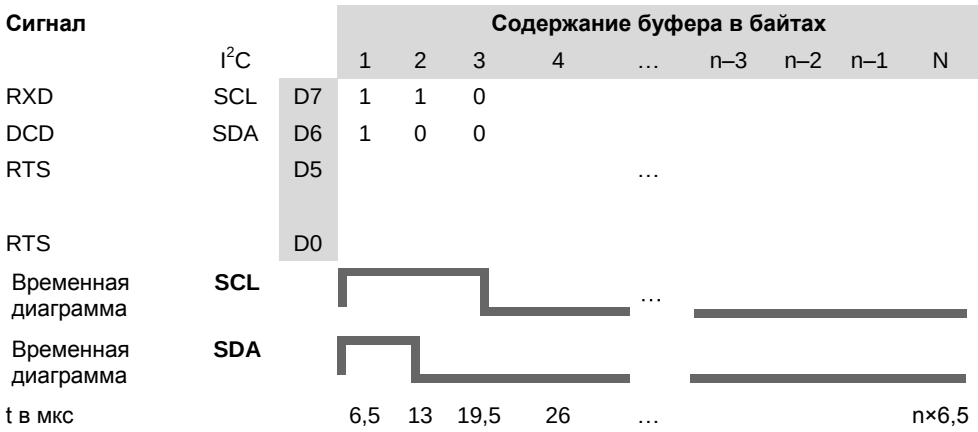


Рис. 13.7

Программа обрабатывает RXD- и DCD-биты в буфере и с помощью функции `FTWriteRead` отдельными линиями вычерчивает соответствующие временные диаграммы.

Шаг 2: генерирование условия останова для интерфейса I^2C

Со знаниями, полученными ранее, здесь нам наверняка удастся быстрее получить сигналы для дальнейших преобразований. Для начала, представим функцию останова в листинге 13.1.

Листинг 13.1

```
Private Sub I2CStop()
Databyte = 0
strWriteBuffer = strWriteBuffer & Chr$(Databyte) ' сбросить сигналы
                                                 ' SDA и SCL
Databyte = Databyte Or 2 ^ SCL ' установить сигнал SCL
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
Databyte = Databyte Or 2 ^ SDA ' установить сигнал SDA
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
END Sub
```

Сигналы на линии SCL и SDA последовательно устанавливаются на высокий логический уровень. На данном шаге сначала вызываем функцию старта, а затем функцию останова (соответствует кнопке **2. I2C Start + Stopp**). Третий шаг является наиболее сложным.

Шаг 3: разложение байта на биты для последовательной передачи

Для последовательной передачи каждый байт должен быть разложен на отдельные биты: сначала самый старший бит, т. е. MSB, а в конце самый младший — LSB. В это же время должен формироваться соответствующий синхронизирующий сигнал SCL. Это выполняется в цикле подпрограммой I2CByte (листинг 13.2).

Листинг 13.2

```
Private Sub I2CByte(ByVal data As Byte)
' разложить байт в последовательность битов
' и сразу записать в strWriteBuffer
Dim n As Integer
Dim BitPosition As Byte
BitPosition = 128 ' сначала MSB = 2^7=128, в конце LSB
For n = 1 To 8
   If data And BitPosition Then
        Databyte = Databyte Or (2 ^ SDA) ' установить сигнал SDA
      Else
        Databyte = Databyte And Not (2 ^ SDA) ' сбросить SDA
   End If
   strWriteBuffer = strWriteBuffer & Chr$(Databyte)' и один такт
   Databyte = Databyte Or (2 ^ SCL) ' установить сигнал SCL
   strWriteBuffer = strWriteBuffer & Chr$(Databyte)
   Databyte = Databyte And Not (2 ^ SCL) ' сбросить SCL
   strWriteBuffer = strWriteBuffer & Chr$(Databyte)
   BitPosition = BitPosition / 2
Next n
   ' и сгенерировать девятый бит
   ...
End Sub
```

Передаваемый байт находится в переменной `data`. Начиная с позиции самого старшего бита MSB ($128 = 2^7$), сигнал SDA устанавливается, если бит имеет значение логической единицы, иначе сигнал сбрасывается. В конце цикла переменная `BitPosition` (весовое значение соответствующего бита) каждый раз делится на два ($64 = 2^6$, $32 = 2^5$ и т. д.), последующие биты в следующих проходах цикла либо устанавливаются, либо сбрасываются. Для каждого бита с помощью функций `AND` и `OR` обязательно формируется соответствующий синхронизирующий сигнал SCL.

И в самом конце должен быть сформирован еще и девятый бит. Девятый бит подтверждения ACK сигнализирует передатчику, что данные действительно получены. Теперь сигнал SDA должен быть установлен на высокий логический уровень. Благодаря этому для чтения освобождается линия RTS.

Третий шаг объединяет все предыдущие состояния интерфейса I^2C: старт, генерирование битов при передаче байтов и останов:

```
Private Sub bt_I2C_Command1_Click()
strWriteBuffer = ""
I2CStart
I2CByte (&H55) ' писать произвольный байт
I2CStop
If FTWriteRead(True) = False Then MsgBox "FTWriteRead Error"
LoggerList.AddItem "Beendet " & Time
END Sub
```

Временная диаграмма в техническом описании микросхемы EEPROM-памяти (рис. 13.8), в сравнении с генерируемой временной диаграммой (рис. 13.9), растянута по горизонтали (имеет более пологие фронты сигналов).

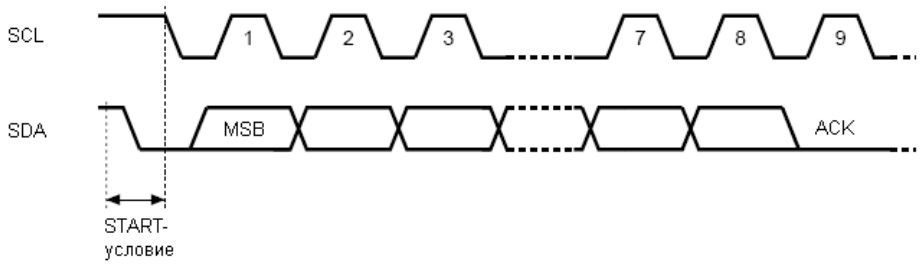


Рис. 13.8

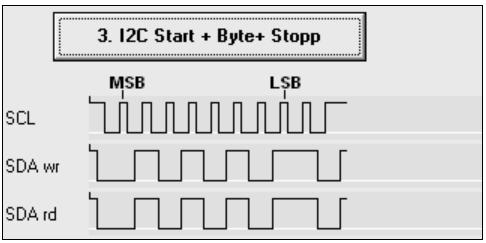


Рис. 13.9

Переданным байтом было число 85_{10} = 55H = 01010101 в двоичной системе. Можно легко заметить, что формирование состояния старта, передача байта и генерирование состояния останова выполняется совершенно правильно. Кроме того, видно формирование и девятого бита. Однако в отличие от временной диаграммы, приведенной в техническом описании, сигнал SDA во время девятого импульса синхронизации SCL остается на высоком логическом уровне.

Шаг 4: запись байта

Чтобы записать данные в микросхему EEPROM-памяти типа 24C16, нужно обязательно соблюсти правильную последовательность сигналов для функционирования интерфейса I^2C.

```
Private Sub bt_i2C_write_Click()
'DevSelNumber = 160 (адрес устройства при записи) = 1010 0000 A0 = 0

strWriteBuffer = ""
I2CStart
I2CByte Val(Me.tb_devsel.Text)     ' адрес EEPROM-памяти с интерфейсом I2C
I2CByte Val(Me.tb_byte_addr.Text) ' адрес ячейки памяти для записи данных
I2CByte Val(Me.tb_data.Text)       ' данные для записи
I2CStop
I2CByte (&HFF)                     ' сформировать 255 синхроимпульсов!!

If FTWriteRead(True) = False Then MsgBox "FTWriteRead Error"
…
End Sub
```

После формирования условия старта должен выводиться адрес устройства, в данном случае адрес EEPROM-памяти, представленный в текстовом диалоговом поле **DevSel** (Device Select) (рис. 13.10) и соответствующей переменной `tb_devsel`. Этот адрес в данном случае будет равен 160_{10}, причем последним битом A0 адреса 1010 0000 в двоичной системе является 0, который определяет признак операции записи. Далее определяется адрес ячейки памяти, в которую должна быть произведена запись. Предварительно настроенный адрес ячейки памяти равный 0, представленный в текстовом поле **Address** и соответствующей переменной `tb_byte_addr`, может быть при желании изменен пользователем. Байт данных, предназначенный для записи, отображается в текстовом поле **Data** и представляется соответствующей переменной `tb_data`. В это текстовое поле нужно вводить числа только из диапазона от 0 до 255, поскольку программа не определяет диапазон вводимого числа. После передачи данных формируется условие останова. Кроме этого в последовательности сигналов на шине интерфейса I^2C должны быть два дополнительных такта ACK — один, подтверждающий получение адреса ячейки, и другой — данных.

Сгенерированную временную диаграмму (см. рис. 13.10) вы можете проанализировать и сравнить с техническим описанием микросхемы EEPROM-памяти.

После старт-условия следуют первые биты 1010 0000, в соответствии с адресом устройства 160_{10} (DEVSEL) (восьмой бит указывает на операцию записи, а девятый — подтверждает получение приемником адреса устройства). За битами адреса устройства передаются адрес запоминающей ячейки равный 0 и соответствующий бит подтверждения. Затем следуют данные (0101 0101), соответствующие значению 85, которое было введено в поле **Data.**

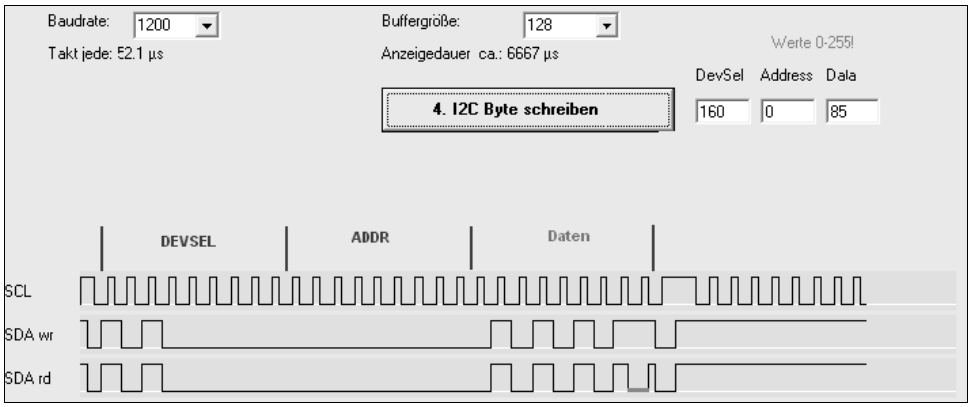


Рис. 13.10

После восьмого бита данных сигналы SDA wr и SDA rd расходятся. Вспомним (см. рис. 13.3), что линия SDA интерфейса I^2C управляется через диод, для того чтобы при записи одновременно можно было осуществлять чтение по другой линии (в данном случае RTS). Итак, линия DCD соответствует сигналу SDA wr, а линия RTS — CDA rd. Диод с помощью сигнала на линии DCD заперт. Микросхема EEPROM-памяти сразу же после окончания восьмого бита данных с помощью сигнала CDA низкого логического уровня (сигнал SDA rd) подтверждает получение данных. Однако является ли верным результат, полученный без использования какого-либо измерительного прибора, и был ли в действительности записан байт?

Шаг 5: чтение байта

Результат, полученный после шага 4, может быть проверен путем чтения из той же самой ячейки EEPROM-памяти, в которую была выполнена запись. Если вы выберете пятый шаг в программе (кнопка **5. I2C Byte lessen**), то получите результат, приведенный на рис. 13.11.

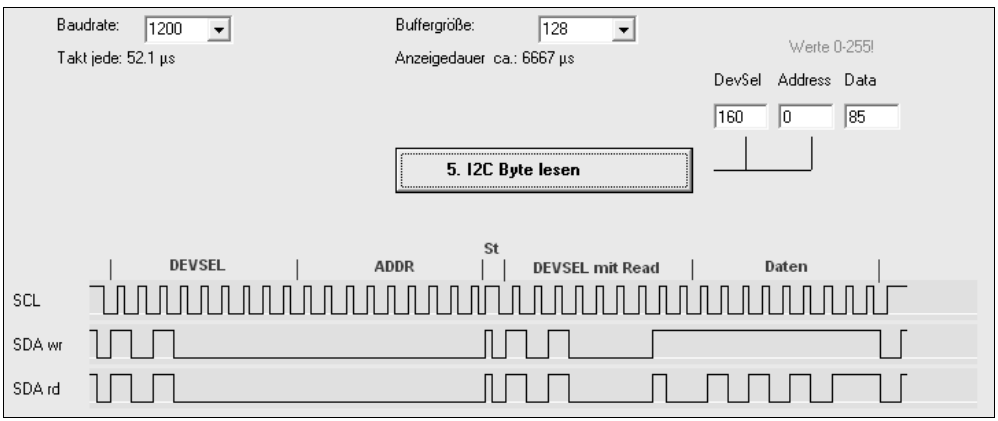


Рис. 13.11

Читаемые данные находятся в конце временной диаграммы. Это значение в десятичном виде представляется в поле **Data** (85_{10} равное 0101 0101_2), которое таким образом соответствует записанному ранее значению.

> ***Указание***
>
> Определять данные на временной диаграмме следует только во время положительного импульса синхронизации SCL, т. е. при высоком логическом его уровне, начиная от положительного его фронта и заканчивая отрицательным.

А теперь рассмотрим листинг 13.3. Чтобы читать данные из запоминающей ячейки микросхемы EEPROM-памяти типа 24C16, все I^2C-функции должны выполняться строго в определенной последовательности друг за другом.

Листинг 13.3

```
Private Sub bt_I2C_Read_Click()
strWriteBuffer = ""
I2CStart
I2CByte Val(Me.tb_devsel.Text)     ' адрес EEPROM-памяти с интерфейсом I2C
I2CByte Val(Me.tb_byte_addr.Text) ' адрес ячейки памяти с данными
I2CStart
I2CByte (Val(Me.tb_devsel.Text) + 1) ' читать байт данных, начиная с LSB
I2CByte (&HFF)                        ' сформировать синхроимпульсы
I2CStop
If FTWriteRead(True) = False Then MsgBox "FTWriteRead Error"
...
End Sub
```

После формирования состояния старта первым выдается адрес EEPROM-памяти. Адрес равен 160_{10}; последним битом 160 (A0H или 1010 0000 в двоичной системе) является 0, который указывает на операцию записи. Какая же ячейка памяти в EEPROM-памяти должна быть прочитана? Предварительно заданный в текстовом поле **Address** и с помощью соответствующей переменной `tb_byte_addr` *(см. шаг 4)* адрес ячейки памяти и равный в данном случае 0 передается следующим байтом.

После этого для выполнения операции чтения EEPROM-памяти формируется новое состояние старта (St) и снова адрес EEPROM-памяти DEVSEL, но теперь с 1 в самом младшем бите (LSB), определяющим операцию чтения. В заключение осталось добавить только лишь синхросигналы и сформировать состояние останова.

Приведенные временные диаграммы получаются из буфера чтения. Имея отдельные байты при выполнении операций записи и чтения и передавая их с определенными задержками во времени, можно получить изображение временных диаграмм в реальном масштабе времени! Таким образом, всего за пять шагов вы не только разработали преобразователь интерфейсов USB-I^2C, но и одновременно с этим запрограммировали его для использования с EEPROM-памятью!

Дальнейшие указания для разработки программного обеспечения

Скорость передачи для всех рассмотренных шагов устанавливалась в поле ввода **Baudrate** неизменно на значение 1200 бод.

Проверьте следующее: удваивается ли действительно скорость передачи при увеличении скорости передачи данных в 2 раза.

С изменением размера буфера в раскрывающемся списке **Buffergröße** со 128 до 256 байтов удваивается ли длительность записи. Всегда используйте максимальное значение размера буфера для получения временной диаграммы.

Изменив адрес устройства в поле **DevSel** на значение отличное от 160, вы, скорее всего, больше не сможете обратиться к подключенной микросхеме EEPROM-памяти. После нажатия кнопки **5. I2C Byte lesen** адрес устройства памяти автоматически увеличивается на 1, так что менять его вручную не нужно. Как обращаться к ячейкам памяти, начиная с адреса 256, вы можете найти в описании использования EEPROM-памяти в примере, размещенном на компакт-диске в каталоге \Beispielprogramme\Bsp_I2C_EEProm\Bsp_Kennwortspeicher.

13.6. "Выуживание" данных

В программе после нажатия кнопки **6 Bytes lesen** (рис. 13.12) считываются шесть последовательных байтов, начиная с введенного адреса. Размер буфера автоматически уставляется на значение 512 байтов.

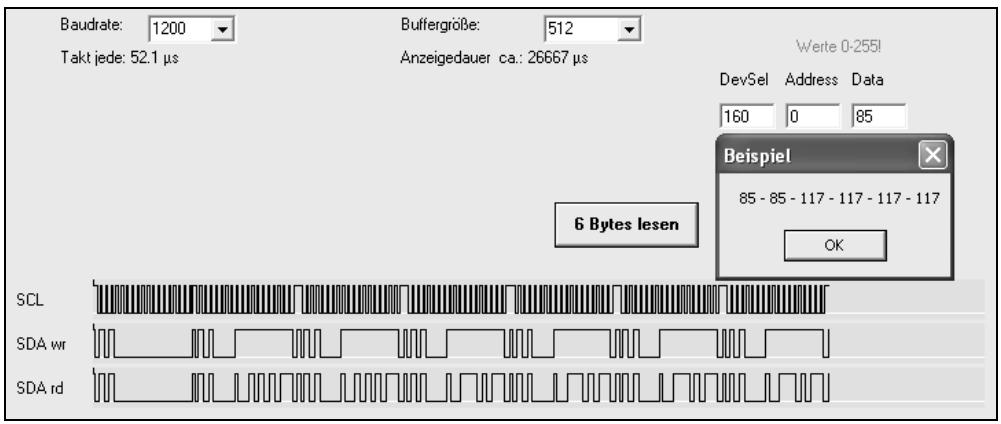


Рис. 13.12

Все I^2C-функции в подпрограмме `bt_read6Bytes_Click` должны выполняться в необходимой последовательности:

```
Private Sub bt_read6Bytes_Click()
...
strWriteBuffer = ""
```

```
I2CStart
I2CByte Val(Me.tb_devsel.Text)           ' адрес EEPROM-памяти
I2CByte Val(Me.tb_byte_addr.Text)        ' адрес ячейки памяти
BMerkmal = Len(strWriteBuffer)           ' задать число байтов для буфера
For i% = 0 To 5
    I2CStart '
    I2CByte (Val(Me.tb_devsel.Text) + 1) ' читать данные
    I2CByte (&HFF)                       ' генерировать синхроимпульсы
    I2CStop
Next i
If FTWriteRead(True) = False Then MsgBox "FTWriteRead Error"
...
End Sub
```

После формирования состояния старта `I2CStart` передаются адрес EEPROM-памяти (DevSel) и первый начальный адрес ячейки памяти. При каждом новом чтении внутренний счетчик адреса автоматически увеличивается на 1 и таким образом определяется следующий адрес ячейки для чтения данных. В цикле шесть раз формируется состояние старта, а в буфер записи `strWriteBuffer` записываются считанные данные, формируются синхроимпульсы и состояние останова.

После записи в буфер записи при помощи функции `FTWriteRead` данные будут помещены в буфер чтения `strReadBuffer`. Последовательные биты должны быть "выловлены" из этого потока данных и преобразованы в байты: результат представляется в окне сообщения, например, в следующем виде:

85 – 85 – 117 – 117 – 117 – 117.

