А.Ю. Кузьминов

Интерфейс RS232

Связь между компьютером и микроконтроллером

Москва
«Радио и связь»
2004

УДК 681.3
ББК 32.97
К 89

Кузьминов А.Ю.

К 89 Интерфейс RS232. Связь между компьютером и микроконтроллером. – М.: Радио и связь, 2004. – 168 с.: ил.

ISBN 5-256-01715-2.

Изложены принципы обмена информацией между компьютерами и 51-совместимыми микроконтроллерами по интерфейсу RS232. Представлены многочисленные примеры аппаратных средств на современной элементной базе и программного обеспечения обмена по RS232. Наряду со справочными данными, касающимися инициализации интерфейса и команд ввода/вывода, приведен предложенный автором алгоритм (протокол) обмена, позволяющий передавать и принимать информацию компьютером и микроконтроллером со скоростью 115200 бод с высокой надежностью. Программы для компьютера написаны на ассемблере, Бейсике и Кларионе, а для микроконтроллеров – на ассемблере и СИ. Рассмотрены схемные решения и программное обеспечение программирования современных микроконтроллеров – систем на кристалле (ADUC8XX и MSC1210YX и др.), обладающих возможностью программирования в системе, по интерфейсу RS232. Описаны примеры использования RS232 в системах сбора и обработки информации с датчиков, а также в программаторах микроконтроллеров.

Для опытных разработчиков компьютерных и автономных систем сбора и обработки информации, в составе которых используются микроконтроллеры, а также для начинающих специалистов в этой области; может быть полезна студентам вузов соответствующих специальностей.

ББК 32.97

Производственное издание

Кузьминов Алексей Юрьевич

Интерфейс RS232

Связь между компьютером и микроконтроллером

Редактор *Н.Г. Давыдова*
Обложка художника *В.Г. Ситникова*
Художественные и технические редакторы *Л.А. Горшкова, Т.Н. Зыкина*
Корректор *Т.В. Дземидович*
Компьютерная верстка *Р.А. Сафиной*

ИБ 3128

ЛР 010164 от 29.01.97
Подписано в печать 19.03.2004 г. Формат 60×90/16
Усл. печ. л. 10,5 Гарнитура Таймс Печать офсетная
Изд. № 24436 Заказ № 4 Тираж 2000 экз.

www.radiosv.ru
Издательство «Радио и связь». 127473 Москва, 2-й Щемиловский пер., 5/4, стр.1
Типография издательства «Радио и связь».
127473 Москва, 2-й Щемиловский пер., 5/4, стр.1

ISBN 5-256-01715-2

Предисловие

Последовательный интерфейс RS232, разработанный более 25 лет назад для компьютеров (в основном для их связи с модемами), до сих пор не утратил своего коммуникационного назначения. Даже сейчас, в связи с появлением множества других последовательных интерфейсов, обладающих несомненными преимуществами перед этим интерфейсом (например, интерфейсов USB, RS485, RS422, применяемых в компьютерах, и I²C, CAN, SPI, применяемых в микроконтроллерах), интерфейс RS232, похоже, не скоро уйдет в разряд "отставников". Такое положение, на первый взгляд, может показаться странным, особенно из-за недостатков RS232, "глюков" и т.п. Однако следует учесть, что это едва ли не единственное средство связи между компьютером и микроконтроллером, аппаратно присутствующее и в первом и во втором. Во всяком случае, косвенным подтверждением исключительности интерфейса RS232 может служить тот факт, что в современных персональных компьютерах RS232 интегрирован в материнскую плату. Что касается микроконтроллеров, то сейчас трудно найти такой микроконтроллер, в котором бы аппаратно не присутствовал хотя бы один интерфейс RS232 (иногда их бывает и два).

В дальнейшем под словом "микроконтроллер" будем подразумевать 51-совместимый микроконтроллер. Единственный, правда, на наш взгляд, достаточно слабый довод в пользу 51-совместимых микроконтроллеров состоит в том, что они в настоящее время стали де-факто "промышленным стандартом".

В книге собраны многочисленные примеры применения автором интерфейса RS232 как в компьютере, так и в микроконтроллере. Приводятся схемные решения аппаратных средств и программное обеспечение для них.

Автор не претендует на полноту охвата темы, однако многочисленные особенности работы с интерфейсом RS232, приведенные в книге, помогут разработчикам компьютерных и автономных систем (на базе микроконтроллеров) сбора и обработки информации, поступающей с различного рода датчиков, избежать многих ошибок и неясностей. Книга может быть также полезна студентам вузов компьютерного профиля.

С вопросами и предложениями читатели могут обращаться либо по адресу электронной почты автора compmicrosys@mail.ru, либо в издательство.

1. Интерфейс RS232 в компьютере

1.1. Электрические характеристики сигналов на линиях RS232 в компьютере

Интерфейс RS232 является последовательным. Это означает, что данные (информация) передаются последовательно, бит за битом по одному проводу (в отличие от параллельного интерфейса, в котором, например, каждый бит байта передается по отдельному проводу, т.е. байт передается по восьми проводам). Формат посылки – 1 байт данных и несколько управляющих бит, некоторые из которых могут отсутствовать.

Обмен информацией между компьютером и периферийным устройством по интерфейсу RS232 двусторонний, т.е. данные могут передаваться компьютером в периферийное устройство и приниматься компьютером от периферийного устройства.

В компьютере предусмотрен специальный разъем, называемый *коммуникационным* (COM); иногда их бывает два (COM1 и COM2) или более. К разъему подключается кабель, соединяющий компьютер с периферийным устройством. В кабеле находятся несколько проводов, которые называют *линиями интерфейса*. Термин "линия" достаточно условен, так как английское слово line, которому он соответствует, имеет более широкое значение.

На рис.1.1 показан формат данных, посылаемых по линии данных TxD интерфейса RS232. Как следует из рисунка, передача начинается с так называемого старт-бита, затем идут биты данных (их может быть от пяти до восьми), далее следует бит паритета или четности (который может отсутствовать) и затем следуют стоп-биты (их может быть либо один, либо два). На практике используют 8 бит данных; бит паритета (на рис.1.1 не показан), как правило, не используется; при максимально высокой скорости передачи желательно передавать два стоп-бита.

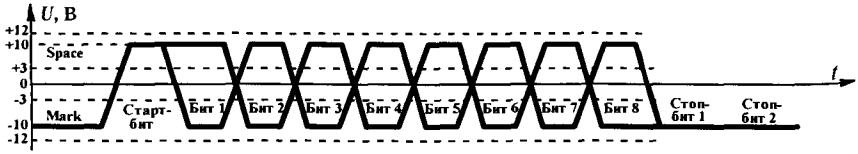


Рис.1.1. Формат байта данных и уровни напряжений в интерфейсе RS232

Состояние линии RS232 при отсутствии передачи называется Mark (отметка), состояние линии при начале передачи (старт-бит) – Space (пробел). Низкий уровень напряжения относительно "земли" на линии в состоянии Mark соответствует от –3 до –15 В, высокий уровень напряжения линии в состоянии Space – от +3 до +15 В. В интервале –3...+3 В состояние линии не определено. На практике в подавляющем большинстве случаев высокий уровень напряжения составляет около +10 В, низкий – около –10 В. В редких случаях напряжения могут быть снижены соответственно до +5 и –5 В. Это бывает, когда от интерфейса не требуется особенно высокая скорость передачи, а устройство, обменивающееся информацией с компьютером по RS232, имеет батарейное питание (как правило, литиевую батарейку напряжением около 3 В) и критично к расходу энергии.

Условимся называть *логическим нулем* (лог. 0) низкий уровень напряжения на линии в состоянии Mark (–10 В) и *логической единицей* (лог.1) – высокий уровень на линии в состоянии Space (+10 В). Как будет видно из дальнейшего изложения, эта условность оправдана несколькими причинами. Во-первых, стандартный высокий TTL-уровень (см. далее), т.е. лог.1, соответствует +5 В, низкий, т.е. лог. 0, – 0 В. Аналогия по уровням напряжения очевидна. Во-вторых, *при подаче высокого TTL-уровня (лог.1) на вход передатчика RS232 (гл. 3) на его выходе получаем низкий уровень напряжения (лог. 0).* По этой причине очень часто состояние Mark называют логической единицей, что приводит к путанице. И, наконец, в-третьих, при программировании интерфейса RS232 в компьютере, в частности *для установки*, например, на линии *DTR высокого уровня* напряжения, *необходимо установить бит, управляющий линией DTR, в 1* (out 3fch,1 – п.1.3.2).

Помимо уровней напряжений линии интерфейса RS232 характеризуются также скоростью изменений состояний из логической единицы в логический нуль и обратно, т.е. длительностью фронта и длительностью спада сигнала. Эти длительности тесно связаны со скоростью обмена, которая измеряется в битах в секунду (или в бодах). Максимально возможная скорость обмена по RS232 составляет 115200 бод (для компьютера). Если используется посылка из 8 бит данных без контроля по паритету и с одним стоп-битом, то вместе со старт-битом такая посылка будет состоять из 10 бит. Нетрудно подсчитать, что в этом случае время передачи каждого бита будет равно около 8,68 мкс. Обычно производители микросхем – преобразователей уровней RS232 оговаривают либо скорость обмена конкретного преобразователя, либо скорость изменения сигналов из логической единицы в логический нуль и обратно. (Мы еще вернемся к этому вопросу при обсуждении

конкретных микросхем преобразователей уровней RS232.) Эта скорость в большинстве случаев должна быть не менее 30 В/мкс.

Рассмотрим передачу по линии единичного импульса, длительность которого соответствует времени передачи 1 бита при скорости обмена 115200 бод. При уровнях логической единицы и логического нуля соответственно +10 и –10 В и скорости изменения состояний линии из логического нуля в логическую единицу и обратно, составляющей 30 В/мкс, длительность фронта и спада импульса будет равна (каждая) около 0,67 мкс (в сумме 1,33 мкс). Таким образом, длительность состояния линии в логической единице будет 8,68 – 1,33 = 7,35 мкс, что соответствует приблизительно 84,7% длительности всего импульса. Превышение длительностей фронта и спада импульса и, как следствие, уменьшение указанного процентного соотношения может привести к срыву обмена и потере информации.

Необходимо отметить, что далеко не все выпускаемые микросхемы преобразователей уровней интерфейсов RS232 могут обеспечить такую высокую скорость обмена, хотя некоторые из них могут работать значительно быстрее (до 460 кбод). Кроме того, следует также знать, что скорость обмена 115 кбод могут обеспечить только высокоскоростные оптроны (если речь идет о гальванических развязках). (Этот вопрос будет еще не раз обсуждаться в гл. 3.) На практике чаще всего используются три скорости обмена: 9600, 115200 и (реже) 57600 бод .

1.2. Контакты разъемов RS232 в компьютере

В компьютере могут присутствовать как 25-штырьковый (DB25), так и 9-штырьковый (DB9) разъемы RS232. В табл.1.1 приведены названия сигналов и соответствующие им номера контактов обоих типов разъемов. Как видно из таблицы, разъем содержит контакты как входных линий, так и выходных.

Таблица 1.1

Контакты разъемов интерфейса RS232 в компьютере

Номер контакта		Название сигнала	Расшифровка	Тип линий
DB25	DB9			
2	3	TxD	Transmitter Data – передатчик данных	Выходная
3	2	RxD	Receiver Data – приемник данных	Входная
4	7	RTS	Request To Send – запрос передачи	Выходная
5	8	CTS	Clear To Send – сброс передачи	Входная
6	6	DSR	Data Set Ready – готовность данных	Входная
7	5	SG	Signal Ground – сигнальная земля	–
8	1	DCD	Data Carrier Detect – обнаружение несущей	Входная
20	4	DTR	Data Terminal Ready – готовность терминала	Выходная
22	9	RI	Ring Indicator – индикатор звонка	Входная

Основными линиями, по которым осуществляется обмен данными, являются две: TxD – линия, по которой из компьютера передаются данные во внешнее устройство, и RxD – линия, по которой компьютером принимаются данные из внешнего устройства. Сигналы на этих линиях изменяются в соответствии с рис.1.1.

Линии DTR и RTS являются выходными. Это означает, что уровнями сигналов на этих линиях можно управлять, устанавливая биты соответствующих регистров в нуль или единицу программным способом.

Линии CTS, DSR, DCD и RI являются входными. Это означает, что состояния этих линий можно проверять (т.е. выяснять, в каком состоянии – нулевом или единичном они находятся), читая соответствующие регистры состояний и выделяя соответствующие биты (§ 1.3).

Необходимо отметить следующие свойства линий TxD и RxD. Линия TxD является выходной. Помимо того, что по ней передаются данные, в отсутствие передачи состоянием этой линии можно также управлять программно, т.е. устанавливать в единичное или нулевое состояние. Линия RxD является входной. Однако прочитать состояние этой линии (как линий CTS, DSR, DCD и RI) при отсутствии передачи нельзя!

Кроме того, заметим, что линии DTR, RTS, CTS, DSR, DCD и RI называют еще линиями квитирования (иногда модемными, так как они используются в модемах). Существует как множество алгоритмов обмена по RS232, в которых эти линии (или некоторые из них) используются, так и множество алгоритмов обмена, в которых эти линии не используются вообще (задействованы только линии RxD и TxD). В связи с этим аббревиатура линий квитирования достаточно условна, и поэтому детальное рассмотрение функций (и названий) линий квитирования без рассмотрения конкретного алгоритма обмена лишено всякого смысла. Этот вопрос мы обсудим при рассмотрении предложенного автором достаточно простого и надежного алгоритма обмена, особенно эффективного при использовании гальванических развязок (§ 6.2).

1.3. Программирование интерфейса RS232

Программирование интерфейса RS232 в компьютере состоит из нескольких этапов.

1.3.1. Выбор языка программирования

Выбор языка программирования должен определяться, прежде всего, поставленной целью. Кроме того, язык должен, на наш взгляд, содержать встроенные функции ввода-вывода через порт.

Если целью является проверка работы конкретного устройства или его, например, интерфейсной части – "погонять" его на высоких скоростях обмена, – то желательно использовать наиболее "скоростной" язык, т.е. ассемблер, либо Бейсик с ассемблерными вставками. Для более простых применений достаточно только Бейсика. И, наконец, если на компьютере пишется сложная программа, которая, например, обслуживает систему сбора информации, имеет множество меню, поддержку мыши, большое число экранных форм и форм для распечатки информации на бумаге, то мы считаем, что следует использовать такой язык программирования, как Кларион, с помощью которого можно выполнить все перечисленные функции и который обладает достаточно высоким быстродействием, а самое главное, – встроенными функциями ввода-вывода через порт (командами in и out).

В связи с вышесказанным фрагменты программ будут приводиться именно на этих языках в зависимости от контекста. Спорить и указывать достоинства и недостатки этих и других языков программирования – дело неблагодарное и бессмысленное, тем более, что, с одной стороны, писать программы на том или ином языке – это дело вкуса, а с другой – команды ввода-вывода через порты (in и out) на всех языках выглядят однотипно. А именно эти команды в первую очередь, на наш взгляд, необходимо правильно применять для программирования работы последовательного порта компьютера. Что касается встроенных функций DOS, пользуясь которыми также можно запрограммировать последовательный порт, то либо их недостаточно, либо они имеют ограниченную область применения, либо не всегда выполняют заданные действия.

1.3.2. Управление состояниями и чтение состояний линий RS232

В подавляющем большинстве компьютеров имеются два последовательных порта с интерфейсом RS232: COM1 и COM2, реже встречаются компьютеры, оснащенные четырьмя последовательными портами: COM1, COM2, COM3 и COM4. Базовые адреса портов следующие: COM1 – 3f8h, COM2 – 2f8h, COM3 – 3e8h и COM4 – 2e8h. Порт COM1 занимает адресное пространство от 3f8h до 3ffh.

Все программы будут приводиться для порта COM1. Адаптировать программы к другим портам не составляет большого труда. В скобках дано международное название регистра порта. Рассмотрим регистры порта COM1 более детально.

Регистр данных (data register). **Адрес 3f8h.** Доступен по записи и по чтению. Используется для двух целей.

1. Ввод из порта и вывод в порт байта данных.

Пример 1.1. Вывести байт 41h (код латинской прописной буквы A) в порт RS232.

ассемблер	Бейсик	Кларион

mov al,41h
mov dx,3f8h
out dx,al
	out &h3f8,&h41	b byte
		CODE
		b=41h
		out(3f8h,b)

Пример 1.2. Ввести байт данных из порта

ассемблер	Бейсик	Кларион
mov dx,3f8h in al,dx ;Введенный байт в ; регистре al.	byte%=inp(&h3f8) 'Введенный байт в переменной 'byte%.	b byte CODE !Введенный байт в in(3f8h,b) !переменной b.

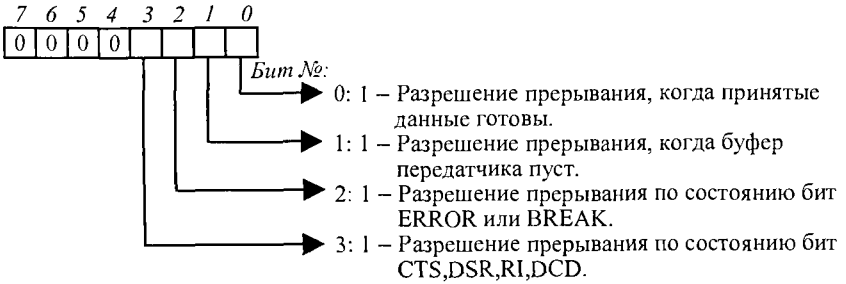
2. Установка скорости обмена по интерфейсу RS232. Скорость обмена по интерфейсу устанавливается следующим образом. Вначале в порт с адресом 3fbh (см. далее) необходимо записать байт, равный 80h, далее в порт с адресом 3f8h – младший байт делителя максимальной скорости обмена 115200 бод, а затем в порт 3f9h – старший байт делителя. Как правило, старший байт делителя используется достаточно редко (для очень медленных скоростей обмена), поэтому он должен быть равен нулю.

Если в младший байт (3f8h) записать 1, то скорость обмена будет равна, как указывалась, 115200 бод. При записи 2 имеем: 115200/2 = 57600 бод; при записи 3 – 115200/3 = 38400 бод; при записи 12 – 115200/12 = 9600 бод и т.п.

Пример 1.3. Установить скорость обмена 9600 и 115200 бод. Поскольку эта операция осуществляется только один раз – при инициализации порта, она не требует высокой скорости и приводится только на Бейсике и Кларионе.

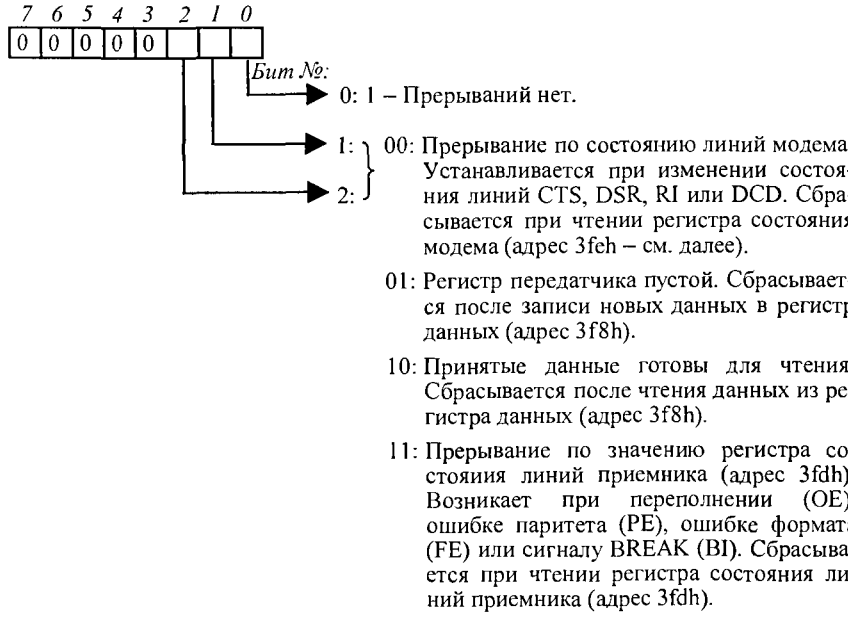
Бейсик	Кларион
out &h3fb,&h80 out &h3f8,12 'Установка скорости out &h3f9,0 'обмена 9600 бод.	b byte CODE b=80h out(3fbh,b) !Установка скорости b=1 !обмена out(3f8h,b) !115200 бод. b=0 out(3f9h,b)

Регистр разрешения прерываний (interrupt enable register). **Адрес 3f9h.** Доступен только для записи. Для того чтобы разрешить перечисленные ниже прерывания, необходимо записать в этот регистр байт, содержимое которого приводится в следующей таблице.

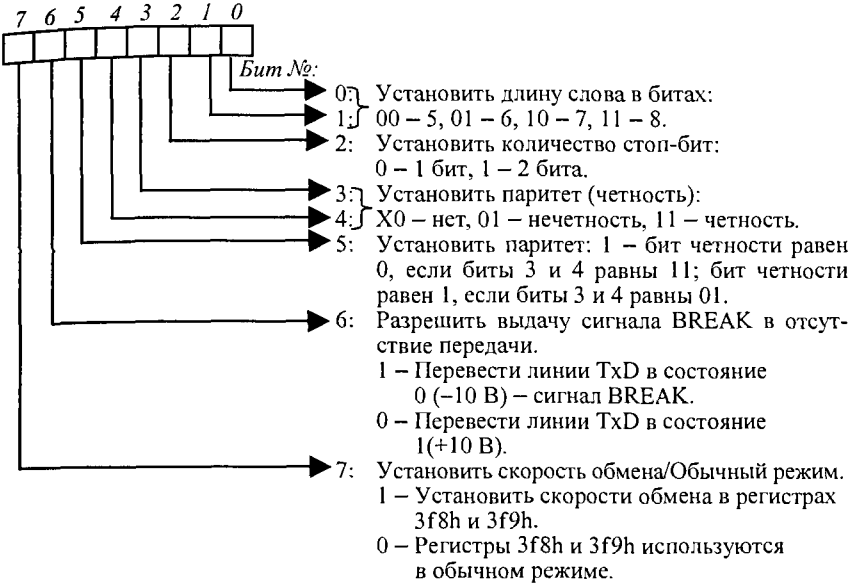


Отметим, что для того, чтобы задействовать этот регистр как регистр разрешения прерываний, а не как старший байт делителя для установки скорости обмена, имеющий тот же адрес (3f9h), нужно вначале обнулить бит 7 в порту с адресом 3fbh (см. с.11).

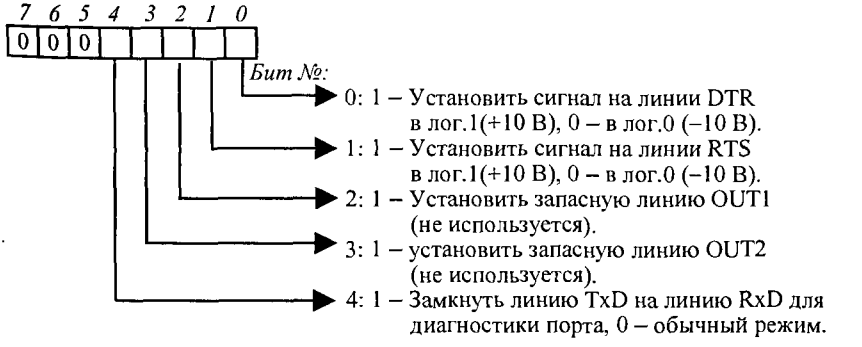
Регистр идентификации прерывания (interrupt identification register). **Адрес 3fah.** Доступен только для чтения. Читая содержимое трех младших бит этого регистра, можно определить источник прерывания.



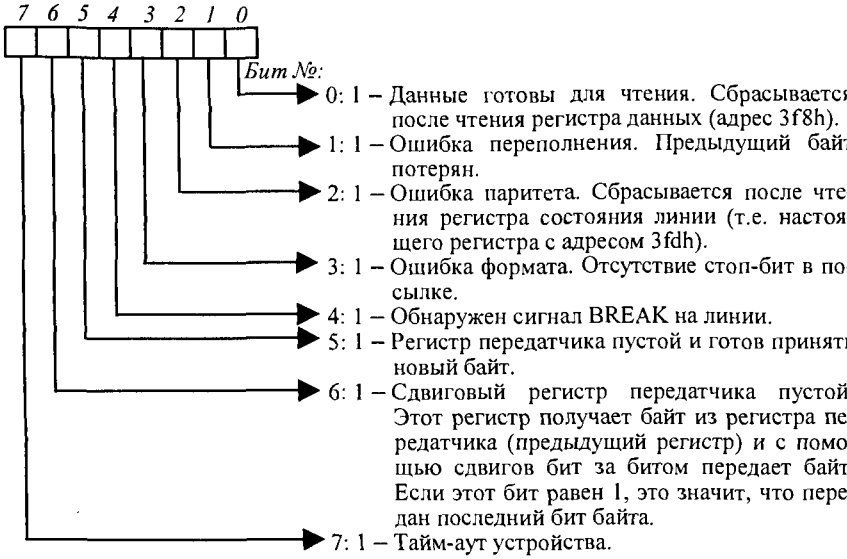
Регистр управления линиями (line control register). **Адрес 3fbh.** Доступен по записи и чтению. Определяет формат передаваемых данных, контроль паритета, количество стоп-бит, управляет линией TxD при отсутствии передачи и устанавливает скорость обмена.



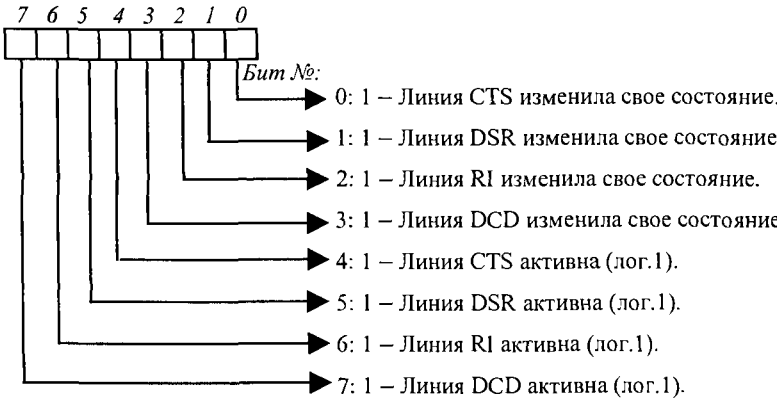
Регистр управления модемом (modem control register). **Адрес 3fch.** Доступен по записи. Управляет линиями DTR и RTS и диагностикой интерфейса RS232.



Регистр состояния линии (line status register). **Адрес 3fdh.** Доступен по чтению. Контролирует правильность обмена по интерфейсу RS232.



Регистр состояния модема (modem status register). **Адрес 3feh.** Доступен по чтению. Определяет состояние линий квитирования интерфейса RS232.



1.3.3. Инициализация интерфейса RS232

Инициализация порта является достаточно простой, но очень важной процедурой. Здесь главное – все установить правильно и ничего не забыть.

Несколько слов о прерываниях. Обмен данными по интерфейсу RS232 между компьютером и периферийным устройством по сравнению со скоростью работы компьютера происходит очень

медленно. Даже если скорость обмена максимальная (115200 бод), частота следования бит 115 кГц. По сравнению с *тактовой* частотой компьютера, которая составляет от 100 МГц (достаточно старенький компьютер) до 3000 МГц (современный), частота 115 кГц на несколько порядков ниже. Этот очевидный факт часто приводят как основной аргумент в пользу применения прерываний, которые может генерировать последовательный порт (см. "Регистр разрешения прерываний" и "Регистр идентификации прерываний"). На практике, однако, это не так.

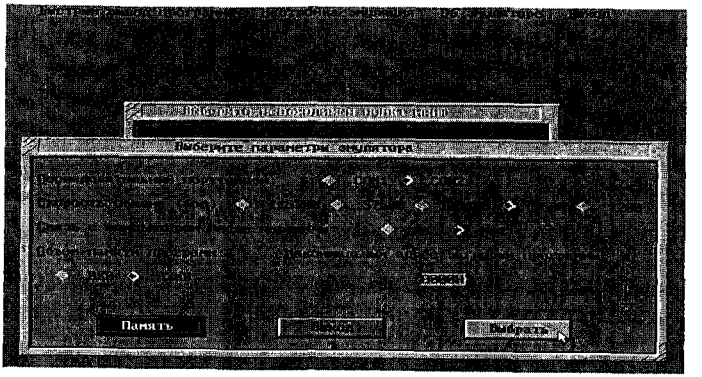
Дело в том, что компьютер "загружен" множеством прерываний, и чем сложнее операционная система, тем прерываний больше. Часто это приводит к самопроизвольному "зависанию" компьютера даже без всякого обмена по RS232. Кроме того, аргументы, приводящиеся в пользу прерываний, в основном касаются того факта, что во время медленной операции, каковой является обмен по RS232 (а точнее, в перерывах), компьютер может выполнять массу другой работы. Но другой работы во время обмена по RS232 от компьютера, как правило, не требуется! И еще. Скорость обмена 115200 бод не является такой уж маленькой для микроконтроллера, с которым обменивается информацией компьютер (напомним, что в книге основное внимание уделено именно такому назначению RS232), особенно если при этом микроконтроллер работает с достаточно низкочастотным кварцевым резонатором. При использовании прерываний (пусть даже только в компьютере) либо может сорваться связь, либо может передаться неверный байт. Если каждый байт, передаваемый из компьютера в микроконтроллер, является, например, байтом программы микроконтроллера, которая после ее приема микроконтроллером будет немедленно исполняться, то можно себе представить во что это может "вылиться" в дальнейшем!

В связи с вышеизложенным пользоваться прерываниями интерфейса RS232 мы не станем. Прерывания должны быть запрещены (будем считать термин "запрещение" состоянием регистра разрешения прерываний).

Несколько слов о формате данных, бите паритета и количестве стоп-бит. Хотя порт компьютера и поддерживает длину слова от 5 до 8 бит (см. "Регистр управления линиями"), уже достаточно давно длина слова всегда принимается равной 8 битам. Бит паритета также не используется (из-за его неэффективности), а вот количество стоп-бит целесообразно выбирать в зависимости от скорости обмена. При низких скоростях обмена (например, 9600 бод) достаточно одного стоп-бита. При высоких скоростях (57600 и 115200 бод) для более надежного обмена количество стоп-бит желательно устанавливать равным двум.

Итак, проинициализируем порт интерфейса RS232 в компьютере.

Вариант 1



Вариаит 2

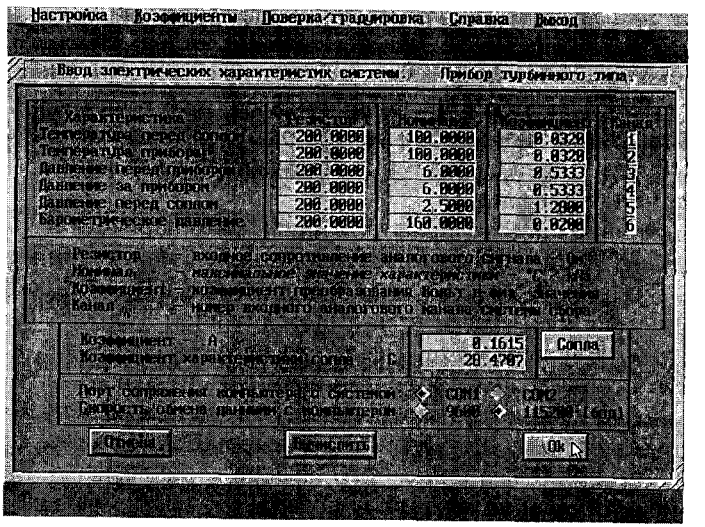
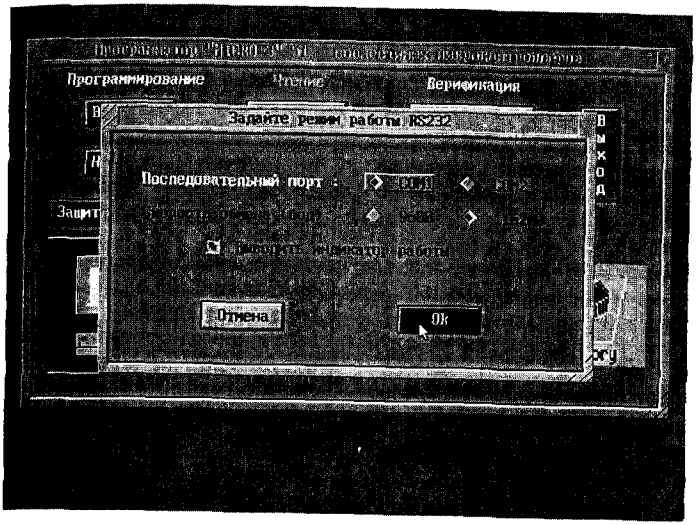


Рис.1.2. Варианты экранных форм выбора режимов работы интерфейса RS232

Вариант 3



Вариант 4

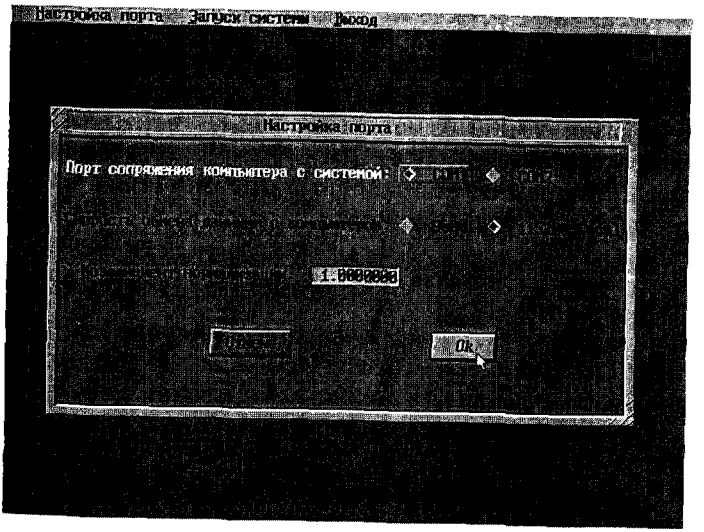


Рис.1.2. (*Продолжение*)

Пример 1.4. Проинициализировать порт компьютера COM1 со следующими параметрами: 1) прерывания запрещены, 8 бит данных, нет паритета, один стоп-бит, скорость 9600 бод; 2) прерывания запрещены, 8 бит данных, нет паритета, два стоп-бита, скорость 115200 бод.

Бейсик	Кларион
'Установка скорости обмена 9600 бод. '---------- out &h3fb,&h80 out &h3f8,12 out &h3f9,0 '---------- 'Установка формата данных out &h3fb,3 '8 бит,1стоп-бит, нет паритета. out &h3f9,0 ' Запретить прерывания.	b byte CODE ! Установка скорости обмена 115200 бод. b=80h; out(3fbh,b) b=1; out(3f8h,b) b=0; out(3f9h,b) ! Установка формата данных. b=7; out(3fbh,b) !8бит, 2стоп-бита, нет паритета. b=0; out(3f9h,b) !Прерывания запрещены.

Необходимо отметить, что после включения компьютера линии TxD, DTR и RTS интерфейса устанавливаются в состояние логического нуля (–10 В).

При разработке прикладной программы для пользователя последний должен иметь возможность инициализировать последовательный порт. Как правило, пользователю предоставляется возможность выбирать номер COM-порта и скорость обмена. Вариантов экранных форм (меню) может быть несколько (рис.1.2). Инициализация порта может входить составной частью в некоторые начальные установки (варианты 1 и 2) , либо использоваться самостоятельно (варианты 3 и 4).

Представленные экранные формы взяты из некоторых наших программ, написанных в операционной системе DOS на языке Кларион (Clarion V3.100) с использованием утилит графического пользовательского интерфейса (GUI). Никаких вставок на ассемблере или Си программы не имеют. Программы могут вызываться непосредственно из операционных систем Win'95, Win'98 (но не Win'XP).

2. Интерфейс RS232 в микроконтроллере

2.1. Электрические характеристики RS232 в микроконтроллере

В микроконтроллере обмен по интерфейсу RS232 осуществляется по линиям TxD (передатчик) и RxD (приемник). Уровни напряжения на этих линиях соответствуют стандартным (цифровым) уровням напряжения микроконтроллера. Это означает, что уровень напряжения логической единицы соответствует напряжению питания микроконтроллера (3 или 5 В), уровень напряжения логического нуля – нулевому напряжению (или "земле"). Обычно уровни напряжений питания и земли называют TTL-уровнями, хотя в настоящее время это понятие значительно видоизменилось (с электрической точки зрения), а аббревиатура TTL (транзисторно-транзисторная логика) давно утратила свой первоначальный смысл. Отметим, что для сопряжения со стандартными уровнями напряжения сигналов на линиях интерфейса RS232 (приблизительно равными ±10 В, как было указано ранее) необходимо использовать преобразователи уровней RS232.

В микроконтроллере, так же, как и в компьютере, есть возможность программно устанавливать скорость обмена, формат данных и некоторые другие характеристики интерфейса RS232. Однако эти характеристики напрямую зависят от частоты используемого кварцевого резонатора, от таймера (а их в микроконтроллере может быть несколько), от еще некоторых устройств микроконтроллера, а также от самого микроконтроллера. Кроме того, микроконтроллер может содержать два интерфейса RS232.

2.2. Использование сигналов интерфейса RS232 для запуска и программирования микроконтроллера

В любом микроконтроллере (так же, кстати, как и в любом микропроцессоре) есть вывод, который называется RESET, низкий уровень напряжения сигнала на котором запускает микроконтроллер, а высокий – полностью останавливает и блокирует его работу. Кроме того, при определенных условиях (а точнее, при определенных дополнительных уровнях напряжения сигналов на выводах микроконтроллера, помимо вывода RESET) микроконтроллер может быть переключен в режим программирования. Еще несколько

лет назад выполнить программирование микроконтроллера можно было только с помощью специального программатора, который программировал микроконтроллер по параллельному интерфейсу (или в параллельном режиме).

Программирование микроконтроллера в параллельном режиме заключается в том, что при определенных состояниях сигналов на выводах микроконтроллера, предназначенных для параллельного режима программирования, на шину адреса выставляется адрес памяти микроконтроллера, а на шину данных – байт данных, подлежащий записи в память программ микроконтроллера. После установки адреса и данных путем манипулирования сигналами на определенных выводах микроконтроллера данные можно записать в память программ. Подобный режим программирования возможен и в современных микроконтроллерах для совместимости их с достаточно старыми микроконтроллерами.

В последнее время появились микроконтроллеры, которые могут быть запрограммированы по последовательному интерфейсу RS232. Такой режим программирования микроконтроллера еще называют *последовательным* или *программированием в системе* (in system programming). Это название режима программирования появилось в связи с тем, что такие микроконтроллеры совершенно необязательно вытаскивать из панельки, расположенной в готовом устройстве, вставлять (микроконтроллер) в программатор, программировать, а затем опять возвращать на место.

Кстати, если микроконтроллер изготовлен в планарном корпусе и впаян в плату, программирование его с помощью программатора, естественно, исключено.

Необходимо добавить, что программирование микроконтроллера в системе или последовательный режим программирования необязательно означает программирование по интерфейсу RS232. Существуют и другие последовательные интерфейсы (например, SPI, I^2C), и другие микроконтроллеры, которые можно запрограммировать, пользуясь этими интерфейсами. Примером могут служить 51-совместимые микроконтроллеры фирмы ATMEL, имеющие в названии суффикс "S"(AT89S51, AT89S52, AT89S53, AT89S8252), которые могут быть запрограммированы при помощи интерфейса SPI. По этому же интерфейсу могут быть запрограммированы микроконтроллеры фирмы GOAL (VERSA DSP, VERSA HV100/HV300, VERSA1, VERSA MIX). Микроконтроллеры фирмы CYGNAL (все, кроме C8051F3XX) могут быть запрограммированы с помощью интерфейса JTAG (это физически – SPI, но с дополнительными свойствами), а C8051F3XX – с помощью интерфейса I^2C.

Впервые, насколько это известно автору, 51-совместимые микроконтроллеры, имеющие возможность программирования по интерфейсу RS232, были выпущены фирмой Dallas (1993 г.). Эти

микроконтроллеры содержат программу-загрузчик (bootstrap loader), располагающуюся в недоступном пользователю месте памяти микроконтроллера. Программа-загрузчик имеет много различных опций (например, запись/чтение памяти программ, запись/чтение памяти данных и различных регистров и т.п.).

