# ТЕХНИЧЕСКИЕ РЕШЕНИЯ ДЛЯ СОПРЯЖЕНИЯ СПЕЦИАЛИЗИРОВАННЫХ ЭВМ С ОБЪЕКТАМИ ВВОДА-ВЫВОДА ЧЕРЕЗ ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС

Ульяновск 2006

Министерство образования и науки Российской Федерации

Федеральное агентство по образованию

Ульяновский государственный техниический университет

# ТЕХНИЧЕСКИЕ РЕШЕНИЯ ДЛЯ СОПРЯЖЕНИЯ СПЕЦИАЛИЗИРОВАННЫХ ЭВМ С ОБЪЕКТАМИ ВВОДА-ВЫВОДА ЧЕРЕЗ ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС

Учебно-методические указания к лабораторным работам

для студентов специальности

«Вычислительные машины, комплексы, системы и сети»

Составитель В. Н. Негода

Ульяновск 2006

УДК 681.3(076)
ББК 32.973-01я7
Т38

Рецензент
кандидат технических наук, доцент, декан ФИСТ В. В. Шишкин

Одобрено секцией методических пособий научно-методического совета университета

Т38 **Технические решения для сопряжения специализированных ЭВМ с объектами ввода-вывода через последовательный интерфейс:** учебно-методические указания к лабораторным работам для студентов специальности «Вычислительные машины, комплексы, системы и сети» / сост. В. Н. Негода. – Ульяновск: УлГТУ, 2006. – 43 с.

Указания написаны в соответствии с рабочей программой дисциплины «Микропроцессорные системы» для студентов пятого курса специальности 230100 «Вычислительные машины, комплексы, системы и сети». Представлены основные технические решения по аппаратным и программным компонентам сопряжения, реализуемым на стороне специализированной ЭВМ, построенной на базе однокристальных микропроцессоров.
Подготовлены на кафедре «Вычислительная техника».

**УДК 681.3(076)**
**ББК 32.973-01я7**

## Оглавление

## Введение

Учебно-инженерные проекты, посвященные сопряжению специализированных ЭВМ с объектами ввода-вывода через последовательный интерфейс, в учебном курсе «Микропроцессорные системы» могут быть предметом задания на лабораторные работы, а также на курсовое проектирование. В первом случае индивидуальное задание предусматривает реализацию частичных проектных решений, во втором – создание и реализацию проекта аппаратно-программной подсистемы с достаточно сложным функционированием. В любом случае проектирование строится, опираясь на учет следующих важнейших особенностей средств сопряжения ЭВМ с объектами контроля и управления автоматизированных систем:

1. В подавляющем большинстве случаев связывание ЭВМ с объектами контроля и управления выполняется на основе трехкомпонентной системы, в центре которой находится контроллер. Два другие компонента представляют собой адаптеры связи: «ЭВМ – контроллер» и «контроллер – объект». Тем самым удается уменьшить сроки проектирования устройства сопряжения и его стоимость за счет применения типовых проектных решений и существующих типовых интерфейсов ввода-вывода ЭВМ.

2. В качестве типовых интерфейсов ЭВМ чаще все используются последовательные интерфейсы, что обеспечивает, во-первых, существенное удешевление средств сопряжения, а во-вторых, возможность удалять ЭВМ от объектов контроля и управления на достаточно большое расстояние. Одним из наиболее используемых последовательных интерфейсов являются интерфейс RS232 и родственные ему интерфейсы RS422, RS423, RS485 [1].

3. В качестве контроллера обычно используется однокристальный микроконтроллер, что существенно уменьшает аппаратные затраты устройства сопряжения ЭВМ с объектом; более того, в микроконтроллере возможно интегрировать функции сопряжения с функциями управления, сбора и предварительной обработки данных.

В настоящих методических указаниях рассматриваются основные технические решения, которые обеспечивают реализацию аппаратной и программной частей устройства сопряжения, а также программы ЭВМ. Эти решения доведены до примеров программ и принципиальных схем аппаратной части. Приводятся базовые сведения справочного характера, в большинстве случаев доста-

точные для учебно-инженерного проектирования устройства сопряжения с заданными параметрами. В этой связи студенту рекомендуется тщательно изучить описываемые в методических указаниях решения, сориентироваться в содержании справочных данных и в соответствии с формулировкой индивидуального задания выполнить компоновку и модификацию проектных решений так, чтобы в результате сформировался целостный и непротиворечивый учебно-инженерный проект.

Ограничения на объем методических указаний позволяют рассмотреть в данном издании только технические решения на стороне ЭВМ. Для изучения технических решений на стороне контроллера можно обратиться к работам, указанным в списке литературы, начиная с позиции 9. При этом целесообразно сопоставить технические решения на стороне контроллера с решениями на стороне ЭВМ. Такой подход позволяет углубить понимание логики функционирования последовательного интерфейса и механизмов программирования обмена данными.

# 1. Базовые технические решения устройства сопряжения компьютера

## 1.1. Общая организация устройства сопряжения

Подключение группы внешних устройств к ЭВМ через встроенный в нее адаптер интерфейса RS232 и микроконтроллерное устройство сопряжения строится чаще всего на основе структурной схемы, приведенной на рис. 1.1.