Функция `I2C_ConverterReadByte` выполняет преобразование битов в байты. В качестве параметра вводится та позиция байта, начиная с которой из буфера чтения последовательные биты должны соединяться в байты:

```
Private Function I2C_ConvertReadbyte(ByVal AbBytePosition As Long)
As Byte
.
' последовательные биты снова соединить в байт
' действительные данные получить при положительном фронте сигнала SCL
' сначала взять самый старший бит (MSB), а в конце - младший бит (LSB)
Dim i As Integer
Dim BitPosition As Byte
Dim Data_in As Byte
Dim rb As String
rb = strReadBuffer
data = 0
BitPosition = 128
For i = 1 To 25 Step 3 ' до позиции 2 байта
        ' получить байт в переменной Data_in
        Data_in = Asc(Mid$(rb, i + AbBytePosition, 1))
```

```
        ' сигнал SDA низкого или высокого логического уровня?
        If Data_in And (2 ^ SDAREAD) Then
           data = data Or BitPosition
        End If
        BitPosition = BitPosition / 2
Next i
I2C_ConvertReadbyte = data
End Function
```

Сначала обрабатывается самый старший бит (MSB), в конце самый младший (LSB). Эта функция похожа на преобразование байтов в последовательные биты, но отличается величиной шага цикла равного в данном случае 3 (`For i = to 25 Step 3`). Необходимые тактовые сигналы SCL отнимают 3 байта, поэтому обработка и анализ происходят каждый раз с интервалом в 3 байта. Переменная `SDAREAD` при пуске программы была определена для сигнальной линии RTS (соответствующей чтению сигнала CDA (CDA rd)). По окончании работы функции результат сохраняется в переменной `data`.

Параметр `AbBytePosition` должен быть определен в буфере чтения для вызова функции `I2C_ConvertReadbyte`. Перед записью переменной `BMerkmal` задается значение, соответствующее исходной позиции `BMerkmal = Len(strWriteBuffer)`. От этой позиции отдельные биты читаемых байтов встречаются с определенным интервалом:

```
Private Sub bt_read6Bytes_Click()
.
strWriteBuffer = ""
I2CStart
I2CByte Val(Me.tb_devsel.Text)           ' адрес EEPROM-памяти
I2CByte Val(Me.tb_byte_addr.Text)        ' адрес ячейки памяти
BMerkmal = Len(strWriteBuffer)           ' исходная позиция
For i% = 0 To 5
    I2CStart                             ' 3 байта
    I2CByte (Val(Me.tb_devsel.Text) + 1) ' 27 байтов
    I2CByte (&HFF)                       ' 27 байтов
    I2CStop                              ' 4 байта
Next i
If FTWriteRead(True) = False Then MsgBox "FTWriteRead Error"
BMerkmal = BMerkmal + 3 + 27
Ausgabestring = Val(I2C_ConvertReadbyte(BMerkmal))
For i = 1 To 5
   BMerkmal = BMerkmal + 61
   Ausgabestring = Ausgabestring & " – " &
                                   Val(I2C_ConvertReadbyte(BMerkmal))
Next i
MsgBox Ausgabestring
.
End Sub
```

Обратите внимание, что для переменной `I2CStart` требуется 3 байта, переменной `I2CByte` — 27 байтов. Первое значение начинается после 30 байтов, другие значения следуют друг за другом через 61:

`I2CStart` + 2 × `I2CByte` + `I2CStop` = 3 + 54 + 4 (байта),

что в сумме составляют 61 байт.

Строка вывода для окна сообщения заполняется первым байтом, начиная с положения `BMerkmal` + 30, и последующими байтами в цикле с позиций `BMerkmal` + 61.

13.7. Эксплуатация программы EEPROM-накопителя условного кода

Емкость микросхемы EEPROM-памяти типа 24C16 составляет 16 Кбит или соответственно 2048 байтов. В демонстрационной программе, приведенной на рис. 13.13, используются только первые 4×128 = 512 байт, т. е. 25 % имеющейся в распоряжении емкости памяти. При этом программа позволяет записать и прочитать 4 строки по 128 символов любого условного кода, пароля или идентификатора.

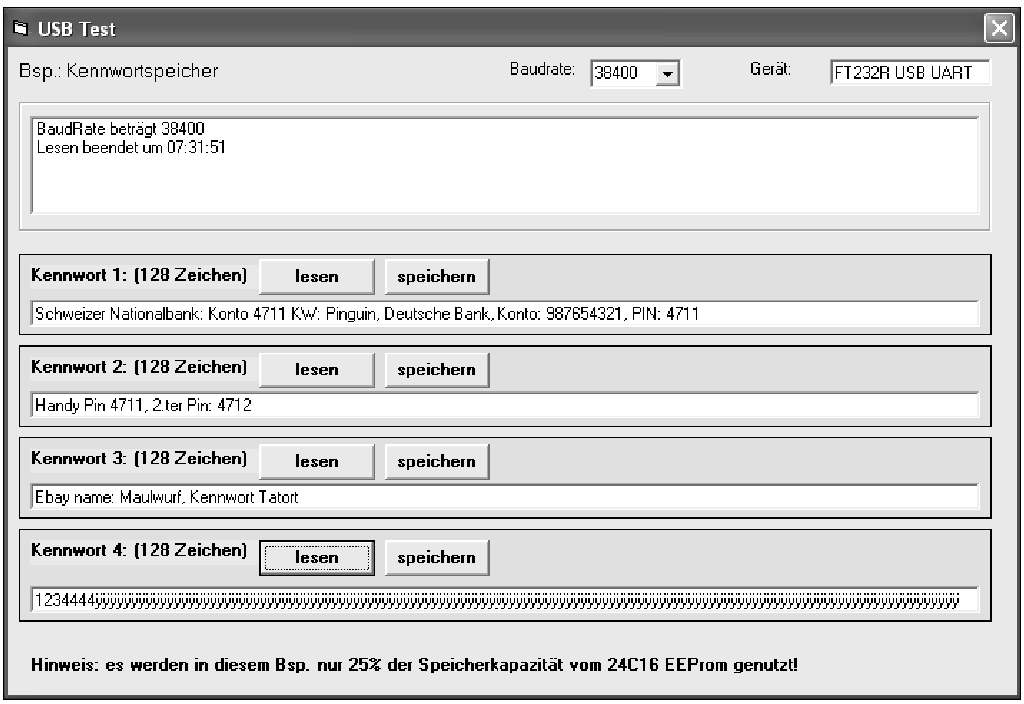


Рис. 13.13

Если вы в первый раз читаете данные из "пустого" EEPROM, то наверняка обратите внимание на странные символы (ÿ), такие как показаны на рисунке в 4-й строке символов. Это объясняется тем, что при стирании EEPROM-памяти, в нее записы-

ваются символы с кодом FF_{16} или 255_{10} и при чтении отображаются служебными знаками.

Чтение EEPROM-памяти выполняется достаточно быстро, а вот при записи данных возможна потеря скорости. Причину вы узнаете в описании исходного текста программы. Протестируйте программу на скорости передачи в 1200 бод, выбрав ее в выпадающем списке **Baudrate** — в этом случае даже при чтении вы также заметите замедление работы.

13.8. Программа EEPROM-накопителя условного кода — фрагменты исходного текста

После формирования условия старта по интерфейсу I^2C в микросхему EEPROM-накопителя передается байт выбора устройства (Device Select Byte) или, иначе, байт адреса DEVSEL, формат которого приведен в табл. 13.1. Биты B7 по B4 адреса DEVSEL для микросхем ST24Cxx заданы значением 1010. Биты с B3 по B1 в DEVSEL — это старшие биты адреса ячейки памяти A10, A9 и A8. Как обычно в интерфейсе I^2C младший бит байта выбора устройства (B0 = R/$\overline{W}$ в DEVSEL) определяет выполняемую операцию запись или чтение.

В табл. 13.2 — оставшиеся младшие биты байта адреса ячейки памяти с 0 до 255_{10}. При обращении к запоминающей ячейке 261_{10} (100000101 в двоичной системе) значение адреса, определяемое битами от A10 до A0, должно быть разделено на два байта (две части адреса). Далее приводится соответствующий фрагмент текста программы:

```
...
DEVSEL = 160                                ' адрес устройства EEPROM-памяти
DEVSEL = DEVSEL + (Int(Start / 256) * 2) ' биты A10-A8 адреса ячейки
' передача I2C-команд
I2CStart
I2CByte Val(DEVSEL)
I2CByte Val(Start And &HFF)            ' младший байт адреса ячейки памяти
```

Таблица 13.1

Микросхема	Биты байта выбора устройства DEVSEL							
	B7	B6	B5	B4	B3	B2	B1	B0
24C16	1	0	1	0	A10	A9	A8	R/$\overline{W}$

Таблица 13.2

Микросхема	Младшая часть адреса ячейки памяти							
	B7	B6	B5	B4	B3	B2	B1	B0
24C16	A7	A6	A5	A4	A3	A2	A1	A0

Сначала для байта выбора устройства DEVSEL задается опорное значение, равное 160_{10} (10100000_2). `Start` — это переменная, соответствующая желаемому адресу запоминающей ячейки памяти, в данном случае это число 261_{10} (100000101_2). Выполнением операции `Start/256` отделяются младшие 8 битов от адреса ячейки памяти, и получаем старшую часть адреса, в данном примере — 1. Эта логическая единица не должна соответствовать младшему биту B0 регистра выбора устройства DEVSEL, поскольку в нем хранится бит, определяющий операцию чтения или записи — R/$\overline{W}$-бит. Поэтому этот бит нужно сдвинуть на одну позицию влево, к позиции бита B1. Для этого полученный результат умножается на 2, а затем складывается с числом 160_{10}.

Остальные восемь битов для байта младшей части адреса от 0 до 255 получают при помощи маски, накладываемой на переменную `Start`, т. е. при помощи логической операции AND с числом FF в шестнадцатеричной системе счисления.

Для чтения 128 байтов происходит в вызове функции `read128Bytes`:

```
Private function read128Bytes(Start As Integer)
. Auszug:
' передача I2C-команд
I2CStart
I2CByte Val(DEVSEL)
I2CByte Val(Start And &HFF)        ' младший байт адреса ячейки памяти
BMerkmal = Len(strWriteBuffer)     ' исходная позиция
For i% = 0 To 127                  ' 128 байт
    I2CStart
    I2CByte (DEVSEL + 1)           ' задать чтение (LSB=1 в DEVSEL)
    I2CByte (&HFF)                 ' генерация синхроимпульсов для чтения
    I2CStop
Next i
I2CByte (&HFF)                     ' еще генерация синхроимпульсов
SendRead_FTDI_Bytes
```

Эта функция вам уже известна из примера *разд. 13.6* и здесь приспособлена для 128 байтов. В функции `writebytes` записываются 128 байтов. Каждый байт передается в программном цикле по отдельности:

```
Private function writebytes(Start As Integer)
. Auszug:
'
For i = 0 To Len(wrtxt) — 1
   strWriteBuffer = ""
   I2CStart
   I2CByte Val(DEVSEL)
   I2CByte Val(Start + i)             ' адрес Start = 160 + i
   I2CByte Asc(Mid(wrtxt, i + 1, 1)) ' каждый байт в wrtxt
   I2CStop
```

```
   I2CByte (&HFF)                    ' генерация синхроимпульсов
   SendRead_FTDI_Bytes
Next i
```

Перед этим данные подготавливаются в строке данных, для того чтобы удалить служебные знаки (не буквы и не цифры) предварительно очищенной EEPROM-памяти. В цикле каждый отдельный байт передается по интерфейсу USB и помещается в строковую переменную `wrtxt`. В данном случае не осуществляется проверка успешности передачи данных с помощью бита подтверждения.

Поскольку побайтный процесс записи приводит к потере скорости, то ее можно было бы улучшить благодаря постраничной записи (Page Write) и проверке подтверждающего бита для микросхемы EEPROM-памяти типа 24C16.

Длительность цикла записи, согласно техническому паспорту, составляет максимально 10 мс. Для оптимизации возможно также установить время цикла записи в зависимости от скорости передачи в бодах и после записи многих байтов сравнить данные после выполнения проверочной операции чтения.

13.9. Пример программы двухпроводной связи по интерфейсу I^2C

Этот пример программы показывает, как использовать и работать с данными и синхроимпульсами двухпроводной связи интерфейса I^2C.

Сигнальная линия RXD остается в качестве тактовой шины SCL интерфейса I^2C, а линия RTS предназначается для шины данных SDA. Кроме того, линия DCD с помощью программы включается в качестве входа и соединяется с внутренним резистором микросхемы FT232R, имеющим сопротивление 200 кОм, а другим своим выводом подключенного к положительному напряжению питания.

Поскольку обработка сигналов уже была разобрана в предыдущем примере, в данном случае остановимся лишь на некоторых особенностях демонстрационного программного обеспечения для операций записи и чтения.

Отсоедините USB-адаптер от компьютера, а также дополнительную плату от адаптера (рис. 13.14).

Вставьте микросхему 24C16 EEPROM-памяти в панельку на дополнительной плате. Вам не нужно вставлять изображенный диод. Соедините дополнительную плату с USB-адаптером и только после этого подключите USB-адаптер к вашему компьютеру.

Соответствующую программу вы найдете на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_I2C_EEPROM\zweidraht. Запустите программу I2C_2Drahtverbindung.exe и щелкните кнопку **4. I2C Byte schreiben** (запись байта) (рис. 13.15).

В результате вы увидите процесс записи байта данных, отображенный в нижней части диалогового окна программы в виде временных диаграмм, для сигналов SCL

и SDA. Поскольку линия SDA в данном случае используется лишь в качестве выхода, то остается только надеяться, что EEPROM-память безошибочно получила переданные ей данные.

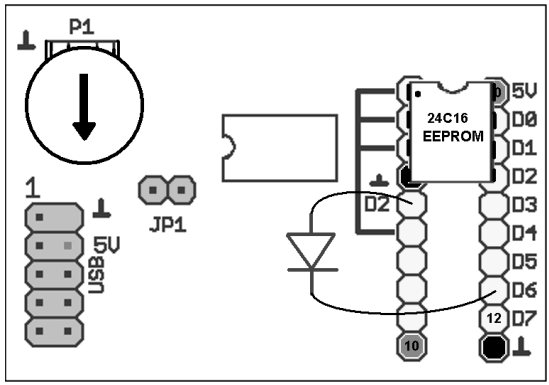


Рис. 13.14

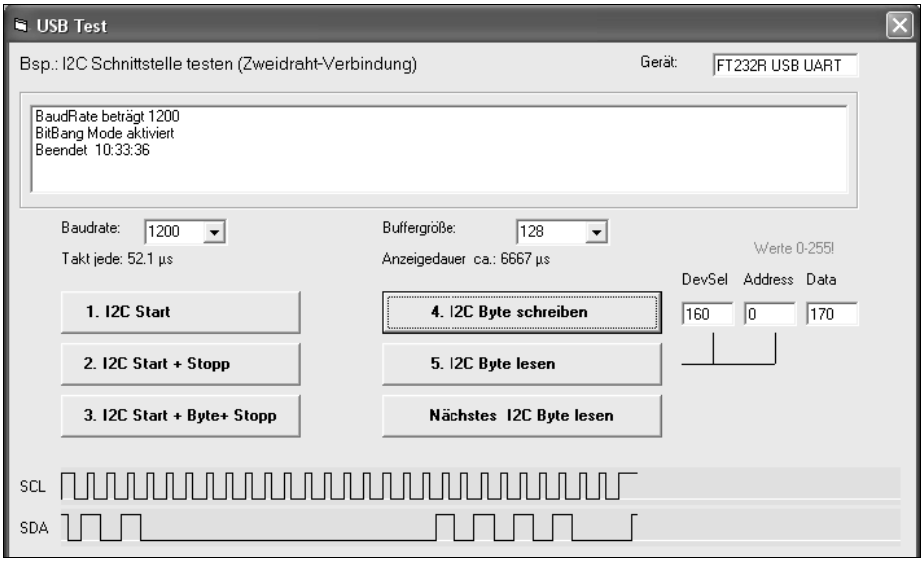


Рис. 13.15

Ранее записанные данные можно, конечно, проверить и при помощи операции чтения, выполняемой после нажатия кнопки **5. I2C Byte lesen** (чтение байта): во временных диаграммах в данном случае представляются только прочитанные данные (рис. 13.16).

С помощью функции `FTWriteReadAnalyze` (в VB-программе) сигнальная линия SDA может переключаться с выхода на вход и обратно. Так при передаче адреса EEPROM-памяти линия SDA включается в качестве выхода. Затем при операции чтения линия SDA снова переключается на вход.

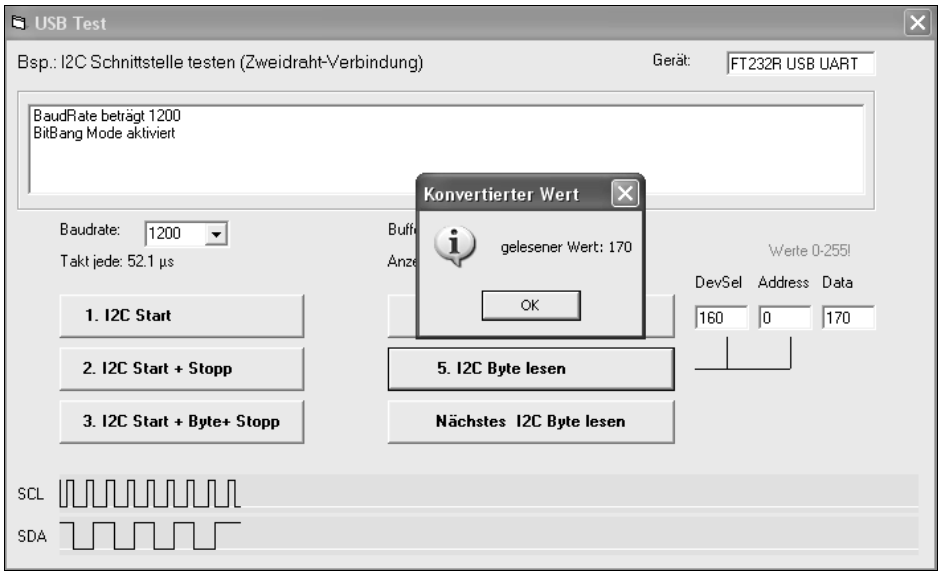


Рис. 13.16

Само собой, можно было бы сохранить эти отдельные шаги и при необходимости снова присоединить таким образом полученные данные к временной диаграмме, но тогда затруднялось бы ее дальнейшее исследование.

С помощью функции `I2C_ConvertReadByte` вычисляется числовое значение прочитанных данных, которые в диалоговом окне представляются в виде временной диаграммы и числового значения, которое в данном примере равно 170.

ГЛАВА 14

Инфракрасное дистанционное управление

Инфракрасные передачи данных используются, например, в мобильном телефоне или пульте дистанционного управления телевизора, CD/DVD-плеера и т. д.

Обычно на стороне передатчика используются инфракрасные передающие диоды, а на стороне приемника — инфракрасные приемные диоды. Спектральная чувствительность фотодиодов зависит от используемого полупроводникового материала. На рис. 14.1 показан пример для элементов из германия Ge (germanium) и кремния Si (silicium). Инфракрасные передачи функционируют в диапазоне длин волн от 850 до 950 мкм и являются невидимыми для человеческого глаза. Для предотвращения реагирования фотодиода на обычный дневной свет применяют спектральный фильтр, который устанавливается перед приемником и пропускает из общего спектра света только инфракрасный свет.

Рис. 14.1

В отличие от фотодиода светочувствительный фоторезистор LDR (Light Dependent Resistor) является очень инертным и не подходит для достаточно высокоскоростных сигналов инфракрасной передачи данных.

Фотодиоды эксплуатируют в обратном направлении. В темноте фотодиод имеет высокое внутреннее сопротивление. С повышением освещенности сопротивление уменьшается, а обратный ток увеличивается. Из-за линейной зависимости между обратным током и освещенностью фотодиодов они часто используются для измерительных целей.

Фотодиоды или более чувствительные фототранзисторы, кроме того, применяются в оптопарах (оптронах) для гальванической развязки сигналов в электрических цепях, в устройствах ввода данных с перфокарт, в оптических приводах компакт-дисков и DVD, в фотоячейках, например, для счета деталей, при бесконтактных измерениях температуры или в схемах ведомых фотовспышек.

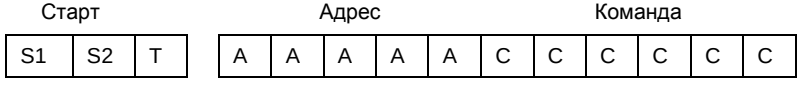
Фотодиоды имеют очень малое время реакции и переключения (время срабатывания). Время срабатывания фототранзистора BPW-40 составляет 3 мкс. Время срабатывания фотодиода BPW-34 равно 20 нс, а фотодиода SFH 2400 — даже 5 нс! Чем меньше время реакции фотодиода, тем быстрее может выполняться передача данных.

Достаточно часто применяются микросхемы инфракрасных приемников SFH 506-36 или TSOP 1736, которые имеют фотодиод со встроенным усилителем и в то же время одновременно демодулятор, позволяющий сразу же в микросхеме преобразовать световой сигнал в цифровое значение.