В последнее время фирма Analog Devices выпустила целое семейство 51-совместимых микроконтроллеров ADUC8XX, в некоторые из которых входят 16- или 24-разрядные сигма-дельта АЦП с самокалибровкой, низкочастотные фильтры, память программ объемом до 62 кбайт и многие другие новшества. Программирование микроконтроллеров ADUC8XX может производиться как в параллельном режиме, так и в последовательном – по интерфейсу RS232.

Фирма Texas Instruments (а точнее, влившаяся в нее фирма Burr-Brown) также выпустила семейство микроконтроллеров MSC1210YX, содержащих 24-разрядный сигма-дельта АЦП с самокалибровкой, низкочастотные фильтры, память программ объемом до 32 кбайт. Программирование этих микроконтроллеров может быть произведено как по RS232, так и в параллельном режиме.

Может сложиться мнение, что если микроконтроллер оборудован устройством программирования в последовательном режиме, то использовать параллельный режим совершенно бессмысленно, и про него (режим) можно забыть. Однако это не совсем так. Дело в том, что некоторые микроконтроллеры имеют специальные биты защиты памяти программ от несанкционированного доступа (или секретности), в том числе бит, *запрещающий* последовательный режим программирования. Так вот, снять этот бит защиты можно только в режиме параллельного программирования (правда, путем стирания всей памяти микроконтроллера). Так что режим параллельного программирования не надо сбрасывать со счетов. Теперь несколько слов о последовательном режиме программирования. Как уже указывалось, для запуска и остановки микроконтроллера используется вывод RESET. При программировании микроконтроллера в последовательном режиме используется еще один сигнал, который, как правило, подается на вывод PSEN микроконтроллера. Сигналы RESET и PSEN, таким образом, в различных комбинациях (а их, понятно, четыре) определяют режим работы микроконтроллера.

Обратимся к интерфейсу RS232. В RS232 компьютера есть только два (выходных) сигнала управления, которыми можно программно манипулировать. Это сигналы DTR и RTS. Если устройство, сопряженное с компьютером по RS232, необходимо только запускать и останавливать, то для этого достаточно одного сигнала, например DTR, который следует предварительно преобразовать в уровень TTL. Соответствующую линию следует подключить

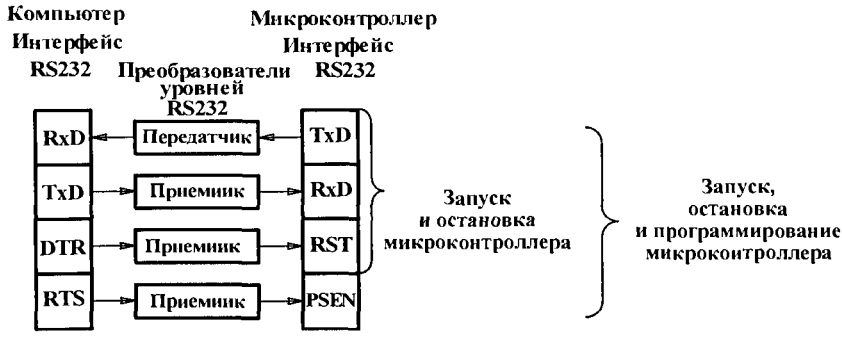


Рис. 2.1. Структурная схема сопряжения компьютера и микроконтроллера по RS232

к выводу RESET. Если же микроконтроллер, находящийся в устройстве, необходимо еще и программировать, то нужно использовать второй управляющий сигнал – RTS (подключив соответствующую линию к выводу PSEN). Кроме сигналов DTR и RTS, естественно, с микроконтроллером необходимо сопрячь еще и основные сигналы, по которым происходит обмен информацией: TxD и RxD.

Таким образом, получаем схему сопряжения микроконтроллера с компьютером (рис. 2.1). Из схемы следует, что для запуска (и остановки) микроконтроллера требуются *два приемника RS232 и один передатчик*; для запуска и программирования микроконтроллера нужно уже *три приемника и один передатчик*. Этой схемой мы будем неоднократно пользоваться в дальнейшем при рассмотрении как аппаратных, так и программных средств сопряжения микроконтроллера с компьютером по RS232.

3. Микросхемы преобразователей уровней интерфейса RS232

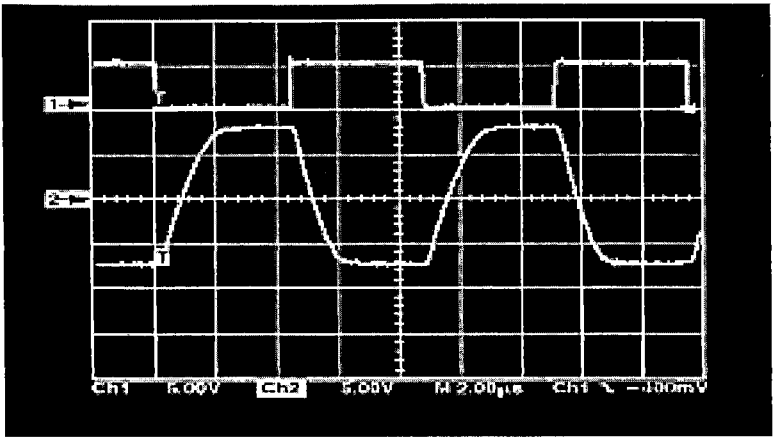
3.1. Свойства и параметры преобразователей уровней RS232

Какими же свойствами должен обладать преобразователь уровней интерфейса RS232, чтобы обеспечить безукоризненный обмен данными между компьютером и микроконтроллером?

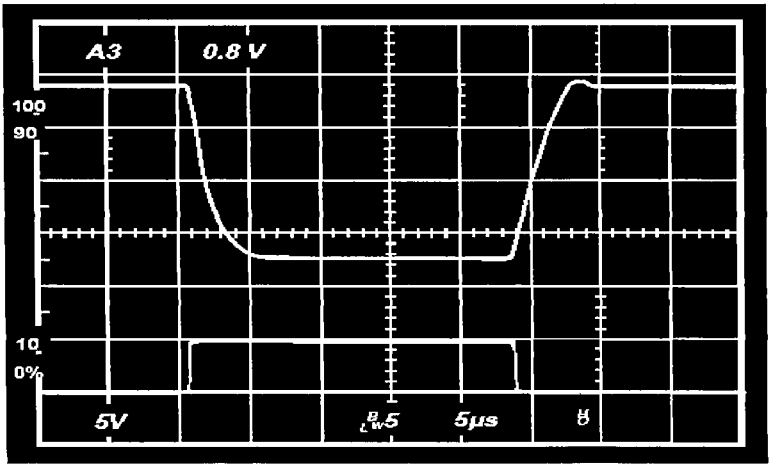
Как уже говорилось, микросхемы преобразователей уровней интерфейса RS232 могут преобразовывать стандартный TTL-сигнал в уровни RS232 и обратно. Если TTL-сигнал преобразовывается в уровни RS232, то такой преобразователь называется *передатчиком RS232*, в противном случае – *приемником RS232*. Как передатчик, так и приемник обладают определенными амплитудно-временными характеристиками и, кроме того, нагрузочной способностью. Причем амплитудно-временные характеристики преобразователя в значительной степени зависят от нагрузки (активной и реактивной; как правило, реактивной является емкостная нагрузка), подключаемой к его выходу. Обычно производители подобных микросхем оговаривают максимальную нагрузочную способность конкретного преобразователя или ток, который он может отдать в нагрузку. Кроме того, указывается максимальный размах напряжения на входе преобразователя (это больше относится к приемникам RS232). Максимальный размах напряжения на входе приемника обычно не должен превышать ±15 В. Во многих преобразователях гарантируется их исправная работа и при значительно бóльших размахах напряжения (до ±30 В).

Что касается нагрузочной способности как передатчика, так и приемника, то этот параметр очень важен и будет использоваться в дальнейшем. Как правило, ток, отдаваемый в нагрузку как приемником, так и передатчиком, составляет единицы миллиампер (иногда – до 10 мА).

Теперь обсудим временны́е характеристики. На рис. 3.1 приведены осциллограммы работы передатчика (*а*) и приемника (*б*) RS232, снятые непосредственно с осциллографа. На рис. 3.2 и 3.3 – более подробные (идеальные) их начертания с указанными временными соотношениями. На верхнем графике рис. 3.2 показана временная диаграмма входного напряжения передатчика, а на нижнем – выходного напряжения передатчика. Как видно из этих



а



б

Рис. 3.1. Осциллограммы напряжений типичного преобразователя уровней интерфейса RS232:
а – передатчика; *б* – приемника

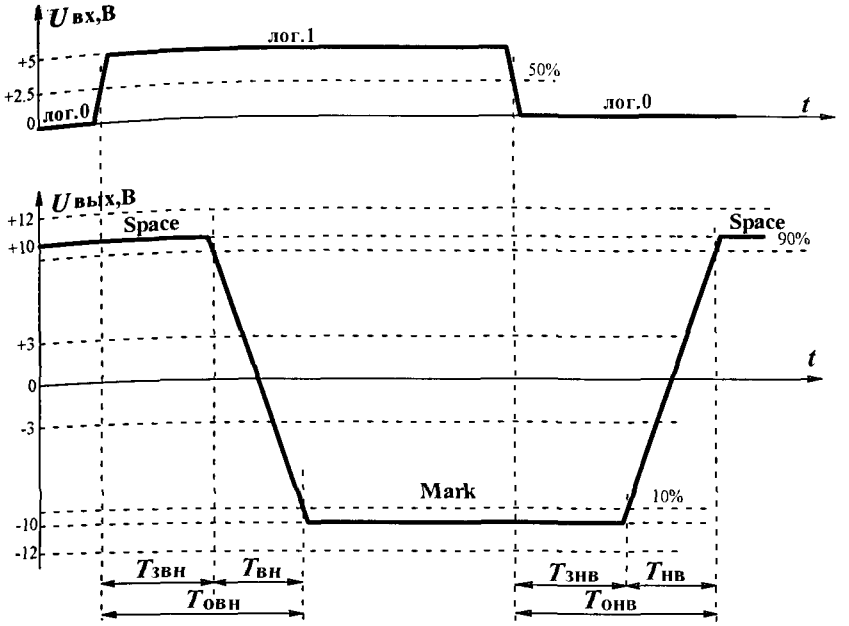


Рис. 3.2. Типичная передаточная характеристика передатчика RS232

двух графиков, общие времена перехода сигнала на выходе передатчика из высокого уровня в низкий уровень ($T_{овн}$) и с низкого уровня на высокий ($T_{онв}$) складываются из соответствующих задержек ($T_{звн}$, $T_{знв}$) и длительностей спада ($T_{вн}$) и фронта ($T_{нв}$): $T_{овн} = T_{звн} + T_{вн}$ и $T_{онв} = T_{знв} + T_{нв}$. Началом задержек $T_{звн}$ и $T_{знв}$ считается переход сигнала на входе через значение напряжения, равное 50% амплитуды входного сигнала (у передатчика). Концом – переход напряжения выходного сигнала через уровень 90% и 10% соответственно. Из графиков также видно, как вычисляются длительности фронта и спада выходного сигнала.

На практике $T_{звн}$ и $T_{знв}$ по сравнению с $T_{вн}$ и $T_{нв}$ пренебрежимо малы (исключения составляют некоторые микросхемы, на которые мы обратим в дальнейшем внимание), и ими можно пренебречь. Что касается длительностей фронта и спада, то у хорошего преобразователя их значения находятся в диапазоне 30–50 В/мкс. Увеличение длительностей фронта и спада может нарушить обмен информацией при большой скорости (115200 бод), а уменьшение – привести к возникновению дополнительных помех, особенно на расстояниях порядка десятков метров и более. *Если же обмен информацией идет по достаточно короткому кабелю (1–2 м), то уменьшение длительностей фронта и спада не приводит к каким-либо негативным результатам.*

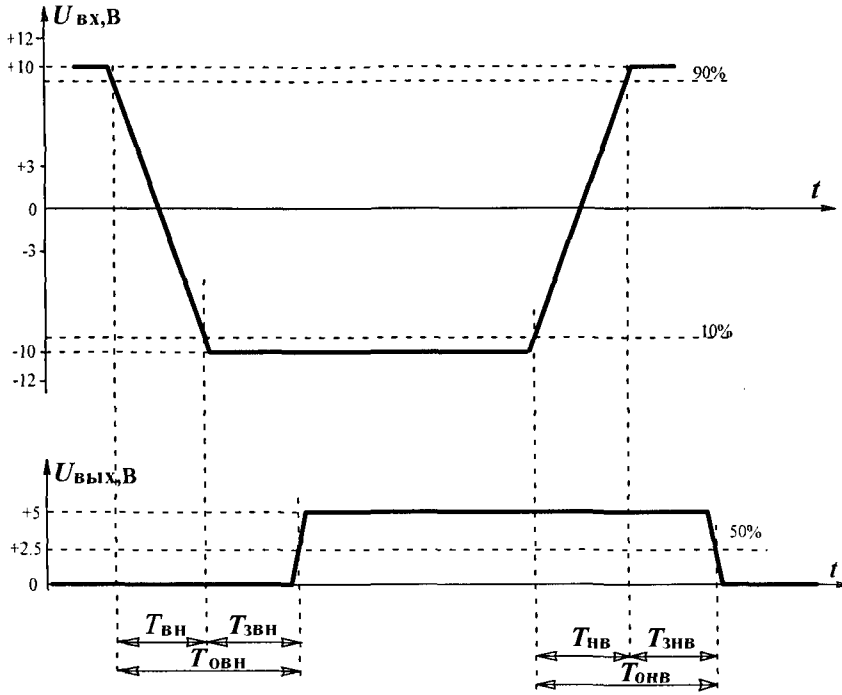


Рис. 3.3. Типичная передаточная характеристика приемника RS232

Отметим, что длительности фронта и спада, так же, как и амплитуда выходного сигнала передатчика, тесно связаны с нагрузкой, подключаемой к его выходу. *При большей нагрузке на передатчик амплитуда выходного сигнала уменьшается, а длительности фронта и спада увеличиваются.* Мы еще вернемся к этому вопросу, когда будем рассматривать оптронные развязки на RS232 (гл. 4).

Из рис. 3.3 видно, как вычисляются аналогичные временные характеристики приемника.

И, наконец, последнее, чем характеризуются преобразователи уровней интерфейса RS232, это потребление энергии (или потребление тока). В более старых микросхемах, которые, кстати, до сих пор применяются (правда, только в настольных компьютерах) потребление тока может достигать 20, 30 мА и более (иногда – до 60 мА), так как они сконструированы на TTL-элементах (биполярных транзисторах). Современные КМОП-микросхемы преобразователей потребляют ток от долей миллиампер (иногда даже от единиц микроампер) до 1–2 мА.

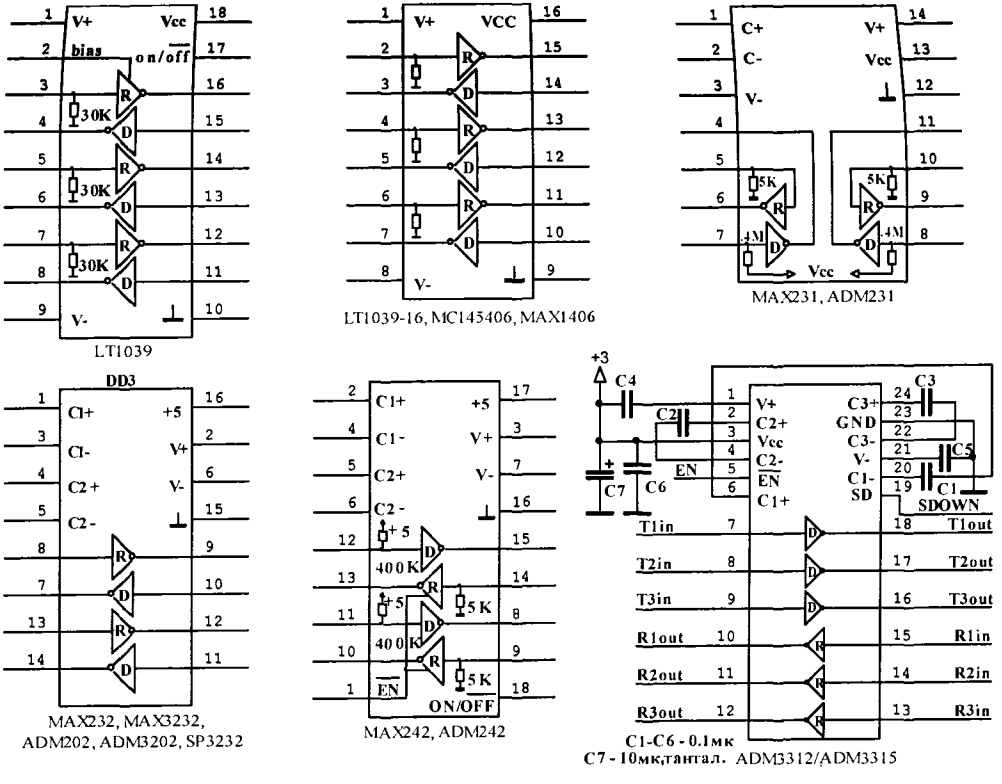


Рис. 3.4. Микросхемы традиционных преобразователей уровней RS232;
R (Receiver) – приемник, D (Driver) – передатчик

Существуют десятки различных микросхем преобразователей уровней интерфейса RS232, которые выпускаются множеством фирм. Нас будут интересовать такие преобразователи, которые:

а) имеют малое потребление тока;
б) поддерживают (по возможности) скорость обмена 115200 бод;
в) содержат один передатчик и два (лучше три) приемника (рис. 3.4);
г) могут, по возможности, работать от напряжения 3 В (а лучше и от 3, и от 5 В);
д) имеют приемлемую цену (не более 1–2 долл.);
е) имеют не очень большое число выводов;
ж) работают от одного источника питания (3 или 5 В).

На рис. 3.4 приведены микросхемы преобразователей, которые сочетают в себе многие из перечисленных в пп. а)–ж) свойств, но, к сожалению, не все.

Так, микросхемы LT1039, LT1039-16, MC145406, MAX1406 имеют по три передатчика и приемника (+), достаточно малое потребление тока (+), однако требуют трех источников питания (+12, –12 и +5 В) (–); при скорости обмена 115200 бод устойчиво работает только MAX1406 (максимально устойчивая скорость работы LT1039 и LT1039-16 – 57600 бод, MC145406 – 38400 бод) (–). MAX231 и ADM231 великолепно работают на скорости 115200 бод (вообще они поддерживают скорость до 230 кбод) (+), имеют размах напряжения выходного сигнала до ±12 В (+), достаточно малое число выводов (+), однако требуют двух источников питания: +12 и +5 В (–), содержат по два передатчика и приемника (–). MAX3232, ADM3202 могут работать от единственного источника питания (+3 или +5 В) (+), поддерживают скорость обмена 115200 бод (+), имеют достаточно малое число выводов (+), приемлемую цену (+), однако содержат по два передатчика и приемника (–) и не могут быть переведены в режим микропотребления (shut down) (–). Все же, на наш взгляд, *эти микросхемы преобразователей (MAX3232,ADM3202,SP3232) в наибольшей степени заслуживают внимания*. MAX242 и ADM242 поддерживают скорость 115200 бод (+), могут быть переведены в режим пониженного потребления (+), требуют одного источника питания (+), имеют приемлемую цену(+), малое число выводов (+), однако не могут работать от 3 В (–); микросхема ADM242, кроме того, может иногда "защелкиваться" при условии, что она находится в режиме микропотребления и включение ее питания происходит после включения питания компьютера (–) (MAX242 в этом отношении работает безукоризненно). И, наконец, микросхемы ADM3312, ADM3315 – хорошие современные микросхемы, имеющие по три передатчика и приемника (+), поддерживают скорость 115200 бод (+), требуют одного

источника питания (+), имеют размах напряжения сигнала передатчика RS232, в три раза больший напряжения источника питания, так как содержат не удвоитель, а утроитель напряжения (!) (+), однако не работают при напряжении +5 В (для них максимальное напряжение источника питания 3,6 В) (–), имеют относительно большое число выводов (24) (–) и, кроме того, пока достаточно дóроги (стоимость их чуть более 4 долл.) (–).

Необходимо отметить, что вышеперечисленные недостатки преобразователей уровней RS232 ни в коем случае не являются неустранимым препятствием к их применению. Эти недостатки с успехом могут быть скомпенсированы дополнительными *схемными решениями* (это утверждение не распространяется на скорость обмена: если микросхема не поддерживает скорость обмена 115200 бод, понятно, что никакими средствами увеличить эту скорость нельзя). Эти схемные решения – предмет рассмотрения последующих глав. Сейчас же рассмотрим некоторые нетрадиционные микросхемы преобразователей уровней интерфейса RS232, которые, с одной стороны, вообще говоря, таковыми не считаются, а с другой – как раз и послужат базой для вышеупомянутых *схемных решений*.

3.3. Нетрадиционные преобразователи RS232

Сразу оговоримся, что приводимые далее микросхемы, используемые автором в качестве преобразователей уровней интерфейса RS232 (как приемников, так и передатчиков), не узаконены никакими гостами, ISO и т.п. Рекомендации по использованию этих микросхем в качестве преобразователей уровней RS232, приводимые далее, являются результатом многочисленных экспериментов автора с интерфейсом RS232, и не более того. С другой стороны, как уже было сказано (§ 3.2), традиционные преобразователи, хоть и узаконены, не всегда удовлетворяют поставленным целям.

Итак, рассмотрим некоторые известные микросхемы, которые с успехом могут быть использованы в качестве преобразователей уровней RS232. В качестве приемника может быть использован простейший преобразователь уровней КМОП-ТТЛ (например, один элемент инвертора 561ЛН2) с дополнительным резистором. *Оптроны* могут работать как приемниками, так и передатчиками; при необходимости они должны быть высокоскоростными. КМОП-мультиплексоры и переключатели также могут работать в двух направлениях: RS232→TTL и обратно.

На рис. 3.5 приведены примеры преобразователей уровней интерфейса RS232 на базе КМОП-переключателей (*а–г*), преобразователей КМОП-ТТЛ (*д*) и оптронов (*е*).

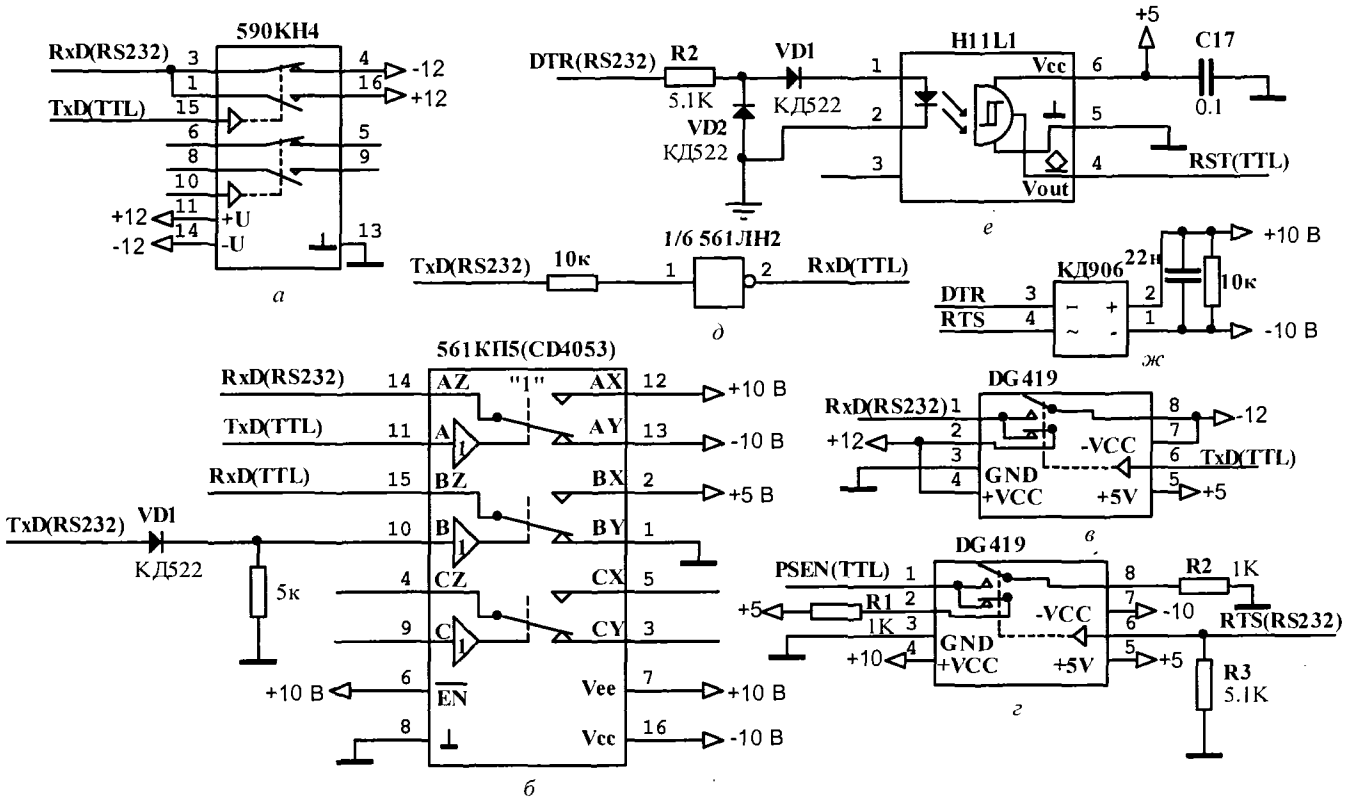


Рис. 3.5. Примеры преобразователей уровней интерфейса RS232 на базе КМОП-микросхем и оптронов

Отметим также, что нагрузочной способности интерфейса RS232 вполне достаточно, чтобы получить необходимые напряжения (±10–±12 В) для питания приведенных на рис. 3.5 преобразователей. Для этого можно использовать простейший выпрямитель (рис. 3.5, *ж*, DTR = +10 В, RTS =–10 В).

Нельзя не сказать еще об одном свойстве приведенных на рис. 3.5 (и подобных) преобразователей: они чрезвычайно дешевы (некоторые из них стоят 3–10 руб.).

И последнее, о чем не надо забывать, – применение нетрадиционных преобразователей уровней RS232 отнюдь не исключает их совместное использование с традиционными. В этом читатель сможет убедиться далее, когда будут приведены конкретные надежно работающие схемы сопряжения микроконтроллеров с компьютером по интерфейсу RS232.

4. Примеры сопряжения микроконтроллеров с компьютером по интерфейсу RS232

Как правило, при сопряжении микроконтроллера с компьютером по RS232 между ними существует гальваническая связь, т.е. микроконтроллер и компьютер имеют общую сигнальную "землю" (SG). Однако в некоторых случаях – либо когда используются встроенные в микроконтроллер АЦП, особенно имеющие достаточно высокое разрешение и в связи с этим свою "аналоговую землю", либо когда микроконтроллер находится под значительным напряжением по отношению к компьютеру – может потребоваться гальванически развязанный интерфейс RS232.

4.1. Примеры RS232, имеющих гальваническую связь компьютера с микроконтроллером

Приведенные на рис. 4.1–4.4 схемы сопряжений микроконтроллеров с компьютером в особых комментариях не нуждаются за исключением того факта, что все схемы прекрасно работают (в том числе) на скорости 115200 бод.

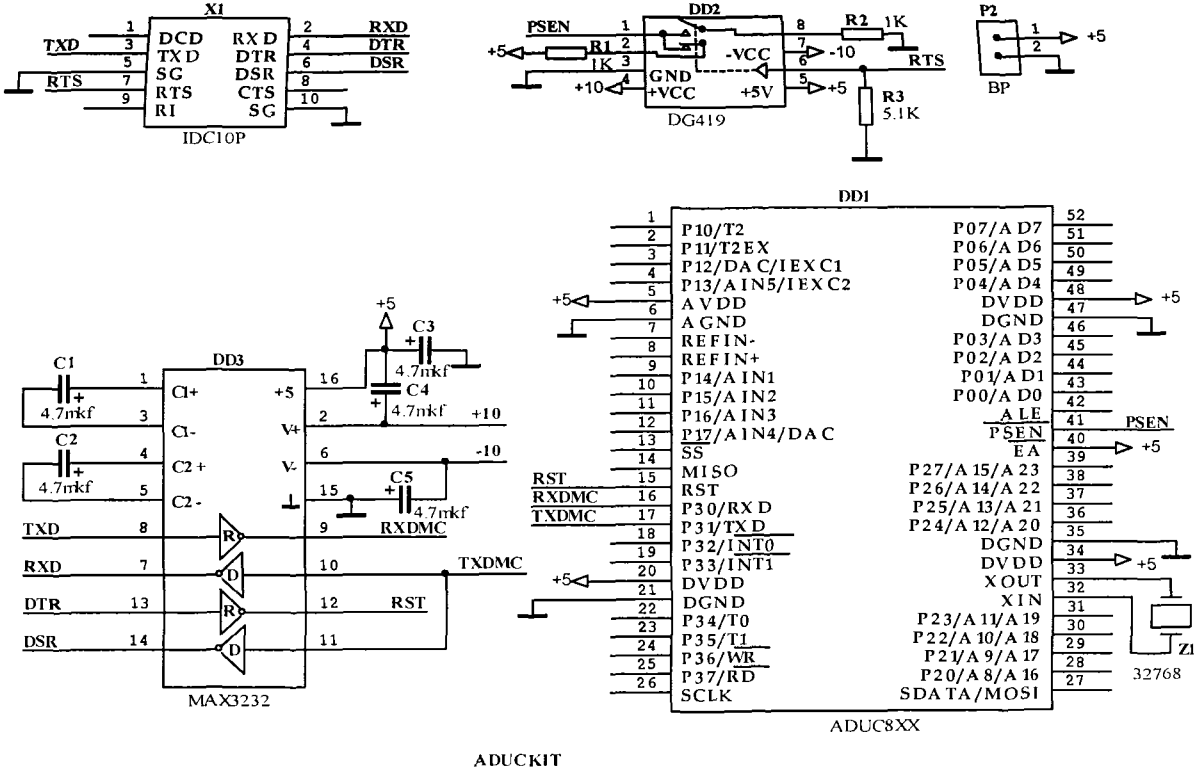


Рис. 4.1. Пример сопряжения микроконтроллера (микроконвертора) ADUC8XX с компьютером для последовательного программирования и запуска загруженной программы

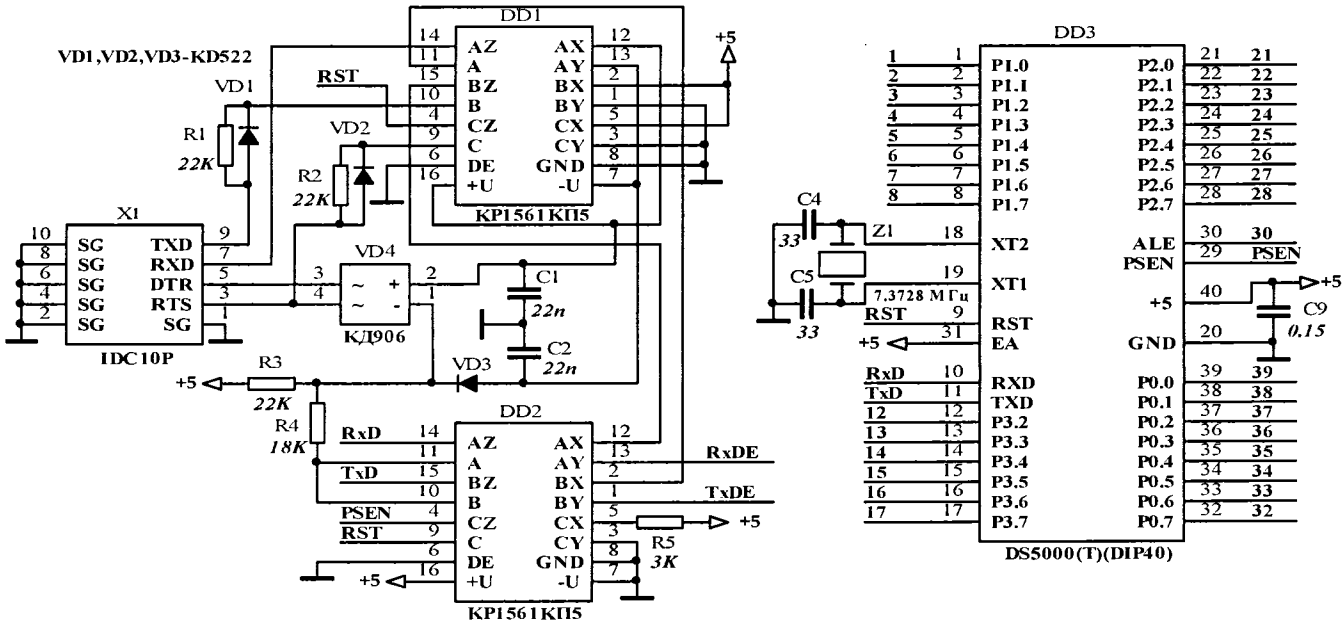


Рис. 4.2. Пример сопряжения soft-микроконтроллера DS5000(T) с компьютером для последовательного программирования и запуска загруженной программы

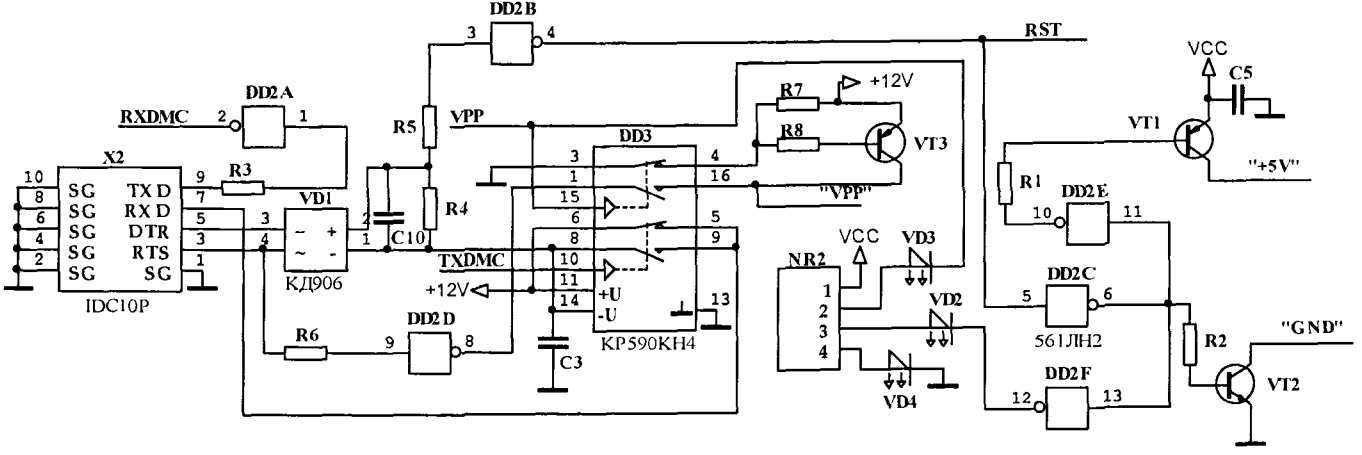


Рис. 4.3. Фрагмент схемы программатора (на базе микроконтроллера AT89C52) 51-совместимых микроконтроллеров. Микроконтроллер AT89C52 сопряжен с компьютером по RS232. Сигналы TxDMC, RxDMC и RST подходят соответственно к выводам TxD, RxD и RST микроконтроллера AT89C52

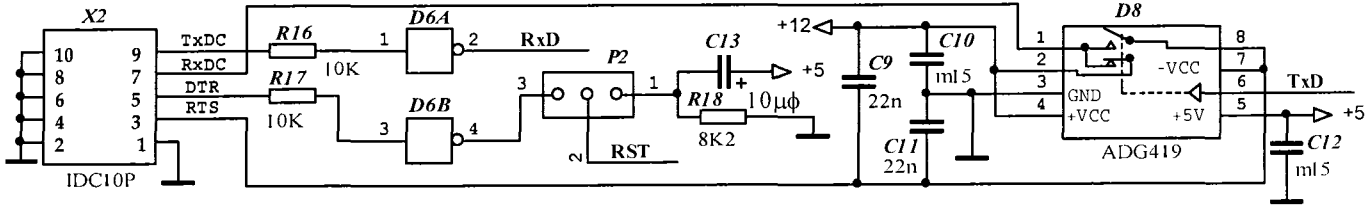


Рис. 4.4. Фрагмент схемы прибора для поверки счетчиков объема газа на базе микроконтроллера AT89C4051.
При соединении контактов *2* и *3* перемычки *P2* "джампером" микроконтроллер становится сопряженным с компьютером по RS232.
Сигналы RxD, TxD и RST подходят соответственно к выводам RxD, TxD и RST микроконтроллера.
D6 – микросхема 561ЛН2

4.2. Примеры гальванически развязанных RS232

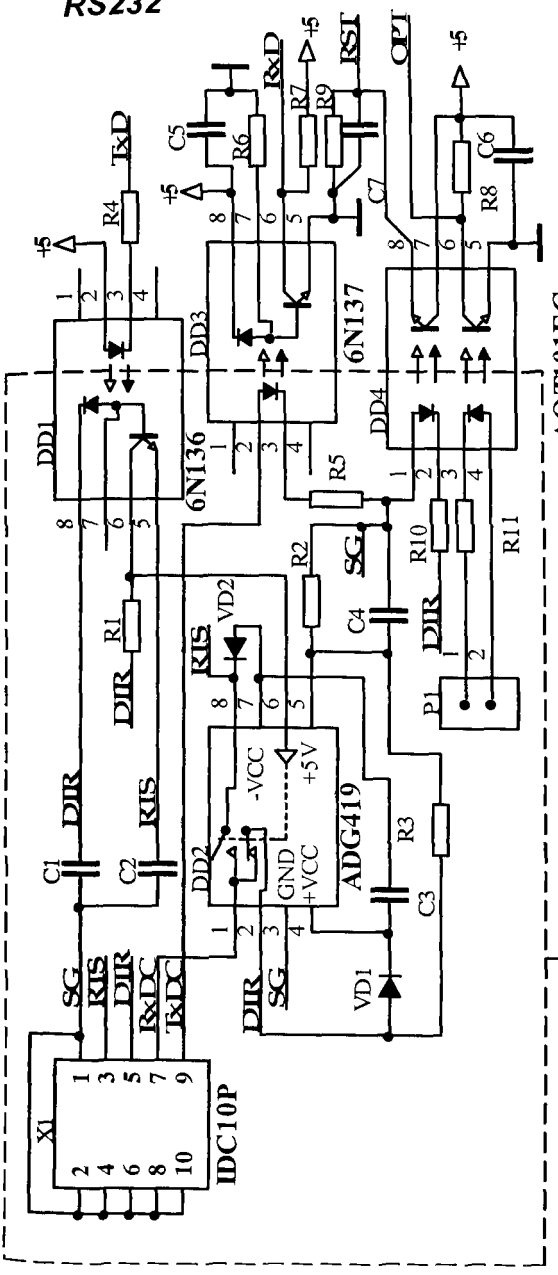
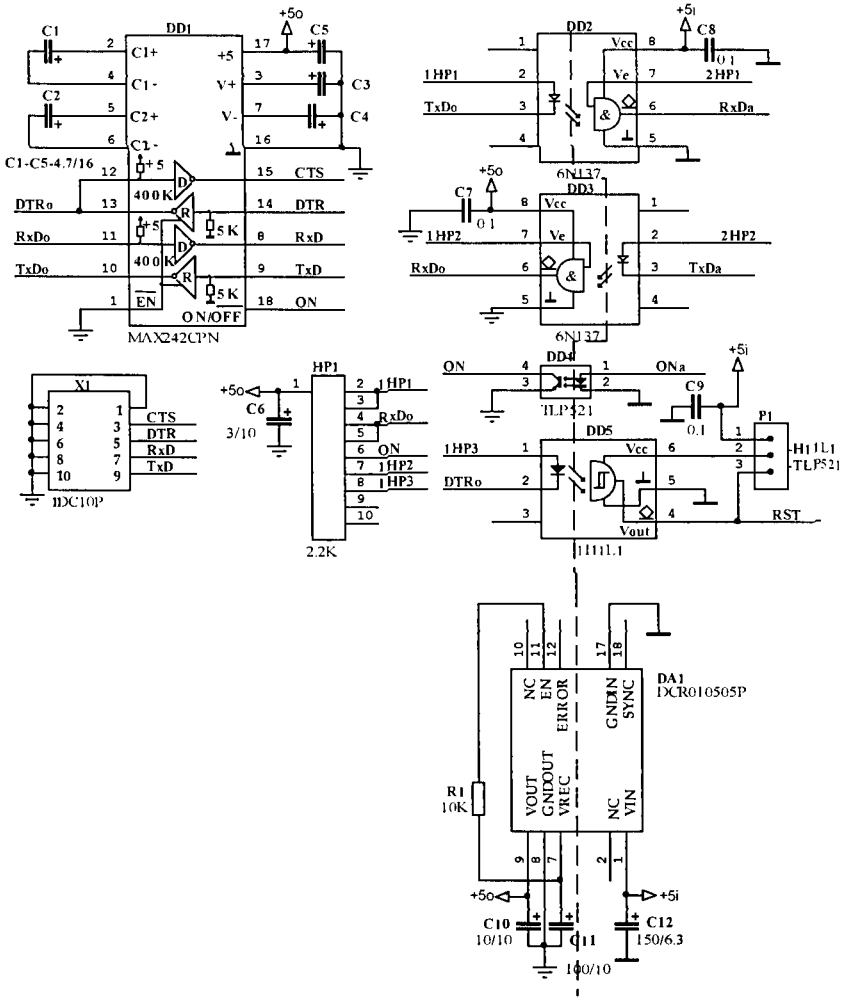


Рис. 4.5. Пример сопряжения компьютера с микроконтроллером по гальванически развязанному интерфейсу RS232. Питающие интерфейс напряжения снимаются с линий DTR (+10 В) и RTS (–10 В). Сигналы RxDC и TxDC подходят соответственно к выводам RxD и TxD компьютера; сигналы TxD и RxD – соответственно к выводам TxD и RxD микроконтроллера (не показано).