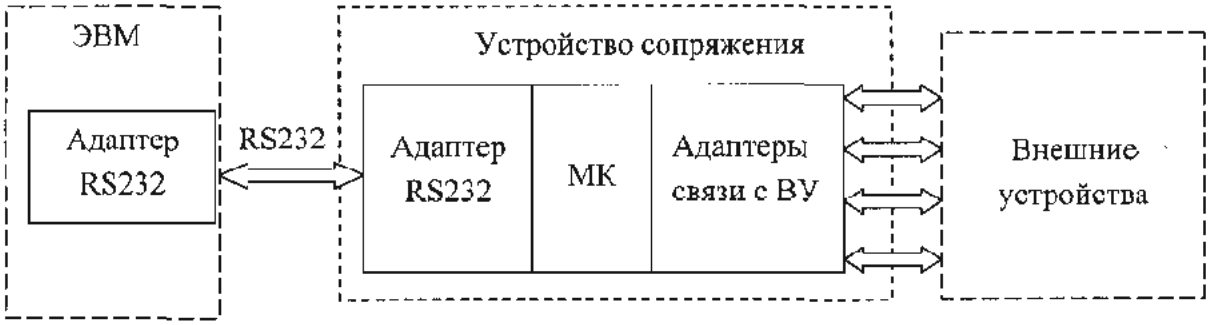



Рис. 1.1. Структурная схема устройства сопряжения

Построение устройства сопряжения по данной схеме является наиболее эффективным способом расширения спектра внешних устройств, подключаемых к ЭВМ. Эффективность определяется следующими основными обстоятельствами:

- в большинстве ЭВМ имеется встроенный адаптер интерфейса RS232; причем именно этот интерфейс чаще всего свободен для подключения новых устройств; если это не так, то приобретение адаптеров RS232 с несколькими каналами ввода-вывода является относительно несложным делом;
- устройство сопряжение может быть удалено от ЭВМ на значительное расстояние – до 15 м при использовании стандартных физических линий RS232, до нескольких километров при использовании модификации физического интерфейса типа «токовая петля»;
- кабели для последовательной передачи данных, используемые в интерфейсе RS232, существенно дешевле кабелей для параллельных интерфейсов;

- протокол RS232 весьма прост и легко может быть реализован программными средствами с использованием двух битов порта ввода-вывода МК и внешнего преобразователя уровней напряжения; однако адаптер последовательной связи, реализующий логику интерфейса RS232, встроен во многие недорогие МК, что существенно упрощает реализацию адаптера RS232 на стороне устройства сопряжения;
- для достаточно простых объектов ввода-вывода микроконтроллер может играть роль устройства местного управления, сбора и предварительной обработки данных; к таким устройствам относятся, прежде всего, датчики, шаговые двигатели, специализированные индикаторы и индикационные панели;
- многие МК допускают возможность изменения содержимого программной памяти «на борту», т. е. без извлечения из печатной платы; это позволяет легко адаптировать логику функционирования устройства сопряжения под изменяющиеся условия на стороне внешних устройств.

### 1.2. Логическое описание интерфейса RS232

Протокол последовательного асинхронного обмена RS232 является наиболее известным и распространенным среди стандартов для последовательной передачи данных. Стандарт этого интерфейса был введен в 1969 г. ассоциацией электронной промышленности (Electronics Industries Alliance, EIA) как «рекомендуемый стандарт» (RS – Recommended Standard) для описания требований к интерфейсу между оборудованием обработки данных (ООД, Data Terminal Equipment – DTE) и аппаратурой окончания канала данных (АКД, Data Communication Equipment - DCE) [2]. В 1987 году название стандарта было изменено на EIA-232-D, однако широкая распространенность самого интерфейса настолько закрепила его исходное наименование, что формально действующее уже более 15 лет название стандарта в технической литературе используется очень редко. На основе этого стандарта свои собственные спецификации интерфейса были разработаны международными организациями стандартизации МККТ (стандарт МККТ V.24/V.28) и ISO, а также организациями стандартизации многих стран (в России это стандарт «ГОСТ 18145–81. Стык С2»).

Протокол RS232 широко применяется для подключения к компьютеру алфавитно-цифровых терминалов, низкоскоростных печатающих устройств, позиционных устройств ввода (мышей, планшетов), низкоскоростного телекоммуникационного оборудования, прежде всего модемов, и т. д., а иногда и для соединения компьютеров между собой, например, если более скоростное сете-

вое оборудование отсутствует или не может быть использовано. В системах управления этот протокол часто используется для связи между контроллерами в распределенной микропроцессорной системе управления, а также для подключения интегральных датчиков. В большинстве случаев используется некоторое подмножество линий и функциональных возможностей RS232, охватывающих только асинхронный последовательный обмен данными. Полная версия стандарта определяет также синхронную передачу данных. В настоящих методических материалах будут рассматриваться только решения для асинхронного обмена. Описание логики функционирования и механизмов программирования режимов синхронного последовательного обмена можно найти в работах [1, 3].

Обмен данными осуществляется кадрами, состоящими из стартового бита, от пяти до восьми битов данных (младший бит передается первым), возможно – контрольного бита чстности, и одного или двух стоповых битов. Игнорируя ошибки четности или вообще не проверяя четность, можно использовать этот бит для передачи данных и получить, таким образом, девять битов данных в одном кадре. Наличие стартового и стоповых битов в кадре позволяет передавать данные без отдельных линий синхронизации. Для этого достаточно иметь различными уровни стартового и стоповых битов. Стартовый уровень по спецификации RS232 совпадает с уровнем логического нуля, а стоповый – с уровнем логической единицы. Уровень пассивного состояния линии, т.е. состояния, когда в линии нет никаких сигналов, естественно должен отличаться от уровня стартового бита. При этом приемник данных способен однозначно определить начало кадра. Обнаружив стартовый бит, приемник отсчитывает середину бит-интервала и далее после отсчета каждого бит-интервала читает и сдвигает во входной регистр сигнал с линии.