14.1. Инфракрасная передача данных по протоколу RC5

Инфракрасная передача данных является последовательной передачей данных. Однако поскольку человеческий глаз не может воспринимать свет в инфракрасном диапазоне, то очень трудно без каких-либо вспомогательных средств перепроверить работоспособность дистанционного управления или инфракрасной передачи данных.

Одним из немногих стандартизованных методов передачи при дистанционном управлении является передача с помощью разработанного компанией Philips протокола RC5. Согласно протоколу RC5 при каждом нажатии кнопки передается пакет данных, состоящий из 14 битов. При этом первые два бита — это стартовые биты, которым соответствует передача двух логических единиц — "1". Затем следует бит T (toggle bit), 5 битов для адреса устройства и 6 битов команды:



Протокол RC5 использует бифазное (манчестерское) кодирование информационных битов. При этом каждый отдельный бит состоит из двух частей — паузы, равной 900 мкс, и фазы длительностью около 900 мкс, состоящей из синхросигналов инфракрасного света частотой 36 кГц. Такой способ передачи позволяет приемнику лучше отличить передаваемые данные от света окружающей среды.

Биту со значением логической "1" соответствует вначале формирование паузы и потом генерирование синхросигналов, а для бита со значением логического "0" все обстоит противоположным образом, сначала осуществляется генерирование синхросигналов, а затем следует пауза (рис. 14.2).

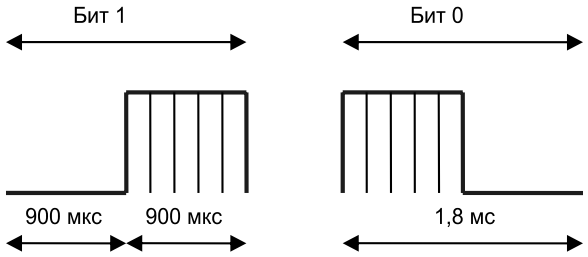


Рис. 14.2

Передача одного бита всегда продолжается ровно 1,8 мс, т. е. фаза генерирования синхросигналов и пауза длятся каждая по 900 мс. Благодаря этому передача пакета данных, состоящего из 14 битов, завершится через 25 мс. Если на пульте дистанционного управления держать кнопку в нажатом состоянии, то передаваемое слово данных будет повторяться каждые 114 мс.

Описываемая здесь временная характеристика соответствует коду протокола RC5. В зависимости от изготовителя протокол инфракрасной передачи может отличаться. При инфракрасной передаче данных в мобильных телефонах используются передача по стандарту SIR (Serial Infrared, последовательная инфракрасная передача) со скоростью 115 Кбит/с или FIR (Fast Infrared, быстрая инфракрасная передача) со скоростью от 567 Кбит/с до 4 Мбит/с. Универсальный стандарт IrDA (Infrared Data Association) определяет и описывает спецификации и протоколы инфракрасной передачи данных.

Способ передачи SIR для экономии энергии характеризуется очень короткими импульсами, около 1,6 мкс. Если увеличить длительность этих импульсов, передаваемые данные также могли бы обрабатываться микросхемой FT232R. Время реакции фототранзистора BPW-40 составляет около 3 мкс, и в данном случае он не подходит.

BPW-40, иногда называемый фотодиодом, на самом деле является фототранзистором. Поскольку не используется какой-либо дополнительный инфракрасный фильтр, то фотоэлемент будет реагировать и на дневной свет.

В данном эксперименте нужно не только определить, включен ли инфракрасный сигнал дистанционного управления. Эту проблему можно было бы решить, в случае необходимости, с помощью схемы усилителя со светодиодным индикатором или с зуммером. Здесь следует также контролировать, происходит ли изменение цифрового сигнала с 0 на 1, и наоборот, что невозможно сделать, имея только простую схему усилителя.

14.2. Пример программы тестирования инфракрасного дистанционного управления

Если устройство больше не реагирует на пульт дистанционного управления, то зачастую причиной являются плохие батареи питания. Если после смены батарей дистанционное управление все еще не функционирует, то возникает вопрос, явля-

ется ли причиной сам пульт дистанционного управления или же виноват приемник. При помощи USB-адаптера и фотодиода вы можете протестировать работоспособность пульта дистанционного управления.

Выполните ряд следующих действий для подготовки к тестированию при помощи программы, расположенной на компакт-диске в каталоге \Beispielprogramme\Bsp_16, USB-адаптера и фототранзистора BPW-40:

1. Сделайте тонкую картонную трубочку — для этого подойдут карандаш и липкая лента — и наденьте ее сверху на фототранзистор. Трубка должна ограничивать падение света на фототранзистор, т. к. в ином случае дневной свет сразу же активизирует на нем сигнал CTS (D3).
2. Удалите USB-адаптер от компьютера и подключите к нему дополнительную плату (рис. 14.3).

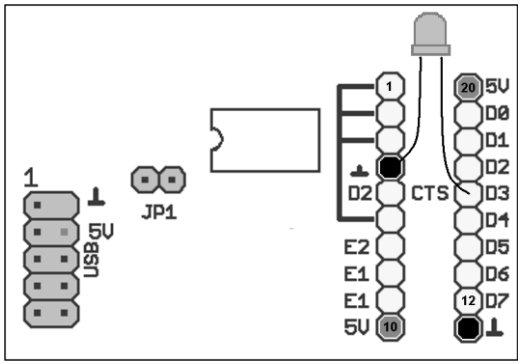


Рис. 14.3

3. Подсоедините фототранзистор BPW-40 к выводу 4 (земля = GND) и выводу 16 (CTS) 20-контактной панельки, расположенной на дополнительной плате. Анод фотоэлемента обычно длиннее, и он должен быть подключен к выводу 16 (CTS = D3).
4. Снова подключите USB-адаптер с дополнительной платой и запустите программу beispiel_16.exe, расположенную на прилагаемом к книге компакт-диске в каталоге \Beispielprogramme\Bsp_16.

После этого нажмите кнопку **BitBang an** (включить режим Bit Bang), чтобы активировать режим Bit Bang, а затем кнопку **Lesen mit Timer** (считывать с таймером) для контролирования состояния линий (рис. 14.4). Для последующей активации сигнала D3 (CTS), как и других сигналов, фототранзистор изначально должен быть затемнен. Если держать включенную настольную лампу по направлению к фототранзистору, то состояние сигнала изменится. Направляя пульт на фототранзистор и нажимая кнопку пульта дистанционного управления, вы, возможно, заметите кратковременные изменения состояния — пульт дистанционного управления посылает ИК-сигналы.

Сигнал D3 можно анализировать в режиме Bit Bang микросхемы FT232R. Частоту опроса можно настроить указанием скорости передачи в бодах (**Baudrate**), а дли-

тельность считывания данных — указанием размера буфера (**Buffergröße**). Если вы изменяете одно из этих значений, программа автоматически определяет длительность опроса и длительность регистрации.

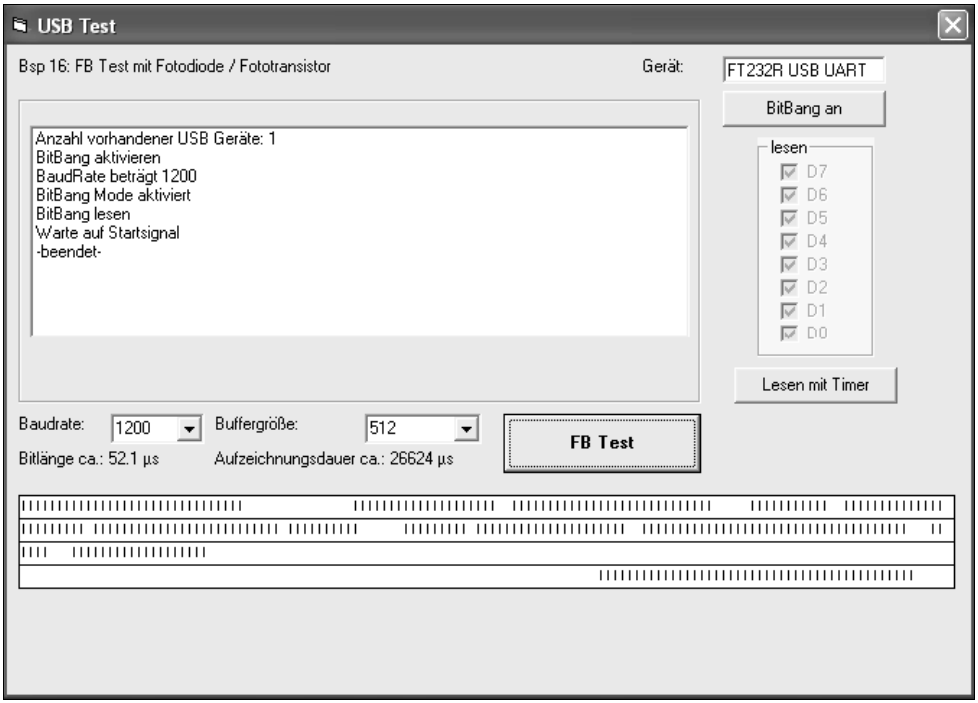


Рис. 14.4

С частотой кратной 16-ти значениям настроенной скорости передачи в бодах можно опрашивать сигнальную линию CTS (D3), начиная с сигнала "старт" до тех пор, пока не будет заполнена настроенная емкость буфера. Результат изображается построчно в нижней части диалогового окна программы. Каждая строка показывает 128 байт информации.

Прежде чем нажмете кнопку **FB Test**, убедитесь, что сигнальная линия D3 активизирована (установлен соответствующий флажок, как и для других сигналов). Запись должна начинаться только тогда, когда будет изменяться состояние сигнала на линии D3. В противном случае вы должны предварительно затемнить фототранзистор. После этого быстро нажмите клавишу вашего пульта дистанционного управления (конечно, держа по направлению к фототранзистору), и тогда вы получите первые результаты, возможно, схожие с результатами, показанными на рис. 14.4. В этом случае можно оценить функциональное состояние пульта дистанционного управления.

Примечание

В данном примере испытывался пульт дистанционного управления японского производства, который использует сигналы, отличные от протокола RC5!

Графическое отображение результатов занимает максимум 8 строк по 128 байтов и определяется настройками размера буфера. Первая строка содержит первые 128 байтов, вторая строка — следующие 128 байтов. Для отображения более чем одной строки вы можете мысленно добавить каждую последующую строку в область для отображения. При каждом нажатии одной и той же кнопки на пульте дистанционного управления вы будете видеть схожие результаты.

Частота опросов сигналов в данном примере (при скорости передачи 1200 бод) будет примерно 19200 Гц (период равен 52 мкс), а длительность записи при размере буфера 512 байтов составляет 26 мс. Тактовая частота исследуемого пульта дистанционного управления, равная 36 кГц (с периодом около 27,7 мкс), во время фазы передачи инфракрасного света содержит 32 световые вспышки длительностью свечения около 6,9 мкс, с паузами между вспышками около 20,8 мкс. Таким образом, в общей сложности длительность фазы свечения составляет $32 \times (6{,}9 + 20{,}8) = 886{,}4 \approx 900$ (мкс). Частота отсчетов с периодом 52 мкс в этом примере для периода световых вспышек с длительностью отдельной вспышки 7 мкс не совсем подходит и приводит к ошибкам (пропускам световых вспышек). В данном случае более подходящей частотой отсчетов была бы частота, соответствующая, например, скорости передачи 38400 бод с периодом равным примерно 1,6 мкс. Однако поскольку фаза паузы, в которой, согласно протоколу RC5, нет передач вспышек света, длится 900 мкс, что существенно больше периода опроса, а поэтому анализирование сигналов при скорости передачи 1200 бод в данном случае может вполне устроить. При размере буфера 512 байтов длительность записи составляет около 26 мс, что соответствует времени, за которое один пакет данных (14 битов) передается по протоколу RC5. Если повысить продолжительность записи и уменьшить скорость передачи в бодах, то примерно через 120 мс произойдет повторная отправка.

Более детальный анализ при длительных записях очень критичен по времени, т. к. микросхема TF232R имеет буфер от 128 до 256 байтов. Кроме этого, в программе после опроса сигнальной линии D3 выполняется операция чтения по USB. При этом также нет точно определенной временной задержки. Поэтому при повторе чтения могут произойти дополнительные искажения результатов.

Достигнуть более высокой точности можно было бы, применив элементы временной задержки (RC-цепи) для тактового сигнала 36 кГц в фазе передачи или выполнив более детальный анализ в программном обеспечении. В данном случае проще всего было бы сразу применить инфракрасные приемники SFH 506-36 или TSOP 1736, кратко описанные в начале главы.

14.3. Исходный код программы тестирования инфракрасного дистанционного управления

Исходный код для примера содержится в файле Bsp.VBP, расположенном на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_16. Следующий фрагмент из исходного кода показывает важнейшие вызовы функций для тестирования дистанционного управления при нажатой кнопке **FB Test**:

```
Private Sub bt_fb_test_Click()
' включить режим Bit Bang с заданной скоростью
Call bt_BitBangAn_Click
..
' D3 деактивирован? тогда старт чтения
.
Do
   FtStatus = FT_GetBitMode(lngHandle, Databyte)
   If FtStatus <> FT_OK Then
      LoggerList.AddItem "Fehler bei FT_GetBitMode"
      GoTo CloseHandle
   End If
Loop Until ((Databyte And 2 ^ 3) / 2 ^ 3) = 0
.
SendRead_FTDI_Bytes
.
End Sub
```

Команда `Call bt_BitBangAn_Click` вызывает функцию `bt_BitBangAn_Click` и активирует у микросхемы FT232R режим Bit Bang с заданной скоростью передачи данных. Вызов функции `FT_GetBitMode` выполняется в цикле до тех пор, пока бит данных D3 (CTS на выводе 16 20-контактной панельки дополнительной платы) не примет значение логического "0". Бит данных D3 перейдет в "0", как только будет нажата какая-либо кнопка пульта дистанционного управления.

Исследование продолжается вызовом функции `SendRead_FTDI_Bytes`. Вызовы отдельных функций нам уже знакомы из *главы 9*, а теперь лишь были соединены в одной собственной функции. Данные в синхронном режиме Bit Bang выдаются и помещаются в переменную `strWriteBuffer`, и могут быть считаны при обращении к переменной `strReadBuffer`.

Данные после вызова функции `bt_fb_test_Click` помещаются в строковую переменную `strWriteBuffer` в соответствии с установленным размером буфера.

Данные для графического отображения обрабатываются в подпрограмме, представленной в листинге 14.1.

Листинг 14.1

```
strWriteBuffer = ""
   For il% = 1 To Len(strReadBuffer)
      If ((Asc(Mid(strReadBuffer, il, 1)) And 2 ^ 3) / 2 ^ 3) = 0 Then
           strWriteBuffer = strWriteBuffer & "|"
      Else
           strWriteBuffer = strWriteBuffer & " "
      End If
   Next il
```

```
' максимум 128 знаков в результате
Dim i As Integer
Me.lb_erg0.Caption = Mid(strWriteBuffer, 1, 128)
' предварительно все остальные
For i = 1 To 7
   Me("lb_erg" & Trim(Str(i))).Visible = False
Next i
If Val(lb_bsize.Text) > 128 Then
   iz = Val(lb_bsize.Text) / 128
   For i = 1 To iz — 1
    Me("lb_erg" & Trim(Str(i))).Caption = Mid(strWriteBuffer, i 128, 128)
    Me("lb_erg" & Trim(Str(i))).Visible = True
   Next i
End If
```

Данные из строковой переменной `strWriteBuffer`, в зависимости от состояния сигнальной линии D3 каждого принятого байта, записываются в переменную `strReadBuffer` значением, которое соответствует вертикальному символу "|" или пробелу. Символом "|" обозначается инфракрасный световой сигнал, когда фототранзистор в этот момент времени устанавливает входной сигнал на низкий логический уровень. Таким образом в нижней части диалогового окна отображается своеобразная временная диаграмма. Содержимое строковой переменной `strWriteBuffer` после этого с помощью VB-функции `Mid` разделяется максимум на 8 строк по 128 байтов каждая, с метками от `lb_erg0` до `lb_erg7`.

14.4. Управление дополнительной ведомой вспышкой при помощи фотодиода

Небольшой пример электрической схемы ведомой фотовспышки для радиолюбителя представлен на сайте **www.heise.de** и рис. 14.5.

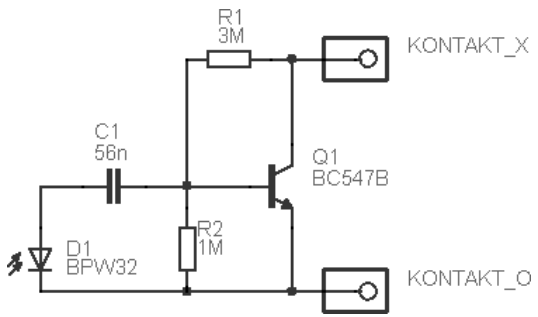


Рис. 14.5

Контакты для внешней фотовспышки предлагают только дорогие цифровые камеры. По случаю можно найти те или иные бракованные фотовспышки. При помощи фотодиода и с учетом небольших затрат на схему вы можете зажечь фотовспышку

без проводов. Осторожно, существуют старые вспышки, которые на контактах имеют до 150 В, для таких устройств эта схема, конечно же, непригодна.

При помощи короткого светового импульса (вспышки) открывается транзистор Q1 устройства и тем самым зажигает фотовспышку. Конденсатор C1 предназначен для того, чтобы вспышка реагировала не на постоянный свет, а только на световой импульс. Если вы желаете, чтобы вспышка вашей цифровой камеры работала постоянно, то нужно обратиться к примеру программы в *разд. 6.4*.

14.5. Обработка сигналов с представлением результата в виде временной диаграммы

В качестве альтернативы для представления описанной ранее построчной обработки сигналов М. Кушель разработал небольшую подпрограмму на Visual Basic. Файл находится на компакт-диске в каталоге с примерами Bsp_16.

Результаты изображаются в нижней части диалогового окна в виде временной диаграммы (рис. 14.6). Каждый раз изображается общая продолжительность записи, чтобы получить максимальное графическое разрешение. Пример иллюстрирует развертку сигналов дистанционного управления с частотой отсчетов 38400 Гц (с периодом 26 мкс) и продолжительностью записи 13,3 мс, при размере буфера 512 байтов.

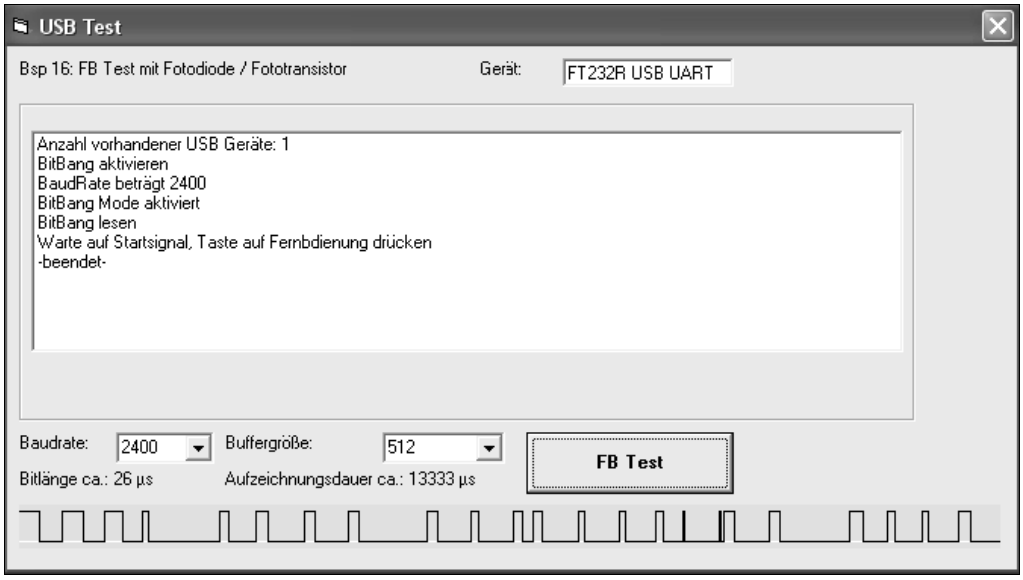


Рис. 14.6

ГЛАВА 15

Анализатор для цифровых сигналов с частотами до 60 кГц

Программа из предыдущего примера может подойти в качестве простого "одноканального анализатора" цифровых сигналов с частотами до 60 кГц.