Сигнал RST подходит к выводу RESET микроконтроллера. Максимальная скорость обмена 57600 бод


C1
C2
C1-C5-4.7/16
DD1
MAX242CPN
C1+
C1-
C2+
C2-
V+
V-
+5o
+5
C3
C4
C5
DTRo
RxDo
TxDo
400K
5K
ON/OFF
CTS
DTR
RxD
TxD
ON
EN
X1
IIDC10P
HP1
2.2K
C6
3/10
DD2
DD3
6N137
1HP1
2HP1
1HP2
2HP2
C7
C8
0.1
RxDa
TxDa
+5i
DD4
TLP521
ONa
C9
P1
H1L1
DD5
1HP3
Vcc
Ve
Vout
RST
1H1L1
DA1
DCR010505P
NC
EN
ERROR
GNDIN
SYNC
VOUT
GNDOUT
VREC
VIN
R1
10K
C10
10/10
C11
C12
150/6.3

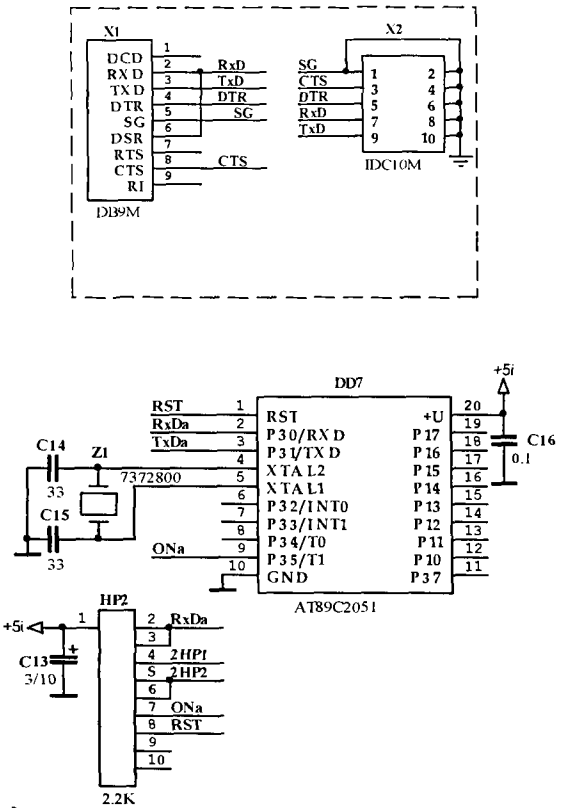


Рис. 4.6. Пример сопряжения компьютера с микроконтроллером по гальванически развязанному интерфейсу RS232. Максимальная скорость обмена 115200 бод. Питающее напряжение +5 В (стабилизированное) подается на сторону компьютера с помощью изолирующего преобразователя напряжения DCR010505P

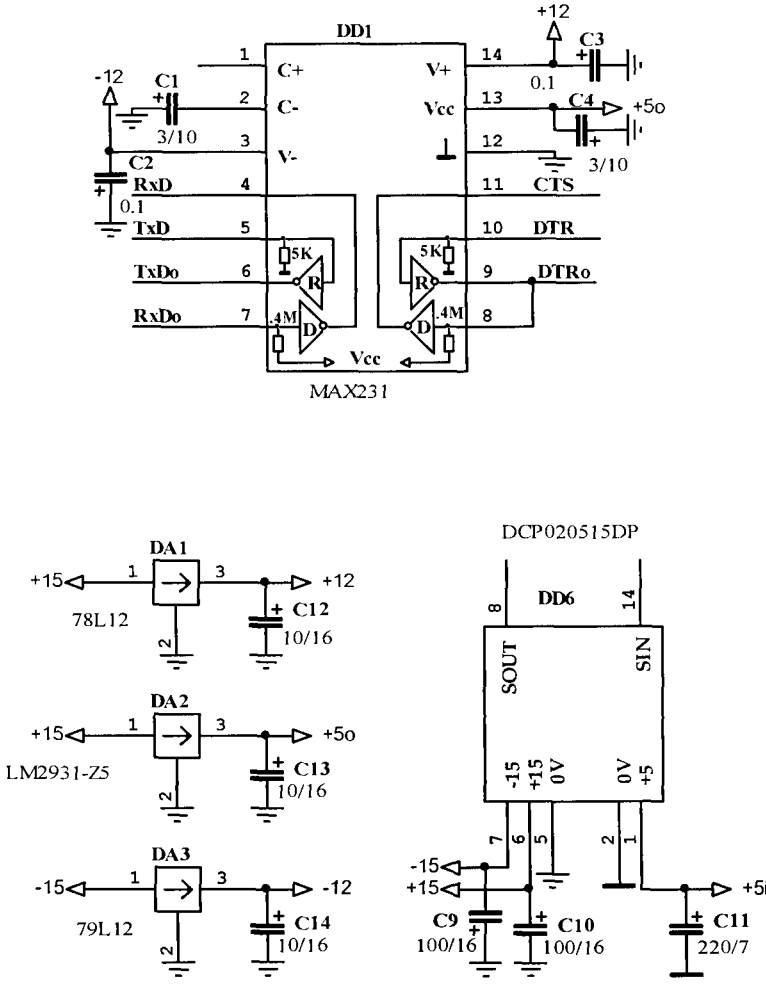


Рис. 4.7. Пример подачи питающих напряжений на сторону компьютера с помощью двуполярного нестабилизированного изолирующего преобразователя DCP020515DP и трех стабилизаторов (+12, +5 и –12 В). Сверху показан вариант подключения преобразователя уровней интерфейса RS232 – микросхема MAX231 (отрицательное напряжение –12 В подключено к выводу V)

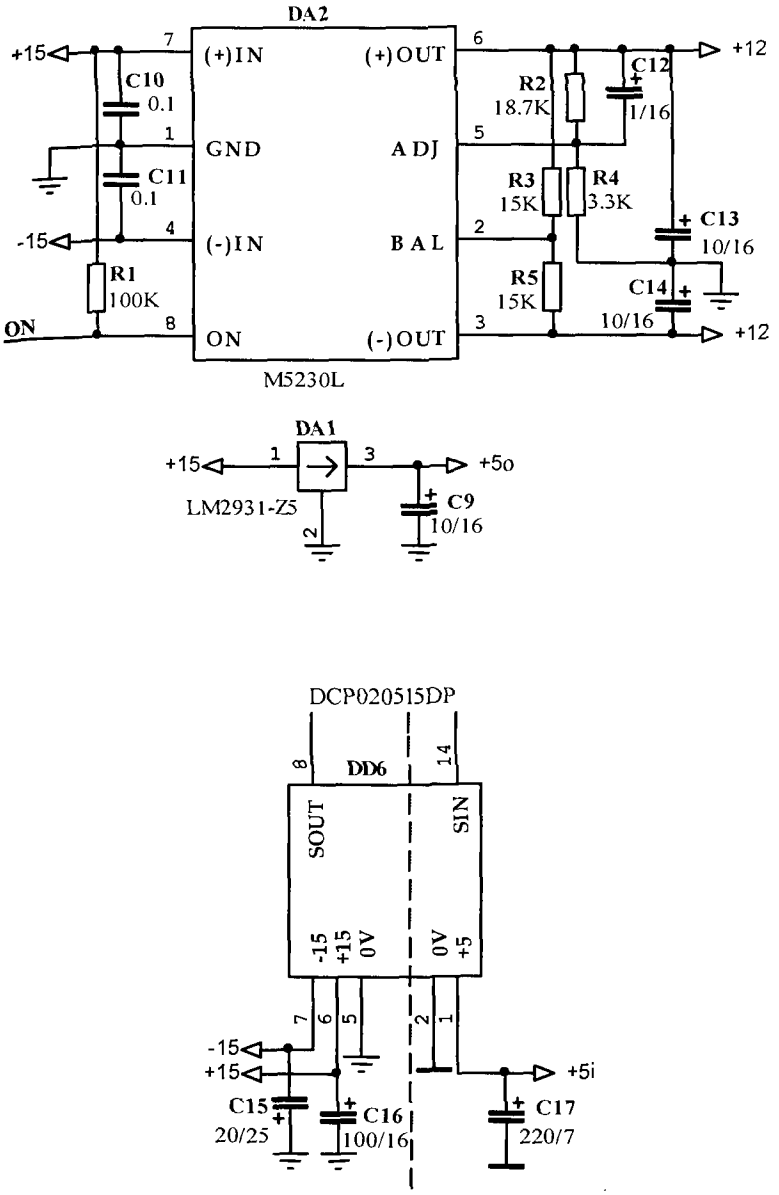
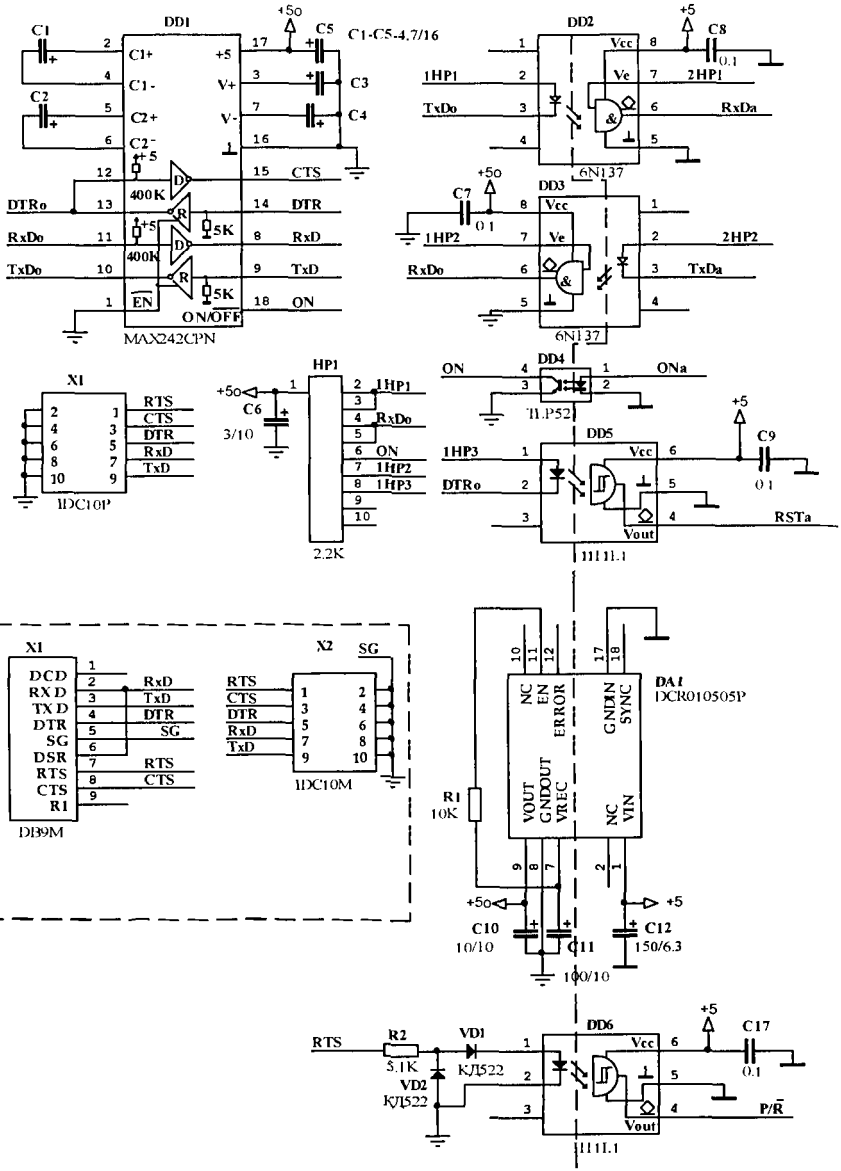


Рис. 4.8. Схема стабилизации напряжений +12 и –12 В с помощью двуполярного стабилизатора М5230


DD1
MAX242CPN
C1-C5-4.7/16
DD2
6N137
DD3
6N137
DD4
TLP521
DD5
1111L1
X1
1DC10P
HP1
2.2K
X1
DB9M
X2
1DC10M
DA1
DCR010505P
DD6
1111L1
ONOFF
400K
5K
CTS
DTR
RxD
TxD
ON
RTS
SG
RxDa
TxDa
2HP1
2HP2
1HP1
1HP2
1HP3
RSTa
P/R
VD1
VD2
KJ522
R1
10K
R2
5.1K
C10
10/10
C11
100/10
C12
150/6.3
C17
0.1
C9
0.1
C8
0.1
C7
0.1
C6
3/10

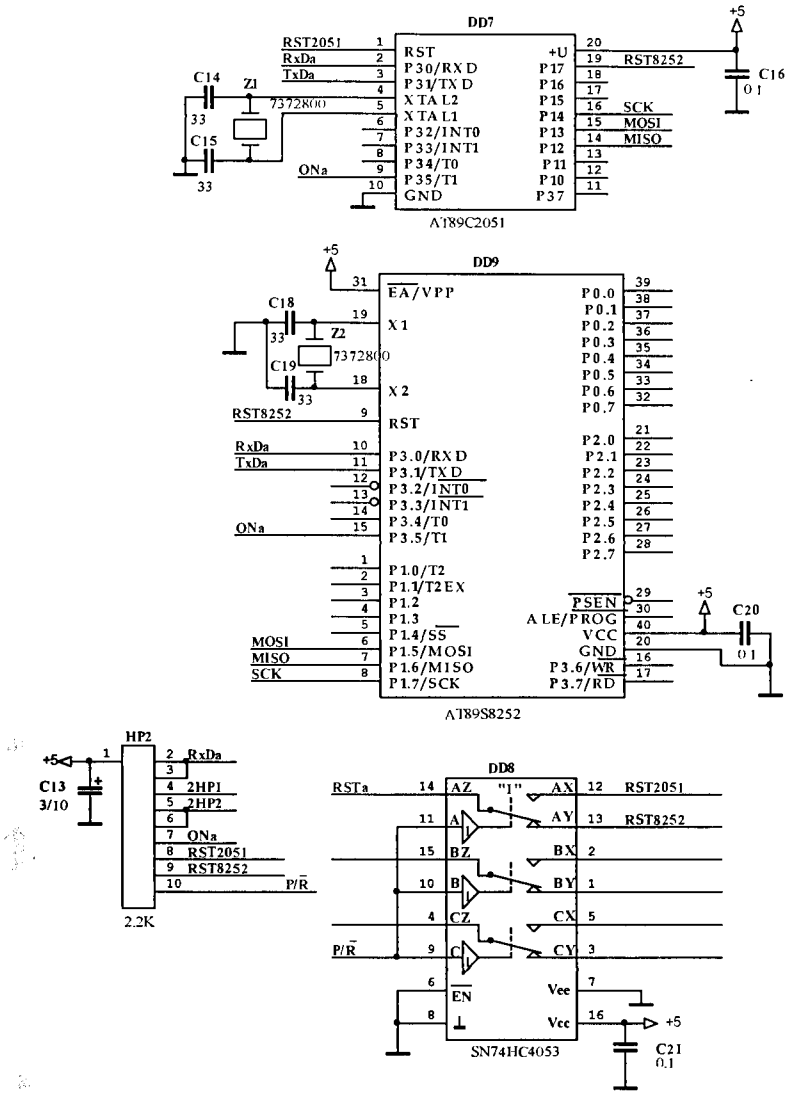


Рис. 4.9. Схема гальванически развязанного интерфейса RS232, используемого для последовательного программирования микроконтроллера AT89S8252 и запуска загруженной в него программы. Микроконтроллер AT89C2051 сопряжен с компьютером по гальванически развязанному интерфейсу RS232, по которому идет обмен информацией (со скоростью 115200 бод). Программирование микроконтроллера AT89S8252 осуществляется микроконтроллером AT89C2051 по интерфейсу SPI (сигналы MOSI, MISO и SCK). Сигналом P/R (программирование – Prog или запуск – Run) управляет линия RTS, сигналом RSTa – линия DTR

5. Программирование интерфейса RS232 в микроконтроллере

5.1. Предварительные замечания

В отличие от интерфейса RS232 компьютера, в интерфейсе RS232 микроконтроллера:

- нельзя изменять формат посылаемых данных – это всегда 8 бит;
- бит паритета, как правило, не используется, хотя может быть задействован;
- могут быть один или два стоп-бита, при этом второй стоп-бит "искусственный" (в табл. 5.2 режимы 2 и 3);
- введен дополнительный синхронный режим работы RS232 – режим 0 (табл. 5.2);
- скорость обмена определяется большим, чем в компьютере, числом факторов.

После такого "предварительного ознакомления" с интерфейсом RS232 в микроконтроллере может сложиться мнение, что программирование интерфейса "проще", чем программирование интерфейса RS232 компьютера. Отчасти это так, хотя в микроконтроллере больше возможностей для задания скорости обмена.

Скорость обмена может быть установлена без применения таймеров. В этом случае она может быть равна тактовой частоте микроконтроллера, деленной на 32 или 64. Если, например, необходимо установить скорость 115200 бод, то частоту кварцевого резонатора целесообразно выбирать равной 115200·32=3686400 Гц или 115200·64 = 7372800 Гц. Этот режим установки скорости может быть полезен, если используется достаточно простой 51-совместимый микроконтроллер, у которого таймеры применяются для других целей (более подробно – см. далее).

Во всех остальных случаях скорость обмена устанавливается с помощью таймеров (Т). Их в микроконтроллере может быть несколько. Частота, которая подается на таймер, непосредственно зависит от циклической частоты работы микроконтроллера, которая, в свою очередь, определяется частотой кварцевого резонатора. В микроконтроллерах, выпускавшихся ранее, циклическая частота в 12 раз ниже тактовой. Другими словами, команда микроконтроллера, выполняющаяся за один цикл, занимает 12 тактов. В более современных микроконтроллерах, таких, например, как DS80520,

DS80530 производства фирмы Dallas Semiconductor или MSC1210YX производства фирмы Texas Instruments, циклическая частота может быть повышена (программным способом) в три раза. В этом случае одна команда микроконтроллера, выполняющаяся за один цикл, будет занимать уже четыре такта. В микроконтроллерах ADUC8XX производства фирмы Analog Devices для задания тактовой частоты процессора используется часовой кварцевый резонатор с частотой 32768 Гц совместно с умножителем частоты (PLL), который умножает частоту 32768 Гц (максимум) на 384. В результате получается тактовая частота 12,582912 МГц. Коэффициент умножения может изменяться программно в пределах от 3 до 384 и составлять 3, 6, 12, 24, 48, 96, 192 и 384. В связи с этим частоты, на которых работает процессор, соответственно равны 98304, 196608, 393216, 786432, 1572864, 3145728 и 12582912 Гц. Циклическая же частота процессора в микроконтроллерах ADUC8XX в 12 раз ниже тактовой. Кроме того, в микроконтроллерах ADUC83X в качестве генератора скорости обмена по RS232 используется специальный таймер (T3), с помощью которого можно устанавливать скорость обмена не только 115200, но и 230400 бод (!), хотя, конечно, такую скорость обмена по RS232 уже не поддерживает компьютер.

В новейших микроконтроллерах ADUC84X производства фирмы Analog Devices, в микроконтроллерах C8051FXXX производства фирмы CYGNAL, а также в некоторых микроконтроллерах серии VERSA производства фирмы Goal Semiconductor Inc., имеющих быстрое 51-совместимое ядро (RISC-процессор), циклическая частота процессора еще более приближена к тактовой. В этих микроконтроллерах одна команда процессора выполняется уже за один-два такта. К тому же эти микроконтроллеры оборудованы специальными делителями тактовой частоты, предназначенными для генерации скорости обмена по RS232. Таким образом, в микроконтроллерах имеется множество возможностей для задания скорости обмена по RS232.

Далее будут приведены сведения по программированию интерфейса RS232, которым оборудованы самые простые микроконтроллеры, а также некоторые современные микроконтроллеры.

Необходимо еще добавить следующее. Поскольку для написания программ в микроконтроллерах в настоящее время большей частью используются два языка программирования: ассемблер и Си, программы будут приведены именно на этих языках.

С каждым (особенно современным) микроконтроллером поставляется демонстрационное программное обеспечение, в котором, в частности, приведены файлы, включаемые (оператором $include) как в ассемблерные программы, так и в программы на Си, с указанием названий регистров и их адресов в памяти микрокон-

троллера. Для включения в ассемблерные программы эти файлы имеют расширения *.INC (например, REG1210.INC) или названия MOD* (например, MOD834); для включения в программы на языке Си – расширения *.H (например, ADUC834.H). Во всех приводимых далее программах (особенно для современных микроконтроллеров) будут использоваться такие файлы. Файлы приведены в описаниях на микроконтроллеры. В программах, где такие файлы не используются (программах, обычно написанных для более простых или "старых" микроконтроллеров и, как правило, на ассемблере), адреса регистров приняты по умолчанию и соответствующим образом включены в язык. Поэтому отсутствие этих файлов в приводимых программах не является ошибкой.

И последнее, что необходимо добавить. Существует несколько режимов работы интерфейса RS232 в микроконтроллере. Номера режимов (например, режим 0, режим 1 и т.п.) будут приводиться не в порядке их возрастания, а в том, в котором, на взгляд автора, целесообразнее вести изложение. Например, режим работы RS232 без использования таймеров – это режим 2, и именно он будет рассмотрен в первую очередь.

После этих предварительных замечаний приведем примеры инициализации интерфейса RS232 и примеры простейших команд ввода/вывода через этот интерфейс.

5.2. Инициализация RS232 и команды ввода/вывода

5.2.1. Инициализация RS232 без использования таймеров

Работа интерфейса RS232 в микроконтроллере в этом случае определяется содержимым только двух регистров:

а) регистра управления последовательным интерфейсом (Serial CONtrol – SCON);

б) 7-м разрядом регистра микропотребления (Power CONtrol – PCON) – битом SMOD.

В табл. 5.1 показано содержимое регистра управления последовательным интерфейсом. Биты SM0, SM1 определяют режимы работы интерфейса в соответствии с табл. 5.2, из которой, в частности, следует, что в режиме 2 (SM0 = 1, SM1 = 0) интерфейс работает с постоянной скоростью $F_{кв}/64$ или $F_{кв}/32$, которая не зависит от таймеров.

Бит SM2 разрешает (SM2 = 1) или запрещает (SM2 = 0) многопроцессорную работу интерфейса (в нашем случае рассматривается только однопроцессорный режим, поэтому SM2 всегда должен быть равен 0).

Таблица 5.1

Содержимое регистра SCON

Номер бита	7	6	5	4	3	2	1	0
Название бита	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Таблица 5.2

Режимы работы интерфейса RS232

SM0	SM1	Номер режима	Режим работы интерфейса	Скорость обмена
0	0	0	Сдвиговый регистр	$F_{кв}/12$
0	1	1	8-разрядный приемник/передатчик	Определяется T1
1	0	2	9-разрядный приемник/передатчик	$F_{кв}/64$ или $F_{кв}/32$
1	1	3	9-разрядный приемник/передатчик	Определяется T1

Бит REN (Receiver ENable) разрешает (REN = 1) или запрещает (REN = 0) прием данных интерфейсом; во многих случаях при инициализации интерфейса его целесообразно установить в 1.

Бит TB8 – 9-й бит передаваемых данных в режимах 2 и 3; при его равенстве 1 получаем дополнительный стоп-бит, поэтому его целесообразно установить в 1.

Бит RB8 – 9-й бит принятых данных в режимах 2 и 3; этот бит в рассматриваемом случае не используется, поэтому его целесообразно установить в 0.

Бит TI (Tranceiver Interrupt) – флаг прерывания передатчика, который устанавливается в 1 в начале стоп-бита (кроме режима 0); при инициализации интерфейса целесообразно установить его в 1.

Бит RI (Receiver Interrupt) – флаг прерывания приемника; этот бит устанавливается в 1 в середине интервала времени принятого стоп-бита (кроме режима 0); при инициализации интерфейса целесообразно установить его в 0.

Таким образом, получаем значение регистра: SCON=10011010b или SCON = 9ah.

Как уже говорилось, старший бит регистра микропотребления (бит SMOD регистра PCON) определяет удвоение скорости обмена. При SMOD = 1 скорость обмена в этом примере равна $F_{кв}/32$, при SMOD = 0 – $F_{кв}/64$. Регистр PCON не имеет побитную адресацию; кроме того, остальные его разряды к интерфейсу не имеют отношения, поэтому в этом примере целесообразно устанавливать PCON = 80h (SMOD = 1) либо PCON = 0 (SMOD = 0).

Пример 5.1. Проинициализировать интерфейс для получения скорости 115200 бод без использования таймеров при частотах кварцевого резонатора, равных 7372800 и 3686400 Гц.

Фрагменты программ на ассемблере и Си приведены в таблице:

ассемблер, $F_{кв}$ = 7372800 Гц	Си, $F_{кв}$ = 3686400 Гц
mov SCON,#9ah mov PCON,#00h	SCON=0x9a; PCON=0x80;

Как видно из приведенного примера, инициализация интерфейса без применения таймеров достаточно проста. Недостатком такого режима работы интерфейса является использование строго ограниченных значений частот кварцевых резонаторов, что не всегда возможно.

5.2.2. Инициализация RS232 с использованием таймеров

Во многих случаях от микроконтроллера требуется переменная скорость обмена по RS232. Как правило, диапазон значений скоростей обмена невелик, и часто ограничивается всего двумя значениями: 9600 и 115200 бод.

В качестве генератора скорости обмена в простейшем случае может выступать таймер T1, который присутствует в подавляющем числе микроконтроллеров. Некоторые микроконтроллеры оборудованы дополнительным таймером T2, который также может быть использован в качестве генератора скорости обмена. Диапазон скоростей обмена, генерируемых таймером T2, намного шире, чем генерируемый таймером T1. И, наконец, очень ограниченное число микроконтроллеров (это некоторые современные микроконтроллеры ADUC8XX производства фирмы Analog Devices) имеет таймер T3, диапазон генерируемых скоростей обмена которого еще больше, чем таймера T2.

Инициализация RS232 с использованием таймера T1. Из табл. 5.2 следует, что в режиме 1 (SM0 = 0, SM1 = 1) и 3 (SM0 = = SM1 = 1) скорость обмена определяется таймером T1. Для генерации скорости обмена по RS232 таймер T1, как правило, используется в режиме автозагрузки, т.е. в режиме 2 (см. описание работы таймера T1 в любом микроконтроллере).

Скорость обмена по RS232 в режимах 1 и 3 может быть определена по следующей формуле:

$$V_{\text{обм.1,3}} = (2^{\text{SMOD}} \cdot F_{\text{кв}})/(32 \cdot 12 \cdot [256 - \text{TH1}]),$$

где $F_{\text{кв}}$ – частота кварцевого резонатора, TH1 – содержимое старшего байта регистра счета таймера T1.

Если, например, требуется установить скорость обмена 115200 бод с помощью таймера T1, то минимальная частота кварцевого резонатора должна быть равна 22118400 Гц (при SMOD = 1 и TH1 = 255):

$$2^1 \cdot 22118400/(3 \cdot 12 \cdot [256 - 255]) = 115200 \text{ бод}.$$

Во многих случаях для сокращения энергопотребления частоту кварцевого резонатора стремятся уменьшить (при этом, например, требуются две скорости обмена: 9600 и 115200 бод). В этом случае (например, при $F_{\text{кв}} = 3686400$ Гц) для генерации скорости

обмена 115200 бод можно использовать режим 2 (см. предыдущий пример 5.1), а для генерации скорости обмена 9600 бод – таймер 1 (SMOD = 1, TH1 = 254):

$$2^1 \cdot 3686400/(32 \cdot 12 \cdot [256 - 254]) = 9600 \text{ бод}.$$

Пример 5.2. Проинициализировать интерфейс RS232 для получения скоростей обмена 115200 и 9600 бод с использованием таймера 1.

ассемблер, $F_{кв}$ = 22118400 Гц, $V_{обм}$ = 115200 бод	Си, $F_{кв}$ = 3686400 Гц, $V_{обм}$ = 9600 бод

mov SCON, #52h ;SM1=REN=TI=1, режим 1.
mov PCON, #80h ;SMOD=1 (удвоение скорости).
mov TMOD,#20h ; режим автозагрузки T1.
mov TH1,#255 ; TH1=255_{10}.
setb TR1 ;TCON.6=1 – запуск T1.
	SCON=0x52 // режим 1.
	PCON=0 // SMOD=0, скорость не удваивается.
	TMOD=0x20 // режим автозагрузки T1.
	TH1=255 // TH1=255_{10}.
	TR1=1 // Запуск T1.

Из вышеизложенного можно сделать следующие выводы. Таймер T1 можно использовать для генерации скорости обмена по RS232 в достаточно узкой области значений частот кварцевых резонаторов (точнее – только при строго определенных значениях этих частот). Если частота кварцевого резонатора достаточно низкая (задано), а скоростей обмена должны быть несколько (или хотя бы две), одна из которых – высокая, например, 115200 бод, то инициализацию интерфейса целесообразно проводить с использованием T1 и без него (т.е. комбинировать). В связи с этим при установке скорости обмена с помощью T1 ощущается некий "дискомфорт".

Инициализация RS232 с использованием таймера T2. В состав многих микроконтроллеров последних лет выпуска и всех без исключения современных микроконтроллеров (имеются в виду 51-совместимые) входит таймер T2 (помимо T0 и T1), который предоставляет пользователю более широкие (хотя и не безграничные) возможности для генерации скоростей обмена по RS232. Эти микроконтроллеры в подавляющем большинстве случаев де-факто стали уже называться 52-совместимые или имеющие 52-совместимое ядро (например, AT89C52). Есть, конечно, и исключения, например микроконтроллеры 87C51FA-FC фирм Intel и Philips, в состав которых входит T2.

Скорость обмена по RS232 (также в режимах 1 и 3) при использовании таймера T2 определяется следующей формулой:

$$V_{обм.1,3} = F_{кв}/(32 \cdot [65536 - (RCAP2H, RCAP2L)]),$$

где RCAP2L и RCAP2H – соответственно младший и старший байты 16-разрядного регистра захвата/сравнения RCAP2 таймера T2.

Необходимо добавить, что для генерации скорости обмена с помощью таймера T2 в соответствии с вышеприведенной формулой в его управляющем регистре T2CON требуется установить биты RCLK (бит 5), TCLK (бит 4), определяющие соответственно работу приемника и передатчика; кроме того, нужно запустить таймер T2 (TP2 – бит 2). Таким образом, при T2CON = 00110100b = = 34h скорость обмена по RS232 будет определяться T2.

И еще небольшое добавление: бит SMOD не влияет на скорость обмена при использовании T2.

Пример 5.3. Проинициализировать интерфейс RS232 для получения скоростей обмена 115200 и 9600 бод с использованием таймера T2.

ассемблер, $F_{кв}$ = 11059200 Гц, $V_{обм}$ = 115200 бод		Си, $F_{кв}$ = 11059200 Гц, $V_{обм}$ = 9600 бод	
mov SCON,#52h	; Режим 1.	SCON=0x52	// Режим 1.
mov RCAP2H, #0ffh	; Загрузка двух байт	RCAP2H=0xff	// Загрузка двух байт
mov RCAP2L,# –3	; в регистр RCAP2.	RCAP2L= –36	// в регистр RCAP2.
mov TH2,#0ffh	; Загрузка двух байт	TH2=0xff	// Загрузка двух байт
mov TL2,# –3	; в таймер T2.	TL2= –36	// в таймер T2.
mov T2CON,#34h	; Запуск T2 и генератора бод.	T2CON=0x34	// Запуск таймера и генератора бод.

Некоторые микроконтроллеры последних лет выпуска, такие, например, как DS87C520, DS87C530 производства фирмы Dallas Semiconductor, современные, например, MSC1210 производства фирмы Texas Instruments, оборудованы специальным устройством, которое может изменять циклическую частоту работы процессора и периферийных устройств микроконтроллера, в частности таймера T2. Это устройство, а точнее – регистр CKCON (ClocKCONtrol), присутствующий как в микроконтроллерах DS87C5XX, так и MSC1210, может делить входную частоту, подаваемую на таймер, либо на 12, либо на 4. Так вот, у микроконтроллера MSC1210 скорость обмена, генерируемая T2, не зависит от значения регистра CKCON, а у микроконтроллера DS87C5XX – зависит и может изменяться. Это надо иметь в виду.

Как уже упоминалось, в некоторых микроконтроллерах ADUC8XX производства фирмы Analog Devices, например ADUC816, ADUC824, ADUC832, ADUC834, ADUC836 и др., используется часовой кварцевый резонатор с частотой 32768 Гц. Эти микроконтроллеры оборудованы специальным умножителем частоты (PLL), который умножает входную частоту (32768 Гц) на 384 (максимум), и в результате эффективная тактовая частота, на которой работает процессор и все периферийные устройства (в частности, таймеры), равна 32768·384 = 12582912 Гц. Очевидно, что получить скорость обмена 115200 Гц при такой тактовой частоте

либо с использованием таймеров T1 или T2, либо без них нельзя. Если для микроконтроллеров ADUC834 и ADUC836, оборудованных таймером T3, специально предназначенным для генерации скорости обмена по RS232, этот вопрос снят (см. с. 50), то для того, чтобы микроконтроллеры ADUC816, ADUC824 и ADUC832 могли работать со скоростью обмена 115200 Гц, можно рекомендовать следующий прием.

Как известно, ближайшие числовые значения скоростей обмена по RS232 к числовому значению частоты 32768 Гц составляют 38400 и 28800 бод. Если на вход микроконтроллера XTAL1 (32 вывод корпуса MQFP-52) подать частоту 38400 либо 28800 Гц, то можно получить следующее. При подаче частоты 38400 Гц эту частоту можно программно умножить на 384/2 = 192, в результате эффективная тактовая частота работы процессора составит: 38400·192 = 7372800 Гц. При подаче частоты 28800 Гц, программно умножив ее на 384, можно получить: 28800·384 = 11059200 Гц. Как следует из приводимых примеров, если эффективная тактовая частота работы микроконтроллера равна 7372800 Гц, то получить скорость обмена 115200 бод можно либо вообще без использования таймера (пример 5.1), либо с использованием таймера T2 (пример 5.3). При тактовой частоте 11059200 Гц получить скорость обмена 115200 бод можно, также воспользовавшись таймером T2 (пример 5.3).

Как получить частоты 28800 и 38400 Гц? Кварцевые резонаторы на такие частоты очень большая редкость, они, как правило, весьма громоздки, дорого стоят и имеют небольшую стабильность. Однако, если взять достаточно распространенные кварцевые резонаторы с частотами 1228800, 2457600 и 1843200 Гц, сделать простейший генератор и делитель частоты, то можно добиться требуемого: 1228800/32 = 38400 Гц, 2457600/64 = 38400 Гц и 1843200/64 = 28800 Гц. (Схемы генератора и делителя на 32 или 64 приведены в гл. 9).

Нельзя не упомянуть о двух небольших проблемах, возникающих при использовании частот, отличающихся от 32768 Гц, в микроконтроллерах ADUC8XX.

Первая касается работы цифрового низкочастотного фильтра, установленного в тракт сигма-дельта АЦП. В описании микроконтроллера указано, что частота среза фильтра устанавливается автоматически в зависимости от частоты, с которой производится оцифровка аналогового сигнала (точнее, от частоты Найквиста-Котельникова). Автомат, устанавливающий частоту среза фильтра, в качестве временно́й базы использует как раз частоту 32768 Гц (плюс содержимое PLL). При изменении частоты 32768 Гц в ту или иную сторону частота среза фильтра будет несколько отличаться от стандартного значения. В связи с этим, как утверждают произ-

водители микроконтроллера, возможна некая несогласованность в работе фильтра с АЦП. К счастью, особенной проблемы здесь нет по двум причинам. Во-первых, часто́ты 38400 Гц и 28800 Гц отличаются от частоты 32768 Гц не более чем на 17% и 14% соответственно, а во-вторых, цифровой фильтр $(\sin x/x)^3$, который установлен в этих микроконтроллерах, не идеален, и имеет достаточно пологую (если попытаться ее осреднить) частотную характеристику с множеством полюсов (с затуханием до 120 дБ и более). В связи с этим изменение частоты 32768 Гц на 14% или 17%, на взгляд автора, вряд ли повлияет на результаты работы АЦП.

Вторая проблема касается интервального таймера, входящего в микроконтроллер. Этот интервальный таймер, так же, как и вышеупомянутый цифровой фильтр, использует временну́ю базу, связанную с частотой 32768 Гц. В частности, минимальный интервал времени, с которым работает таймер, равен 1/128 с. Изменение частоты, подаваемой в микроконтроллер, на 14% здесь уже сильно повлияет на точность интервального таймера. Однако, поскольку программисту точно известна частота, с которой работает микроконтроллер (а именно 38400 Гц либо 28800 Гц), на взгляд автора, не составляет большого труда сделать программную коррекцию интервального таймера.

Инициализация RS232 с использованием таймера Т3. В микроконтроллерах ADUC834, ADUC836 введен дополнительный четвертый таймер Т3, который совместно с делителем, также входящим в микроконтроллер, позволяет устанавливать скорость обмена по RS232 вплоть до 230400 бод (в том числе и 115200 бод) с ошибкой, не превышающей 0,2%.

Для установки генератора скорости обмена с помощью таймера Т3 используются два регистра: T3CON и T3FD. Значения бит регистра T3CON приведены в табл. 5.3.

Значение DIV рассчитывается по следующей формуле и округляется до ближайшего минимального целого:

$$\mathrm{DIV} = \frac{\log\left[F_{\text{такт}}/(32 \cdot V_{\text{обм}})\right]}{\log(2)};$$

основание логарифма может быть любым.

Значение T3FD рассчитывается по следующей формуле и округляется до ближайшего целого:

$$\mathrm{T3FD} = \frac{2 \cdot F_{\text{такт}}}{2^{\mathrm{DIV}} \cdot V_{\text{обм}}}.$$

После того как значения DIV и T3FD подсчитаны, можно определить скорость обмена:

$$V_{\text{обм}} = \frac{2 \cdot F_{\text{такт}}}{2^{\mathrm{DIV}} \cdot (\mathrm{T3FD} + 64)}.$$

Таблица 5.3

Номер бита	Название бита	Назначение бита
7	T3EN	*Установка* бита разрешает таймеру T3 генерировать скорость обмена. Когда бит установлен, PCON.7, T2CON.4, T2CON5 игнорируются. *Сброс* бита возвращает генератор скорости обмена к стандарту 8052.
6	–	Не используются
5	–	
4	–	
3	–	
2	DIV2	Фактор двоичного делителя
1	DIV1	
0	DIV0	

DIV2	DIV1	DIV0	Значение делителя
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Так, при CD = 0 (PLLCON = 0) (см. техническое описание микроконтроллера ADUC834) тактовая частота $F_{такт}$ = 12582912 Гц, и для скорости 115200 бод получаем:

$$DIV = \log(12582912/32 \cdot 115200)/\log 2 = 1{,}77 \approx 1;$$

$$T3FD = (2 \cdot 12582912)/(2^1 \cdot 115200) - 64 = 45{,}22 \approx 45 = 2dh.$$

Для скорости 9600 бод при CD = 0, как можно убедиться, T3CON = 85h и T3FD = 12h.

Пример 5.4. Установить скорости обмена 9600 и 115200 бод для микроконтроллера ADUC834, используя таймер T3.