Процесс передачи данных по физической линии представлен на рис. 1.2.

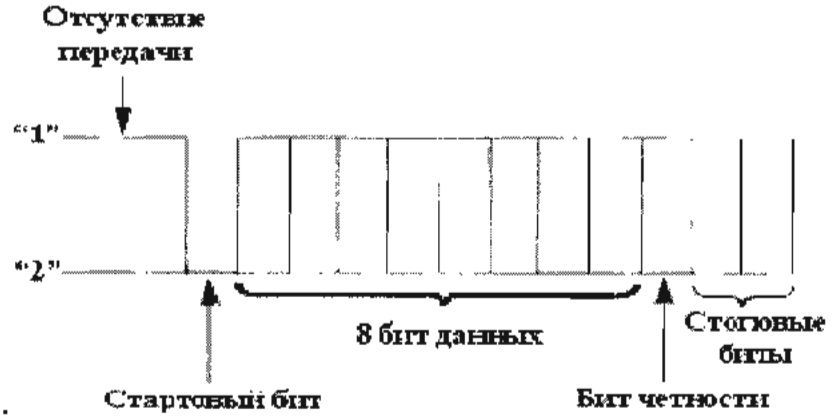



Рис. 1.2. Процесс передачи данных по линии RS232

В табл. 1.1 представлено несколько кадров, содержащих символ '5', кодируемый в семиразрядном коде ASCII значением 0110101 (информационные биты выделены полужирным шрифтом). Младший разряд байта передается сразу после стартового бата, а семиразрядный код при размере кадра в 8 информационных разрядов дополняется нулем.

Таблица 1.1

Форматы кадров, передающих код символа '5'

| № п/п | Содержимое кадра | Формат кадра |
|---|---|---|
| 1 | 010101101 | 7 битов данных, 1 стоповый бит, нет контроля |
| 2 | 0101011011 | 7 битов данных, 2 стоповых бита, нет контроля |
| 3 | 01010110011 | 7 битов данных, 2 стоповых бита, контроль четности |
| 4 | 01010110111 | 7 битов данных, 2 стоповых бита, контроль нечетности |
| 5 | 01010110011 | 8 битов данных, 2 стоповых бита, нет контроля |
| 6 | 01010110011 | 8 битов данных, 1 стоповый бит, контроль нечетности |
| 7 | 010101100011 | 8 битов данных, 2 стоповых бита, контроль четности |

Поскольку байты или слова данных обычно представлены на шине микропроцессора в параллельной форме, их последовательный ввод–вывод требует преобразования параллельных данных в последовательные и наоборот. Существуют специальные микросхемы ввода и вывода, решающие проблемы преобразования. Вот список наиболее типичных сигналов таких микросхем:

*D0–D7* — входные–выходные линии данных, подключаемые непосредственно к шине процессора;

*RXD* — принимаемые данные (входные последовательные данные);

*TXD* — передаваемые данные (выходные последовательные данные);

*CTS* — сброс передачи. На этой линии периферийное устройство формирует сигнал низкого уровня, когда оно готово воспринимать информацию от процессора;

*RTS — запрос передачи. На эту линию микропроцессорная система выдает* сигнал низкого уровня, когда она намерена передавать данные в периферийное устройство.

Ясно, что приемник должен использовать одинаковый формат кадра и значение бит-интервала с передатчиком. Бит-интервал определяется частотой следования сигналов, которая, в свою очередь, определяет скорость передачи данных. Скорость для цифровых линий передачи на основе RS232 принято измерять в бодах (baud). Бод учитывает количество всех двоичных сигналов, в том числе служебных, передаваемых за секунду. Когда используют единицу измерения «бит/с», то за этим может скрываться как число бодов, так и количество только информационных битов, передаваемых за секунду, без учета стартовых, контрольных и стоповых битов.

Скорость COM-портов ПЭВМ может выбираться из следующего набора значений (бод): 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200, 38 400, 57 600, 115 200. Современные реализации RS232 в компьютерах могут поддерживать даже более высокие скорости.

Внутренний процесс адаптера асинхронной последовательной связи несколько сложнее представленного на рис. 1.1. Дело в том, что отслеживание середины бит-интервала выполняется с использованием внутреннего счетчика, называемого предделителем и имеющего частоту сигналов счета обычно в 16, а иногда в 8 раз больше, чем частота сигналов линии связи. Это значит, что на один бит-интервал приходится 16 или 8 микротактов внутреннего генератора отсчетов. В то же время, частота этого внутреннего генератора определяется частотой внешнего генератора и настройкой программируемого делителя. Именно благодаря наличию этого программируемого делителя удается задавать различные частоты передачи данных. Таким образом, схема программируемого адаптера связи обычно содержит два делителя: а) делитель с программно устанавливаемым коэффициентом деления $k$; б) предделитель для формирования импульсов отсчета либо с жестким коэффициентом деления $q$, обычно равным 16, либо возможно переключаемым коэффициентом деления с помощью одного управляющего бита между значениями 8 и 16.

Описанная выше логика внутреннего функционирования адаптера асинхронного последовательного интерфейса дает нам следующее выражение для определения частоты следования сигналов *BAUD* (бод) в линии связи:

$$BAUD = F / (q*k),$$

где $F$ – частота задающего генератора. Длительность сигнала в линии выражается при этом формулой $T_{BAUD} = 1 / BAUD = q*k / F$, а длительность такта генератора отсчетов – формулой $T_C = T_{BAUD} / q = k / F$.