При тестировании пульта дистанционного управления одновременно могут записываться все восемь сигнальных входов (D0—D7). Результаты, таким образом, уже будут находиться в памяти вашего компьютера, а именно в переменной `strReadBuffer`, и биты каждой сигнальной линии для графического отображения должны быть предварительно подготовлены! В предыдущем примере тестирования пульта дистанционного управления обрабатывалась лишь одна сигнальная линия.

В программе тестирования дистанционного управления можно осуществить ожидание события стартового сигнала для начала записи, а в данном случае для начала запуска логического анализатора. Запуск можно было бы, конечно, организовать при самых различных состояниях всех сигнальных линий. Скорее всего при запуске программного обеспечения может присутствовать небольшая задержка, которая будет зависеть от быстродействия компьютера.

Если у вас есть генератор частот или тактовый генератор, то подключите его к дополнительной плате аналогично подключению фототранзистора в предыдущем примере, а затем самостоятельно выполните тестирование.

> ***Указание***
>
> Вы должны учитывать, что скорость передачи по USB при полноскоростном режиме передачи (Full-Speed) составляет только 64 байта/кадр. Однако для достаточно простых исследований этот анализатор будет очень полезен.

Далее представлены некоторые примеры результатов исследования сигналов. На рис. 15.1 показан пример для сигнала с частотой около 60 кГц. Продолжительность записи в данном случае установлена на значение 104 мкс при заданном размере буфера, равном 128 байтов. На рис. 15.2 приведен пример для входного сигнала частотой около 10 кГц, продолжительность записи установлена на 104 мкс при размере буфера 128 байт. На рис. 15.3 показан пример исследования сигнала с такой же частотой 10 кГц, но при задании размера буфера 256 байтов и соответственно продолжительности записи 208 мкс.

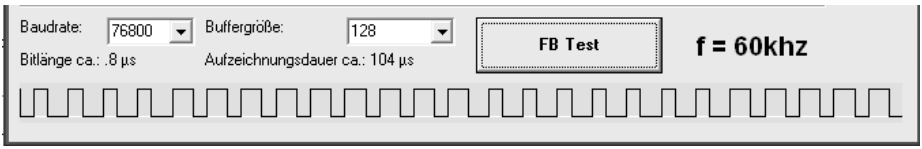


Рис. 15.1

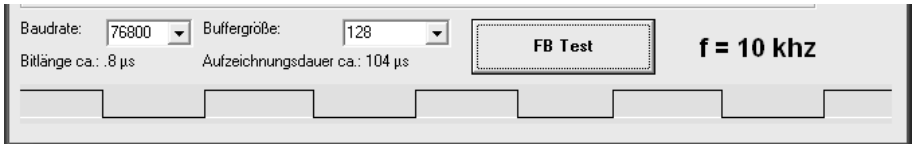


Рис. 15.2

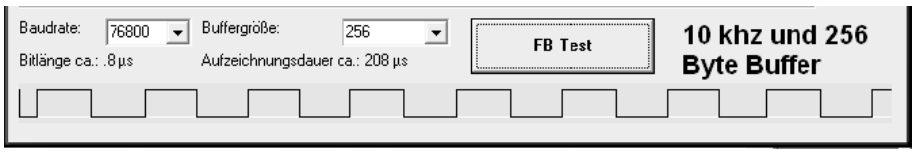


Рис. 15.3

Пожалуйста, не связывайте полученное графическое представление результата в третьем примере (см. рис. 15.3) с двукратным увеличением частоты исследуемого сигнала по сравнению с предыдущим примером (см. рис. 15.2). В обоих примерах частота входных сигналов одинакова и составляет 10 кГц. При изменении размера буфера со 128 до 256 байтов удваивается длительность записи. Таким образом, временная диаграмма на рис. 15.2 продолжается 104 мкс, а на рис. 15.3 — 208 мкс! Поэтому-то изображение, полученное на последнем рисунке при той же частоте входного сигнала, выглядит соответственно более сжатым.

ГЛАВА 16

8-канальный логический анализатор

Данный логический анализатор — это расширение предыдущего примера на восемь каналов. Правда, он не совсем совершенен, зато бесплатный и вполне выполняет свою роль. Можно перечислить следующие недостатки:

1. Запуск (анализирование входных сигналов и ожидание стартового сигнала для начала записи) осуществляется с помощью программного обеспечения компьютера и в зависимости от быстродействия компьютера вызывает различные временные задержки.
2. Не может быть гарантирована постоянная скорость записи данных, поскольку используются поточные USB-передачи данных типа Bulk, с максимально возможным размером пакета 64 байт при полной скорости USB-передачи (Full-Speed).

Однако, несмотря на эти недостатки, если вы используете сигнальную линию в качестве тактового выхода, то это вполне хороший анализатор для исследования цифровых схем.

Программа находится на прилагаемом к книге компакт-диске в каталоге \Beispielprogramme\Bsp_LA_8Kanal. Скорость передачи данных в программе (рис. 16.1), задаваемая с помощью раскрывающегося списка **Baudrate**, определяет частоту отсчетов (частоту дискретизации) цифровых сигналов, а размер буфера с помощью раскрывающегося списка **Buffergröße** устанавливает продолжительность записи. На рисунке показан пример исследования 8-разрядного счетчика при использовании логического анализатора с буфером на 256 байт.

На следующем рис. 16.2 показан пример исследования того же 8-разрядного счетчика при неизменной скорости передачи, но при размере буфера 512 байт. По сравнению с предыдущим случаем с 256-байтовым размером буфера в данном случае продолжительность записи удваивается.

В программе можно для каждого сигнального канала задать 0 или 1 в качестве низкого или высокого логического уровня.

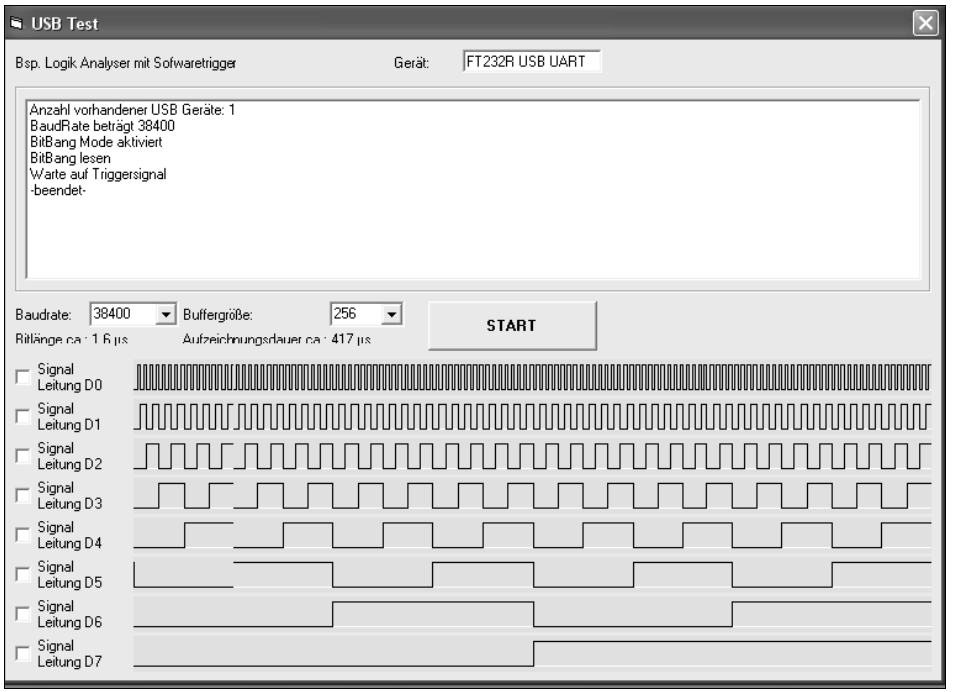


Рис. 16.1

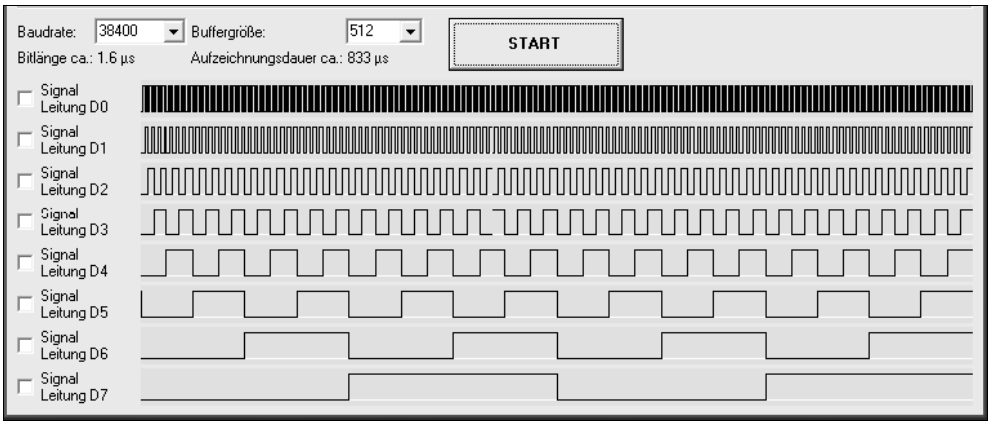


Рис. 16.2

Приведем фрагмент из исходного текста программы:

```
Private Sub bt_fb_test_Click()
.
' для инвертирования отображения исследуемого сигнала
' измените значение = 0 на = 1 в функции CBool
.
iAnzahl = CInt(Len(strReadBuffer))
```

```
For iPos = 1 To iAnzahl
     LinieXyWertZeichnen 0, CBool(((Asc(Mid$(strReadBuffer, iPos, _1&))
         And 2 ^ 0) / 2 ^ 0) = 0)
     LinieXyWertZeichnen 1, CBool(((Asc(Mid$(strReadBuffer, iPos, _1&))
         And 2 ^ 1) / 2 ^ 1) = 0)
.
.
Next iPos.
..
End Sub
```

Запуск осуществляется достаточно просто. Сначала составляется байт запуска `Triggerbyte`, в соответствии с состоянием флажков (`chkSignalLeitung (…)`) для каждого канала анализатора (листинг 16.1).

Листинг 16.1

```
Private Sub bt_fb_test_Click()
.
Dim Triggerbyte As Byte
Triggerbyte = 2 ^ 0 · chkSignalLeitung(0).Value + _
              2 ^ 1 · chkSignalLeitung(1).Value + _
              2 ^ 2 · chkSignalLeitung(2).Value + _
              2 ^ 3 · chkSignalLeitung(3).Value + _
              2 ^ 4 · chkSignalLeitung(4).Value + _
              2 ^ 5 · chkSignalLeitung(5).Value + _
              2 ^ 6 · chkSignalLeitung(6).Value + _
              2 ^ 7 · chkSignalLeitung(7).Value
.
' получать состояние линий ввода/вывода и ожидать,
' пока Databyte = Triggerbyte
Do
     FtStatus = FT_GetBitMode(lngHandle, Databyte)
     If FtStatus <> FT_OK Then
        LoggerList.AddItem "Fehler bei FT_GetBitMode"
        GoTo CloseHandle
     End If
Loop Until Databyte = Triggerbyte
SendRead_FTDI_Bytes
…
End Sub
```

В программном цикле выполняется ожидание момента времени, когда значения входных сигналов (в байте данных или переменной `Databyte`) не будут идентичны информации в байте запуска (переменной `Triggerbyte`).

16.1. Исследование цифровых схем

Если вы уже задались вопросом, почему приведенные временные диаграммы были почти идеальными, а вот ваши полученные результаты выглядят по-разному, то причина может быть в том, что сигнальная линия D0 была установлена в программе не в качестве входа, а в качестве выхода. Этот выход соответствует синхросигналам двоичного счетчика. На самом деле, кроме тактового синхросигнала с 8-разрядным счетчиком связаны только первые семь битов.

Таким образом можно определять не только синхросигнал, но и анализировать цифровую схему (сравните с процессом при разработке программы для исследования интерфейса I^2C, описанным в *главе 13*).

ГЛАВА 17

Управление шаговыми двигателями

С помощью единственной микросхемы токового усилителя (драйвера) и USB-адаптера можно выполнить устройство управления двумя униполярными шаговыми двигателями. Применив еще такую же микросхему, можно управлять и четырьмя униполярными шаговыми двигателями.

Шаговые двигатели преобразуют электрические сигналы в отдельные шаги перемещения ротора двигателя. Шаговый двигатель со 180 шагами за один оборот с каждым импульсом перемещается на угол 2º. Шаговый двигатель, имеющий угловое перемещение 3,6° за один шаг, совершает полный оборот за 100 шагов при полношаговом режиме управления. Помимо этого существует также и режим полушагового управления, при котором двигателю с угловым перемещением 3,6º для совершения полного оборота требуется 200 полушагов, что по сути соответствует двигателю с угловым перемещением равным 1,8° при полношаговом управлении.

Шаговые двигатели предназначены для приведения в движение тех или иных механизмов и, в зависимости от их применения, могут использоваться в разных режимах, которые удовлетворяют различным требованиям к точности и производительности.

Шаговые двигатели применяются не только в роботах и в промышленных приводах механизмов, а везде, где требуется линейное перемещение, например, в дисководах для дискет и CD/DVD для позиционирования головок, а также в принтерах, сканерах и фотокопировальных аппаратах.

На рис. 17.1 в качестве примера показан шаговый двигатель с метрической резьбой на валу ротора двигателя, используемый в качестве линейного серводвигателя. Диаметр корпуса двигателя составляет 8 мм. При помощи гайки вращательное движение ротора двигателя преобразуется в линейное поступательное движение 0,014 мм/шаг. Собственное угловое перемещение двигателя составляет 18º/шаг.

Если вы хотите поэкспериментировать с шаговыми двигателями, то можете взять все еще функционирующий двигатель из неисправного дисковода CD-ROM, привода дискет или сканера.

Униполярный шаговый двигатель имеет пять (или шесть) электрических контактов; четыре из них предназначены для управления обмотками и один (или два) — для напряжения питания.

Для управления обмотками униполярного шагового двигателя нужны мощные задающие устройства, которые включают напряжение питания для соответствующей обмотки. Таких мощных устройств для управления шаговыми двигателями на нашей дополнительной плате не имеется.

Если у шагового двигателя есть только 4 контакта, то это скорее всего *биполярный* шаговый двигатель. Для управления таким биполярным шаговым двигателем можно использовать микросхему L293D четырехканального токового переключателя (драйвера) (рис. 17.2) или аналогичную микросхему.

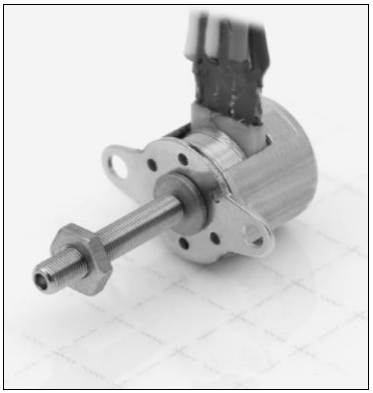


Рис. 17.1

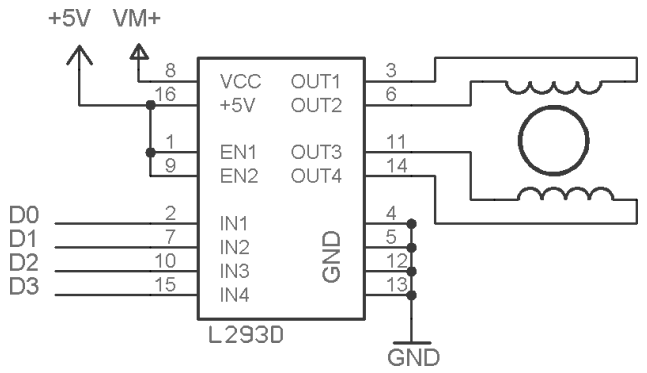


Рис. 17.2

17.1. Схема подключения униполярного шагового двигателя

На рынке имеется достаточно много разных микросхем драйверов для управления униполярными шаговыми двигателями различной мощности. Схему управления мощными шаговыми двигателями вы можете собрать на транзисторах большой мощности, таких, например, как BD697, или микросхемах L298, а для управления двигателями средней мощности подойдут 16 полевых транзисторов или ULN-микросхемы драйверов.

Существует целый ряд различных ULN-микросхем. В данном примере используется микросхема драйвера ULN2803, которая имеет TTL/CMOS-совместимые входы для прямого соединения с микросхемой FT232R.

На рис. 17.3 показана электрическая схема управления шаговым двигателем с помощью четырех входных сигналов. Каждый вход предназначен для подачи напряжения питания на одну отдельную обмотку; D0 — на обмотку L1, D1 — на L2, D2 — на L3, и D3 — на L4.

На следующей электрической схеме показано применение аналогичной микросхемы в схеме управления тем же двигателем, но использующей только два входных сигнала: D0 и D1 (рис. 17.4). В данном случае на входы I2 (вывод 2) и I4 (вывод 4)

подаются инвертированные выходные сигналы соответствующих входных сигналов I1 (вывод 1) и I3 (вывод 3).

Таким образом, применив две такие же микросхемы драйвера ULN2803 и используя 8 входных управляющих сигналов D0—D7, можно управлять четырьмя аналогичными шаговыми двигателями.

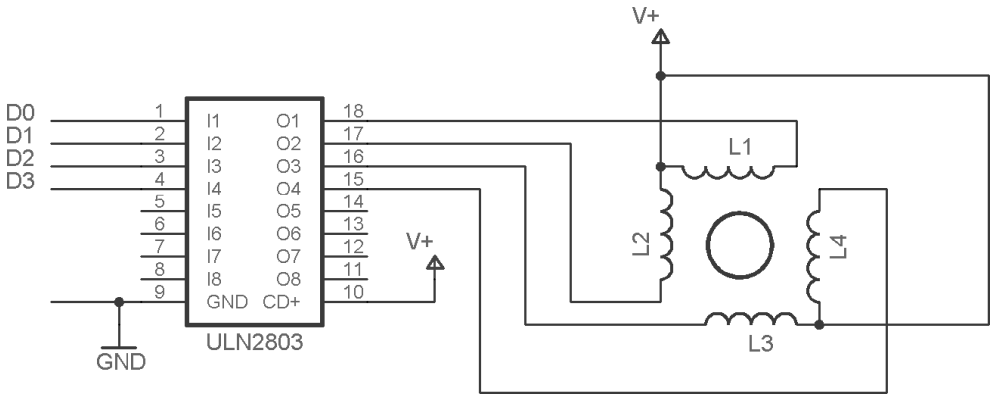


Рис. 17.3

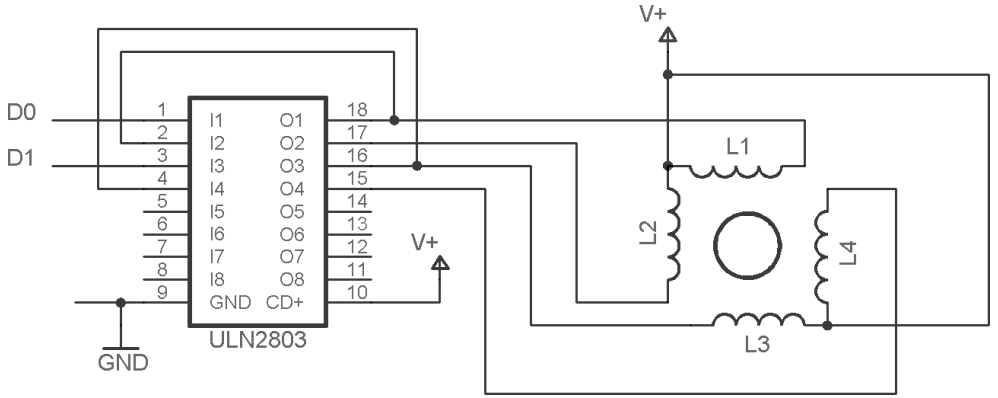


Рис. 17.4

17.2. Пошаговое управление

Обмотки двигателя должны включаться последовательно друг за другом.

Табличное представление управления двигателем для полношагового режима показано в табл. 17.1.

Табличное представление управления для полношагового режима с двумя входными сигналами приведено в табл. 17.2.

Табличное представление управления для полушагового режима показано в табл. 17.3.