ассемблер, $V_{обм}$ = 115200 бод, $F_{такт}$ = 12582912 Гц	Си, $V_{обм}$ = 9600 бод, $F_{такт}$ = 12582912 Гц

mov SCON,#52h
mov PLLCON,#0
mov T3CON,#81h
mov T3FD,#2dh
	SCON=0x52
	PLLCON=0
	T3CON=0x85
	T3FD=0x12

5.2.3. Команды ввода/вывода

Для ввода/вывода байта по интерфейсу RS232 используется регистр SBUF. Например, для того, чтобы вывести байт по RS232, достаточно поместить этот байт в SBUF. Байт, принятый микроконтроллером по RS232, автоматически помещается в регистр SBUF, прочитав содержимое которого можно получить значение принятого байта.

Пример 5.5. Ввести и вывести байт по RS232.

ассемблер		Си	
mov a,SBUF	; ввести байт из RS232 в аккумулятор	ARRAY(i)=SBUF	// принять байт в ARRAY(i)
mov SBUF,r0	; вывести r0 по RS232	SBUF=K	// вывести K по RS232

5.3. Зависимость скорости обмена информацией по RS232 микроконтроллера с компьютером от типа системы сбора (автономная или компьютерная)

Попытаемся взглянуть на проблему установки высокой скорости обмена по интерфейсу RS232 (в частности, 115200 бод) в микроконтроллерах с несколько иных позиций и выяснить: во всех ли случаях требуется такая скорость обмена. Для этого сделаем некоторое отступление.

Прежде всего, необходимо определить, для каких целей используется микроконтроллер. Системы сбора и обработки информации, поступающей с различного рода датчиков (о применении микроконтроллеров именно в таких системах здесь идет речь), можно условно разделить на два класса.

К первому можно отнести так называемые *компьютерные системы сбора*, когда микроконтроллер расположен в специальном устройстве сопряжения с объектом (УСО), имеющем отдельный корпус, как правило, со своим блоком питания, подключаемым к сети (≈ 220 В). К этому устройству подведены кабели от датчиков, с которых снимается информация. УСО сопрягается с компьютером по интерфейсу RS232, по которому в компьютер передается измерительная информация. Управление УСО осуществляется также компьютером по этому же интерфейсу, т.е. микроконтроллер получает команды от компьютера, выполняет их, и результат выполнения посылает в компьютер. Последний получает предварительно обработанную (как правило, только оцифрованную) микроконтроллером информацию, окончательно ее обрабатывает и выводит результаты этой обработки на экран монитора и/или на принтер. Иногда УСО может находиться на значительном удалении от компьютера (до десятков метров и более); такие системы сбора часто называют удаленными.

Отличительные черты компьютерных систем сбора следующие:

• Обмен информацией микроконтроллера с компьютером идет постоянно, кроме того, в некоторых случаях основная программа работы микроконтроллера может передаваться в него из компью-

тера; в этих условиях от интерфейса требуется повышенные надежность и скорость обмена (115200 бод); в связи с этим *обязательным является возможность программирования микроконтроллера по RS232.*

• "Интеллектуальная" нагрузка на микроконтроллер достаточно низкая; микроконтроллер ничего не считает, ввод/вывод информации осуществляется только по интерфейсу, поэтому программа для микроконтроллера примитивна, как правило, написана на ассемблере и в очень редких случаях ее объем превышает 2 кбайта; основную "интеллектуальную" нагрузку несет программа, написанная для компьютера.

• Повышены требования к точности измерений; особенно это касается точности временных характеристик: погрешности таких временных характеристик, как общее время измерений (T), интервал времени измерений (ΔT) и т.п., как правило, не должны превышать 1 мс, а в некоторых случаях (если, например, такая система используется в различного рода поверочных установках, да еще связанных с приборами коммерческого учета) должна быть на порядок выше.

• Повышены требования к АЦП: многоканальность (не менее восьми), повышенная точность, дискретность измерений должна быть не менее 16 разрядов; АЦП должен иметь самокалибровку; перед АЦП обязательно должен располагаться низкочастотный фильтр, частота среза которого равна частоте Найквиста–Котельникова (в два раза ниже частоты дискретизации АЦП), кроме того, желательно, чтобы фильтр был еще и заграждающим, т.е. имеющим полюс (0). При этом затухание должно быть не менее |–100| дБ на частоте питающей сети (50 Гц).

• Системы, как правило, многоканальные и достаточно универсальные, т.е. имеют несколько (восемь и более) аналоговых, частотных и/или дискретных каналов.

• Имеют свой блок питания, подключаемый к сети, потребление энергии такими системами не имеет особого значения; как правило, микроконтроллер работает при напряжении питания 5 В.

• Выпускаются, как правило, в небольших количествах, причем, львиная доля стоимости системы приходится на программное обеспечение, написанное для компьютера, поэтому стоимость аппаратных средств (а тем более – стоимость микроконтроллера) особого значения не имеет.

Ко второму классу можно отнести так называемые *автономные* системы сбора, представляющие собой по существу приборы, правда, достаточно "интеллектуальные". Здесь микроконтроллер выступает в качестве основы как аппаратных, так и программных средств. Автономные системы сбора представляют собой устрой-

ства, оснащенные не только средствами измерений (датчиками измерительной информации), сбора и обработки измерительной информации, но и средствами индикации результатов измерений (в простейших системах это может быть, например, цифровой семисегментный индикатор с несколькими цифрами, в более сложных – одно-, либо двустрочный алфавитно-цифровой или матричный графический дисплей; иногда автономная система оснащается даже примитивным печатающим устройством, пример – кассовый аппарат), а также средствами ручного ввода информации (например, кнопочной мембранной клавиатурой) или хотя бы примитивного ручного управления.

Автономные системы конструируются в отдельном корпусе и оснащаются своим блоком питания (во многих случаях батарейным). Часто автономные системы содержат в своем составе интерфейс сопряжения с компьютером (например, RS232). Иногда этот интерфейс выполнен только опционально, реже он все-таки используется, но основные его задачи – плановый съем архивной информации, тестирование системы, начальное занесение в систему необходимых коэффициентов и/или режимов работы и т.п. При штатном режиме работы автономной системы интерфейс, как правило, не используется.

Отличительные черты автономных систем сбора следующие:

• Обмен информацией с компьютером по интерфейсу (если он вообще существует) происходит время от времени и достаточно редко; от интерфейса не требуется высокое быстродействие (115200 бод); если автономная система сбора все-таки оборудована интерфейсом RS232 для обмена информацией с компьютером, то *желательна возможность программирования микроконтроллера по RS232.*

• "Интеллектуальная нагрузка" на микроконтроллер достаточно высокая; в связи с этим программа для микроконтроллера сложная, как правило, написана на языке Си и, хотя даже современные компиляторы Си (например C-Keil, V.6.14–6.20) для микроконтроллера и обладают массой оптимизационных свойств (в том числе – оптимизацией объема памяти), эта программа занимает десятки килобайт программной памяти и/или памяти данных; кроме того, повышены требования и к объему ОЗУ.

• Повышенная точность измерения временных характеристик, как правило, не требуется, зато необходим их большой временно́й диапазон: секунды, минуты, часы, сутки, месяцы и даже годы; это нужно для возможного архивирования результатов измерений, так как автономная система (оснащенная, например, батарейным блоком питания с литиевыми элементами) может работать несколько лет.

• Повышены требования к АЦП: повышенная точность, дискретность измерений должна быть не менее 16 разрядов; АЦП должен иметь самокалибровку; перед АЦП обязательно должен находиться низкочастотный фильтр, частота среза которого равна частоте Найквиста-Котельникова (в два раза ниже частоты дискретизации АЦП), кроме того, желательно, чтобы фильтр был еще и заграждающим, имеющим полюс (0). При этом затухание должно быть не менее |–100| дБ на частоте питающей сети (50 Гц).

• Не содержат большого числа однотипных измерительных каналов (как правило, число однотипных каналов не превышает трех–четырех), но *число различного рода устройств*, входящих в состав микроконтроллера, может быть большим (АЦП, ЦАП, таймеры/счетчики, часы, источники опорного напряжения и/или тока и т.п.), т.е. от микроконтроллера требуется повышенная универсальность.

• Во многих случаях работают от батарейного блока питания, поэтому от микроконтроллера требуется пониженное энергопотребление; в связи с этим такие системы, как правило, работают при напряжении 3 В (литиевые батарейки).

• Выпускаются в достаточно больших количествах, поэтому стоимость аппаратных средств (в том числе, стоимость микроконтроллера) играет важную роль, а вот стоимость программного обеспечения, которая иногда даже превышает стоимость программного обеспечения компьютерных систем (программа для микроконтроллера + программа для компьютера), существенного значения не имеет, так как распределяется на большое число устройств.

Теперь сравним микроконтроллеры, специально предназначенные для применения в системах сбора. Из выпускающихся в настоящее время (и предполагаемых фирмами к выпуску в самое ближайшее время) 51-совместимых систем на кристалле это следующие: микроконтроллеры ADUC8XX производства фирмы Analog Devises, микроконтроллеры MSC1210YX производства фирмы Texas Instruments, микроконтроллеры C8051F06X производства фирмы CYGNAL, микроконтроллеры VERSA DSP (VDSP1040) производства фирмы GOAL.

В табл. 5.4 приведены сведения о некоторых общих параметрах микроконтроллеров, а в табл. 5.5 – параметры АЦП рассматриваемых микроконтроллеров. На наш взгляд, именно эти общие параметры микроконтроллеров и параметры их АЦП и определяют тип системы сбора (автономная или компьютерная), в которой целесообразно применять конкретный микроконтроллер. Из сравнения микроконтроллеров по этим двум таблицам, с учетом вышеприведенных требований к микроконтроллерам в зависимости от

Таблица 5.4

Сравнение некоторых параметров микроконтроллеров

Микроконтроллер	Объем памяти программ, кбайт	Объем памяти данных, кбайт	Объем памяти ОЗУ, кбайт	Интерфейс программирования	Универсальный таймер	Часовой таймер	Цена (долл.)
MSC1210Y2	4	[2]	1,28	RS232	+	[3]	10
MSC1210Y3	8	[2]	1,28	RS232	+	[3]	11
MSC1210Y4	16	[2]	1,28	RS232	+	[3]	12
MSC1210Y5	32	[2]	1,28	RS232	+	[3]	14
ADUC816	8	0,64	0,256	RS232	–	+	17
ADUC824	8	0,64	0,256	RS232	–	+	18
ADUC834	62	4	2	RS232	–	+	25
ADUC836	62	4	2	RS232	–	+	22
ADUC844[1]	62	4	2,3	RS232	–	+	
ADUC846[1]	62	4	2,3	RS232	–	+	
C8051F06(0,1,2,3)[1]	64	[2]	4,352	JTAG	–	–	50
VDSP1040[1]	32	[2]	2,304	JTAG, I^2C	–	–	25

Примечания: [1] – бета-версия; [2] – граница между памятью программ и памятью данных перераспределяется пользователем; [3] – может быть сконфигурирован пользователем из универсального таймера.

Таблица 5.5

Сравнение параметров АЦП микроконтроллеров

Микроконтроллер	Количество разрядов	Количество каналов	НЧ фильтр	Частота преобразования (max)
MSC1210Y(2,3,4,5)	24	8	+	2 КГц
ADUC8(16,36,46)	16	5	+	100 Гц
ADUC8(24,34,44)	16 и 24	2 24-разрядных + 3·16-разрядных	+	100 Гц
C8051F06(0,1,2,3)	16	2	–	1 МГц
VDSP1040	16	4	–	50 КГц

той или иной системы сбора, в которой микроконтроллер может быть применен, можно сделать следующий вывод. В компьютерных системах сбора целесообразнее всего использовать микроконтроллеры MSC1210YX, в автономных – все остальные. Микроконтроллеры ADUC816 и ADUC824 могут быть использованы только в достаточно примитивных автономных системах сбора из-за ограниченного объема памяти программ (8 кбайт) и, кроме того, так как автономные системы не требуют высокоскоростного обмена по RS232, в этих двух микроконтроллерах скорость обмена 115200 бод не нужна.

Некоторую информацию о новых разработках фирмы Analog Devices Inc. по микроконвертерам можно найти в интернете: www.analog.com. В частности, помимо двух новых микроконверторов ADUC844 и ADUC846 с "быстрым" 8052-ядром (один цикл – одна команда), приведенных в табл. 5.4, фирма Analog Devices анонсировала еще две микросхемы микроконверторов также с быстрым 52-ядром: ADUC845 и ADUC847. Микроконвертер ADUC845 будет выпускаться с внутренней памятью программ объемом в 8, 32 и 62 кбайт и оснащаться малошумящим 10-канальным мультиплексором, стоящим перед АЦП, а также 12-разрядным ЦАП. ADUC847 – более дешевая версия ADUC845 без ЦАП.

Фирма Texas Instruments (www.ti.com) анонсировала микроконтроллеры MSC1211YX, которые будут обладать всеми свойствами MSC1210YX и содержать 12-разрядный ЦАП (по напряжению и току), а MSC1212YX – свойствами MSC1211YX и содержать аппаратный интерфейс I^2C. Кроме того, анонсирована еще одна микросхема – MSC1200YX, которая будет выпускаться в TQFP48-корпусе, содержать 24-разрядный 8-канальный АЦП, 12-разрядный токовый ЦАП и PLL, позволяющий работать с кварцевым резонатором 32 кГц (помимо обычного высокочастотного).

6. Протоколы (алгоритмы) обмена по интерфейсу RS232

6.1. Классификация протоколов обмена

Как было упомянуто во введении, интерфейс RS232 был задуман как средство обмена информацией компьютера с модемом. В настоящее время он и продолжает использоваться в этом качестве: выносной модем до сих пор сопрягается с компьютером по RS232. Обмен информацией компьютера с модемом происходит с помощью линий квитирования (DTR, DSR, RTS, CTS и т.п.), которые предназначены для синхронизации. Эта синхронизация необходима для того, чтобы обмен информацией низкоскоростного модема с высокоскоростным компьютером происходил без сбоев и потерь. Метод (или алгоритм) такой синхронизации называют *протоколом* обмена. Протоколов обмена несколько. Некоторые из них стали де-факто промышленным стандартом обмена информацией компьютера с модемом.

Суть протокола обмена состоит в следующем. Предположим для примера, что линия DTR компьютера соединена с линией DSR модема, а линия DTR модема – с линией DSR компьютера. Для того чтобы передать байт данных по RS232 в модем, компьютер проверяет свою линию DSR. Если линия DSR компьютера (линия DTR модема) находится в "разрешающем" состоянии (т.е. модем "разрешает" компьютеру передачу), то компьютер передает очередной байт. При "запрещающем" состоянии линии DSR компьютер не передает байт, а ждет разрешения на его передачу. В обратную сторону (т.е. от модема в компьютер) передача информации идет аналогично. Это один из протоколов обмена, который иногда называют протоколом DTR-DSR. Бывают более сложные протоколы с использованием обеих линий DTR-DSR и RTS-CTS. Например, линия DTR используется для указания *направления* передачи (от компьютера к модему или обратно), а линия RTS – для *разрешения* передачи (как в предыдущем случае). Подобные протоколы обмена называют *аппаратными*, так как для синхронизации применяются физические линии (их состояния), т.е. аппаратные средства. Преимущество аппаратных протоколов обмена состоит в высокой надежности и большой скорости обмена, поскольку передатчик и приемник тратят минимум времени на установку и проверку состояний соответствующих линий квитирования. Не-

достаток аппаратных протоколов обмена – необходимость наличия линий квитирования (т.е. дополнительных проводов).

Существуют еще протоколы обмена, которые называют *программными*. В программных протоколах обмена вместо линий квитирования для синхронизации обмена используются те же линии данных (TxD и RxD), по которым передается и принимается информация (линия RxD компьютера соединена с линией TxD модема, а линия RxD модема соединена с линией TxD компьютера). Примером может служить известный протокол обмена XON/XOFF, часто используемый при обмене информацией по RS232 компьютера с принтером.

Суть его заключается в следующем. Компьютер передает информацию (читая перед передачей состояние своей линии RxD). Если по линии передается разрешающий код XON (десятичное значение кода часто равно 17), то передача информации продолжается. При приеме кода XOFF (19) передача приостанавливается, и компьютер ждет, пока не будет передан код разрешения передачи XON.

При программировании микроконтроллеров по RS232 применяют программные протоколы обмена, которые основаны на подсчете контрольной суммы переданного в микроконтроллер пакета информации (до десятков байт). При совпадении контрольной суммы пакета байт, переданных компьютером в микроконтроллер, с подсчитанной микроконтроллером (по принятым байтам), последний передает в компьютер информацию (например, контрольную сумму), разрешающую компьютеру продолжать обмен. Если контрольная сумма не совпадает, микроконтроллер передает в компьютер информацию с требованием еще раз повторить передачу предыдущего пакета данных. Такой протокол является более надежным (и, естественно, более медленным), чем, например, ранее рассмотренный протокол XON/XOFF, однако в данном случае превалирует надежность обмена.

6.2. Высокоскоростной протокол обмена, предложенный автором

6.2.1. Суть протокола обмена

Вернемся еще раз к ранее упомянутому аппаратному протоколу обмена и проанализируем работу линий интерфейса RS232, которые используются при передаче и приеме информации. Предположим, что обмен информацией происходит между компьютером и микроконтроллером.

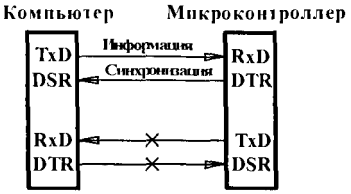


Рис. 6.1. Структурная схема передачи информации из компьютера в микроконтроллер

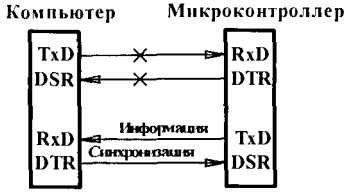


Рис. 6.2. Структурная схема передачи информации из микроконтроллера в компьютер

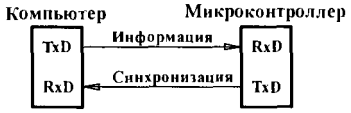


Рис. 6.3. Структурная схема передачи информации из компьютера в микроконтроллер

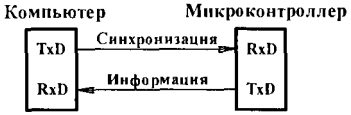


Рис. 6.4. Структурная схема передачи информации из микроконтроллера в компьютер

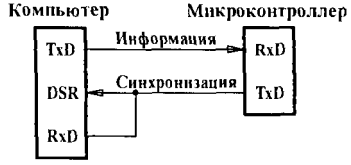


Рис. 6.5. Структурная схема передачи информации из компьютера в микроконтроллер, использующая для синхронизации DSR

При передаче информации из компьютера в микроконтроллер работают *только* следующие линии: TxD и DSR компьютера и RxD и DTR микроконтроллера. А линии RxD и DTR компьютера и TxD и DSR микроконтроллера *не используются.*

При передаче информации из микроконтроллера в компьютер все происходит с точностью до наоборот: работают линии TxD и DSR микроконтроллера и RxD и DTR компьютера. Линии RxD и DTR микроконтроллера и TxD и DSR компьютера *не используются.*

Сказанное иллюстрируется рис. 6.1 и рис. 6.2. Из рисунков, в частности, видно, что при передаче информации из компьютера в микроконтроллер связь между линией RxD компьютера и линией TxD микроконтроллера не используется, а при передаче информации из микроконтроллера в компьютер не используется связь линии RxD микроконтроллера с линией TxD компьютера.

Возможно ли задействовать неиспользуемые линии данных для синхронизации? В этом случае от связи между линиями DTR и DSR компьютера и микроконтроллера можно отказаться. Сказанное иллюстрируется рис. 6.3 и рис. 6.4. Как видно из рис. 6.3 и рис. 6.4, для синхронизации используются линии данных: TxD и RxD. Правомерно ли применять их для таких целей? Ведь синхро-

низация (если проводить аналогию с линиями DTR и DSR) предполагает *не передачу* данных по этим линиям, а *установку* линии TxD в определенное состояние и *чтение состояния* линии RxD. Можно ли устанавливать линии TxD компьютера и микроконтроллера в определенное состояние и можно ли читать состояние линии RxD компьютера и микроконтроллера?

Что касается микроконтроллера, то, поскольку его линии RxD и TxD являются одновременно выводами порта p3.0 и p3.1, а точнее, выводы порта p3.0 и p3.1 обладают альтернативными функциями обмена по RS232 соответственно RxD и TxD, то в микроконтроллере все обстоит благополучно: вывод p3.1 (линию TxD) *можно* устанавливать в единичное и нулевое состояние (например, командами ассемблера: setb TxD, clr TxD или командами Си: TxD = 1, TxD = 0), а вывод p3.0 (линию RxD) можно читать и анализировать (например, на ассемблере: jb RxD,... или jnb RxD,... и на Си: if TxD = 0 ... или if TxD = 1 ...).

С компьютером дело обстоит несколько сложнее. Как было упомянуто при описании порта COM1 интерфейса RS232 компьютера, в *регистре управления линиями (line control register),* имеющем адрес 3fbh, бит 6 предназначен для передачи сигнала BREAK. Это означает, что линию TxD компьютера программно (установив, либо сбросив бит 6 регистра 3fbh) *можно* установить в единичное либо нулевое состояние. Таким образом, при передаче информации в компьютер (рис. 6.4) все обстоит благополучно. А вот состояние линии RxD, как было упомянуто ранее, к сожалению, прочитать *нельзя.* Это означает, что при передаче информации из компьютера в микроконтроллер использовать схему, показанную на рис. 6.3, невозможно. Как же быть? К счастью, выход из создавшегося положения может быть найден достаточно простым способом.

Соединим линию RxD компьютера (см. рис. 6.3) с какой-либо входной линией квитирования, например DSR. Ее состояние (как и состояние любой входной линии квитирования) определить *можно,* читая и анализируя бит 5 *регистра состояния модема (modem status register),* имеющего адрес 3feh. Схема такого соединения показана на рис. 6.5. Из рис. 6.5 видно, что вышеуказанную синхронизацию можно обеспечить, читая и анализируя состояние линии DSR. При передаче информации из микроконтроллера в компьютер состояние линии DSR не имеет никакого значения. Ввод информации должен происходить стандартным образом (т.е. в компьютер просто вводится байт из регистра данных, имеющего адрес 3f8h по линии RxD).

6.2.2. Аппаратные средства протокола обмена

Таким образом, приходим к достаточно простой схеме обмена информацией между микроконтроллером и компьютером, показанной на рис. 6.6. Из рис. 6.6 можно увидеть, что "аппаратная" синхронизация возможна даже при соединении компьютера с микроконтроллером всего по двум линиям (не считая линии заземления SG). Здесь следует сделать некоторое уточнение. Если при сопряжении компьютера с микроконтроллером используется гальванически изолированный интерфейс RS232 (см., например, рис. 4.6 или рис. 4.9), то, естественно, никакой "общей земли" не требуется, поэтому линия SG показана пунктиром.

Необходимо еще добавить следующее. При таком способе сопряжения микроконтроллера с компьютером, поскольку используются всего две информационные линии, требуется всего *два* преобразователя уровней RS232 (если используется общее заземление), а при использовании гальванических развязок – их тоже должно быть *две*.

Помимо информационных связей компьютера с микроконтроллером, как уже ранее упоминалось (см. рис. 2.1), требуется еще как минимум одна связь микроконтроллера с компьютером: микроконтроллер нуждается в запуске и остановке. Если программа в микроконтроллере уже есть, то, как видно из рис. 2.1, сигнал

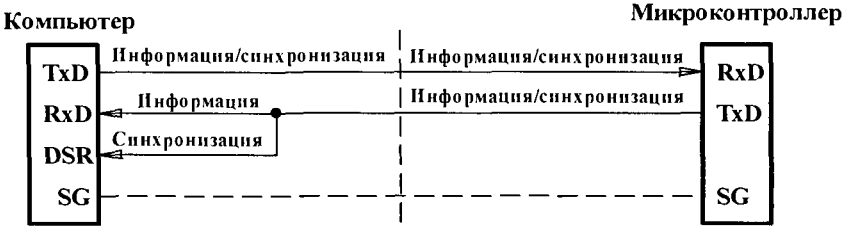


Рис. 6.6. Структурная схема обмена информацией компьютера с микроконтроллером по RS232

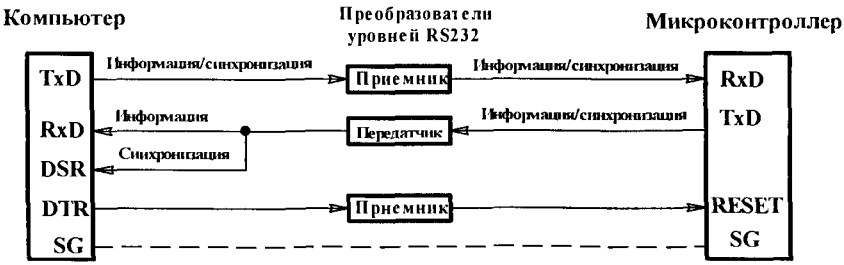


Рис. 6.7. Функциональная схема сопряжения компьютера с микроконтроллером для реализации алгоритма обмена

DTR компьютера (предварительно преобразовав его в TTL-уровень) можно использовать в качестве сигнала RESET микроконтроллера. С учетом вышесказанного окончательно схема сопряжения компьютера с микроконтроллером приобретает вид, показанный на рис. 6.7. Здесь уже линии являются проводами, а стрелки указывают направление передачи.

6.2.3. Программное обеспечение протокола обмена

Уточним (пока в словесном описании) еще раз алгоритм обмена (см. рис. 6.7):

1. Передатчик ждет от приемника разрешение на передачу байта.
2. Приемник при готовности приема дает передатчику разрешение на передачу байта.
3. Передатчик, получив разрешение от приемника, передает очередной байт.
4. Приемник, принимая очередной байт, сбрасывает разрешение передатчику передавать следующий байт и не восстанавливает это разрешение до тех пор, пока не примет этот байт полностью и не будет готов к приему следующего байта.
5. Передатчик, передав очередной байт, ждет запрета от приемника на передачу следующего байта[1].
6. Если передатчику есть еще что передавать, он переходит к шагу 1, иначе, если передан последний байт, то передача заканчивается.

В качестве примера приведем программы для компьютера и микроконтроллера, реализующие вышеописанный алгоритм обмена по RS232 (и схему рис. 6.7).

Пример 6.1.

Задание для компьютера. Вывести по интерфейсу RS232 из компьютера в микроконтроллер строку символов определенной длины, затем принять из микроконтроллера строку символов такой же длины. Вывести переданную и принятую строки на экран монитора для визуального сравнения. Скорость обмена установить 115200 бод.

Задание для микроконтроллера. Принять из компьютера по RS232 строку определенной длины, запомнить ее (строку), а затем передать ее обратно в компьютер. Скорость обмена установить 115200 бод.

---

[1] Этот шаг требуется более скоростному передатчику (например, компьютеру). Менее скоростной передатчик (например, микроконтроллер) этот шаг может пропустить.

Несколько усложним (а возможно, и упростим, но уточним) поставленную задачу небольшим дополнительным требованием: компьютер должен перед передачей информационных байт строки передать число, равное количеству байт строки, т.е. ее длину, а микроконтроллер – принять эту длину и в дальнейшем принимать количество информационных байт, равное этой длине.

Решение поставленной задачи для наглядности представим двумя вариантами, различающимися языками программирования.

Первый вариант.

А. Компьютерная программа представлена на языке Бейсик с двумя inline-подпрограммами на ассемблере в *.com-формате. Название файла программы на Бейсике RdAt2051.bas. Названия ассемблерных подпрограмм: первой – prgrsis.com, второй – p86rd.com. Текстовые файлы подпрограмм: prgrsis.asm и p86rd.asm. Подпрограмма prgrsis.asm передает строку байт в микроконтроллер, а подпрограмма p86rd.asm – принимает ее.

Бейсик – Turbo Basic V1.0 фирмы Borland (tb.exe). Ассемблер – Turbo Assembler Version 1.0 1988 фирмы Borland (tasm.exe). Линкер – Turbo Link Version 2.0 1987, 1988 фирмы Borland (tlink.exe).

Для получения файлов в *.com-формате необходимо использовать опцию /t, например:

```
tlink.exe prgrsis.obj  /t
```

Б. Программа для микроконтроллера представлена на ассемблере. Название программы – inrs115.asm. В качестве микроконтроллера взят один из современных – MSC1210Y5. Ассемблер – 8051 Macro Assembler Version 4.02a 2500 A.D. Software Inc, 1985. Линкер – 2500 A.D. Linker, Version 4.02d, 1985. *.bat – файл для получения файла программы в *.hex-формате:

```
----------
x8051 %1.asm
pause
link -c %1.obj
----------
```

Второй вариант.

Компьютерная программа представлена на языке Кларион. Название программы outbytec.cla. Кларион V.3100 – Clarion Software Inc., использована опция Static Model. Программа для микроконтроллера представлена на языке Си – C51 Keil Software Inc. V6.14. Название программы inrs115ac.c

Необходимо отметить, что представленные далее программы для компьютера одинаково хорошо работают (и работали) на компьютерах с процессорами: i486-DX2-66 МГц, Pentium 1 – 100 МГц и Pentium 4-1700 МГц.

```
'Программа на BASIC'е:  RdAt2051.bas
'-----------------------------------

sub REC inline
$inline "prgrsis.com"
end sub

sub RED inline
$inline "p86rd.com"
end sub

'------
' Инициализация последовательного порта
'------
' Установка скорости:
' 96- 1200,48- 2400,24- 4800,12 - 9600
'  6-19200, 3-38400, 2-57600, 1-115200
'------
out &h3fb,&h80
out &h3f8,1    '-Коэффициент скорости
out &h3f9,0    '-Старший байт делителя
'------
' Установка режима
'------
out &h3fb,7 '2 стопа,8 бит,нет паритета
out &h3f9,0 'Запрет всех прерываний по COM-порту
'------

out &h3fc,0 'Сброс микроконтроллера.
delay .2
out &h3fc,1 'Запуск микроконтроллера.
delay .2

          if (inp(&h3fe) and &h10)=&h10 then goto OK
'          goto WYKL
OK:
K=1
START:
```

```
cls
locate 2,7
print""
 C$="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzАБВГДЕЖЗИЙКЛМНОПРСТУФХЦ"
 'Ц-75-й символ.
C1$="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
   print " C$=";C$
   print "C1$=";C1$
'-------
print"Начало передачи"
'----------------------------------
call REC(C$)
a=inp(&h3f8)' Холостой ввод
a=inp(&h3f8)' Холостой ввод
CALL RED(C1$)
'----------------------------------
print "-------------------------------------------------------------"
print "Конец приема"
print " C$=";C$
print "C1$=";C1$
locate 10,5
   print "K=";K
   K=K+1
locate 15,15

print " Нажми ENTER - для продолжения и ESC - для выхода"
print""
print""

       while not instat
       wend
       a$=inkey$
       if a$=chr$(13) then cls: goto WWW
       cls
       goto E
'------------------
' Вывод байта
'------------------
```

```
WWW:
       wait &h3fd,&h20
       wait &h3fe,&h20
       out  &h3f8,&h40  'Вывод байта 40h("@")-продолжения работы.
       wait &h3fe,&h20,&h20
       wait &h3fd,&h40
'---------------------
       goto START

WYKL:
cls
locate 10,25
print " Система выключена"
locate 15,22
print " Нажми ENTER - для выхода"
print""
print""
       while not instat
       wend
       a$=inkey$
       if a$=chr$(13) then cls: goto E
       goto WYKL
E:
       out &h3fc,0' Установка DTR=-9в., RTS=-9в.Сброс микроконтроллера.
       cls
end

****************************************
;Программа на ассемблере компьютерного процессора X86,
;осуществляющая высокоскоростной синхронный вывод строки
;байт компьютером в однокристальный микроконтроллер по
;интерфейсу RS232 и использующаяся как "inline"
;подпрограмма в *.com-формате
;в программе на TB RdAt2051.EXE(BAS).
;Название настоящего файла - prgrsis.asm,
;com-файла  - prgrsis.com.
;****************************************
;          Макрос вывода байта.
;============================
```

```
OUTBYTE macro                   ;Байт должен быть в bl
local  MET1,MET2,MET3,MET4

       mov dx,3fdh      ;Ожидание готовности буффера передатчика
MET1:  in al,dx         ;компьютера принять новый байт
       test al,20h      ;(transmitter empty).
       jz MET1          ;(не 40h, а 20h !!!)

       mov dx,3feh      ;Ожидание разрешения
MET2:  in al,dx         ;передачи от микроконтроллера
       test al,20h      ;(установки линии DSR).
       jz MET2

       mov al,bl        ;Вывод байта в буффер передатчика
       mov dx,3f8h      ;и инициализация аппаратного
       out dx,al        ;вывода байта через порт RS232.

       mov dx,3feh      ;Ожидание запрета
MET3:  in al,dx         ;передачи от микроконтроллера
       test al,20h      ;(сброса линии DSR).
       jnz MET3

       mov dx,3fdh      ;Ожидание окончания передачи байта
MET4:  in al,dx         ;(установки бита "Transmitter holding
       test al,40h      ;register empty. OK to send"),  (не 20h, а 40h !!!)
       jz MET4          ;т.е. опустошения сдвигового регистра передатчика.
;----------
endm
;======================
;  Основная подпрограмма
;======================
PROGRAM segment
        org 100h
assume  cs:PROGRAM,ds:PROGRAM,es:PROGRAM,ss:PROGRAM
START:  push bp
        mov bp,sp
        push es
        push ds
```

```
;-----
        les di,[bp+6]
        mov dx,ds:[0]           ;Сопряжение с ТВ,
        mov ds,dx               ;прием строки C$ из ТВ.
        mov si,es:[di+2]
        mov cx,es:[di]
        and cx,7fffh            ;Длина строки C$ в cx.
;-----
; Передача строки по RS232 в микроконтроллер.
;-----
       mov bl,cl                ;Передача младшего
       OUTBYTE                  ;байта длины строки C$.
       mov bl,ch                ;Передача старшего
       OUTBYTE                  ;байта длины строки C$.

MET:   mov bl,byte ptr [si]     ;Вывод
       OUTBYTE                  ;строки C$   !!! (байт в bl).
       inc si                   ;побайтно
       loop MET                 ;через RS232.
;-----
       pop ds
       pop es
       pop bp
PROGRAM ends
end START

================================================================================
p86rd.asm
;-----------------
; Программа чтения байт по RS232C.
;-----------------
;      Макросы:
;-----------------
;   Макрос ввода байта
;(введенный байт - в al)
;------------------
INBYTE  macro
        local MET1,MET2
```

```
        mov  dx,3fbh            ;Установка
        mov  al,47h             ;линии TxD
        out  dx,al              ;(разрешение передачи).

        mov dx,3feh             ;Ожидание
MET1:   in al,dx                ;старт-бита,
        test al,20h             ;(установки
        jz MET1                 ;линии DSR).

        mov  dx,3fbh            ;Сброс
        mov  al,07h             ;линии TxD
        out  dx,al              ;(запрет передачи).

        mov  dx,3fdh            ;Ожидание
MET2:   in   al,dx              ;окончания прихода
        test al,1               ;байта в буффер приемника
        jz   MET2               ;(установки бита DR-data ready).

        mov  dx,3f8h            ;Ввод байта из буффера приемника
        in   al,dx              ;(введенный байт - в al).
endm
;--------------
;   Основная подпрограмма
;---------------
PROGRAM SEGMENT
org 100h
ASSUME CS:PROGRAM,DS:PROGRAM,ES:PROGRAM,SS:PROGRAM
START:  push bp                 ;Сохранение адреса
        mov bp,sp               ;возврата в стеке
        push es
        push ds
;-----------------------------------
        les di,[bp+6]           ;Сопряжение  с
        mov dx,ds:[0]           ;ТВ.
        mov ds,dx
        mov si,es:[di+2]        ;Смещение строки C1$ в si.
        mov cx,es:[di]          ;Длина
        and cx,7fffh            ;строки C1$ - в cx.
```

```
;--------------------------------
MET:    INBYTE                  ;В al - считанный байт.
        mov [si],al
        inc si
        loop MET
;--------------------------------
        pop ds
        pop es
        pop bp;Вызов адреса возврата из стека
PROGRAM ENDS
end     START
;--------------------
```

```
;inrs115.asm
;-------------------------------------------------------------------------
; Программа приема МК строки символов по RS232 (с передачей длины строки)
; и вывода символов из МК в компьютер.
; Fcore=11059200 Гц (Fкв=28800 Гц). Скорость обмена - 115200 и 9600 бод.
; Синхронизация - с помощью линий TxD и RxD.
; Для проверки использовать программы OUTBYTEC.EXE(CLA) и OUTBYTE1.EXE(BAS).
;-------------------------------------------------------------------------
;Макрос ввода байта
;-------------------------------------------------------------------------
;  В МАКРОСАХ НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ РУССКИЙ ЯЗЫК В КОММЕНТАРИЯХ!!!
;-----------------------------------------------------------------------
; Алгоритм ввода байта
;-----------------------
INBYTE  .macro
         clr TxD        ;Enable transfer
         jb RxD,$       ;Waiting start-bit
         setb TxD       ;Disable transfer
         jnb ri,$       ;Waiting set flag ri
         mov a,sbuf     ;Input byte from buffer
         clr ri         ;Clear flag ri
        .endm
;----------------------------
```

```
;   Алгоритм вывода байта
;------------------------------
OUTBYTE .macro
         jb RxD,$       ;Waiting enable transfer
         mov sbuf,a     ;Output byte in buffer
         jnb RxD,$      ;Waiting disable transfer.WARNING!!!
         jnb ti,$       ;Waiting output last bit (flag ti)
         clr ti         ;Clear flag ti
        .endm
;---------------------------------
           .DATA
;--------------------------------
; Переменные
;--------------------------------
PLLCON      equ 0d7h  ; Умножитель частоты для работы процессора
T2CON       equ 0c8h
RCAP2H      equ 0cbh
RCAP2L      equ 0cah
;--------------------
CKCON       equ 08eh
USEC        equ 0fbh
PDCON       equ 0f1h
ACLK        equ 0f6h
PASEL       equ 0f2h
;-----------------------------
           .CODE
           .org 0000h ;Вектор прерывания по RST
            jmp MAIN
           .org 0003h ;Вектор прерывания по INT0
            reti
           .org 001bh ;Вектор прерывания по TF1
            reti
           .org 0053h ;вектор прерывания от TIC
            reti
;---------------------------------------
; Основная программа
;---------------------------------------
          .org 0060h
```

```
          mov sp,#20h
          mov PASEL,#20h

MAIN:  ;    mov PLLCON,#0h   ;(11059200 Гц при кварце 28800 Гц в ADUC816).
       ;    clr p3.5  ;ON
;----------------------------

;----------------------------
; Инициал. посл. порта
;-------------------------------------------------------------------------
       mov SCON,#11101100b ;8 бит,2 стопа,REN=0(запрет приема ReciveENable=0),
                           ;TI=0,RI=0,Vобм определяется T1 или T2,
                           ;SM1=SM2=1 режим 3. RB8=TB8=1.
;---------------------------------------
;Инициализация таймера T2 - как таймера.
;---------------------------------------
       mov T2CON,#00110100b; TF2=0,EXF2=0,RCLK=1,TCLK=1,EXEN2=0,
                           ; TR2=1-start,CNT2=0,CAP2=0.
;-----------------------------------------------
       mov RCAP2H,#0ffh    ;Fобм.=Freq/(32*(65536-(RCAP2H,RCAP2L)))
                           ;65536-RCAP2H,RCAP2L=65536-65533=3 (253=fch)
       mov RCAP2L,#-3      ;Fобм.=11059200/(32*3)=115200 бод.
       ;mov RCAP2L,#-1     ;Fобм.= 3686400/(32*1)=115200 бод.
       ;mov RCAP2L,#-2     ;Fобм.= 7372800/(32*2)=115200 бод.
       ;mov RCAP2L,#-36    ;65536-RCAP2H,RCAP2L=65536-(65536,(256-36))=36
                           ;Fобм.=11059200/(32*36)=9600 бод.
       ;mov RCAP2L,#-12    ;Fобм.= 3686400/(32*12)=9600 бод.


       mov PCON,#80h
       mov CKCON,#20h

;       mov USEC,#10
;       mov PDCON,#1dh
;       mov ACLK,#1

;       mov PASEL,#20h
```

```
;-------------------------------------------------------------------------
;Ввод байт по посл. порту
;--------------------------
START:      setb REN    ;Разрешение приема.

            mov dptr,#0
            INBYTE
            mov r0,a ; Мл. байт длины
            INBYTE
            mov r1,a ; Ст. байт длины

spin:       INBYTE; Байт в a
;----
            push a
;----
            inc dptr
            mov a,r1
            cjne a,dph,spin
            mov a,r0
            cjne a,dpl,spin
;-------------------------------------
; Вывод байт по посл. порту
;-------------------------------------
           clr REN          ;Запрет приема.
spout:     pop a
           OUTBYTE
           djnz r0,spout
;--------------------------------
; Ввод команды продолжения работы
;--------------------------------
           setb REN         ;Разрешение приема.
IB:        INBYTE
           cjne a,#40h,E
           jmp START

E:         mov 0f2h,#0
           jmp $
           .end
```

```
!-----------------------------------------------
!outbytec.cla
!
!Программа передачи и приема строки из 75 символов по RS232
!-----------------------------------------------

                    PROGRAM

!-----------------------------------------------
! Определение переменных
!-----------------------------------------------
C                   byte
A                   byte
B                   byte
S                   string(75)
BT                  byte
M                   byte,dim(75),over(S)
S1                  string(75)
M1                  byte,dim(75),over(S1)
i                   ushort
j                   ushort
k                   ushort
Y00h                byte
Y01h                byte
Y03h                byte
Y04h                byte
Y07h                byte
Y12                 byte
Y47h                byte
Y80h                byte
Y0fh                byte
!-----------------------------------------------

                    CODE

!--------------------------------
!    Очистка экрана
!--------------------------------
```

```
          blank
          k=1
!------------------------------
        Y00h=00h;Y01h=01h;Y03h=03h;Y04h=04h;Y07h=07h;Y0fh=0fh;Y12=12;Y47h=47h;Y80h=80h
                    do INIT        !Инициализация RS232

    goto OK

    in(3feh,B)

    if band(10h,B)=10h then goto OK.

   show(8,28,'Система выключена')
          goto WYKL

OK

!-----------------------------------------------
!                   i=1
!        setcursor(4,1)
!INB                do INBYTE
              !    type(chr(M1[i]))
!                   goto INB
!-----------------------------------------------

          setcursor(20,10)
          type(k)

          k=k+1

 S='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdifjhigklmnopqrstuvwxyzАБВГДЕЖЗИКЛМНОПРСТУФХЦЧ'

         ! Ч-75-й символ

       show(1,1,'Начало передачи из PC')
```

```
!---------------------------
!           i=1
!           M[1]=41h
!OUB        do OUTBYTE
    !     type(chr(M[1]))
!           goto OUB
!---------------------------
   i=1
   C=M[1]
   M[1]=75
   do OUTBYTE !Передача мл. байта длины (75)
   M[1]=0
   do OUTBYTE !Передача ст. байта длины (0)

      show(2,1,'Длина передана')

   M[1]=C
   setcursor(3,1)
   loop i=1 to 75
   do OUTBYTE

 !type(chr(M[i]))

   .

   show(4,1,S)
   show(8,1,'Конец передачи из PC')
   show(9,1,'_____________________________________________________')

!-------------------------------------------------------------------------
  in(3f8h,B)      !Холостой ввод - делать ОБЯЗАТЕЛЬНО !!!(Для сброса бита DR-data ready).
  in(3f8h,B)      !Холостой ввод - делать ОБЯЗАТЕЛЬНО !!!(Для сброса бита DR-data ready).
!---------------------------------

   show(10,1,'Начало передачи из микроконтроллера')

   setcursor(12,1)
   loop i=1 to 75
```

```
   do INBYTE
  ! type(i)
   type(chr(M1[i]))
   .

   show(15,1,S1)
   show(18,1,'Конец передачи из микроконтроллера')
   show(19,1,'================================')
!--------
   show(22,26,'Для продолжения нажми ENTER,выход-ESC')

                  loop until keyboard()
                      ask
                      if keycode()=256 then break.      !esc
                    if keycode()=257                    !enter
                       blank
                       i=1
                       M[1]=40h
                       do OUTBYTE
                       goto OK
                    .
                  .
                  goto E
!---------------------------------------
WYKL show(23,26,'Для выхода нажми ESC')
                  loop until keyboard()
                    ask
                    if keycode()=256 then break.      !esc
                  .
!---------------------------------------
E               blank
                out(3fch,Y00h) !Сброс микроконтроллера
                return
!--------------------------------
! П/п инициализации RS232
!--------------------------------
INIT            routine
!--------------------------------
```

```
!             Инициализация COM-порта.
!-------------------------------------------------------------------
! Установка скорости обмена.
!---------------------------
    out(3fbh,Y80h) ! DLAB=1 для установки делителя.
    out(3f8h,Y01h) ! Установить мл.б. скор.:Y01h-115200 бод,Y12-9600 бод.
    out(3f9h,Y00h) ! Установить ст.б. скор.=0.
!-------------------------------------------------------------------
! Установка режима.
!------------------
    out(3fbh,Y07h) ! DLAB=0,Режим: 8 бит данных,2 стопа,нет пар.,сброс TxD.
!-------------------------------------------------------------------
! Установка запрета прерываний по COM-порту.
!-------------------------------------------
    out(3f9h,Y00h) ! Запрет всех прерываний по порту 3f8h.
!-----------------------------------------------------------

!-------------------------------------
! Инициализация микроконтроллера.
!--------------------
    out(3fch,Y00h)    ! Сброс линии DTR (Reset микроконтроллера).
    beep (0,20)       ! Задержка 0.1 сек.
    out(3fch,Y01h)    ! Установка линии DTR (Запуск микроконтроллера).
    beep (0,20)       ! Задержка 0.1 сек.
    in(3f8h,B)        ! Холостой ввод - для сброса бита 0(DR) в 3fdh (в "0").
    beep(0,20)
                exit
 !----------------------------------------
 ! П/п ввода байта (байт в M1[i] )
 !----------------------------------------
INBYTE          routine
 out(3fbh,Y47h)                          !Установка TxD (разрешение передачи)
 loop;in(3feh,B);if band(B,20h)<>0;break.. !Ожидание старт-бита (уст-ки DSR)
 out(3fbh,Y07h)                          !Сброс линии TxD (запрет передачи)
 loop;in(3fdh,B);if band(B,1)<>0;break..  !Ож-е конца прихода байта(бит "DR"-data ready)
 in(3f8h,M1[i])                          !Чтение байта данных
                exit
!----------------------------------------
```

```
! П/п вывода байта (байт в M[i] )
!----------------------------------------
OUTBYTE            routine
 loop;in(3fdh,B);if band(B,20h)<>0;break.. !Ож-е гот-ти передатчика (transmitter empty).
 loop;in(3feh,B);if band(B,20h)<>0;break.. !Ож-е разреш.передачи(установки DSR)
 out(3f8h,M[i])                           !Вывод байта
 loop;in(3feh,B);if band(B,20h)=0;break..  !Ож-е запрета передачи(сброса DSR)
 loop;in(3fdh,B);if band(B,40h)<>0;break.. !Ож-е выхода байта из PC(trasmitter holding
                                          !register empty.OK to send).
                 exit
 !----------------------------------------

================================================================================
inrs115ac.c

#include "stdarg.h"
#include "stdio.h"    /* для printf */
#include "stdlib.h"
#include "aduc834.h" /* для RS232 */
void initrs(void) {
     PLLCON=0;
     T3CON=0x81;
     T3FD=0x2d;
     SCON=0x40;
}
void outbyte(unsigned char byte) {
   while (RXD);
   SBUF=byte;
   while (!RXD);
   while (!TI);
   TI = 0;
}

   unsigned char inbyte() {
   unsigned char byte;
   TXD=0;
   while (RXD);
   TXD=1;
```

```
    while(!RI);
    byte=SBUF;
    RI = 0;
    return(byte);
}

/*************************************************/
void main(void)
 {
    unsigned char c,lenl,lenh,a[80];
    int i,l;
    initrs();
START:
    REN=1;
    lenl=inbyte();
    lenh=inbyte();
    l=lenh*256+lenl;

    for(i=0;i<l;i++){
    a[i]=inbyte();

    }
    REN=0;
    for(i=0;i<l;i++){
    outbyte(a[i]);
    }
    REN=1;
    c=inbyte();
    if (c==0x40) goto START;
E:
    goto E;
}

/***********************************/
```