Наличие внутреннего генератора отсчетов, кроме функции позиционирования на середину бит-интервала, обеспечивает также функции фильтрации от помех. Для этого функционально развитый адаптер последовательной связи отслеживает обычно 3 отсчета, начиная от середины бит-интервала (8-й, 9-й, 10-й при $q$=16, либо 4-й, 5-й, 6-й при $q$=8), и сравнивает между собой значения всех сигналов на входной линии во время этих отсчетов. В качестве логического значения берется то, которое в двух отсчетах совпадает. Если в процессе отслеживания стартового бита в ходе фильтрации обнаруживается логическая единица, то фиксируется ошибка обнаружения стартового бита.

По мере продвижения к концу кадра момент чтения сигнала с линии все с меньшей точностью попадает в центр бит-интервала. Это связано со следующими основными обстоятельствами.

Во-первых, реальные частоты следования сигналов передатчика $BAUD_{TX}$ и приемника $BAUD_{RX}$ могут не совпадать. На стороне передатчика мы имеем набор частоты задающего генератора $F_{TX}$ и коэффициентов деления $q_{TX}$ и $k_{TX}$, а на стороне приемника – набор соответственно $F_{RX}$, $q_{RX}$ и $k_{RX}$. Коэффициенты деления всегда являются целыми числами, поэтому деление частоты вполне определенного кварцевого генератора на различные целые числа дает ряд значений скоростей передачи, немного отличающийся от указанных выше нормированных значений. Хуже всего, когда частное $F_{TX}/(q_{TX}*k_{TX})$ для передатчика *TX* отличается от требуемой скорости $BAUD_{TX}$ в одну строну, а частнос $F_{RX}/(q_{RX}*k_{RX})$ для присмника *RX* – в другую. Поскольку длительность бит-интервала равна обратной величине от частоты *BAUD*, т. е. частному q*$k/F$, на *каждом бит-интервале погрешность* $\tau_1$ *на одном бит-интервале равна*

$$\tau_1 = | q_{TX}*k_{TX}/F_{TX} - q_{RX}*k_{RX}/F_{RX} |.$$

Общая погрешность $\tau_\Sigma$ будет вычисляться согласно выражению:

$$\tau_\Sigma = (n+0,5)* \tau_1,$$

где n – длина пакета с учетом контрольного бита, а константа 0,5 учитывает ошибку позиционирования в центр стартового бита.

Во-вторых, момент обнаружения стартового бита фиксируется в приемнике по фронту внутреннего генератора, а этот фронт может пройти чуть раньше появления стартового бита, в результате чего почти целый такт внутреннего генератора отсчетов стартовый бит считается необнаруженным. В итоге мы имеем еще одну погрешность позиционирования в центр бит-интервала. Эта погрешность имеет место только на стороне приемника и не накапливается по мере прохождения сигналов кадра. Ее максимально возможное значение меньше длительности интервала генератора отсчетов $T_C = T_{BAUD} / q = k / F$.

Во-третьих, даже при одинаковых расчетных значениях частот кварцевых резонаторов для передатчика и приемника реальные значения этих частот могут различаться как из-за ограниченной точности обеспечения параметров резонатора во время изготовления, так и из-за влияния условий эксплуатации.

В-четвертых, передача сигнала по линии сопровождается фазовыми искажениями, что проявляется в некотором временном сдвиге фронтов сигналов относительно расчетных моментов времени.

Указанные обстоятельства не должны приводить к тому, чтобы к концу приема кадра ошибка позиционирования момента чтения в середину бит-интервала составила более половины этого интервала $T_{BAUD}/2$. Учет последних двух обстоятельств затруднен, поэтому логично использовать для оценки максимума абсолютной погрешности величину $\tau_\Sigma + T_C$. Разность между длительностью половины бит-интервала и этой оценкой погрешности ($T_{BAUD}/2 - \tau_\Sigma - T_C$) образует определенный запас устойчивости, который обеспечивает правильное чтение даже при наличии фазовых искажений и отклонений реальных параметров кварцевых генераторов от паспортных.

В случае, когда интерфейс строится в расчете на подключение с разными скоростями и с различными устройствами, то погрешности оценивают через относительную погрешность частоты сигналов последовательного канала *BAUD*. При этом значение частоты, получаемое через отношение $F/(q*k)$, соотносят с целевой (требуемой) частотой $BAUD_T$. Так формируется значение относительной погрешности частоты сигналов интерфейса в процентах:

$$E_R = 100* |BAUD - BAUD_T| \, BAUD_T \ \% .$$

В технической документации для разработчиков встроенных систем фигурирует обычно ограничение относительной погрешности в 0,5%, либо чуть более слабое ограничение – 1%.

Пример 1.1.

Пусть целевая частота $BAUD_T$ = 9600 бод, имеется в наличии кварцевый генератор в 1 МГц и адаптер с предделителем, характеризующимся коэффициентом $q$ = 16. Ближайшим из значений для коэффициента деления будет k = 7 и пусть адаптер способен обеспечить такой коэффициент деления. При этом $BAUD$ = 1 000 000/(7*16) = 8928,57. Относительная погрешность получается (9600 - 8928,57) / 9600 = +7%. Эта погрешность в общем случае исключает правильное функционирование интерфейса. Если на другой стороне будет идеальная реализация интерфейса с нулевой погрешностью, то к восьмому разряду информационного байта накопленная ошибка составит:

$$\tau_\Sigma = 8 * | 1/9600 - 7*16/1000000)| = 8* |(104.17 - 112)*10^{-6}| = 62{,}64 \text{ мкс}$$

Полученная величина больше половины бит-интервала, равной 52 мкс. Однако если на другой стороне линии связи мы будем гарантированно иметь такой же набор параметров ($F$–1МГц, $q$ = 16, $k$ – 7), то связь будет надежной.