Таблица 17.1

Шаг	Обмотка 1	Обмотка 2	Обмотка 3	Обмотка 3
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Таблица 17.2

Шаг	D0	D1
1	0	0
2	1	0
3	1	1
4	0	1

Таблица 17.3

Шаг	Обмотка 1	Обмотка 2	Обмотка 3	Обмотка 4
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

В представленной микросхеме ULN2803 имеются внутренние диоды, которые служат для защиты ключевых транзисторов от выбросов напряжения, возникающего при переключении обмоток. Максимальный ток в нагрузке, подключаемой к каждому выходу микросхемы драйвера, должен быть не более 500 мА, а рабочее напряжение ключевых транзисторов не должно превышать 50 В. Для увеличения тока в нагрузке более 500 мА разрешено входы и выходы микросхемы ULN2803 подключать параллельно.

17.3. Пример программы для управления шаговым двигателем

Написать программу (рис. 17.5) для режима Bit Bang микросхемы FT232R достаточно просто. Исходный код программы находится на прилагаемом к книге компакт-диске в каталоге \Beispielprogramme\Bsp_17 в файле Bsp.VBP (листинг 17.1).

Рис. 17.5

Листинг 17.1

```
private Sub step()
On Error GoTo step_error
strBeschreibung = Trim(Me.DeviceName.Text) & Chr(0)
LoggerList.Clear
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION,lngHandle) <>_
                                                          FT_OKThen
     LoggerList.AddItem "Fehler bei Aufruf: FT_OpenEx"
     Exit Sub
End If
' установить скорость в бодах
FtStatus = FT_SetBaudRate(lngHandle, Val(lb_baudrate.Text))
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler SetBaudRate"
     GoTo CloseHandle
  Else
     LoggerList.AddItem "BaudRate beträgt " & lb_baudrate.Text
End If
intMask = &HFF              ' установить режим Bit Bang — для всех выходов
intMode = 1                 ' асинхронный режим Bit Bang
FtStatus = FT_SetBitMode(lngHandle, intMask, intMode)
```

```
If FtStatus <> FT_OK Then
     LoggerList.AddItem "Fehler bei FT_SetBitMode"
     GoTo CloseHandle
  Else
     LoggerList.AddItem "asynchronen Bit Bang Mode aktiviert"
End If
' записать байт
If FT_Write(lngHandle, strWriteBuffer, Len(strWriteBuffer), _
                                lngBytesWritten) <> FT_OK Then
     LoggerList.AddItem "Write Failed"
     GoTo CloseHandle
  Else
     LoggerList.AddItem Len(strWriteBuffer) & " Byte geschrieben"
End If
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
     LoggerList.AddItem "Fehler bei Aufruf: FT_Close"
     Exit Sub
  Else
     LoggerList.AddItem "-beendet-" & Time()
End If
End sub
```

В данном случае используется режим Bit Bang, поскольку требуется выполнить операцию записи, а не чтения.

Перед вызовом подпрограммы `Step` в переменную `strWriteBuffer` должен быть записан соответствующий байт. Это осуществляется для каждой отдельной кнопки от **Schritt1** до **Schritt4** (шаг1—шаг4):

```
Private Sub bt_s1_Click()
   strWriteBuffer = Chr$(1) 'D0=1
   Call step
End Sub
Private Sub bt_s2_Click()
   strWriteBuffer = Chr$(2)
   Call step
End Sub
Private Sub bt_s3_Click()
   strWriteBuffer = Chr$(4)
   Call step
End Sub
Private Sub bt_s4_Click()
   strWriteBuffer = Chr$(8) 'D3=1
   Call step
End Sub
```

Для управления шаговым двигателем с двумя входными сигналами D0 и D1 вы можете попробовать последовательность шагов 1, 2, 3 и 4 (вместо 1, 2, 4 и 8). Возможна ли также последовательность 0, 1, 3 и 2?

Шаговый двигатель через микросхему ULN2803 должен быть подключен к сигнальным линиям от D0 до D3 на дополнительной плате. Не забудьте выполнить необходимые соединения с корпусом.

Шаговый двигатель и управляющая электроника были демонтированы из старого дисковода 5,25" (рис. 17.6). На плате, к счастью, уже находится микросхема ULN2003/MC1413P. Входы микросхемы были напрямую соединены с USB-адаптером. Функциональность проверялась тестовой программой. Быстро может быть установлена правильная последовательность шагов с 3-1-2-4 (вперед) и 4-2-1-3 (назад). Вместо того чтобы изменить последовательности кабелем, можно, конечно, изменить байты в программе.

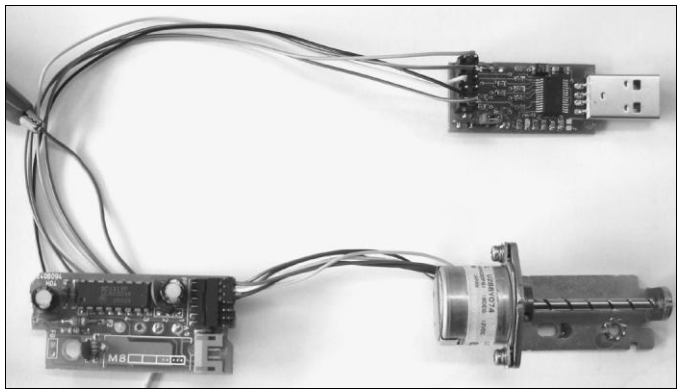


Рис. 17.6

На рис. 17.7 представлены некоторые шаговые двигатели линейных приводных механизмов для сравнивания их по размеру: слева (однополярный) LSP25 с напряжением питания 5 В и током потребления 0,5 А компании Nanotec, в середине (биполярный) LSP08 с 1,8 В/0,12 А и диаметром корпуса 8 мм и справа взятый из 5,25" дисковода для дискет 12-вольтовый двигатель компании Mitsubishi. Все эти двигатели работают с угловым перемещением 18°/шаг.

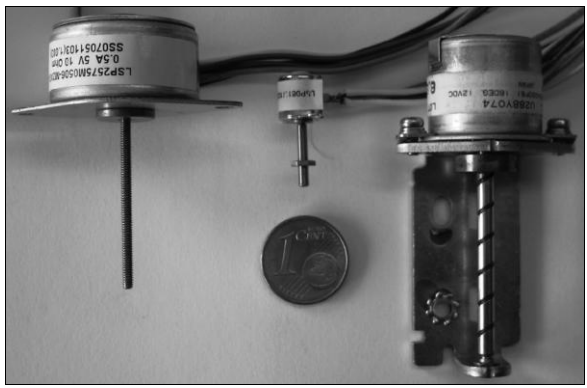


Рис. 17.7

ГЛАВА 18

Использование USB для защиты программ от копирования

Путем вызова функции `FT_ListDevices` можно получить информацию о подключенных в настоящее время FTDI-устройствах, т. е. в данном случае получить описание устройства или серийный номер микросхемы FT232R. Возможно (и разрешается) перепрограммировать описание устройства и серийный номер во внутренней EEPROM-памяти микросхемы. Для этого компания FTDI предлагает специальную программу под названием MPROG.

При приобретении большой партии микросхем FT232R клиент компании FTDI может даже получить собственный идентификатор изделия — PID. В обоих случаях следует изменить FTDI-драйвер; согласование выполняются в inf-файле! Драйвер после изменений должен быть заново установлен или загружен.

Каждая поставляемая компанией FTDI микросхема имеет определенный *идентификатор микросхемы* (чипа или, иначе, кристалла) (Chip-ID), который является неизменяемым. Этот идентификатор может быть дополнительно закодирован другой информацией, которая хранится в свободной области в EEPROM-памяти микросхемы FT232R. Поэтому эта микросхема хорошо подходит в качестве ключа для защиты от несанкционированного доступа (Dongle), чтобы защитить программное обеспечение от неразрешенного копирования.

Вызов функции `FT_EE_UASize` (в FTD2xxx.DLL) применяется для получения размера неиспользуемого свободного пространства в EEPROM-памяти, называемого *областью пользователя* (User Area). Функция возвращает число байт, которые в EEPROM-памяти могут применяться в качестве ключа защиты от копирования. Эти байты области пользователя можно записывать и читать с помощью соответствующих двух функций: `FT_EE_UAWrite` и `FT_EE_UARead`.

Чтобы прочитать идентификатор микросхемы, используется библиотека FTChipID.DLL, которая находится на прилагаемом компакт-диске в каталоге \Beispielprogramme\Bsp_Dongle.

Программа FTDI_ChipID.exe, расположенная в этом же каталоге \Beispielprogramme\Bsp_Dongle, после запуска показывает различие между серийным номером в EEPROM и неизменяемым идентификатором микросхемы FTDI (рис. 18.1).

Исходный код программы содержится в файле Bsp.VBP в каталоге Bsp_Dongle. В данном примере вызов функции `FTID_GetDeviceChipID` ссылается на устройство 0.

В случае использования нескольких FT232R-устройств нужно принимать во внимание индексы устройств.

```
' описание в Modul.Bas
Public Declare Function FTID_GetDeviceChipID Lib "FTChipID.DLL" _
(ByVal dwDeviceIndex As Long, ByRef lpChipIDBuffer As Long) As Long
' вызов
Private Sub GetDeviceChipID_Click()
On Error GoTo FtdiChipId_error
ftid_Status = FTID_GetDeviceChipID(0, ChipID)
     If ftid_Status <> FTID_SUCCESS Then
        LoggerList.AddItem "FTID_GetDeviceChipID failed status = "_
                                                    & ftid_Status
        Exit Sub
     Else
           LoggerList.AddItem ("Chip ID = " & ChipID)
     End If
FtdiChipId_error_ende:
     Exit Sub
FtdiChipId_error:
   MsgBox Err.Description
   Resume FtdiChipId_error_ende
End Sub
```

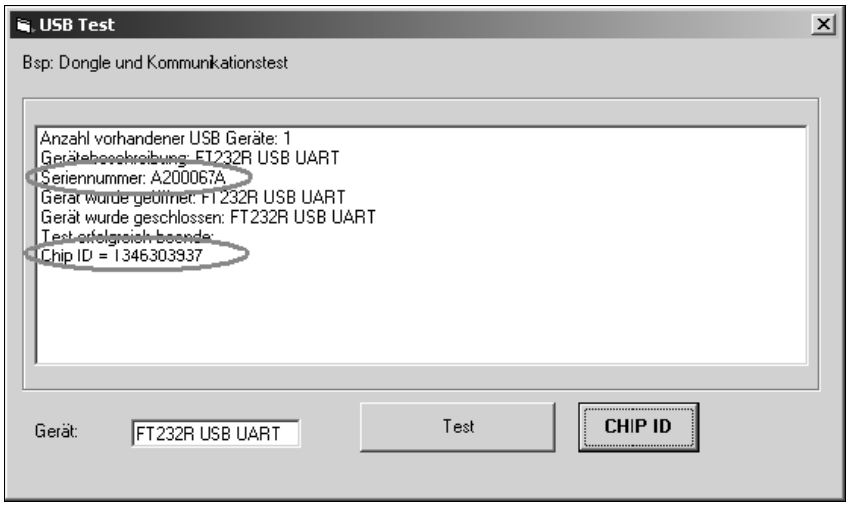


Рис. 18.1

Другие вызовы функций, реализованных в библиотеке FTChipID.DLL, даны в табл. 18.1.

Если этих возможностей для вас недостаточно или вам не хватает места в EEPROM-памяти микросхемы для хранения кодовой информации, то можно ис-

пользовать дополнительную EEPROM-память с интерфейсом I^2C. Информацию о том, как подключить дополнительную EEPROM-память с интерфейсом I^2C, вы найдете в *главе 13*.

Таблица 18.1

Вызов функции	Краткое описание
`FTID_GetNumDevices`	Определяет количество FT232R и FT245R
`FTID_GetDeviceSerialNumber`	Определяет серийный номер
`FTID_GetDeviceDescription`	Определяет описание устройства
`FTID_GetDeviceLocationID`	Определяет местонахождение идентификатора
`FTID_GetDllVersion`	Определяет версию библиотеки FTChipID.DLL

18.1. Вызов FTDI-функций в Visual C

В каталоге Beispielprogramme\Bsp_Dongle\VCCWin32Beispiel компакт-диска находится простой пример на Visual C для вызовов функций библиотеки FTChipID.DLL. Предоставленная готовая программа ChipID.exe находится в каталоге отладки Beispielprogramme\Bsp_Dongle\VCCWin32Beispiel\Debug. Если вы сразу не скопировали библиотеку FTChipID.DLL в системный каталог Windows, то будет нужен файл в каталоге программы (сравните с дополнительными примерами на главной странице сайта **www.FTDIChip.com**).

Запустите программу ChipID.exe и нажмите кнопку **Find 232R Devices**. В результате программа отобразит параметры всех устройств на микросхемах FT232R и FT245R, подключенных к USB. На рис. 18.2 показаны результаты, полученные с помощью программы на Visual Basic (слева) и Visual C (справа).

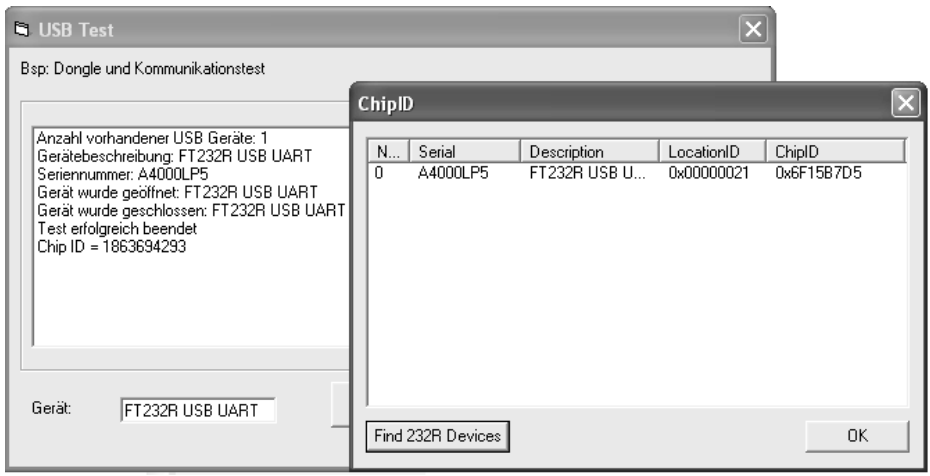


Рис. 18.2

Данные программа на Visual C (листинг 18.1) представляет в виде таблицы с помощью элемента управления `ListView`. В программе на Visual Basic, напротив, вы видите в идентификаторе микросхемы (Chip ID) последовательность цифр 1863694293_{10}. Этот идентификатор в программе на Visual C представляется в шестнадцатеричной системе счисления как $6F15B7D5_{16}$.

Листинг 18.1

```
Void PopulateListView(HWND hWndDlg, HWND hList)
{
       FTID_STATUS dStatus;
       unsigned long ulNumDevices, ulLocID, ulChipID;
       char cSerialNumber[50], cDescription[100], cLocationIDS[10]
char cChipIDS[10], ErrorMessage[256];
       LVITEM lvItem;
       if(hList == NULL) {
              hList = GetDlgItem(hWndDlg, IDC_DEVICE_LIST);
       }
       SendMessage(hList, LVM_DELETEALLITEMS, 0, 0);
       dStatus = FTID_GetNumDevices(&ulNumDevices);
       if((dStatus == FTID_SUCCESS) && ulNumDevices) {
            for(int i = 0; i < (int)ulNumDevices; i++) {
                  InsertDevNumberItem(hList, &lvItem, i);
              dStatus = FTID_GetDeviceSerialNumber(i, cSerialNumber, 50);
                  if(dStatus == FTID_SUCCESS) {
                    InsertSubItem(hList, &lvItem, 1, cSerialNumber);
                      }
              dStatus = FTID_GetDeviceDescription(i, cDescription, 100);
                  if(dStatus == FTID_SUCCESS) {
                      InsertSubItem(hList, &lvItem, 2, cDescription);
                      }
              dStatus = FTID_GetDeviceLocationID(i, &ulLocID);
                  if(dStatus == FTID_SUCCESS) {
                      sprintf(cLocationIDS, «0x%08X», ulLocID);
                      InsertSubItem(hList, &lvItem, 3, cLocationIDS);
                      }
              dStatus = FTID_GetDeviceChipID(i, &ulChipID);
                  if(dStatus == FTID_SUCCESS) {
                      sprintf(cChipIDS, «0x%08X», ulChipID);
                      InsertSubItem(hList, &lvItem, 4, cChipIDS);
                      }
              }
       }
       if(dStatus != FTID_SUCCESS) {
              FTID_GetErrorCodeString("EN", dStatus, ErrorMessage, 256);
              MessageBox(hWndDlg, ErrorMessage, NULL, MB_OK);
       }
}
```

Исходный код программы содержится в файле ChipID.cpp в каталоге Bsp_Dongle\VCCWin32Beispiel. Таблица результатов заполняется при вызове функции `Void PopulateListView(HWND hWndDlg, HWND hList)`.

Для определения количества подключенных FTDI-устройств первой вызывается функция `FTID_GetNumDevices`. Если были найдены одна или несколько микросхем FT232R (в случае успеха), будут установлены параметры для каждого устройства FT232R в цикле от 0 до `(int)ulNumDevices`.

Вызов функции `sprintf` преобразует содержащееся в переменной `ulChipID` число для последующего его вывода в шестнадцатеричном формате.

ГЛАВА 19

Изменение данных в EEPROM-памяти

Компания FTDI со своим драйвером D2XX предоставляет в распоряжение пользователей много дополнительных функций для интегрированной EEPROM-памяти. Изменения в EEPROM-памяти требуются тогда, когда вы хотите приспособить CBUS для других приложений или использовать идентификатор изделия (PID). При этом следует быть очень осторожным, чтобы ваша микросхема FT232R и драйвер и далее оставались в работоспособном состоянии.

Для чтения EEPROM-памяти могут использоваться вызовы функций `FT_EE_Read`, `FT_EE_ReadEx` и `FT_ReadEE`. Для программирования EEPROM-памяти применяют вызовы функций `FT_EE_Program`, `FT_EE_ProgramEx` и `FT_WriteEE`. При этом стереть данные можно с помощью функции `FT_EraseEE`.

После вызова функции `FT_EE_UASize` возвращается количество байтов области пользователя, которые свободны в EEPROM-памяти и могут использоваться в качестве ключа для защиты от копирования. Эти байты могут записываться и читаться при помощи вызовов соответствующих функций `FT_EE_UAWrite` и `FT_EE_UARead`.

Для некоторых функций необходим указатель на заданную структуру данных (см. программную документацию D2XX Programmer's Guide компании FTDI, расположенную на компакт-диске в каталоге \FTDI_Manuals).

При помощи программы MProg в вашем распоряжении оказывается лучшая и более надежная возможность изменять данные в EEPROM-памяти некоторых микросхем компании FTDI. Если вы что-то изменяли, то после наверняка обнаружите это, когда внезапно будет запрошена новая установка драйвера.

Самую последнюю версию программы MProg Utility вы можете найти на сайте **www.FTDIChip.com** компании FTDI. Версию 3.0 (MProg3.0_Setup.exe) можете взять на прилагаемом к книге компакт-диске (эта версия не подходит для версий Windows 98 и Windows Me).

Процедура установки очень простая и не требует пояснений.

Подключите USB-адаптер и проверьте, обнаруживается ли микросхема FT232R после запуска программы MProg версии 3.0.

Если вы нажмете кнопку  для сканирования имеющихся в распоряжении устройств, то в нижнем небольшом поле списка диалогового окна программы при ус-

пешном обнаружении микросхемы должно будет выведено сообщение **Number Of Programmed Devices = 1**. Далее вы можете в меню **Tools** (сервис) выбрать пункт **Read and Parse** (читать и анализировать) (рис. 19.1), что позволит автоматически определять текущие настройки, если хотя бы одно FTDI-устройство будет подключено.