7. Применение интерфейса RS232 для загрузки памяти программ микроконтроллера

7.1. Предварительные замечания

Вообще говоря, выражение "загрузка памяти программ микроконтроллера" не совсем удачно; существует еще термин "программирование микроконтроллера", который сейчас часто применяют. Однако мы умышленно не стали использовать в заглавии термин "программирование", так как он в последнее время стал чаще применяться именно как обозначение "загрузки внутренней памяти программ" микроконтроллера. Между тем, как известно, микроконтроллер может работать по программе, которая загружена не только в его внутреннюю память программ (иногда такой памяти программ микроконтроллер не имеет вовсе), но и во внешнюю. Подавляющее число микроконтроллеров (у которых присутствует внутренняя память программ довольно большого объема) оснащены аппаратными средствами, позволяющими управлять внешней памятью программ (точнее – выполнять команды программы, загруженной во внешнюю память).

Этот механизм широко известен: микроконтроллер выставляет в порт p2 старший байт адреса программы, в порт p0 – младший байт адреса, затем "защелкивает" младший байт адреса в какомлибо 8-разрядном регистре (например, КР1564ИР33-SN75HCT573) с помощью сигнала ALE (Address Latch Enable – разрешение защелкивания адреса). Порт p2 напрямую подключается к адресным линиям старшего байта адреса микросхемы памяти с загруженной программой, выход регистра – к линиям младшего байта адреса. Затем на микросхему памяти подается сигнал PSEN (Programm Store ENable – разрешение чтения памяти), байт программы (с соответствующей командой) выставляется микросхемой памяти на линиях данных порта p0, вводится в микроконтроллер по этому же порту p0 (который сигналом ALE переводится в высокоимпедансное состояние) и затем выполняется. Этот механизм позволяет адресоваться к внешней памяти программ объемом 64 кбайт. Некоторые современные микроконтроллеры оснащены механизмом, позволяющим адресоваться к бóльшему объему внешней памяти программ (до 16 Мбайт) с помощью еще двух байт адреса (например, ADUC8XX).

Необходимо отметить, что внешняя память программ может быть организована с применением как микросхем ПЗУ (ROM), так и микросхем ОЗУ (RAM). В первом случае предварительно запрограммированную с помощью программатора микросхему ПЗУ вставляют в панельку платы (или впаивают в нее) с микроконтроллером. Во втором – микросхема ОЗУ впаена в плату. Так вот – загрузить программу в эту микросхему ОЗУ (при наличии некоторых небольших дополнительных аппаратных средств) можно, применив интерфейс RS232.

Недостатки применения ПЗУ известны: 1) программу невозможно защитить от несанкционированного доступа, так как ее легко считать и скопировать, 2) программу нельзя оперативно изменить. Достоинства ПЗУ – минимум аппаратных средств.

Недостатки применения ОЗУ – наличие дополнительных аппаратных и программных средств. Достоинства ОЗУ: 1) возможность оперативного изменения программы, 2) несанкционированный доступ к программе значительно затруднен в связи с тем, что при выключении питания информация в ОЗУ теряется.

Помимо загрузки внешней памяти программ в микроконтроллер, как уже говорилось, интерфейс RS232 можно использовать для загрузки внутренней памяти программ микроконтроллера или, как в настоящее время принято говорить, – его программирования.

Программирование микроконтроллера возможно по нескольким последовательным интерфейсам: SPI (например, микроконтроллер AT89S8252 производства фирмы ATMEL), его разновидность JTAG (микроконтроллеры C8051FXXX производства фирмы CYGNAL, VERSA DSP производства фирмы Goal Semiconductor inc.), I^2C (микроконтроллер VERSA DSP), а также по рассматриваемому здесь RS232 (микроконтроллеры DS5000(T), DS2250(T) производства фирмы Dallas Semiconductor, ADUC8XX производства фирмы Analog Devices, MSC1210YX производства фирмы Texas Instruments).

Поскольку по сравнению с программированием по параллельному интерфейсу программирование по последовательному интерфейсу требует от микроконтроллера минимального числа выводов, данная операция применима к микроконтроллеру, расположенному на плате готового изделия. В связи с этим эту операцию в настоящее время называют еще “программированием в системе” (in-system-programming). Кстати, упомянутую выше загрузку внешней памяти программ (ОЗУ) микроконтроллера с помощью интерфейса RS232 также можно отнести к программированию в системе. Программирование в системе имеет два неоспоримых преимущества перед традиционным программированием микроконтроллера с помощью программатора. Первое – быстрое написание

и отладка программы. Второе – программирование микроконтроллера непосредственно в готовом изделии.

Вообще, программирование в системе имеет некоторую историю. Первой (разумеется, применительно к 51-совместимым микроконтроллерам) программирование в системе предложила фирма Dallas Semiconductor, выпустив микроконтроллеры DS5000(T) и DS2250(T) (и им подобные soft-microcontrollers). Это было в 1993 г. Микроконтроллер DS5000(T) сконструирован следующим образом. Он состоит из микроконтроллера DS5000FP, имеющего немультиплексируемую шину адреса/данных для сопряжения с ОЗУ емкостью до 128 кбайт. В состав DS5000(T) входит память (ОЗУ), часы(T) с кварцевым резонатором на 32768 Гц и литиевая батарейка напряжением 3 В достаточной емкости, чтобы сохранять содержимое памяти после отключения основного питания в течение 10 лет. Микроконтроллер DS5000FP содержит в своей постоянной памяти (ROM) программу-загрузчик (bootstrap loader). Программа-загрузчик предоставляет возможность пользователю загрузить свою программу в память программ микроконтроллера в intel-hex-формате. Для этого предусмотрена специальная команда (L), которую пользователь должен подать в микроконтроллер. Помимо команды загрузки (L) в программе-загрузчике предусмотрены и другие команды (всего около 20), которые позволяют прочитать содержимое некоторой области памяти, заполнить ее определенной информацией, закрыть для чтения, т.е. установить бит секретности, стереть и т.п.

Микроконтроллер DS5000FP, ОЗУ, таймер с кварцевым резонатором и батарейка расположены в стандартном DIP-корпусе с 40 достаточно прочными выводами, позволяющими многократно устанавливать микросхему в панельку.

При программировании в системе уже не нужно было при разработке какого-либо программного обеспечения бесконечно программировать микроконтроллер в программаторе. Таким образом, можно было применять микроконтроллер DS5000(T) в качестве настоящего эмулятора и полностью использовать все возможности (и выводы) микроконтроллера при работе в реальном масштабе времени. По сравнению с фирменными эмуляторами, стоимость которых достаточно высока, эмулятор на базе этого микроконтроллера, основанный только на загрузке и запуске программы, не имеет пошагового режима (хотя запрограммировать подобный режим работы DS5000(T) позволяет). Но пошаговый режим используется достаточно редко: в основном при программировании вычислительных операций или, например, при расчетах временных задержек. Для написания программ с подобными и некоторыми другими приложениями с успехом могут быть использованы программные *симуляторы*, которых великое множест-

во. Что касается точек останова, используемых в профессиональных эмуляторах, то достаточно опытному программисту не составляет большого труда остановить программу на определенном шаге, чтобы посмотреть результат. К сожалению, из-за относительно высокой цены (около 100 долл.) применять микроконтроллеры DS5000(T) в готовых изделиях достаточно накладно.

Следующий шаг на пути к программированию в системе был сделан фирмой ATMEL, которая несколькими годами позже выпустила микроконтроллер AT89S8252. Его программирование возможно по интерфейсу SPI с достаточно высокой скоростью, а значит, возможен режим программирования в системе. Стоимость AT89S8252 составляет всего несколько долларов, поэтому его применение, например, при создании систем сбора результативно. Подключив к этому микроконтроллеру хороший АЦП (например, DS1210 фирмы Burr Brown – сейчас Texas Instruments) с 8-канальным аналоговым коммутатором и несколько таймеров (например, I8254), получим достаточно дешевую систему сбора с очень хорошими точностными свойствами.

И, наконец, последний шаг к программированию в системе в последнее время сделали сразу несколько фирм, выпустив микроконтроллеры ADUC8XX, MSC1210, C8051F06X и VERSA-DSP, о которых было уже несколько раз упомянуто ранее.

Далее будут приведены примеры использования интерфейса RS232 для загрузки памяти программ микроконтроллеров.

7.2. Пример применения RS232 для загрузки внешней памяти программ микроконтроллера P80C552

Рассмотрение этого примера (и всех остальных) разобьем на две части. Вначале приведем аппаратные средства, т.е. принципиальную схему, а затем – программное обеспечение.

7.2.1. Аппаратные средства

Принципиальная схема подобного сопряжения микроконтроллера с компьютером приведена на рис.7.1.

Из схемы видно, что микроконтроллер P80C552 – DD1 (показаны только выводы, имеющие отношение к излагаемому) сопряжен с внешней памятью программ, реализованной на микросхеме ОЗУ емкостью 8 кбайт марки 62C64 – DD5 по мультиплексированной шине адреса/данных. Это мультиплексирование осуществляется с помощью 8-разрядного регистра AHCT573(ИР33) – DD4. Запись в регистр производится сигналом ALE, а чтение содержимого памяти – сигналом PSEN. Дешифратор HCT139(ИД14) – DD6

служит для разделения адресного пространства памяти и периферийных устройств, сопряженных с P80C552 по тому же параллельному интерфейсу адреса/данных. Для примера показано одно из них – это таймер I82C54 – DD7. Сигналом разделения адресного пространства является A15. Резисторная сборка NR1, как это обычно делается, подключена к порту p0. P80C552 сопряжен с компьютером по RS232 посредством двух выводов: TxD и RxD, соединенных соответственно с линиями RxDC и TxDC. Сигнал RxDC подается на вход преобразователя уровня TTL в уровень RS232 (на схеме не показан), а с его выхода – на вход RxD COM-

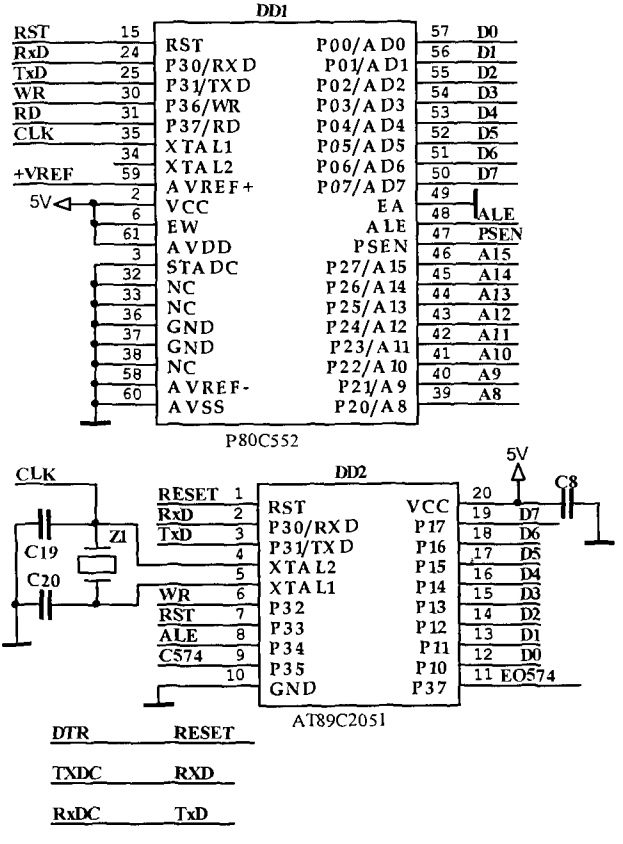
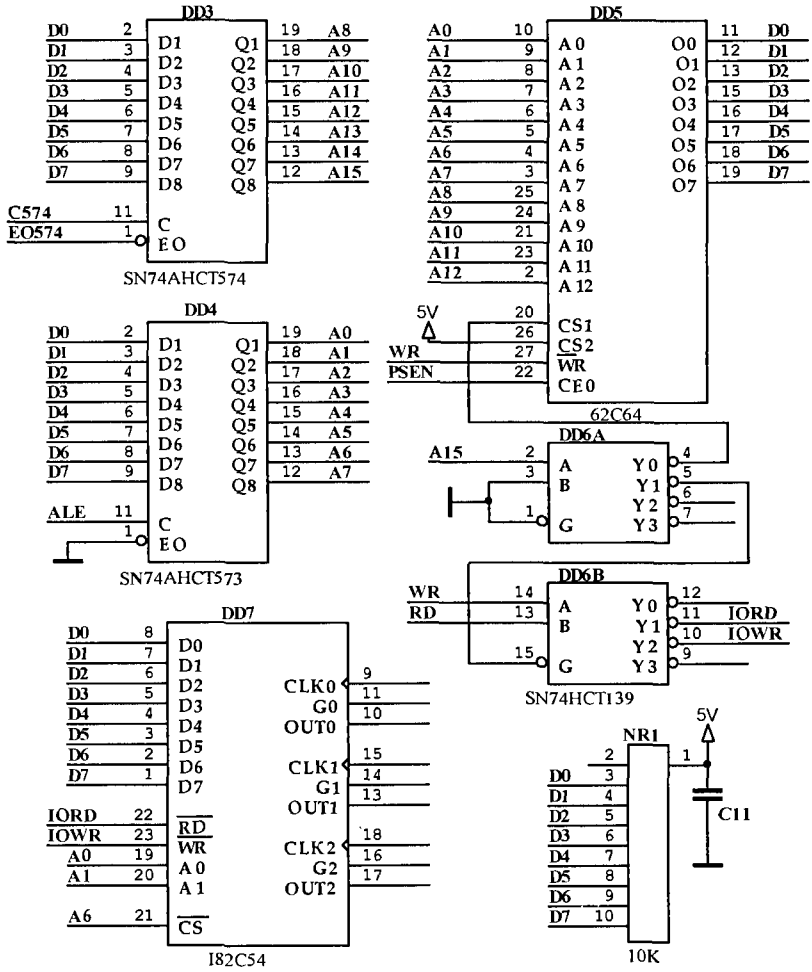


Рис. 7.1. Фрагмент принципиальной схемы загрузки внешней памяти программ микроконтроллера P80C552

порта компьютера. Сигнал с выхода TxD COM-порта компьютера подается на вход преобразователя уровня RS232 в уровень TTL (на схеме также не показан), а на выходе преобразователя образуется сигнал TxDC. На схеме показано, что сигнал TxDC подается на вывод RxD микроконтроллера, а сигнал с вывода TxD микроконтроллера подается на линию RxDC.

Все перечисленные микросхемы работают при функционировании *микроконтроллера P80C552 в штатном режиме при выполнении* программы.



В схеме присутствуют две дополнительные 20-выводные микросхемы, которые не функционируют при штатном режиме работы микроконтроллера P80C552. Это дополнительный регистр HCT574 (ИР37) – DD3 и микроконтроллер AT89C2051 – DD2. Они совместно с регистром HCT573 – DD4 осуществляют прием кодов программы с интерфейса RS232 и загрузку программы в память DD5. Необходимо отметить, что тактовая частота на микроконтроллер P80C552 подается с микроконтроллера AT89C2051 (сигнал CLK). Таким образом, в схеме используется единственный кварцевый резонатор Z1, подключенный к выводам XTAL1 и XTAL2 микроконтроллера AT89C2051. При частоте кварцевого резонатора 22118400 Гц обмен по RS232 может происходить со скоростью 115200 Гц (генератор скорости T1).

Схема работает следующим образом. Запуск схемы осуществляется компьютером по интерфейсу RS232 сигналом COM-порта DTR, который после преобразователя уровня RS232 в уровень TTL подается на вывод RESET микроконтроллера AT89C2051 и запускает его программу. Эта программа вначале запрещает работу микроконтроллера P80C552, выставляя на его выводе RST высокий уровень. Затем AT89C2051 принимает по RS232 коды загрузки памяти DD5. Приняв очередной байт, AT89C2051 вначале выставляет через свой порт p0 младший байт адреса памяти DD5 на шину данных D0-D7, защелкивает его в регистре DD4 сигналом ALE, затем аналогично выставляет старший байт адреса и защелкивает его в регистре HCT574 – DD3 сигналом C574. На вход EO этого же регистра AT89C2051 подает низкий уровень сигналом EO574 для разрешения выходов регистра, на которых появляется старший байт адреса. После этого AT89C2051 выставляет информационный байт опять на шине данных D0-D7. Таким образом, на шине адреса A0-A12 DD5 выставляется адрес памяти, а на шине данных D0-D7 – информационный байт. Затем AT89C2051 импульсом WR со своего порта p3.2 записывает этот информационный байт в память DD5. Когда все информационные байты приняты и записаны в память DD5, микроконтроллер AT89C2051 выставляет все свои сигналы на выводах, кроме вывода p3.3, в высокоимпедансное состояние, освобождая (от своих сигналов) необходимые связи для работы основного микроконтроллера P80C552, и заодно запрещает работу ненужного уже регистра DD3. *На вывод p3.3* AT89C2051, соединенный с выводом *RST* P80C552, подает низкий уровень, и основной микроконтроллер P80C552 начинает работу, выполняя программу, записанную в памяти DD5, с нулевого адреса.

После того как программа микроконтроллера отработала и компьютер получил от него все необходимые данные, сигналом DTR COM-порта компьютера, поданным на вывод RESET микроконтроллера AT89C2051 (высоким уровнем напряжения), работа

всей схемы завершается. Здесь необходимо сделать некоторые дополнения. Вышеописанный способ загрузки программы в микроконтроллер P80C552 по интерфейсу RS232 целесообразно применять в двух случаях. Во-первых, для предварительной загрузки и последующего запуска, которые выполняет основная программа компьютера, обслуживающая ту или иную компьютерную систему (см., например, приложение). Во-вторых, для разработки программы микроконтроллера. Разработка программы для микроконтроллера, как известно, требует проверки ее работоспособности в комплексе с компьютерной программой. Вышеописанный способ загрузки и запуска программы микроконтроллера позволяет загружать не только всю программу, но и ее части ("куски") и последовательно их проверять на работоспособность. Этот процесс достаточно быстрый и удобный.

Что касается загрузки и запуска уже разработанной программы для микроконтроллера, которая должна работать в комплексе с компьютерной программой, то каждый раз загружать программу для микроконтроллера по интерфейсу RS232 необязательно и даже нецелесообразно по двум причинам. Первая – это достаточно долгая процедура, вторая – очень слабая защита от несанкционированного доступа. Вместо загрузки программы в ОЗУ с помощью интерфейса RS232 автор может рекомендовать альтернативный способ загрузки ОЗУ – из вспомогательного микроконтроллера (в данном случае – из AT89C2051). Этот способ заключается в следующем.

Предположим, программа для микроконтроллера P80C552 занимает менее 2 кбайт (максимальный объем памяти AT89C2051). Тогда эту программу можно запрограммировать в AT89C2051, но не с нулевого адреса, а, например, с 100-го. В этом случае в адресное пространство, располагающееся с нулевого адреса по 100-й, можно запрограммировать *программу-загрузчик*, которая будет последовательно брать байты программы, располагающиеся с 100-го адреса (т.е. рабочую программу для P80552), и записывать их (байты) с помощью вышеописанной процедуры записи в ОЗУ. Время записи в ОЗУ одного байта – несколько микросекунд. Всю программу, таким образом, можно записать за несколько миллисекунд, что намного быстрее, чем передавать программу по интерфейсу и записывать в ОЗУ. От компьютера в этом случае требуется только небольшая задержка (после запуска AT89C2051), прежде чем начать работать с загруженной программой в штатном режиме. Кстати, AT89C2051 после успешной записи программы в ОЗУ может послать в компьютер какой-либо байт (с кодом, например, @), что будет означать успешную загрузку программы в ОЗУ и готовность системы к работе с компьютером. Компьютер не начнет работать с программой, пока не получит такой код.

Если программа для микроконтроллера P80C552 не укладывается в 2 кбайта, а занимает, например, 3,7 кбайт, то в качестве вспомогательного можно применить микроконтроллер AT89C4051 с максимальным объемом памяти программ 4 кбайт.

Автор неоднократно пользовался вышеописанной процедурой в своих разработках и может рекомендовать ее как исключительно быструю и надежную. В книге для этой процедуры программы не приводятся ввиду их примитивности.

7.2.2. Программное обеспечение

Программное обеспечение состоит из двух программ. Первая – программа для микроконтроллера AT89C2051 на ассемблере, вторая – программа для компьютера – на Бейсике. Обе программы приведены ниже (см. Программу 1 на с. 95–98).

7.3. Пример применения RS232 для программирования микроконтроллера AT89S8252 по интерфейсу SPI

7.3.1. Аппаратные средства

Принципиальная схема для этого примера уже рассматривалась, когда речь шла о гальванических развязках (см. гл. 4). Схема и необходимые пояснения приведены на рис. 4.9.

7.3.2. Программное обеспечение

Программа для компьютера на Бейсике и для микроконтроллера AT89C2051 на ассемблере приведены ниже (см. Программу 2 на с. 99–111); inline-подпрограммы prgrsis.asm и p86rd.asm уже приводились.

7.4. Пример программирования микроконтроллера DS5000(T) по RS232

7.4.1. Аппаратные средства

Аппаратные средства сопряжения микроконтроллера DS5000(T) с компьютером по интерфейсу RS232 уже обсуждались ранее, когда рассматривались нетрадиционные преобразователи уровней RS232 в уровни TTL и обратно. В гл. 4 (на рис. 4.2) показана схема программирования DS5000(T), использующая в качестве преобразователей уровней микросхему коммутатора КР1561КП5. Эта же микросхема применяется для подключения линий TxD и RxD микроконтроллера либо к интерфейсу компьютера, либо к схеме уст-

ройства, расположенного на плате. Управление этим переключением осуществляется сигналом интерфейса RTS.

7.4.2. Программное обеспечение

В гл. 4 при обсуждении протоколов обмена по интерфейсу RS232 было приведено несколько алгоритмов, которые применяются в самых различных ситуациях. Алгоритм обмена по RS232, использующийся фирмой Dallas Semiconductor для программирования микроконтроллера DS5000(T), достаточно простой, без синхронизации. Синхронизация используется как бы косвенно, так как в микроконтроллер передается программа в intel-hex-формате. Этот формат известен тем, что после передачи небольшого количества байт (не более 16) вслед за ними следует контрольная сумма. Если переданная контрольная сумма совпадает с подсчитанной микроконтроллером DS5000(T), то переданный пакет байт программируется в микроконтроллер. В противном случае выдается сообщение об ошибке, и переданная пачка байт теряется. Но сама процедура передачи одного байта в микроконтроллер по RS232 очень простая: перед передачей очередного байта компьютер проверяет готовность буфера к передаче очередного байта, а после передачи – окончания выхода байта из буфера (конец выхода последнего бита).

Программа программирования DS5000(T), написанная на Бейсике, приведена ниже (Программа 3 на с.112–115). Эта программа использует inline-подпрограмму в *.com-формате, написанную на ассемблере, для передачи строки байт. Алгоритм передачи байта в этой ассемблерной подпрограмме как раз и основан на вышеописанной процедуре передачи байта. Кроме того, эта подпрограмма не передает длину строки байт. Название ассемблерной подпрограммы prgrsds.asm, а ее текст приведен вслед за основной программой на Бейсике. Эта программа, как можно убедиться, достаточно простая. Для программирования используется команда загрузчика L, "обрамленная" двумя байтами возврата каретки (CR).

7.5. Пример применения RS232 для программирования микроконтроллеров ADUC8XX

7.5.1. Аппаратные средства

В гл. 4 на рис. 4.1 приведена схема сопряжения микроконтроллера ADUC8XX с компьютером по интерфейсу RS232. В схеме используется один традиционный преобразователь уровней интерфейса RS232 в уровни TTL и обратно – MAX3232 и один не-

традиционный – коммутатор DG419. Схема очень проста, стабильно работает при скорости обмена 115200 бод и в комментариях не нуждается.

7.5.2. Программное обеспечение

Программное обеспечение представлено двумя программами: на Бейсике с inline-подпрограммой на ассемблере и на Кларионе (см. Программу 4 на с.116–133).

Для микроконтроллеров, допускающих обмен по RS232 со скоростью 115200 бод (ADUC832, ADUC834 и ADUC836), можно вручную установить такую скорость обмена, убрав апостроф перед оператором goto PROG.

Программа на Кларионе *автоматически* устанавливает скорость обмена 115200 бод для соответствующего микроконтроллера по информации о его марке. Марка микроконтроллера передается в компьютер микроконтроллером.

Для запуска микроконтроллера может быть использована приведенная на с.126 программа run834.bas.

7.6. Пример применения RS232 для программирования микроконтроллеров MSC1210YX

7.6.1. Аппаратные средства

Схема сопряжения микроконтроллеров MSC1210YX с компьютером по интерфейсу RS232 аналогична схеме сопряжения для ADUC8XX. Исключением является подключение не часового кварцевого резонатора, а высокочастотного (например, с частотой 11059200 Гц). Кроме того, к кварцевому резонатору должны быть подключены два конденсатора емкостью 18 пФ каждый. Схема показана на рис.7.2.

7.6.2. Программное обеспечение

Бесплатно поставляемая фирмой Texas Instruments программа "download.exe" загрузки микроконтроллера MSC1210YX безукоризненно работает, если микроконтроллер подключен по схеме, показанной на рис.7.2. Вместе с этой программой приводится руководство пользователя. Для работы необходимо указать название программируемого файла в *.hex-формате, частоту кварцевого резонатора, номер COM-порта и опционально: скорость обмена, флаг "T" для закрытия окна после программирования. На рис.7.3 показан общий вид окна при работе с этой программой.

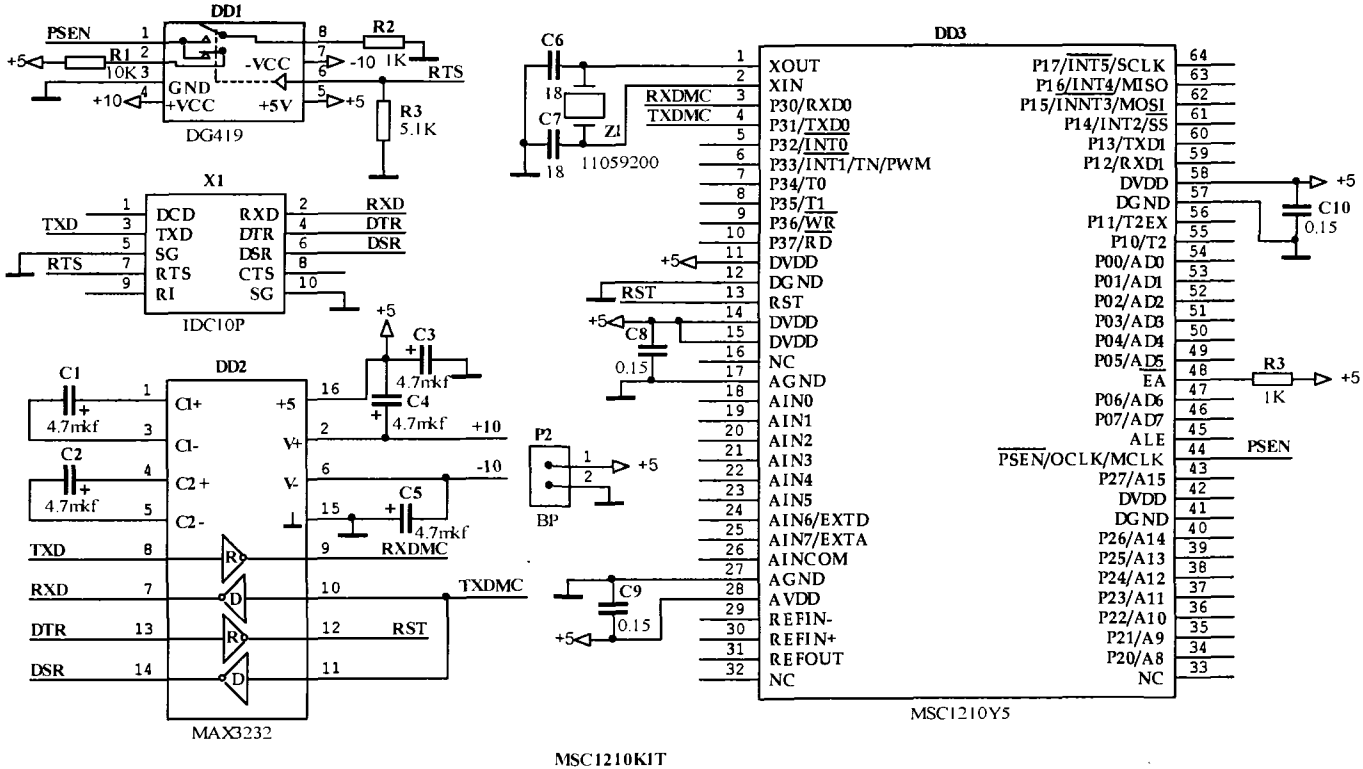


Рис. 7.2. Пример сопряжения микроконтроллера MSC1210YX с COM-портом компьютера по интерфейсу RS232 для программирования и запуска программы

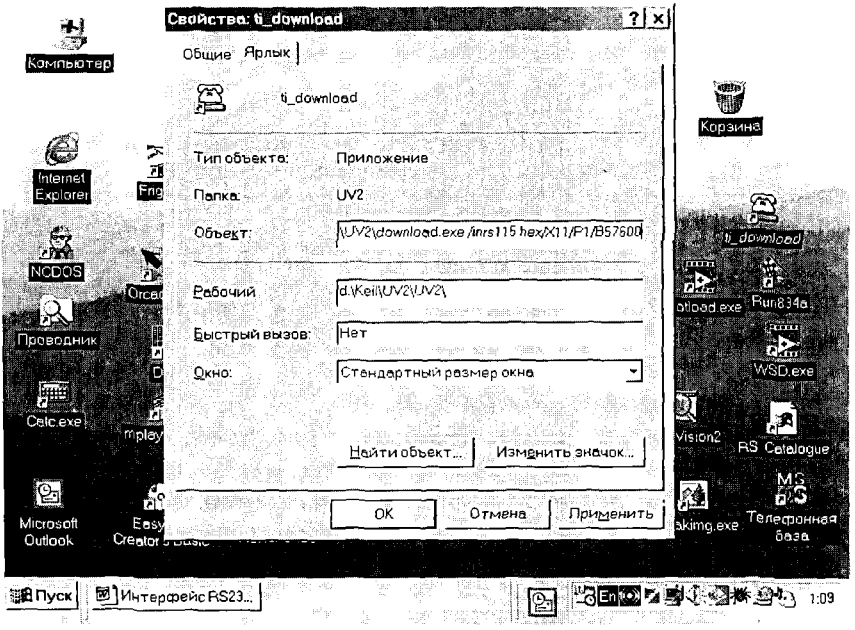


Рис. 7.3. Общий вид окна при работе с программой загрузки микроконтроллера MSC1210YX

Максимальная скорость обмена при использовании кварцевого резонатора с частотой 11059200 Гц, к сожалению, не превышает 57600 бод, да и на этой скорости программа загрузки работает не очень стабильно. Реальная скорость обмена при стабильной работе не превышает 38400 бод. Однако, по мнению автора, этого достаточно, чтобы с данным микроконтроллером можно было работать (и не писать свою программу загрузки).

Вышесказанное не означает, что микроконтроллер не будет работать с компьютером на скорости 115200 бод. *Он будет прекрасно работать на скорости 115200 бод*, если, например, в него загрузить и запустить программу, приведенную в гл. 6 (inrs115.asm) с там же рассмотренным протоколом обмена. А вот *загрузка программы* inrs115.hex с помощью программы download.exe *будет* происходить *со скоростью*, не превышающей *38400 бод*.