Адаптеры могут иметь два раздельных входа синхронизации: вход общего тактового генератора с частотой $F$ и вход синхронизации приема-передачи с частотой $F_P$, который может быть обработан программируемым делителем с коэффициентом $k$. В этом случае $BAUD = F_P / k$ и $q = F / F_P$.

Набор параметров ($F$, $q$, $k$) часто приходится подбирать из нескольких возможных вариантов, чтобы обеспечить заданную частоту BAUD и не выйти за пределы допустимой погрешности.

### 1.3. Физические сигналы, линии и разъемы интерфейса RS232

На физическом уровне порты RS232 используются в ЭВМ для передачи данных за пределы корпуса компьютера, поэтому, кроме линий передачи данных, предусмотрен также провод, передающий опорное нулевое напряжение. Вместо TTL-совместимых напряжений RS232 использует в качестве логической единицы напряжения в диапазоне от -25 до -3 В, а в качестве нуля – в диапазоне от +3 до +25 В (рис. 1.3). Это обеспечивает более высокую помехоустойчивость канала передачи данных по отношению к TTL. В то же время специальные микросхемы адаптеров последовательного интерфейса, а также встроенные в МК адаптеры по внешним выходам совместимы с TTL, что позволяет обеспечить работу таких микросхем при одном напряжении питания – +5 В. Это значит, что между физическим каналом RS232 и линиями передачи данных этих микросхем должен быть блок преобразования уровней. При построении преобразователей уровня используются два вида технических решений: преобразователи на дискретных компонентах и преобразователи на специальных микросхемах.

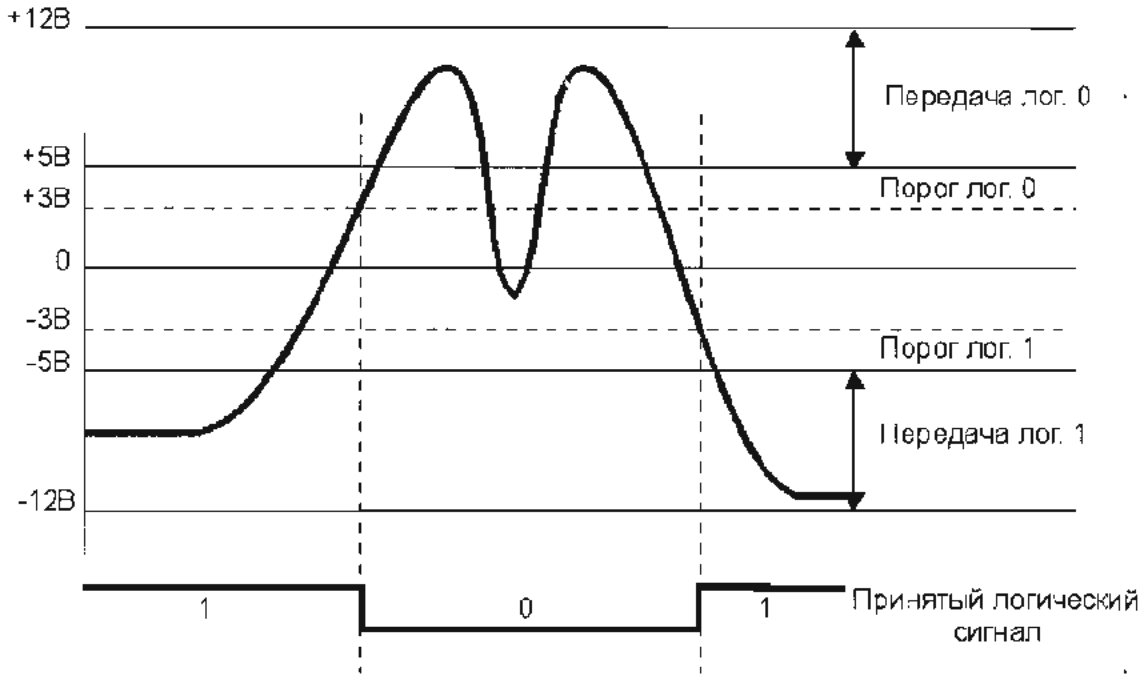



Рис. 1.3. Вид сигнала в линии интерфейса RS232

Преобразователь TTL-сигналов в уровни стандарта RS232 используется в узле передачи данных. Он должен «растягивать» интервал амплитуд (0 – 5) вольт в интервал (-12 – +12 ) вольт. Одна из возможных схем представлена на рис. 1.4.

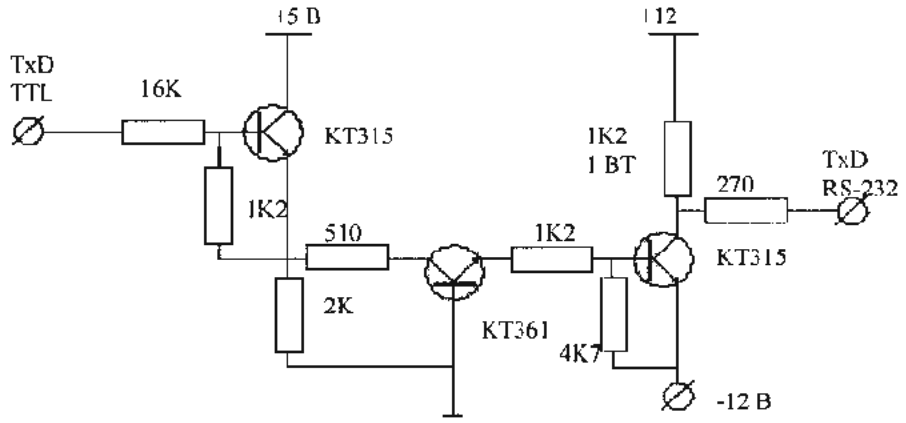



Рис. 1.4. Преобразователь уровня передатчика на дискретных компонентах

Преобразователь сигналов стандарта RS232 в TTL-сигналы легче всего строится на основе механизма воспроизведения переключательного процесса, представляемого последовательностью сигналов на входной линии RS232, с помощью цифровых TTL-элементов, например, по схеме, представленной на