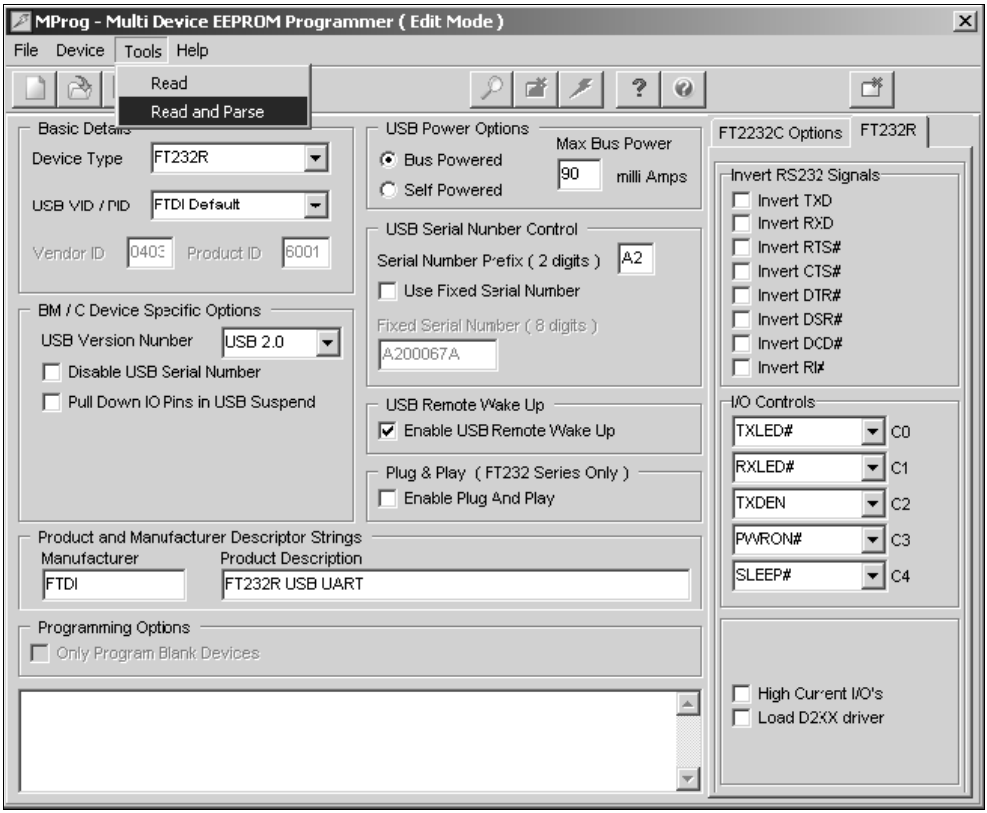


Рис. 19.1

Микросхема FT232R распознается автоматически, отображаются ее программные настройки (конфигурация), которые вы можете сохранить в файле образца (в папке Templates), как это показано на рис. 19.2.

Для сохранения данной конфигурации в меню **File** (файл) выберите пункт **Save as** (сохранить как) или же в главном окне программы щелкните кнопку с изображением дискеты  (см. рис. 19.2, кнопка обведена кружком). Эту конфигурацию позже вы снова сможете загрузить и при необходимости вновь записать в EEPROM-память микросхемы FT232R.

Кнопка режима правки  (Edit Mode) является активной только тогда, когда вы перед этим составили новый образец (шаблон) и сохранили его или загрузили уже

существующий. После нажатия этой кнопки находящиеся в EEPROM-памяти основные настройки можно изменить.

При нажатии кнопки (Program All Existing Divices) EEPROM-память записывается с заново установленными значениями.

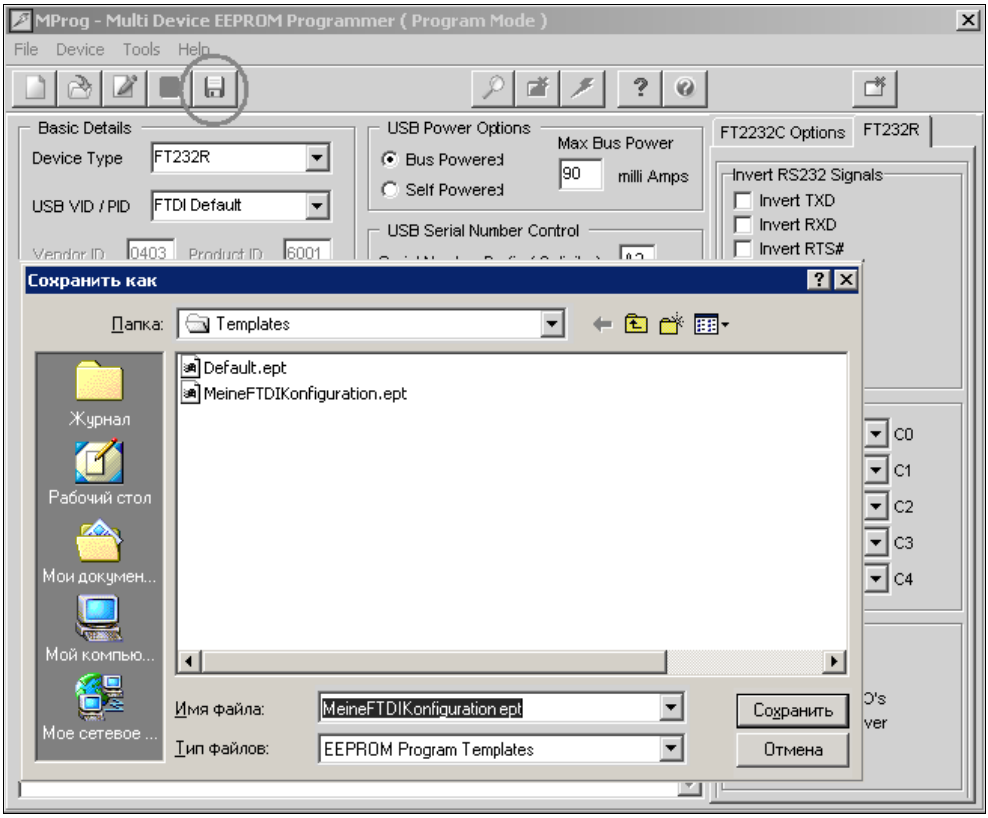


Рис. 19.2

Если вы внесли какие-нибудь произвольные значения для идентификатора изделия (в поле **Product ID**) или при описании изделия (поля **Manufacturer** и **Product Description**) (рис. 19.3), то USB-драйвер компании FTDI при следующем подключении USB-адаптера может более не загружаться. В этом случае, прежде чем драйвер снова будет корректно установлен, следует при помощи редактора изменить значения в файле ABE2XX.inf в соответствии со значениями, внесенными в EEPROM-память.

Наиболее важные изменяемые вами настройки (на рис. 19.3 они обведены овалом и кружком) — это опции питания USB, а именно переключатели **Bus Powered** или **Self Powered**, а также назначенные значения управляющих CBUS-сигналов с CBUS0 по CBSU4 (в окне это соответствующие раскрывающиеся списки **C0**, **C1**, **C2**, **C4**).

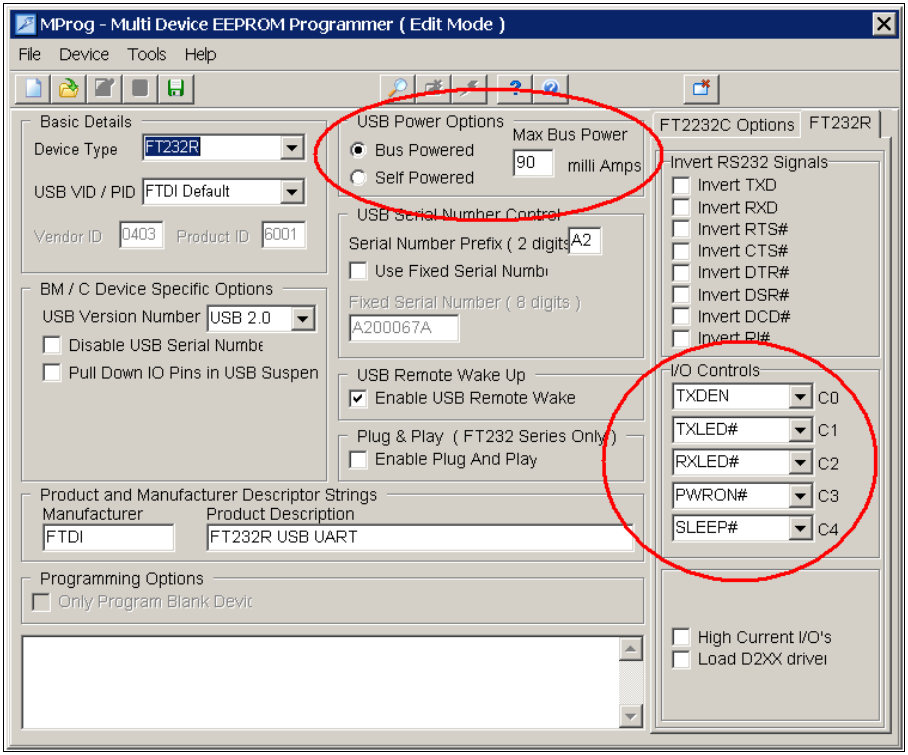

MProg - Multi Device EEPROM Programmer (Edit Mode)
File Device Tools Help
Basic Details
Device Type FT232R
USB VID / PID FTDI Default
Vendor ID 0403 Product ID 6001
BM / C Device Specific Options
USB Version Number USB 2.0
Disable USB Serial Numbe
Pull Down IO Pins in USB Suspen
USB Power Options
Bus Powered
Self Powered
Max Bus Power 90 milli Amps
USB Serial Number Control
Serial Number Prefix (2 digits A2
Use Fixed Serial Numb
Fixed Serial Number (8 digits)
A200067A
USB Remote Wake Up
Enable USB Remote Wake
Plug & Play (FT232 Series Only)
Enable Plug And Play
Product and Manufacturer Descriptor Strings
Manufacturer FTDI
Product Description FT232R USB UART
Programming Options
Only Program Blank Devic
FT2232C Options FT232R
Invert RS232 Signals
Invert TXD
Invert RXD
Invert RTS#
Invert CTS#
Invert DTR#
Invert DSR#
Invert DCD#
Invert RI#
I/O Controls
TXDEN C0
TXLED# C1
RXLED# C2
PWRON# C3
SLEEP# C4
High Current I/O's
Load D2XX drive

Рис. 19.3

ГЛАВА 20

Последовательная запись и чтение без VCP-драйвера

USB-адаптер может быть соединен с другими микроконтроллерами с последовательным интерфейсом. Сигнал TxD USB-адаптера (вывод 7 двухрядного 10-контактного штырькового соединительного разъема, см. рис. 3.2) соединяется с линией RxD такого микроконтроллера, а сигнал RxD USB-адаптера (штырьковый вывод 6 разъема) — с линией TxD микроконтроллера (табл. 20.1). Теперь не хватает только соединения корпуса микроконтроллера ("земля") с выводом 2 штырькового соединительного разъема адаптера.

Таблица 20.1

Последовательные сигналы адаптера	**Выводы разъема USB-адаптера**		**Сигналы микро-контроллера**	
TxD	Вывод 7 (выход)	отправка данных →	RxD	вход
RxD	Вывод 6 (выход)	← прием данных	TxD	вход
"земля"	Вывод 2			"земля"

После установки FTDI-драйверов связь с микроконтроллером может осуществляться по (виртуальному) COM-порту. Для виртуального COM-порта драйвер FTD2XX возможно использовать непосредственно, без установки VCP-драйвера (Virtual Com Port).

Примером устройства для подобного соединения с USB-адаптером может быть простой измерительный прибор. Измерительный прибор отправляет результаты измерения по требованию, т. е. после передачи стартовой команды измерительный прибор активен. Затем устройство устанавливает значения и отсылает результаты обратно по последовательному интерфейсу. В данном случае для измерительного прибора требуется в качестве сигнала "старт" передавать байт со значением 10, а в качестве результата прибор будет возвращать 6 байтов. Для вывода строк сообщения используется известный элемент `LoggerList`.

Полный исходный код на Visual Basic разделен на несколько функциональных блоков (листинги 20.1—20.8).

Листинг 20.1

```
Private Sub read_ti_instrument()
' USB-устройство открыть
strBeschreibung = "FT232R USB UART"& Chr(0)
If FT_OpenEx(strBeschreibung, FT_OPEN_BY_DESCRIPTION, lngHandle)
<> FT_OK Then
      LoggerList.AddItem "Fehler bei Aufruf: FT_OpenEx"
      Exit Function
   Else
      LoggerList.AddItem "----"
End If
..
```

Листинг 20.2

```
' задать значения последовательного интерфейса
' установить скорость  19200 бод
FtStatus = FT_SetBaudRate(lngHandle, 19200)
If FtStatus <> FT_OK Then
      LoggerList.AddItem "SetBaudRate Failed"
      GoTo CloseHandle
   Else
      LoggerList.AddItem "SetBaudRate to 19200"
End If
' установить биты данных, стоп-биты и паритет
' 8 битов данных, 1 стоп-бит, нет паритета
If FT_SetDataCharacteristics(lngHandle, FT_BITS_8, FT_STOP_BITS_
1, FT_PARITY_NONE) <> FT_OK Then
      LoggerList.AddItem "SetDataCharacteristics Failed"
      GoTo CloseHandle
End If
' контроль потока отсутствует — no flow control
If FT_SetFlowControl(lngHandle, FT_FLOW_NONE, 0, 0) <> FT_OK
Then
      LoggerList.AddItem "SetFlowControl Failed"
      GoTo CloseHandle
End If
' 5-секундный тайм-аут second
If FT_SetTimeouts(lngHandle, 5000, 0) <> FT_OK Then
      LoggerList.AddItem "SetFlowControl Failed"
      GoTo CloseHandle
End If
.
```

Листинг 20.3

```
' послать стартовый сигнал 10 на микроконтроллер
.
strWriteBuffer = Chr$(10)
lngBytesWritten = 0.
If FT_Write(lngHandle, strWriteBuffer, 1, lngBytesWritten) <> FT_
OK Then
      LoggerList.AddItem "Startsignal war fehlerhaft"
      GoTo CloseHandle
End If
.
```

Листинг 20.4

```
' прочитать количество байтов
.
flTimedout = False
flFatalError = False
' Get number of bytes waiting to be read
If FT_GetQueueStatus(lngHandle, FT_RxBytes) <> FT_OK Then
       LoggerList.AddItem "Status war fehlerhaft"
       GoTo CloseHandle
End If
```

Листинг 20.5

```
' ожидание, пока не достигнуто требуемое количество
' (ожидается 6 байтов)
.
WarteAufMehr:
FtStatus = FT_GetStatus(lngHandle, FT_RxBytes, FT_TxBytes,
                                                lngevent_get_stat)
      If FtStatus = FT_OK Then
          Else
             If FtStatus = FT_IO_ERROR Then
                LoggerList.AddItem "Status IO Fehler"
                GoTo CloseHandle
             End If
          End If
          i = i + 1
          If i > 20000 Then
             LoggerList.AddItem "Lesen abgebrochen"
             GoTo CloseHandle
      End If
If FT_RxBytes < 6 Then GoTo WarteAufMehr
```

Листинг 20.6

```
.
' прочитать 6 байтов (количество в FT_RxBytes) в strReadbuffer
' обратить внимание на возможные ошибки
.
flTimedout = False
flFatalError = False
lngTotalBytesRead = 0
strReadBuffer = ""
Do
      lngBytesRead = 0
      FtStatus = FT_Read(lngHandle, strReadBuffer, FT_RxBytes, _
                                                        lngBytesRead)
      If (FtStatus = FT_OK) Or (FtStatus = FT_IO_ERROR) Then
          If lngBytesRead > 0 Then
             lngTotalBytesRead = lngTotalBytesRead + lngBytesRead
          Else
             flTimedout = True
          End If
      Else
          flFatalError = True
      End If
Loop Until (lngTotalBytesRead = FT_RxBytes) Or _
           (flTimedout = True) Or _
           (flFatalError = True)
.
```

Листинг 20.7

```
' данные буфера strReadBuffer показать в Loggerlist
' или сообщения об ошибках
.
If (flTimedout = False) And (flFatalError = False) Then
    LoggerList.AddItem strReadBuffer
    flFailed = False
ElseIf flTimedout = True Then
    LoggerList.AddItem "FT_Read timeout Fehler: = " & FtStatus
Else
    LoggerList.AddItem "FT_Read Fehler = " & FtStatus
End If
```

Листинг 20.8

```
' окончание программы
.
CloseHandle:
If FT_Close(lngHandle) <> FT_OK Then
     LoggerList.AddItem "Close Failed"
End If
     LoggerList.AddItem "Closed at:" & Time
End Sub
```

ГЛАВА 21

Подключение набора для изучения микроконтроллера к компьютеру с помощью USB

Издательство Franzis предлагает новый обучающий набор для изучения микроконтроллера ATtiny13 — вводный курс для программирования малогабаритного 8-разрядного микроконтроллера производства компании Atmel. Обучающий набор работает только с последовательными интерфейсами ПК. Подключить набор посредством USB-интерфейса можно при помощи имеющегося в магазинах конвертера интерфейса USB в RS-232. Однако плату из обучающего набора "Mikrocontroller" можно соединить с рассматриваемым в данной книге USB-адаптером (рис. 21.1). После этого вы можете проводить ваши эксперименты с микроконтроллером при помощи USB-адаптера.

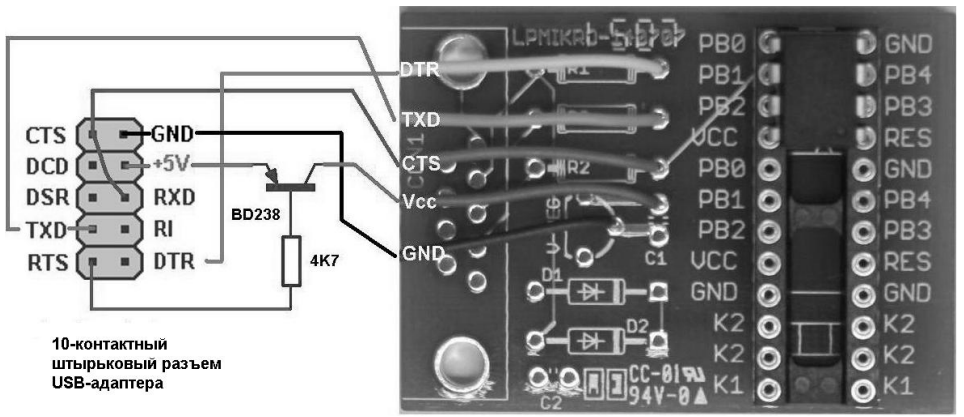


Рис. 21.1

Внимание!

Никогда напрямую не подключайте USB-адаптер к последовательному порту, поскольку 12-вольтовые сигналы интерфейса RS-232 могут привести к выходу из строя USB-адаптера!

Для монтажа полезно воспользоваться электрической схемой из документации к обучающему набору "Mikrocontroller".

Сигнальная линия CTS должна быть подключена с RXD. Данное соединение выполняется прямо на разъеме, т. к. на плате не размещено никаких других монтажных элементов.

Можно было бы также подключить напряжение питания +5 В непосредственно к питающему напряжению Vcc на плате. В наборе "Mikrocontroller" напряжение питания формируется с помощью 5-вольтового регулятора, управляемого сигналом RTS. В этом примере в качестве регулирующего элемента для надежности применяется мощный транзистор BD236/238. Сигнал RTS через резистор сопротивлением 4,7 кОм поступает на базу транзистора. Благодаря такой схеме с помощью программного обеспечения, поставляемого вместе с обучающим набором, при необходимости осуществляется отключение питающего напряжения.

Для повышения мощности источника питания дополнительно используется линия DTR. Она в данном случае соединяется с сигнальной линией DTR на USB-адаптере. О других сигналах: так сигнал TXD непосредственно соединен с PB2, RXD и CTS — с PB1, а DTR — с PB0. Кроме того, необходимо выполнить подключение общего вывода "земля" (GND).

Уровни сигналов интерфейса RS-232 имеют обратные значения относительно 5-вольтовых сигналов ТТЛ-уровня, которые присутствуют в USB-адаптере.

Примечание редактора

В интерфейсе RS-232 логической единице соответствует сигнал, уровень напряжения которого попадает в диапазон от –12 до –3 В, а для логического нуля этот диапазон будет от +3 до +12 В.

Таким образом, необходим еще один мощный *p*-*n*-*p*-транзистор. Инверсия сигналов может быть выполнена в основных настройках EEPROM-памяти микросхемы FT232R. Для этого можно воспользоваться программой MProg (рис. 21.2), работа с которой была описана в *главе 19*.

Сигналы TXD, RXD, CTS и DTR должны быть инвертированы и записаны в EEPROM-память микросхемы FT232R. Для проверки успешного выполнения операции записи EEPROM нужно с помощью команды **Read and Parse** меню **Tools** программы MProg снова прочитать и проанализировать записанные данные. Если вы инвертируете еще и сигнал RTS, то для управления напряжением питания вместо мощного *p*-*n*-*p*-транзистора нужно воспользоваться транзистором *n*-*p*-*n*-типа.

Внимание!

Не забудьте после ваших экспериментов с обучающим набором "Mikrocontroller" снова вернуть прежние состояния инвертированных сигналов!