Программа 1

```
;--------
; Программа загрузки PCB80C552
; с помощью AT89C2051 по интерфейсу
; RS232 с постоянной скоростью.
;-------
;Макрос ввода байта
;-------
INBYTE .macro
       clr p3.1
       jnb ri,$
       setb p3.1
       mov a,sbuf
       clr ri
       .endm
;--------
;  Начало программы
;--------
; Инициал. посл. порта
;---------------
SPINIT: mov scon,#52h
        mov pcon,#80h
;---------------
TINIT: mov tmod,#20h
       mov th1,#255; - Скорость :250-9600;255-57600
                     ;при Fкв=11059200,
                     ;244-9600,255 -115200
                     ;при Fкв.=22118400.
;----------------------------------------
;               Выбор скорости:
;----------------------------------------
;Скор. | коэф.| При Fкварца=11059200 Гц.
;----------------------------------------
;57600 - 255:  255=256-1 =>57600/1 = 57600
;19200 - 253:  253=256-3 =>57600/3 = 19200
; 9600 - 250:  250=256-6 =>57600/6 =  9600
; 4800 - 244:  244=256-12=>57600/12 = 4800
```

```
; 2400 - 232:  232=256-24=>57600/24 = 2400
; 1200 - 208:  208=256-48=>57600/48 = 1200
;----------------------------------------
       setb tr1 ;Запуск таймера !!!
;---------------
       clr p3.4 ;  С ИР33
;---------------
;---------------
       clr p3.7 ;  ЕО ИР38
;---------------
; Ввод байт по посл. порту
;---------------
       mov dptr,#0000h; Нач. адрес
       INBYTE
       mov r0,a ; Мл. байт длины
       INBYTE
       mov r1,a ; Ст. байт длины + смещение
spin:  INBYTE   ; Байт в а
;----
; Запись байта в память
;----

       mov p1,dpl ;              Защелкивание
       setb p3.4  ; Строб на     младшего
       clr p3.4   ; С ИР33       байта адреса.

       mov p1,dph ;              Защелкивание
       clr p3.5   ; Строб на     Старшего
       setb p3.5  ; С ИР38       байта адреса.

       mov p1,a   ; Запись принятого
       clr p3.2   ; по RS232C байта в
       setb p3.2  ; память (импульс WR)
;----
;      Цикл по всему массиву.
;----
       inc dptr
       mov a,r1
       cjne a,dph,spin
```

```
        mov a,r0
        cjne a,dpl,spin
;------
; Заканчивание работы AT89C2051
;-------------------------------
        mov p3,#0ffh ;Перевод AT89C2051 |C ИР33 в 1
        mov p1,#0ffh ;в Z-состояние.    |EO ИР38 в 1.
        nop          ;На всякий случай.
;-------------------------------

        clr p3.3     ;Подача 0 на
                     ;RESET PCB80C552,
                     ;т.е. его запуск.
;-------------------------------
        jmp $        ;Остановка работы AT89C2051.
;-------------------------------
e:      end

-----------------------------------------------------------------------
'Программа на BASIC
out &h3fc,0':Установка DTR=-9вольт, RTS=-9вольт. Контрольный сброс
cls
sub REC inline
$inline "prgrsis.com"
end sub
'------
' Считывание файла с диска
'-------------
F$="inrs.tsk"
'-------------

'------
open "b",#1,F$
C$=space$(0)
L%=lof(1)
get$ #1,L%,C$
close #1
'-------
print"Начало программирования"
```

```
print "Программируется файл   ";F$
'------
' Инициализация последовательного порта
'------
' Установка скорости:
' 96- 1200,48- 2400,24- 4800,12 - 9600
'  6-19200, 3-38400, 2-57600, 1-115200
'------
out &h3fb,&h80
out &h3f8,2    '-Коэффициент скорости
out &h3f9,0    '-Старший байт делителя
'------
' Установка режима
'------
out &h3fb,3 '-1 стоп,8 бит,нет паритета
out &h3f9,0 '-Запрет всех прерываний по COM-порту
'--------------------
out &h3fc,0':Установка DTR=-9вольт, RTS=-9вольт. Контрольный сброс

delay .5
out &h3fc,1 'DTR=+9в,RTS=-9в'
delay .3
'------
' Передача длины файла и файла в ассемблер
'------
call REC(C$)
print "Конец программирования"
end
```

Программа 2

```
'------

F$="file.tsk"

'------

sub REC inline
$inline "prgrsis.com"
end sub

sub RED inline
$inline "p86rd.com"
end sub


'------
' Инициализация последовательного порта
'------
' Установка скорости:
' 96- 1200,48- 2400,24- 4800,12 - 9600

'  6-19200, 3-38400, 2-57600, 1-115200
'------
out &h3fb,&h80
out &h3f8,1    '-Коэффициент скорости
out &h3f9,0    '-Старший байт делителя
'------
' Установка режима
'------
out &h3fb,7 '2 стопа,8 бит,нет паритета
out &h3f9,0 'Запрет всех прерываний по COM-порту
'------
out &h3fc,2 'Сброс микроконтроллера AT89C2051.
delay .1
out &h3fc,3 'Запуск микроконтроллера AT89C2051.
delay .1
```

```
          if (inp(&h3fe) and &h10)=&h10 then goto REGIM
          goto WYKL

REGIM:
'-----------------------------------------------
' Определение режима работы программатора
'-----------------------------------------------
cls
goto PROG
locate 4,15
print "Введите код режима работы программатора:"
locate 10,10
print " 1 - программирование в AT89S8252 файла file.tsk"
locate 12,10
print " 2 - чтение AT89C8252 и запись в файл filea.bas"
locate 18,10
input "                ";A$
   if A$="1" then goto PROG
   if A$="2" then goto RED
cls
PROG:
'-----------------------------------------------
' П/п программирования AT89S8252
'-----------------------------------------------
cls
'delay .2
'---------
        BYTE%=&h77    'Вывод буквы w
        gosub OUTBYTE
        print chr$(BYTE%)
'-------
PRODR0:
locate 4,22
print "Программирование AT89S8252";

locate 5,22
print "--------------------------"
locate 7,12
```

```
print "Считывание файла file.tsk для AT89S8252 с диска...";
'**************************************
open "b",#1,F$
L=lof(1)
L%=L
C$=space$(0)
get$ #1,L%,C$
close #1
'**************************************

locate 12,15
print "Начинать программировать? OK? (ENTER)"
locate 13,15
print "----------------------------------------"

while not instat:wend
r$=inkey$
cls
locate 7,15
print "Программирование в AT89S8252 файла ";F$
locate 10,15
print "           Ждите... "

'----------------------------
' Программирование.
'----------------------------
' Передача длины файла и файла в ассемблер
'------
call REC(C$)
locate 10,25
print "Конец программирования"
delay .1
a=inp(&h3f8)
delay .2
'cls
'--------------
gosub INBYTE
'locate 12,25
```

```
print chr$(BYTE%)
  if byte%=&h4f then print "OK":goto E
  if byte%=&h45 then goto ERROR0
     print "Не работает":goto E
ERROR0:
print "Ошибка"
      goto E
'-----------------------------------
'Конец п/п программирования.
'-----------------------------------


RED:
'-----------------------------------
' П/п чтения.
'-----------------------------------
cls
'---------
        BYTE%=&h72  'Вывод буквы r
        gosub OUTBYTE
        print chr$(BYTE%)

'-------
PRODR1:
        delay .5
        a=inp(&h3f8)' Холостой ввод
'-------
locate 2,7
print "Чтение AT89S8252 и запись его содержимого на диск в filea.bas"
locate 3,7
print "------------------------------------------------------------"
locate 7,20
print "Начинать чтение? OK? (ENTER)"
while not instat:wend
a$=inkey$
gosub INBYTE
print chr$(BYTE%)
```

```
C$=space$(&h2800)
locate 7,15
print "                Идет считывание!              "

CALL RED(C$)

'--------------
locate 7,15
print "              Идет запись на диск!            "
open "b",#1,"filea.bas"
if lof(1)<>0 then kill "filea.bas"
close
open "b",#1,"filea.bas"
put$ #1,C$
close

C$=space$(0)
locate 10,15
print "                  Все записалось              "

'--------------
gosub INBYTE
print chr$(BYTE%)
  if BYTE%=&h4f then locate 8,10:print "OK!":goto E
print "Ошибка";
goto E
'------------------------------
' Конец п/п чтения.
'------------------------------

'------------------------------
' Подпрограммы
'------------------------------
INBYTE:
        out &h3fb,&H47       'Установка линии TxD - разрешение передачи
        wait &h3fe,&h20      'Ожидание старт-бита(установки DSR)
        out &h3fb,&h07       'Сброс линии TxD - запрет передачи
      wait &H3fd,1           'Ожидание конца прихода байта
```

```
      BYTE%=inp(&h3f8)       'Ввод байта
return
'------------------------------
OUTBYTE:
       wait &h3fd,&h20        'Ожидание готовности передатчика (в машине).
       wait &h3fe,&h20        'Ожидание готовности линии (установки DSR).
       out  &h3f8,BYTE%       'Вывод байта.
       wait &h3fe,&h20,&h20   'Ожидание сброса готовности линии (сброса DSR).
       wait &h3fd,&h40        'Transmitter holding registr is empty.
return
'------------------------------
WYKL:
cls
locate 10,25
print " Система выключена"
locate 15,22
print " Нажми ENTER - для выхода"
print""
print""
       while not instat
       wend
       a$=inkey$
       if a$=chr$(13) then cls: goto E
       goto WYKL
E:
out &h3fc,0':Установка DTR=-9вольт, RTS=-9вольт. Контрольный сброс
delay .5
locate 15,20
print "Выньте микросхему и нажмите любую клавишу"
while not instat:wend
r$=inkey$
       cls
end

;-----------------------------------
; Программа программатора AT89S8252
;-----------------------------
```

```
;----------------------------
;  Установка названий бит
;------------------------
RST:     .reg p1.7
SCK:     .reg p1.4
MOSI:    .reg p1.3
MISO:    .reg p1.2
ON:      .reg p3.5
;-------------------------
;    Макросы
;-------------------------------------------------------------------------
;  В МАКРОСАХ НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ РУССКИЙ ЯЗЫК В КОММЕНТАРИЯХ !!!!!!!
;-------------------------------------------------------------------------
; Алгоритм ввода байта по RS232
;-----------------------
INBYTE   .macro
         clr TxD        ;Enable transfer
         jb RxD,$       ;Waiting start-bit
         setb TxD       ;Disable transfer
         jnb ri,$       ;Waiting set flag ri
         mov a,sbuf     ;Input byte from buffer
         clr ri         ;Clear flag ri
        .endm
;-----------------------------
;   Алгоритм вывода байта по RS232
;-----------------------------
OUTBYTE .macro
         jb RxD,$       ;Waiting enable transfer
         mov sbuf,a     ;Output byte in buffer
         jnb RxD,$      ;Waiting disable transfer.WARNING!!!
         jnb ti,$       ;Waiting output last bit (flag ti)
         clr ti         ;Clear flag ti
        .endm
;-----------------------------------------
;   Алгоритм вывода байта в AT89S8252 по SPI
;-----------------------------------------
OUTBS    .macro
         mov b,#8
```

```
METO#:   rlc a
         mov MOSI,c
         nop
         nop
         nop
         setb SCK
         nop
         nop
         nop
         nop
         clr SCK
         djnz b,METO#
        .endm
;----------------------------------------
;   Алгоритм ввода байта из AT89S8252 по SPI
;----------------------------------------
INBS     .macro
         mov b,#8
METI#:   setb SCK
         nop
         nop
         nop
         mov c,MISO
         rlc a
         clr SCK
         nop
         nop
         djnz b,METI#
        .endm
;--------------------------------
          .DATA
;----------------------------
inpb      equ 10h
progb     equ 11h
n         equ 12h
MEMAX     equ 13h ;Ст.  байт  максимального
                  ;объема памяти микросхемы.
;--------------------------
```

```
          .CODE
;-----------------------------------
;       Начало кода
;-----------------------------------

        org $
        mov sp,#20h
;-------------------------------------------------------------------------------
       clr ON  ;ON
       clr SCK
       clr RST
       call DEL2_5MS
;---------------
; Инициал. посл. порта
;---------------
                    ;  jmp INIT9600
;-----------------------------------
; Инициал. посл. порта на 115200 бод.
;-------------------------------------------------------------------------
INIT115: mov pcon,#0  ;Частота приема/передачи НЕ УДВАИВАЕТСЯ,
         mov scon,#8ch;8 бит,2 стопа,REN=0(запрет приема ReciveENable=0),
                     ;TI=0,RI=0,Vобм=Fкв/64=7372800/64=115200 бод.,
                     ;SM0=1,SM1=0,SM2=0(#8ch) - режим 2. RB8=TB8=1.
;-----------------------
;Инициализация таймеров T0 и T1: T0 - как счетчика, T1 - не используется !
;-------------------------------------------------------------------------
;                          Регистр TMOD:
;T1: GATE1=0-упр.от процессора;C/T1=0-таймер;M1.1=0,M0.1=0-реж.0 =>0000b
;T0: GATE0=1-управл. от INT0; C/T0=1-счетчик;M1.0=0,M1.1=1-реж.1 =>1101b
;    TMOD=00001101b или 0dh
;-------------------------------------------------------------------------
             mov tmod,#0dh
             jmp START
;-----------------------------------
; Конец инициализации на 115200 бод.
;-----------------------------------
```

```
;-------------------------------------------------------
; Инициал. посл. порта на скорость 9600 бод.
;-------------------------------------------------------
INIT9600:   mov pcon,#80h ;Частота приема/передачи УДВАИВАЕТСЯ,
            mov scon,#cch ;4ch-работает!;52h ;8 бит.,1 стоп,REN=0(Запрет приема:RENable=0),
                          ;TI=0,RI=0, Vобм. определяется T1,
                          ;SM0=0,SM1=1,SM2=0(#4ch) - режим 1. TB8=RB8=1.
                          ;SM0=1,SM1=1;SM2=0(#cch) - режим 3. TB8=RB8=1.
;-------------------------------------------------------------------
;Инициализация 2-х таймеров: T1-для генерации Vобм. по RS232C и T0-как счетчика.
;     Регистр TMOD:
;T1: GATE1=0-упр.от процессора;C/T1=0-таймер;M1.1=1,M0.1=0-реж.2 =>0010b
;T0: GATE0=1-управл. от INT0; C/T0=1-счетчик;M1.0=0,M1.1=1-реж.1 =>1101b
;    TMOD=00101101b или 2dh
;---------------------------
           mov tmod,#2dh
           mov th1,#252     ;9600 бод при fкв.=7372800.
;-------------------------------------------------------------------------
;255-38400,254-19200,252-9600 бод.....................(fкв.= 7372800 Гц.)
;-------------------------------------------------------------------------
;250-9600,253-19200,255-57600 бод.....................(fкв.=11059200 Гц.)
;-------------------------------------------------------------------------
;244-9600,250-19200,253-38400,254-57600,255-115200 бод...(fкв.=22118400 Гц.)
;-------------------------------------------------------------------------
           setb tr1      ;Запуск T1 (T0 пока не запущен)
;---------------------
; Конец инициализации на 9600 бод.
;-----------------------------
START:
;-----------------------------
; Programming Enable
;-----------------------------
            setb RST
            call DEL2_5MS
            mov a,#10101100b
            OUTBS
            mov a,#01010011b
            OUTBS
```

```
            mov a,#ffh
            OUTBS
;-----------------------------
; Chip Erase
;-----------------------------
            mov a,#10101100b
            OUTBS
            mov a,#00000100b
            OUTBS
            OUTBS
            call DEL2_5MS
            call DEL2_5MS
            call DEL2_5MS
            call DEL2_5MS  ;Задержка 17.5ms (>16мс).
            call DEL2_5MS
            call DEL2_5MS
            call DEL2_5MS
;----------------------------
;Ввод байт по посл. порту
;----------------------------
            setb REN    ;Разрешение приема.
            INBYTE      ;Ввод буквы "w"
            cjne a,#77h,ER0
            jmp DALEE
ER0:        jmp ER
;-----------------------------
DALEE:      mov dptr,#0
            INBYTE
            mov r0,a ; Мл. байт длины
            INBYTE
            mov r1,a ; Ст. байт длины

SPIN:       INBYTE; Байт в a
;----
            mov inpb,a
            mov a,dph
            orl a,#01000000b
            rl a
```

```
                rl a
                rl a
                OUTBS       ;output high addr for programming
                mov a,dpl
                OUTBS       ;output low addr for programming
                mov a,inpb  ;data
                OUTBS       ;output data for programming
                call DEL2_5MS
                mov a,dph
                orl a,#00100000b
                rl a
                rl a
                rl a
                OUTBS       ;output high addr for reading
                mov a,dpl
                OUTBS       ;output low addr for reading
                INBS        ;reading data
                cjne a,inpb,ER

;----
                inc dptr
                mov a,r1
                cjne a,dph,SPIN0
                mov a,r0
                cjne a,dpl,SPIN0
                jmp OK
SPIN0:          jmp SPIN
;-------------------------------------
; Вывод байт по посл. порту
;-------------------------------------
OK:           clr REN           ;Запрет приема.
              mov a,#4fh        ; "O"
              OUTBYTE
              jmp E
ER:           clr REN
              mov a,#45h        ; "E"
              OUTBYTE
              jmp E
```

```
;-------------------------------
; Ввод команды продолжения работы
;-------------------------------
;           setb REN          ;Разрешение приема.
;IB:        INBYTE
;           cjne a,#40h,E
;           jmp START
;---------------------------
E:          setb SCK
            setb MOSI
            setb MISO
            setb ON
            jmp $
;-----------------------------------------------------
DEL2_5MS:  mov a,#20h ; Задержка 2508.1 мкс (2.5 мс)
MET:       mov b,#16h ; вместе с "call"
           djnz b,$   ; и "ret" при
           djnz a,MET ; Fкварца=7372800Гц.
           ret
;------------
           .end
```

Программа 3

```
'-------------------------------------------------------------
out &h3fc,0':Установка DTR=-9вольт, RTS=-9вольт. Контрольный сброс
cls
sub REC inline
$inline "prgrsds.com"
end sub
'------
' Считывание файла с диска
'------

F$="inrs.hex"
'-------------
open "b",#1,F$
C$=space$(0)
L%=lof(1)
get$ #1,L%,C$
close #1
'-------
print"Начало программирования"
print "Программируется файл   ";F$
'------
' Инициализация последовательного порта
'------
' Установка скорости:
' 96- 1200,48- 2400,24- 4800,12 - 9600
'  6-19200, 3-38400, 2-57600, 1-115200
'------
out &h3fb,&h80
out &h3f8,2    '-Коэффициент скорости
out &h3f9,0    '-Старший байт делителя
'------
' Установка режима
'------
out &h3fb,3 '-1 стоп,8 бит,нет паритета
out &h3f9,0 '-Запрет всех прерываний по COM-порту
'---------------------
```

```
out &h3fc,0':Установка DTR=-9вольт, RTS=-9вольт. Сброс и программир.
delay .3
byte%=&h0D      '"CR"
gosub OUTBYTE
delay .3
byte%=&h4C      '"L"
gosub OUTBYTE
byte%=&h0D      '"CR"
gosub OUTBYTE
delay .1
'------
' Передача длины файла и файла в ассемблер
'------
call REC(C$)
print "Конец программирования"
out &h3fc,1':Установка DTR=+9вольт, RTS=-9вольт. Запуск соединение с комп.
'out &h3fc,3'Установка DTR=+9вольт, RTS=+9вольт. Запуск и разъединение с комп.

        'out &h3fc,2':Установка DTR=-9вольт, RTS=+9вольт.No connect
delay .3
goto E
'-------------
'-----
' Подпрограмма вывода байта
'-----
OUTBYTE:
      wait &h3fd,&h40   ' Проверка готовности передатчика ( в машине )
      out  &h3f8,byte%  ' Вывод байта
      wait &h3fd,&h20   ' Проверка OK to send ( в машине )
return
E:
end
```

```
;-------------------------------------------------
; prgrsds.asm
;-------------------------------------------------
OUTBYTE macro ; Байт в bl
local  MET1,MET3
       mov dx,3fdh
MET1:  in al,dx
       test al,20h ; Проверка вх. буфера в машине(transmitter empty)
       jz MET1

       mov al,bl
       mov dx,3f8h
       out dx,al   ; Вывод байта
;----------
       mov dx,3fdh
MET3:  in al,dx
       test al,40h ; Transm. holding reg. empty. OK to send.
       jz MET3
;----------
endm

PROGRAM segment
        org 100h
assume  cs:PROGRAM,ds:PROGRAM,es:PROGRAM,ss:PROGRAM
START:  push bp
        mov bp,sp
        push es
        push ds
;-----
        les di,[bp+6]
        mov dx,ds:[0]
        mov ds,dx
        mov si,es:[di+2]
        mov cx,es:[di]
        and cx,7fffh ; Длина в cx
```

```
;-----
; Передача файла по послед. порту в BE51
;-----
       mov bl,cl
;       OUTBYTE
       mov bl,ch
;       OUTBYTE
MET:   mov bl,byte ptr [si]
       OUTBYTE; Байт в bl
       inc si
       loop MET
;-----
       pop ds
       pop es
       pop bp
PROGRAM ends
end START
```

Программа 4

```
'-------------------------
' Inline-подпрограммы
'-------------------------------
' П/п передачи файла из компьютера
' в программатор по COM1 без передачи ее длины.
'-------------------------------
sub REC0 inline
$inline "prgrsds.com"
end sub
'-------------------------
'  Инициализация порта
'-------------------------
cls
        'Установка скорости 9600*12 бод

   OUT &H3FB,&H80
   OUT &H3F8,12' 1-115200 ,2-57600, 3-38400, 4-28800, 6-19200, 12-9600.
   OUT &H3F9,0
'---------------------------
'  Установка режима

   OUT &H3FB,&H03 '1 стоп-бит,8бит,нет паритета
        out &h3f9,0     'Запрет всех прерываний по порту 3f8h
'-------------------------------

'-------------------------------
'Инициализация микроконтроллера
'-------------------------------
        out &h3fc,0'Установка RTS=-9в, DTR=-9в.
        'print "3fc=0: PSEN=1,RST=1 - Общий сброс"
                                'xxxx xx00
                                '  RTS--^^--DTR
        delay .3
   A%=INP(&H3F8) 'Холостой ввод
        out &h3fc,2    'Установка RTS=9в, DTR=-9в.
        'print "3fc=2: PSEN=0,RST=1 - Условия для загрузки"
```

```
        delay .3
        'print "3fc=3: RST=0,PSEN=0 - Сиотри PSEN=0 !!!Загрузка"
'-------------------------------------
print""
print"   ████████████████████████████████████████"
print"   █ ";
'--------------------------------------
    out &h3fc,3   'Установка RTS=9в,DTR=9в.
'--------------------------
'Прием информации из ADUC
'--------------------------


'---------------------------
'10 bytes - Product identifier
'---------------------------
for c=1 to 10
        gosub INBYTE
        print chr$(BYTE%);
next c
'-------------------------------
'4 bytes - firmware version number
'-------------------------------
for c=1 to 4
        gosub INBYTE
        print chr$(BYTE%);
next c
'------------------------------
print " Ид.N=";
'------------------------------
'2 bytes - LF and CR
'-----------------
for c=1 to 2
        gosub INBYTE

        if (BYTE%=&h0d) or (BYTE%=&h0a) then goto 111
        print hex$(BYTE%);
111:
next c
```

```
'-------------------------------
'2 bytes - hardware configuration
'-------------------------------
for c=1 to 2
        gosub INBYTE
        print hex$(BYTE%);
next c
'------------------
'6 bytes - reserved
'-----------------
for c=1 to 6
        gosub INBYTE
        print hex$(BYTE%);
next c
'------------------------
print" K.c.=";
'-----------------
' 1 byte - checksum
'------------------
        gosub INBYTE
        print hex$(BYTE%);"h";
'-------------------------------
        print " █"
               'print "3fc=3 PSEN=0,RST=0,Загрузка"
print"   ████████████████████████████████████"
print""
'-----------------------------
'  Стирание памяти
'-----------------------------
print "====================="
print "Стирание памяти: ";

gosub ER

call REC0(C$)      'Вывели пакет в ADUC
gosub INBYTE       'Ввод ответа ACK или NAK
 if BYTE%=6 then print "█";:goto PRO
 if BYTE%=7 then print "Ошибка"
```

```
 goto KON
PRO:
print " ERASE Ok."
print "================================"

'  goto PROG

'-----------------------------
'  Установка скорости 115 кбод.
'-----------------------------
print "======================="
print "Скорость 115 кбод: ";

gosub FAST

call REC0(C$)      'Вывели пакет в ADUC
gosub INBYTE       'Ввод ответа ACK или NAK
 if BYTE%=6 then print "■";:goto PRO1
 if BYTE%=7 then print "Ошибка"
 goto KON
PRO1:
print " FAST Ok."
print "================================"

'input a
'---------------------------------------
        'Установка скорости 115200 бод

      OUT &H3FB,&H80
      OUT &H3F8,1' 1-115200,12-9600.
      OUT &H3F9,0
'----------------------------
'     Установка режима
      OUT &H3FB,&H03 '1 стоп-бит,8бит,нет паритета
      out &h3f9,0    'Запрет всех прерываний по порту 3f8h
'---------------------------------
```

```
PROG:

print "Чтение файла *.tsk с диска: ";
'---------------------------------
F$="file.tsk"
'---------------------------------
open "b",#1,F$
print "Ok."
print "================================"

print "Программирование: "
print "=================="
'-----------------------------
L=lof(1)
if L<=16 then L%=L:C$=space$(0):get$ #1,L%,C$:ADR%=0:goto PROGET
N=int(L/16)
lostat%=L-N*16
ADR%=0
L%=16
    for i=1 to N
C$=space$(0)
get$ #1,L%,C$
gosub FORMS         'Получили пакет C$
call REC0(C$)       'Вывели пакет в ADUC
gosub INBYTE        'Ввод ответа ACK или NAK
 if BYTE%=6 then print "■";:goto PRODOL
 if BYTE%=7 then print "Ошибка"
 goto KON
PRODOL:
ADR%=ADR%+16
    next i

C$=space$(0)
L%=lostat%   'Длина остатка
get$ #1,L%,C$

PROGET:
gosub FORMS    'Получили пакет C$
call REC0(C$) 'Вывели пакет в ADUC
```

```
gosub INBYTE  'Ввод ответа ACK или NAK
 if BYTE%=6 then print "■";:goto PRODOL1
 if BYTE%=7 then print "Ошибка"
 goto KON
PRODOL1:
close #1
print " Ok."
goto KON
'**************************************
'          Подпрограммы
'**************************************

'-----------------------------------,
'  П/программа формирования         |
'  пакета посылки. Принимает:       |
'  L%   - длину строки данных,      |
'  ADR% - адрес, с которого загружать,|
'  C$   - строку загружаемых данных.  |
'-----------------------------------'
FORMS:
'----------------------
' Стартовая строка.
'----------------------
START$=mki$(&h0e07)
'------------------

'---------------------------------------
' Строка количества байт (L%-принимается).
'---------------------------------------
NB$=LEFT$((mki$(L%+4)),1)
'----------------------

'-----------------------------------
' Строка команды записи ('W'код=57h) +
' + старший адрес ADR_U (код=0).
'-----------------------------------
CMDADRU$=mki$(&h0057)
'----------------------
```

```
'--------------------------
' Строка адреса (ADR%-принимается).
'-------------------------------------------------
ADR$=RIGHT$((mki$(ADR%)),1)+LEFT$((mki$(ADR%)),1)
'-------------------------------------------------

'-------------------------------------
'Строка данных C$. (C$ - принимается).
'-------------------------------------

'------------------------------
'Строка контрольной суммы
'------------------------------
'  1b +   2b   + 2b +L% = L% + 5 байт - общее к-во байт в к.с.
'   |     |       |   |
S$=NB$+CMDADRU$+ADR$+C$ ' - Строка всех байт для вычисления к.с.
'-------------------------------------------------

'------------------
'  Вычисление к.с.
'------------------
S=0
NS%=L%+5
 for j=1 to NS%
S=S+asc(MID$(S$,j,1))       'Сумма байт
 next j
KS%=(256*ceil(S/256))-S     ' - Значение к.с. в цифровом виде.
KS$=LEFT$((mki$(KS%)),1)    ' - Строка к.с. (1 байт).
'-------------------------

'---------------------------
' Формирование пакета посылки
'---------------------------
C$=START$+NB$+CMDADRU$+ADR$+C$+KS$
'-----------------------------------
return
'-----------------------------------
```

```
'----------------------------
' П/п формирования строки ERASE
'------------------------------
ER:
'-------------------------------
' Стартовая строка.
'-------------------------------
START$=mki$(&h0e07)
'------------------

'-----------------------------------------------------
' Строка команды ERASE + 1 байт - количество байт
'      ERASE=43h - стирание памяти программ, к.с.=bch
'      ERASE=41h - стирание памяти программ и данных, к.с.=beh
'-----------------------------------------------------------
CMDERASE$=mki$(&h4101)
'---------------------------

'---------------------------
' Строка контрольной суммы beh
'-------------------------------
KSER$=LEFT$((mki$(&hbe)),1)
'-------------------------------

'--------------------------
' Полная строка C$
'-----------------------------
     'C$=START$+CMDERASE$+KSER$
C$=mki$(&h0e07)+mki$(&h4101)+LEFT$((mki$(&hbe)),1)
'-----------------------------------------------

return
'--------------------

'---------------------------
' П/п установки скорости 115 кбод.
'-----------------------------
FAST:
```

```
'-------------------------
' Полная строка C$
'-----------------------------
C$=mki$(&h0e07)+mki$(&h4203)+mki$(&h2d81)+left$((mki$(&h0d)),1)
'-----------------------------------------------
return

'--------------------
' П/программа ввода байта
'----------------------
INBYTE:
      WAIT &H3FD,1     ' Ожидание факта прихода байта
      BYTE%=INP(&H3F8)' Ввод очередного байта
return
'-------------------------

KON:
'---------------------------
'  Сброс микроконтроллера
'---------------------------
out &h3fc,0'Установка DTR=-9в,RTS=-9в.'Полный сброс.
'---------------------------
        print "_____________________"
        print "Конец программирования."
        delay .2
END

'for j=1 to 5
'print "    ";j;"h  ";hex$(asc(MID$(C$,j,1)));"h";
'print hex$(asc(MID$(C$,j,1)));" ";
'next j


'for j=1 to 24
'print hex$(asc(MID$(C$,j,1)));" ";
'next j

;-----------
```

```
;  prgrsds.asm
;-----------
OUTBYTE macro ; Байт в bl
local  MET1,MET3
       mov dx,3fdh
MET1:  in al,dx
       test al,20h ; Проверка вх. буфера в машине(transmitter empty)
       jz MET1

       mov al,bl
       mov dx,3f8h
       out dx,al   ; Вывод байта
;----------
       mov dx,3fdh
MET3:  in al,dx
       test al,40h ; Transm. holding reg. empty. OK to send.
       jz MET3
;----------
endm
PROGRAM segment
        org 100h
assume  cs:PROGRAM,ds:PROGRAM,es:PROGRAM,ss:PROGRAM
START:  push bp
        mov bp,sp
        push es.
        push ds
;-----
        les di,[bp+6]
        mov dx,ds:[0]
        mov ds,dx
        mov si,es:[di+2]
        mov cx,es:[di]
        and cx,7fffh ; Длина в cx
;-----
; Передача файла по послед. порту в BE51
;-----
       mov bl,cl
;       OUTBYTE
```

```
        mov bl,ch
;        OUTBYTE
MET:    mov bl,byte ptr [si]
        OUTBYTE; Байт в bl
        inc si
        loop MET
;-----
        pop ds
        pop es
        pop bp
PROGRAM ends
end START
;-----------

'run834.bas
'------------
        cls
'--------------------------
'       ИНИЦИАЛИЗАЦИЯ RS232C
'--------------------------
'     Установка скорости
      OUT &H3FB,&H80
      OUT &H3F8,1' Мл.б.=:1-115200,2-57600,12-9600 бод.
      OUT &H3F9,0' Ст.б.=0
'---------------------------
'     Установка режима
      OUT &H3FB,&H07 '1 стоп,8бит,нет паритета
'-------------------------------------------------
'       Установка запрета прерываний по COM-порту
        out &h3f9,0 ' Прерывания запрещены
'-------------------------------------------------
'       Инициализация микроконтроллера
'-------------------------------------------------
        OUT &H3FB,&H47     ' Установка линии TxD - разрешение передачи
        out &h3f8,&h0d

        out &h3fc,0' Установка DTR=-9в., RTS=-9в. Сброс микроконтроллера.
        delay .2
```

```
        print "3fc=0"
        input a
        out &h3fc,1' Установка DTR=9в., RTS=-9в. Запуск микроконтроллера.
'        OUT &H3FB,&H47     ' Установка линии TxD - разрешение передачи
        print"3fc=1, Запуск"
        delay .2
        input a
'--------------------------
inl:
     '   input a
E:
       out &h3fc,0' Установка DTR=-9в., RTS=-9в.Сброс микроконтроллера.
       print"Конец"
       delay .2
end

!-----------------------------------------------------
! down834.cla
!-----------------------------------------------------
! Программа программирования файла file.tsk в ADUC8XX.
!-----------------------------------------------------

                    PROGRAM

!-----------------------------------------------
! Определение переменных
!-----------------------------------------------
CER  string('<07h,0eh,01h,41h,beh>') !Строка с командой ERASE.
MER          byte,dim(5),over(CER)   !Массив для передачи ERASE.
C115 string('<07h,0eh,03h,42h,81h,2dh,0dh>') !Строка с командой 115 кбод.
M115         byte,dim(7),over(C115)  !Массив для передачи 115 кбод.
M1           byte,dim(25)            !Массив принимаемых байт.
S1           string(14),over(M1)     !Строка идентификации ADUC.
SN           string(3)
LEND         byte                    !Длина строки данных.
NB           byte                    !Количество байт в массиве вывода.
L            ushort                  !Длина файла в байтах.
LOST         byte                    !Длина остатка L%16 в байтах.
D            byte,dim(16)            !Массив данных для программирования.
```

```
S0              string(16),over(D)      !Строка данных для программирования.
SD              ushort                  !Сумма данных.
ADR             ushort                  !Двухбайтный адрес.

ADRMAX          ushort                  !Максимальный адрес.
M               byte,dim(25)            !Массив для вывода OUTBYTE.
MKS             byte,dim(21)            !Массив для расчета контрольной суммы.
SUM             ushort                  !Сумма байт данных.
KSUM            byte                    !Контрольная сумма.
N               ushort
k               ushort
B               byte
i               ushort
Y00h            byte(00h)
Y01h            byte(01h)
Y02h            byte(02h)
Y03h            byte(03h)
Y04h            byte(04h)
Y06h            byte(06h)
Y07h            byte(07h)
Y12             byte(12)
Y47h            byte(07h)
Y80h            byte(80h)
Y0fh            byte(0fh)

BINARY                  FILE,PRE(BIN),DRIVER('DOS'),NAME('file.tsk')
RECORD                  RECORD
STR                     string(65000)  !Строка с прочитанным файлом
        ..
!-----------------------------------------------
                        CODE
 blank
 show(20,10,'Вставьте микросхему !!!')
 show(21,10,'Для выхода нажмите Ctrl+Break')
!----------------------------
! Прием информации из ADUC
!----------------------------
        do INIT !Инициализация RS232 и микроконтроллера.
```

```
        loop i=1 to 25
        do INBYTE
        .
        out(3fch,Y01h)
        blank
   !    show(2,1,S1) !Вывод на экран принятой информации.

        SN=sub(S1,5,3)

!--------------------------------
!--------------------------------
! blank
  if (SN='836') or (SN='834') or (SN='832') then goto V115KB.
  if (SN='816') or (SN='824') then goto SH.
  goto M

V115KB
!-------------------------------------------------------------
!   Установка скорости 115 кбод.
!---------------------------------
        loop i=1 to 7
         M[i]=M115[i]
         do OUTBYTE     !Вывод 7-ми байт команды 115 kbod.
        .
        i=1
        do INBYTE       !Прием ответа ACK или NAC от ADUC.
        if (M1[1]=7) OR (M1[1]<>6)
         type('Ошибка...');goto M
        .
!-------------------------------------------------------

!-------------------------
! Инициализация COM-порта на 115 кбод.
!-------------------------
    out(3fbh,Y80h) ! DLAB=1 для установки делителя.
    out(3f8h,Y01h) ! Установить мл.б. скор.:Y01h-115200 бод,Y12-9600 бод.
    out(3f9h,Y00h) ! Установить ст.б. скор.=0.
    out(3fbh,Y03h) ! DLAB=0,Режим: 8 бит данных,1 стоп,нет пар.,сброс TxD.
    out(3f9h,Y00h) ! Запрет всех прерываний по порту 3f8h.
```

```
!-------------------------------------------------------
  ! setcursor (4,1)
  ! type('Установка 115 кбод..<176> Ok<0dh,0ah>') !220
!-------------------------------------------------------

SH    show(4,1,'                          ')
      show(5,1,'              ADUC '&SN&'     ')!<0dh,0ah>')
      show(6,1,'                          ')
   !  type('<0dh,0ah>')
!-------------------------------------------------------
!   Стирание памяти микроконтроллера
!-----------------------------------
        loop i=1 to 5
         M[i]=MER[i]
         do OUTBYTE     !Вывод 5-ти байт команды ERASE
        .
        i=1
        do INBYTE       !Прием ответа ACK или NAC от ADUC.
        if (M1[1]=7) OR (M1[1]<>6)
         type('Ошибка...');goto M
        .
!       type('Стирание памяти..<176> Ok<0dh,0ah>') !220
!-------------------------------------------------------

!-----------------------------------------------
!  Открываем файл 'file.tsk' для программирования.
!-----------------------------------------------
        open(BINARY,00h)
        L=bytes(BINARY)       !Длина файла в байтах.
        set(BINARY)           !Читаем файл
        next(BINARY)          !файл BINARY.

!       type('Чтение файла file.tsk..<178> Ok<0dh,0ah>')
        setcursor(9,10)
        type('Программирование...<0dh,0ah>')
!---------------------------------------
        ADRMAX=16*int(L/16) !Целое количество строк по 16 символов в файле.
        LOST=L%16           !Длина остатка в байтах.
        LEND=16             !Длина стандартного куска программирования.
```

```
        if L<=16 then LEND=L.
!-----------------------------------
        N=1+int(L/256)
        k=1

        setcursor(10,10)
        type('<176>{16}')
        setcursor(10,10)
!-------------------------------------------------
! Готовим строку для программирования.
!=================================================
        loop ADR=0 to ADRMAX by 16      !Начало цикла.
!=================================================
  if ADR=ADRMAX then LEND=LOST.
  S0=''
  S0=sub(BIN:STR,ADR+1,LEND) !Выделяем строку данных - 16 символов.
!-------------------------------------------------------------------------
              NB=LEND+4             !NB=20-для целых, NB=LOST+4 -для остатка.
              SD=0                  !Вычисляем
          loop i=1 to LEND          !сумму
          SD=SD+D[i]                !данных.
            .
! ------START------ --NB--  --'W'--  -ADRU- -----ADRM-------  ----ADRL----
  M[1]=07h;M[2]=0eh;M[3]=NB;M[4]=57h;M[5]=0;M[6]=int(ADR/256);M[7]=ADR%256
  SUM=M[3]+M[4]+M[6]+M[7]+SD !Сумма байт для расчета контрольной суммы.
  KSUM=256*(1+int(SUM/256))-SUM    !Вычисляем контрольную сумму.
!-------------------------------------------------------------
!  Полный массив для вывода
!-------------------------------------------------------------------------
!           2b + 1b + 1b+ 1b +         2b +            X     + 1b =24(при16)
!          START  NB   'W' ADRU  ------ ADR---------  DATA     KS
!         07h 0eh NB   57h   0    Ст.б.Адр. Мл.б.Адр.  D[i]     Контр.с
!-------------------------------------------------------------------------
           loop i=1+7 to (LEND+7)    !Загрузка массива данных для вывода.
           M[i]=D[i-7]
            .
           M[LEND+8]=KSUM         !Загрузка контрольной суммы для вывода.
!-------------------------------------------------------------------------
```

```
      loop i=1 to LEND+8          !Вывод массива
     do OUTBYTE                   !в ADUC.
   !  type (M[i])
   !  type(' ')
      .
!-------------------------------------------------------------------------
          i=1
          do INBYTE                  !Прием ответа ACK или NAC от ADUC.
          if (M1[1]=7) OR (M1[1]<>6)
          type(' Ошибка...');goto M1
          .
         !type('<177>')
!=========================================
          if ADR=N*k*16 then type('<178>');k=k+1.
     .                          !Конец цикла.
!=========================================
       loop i=1 to 17-k !  type('<178>{5}')
       type('<178>')
       .
M      type(' Ok')
M1     type('<0dh,0ah>')
       type('         Конец программирования')
      . beep(0,500)
       goto E

WYKL show(23,26,'Для выхода нажми ESC')
                  loop until keyboard()
                    ask
                    if keycode()=256 then break.        !esc
                  .
!-------------------------------------
E              out(3fch,Y00h) !Сброс микроконтроллера
               return

!#############################################
!                  Подпрограммы
!---------------------------------------------
! П/п инициализации RS232 и микроконтроллера.
```

```
!-----------------------------------------
INIT       routine
!------------------------
!-------------------------
! Инициализация COM-порта.
!-------------------------
    out(3fbh,Y80h) ! DLAB=1 для установки делителя.
    out(3f8h,Y12)  ! Установить мл.б. скор.:Y01h-115200 бод,Y12-9600 бод.
    out(3f9h,Y00h) ! Установить ст.б. скор.=0.
    out(3fbh,Y03h) ! DLAB=0,Режим: 8 бит данных,1 стоп,нет пар.,сброс TxD.
    out(3f9h,Y00h) ! Запрет всех прерываний по порту 3f8h.
!-----------------------------------------------------
!-------------------------------------
! Инициализация микроконтроллера.
!-------------------------------------------------------------------------
    out(3fch,Y00h)    ! RTS=DTR=0: PSEN=1,RST=1 - сброс микроконтроллера.
    beep (0,30)       ! Задержка 0.2 сек.
    in(3f8h,B)        ! Холостой ввод - для сброса бита 0(DR) в 3fdh (в "0").
    out(3fch,Y02h)    ! RTS=1,DTR=0: PSEN=0,RST=1 - условия для загрузки.
    beep (0,30)       ! Задержка 0.2 сек.
    out(3fch,Y03h)    ! RTS=DTR=1: PSEN=0,RST=0 - загрузка микроконтроллера.
!-------------------------------------------------------------------------
! П/п ввода байта (байт в M1[i] )
!--------------------------------
INBYTE             routine
 loop;in(3fdh,B);if band(B,1)<>0;break.!Ожидание конца прихода байта(бит "DR"-data ready)
 in(3f8h,M1[i])                       !Чтение байта данных
!-----------------------------------------
! П/п вывода байта (байт в M[i] )
!-----------------------------------------
OUTBYTE            routine
 loop;in(3fdh,B);if band(B,20h)<>0;break.!Ожидание готовности передатчика
                                         !(transmitter empty).
 out(3f8h,M[i])                           !Вывод байта
 loop;in(3fdh,B);if band(B,40h)<>0;break..  !Ожидание выхода байта из PC(trasmitter holding
                                            !register empty.OK to send).
!                exit
!-----------------------------------------
```

8. Особенности проектирования систем сбора на базе микроконтроллеров, имеющих связь с компьютером по интерфейсу RS232

8.1. Выбор и подключение к микроконтроллерам кварцевых резонаторов и настройка их частоты

Обмен информацией между микроконтроллером и компьютером по интерфейсу RS232 на высоких скоростях вплоть до 115200 бод требует использования "качественных" кварцевых резонаторов. Здесь под словом "качественный" подразумевается, с одной стороны, достаточно близкое совпадение частоты, реально генерируемой резонатором, с той, которая от него требуется, с другой – малый дрейф этой частоты в зависимости от внешних факторов (температуры, времени и т.п.).