рис. 1.5. При этом, кроме собственно преобразования уровней, устройство «поправляет» фронты сигналов, «заваленные» в ходе передачи через линию, импеданс которой обладает значительной емкостной составляющей.

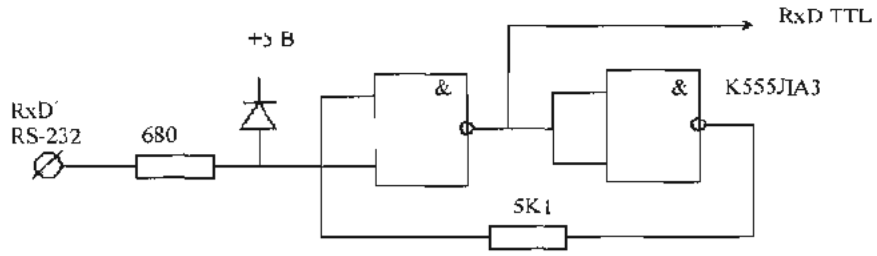



Рис. 1.5. Преобразователь уровня приемника на дискретных компонентах

Технические решения на основе специальных микросхем преобразования уровней сигналов чаще всего базируются на интерфейсных микросхемах фирмы MAXIM. Наиболее важной для разработчика особенностью таких микросхем является наличие в них встроенного источника питания, что позволяет строить двунаправленные преобразователи всего на одной микросхеме с одним номиналом питания. На рис. 1.6 представлена микросхема преобразователя уровней MAX3160 фирмы MAXIM. Преобразователь может быть включен в двух режимах: «Преобразование TTL – RS232 » и «Преобразование TTL – RS485».

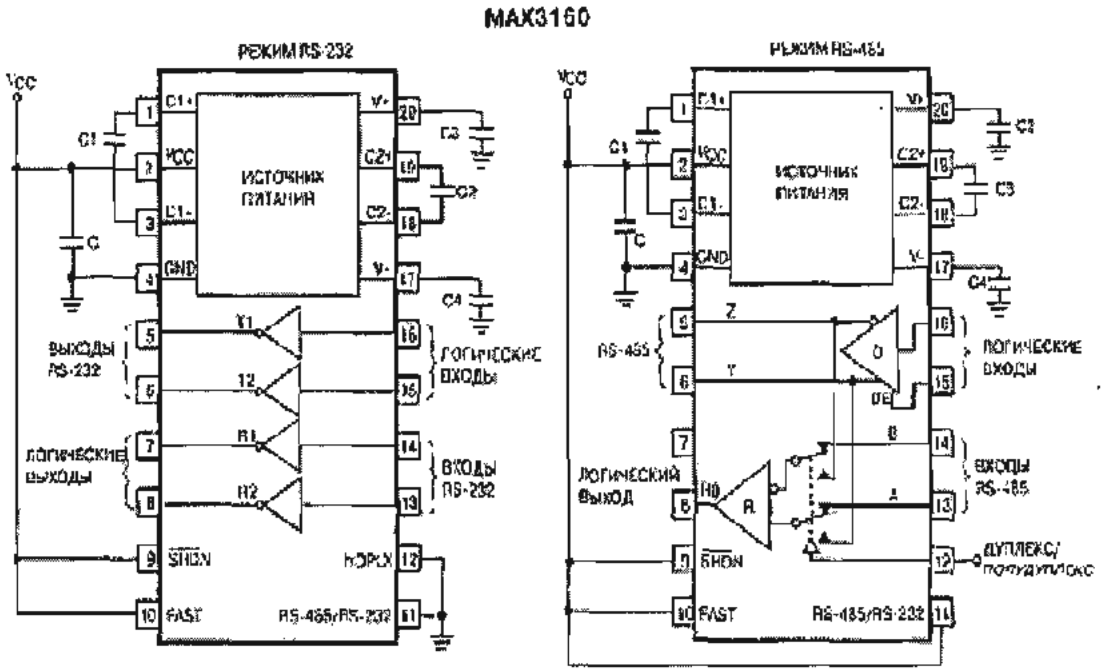
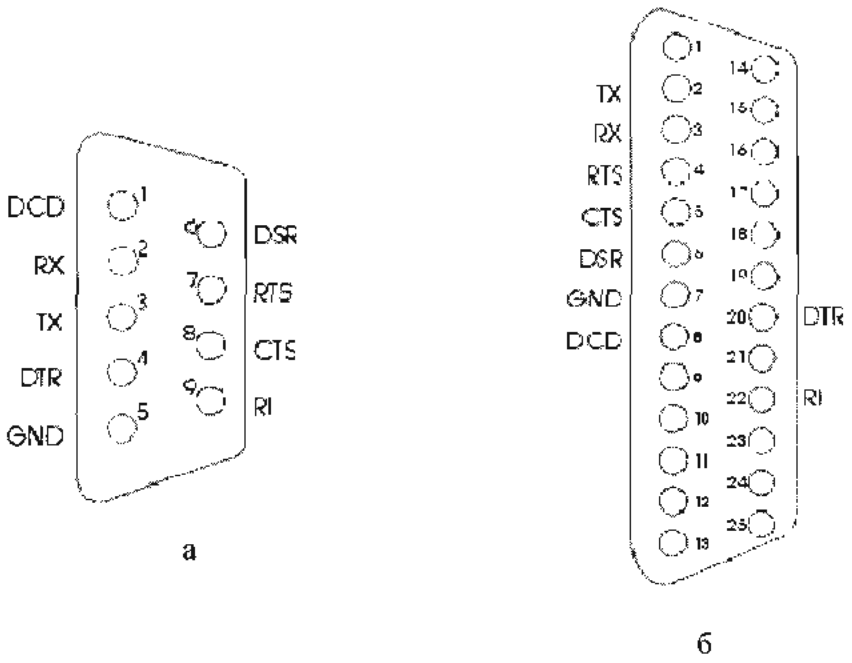