Автор инициализировал микроконтроллер ATtiny13 (**Bootloader und Fuses**) и некоторое время тестировал микросхему, используя порт ввода/вывода в качестве выхода. Приступив к работе, действуйте так же, как описано в документации к обучающему набору "Mikrocontroller". Сначала выберите COM-порт USB-адаптера

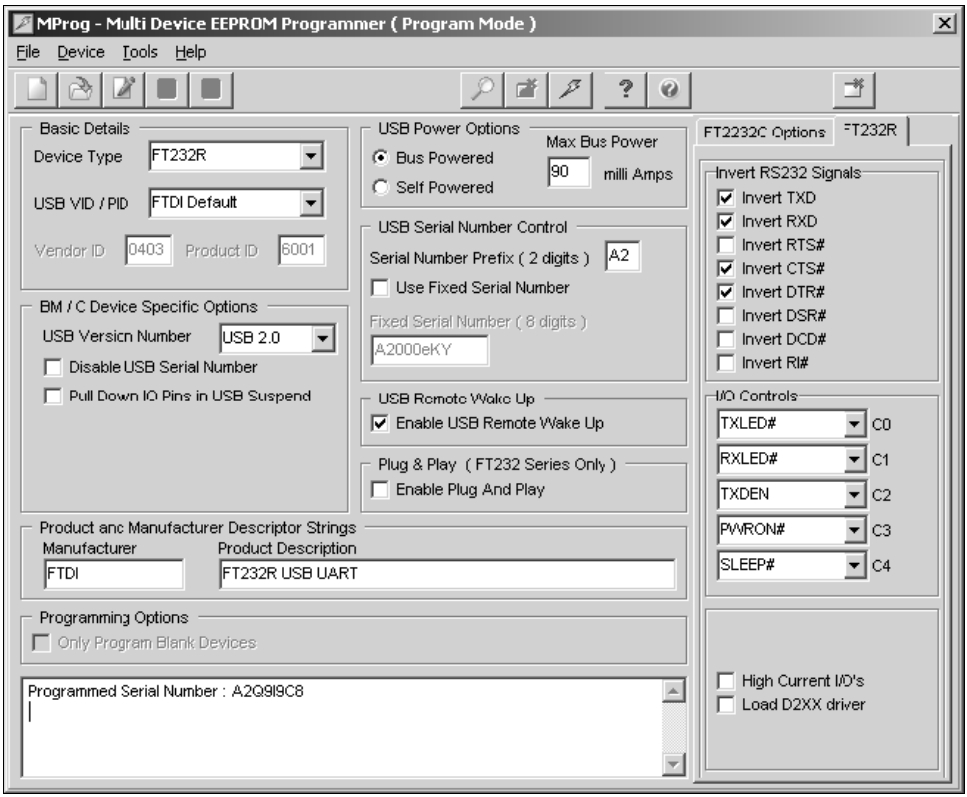


Рис. 21.2

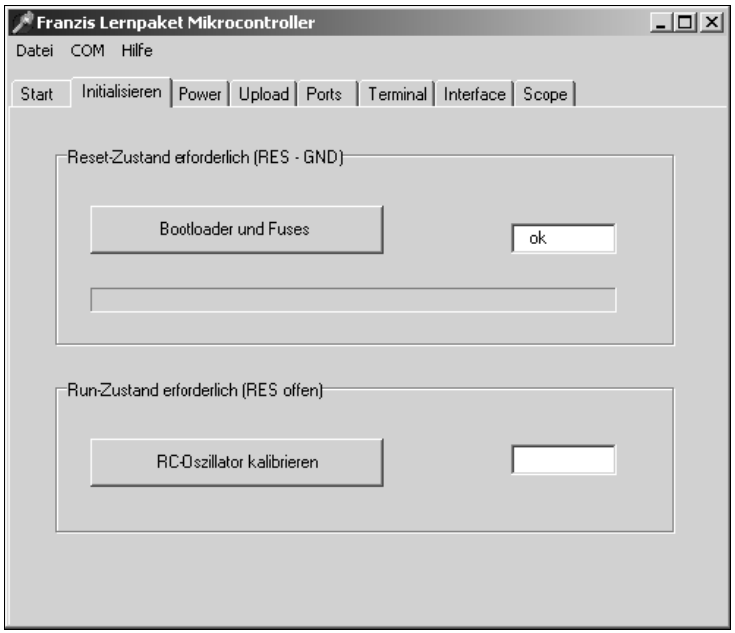


Рис. 21.3

(см. главу 4). Прежде всего, проверьте электропитание платы: оно включается при значении сигнала RTS = 1.

Установите проволочную перемычку между выводом RES платы и "землей" и нажмите кнопку **Bootloader und Fuses** для инициализации (рис. 21.3). Процесс продолжается 30—40 секунд и в заключение подтверждается сообщением "ok".

После этого вы можете начать выполнять свои эксперименты, как это описано в руководстве к обучающему набору "Mikrocontroller".

ГЛАВА 22

Пример флэш-программирования микроконтроллера Atmel-AT89LP

Эта глава задумана, прежде всего, для профессионалов, которые, может быть, захотят создать собственную программу для эмуляции интерфейса SPI.

> ***Примечание для пользователей AT89ISP***
>
> При необходимости вы должны установить перемычку JP_DDA на USB-адаптере таким образом, чтобы замкнуть вывод 10 выходного штырькового разъема на "землю" (см. рис. 3.2), если вы используете этот вывод в соответствии с документацией компании Atmel на внутрисхемный программатор AT89ISP. В противном случае вам при разработке вашего программного обеспечения следует учитывать то, что сигнал D4 (вывод 2 микросхемы FT232R) должен находиться на низком логическом уровне.

Многие микроконтроллеры допускают так называемое внутрисхемное программирование (In-System Programming, ISP) через последовательный интерфейс SPI (Serial Peripheral Interface). В апрельском номере № 424 журнала Elektor за 2006 год в статье "Universelle SPI-Box" (Универсальный SPI-блок) подробно обсуждались проблемы, связанные с использованием последовательного интерфейса RS-232 и интерфейса USB.

Если у вас нет соответствующего программирующего устройства, а вы можете самостоятельно программировать, то для программирования микроконтроллеров можно воспользоваться упомянутым ранее устройством SPI-Box, выполненным на одном микроконтроллере AT89C2051 производства компании Atmel. Другой возможностью устройства является, например, изменение сигналов, которыми можно управлять с помощью соответствующего программного обеспечения. Кроме того, с помощью SPI-блока, вероятно, можно также осуществить преобразование сигналов последовательного интерфейса в сигналы I^2C-интерфейса, однако для этого определенно потребуется соответствующее изменение программного обеспечения микроконтроллера.

Используя микросхему FT232R производства компании FTDI, можно с небольшими затратами и при помощи достаточно простого программного обеспечения реализовать многие последовательные интерфейсы. Кроме того, на своем сайте компания FTDI предоставляет различные примеры использования микросхемы FT2232C в режиме *MPSSE* (Multi-Protocol Synchronous Serial Engine, мультипротокольный синхронный последовательный механизм). Общее управление осуществляется без дополнительного микроконтроллера, но, к сожалению, реализуется только с помощью шины CBUS микросхемы FT232R.

В данном примере показано, как можно использовать микросхему FT232R компании FTDI в качестве ISP-флэш-адаптера для микроконтроллеров Atmel семейства AT89LP. В примере, написанном на Visual Basic, разъясняется, как настроить микросхему FT232R для режима Bit Bang и как можно через интерфейс SPI-USB за 1—2 с прочитать флэш-память емкостью 2 Кбайта микроконтроллера Atmel AT89LP2052.

Разработчик может использовать функции соответствующих DLL-библиотек разных языков программирования (C++, Delphi, Visual Basic и др.) и для этого не требуется основательных знаний. Драйвер WDM (Win32 Driver Model) и VCP-драйвер (Virtual COM Port — виртуальный COM-порт) с 2007 года могут использоваться одновременно. Классические функции Win32-API не следует путать с API-функциями драйвера FTDI. Драйвер WDM в этом случае предоставляет Win32-API-функции с некоторыми расширениями.

Для эмуляции интерфейсов очень интересным является режим Bit Bang микросхемы FT232R. Режим Bit Bang позволяет переключать восемь линий ввода/вывода (I/O) микросхемы FT232R в двунаправленный режим. Каждая линия ввода/вывода может по отдельности устанавливаться как на вывод (Output), так и на ввод (Input). Дополнительно имеются и другие служебные линии CBUS, которые тоже могут быть использованы двунаправленно. Пересылка байтов при этом, однако, ограничена USB-кадрами и, соответственно, является более медленной.

Скорость передачи данных при режиме Bit Bang определяется тактовой частотой бод-генератора и равна 16-кратной скорости передачи в бодах *(см. главу 9 о режиме Bit Bang)*.

Синхронный режим Bit Bang для чтения флэш-памяти является идеальным для ISP-программирования. Сначала байты программно разбиваются на биты, потом, согласно тактовому сигналу и линии синхронизации SCK, записываются в буфер записи, затем накапливаются в нем и передаются микросхемой FT232R. После чтения в буфере чтения будут находиться данные, полученные по линии MISO.

Расположение выводов микросхем FT232 и их описание приведены в табл. 22.1.

Таблица 22.1

FT232RL	FT232RQ	Описание
1	30	D0 — бит 0 шины данных в режиме Bit Bang (ввод/вывод)
5	2	D1 — бит 1 шины данных в режиме Bit Bang (ввод/вывод)
3	32	D2 — бит 2 шины данных в режиме Bit Bang (ввод/вывод)
11	8	D3 — бит 3 шины данных в режиме Bit Bang (ввод/вывод)
2	31	D4 — бит 4 шины данных в режиме Bit Bang (ввод/вывод)
9	6	D5 — бит 5 шины данных в режиме Bit Bang (ввод/вывод)
10	7	D6 — бит 6 шины данных в режиме Bit Bang (ввод/вывод)
6	3	D7 — бит 7 шины данных в режиме Bit Bang (ввод/вывод)

Для синхронного режима необходимы следующие функции драйвера D2XX компании FTDI:

- `FT_SetBitMode` — значение 0x04 активирует синхронный режим Bit Bang, 0 — вызывает сброс в стандартный режим;
- `FT_SetBaudRate` — установка скорости передачи данных в соответствии с заданной скоростью передачи данных в бодах. Максимальная скорость 1 Мбод. Значение 9600 бод определяет скорость передачи данных 9600 × 16 = 153600 байт/с, или около 1 байта за каждые 6,5 мкс;
- `FT_Write` — запись данных в порт;
- `FT_GetBitMode` — получить мгновенное значение данных ввода/вывода;
- `FT_Read` — чтение данных из порта.

22.1. ISP-программирование микроконтроллера Atmel AT89LPx052 посредством интерфейса SPI

SPI-интерфейс (Serial Port Interface) служит в качестве интерфейса программирования для последовательного внутрисхемного программирования ISP (In-System Programming) микроконтроллера AT89LPx052.

Между тем этот способ является общепринятым и предлагает для интегрированной инструментальной среды разработки IDE (Integrated Development Environment) некоторые преимущества, поскольку при этом можно отказаться от дорогостоящего программирующего устройства, а микроконтроллер при программировании может оставаться в схеме. Однако это высказывание не совсем верно. Так, если у микроконтроллера Atmel AT89LPx052 будет отключено внутрисхемное программирование ISP, то в этом случае может помочь только параллельное программирование, которое, в общем, практически не изменилось относительно предыдущих версий.

На рис. 22.1 для примера приведена схема расположения выводов микроконтроллера AT89LP2052.

Итак, микроконтроллер для программирования может непрерывно удерживаться в текущем режиме, а также снова программироваться и запускаться вновь. Микроконтроллер Atmel программируется как ведомое устройство (Slave).

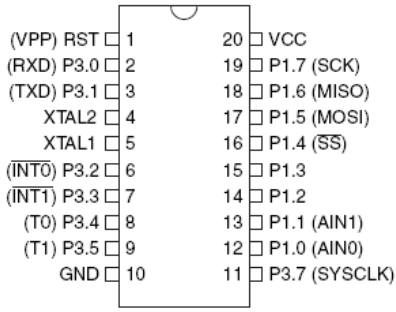


Рис. 22.1

Для этого должна быть соблюдена определенная последовательность сигналов. На рис. 22.1 показаны временные диаграммы процедуры последовательного программирования микроконтроллера. В соответствии со стартовой последовательностью разрешения программирования (Programming Enable), которая служит для активации последовательного внутрисхемного программирования ISP в текущем режиме, передается следующая последовательность сигналов: преамбула AAh (Preamble), операционный код разрешения программирования ACh (Opcode) и старшая часть адреса 53h (Address High) (младшая часть адреса в данном случае не имеет значения). При программировании могут передаваться от 1 до 32 байтов данных за один цикл (Page) с преамбулой (Preamble) AAh, соответствующим операционным кодом (Opcode), адресом (Address) и данными (Data). При страничной передаче после передачи каждого байта данных счетчик адресов автоматически увеличивается на 1. При записи действительные данные, передаваемые по линии MOSI, должны совпадать с фронтом (возрастающим перепадом) тактового синхросигнала SCK. При чтении данных с линии MISO они будут действительными при спаде (отрицательном перепаде) синхросигнала SCK.

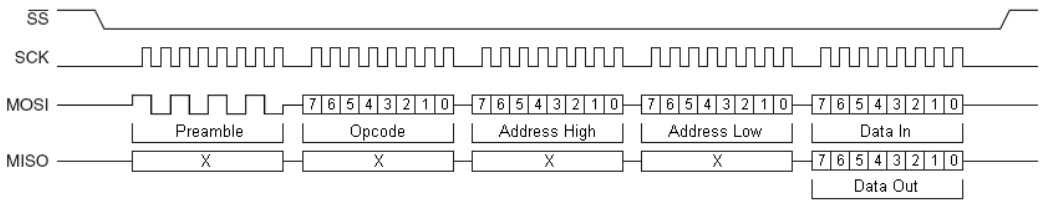


Рис. 22.2

Кроме трех обязательных сигнальных линий при ISP-программировании (SCK, MOSI и MISO) микроконтроллеров Atmel в данном случае используется линия выбора ведомого устройства SS# (Slave Select, которая может совпадать с CS# — Chip Select— выбор кристалла), поскольку микроконтроллер во время программирования является подчиненным (Slave) устройством. В данном случае для сигнала SS# можно использовать обычно незадействованный вывод 8 ISP-разъема (табл. 22.2). Ведомое устройство должно реагировать только на низкий уровень сигнала SS#, который формируется ведомым. При высоком уровне этого сигнала выход микроконтроллера MISO должен быть переведен в высокоимпедансное состояние (SPI-порт ведомого отключается).

В микроконтроллерах AT89LPx052 для программирования флэш-памяти предусмотрены два интерфейса: последовательный и параллельный. Последовательное и параллельное программирование микроконтроллеров имеют одинаковые алгоритмы и команды. Отличается только аппаратное обеспечение. Для параллельного программирования требуется десять сигналов (P1.0—P1.7, XTAL1 и CS#), последовательное использует только 4 SPI-сигнала (SS#, SCK, MOSI, MISO) и лишь при необходимости дополнительный сигнал для сброса, чтобы запрограммировать заново и запустить микроконтроллер в текущем режиме, не прибегая к установке или удалению дополнительной перемычки. Для разрешения параллельного программи-

рования на вывод RST (вывод 1) микроконтроллера AT89LPx052 нужно падать напряжение Vpp, равное 12 В (рис. 22.3).

На рис. 22.4 показаны временные диаграммы последовательности команд при параллельном программировании микроконтроллера.

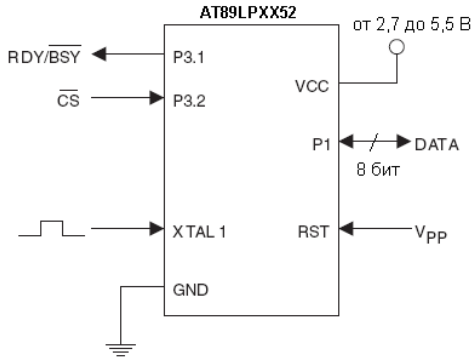


Рис. 22.3

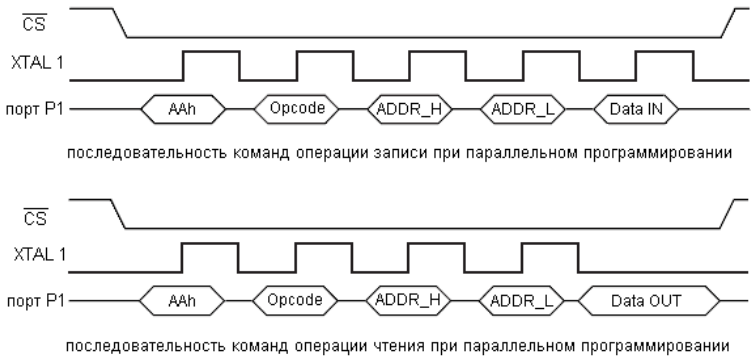


Рис. 22.4

Компания Atmel для адаптера программирования AT89ISP предлагает следующую разводку кабеля ISP-разъема, показанную в табл. 22.2.

Таблица 22.2

Штырьковый вывод	Наименование сигнала	Назначение
1	SCK	Serial Clock — синхросигнал, которым ведущее устройство стробирует каждый бит данных
3	MISO	Master In ← Slave Out — входные данные ведущего устройства (master) и выходные данные ведомого (slave)
4	VCC	Питающее напряжение
5	RST	Reset — сигнал сброса

Таблица 22.2 *(окончание)*

Штырьковый вывод	Наименование сигнала	Назначение
9	MOSI	Master Out → Slave In — выходные данные ведущего устройства (master) и входные данные ведомого (slave)
2, 10	GND	Общий вывод питания ("земля")
6, 7, 8	NC	No Connection — не используются

22.2. Пример на Visual Basic — чтение 2 Кбайт флэш-памяти

В листинге 22.1 описан процесс инициализации и чтения флэш-памяти. В качестве подспорья может послужить спецификация Atmel для ISP-программирования.

Листинг 22.1

```
Private Sub bt_Read2052Flash_Click()
init_programmer ' открыть USB-устройство и установить внутренний контроль
Bang Mode
device_power_up ' последовательность включения питания
strWriteBuffer = ""
command_start ' установить сигнал SS и SCK на низкий уровень
              ' в writeBuffer
  datasend &HAA ' задать последовательность в 24 байта!
  datasend &HAC
  datasend &H53
command_end
strWriteBuffer = strWriteBuffer & Chr$(Databyte) ' небольшая задержка
strWriteBuffer = strWriteBuffer & Chr$(Databyte)
For p% = 0 To 2047 Step 32 ' прочитать 2048 байт 64 страницы из 2052
    command_start ' установить сигнал SS и SCK на низкий лог. уровень
      datasend (&HAA) ' передать преамбулу (preamble)
      datasend (&H30) ' передать код операции (opcode)
                      ' для чтения кода страницы
      datasend (Int(p / 256)) ' передать ст. часть адреса (Adress High)
      datasend (p And &HFC) ' передать мл. часть адреса (Adress Low)
      lngwr_command_end = Len(strWriteBuffer)
  For i = 0 To 255 ' 32·8
    Databyte = Databyte Or (2 ^ SCK) ' сигнал SCK на высокий уровень
    strWriteBuffer = strWriteBuffer & Chr$(Databyte)
    Databyte = Databyte And Not (2 ^ SCK) ' сигнал SCK на низкий уровень
    strWriteBuffer = strWriteBuffer & Chr$(Databyte)
  Next i
```

```
    command_end ' установить сигнал SS на выс. уровень — последний байт
    SendRead_FTDI_Bytes ' FTDI-запись и обратное чтение
                              ' ConvertReadData не содержится
    LoggerList.AddItem & _ ' обращает прочитанные биты в байт
     ConvertReadData_32Bytes(Mid$(strReadBuffer, lngwr_command_end))
    strWriteBuffer = ""
    strReadBuffer = ""
Next p
device_power_down ' выкл. питания и закрыть FTDI-устройство
End Sub
```

Вызов функции `ConvertReadData_32Bytes` не приводится в исходном коде. Функция снова преобразует отдельные биты в буфере записи в читаемые байты (листинг 22.2).