Стабильность работы выпускаемых промышленностью резонаторов достаточно высока. Как правило, уход частоты современных кварцевых резонаторов в зависимости от внешних факторов очень мал и не оказывает существенного влияния на надежность обмена по интерфейсу RS232.

Что касается *значения* частоты, которая генерируется микроконтроллером при подключении к нему конкретного кварцевого резонатора, то здесь возникают некоторые проблемы. Дело в том, что каждый кварцевый резонатор предназначен для конкретной электрической схемы его возбуждения. Схема возбуждения резонатора может быть "последовательной" (рис. 8.1, *б*) либо "параллельной" (рис. 8.1, *в*). Не рассматривая детально принципы работы кварцевых резонаторов, отметим один непреложный факт: кварцевый резонатор, предназначенный для работы, например, в параллельной схеме, будет давать другую частоту в последовательной, и наоборот. Во многих случаях это расхождение частот может достичь таких больших значений, что обмен информации по интерфейсу RS232 будет невозможен не только на скорости 115200 бод, но и на более низкой – вплоть до 9600 бод. В связи с этим многие фирмы-производители микроконтроллеров указывают не только конкретную марку кварцевого резонатора, который необходимо подключать к его выводам XTAL1 и XTAL2, но даже фирму-производитель резонатора. Как правило, кварцевые резонаторы, предназначенные

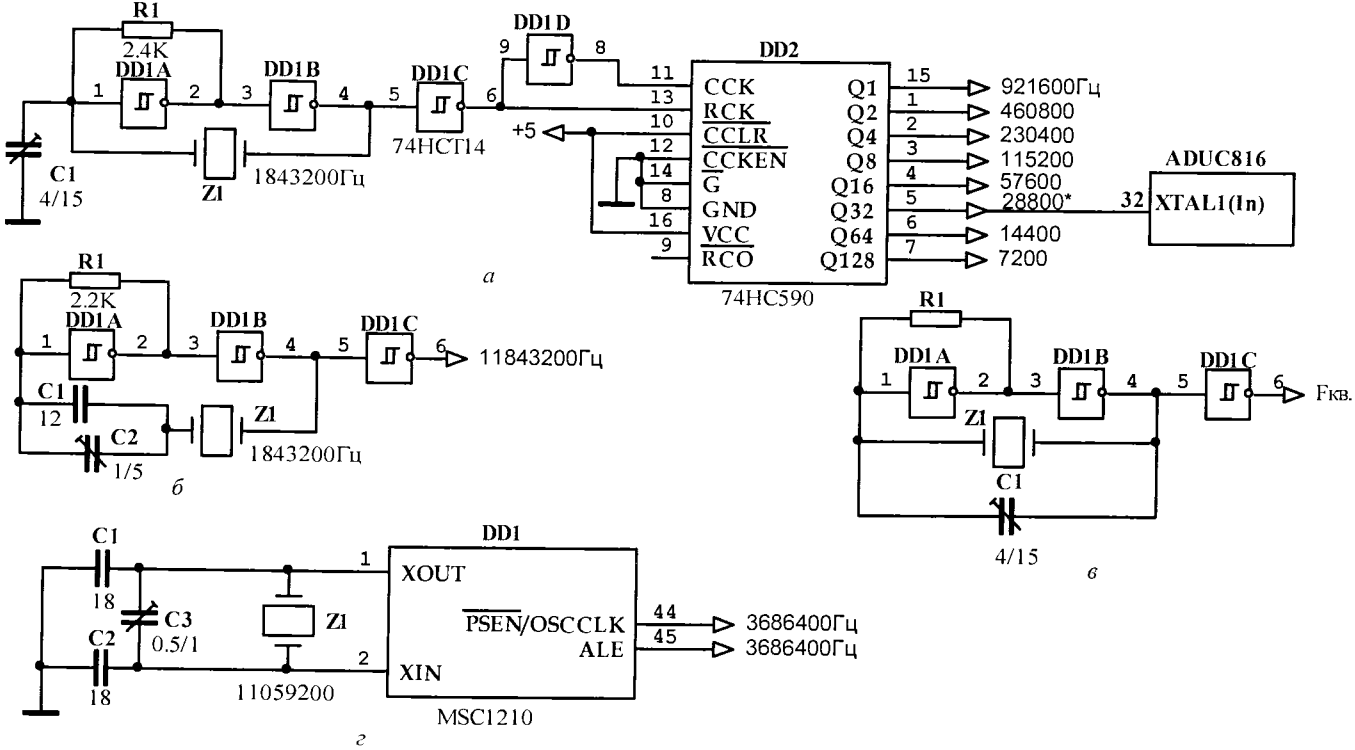


Рис. 8.1. Примеры подключения кварцевых резонаторов к микроконтроллерам

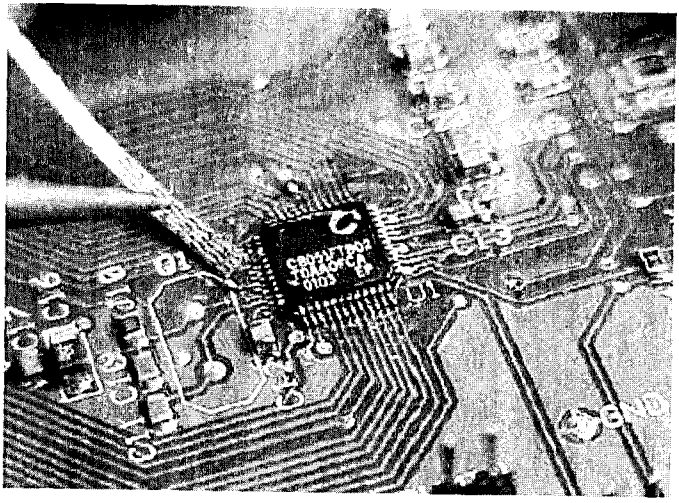
для работы в параллельной схеме, могут быть подключены к микроконтроллеру (разумеется, с двумя дополнительными конденсаторами небольшой емкости: 18–30 пФ) без каких-либо дополнительных подстроечных конденсаторов (триммеров). При необходимости более точной настройки частоты (для задания, например, временно́й базы повышенной точности системным таймером или для других целей) можно использовать триммер (С3 на рис. 8.1, *г*). Применять в такой схеме следует "параллельный" кварцевый резонатор. Для настройки желательно частотомер не подключать в тракт генератора (чтобы не вносить дополнительных искажений), а использовать выводы, на которых частота понижена, например, в три раза (ALE или PSEN).

В гл. 5 было упомянуто, что микроконтроллеры ADUC816 и ADUC824 при подключении к ним часового кварцевого резонатора (32768 Гц) не поддерживают скорость обмена по интерфейсу RS232, равную 115200 бод. Такую скорость обмена этих микроконтроллеров с компьютером можно получить, воспользовавшись схемой, приведенной на рис. 8.1, *а*. Необходимо отметить, что применение многоразрядных *асинхронных* счетчиков либо двух синхронных счетчиков, включенных *последовательно,* в качестве делителей частоты вместо многоразрядного синхронного счетчика (например, 74HC590), недопустимо. Это может значительно повлиять на стабильность частоты 28 800 Гц, которую надо подать на вывод XTAL1 (вывод 32 – вход генератора микроконтроллера). Настройку частоты необходимо производить триммером С1, измеряя частоту 28 800 Гц частотомером.

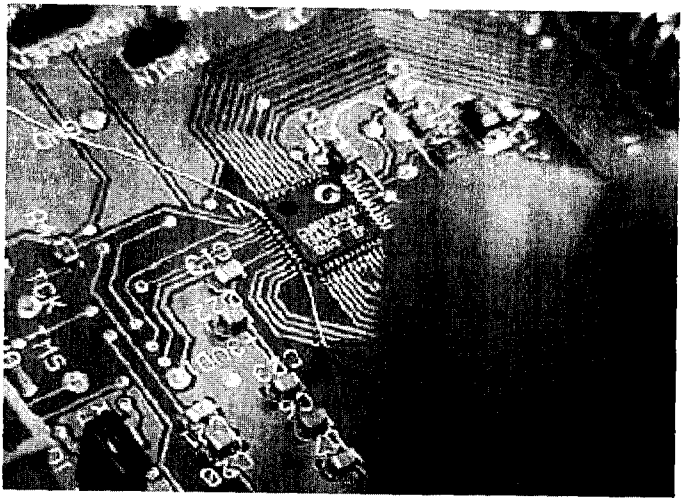
8.2. Макетирование аппаратных средств систем сбора

Микросхемы современных микроконтроллеров – систем на кристалле выпускаются в планарных корпусах TQFP-64 (MSC1210) или MQFP52 (ADUC8XX) с планарным расположением выводов. Ширина выводов микросхем около 0,3 мм, расстояние между ними 0,5 мм (TQFP64) и 0,65 мм (MQFP52). Макетных плат с такими параметрами не выпускается; разводить же плату только для того, чтобы выполнить макетирование каждого конкретного устройства – процедура достаточно долгая (и не дешевая). Кроме того, для того, чтобы поменять один уже распаянный на плате микроконтроллер на другой, т.е. выпаять один и впаять другой, нужно приложить достаточно много усилий. В связи с этим приведем пример, показывающий, как это делается (рис. 8.2).

Из рис. 8.2 видно, что отпайка выводов производится с помощью тонкой проволоки (*б–г*). При этом дорожки неизбежно отслаиваются и гнутся (*е*, *з*) и их необходимо выправлять. Можно себе

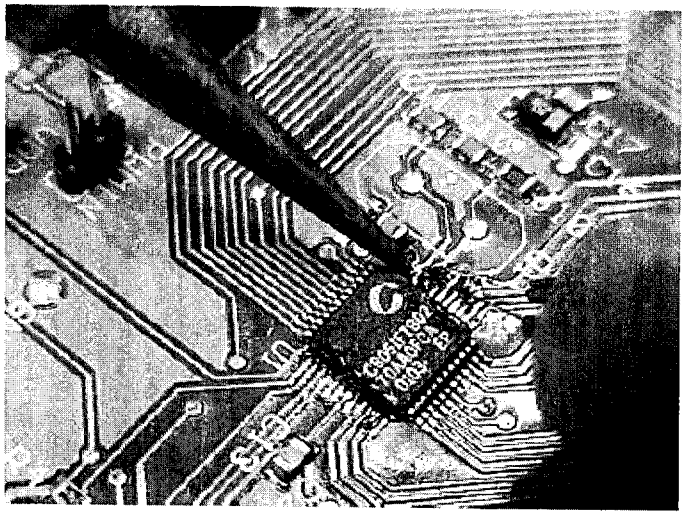


а

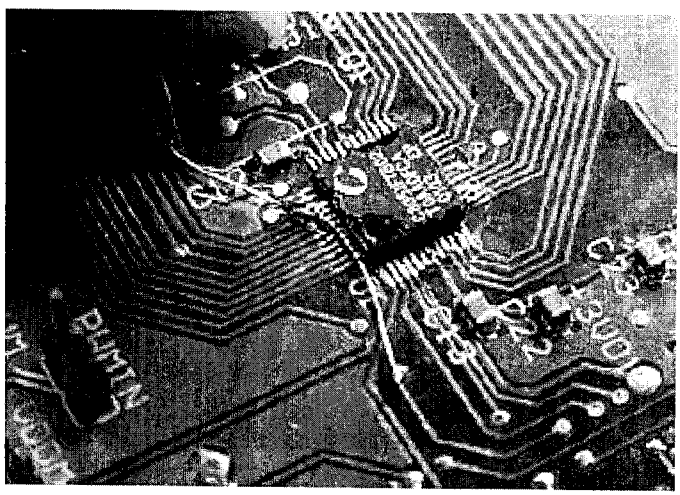


б

Рис. 8.2. Последовательность шагов при замене микроконтроллера в корпусе MQFP48 на новый

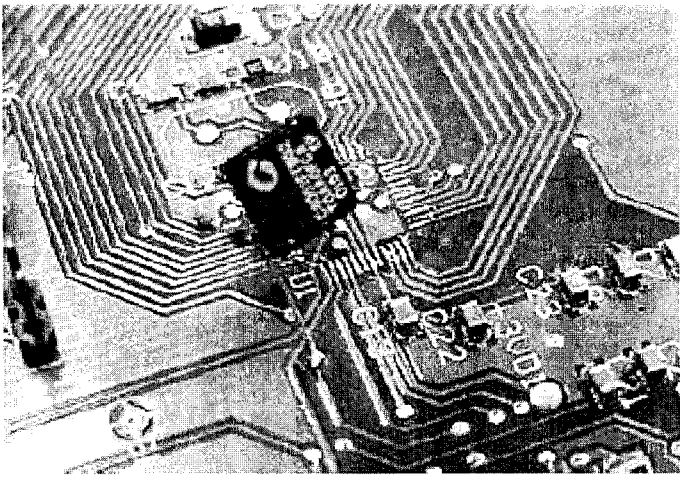


в

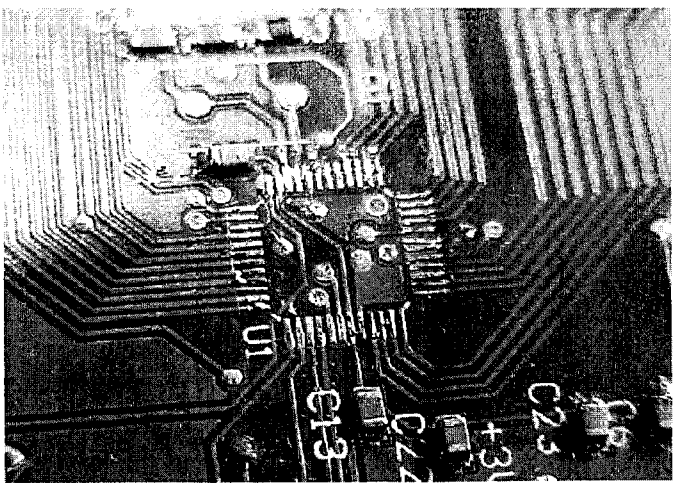


г

Рис. 8.2. *(Продолжение)*

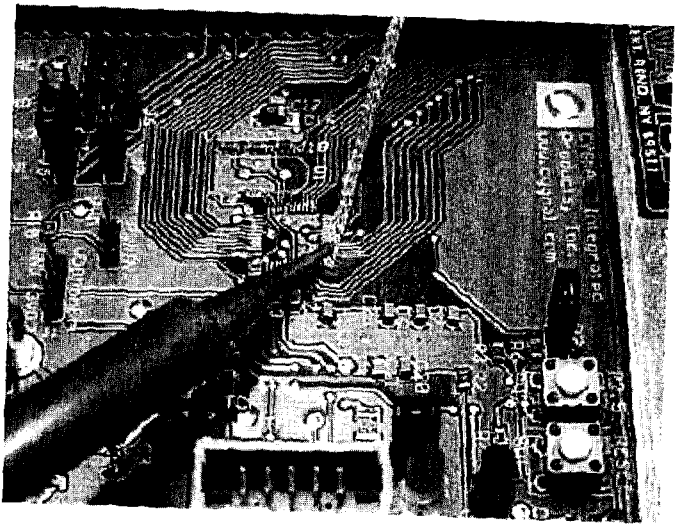


д

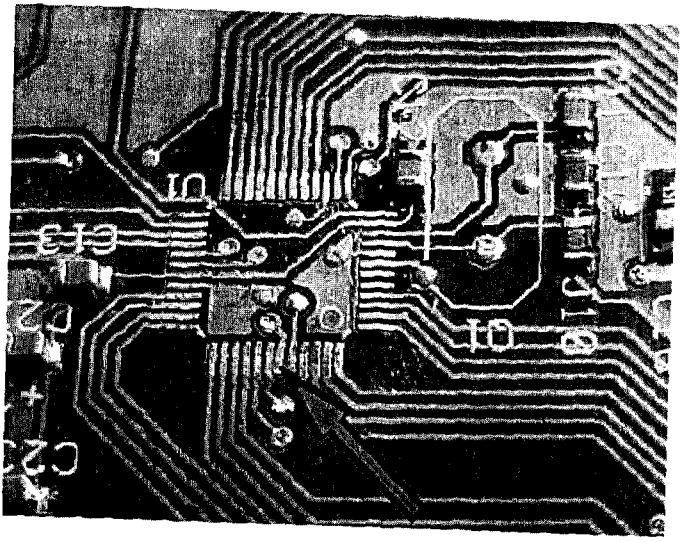


е

Рис. 8.2. *(Продолжение)*

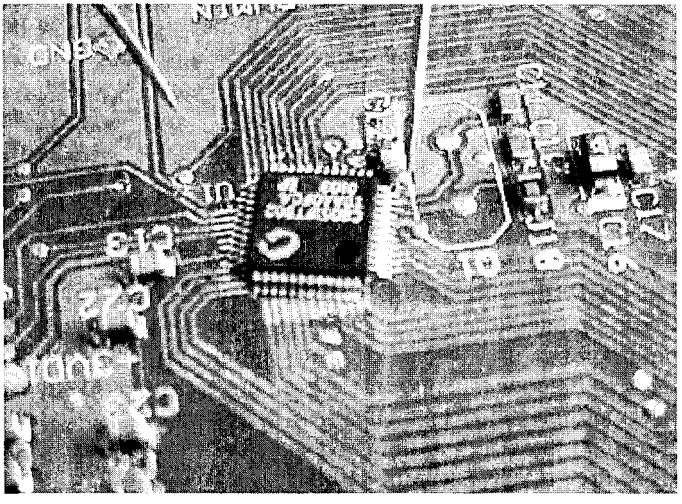


ж

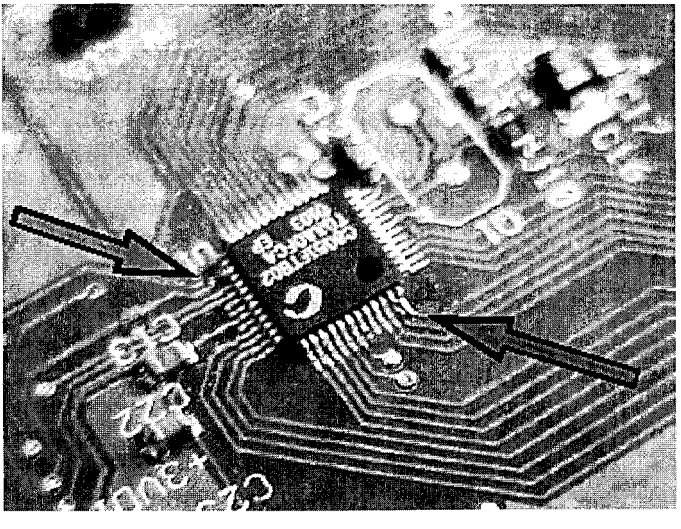


з

Рис. 8.2. *(Продолжение)*



и



к

Рис. 8.2. *(Окончание)*

представить, какой труд необходимо затратить, чтобы все получилось хорошо, и новый микроконтроллер после такой процедуры заработал! Отметим также, что пример приведен для микросхемы в корпусе MQFP48 с шагом 0,65 мм. Для микроконтроллеров в корпусах MQFP52 и тем более – TQFP64 с шагом 0,5 мм процедура существенно усложняется.

Процедуру замены микроконтроллера, приведенную на рис. 8.2, можно рекомендовать только для ремонта готового изделия, но ни в коем случае – для его разработки.

Для разработки устройств, в состав которых входят подобные микроконтроллеры, можно предложить следующий прием. Вначале изготавливается плата-переходник, например TQFP64-PGA, на которой распаиваются: микроконтроллер и система штырьков, точно совпадающих с отверстиями в панельке PGA64. Панелька PGA64 легко впаивается в достаточно простую макетную плату, на которой обычным изолированным многожильным проводом (например, МГТФ) вручную проводятся все необходимые соединения. После этого плата-переходник с микроконтроллером вставляется в панельку PGA, и начинается разработка программного обеспечения. При необходимости на макетную плату добавляются новые микросхемы, и т.п. Такой путь существенно быстрее любой разводки и изготовления платы для разработки. Кроме того, после окончания разработки плата-переходник с микроконтроллером легко вытаскивается из макетной платы для следующей разработки. При необходимости замены одного микроконтроллера другим (например, с целью увеличения (или уменьшения) его внутренней памяти программ), плата-переходник с одним микроконтроллером легко заменяется на другую (рис. 8.3).

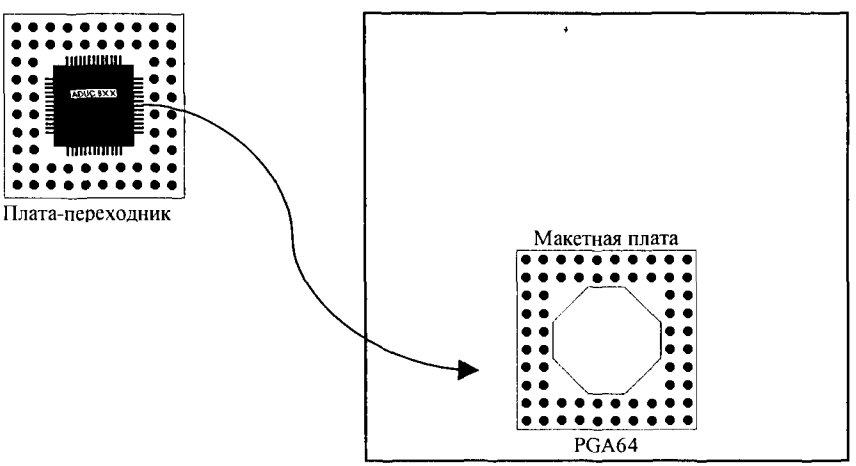


Рис. 8.3. Макетирование устройств при помощи платы-переходника и макетной платы с установленной панелькой PGA

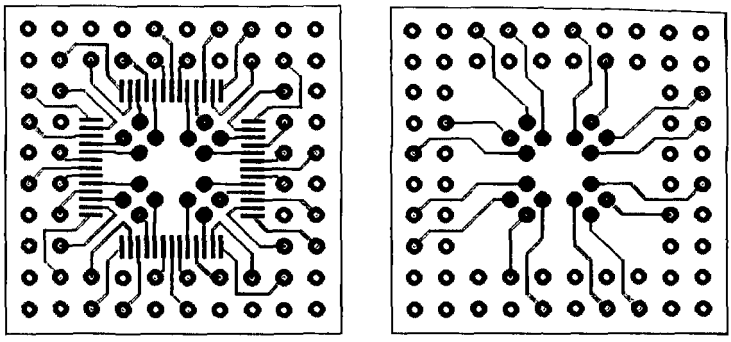


Рис. 8.4. Вариант разводки переходника MQFP52-PGA64

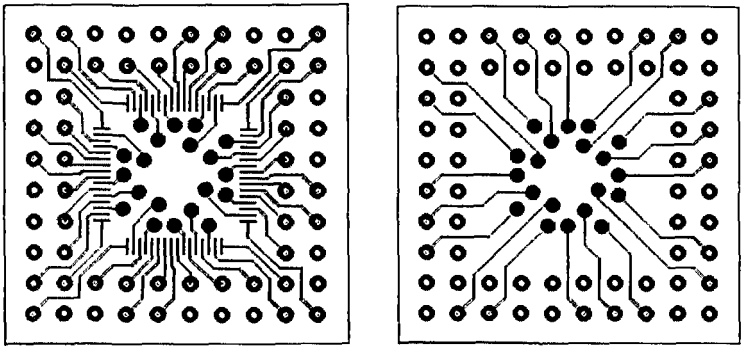


Рис. 8.5. Вариант разводки переходника TQFP64-PGA64

Возможные варианты разводки плат-переходников для корпусов MQFP52 и TQFP64 соответственно приведены на рис. 8.4 и рис. 8.5. Для других типов корпусов микросхем с планарными выводами, но либо с другим расположением выводов, либо с меньшим или большим числом выводов можно порекомендовать аналогичную разводку. Отметим, наконец, что развести и изготовить такие платы-переходники существенно быстрее и дешевле, чем разводить и изготавливать платы, предназначенные только для разработки устройств.

8.3. Программаторы микроконтроллеров

Современные микроконтроллеры – системы на одном кристалле, как уже неоднократно говорилось, позволяют производить программирование в системе, т.е. программировать внутреннюю память программ на готовом изделии (плате). В связи с этим может сложиться ошибочное мнение, что программаторы для них не нужны, а при ориентации производства на использование только таких микроконтроллеров про программаторы можно, наконец, вообще забыть. Однако это не совсем так.

Дело в том, что выпуск конкретных устройств на базе микроконтроллеров налагает определенные экономические требования на сами устройства (например, их себестоимость, стоимость их разработки и т.п.). Далеко не все устройства нуждаются в использовании систем на одном кристалле. Во многих из них достаточно использовать более примитивные (и более дешевые) микроконтроллеры. *В этом случае требуются программаторы.*

Даже при применении только микроконтроллеров, имеющих возможность программирования в системе, перед производством подобных устройств необходимо потратить немало времени и средств на их разработку. А вот при их разработке как раз и требуются программаторы. Причин несколько. Во-первых, программатор обладает более универсальными свойствами работы с программой микроконтроллера. В отличие от только программирования, программатор позволяет прочитать программу, записанную в память программ микроконтроллера, сделать верификацию этой программы с той, которая записана на компьютере в файле, дает возможность программировать как в *.hex-формате, так и в *.bin-формате (это бывает очень полезно при определении фактического объема программы, например, если помимо самой программы в память микроконтроллера записываются еще и данные). Во-вторых, режим параллельного программирования позволяет снять биты секретности, недоступные при работе в режиме программирования в системе. В частности, последовательный режим программирования может быть запрещен при работе в этом самом режиме, а вот снять это запрещение можно только при программировании микроконтроллера в параллельном режиме. Все эти и многие другие не перечисленные здесь свойства программаторов с параллельным режимом программирования являются совершенно необходимыми атрибутами разработчика. Поэтому даже для микроконтроллеров, имеющих возможность программирования в системе, программаторы совершенно необходимы.

Если доводы, приведенные автором по поводу необходимости программаторов, читатель посчитает неубедительными, он может пропустить эту главу.

Перед тем как покупать программатор, пользователь должен четко представлять, какими свойствами должен обладать этот программатор. Приведем некоторые рекомендации по этому вопросу.

1. *Программатор, прежде всего, должен программировать те микросхемы, которые требуется.* Существует множество универсальных программаторов, которые имеют возможность программирования тысяч (!) различных микросхем, в том числе память (последовательную и параллельную), микросхемы логики (PAL, GAL и т.п.) и большое число различных микроконтроллеров.

2. *Программатор должен иметь приемлемую цену.* Универсальные программаторы, как правило, стоят очень дорого, и если от программатора требуется только 10% его возможностей, то такой программатор покупать не рекомендуется.

3. *Программатор должен иметь возможность быстрого и простого подключения к компьютеру.* Это подключение не должно затрагивать других периферийных устройств компьютера (если, например, программатор подключается к принтерному разъему, к которому подключен принтер, то его необходимо отключать, что не совсем удобно). *Наиболее подходящий интерфейс для подключения программатора – это интерфейс RS232.* Тем более, что самый примитивный компьютер оборудован как минимум двумя такими интерфейсами. Это порты COM1 и COM2, один из которых, как правило, свободен.

4. *Программатор должен:*

а) программировать в микроконтроллер файлы *.hex- и *.bin-форматов;

б) производить верификацию программы с файлами этих же форматов;

в) производить защиту памяти микроконтроллера от несанкционированного доступа, т.е. устанавливать биты секретности микроконтроллера;

г) выполнять чтение сигнатурных байт и определять марку микроконтроллера;

д) выполнять чтение памяти микроконтроллера и индикацию этой памяти в шестнадцатеричных кодах на экране; при необходимости должна существовать возможность записи содержимого памяти в какой-либо файл или распечатки на принтере;

е) предоставлять пользователю выбирать номер порта компьютера (COM1 или COM2) и скорость обмена по интерфейсу RS232 (т.е. пользователь должен иметь возможность настройки порта); при работе по этому интерфейсу скорость обмена должна быть максимально возможной для компьютера, т.е. 115200 бод;

ж) работать в операционных системах DOS и Windows одинаково хорошо (т.е. не зависеть от операционной системы, в которой он используется);

з) быть удобным и простым в работе, обязательно иметь поддержку "мыши" и небольшое (оперативно вызываемое одной кнопкой) руководство по использованию.

Этими минимальными требованиями и ограничиваются рекомендации по выбору программатора. Некоторые программаторы обладают дополнительными свойствами, не перечисленными в пп. а)–з). Это, например, дезассемблирование прочитанной программы, изменение кодов программы и повторное программирование исправленной программы, защита микросхемы от неправильной

установки и т.п. Все эти дополнительные свойства, на взгляд автора, значительно усложняют (и, как следствие, удорожают) программатор. А цена программатора все-таки должна быть приемлемой.

Вообще, программатор, подключенный к компьютеру по интерфейсу RS232, представляет собой миниатюрную автоматизированную систему сбора и управления (хотя, конечно, достаточно примитивную). Разработчику достаточно сложных компьютерных систем сбора и обработки информации трудно отказаться от соблазна сконструировать подобное устройство. В связи с этим в настоящее время выпускается и продается множество самых различных программаторов.

Не избежал подобного соблазна и автор настоящей книги. Вниманию читателей предлагается еще один программатор 51-совместимых микроконтроллеров.

Конструкция программатора достаточно проста. Он выполнен в виде платы с ножками (бескорпусной вариант), габаритными размерами 75×75×15 мм, на которой расположены две панельки с нулевым усилием (с рычажком) для установки микроконтроллеров (20 PINDIP – для AT89C1051/2051/4051 и 40 PINDIP – для всех остальных), выключатель питания, три светодиода, предназначенные для контроля работы программатора, разъемы для подключения кабеля связи с компьютером и выносного блока питания (+12 В), а также электрическая схема.

Электрическая схема программатора содержит стабилизатор постоянного тока +5 В, необходимого для работы программатора, последовательный интерфейс RS232 с компьютером, а также вспомогательные устройства.

В программаторе применен однокристальный микроконтроллер AT89C52, который выполняет основную работу программатора: обменивается с компьютером информацией и управляет работой программатора. Среды работы программатора: MSDOS и Windows'95, 98. Читатель, наверное, уже догадался, что программатор обладает всеми перечисленными свойствами (а–з). Некоторые из открывающихся окон при работе программатора приведены на рис. 8.6–8.17. Из рис. 8.7, в частности, можно определить, с какими микроконтроллерами может работать программатор. Остальные окна в особых комментариях не нуждаются. Ранее в гл.1 (п.1.3.3) уже был приведен общий вид окна для настройки порта программатора (вариант 3).

Для программирования микроконтроллеров MSC1210YX и ADUC8XX, распаенных на платах-переходниках (см., например, рис. 8.3), к программатору поставляются дополнительные переходники PGA64-DIP40, а для программирования микроконтроллеров в корпусах PLCC44 – переходник PLCC44-DIP40.

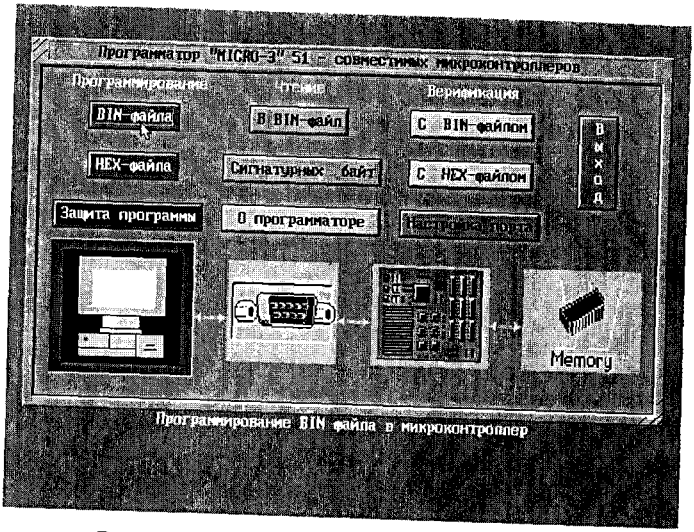


Рис. 8.6. Общий вид окна меню программатора

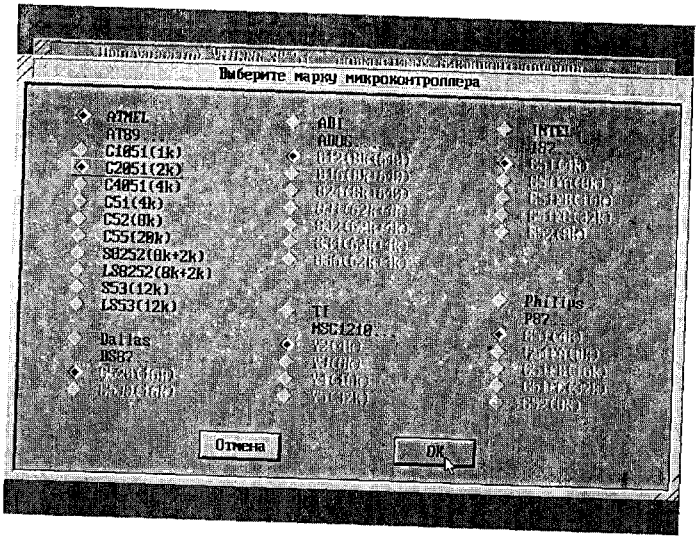


Рис. 8.7. Выбор марки микроконтроллера (AT89C2051)

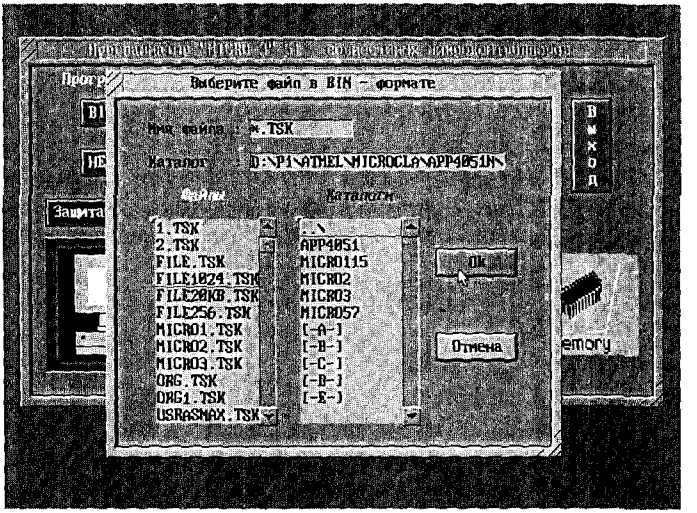


Рис. 8.8. Выбор файла (FILE1024.TSK) для программирования

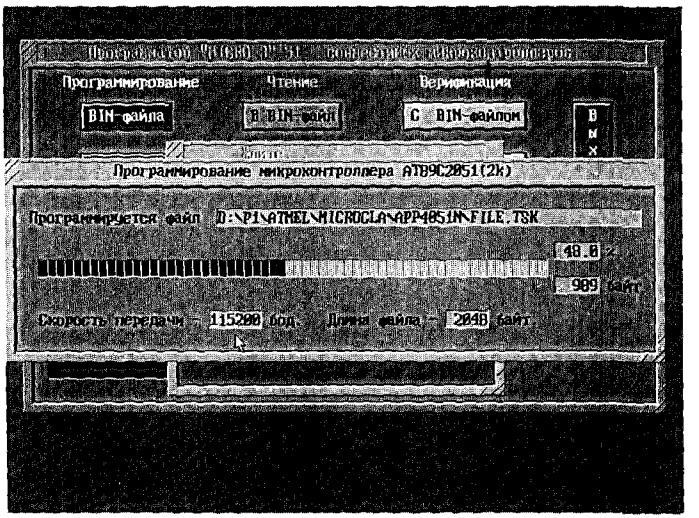


Рис. 8.9. Процесс программирования файла FILE.TSK

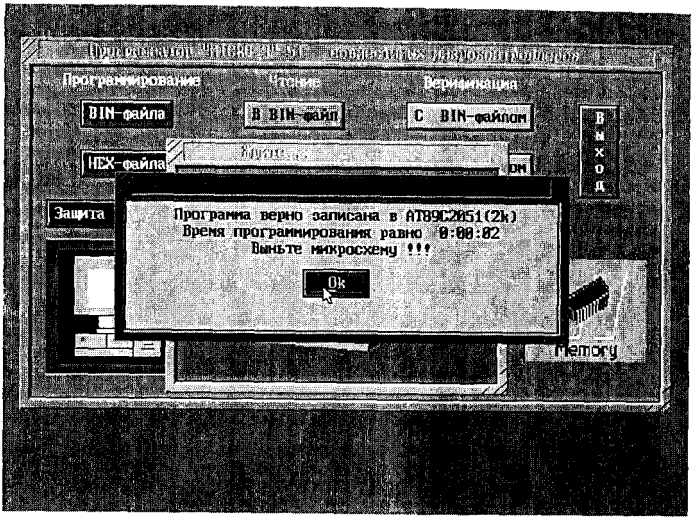


Рис. 8.10. Индикация окончания и времени (2 с) программирования

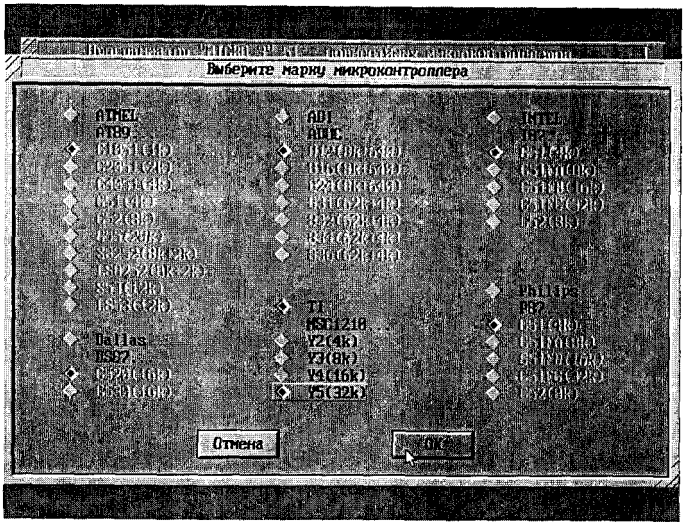


Рис. 8.11. Выбор микроконтроллера MSC1210Y5

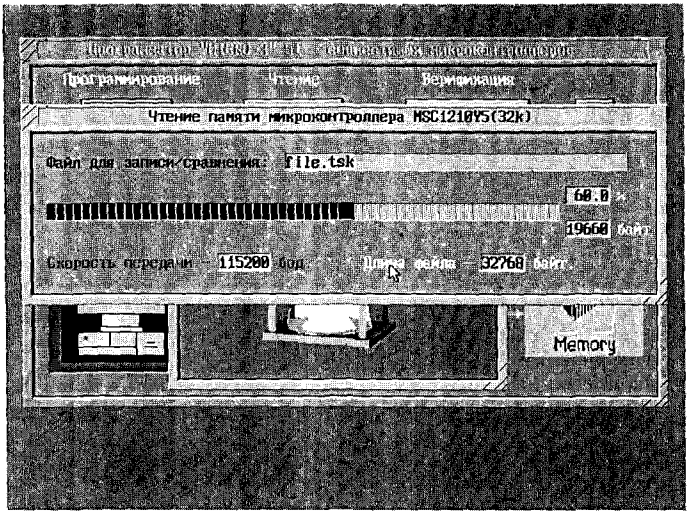


Рис. 8.12. Процесс чтения памяти микроконтроллера MSC1210Y5

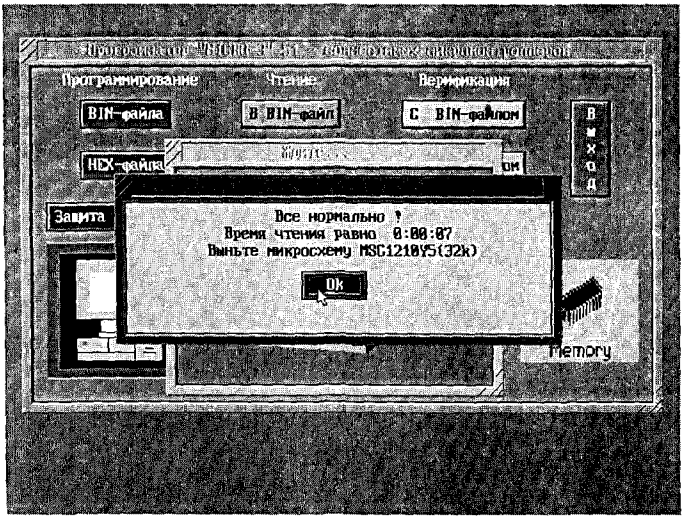


Рис. 8.13. Индикация окончания чтения памяти и времени чтения (7 с) микроконтроллера MSC1210Y5