Рис. 1.6. Интерфейсная микросхема MAX3160

Кроме линий приема, передачи, нуля и питания, спецификация RS232 предусматривает ряд дополнительных линий, в просторечии называемых модемными – признак несущей, разрешение передачи данных (очисткой этого сигнала приемник может сигнализировать передатчику, что он не успевает обрабатывать поступающие данные) и др. Сигналы этих линий не должны обязательно поддерживаться всеми устройствами и используются, главным образом, акустическими модемами, откуда и происходит название. Полная спецификация при использовании 25-контактного разъема предусматривает также возможность синхронной передачи данных с отдельными стробирующими сигналами, но основная масса реализаций RS232 этого не поддерживает.

Стандартом интерфейса RS232 предусмотрено два вида разъемов: 25-контактный и 9-контактный (рис. 1.7). Терминальное оборудование, т. е. оборудование обработки данных, например ПЭВМ, обычно оснащено разъемом со штырьками («вилка»), а связное – разъемом с отверстиями («розетка»), но могут быть и исключения.




а б

Рис. 1.7. Два вида разъемов: а) DB9P; б) DB25P

Интерфейс RS232 соединяет два устройства. Линия передачи первого устройства соединяется с линией приема второго и наоборот (полный дуплекс). Для управления соединенными устройствами используется программное подтверждение (введение в поток передаваемых данных соответствующих управ-

ляющих символов). Возможна организация аппаратного подтверждения путем организации дополнительных RS232 линий для обеспечения функций определения статуса и управления.

Сигналы интерфейса RS232 подразделяются на следующие классы.

*Последовательные данные* (например, *TxD, RxD*). Интерфейс RS232 обеспечивает два независимых последовательных канала данных: первичный (главный) и вторичный (вспомогательный). Оба канала могут работать в дуплексном режиме, т. е. одновременно осуществляют передачу и прием информации.

*Управляющие сигналы квитирования* (например, *RTS, CTS*). Сигналы квитирования — средство, с помощью которого обмен сигналами позволяет *DTE* начать диалог с *DCE* до фактической передачи или приема данных по последовательной линии связи.

*Сигналы синхронизации* (например, *TC, RC*). В синхронном режиме (в отличие от более распространенного асинхронного) между устройствами необходимо передавать сигналы синхронизации, которые упрощают синхронизм принимаемого сигнала в целях его декодирования.

На практике вспомогательный канал RS232 применяется редко, и в асинхронном режиме вместо 25 линий используются 9 линий. Номера контактов, наименование и название всех 25 линий приведены в табл. 1.2. Номера контактов 9-контактного разъема приведены в первом столбце таблицы в скобках.

Существует несколько схем соединения устройств через интерфейс RS232. Наиболее часто в режиме асинхронной передачи данных используют всего 4 линии: *TxD, RxD, FG, SG*. Соответствующая схема представлена на рис. 1.8.

Для интерфейса RS232 принято различать два вида оборудования: терминальное и связное. Терминальное оборудование, например микрокомпьютер, может посылать и/или принимать данные по последовательному интерфейсу. Оно как бы оканчивает (terminate) последовательную линию. Связное оборудование — устройства, которые могут упростить передачу данных совместно с терминальным оборудованием. Наглядным примером связного оборудования служит модем (модулятор–демодулятор). Он оказывается соединительным звеном в последовательной цепочке между компьютером и телефонной линией. Другим примером является интегральный датчик, включающий в себя аналого-цифровой преобразователь и адаптер последовательного интерфейса.

Таблица 1.2.

Функции сигнальных линий интерфейса RS232

| Номер контакта | Сокращение | Направление | Полное название |
|---|---|---|---|
| *1* | *FG* | — | Основная или защитная земля |
| *2(3)* | *TD (TxD)* | К DCE | Передаваемые данные |
| *3(2)* | *RD (RxD)* | К DTE | Принимаемые данные |
| *4(7)* | *RTS* | К DCE | Запрос передачи |
| *5(8)* | *CTS* | К DTE | Сброс передачи |
| *6(6)* | *DSR* | К DTE | Готовность модема |
| *7(5)* | *SG* | — | Сигнальная земля |
| *8(1)* | *DCD* | К DTE | Обнаружение несущей данных |
| *9* | — | К DTE | (Положительное контрольное напряжение) |
| *10* | | К DTE | (Отрицательное контрольнос напряжение) |
| *11* | *QM* | К DTE | Режим выравнивания |
| *12* | *SDCD* | К DTE | Обнаружение несущей вторичных данных |
| *13* | *SCTS* | К DTE | Вторичный сброс передачи |
| *14* | *STD* | К DCE | Вторичные передаваемые данные |
| *15* | *TC* | К DTE | Синхронизация передатчика |
| *16* | *SRD* | К DTE | Вторичные принимаемые данные |
| *17* | *RC* | К DTE | Синхронизация приемника |
| *18* | *DCR* | К DCE | Разделенная синхронизация приемника |
| *19* | *SRTS* | К DCE | Вторичный запрос передачи |
| *20(4)* | *DTR* | К DCE | Готовность терминала |
| *21* | *SQ* | К DTE | Качество сигнала |
| *22(9)* | *RI* | К DTE | Индикатор звонка |
| *23* | — | К DCE | (Селектор скорости данных) |
| *24* | *TC* | К DCE | Внешняя синхронизация передатчика |
| *25* | — | К DCE | (Занятость) |