Листинг 22.2

```
' запись в буфер и повторное чтение байтов
Public Function SendRead_FTDI_Bytes() As Boolean
If FT_Write(lngHandle, strWriteBuffer, Len(strWriteBuffer), lng-
                                        BytesWritten) <> FT_OK Then
       LoggerList.AddItem "Write Failed"
GoTo CloseHandle
End If
Do
FtStatus = FT_GetStatus(lngHandle, FT_RxBytes, FT_TxBytes, lngevent_
                                                              get_stat)
     If FtStatus = FT_OK Then
       Else
          If FtStatus = FT_IO_ERROR Then
             LoggerList.AddItem "Get Status IO Error"
             GoTo CloseHandle
          End If
       End If
Loop Until FT_RxBytes = lngBytesWritten
flTimedout = False
flFatalError = False
lngTotalBytesRead = 0
strReadBuffer = ""
Do
   lngBytesRead = 0 ' количество возвращенных байтов
   FtStatus = FT_Read(lngHandle, strReadBuffer, FT_RxBytes, lngBytesRead)
   If (FtStatus = FT_OK) Or (FtStatus = FT_IO_ERROR) Then
       If lngBytesRead > 0 Then
          lngTotalBytesRead = lngTotalBytesRead + lngBytesRead
       Else
          flTimedout = True
       End If
```

```
   Else
       flFatalError = True
   End If
Loop Until (lngTotalBytesRead = FT_RxBytes) Or &_
             (flTimedout = True) Or (flFatalError = True)
SendRead_FTDI_Bytes = True
Exit Function
CloseHandle:
SendRead_FTDI_Bytes = False
intMode = 0
If FT_Close(lngHandle) <> FT_OK Then
LoggerList.AddItem "FT_Close Failed"
End If
End Function
.
' пример инициализации в режиме Bit Bang
Private Function init_programmer() As Boolean
Dim i As Long
LoggerList.Clear
flFailed = True
init_programmer = false
If FT_ListDevices(0, strDescription, FT_LIST_BY_INDEX &_
           Or FT_OPEN_BY_DESCRIPTION) <> FT_OK Then
    LoggerList.AddItem ("ListDevices failed")
    Exit Function
  Else
    LoggerList.AddItem ("Device Description " & strDescription)
End If
' пример инициализации в режиме Bit Bang
If FT_OpenEx(strDescription, FT_OPEN_BY_DESCRIPTION, &_
                                        lngHandle) <> FT_OK Then
    LoggerList.AddItem "Open Failed"
    Exit Function
End If
FtStatus = FT_SetBaudRate(lngHandle, P_Baudrate) ' установить скорость
                                                 ' передачи
If FtStatus <> FT_OK Then
    LoggerList.AddItem "SetBaudRate Failed"
    Exit Function
End If
' установить MOSI, SCK, RST и SS на выход
intMask = &H0 Or (2 ^ MOSI) Or (2 ^ SCK) Or (2 ^ RST) Or (2 ^ SS)
intMode = 4
FtStatus = FT_SetBitMode(lngHandle, intMask, intMode)
If FtStatus <> FT_OK Then
    LoggerList.AddItem "Set Bit Bang Mode failed"
    Exit Function
```

```
  Else
    LoggerList.AddItem "Bit Bang Mode o.k"
End If
lngBytesWritten = 0
Databyte = 0
init_programmer = True
' обработка ошибок и затем init_programmer = False
End Function

' один байт последовательного потока обменивается с Clock
Private Sub datasend(byval data As Byte)
' подготовить байт для последовательных данных в микросхему
' SPI- добавить / сохранить в writebuffer
Dim n As Integer
Dim stelle As Byte
stelle = 128 ' сначала получить данные бита MSB
For n = 1 To 8
   If data And stelle Then
      Databyte = Databyte Or (2 ^ MOSI) ' установить MOSI
    Else
      Databyte = Databyte And Not (2 ^ MOSI) ' очистить MOSI
   End If
  strWriteBuffer = strWriteBuffer & Chr$(Databyte)
  Databyte = Databyte Or (2 ^ SCK) ' один сигнал CLK, SCK на выс. уровень
  strWriteBuffer = strWriteBuffer & Chr$(Databyte)
  Databyte = Databyte And Not (2 ^ SCK) ' сигнал SCK на низкий уровень
  strWriteBuffer = strWriteBuffer & Chr$(Databyte)
  stelle = stelle / 2
Next n
End Sub

Private Sub device_power_up()
FtStatus = FT_GetBitMode(lngHandle, testbyte)
If FtStatus <> FT_OK Then
    LoggerList.AddItem "Bit Bang Mode failed during power up"
    Exit Sub
End If
Databyte = 0
strWriteBuffer = ""
strWriteBuffer = Chr$(Databyte) ' все устанавливаем в низкий уровень
SendRead_FTDI_Bytes
Databyte = Databyte Or (2 ^ RST) Or (2 ^ SS)
strWriteBuffer = Chr$(Databyte)
SendRead_FTDI_Bytes
' немного ожидания (только 2 мс) согласно техническому паспорту
Sleep (20)
End Sub
```

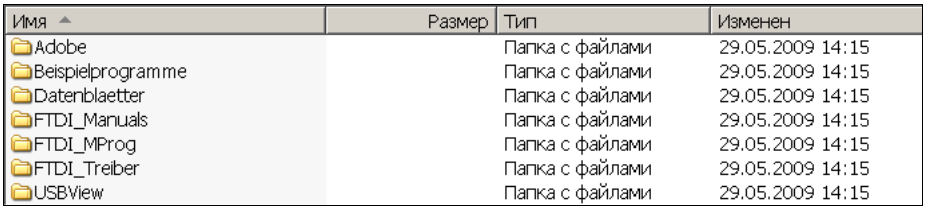
```
Private Sub device_power_down()
Databyte = Databyte And Not (2 ^ SCK)
strWriteBuffer = Chr$(Databyte)
SendRead_FTDI_Bytes
Databyte = Databyte And Not (2 ^ RST) ' сигнал RST на низкий уровень
strWriteBuffer = Chr$(Databyte)
SendRead_FTDI_Bytes
Databyte = Databyte And Not (2 ^ SS)
strWriteBuffer = Chr$(Databyte)
SendRead_FTDI_Bytes
intMask = &H0
intMode = 4
FtStatus = FT_SetBitMode(lngHandle, intMask, intMode)
If FtStatus <> FT_OK Then
    LoggerList.AddItem "Set Bit Bang Mode to inputs failed"
    GoTo CloseHandle
End If
If FT_Close(lngHandle) <> FT_OK Then
    LoggerList.AddItem "Close Failed"
End If
End Sub

Private Sub command_start()
' set hardware conditions for SS and SCK
' сигнал SS и SCK на низкий уровень
  Databyte = Databyte And Not (2 ^ SS) And Not (2 ^ SCK)
  strWriteBuffer = strWriteBuffer & Chr$(Databyte)
End Sub
Private Sub command_end()
' переключить сигнал SS на высокий уровень
    Databyte = Databyte Or (2 ^ SS)
    strWriteBuffer = strWriteBuffer & Chr$(Databyte)
```

ПРИЛОЖЕНИЕ

Описание компакт-диска

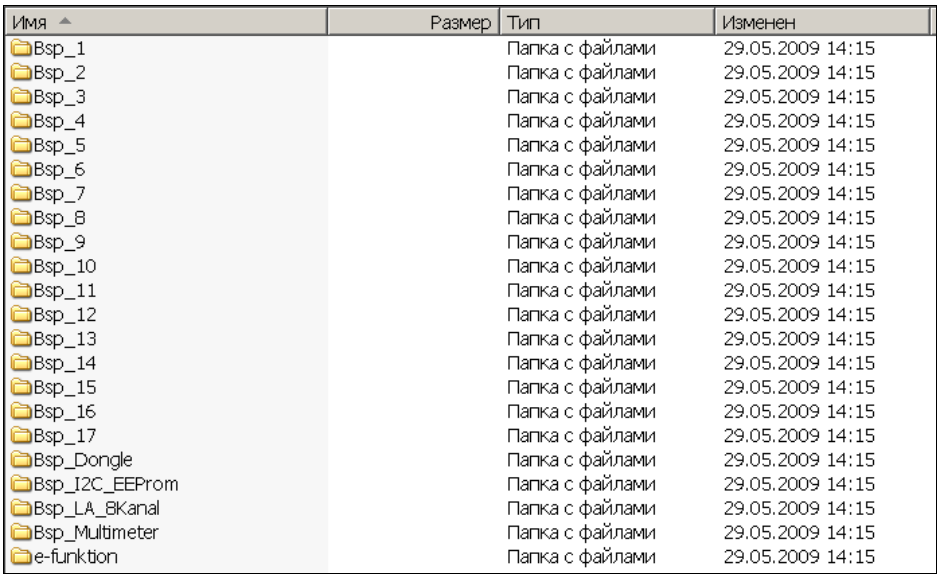
Если вы копируете каталоги (рис. П1) и файлы компакт-диска на жесткий диск, то следует деактивировать защиту от записи файлов.



Имя	Размер	Тип	Изменен
Adobe		Папка с файлами	29.05.2009 14:15
Beispielprogramme		Папка с файлами	29.05.2009 14:15
Datenblaetter		Папка с файлами	29.05.2009 14:15
FTDI_Manuals		Папка с файлами	29.05.2009 14:15
FTDI_MProg		Папка с файлами	29.05.2009 14:15
FTDI_Treiber		Папка с файлами	29.05.2009 14:15
USBView		Папка с файлами	29.05.2009 14:15

Рис. П1

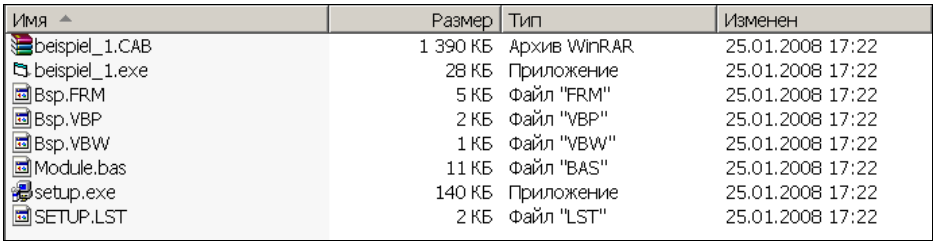
В каталоге \Beispielprogramme (рис. П2) вы найдете отдельные примеры программ и исполняемые exe-файлы.



Имя	Размер	Тип	Изменен
Bsp_1		Папка с файлами	29.05.2009 14:15
Bsp_2		Папка с файлами	29.05.2009 14:15
Bsp_3		Папка с файлами	29.05.2009 14:15
Bsp_4		Папка с файлами	29.05.2009 14:15
Bsp_5		Папка с файлами	29.05.2009 14:15
Bsp_6		Папка с файлами	29.05.2009 14:15
Bsp_7		Папка с файлами	29.05.2009 14:15
Bsp_8		Папка с файлами	29.05.2009 14:15
Bsp_9		Папка с файлами	29.05.2009 14:15
Bsp_10		Папка с файлами	29.05.2009 14:15
Bsp_11		Папка с файлами	29.05.2009 14:15
Bsp_12		Папка с файлами	29.05.2009 14:15
Bsp_13		Папка с файлами	29.05.2009 14:15
Bsp_14		Папка с файлами	29.05.2009 14:15
Bsp_15		Папка с файлами	29.05.2009 14:15
Bsp_16		Папка с файлами	29.05.2009 14:15
Bsp_17		Папка с файлами	29.05.2009 14:15
Bsp_Dongle		Папка с файлами	29.05.2009 14:15
Bsp_I2C_EEProm		Папка с файлами	29.05.2009 14:15
Bsp_LA_8Kanal		Папка с файлами	29.05.2009 14:15
Bsp_Multimeter		Папка с файлами	29.05.2009 14:15
e-funktion		Папка с файлами	29.05.2009 14:15

Рис. П2

В каталоге \Beispielprogramme\Bsp_1 (рис. П3) находится программа инсталляции. Если вы не установили Visual Basic, и исполняемые демонстрационные программы не функционируют из-за отсутствующих VB-библиотек, первым делом запустите setup.exe из каталога \Beispielprogramme\Bsp_1.



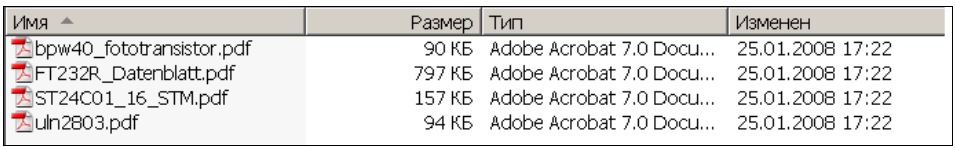
Имя	Размер	Тип	Изменен
beispiel_1.CAB	1 390 КБ	Архив WinRAR	25.01.2008 17:22
beispiel_1.exe	28 КБ	Приложение	25.01.2008 17:22
Bsp.FRM	5 КБ	Файл "FRM"	25.01.2008 17:22
Bsp.VBP	2 КБ	Файл "VBP"	25.01.2008 17:22
Bsp.VBW	1 КБ	Файл "VBW"	25.01.2008 17:22
Module.bas	11 КБ	Файл "BAS"	25.01.2008 17:22
setup.exe	140 КБ	Приложение	25.01.2008 17:22
SETUP.LST	2 КБ	Файл "LST"	25.01.2008 17:22

Рис. П3

После успешной инсталляции вы можете исполнить сгенерированные в этом обучающем пакете exe-файлы.

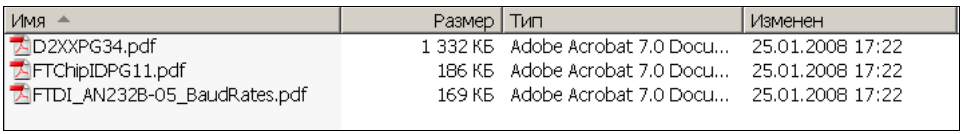
В каталоге \Beispielprogramme\Bsp_Dongle, помимо примера в Visual C++, находятся необходимые библиотеки FTDI-функций для получения идентификатора микросхемы (Chip ID). В каталоге \Beispielprogramme\e-funktion вы найдете программу RC_E_Funktion.exe, которая представляет ход RC-функции *(см. главу 12)*.

В каталогах \Datenblatter (рис. П4) и \FTDI_Manuals (рис. П5) в формате PDF содержатся наиболее важные данные и руководства изготовителей.



Имя	Размер	Тип	Изменен
bpw40_fototransistor.pdf	90 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22
FT232R_Datenblatt.pdf	797 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22
ST24C01_16_STM.pdf	157 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22
uln2803.pdf	94 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22

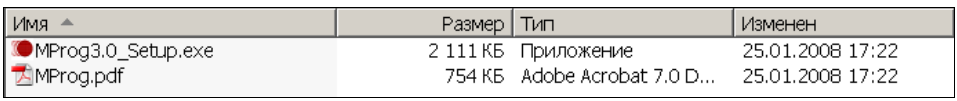
Рис. П4



Имя	Размер	Тип	Изменен
D2XXPG34.pdf	1 332 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22
FTChipIDPG11.pdf	186 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22
FTDI_AN232B-05_BaudRates.pdf	169 КБ	Adobe Acrobat 7.0 Docu...	25.01.2008 17:22

Рис. П5

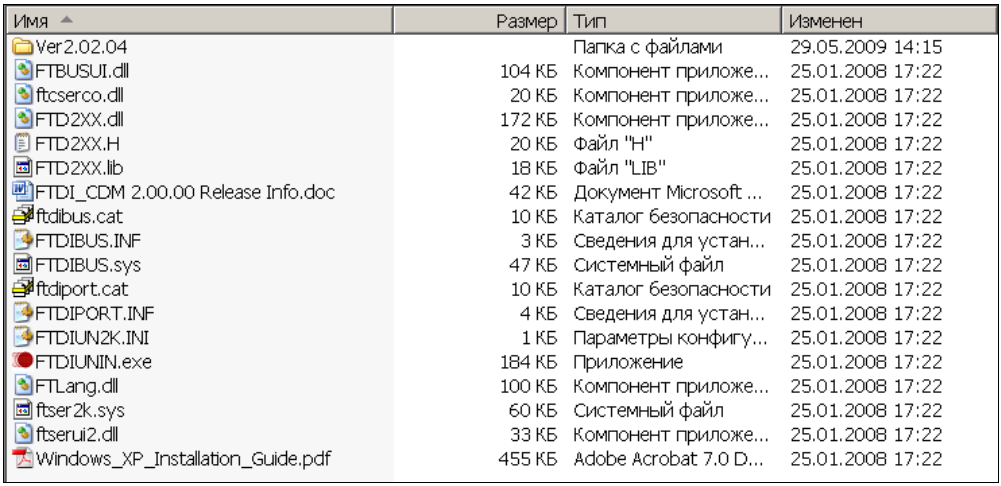
В каталоге \FTDI_MProg (рис. П6) находятся программа и документация для программирования EEPROM-памяти компании FTDI.



Имя	Размер	Тип	Изменен
MProg3.0_Setup.exe	2 111 КБ	Приложение	25.01.2008 17:22
MProg.pdf	754 КБ	Adobe Acrobat 7.0 D...	25.01.2008 17:22

Рис. П6

В каталоге \FTDI_Treiber (рис. П7), который вы должны были выбрать после первого подключения USB-адаптера для инсталляции драйвера, находится также и каталог Ver2.02.04 с драйвером CDM (CDM 2.02.04.exe). Речь идет о непосредственно исполняемом exe-файле для установки драйвера со стандартными идентификаторами VID и PID, как это имеет место у USB-адаптера, предложенного в данной книге.



Имя	Размер	Тип	Изменен
Ver2.02.04		Папка с файлами	29.05.2009 14:15
FTBUSUI.dll	104 КБ	Компонент приложе...	25.01.2008 17:22
ftcserco.dll	20 КБ	Компонент приложе...	25.01.2008 17:22
FTD2XX.dll	172 КБ	Компонент приложе...	25.01.2008 17:22
FTD2XX.H	20 КБ	Файл "H"	25.01.2008 17:22
FTD2XX.lib	18 КБ	Файл "LIB"	25.01.2008 17:22
FTDI_CDM 2.00.00 Release Info.doc	42 КБ	Документ Microsoft ...	25.01.2008 17:22
ftdibus.cat	10 КБ	Каталог безопасности	25.01.2008 17:22
FTDIBUS.INF	3 КБ	Сведения для устан...	25.01.2008 17:22
FTDIBUS.sys	47 КБ	Системный файл	25.01.2008 17:22
ftdiport.cat	10 КБ	Каталог безопасности	25.01.2008 17:22
FTDIPORT.INF	4 КБ	Сведения для устан...	25.01.2008 17:22
FTDIUN2K.INI	1 КБ	Параметры конфигу...	25.01.2008 17:22
FTDIUNIN.exe	184 КБ	Приложение	25.01.2008 17:22
FTLang.dll	100 КБ	Компонент приложе...	25.01.2008 17:22
ftser2k.sys	60 КБ	Системный файл	25.01.2008 17:22
ftserui2.dll	33 КБ	Компонент приложе...	25.01.2008 17:22
Windows_XP_Installation_Guide.pdf	455 КБ	Adobe Acrobat 7.0 D...	25.01.2008 17:22

Рис. П7

Список источников информации

1. USB-справочник для электронщиков
2. **www.wikipedia.de**
3. GSM-Bentheimer Softwarehaus GmbH **www.gms2000.de**, **www.minimikro.de**[1]
4. Спецификации компании Atmel на сайте **www.atmel.com**
5. Сайт компании FTDI, **www.ftdichip.com**
6. B. Kainka, Basiskurs Mikrocontroller
7. Reichelt Elektronik, **www.reichelt.de**
8. Nanotec GmbH, **www.nanotec.de**
9. **www.usb.org**
10. **www.heise.de**

Список источников был тщательно проработан автором. Ни в коем случае не перенимается ответственность за технические и полиграфические ошибки или упущения в данном документе. Кроме того, не перенимается ответственность за повреждения, которые могут возникнуть из-за содержащейся в документе информации.

Изменения содержания оговариваются. Это касается также и изменений объема по форме, оформлению и технике (технологии).

Зарегистрированные товарные знаки, торговые наименования и прочее в этом документе не обозначены в качестве таковых. Они являются собственностью их соответствующих владельцев.

[1] **http://www.atmel.ru**, **www.chip-dip.ru**. — *Прим. пер.*

Предметный указатель

М

Н

О

П

Р

С

Т

У

Ф

Х

Ш

Э

Юрген Хульцебош

USB в электронике

2-е издание

Перевод с немецкого

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Перевод с немецкого	*Владимира Унагаева*
Редактор	*Юрий Рожко*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 29.04.11.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 18,06.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12