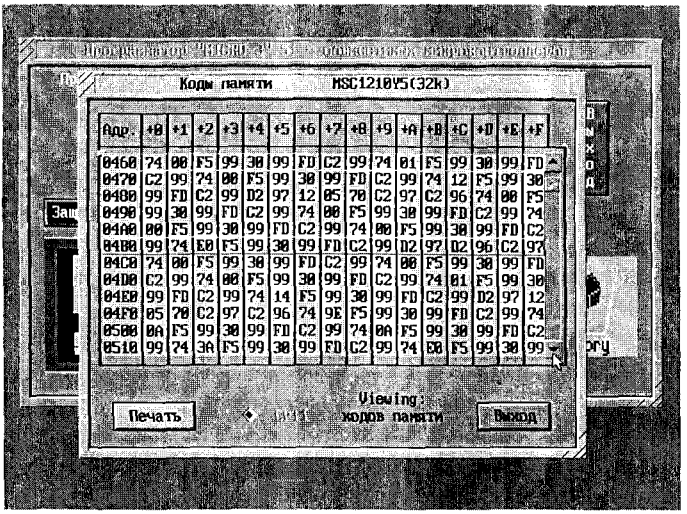


Рис. 8.14. Просмотр шестнадцатеричных кодов памяти, прочитанной из микроконтроллера MSC1210Y5

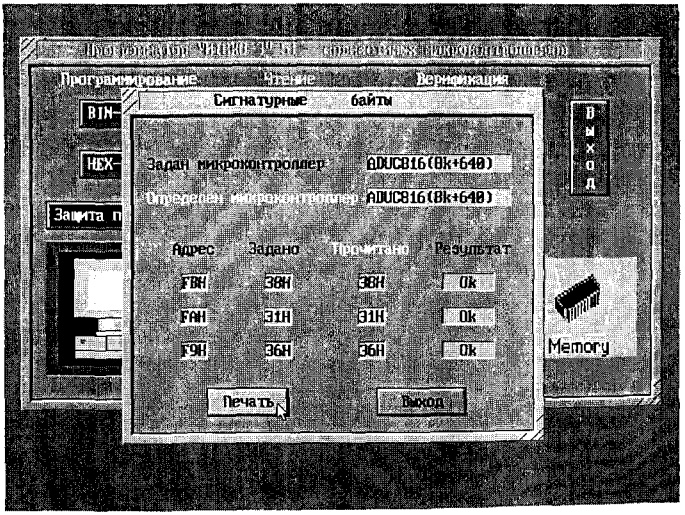


Рис. 8.15. Чтение сигнатурных байт и определение марки микроконтроллера (ADUC816)

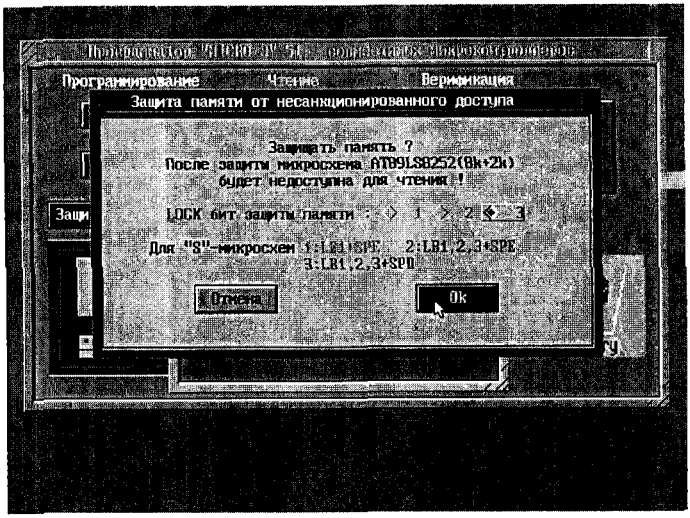


Рис. 8.16. Установка максимальной степени защиты памяти от несанкционированного доступа (3 Lock-бит)

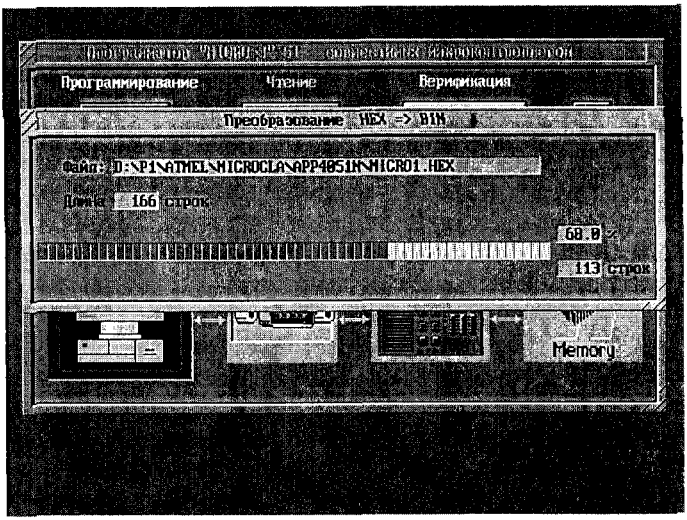


Рис. 8.17. Процесс преобразования файла из *.hex-формата в *.bin-формат для программирования или верификации

9. Вместо заключения: что может и чего не может RS232

Предыдущая глава показывает, что интерфейс RS232 неплохо справляется с устройствами, подобными программаторам. Является ли это его пределом? И вообще, каков предел применения интерфейса RS232 в системах, которые используют микроконтроллеры в качестве удаленных от компьютера устройств, обменивающихся с компьютером информацией? Опыт конструирования подобного типа систем показывает, что интерфейс RS232 целесообразно использовать в таких системах сбора и обработки информации, в которых число датчиков не превышает 20–30, а частота обновления информации, получаемой с них, 20 Гц, т.е. дискретность по времени (Δt) не менее 0,05 с. На практике встречается масса задач, где требуется достаточно прецизионные измерения низкочастотных сигналов, а дискретность обновления данных намного больше чем 0,05 с. (1 с и более). В качестве примера можно привести одну из систем, применяющуюся как автоматизированное средство поверки и градуировки счетчиков объема газа или воды, о которой более подробно можно прочитать в приложении.

Целесообразность применения интерфейса RS232 в таких устройствах доказана многолетним "стажем" их работы (все подобные системы сбора, сконструированные автором и запущенные в эксплуатацию, работают более 10 лет). Косвенным подтверждением этой целесообразности может служить еще и тот факт, что такие ведущие фирмы-производители аналоговых микросхем АЦП и ЦАП, как Analog Devices и Texas Instruments с недавнего времени стали выпускать микроконтроллеры – системы на кристалле (ADUC8XX и MSC1210), которые используют интерфейс RS232 не только в качестве обычной связи с компьютером. Эти микроконтроллеры (как уже обсуждалось ранее) имеют возможность программирования в системе *именно* по интерфейсу RS232.

Несколько слов об оптронных развязках. Оптроны благодаря своей инерционности идеально подходят для работы с интерфейсом RS232. Более высокочастотные интерфейсы (типа SPI, USB и т.п.) плохо работают со стандартными оптронами. Для них требуются либо очень высокочастотные оптроны, либо емкостные гальванические развязки. Стоимость и тех и других – в несколько раз больше, чем обычных оптронов. В этом смысле интерфейс RS232

имеет большое преимущество. Так что интерфейс RS232 вряд ли в скором будущем "пустит" в свою "нишу" какой-либо другой интерфейс.

Что касается более высокочастотных измерений, чем те, о которых сказано выше, то для таких измерений интерфейс RS232 не подходит. Не годятся для них и вышеупомянутые микроконтроллеры.

В настоящее время (как упоминалось ранее) фирмы CYGNAL, GOAL начали выпуск микроконтроллеров, в которые входят 16-разрядные АЦП с частотой оцифровки аналогового сигнала до 1 МГц (CYGNAL). Для высокочастотных измерений целесообразно применять либо такие микроконтроллеры, либо платы с высокочастотными АЦП, вставляющиеся в компьютер и сопрягающиеся с ним по внутренней шине (ISA или PCI). При применении микроконтроллеров связь с компьютером должна быть достаточно высокоскоростной.

И, наконец, последнее, о чем нельзя не упомянуть, – появление новых версий языка Кларион для Windows (последняя версия – Clarion 5.5). По сравнению с Кларионом для DOC Clarion 5.5 обладает поистине фантастическими возможностями. Это еще более удобный и дружественный интерфейс с пользователем, прекрасная и разнообразная графика, работа в Win'98/XP, значительно большие возможности и удобство программирования и, как следствие, необычайно быстрая разработка программ, высокая скорость их работы (особенно в Win'XP).

К сожалению, у Clarion 5.5 отсутствуют встроенные команды ввода/вывода через порт (**in** и **out**). Этот недостаток, однако, легко преодолим – в языке TopSpeed C++ (TSCPP), компилятор которого встроен в Clarion 5.5, эти команды присутствуют. Написав всего две подпрограммы ввода и вывода через порт на TSCPP (несколько операторов) и использовав их как внешние (Extern) в основной программе на Clarion 5.5, можно запрограммировать обмен по RS232. А воспользовавшись новым алгоритмом обмена (гл. 6), можно получить высокие скоростные и надежностные характеристики связи компьютера с микроконтроллером уже в Win'98/XP (сейчас автором ведутся подобные разработки). Это позволит поднять качество программ для компьютерных систем сбора и обработки информации на новый современный уровень.

Приложение

Удаленная система сбора и обработки информации, поступающей с датчиков аналоговых, частотных и дискретных сигналов на базе IBM-совместимого компьютера и MSC-51-совместимого микроконтроллера

1. Краткое описание

Система предназначена для предварительной обработки и ввода в компьютер сигналов с датчиков аналоговых (ток или напряжение), частотных (частота, количество импульсов) и дискретных (включено или выключено) сигналов.

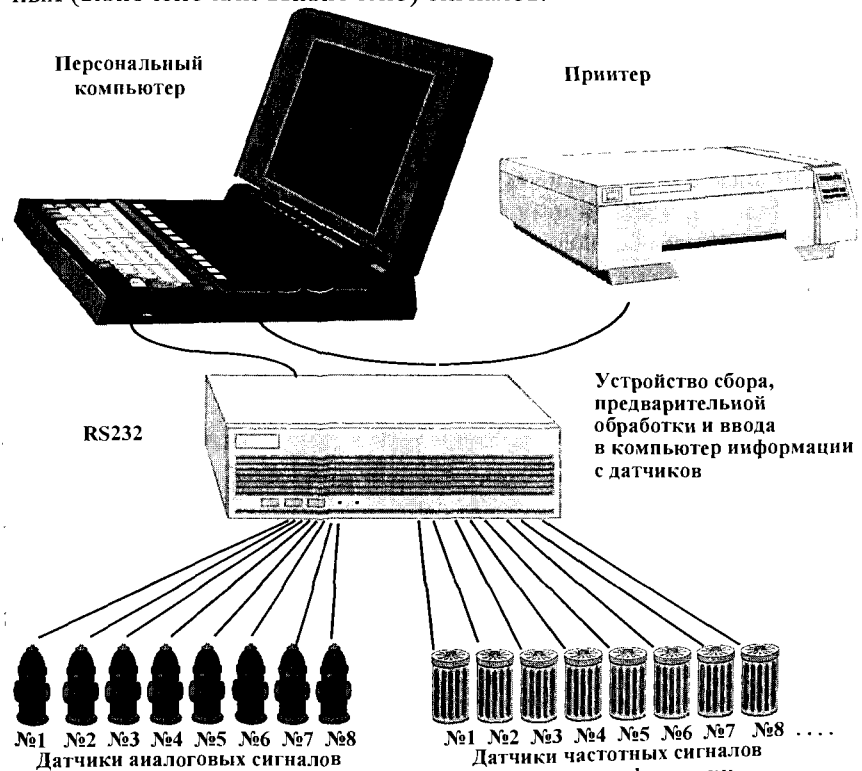


Рис. П.1. Схема системы сбора и обработки информации

Система состоит из компьютера, устройства сбора, предварительной обработки и ввода в компьютер показаний датчиков и программного обеспечения (рис. П.1).

Устройство сбора – прибор в корпусе, который подключается к компьютеру через стандартный интерфейс RS232 (последовательный порт COM1 или COM2). Прибор гальванически развязан от компьютера с помощью оптронных развязок на пробивное напряжение до 1500 В. Удаление прибора от компьютера может достигать нескольких десятков метров и более в зависимости от скорости обмена информацией.

Прибор оснащен 16(24) разъемами, к которым подключаются кабели от 8 аналоговых и от 8(16) частотных датчиков, а также разъемом, к которому подключаются дискретные сигналы.

Прибор является специализированным микрокомпьютером на базе однокристального MSC51-совместимого микроконтроллера P80C552 (фирмы Philips) со встроенным 10-разрядным 8-канальным аналогоцифровым преобразователем или микроконтроллера AT89S8252 (фирмы ATMEL) с сопряженными с ним 8-канальным коммутатором каналов фирмы Analog Devices и одноканальным 24-разрядным сигма-дельта аналогоцифровым преобразователем (используется 16 разрядов) фирмы Burr-Brown.

Прибор имеет встроенную (постоянную и оперативную) память, которая позволяет запоминать входную информацию, предварительно ее обрабатывать и передавать в компьютер в сжатом виде. Наличие встроенной оперативной памяти программ позволяет отказаться от дорогостоящих эмуляторов и программаторов для микроконтроллеров и ПЗУ. Все программы (в том числе и для микроконтроллеров) пишутся на компьютере и (при необходимости) переносятся в оперативную память программ через тот же интерфейс RS232.

Микроконтроллер позволяет предварительно обрабатывать входную информацию и передавать ее в компьютер, где производится окончательная обработка и распечатка измеренных данных на экране или принтере.

Программное обеспечение и конфигурация системы согласуются с заказчиком. Программы работают в оконном режиме (меню), поддерживается "мышь". Операционные системы, в которых работает устройство: MS-DOS или WINDOWS'95(98).

2. Назначение и область использования

Система сбора и обработки может найти применение в разнообразных отраслях науки, техники и производства:

1. Автоматизированная градуировка, калибровка и поверка расходомеров и счетчиков объема жидкостей и газов, а также теплосчетчиков.

2. Автоматизированный ввод в компьютер показаний датчиков температуры и давления.

3. Автоматизированный ввод в компьютер показаний датчиков, измеряющих гидродинамические параметры турбулентных потоков: скорости, давления, сил и т.п. (микровертушки, тензо- и термодатчики и т.д.), используемых в научных исследованиях.

4. Вибродиагностика.

5. Медицина (автоматизированный ввод в компьютер показаний с датчиков, измеряющих медико-биологические параметры человека: частота дыхания и сердечных сокращений, кровяное давление, биотоки мозга и т.п.).

3. Технические характеристики

1 Диапазон аналоговых сигналов:

- напряжение 0–+5 В,
- ток 0–20 мА или 4–20 мА,
- погрешность измерения аналогового сигнала: ±0,005 В (10-разрядный АЦП) или ±0,0005 В (16-разрядный АЦП).

2. Число аналоговых сигналов –8.

3. Диапазон частотных сигналов:

- частота 0,1–10000 Гц,
- число импульсов 1–65536.

4. Число дискретных сигналов вывода – 1, ввода – 1. Дискретные сигналы совместимы со стандартными TTL-уровнями. Возможна поставка системы с гальванической развязкой дискретных сигналов.

5. Скорость обмена информацией с компьютером: 9600 и 115200 бод.

6. Габаритные размеры: 30×20×8 см (корпус от выносного привода CD к компьютеру notebook), 40×60×9,5 см (компьютерный корпус Slim), 40×60×8 (корпус Super Slim), 40×60×7,5 (корпус Work Station).

7. Питание системы: 220 В, 15 Вт.

8. Удаление от компьютера – десятки метров и более.

9. Интервал времени измерения 0,05–3000 с.

10. Число частотных каналов: 8 (с 16-разрядным АЦП), 8 и 15 (с 10-разрядным АЦП).

4. Технико-экономическая эффективность

Система сбора и обработки на несколько порядков увеличивает точность измерений, сокращает время обработки информации как по сравнению с ручной обработкой, так и по сравнению с ручным вводом в компьютер и компьютерной обработкой, так как в состав системы входит микроконтроллер, производящий предварительную обработку. В ряде случаев система сбора является единственно возможным средством измерений и обработки их результатов и не имеет альтернативы.

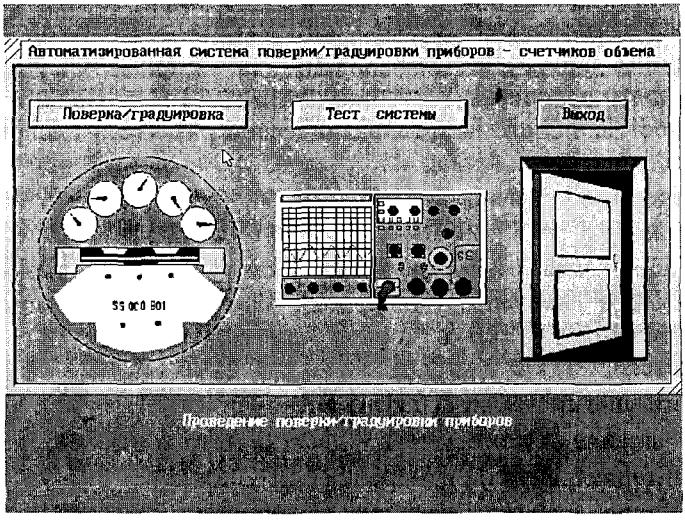
5. Сведения о документации

Документация поставляется с системой сбора на дискете или компакт-диске (программное обеспечение) и распечатанной на бумаге (описание работы).

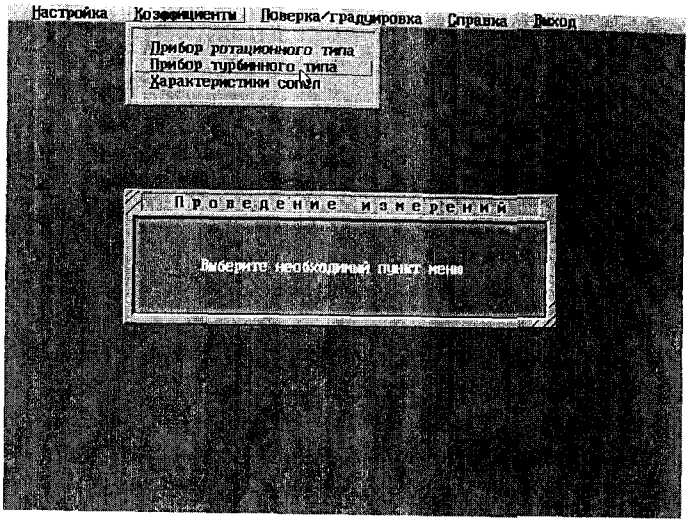
6. Сведения о внедрении

Системы сбора и обработки (в различных модификациях) работают на московском заводе "Водоприбор" (водяная система), в А/О "Моспромгаз" (газовая система), Белгородском заводе расходомеров (две газовые системы), г. Салават (водяная система), г. Уфа (водяная система), г. Тула (газовая система), г. Рязань (водяная система), г. Брянск (газовая система) и др.

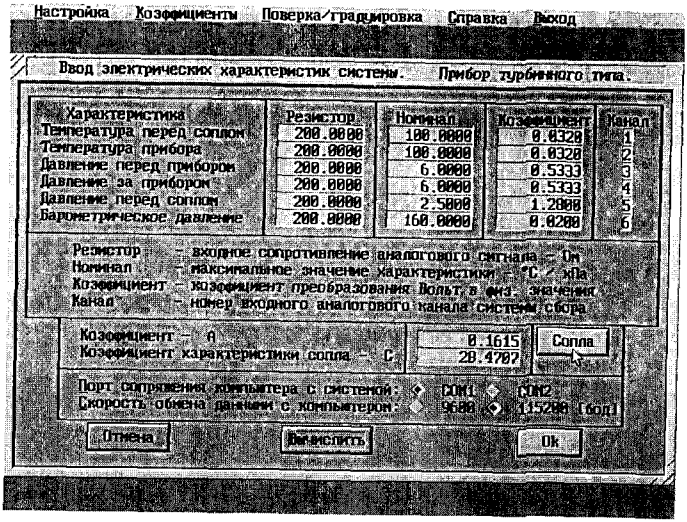
7. Некоторые окна, открывающиеся в программе поверки счетчиков объема



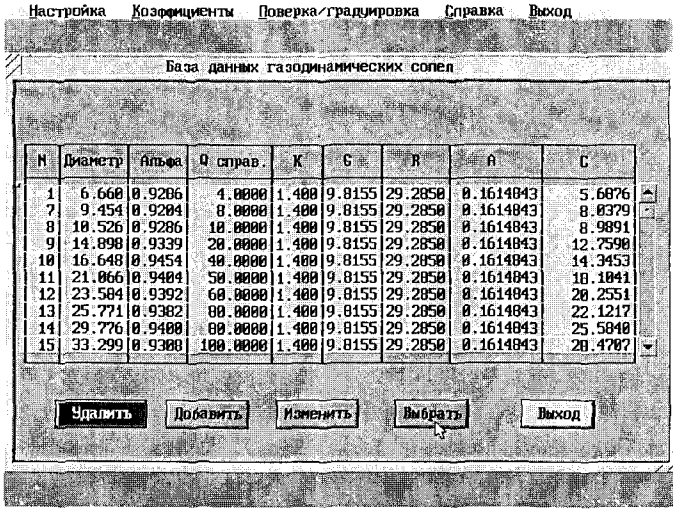
Общий вид меню программы



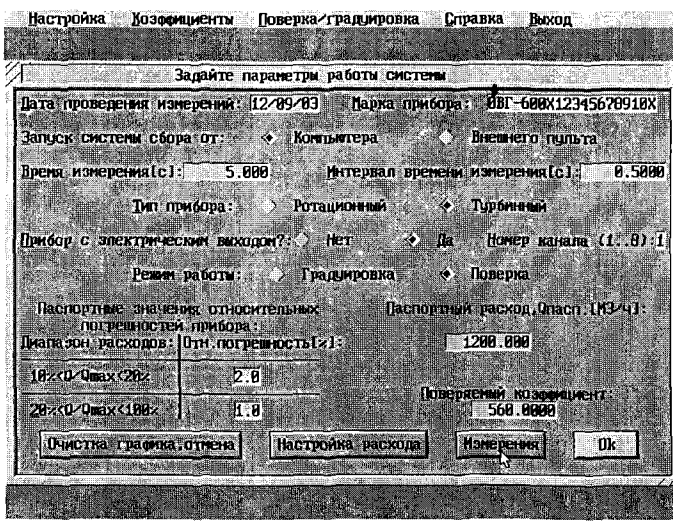
Выбор типа поверяемого прибора



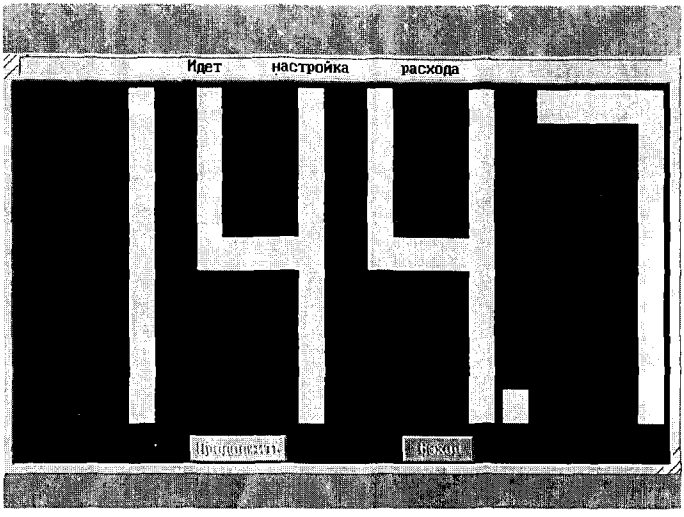
Ввод электрических параметров датчиков



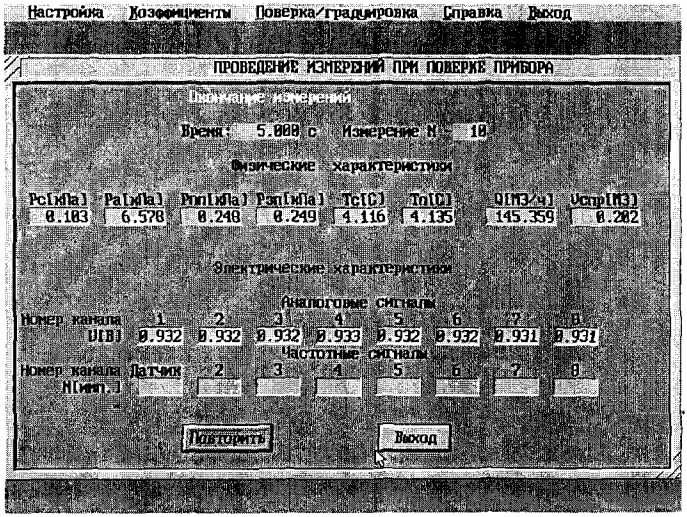
Выбор необходимого сопла для поверки



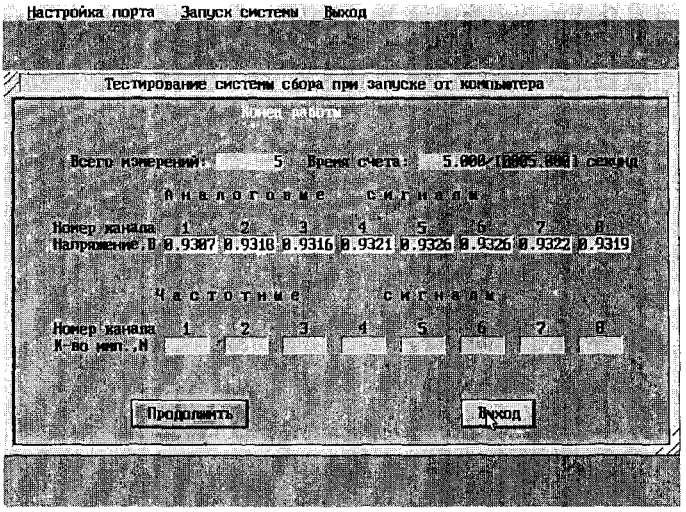
Выбор режимов измерения



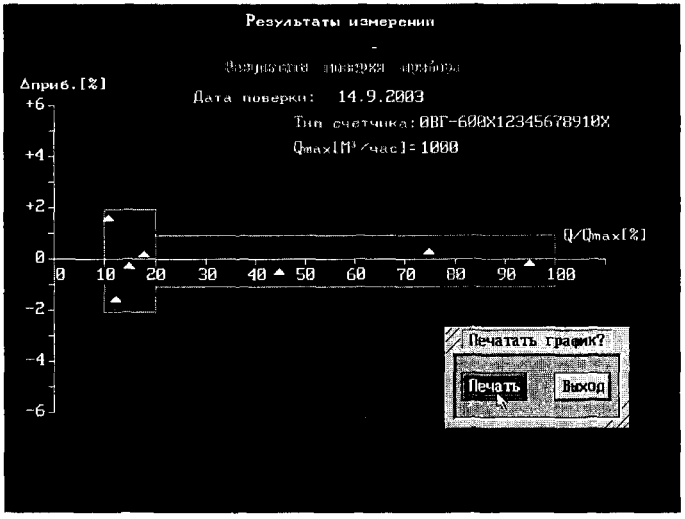
Настройка расхода (экран видно издалека)



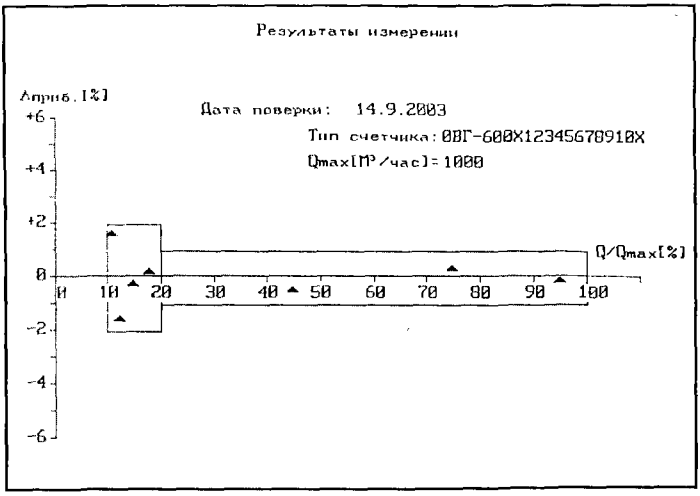
Процесс измерений



Тест системы сбора



Результат поверки: "Прибор годен" (погрешность в заданных границах)



Распечатка графика на бумаге

8. *Общий вид установки по поверке газовых счетчиков (А/О "Моспромгаз").*

Общий вид установки, в которой используется система сбора, показан на рис. П.2. УСО расположено в корпусе выносного привода CD для компьютера notebook (рис. П.3).

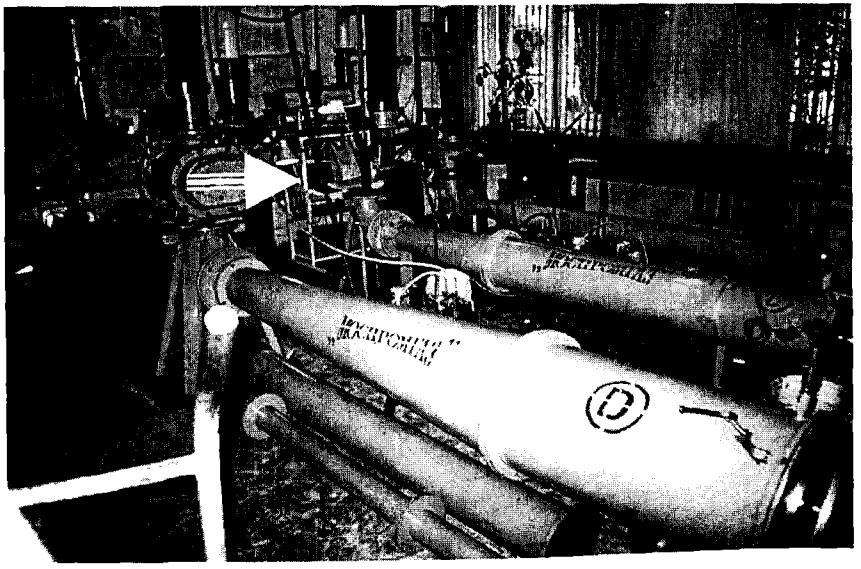


Рис. П.2. Общий вид установки. Стрелкой показана стойка с системой сбора

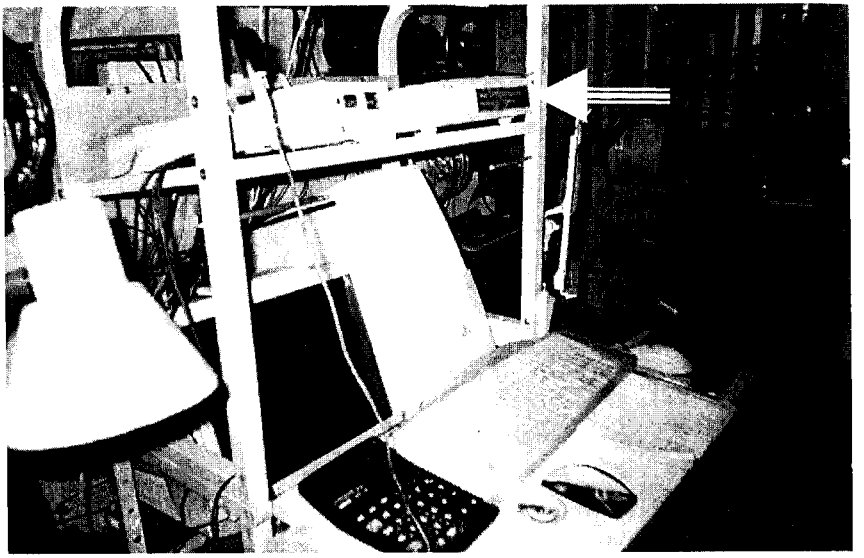


Рис. П.3. Общий вид стойки с системой сбора. Стрелкой показано УСО

Список литературы[1]

1. Печатные издания

Баррингтон Брюс Б. Как создавался Кларион // Мир ПК. – 1993. – № 2. – С. 56–64.

Боборыкин А.В., Липовецкий Т.П. и др. Однокристальные микроконтроллеры. – М.: МИКАП, 1994.

Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах. – М.: Радио и связь, 1996.

Живая электроника России 2003 // Электронные компоненты: Спец. выпуск. – 2003. – С. 86–93.

Интегральные микросхемы: Микросхемы для линейных источников питания и их применение. – М.: ДОДЭКА, 1988.

Исида Х. Программирование для микрокомпьютеров: Пер. с японск. – М.: Мир, 1988.

Карпов Г. Эксплуатация, техническое обслуживание и ремонт IBM PC/XT/AT. – Кишинев: Axul-Z, 1992.

Кузьмиов А.Ю. Универсальная система сбора и обработки данных АСИР-3 // Мир ПК. – 1996. – № 6. – С. 40–41.

Кузьминов А.Ю. Удаленные системы сбора информации с датчиков на базе однокристальных микроэвм // Автоматизация и производство. – 1996. – № 3. – С. 7–9.

Кузьминов А.Ю. Однокристальные микроэвм – основа удаленных систем сбора и обработки сигналов, поступающих с датчиков // Электроника и компоненты. – 1998. – № 2. – С.18–19.

Кузьминов А.Ю. Новые MCS51 – совместимые микроконтроллеры и их применение в системах сбора информации с датчиков // Контрольно-измерительные приборы и системы. – 1997. – № 6. – С. 32–35; 1998. – № 7. – С. 34–36.

---

[1] Список представлен тремя разделами: 1 – печатные издания, 2 – информация, взятая с компакт-дисков, и 3 – информация из Интернета. К сожалению, печатные издания, касающиеся новых разработок ведущих фирм-производителей микросхем, устаревает раньше, чем выходит в свет то или иное печатное издание, поэтому в списке печатных изданий фигурируют работы, изданные не в последние годы; аналогичная ситуация происходит и с компакт-дисками. Относительно новая информация представлена на компакт-дисках дистрибьюторов тех или иных фирм в России. Но во-первых, информация, представляемая фирмами-производителями в Интернете, все равно еще новей, а во-вторых, заниматься рекламой компаний-дистрибьюторов мы не вправе. Поэтому список компакт-дисков дистрибьюторов не приводится. При желании список дистрибьюторов тех или иных фирм-производителей можно найти в журнале "Живая электроника России", например, за 2003 г. на с. 86–93.

Что касается информации по языкам программирования как для компьютера (ассемблер, Си, Бейсик, Кларион), так и для микроконтроллеров (ассемблер, С), то в продаже можно встретить компакт-диски типа "Компиляторы-3", "Микроконтроллеры" и т.п. Как правило, производители подобных компакт-дисков предпочитают не афишировать себя, поэтому пользоваться подобными компакт-дисками нужно с осторожностью, чтобы "не нарваться на вирус". В связи с этим подобные компакт-диски в списке литературы также отсутствуют.

Лебедев О.Н. Применение микросхем памяти в электронных устройствах. Справ. пособие. – М.: Радио и связь, 1994.

Петровский И.И., Прибыльский А.В. и др. Логические ИС КР1533, КР1554: Справ. В 2-х ч. – М.: БИНОМ, 1993.

Скэнлон Л. Персональные ЭВМ IBM PC и XT. Программирование на языке ассемблера: Пер. с англ. – М.: Радио и связь, 1991.

Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990.

Тули М. Справочное пособие по цифровой электронике: Пер. с англ. – М.: Энергоатомиздат, 1990.

Фролов А.В., Фролов Г.В. Программирование модемов. – М.: ДИАЛОГ-МИФИ, 1993.

Хоровиц П., Хилл У. Искусство схемотехники: Пер. с англ. В 3-х т. – М.: Мир, 1993.

Шило В.Л. Популярные микросхемы КМОП. Справ. – М.: ЯГУАР, 1988.

Design-in reference mannual. Data converters, Amplifiers, Special linear products, Support components. – Analog Devices Inc., 1994.

New Releases Data Book. – Maxim, 1996. – V.V.

80C51-Based 8-bit Microcontrollers. Data Handbook. Book IC-20. – Philips Semiconductors, 1994.

Soft microcontroller data book. – Dallas semiconductor, 1993.

Memory ICs data book. – UMC, 1994–1995.

Capture compute communicate. Cygnal product catalog. – Cygnal, 2003.

Maxim integrated products. Product selector guide. – Dallas semiconductor, Maxim, 2001.

Short form 2000 designers' guide. – Analog Devices, Inc., 2000.

2. Литература на компакт-дисках

2002 Designers' Referens Manual. – CD, Analog Devices, Inc., Rev. F1, 2000.

Atmel products. – CD, Atmel Corporation, 2000.

VERSA. Product Family Series Documentation. – CD, Goal semiconductor, Inc., 2003.

Cygnal Integrated Products. – CD, Cygnal, 2003.

Linear technology. Product Catalog. – CD, Linear technology Corporation, 1997.

Analog/Mixed-Signal Products. Designer's Master Selection Guide. – CD, Texas Instruments Inc., 2002.

Data on Disk. 2nd Edition. – STMicroelectronics, 1999.

3. Интернет

www.analog.com www.ti.com www.goalsemi.com
www.tools@cygnal.com www.lingear_tech.com
www.maxim-ic.com www.st.com

Оглавление

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ УНИТАРНОЕ ПРЕДПРИЯТИЕ
НАУЧНО-ТЕХНИЧЕСКОЕ ИЗДАТЕЛЬСТВО
«РАДИО и СВЯЗЬ»
127473 Москва, 2-й Щемиловский пер., д. 5/4, стр.1.
Отдел заказов: тел./факс: 978-54-10 / E-mail: radiosv@bk.ru
Редакция: тел./факс: 978-53-51 / E-mail: radsv@garnet.ru
Интернет: www.radiosv.ru