Примечания:

1. Линии *11, 18, 25* обычно считают незаземленными. Приведенная в таблице спецификация относится к спецификациям Bell 113B и 208A.
2. Линии *9* и *10* используются для контроля отрицательного *(MARK)* и положительного *(SPACE)* уровней напряжения.
3. Во избежание путаницы между *RD* (*Read* — считывать) и *RD* (*Received Data* — принимаемые данные) будут использоваться обозначения *RxD* и *TxD*, а не *RD* и *TD*.

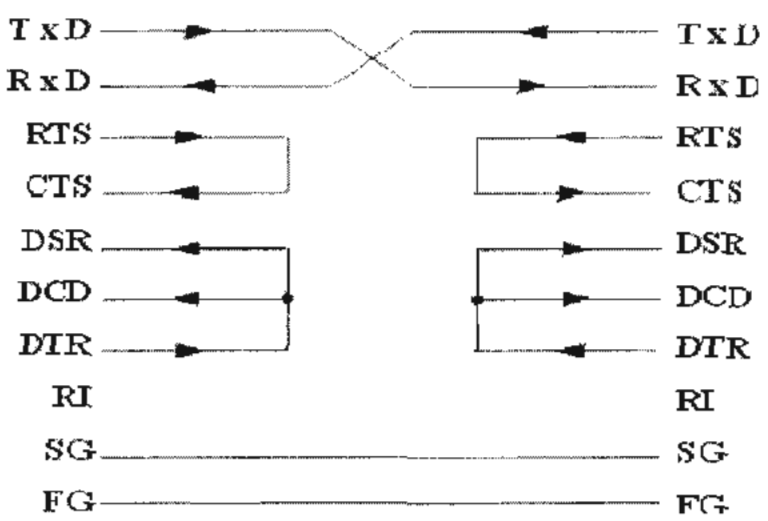



Рис.1.8. Схема 4-проводной линии связи для RS232

Различие между терминальными и связными устройствами довольно расплывчато, поэтому возникают некоторые сложности в понимании того, к какому типу оборудования относится то или иное устройство. Рассмотрим ситуацию с принтером. К какому оборудованию его отнести? Как связать два компьютера, когда они оба действуют как терминальное оборудование. Для ответа на эти вопросы следует рассмотреть физическое соединение устройств. Произведя незначительные изменения в линиях интерфейса RS232, можно заставить связное оборудование функционировать как терминальное.

Спецификации RS232 не ограничивают максимальную длину кабеля, но ограничивают максимальное значение его емкости 2500 пФ. Емкость интерфейсных кабелей различна, однако общепринятой длиной, удовлетворяющей данной спецификации, считается длина 15 м (до 20 000 бод). Чем выше скорость передачи, тем больше искажения сигнала, вызванные емкостными характеристиками кабеля.

Выпускаются специальные интерфейсные кабели прямой связи RS-232-C низкой емкости, которые удовлетворительно работают со скоростью 9600 бод на расстоянии до 150 м. Число подключаемых к одной линии приемников и передатчиков – 1/1, (в отличие от стандартов RS422 – 1 передатчик/ 10 приемников или RS485 – 32/32).

Таким образом, получившие сейчас распространение соединения на скорости 115 килобод выходят за стандарт RS232-C. Это означает, возможности адаптеров последовательной связи поддерживать высокие скорости существенно ограничены параметрами физических линий связи.

В технической литературе можно найти предельные длины физических линий для различных частот передачи (табл. 1.3). Эти данные имеют место при размахе выходного напряжения ±15 В.

Таблица 1.3.

Максимальная длина кабеля интерфейса RS232

| Скорость передачи, бод | Длина экранированного кабеля, м | Длина неэкранированного кабеля, м |
|---|---|---|
| 110 | 1524 | 914 |
| 300 | 1524 | 914 |
| 1200 | 914 | 914 |
| 2400 | 304 | 152 |
| 4800 | 304 | 76 |
| 9600 | 76 | 76 |

Если проэкстраполировать эти данные на более высокие скорости, то для скорости 115 килобод получается величина меньше 5 м. При уменьшении размаха напряжений RS232 допустимая длина кабеля уменьшается. В технической литературе можно найти информацию, согласно которой при размахе ±5В на скоростях до 9600 бод длина кабеля не должна превышать 5 м. Для увеличения дальности соединения необходимо либо использовать интерфейс RS485 (до 1200 метров на всех скоростях), либо использовать усилитель HUB-RS232, который позволяет удлинить линию после него до 70 м.

# 2. Адаптеры асинхронного последовательного интерфейса ЭВМ и их программирование

Адаптеры последовательного интерфейса современных ЭВМ являются частью больших интегральных схем, обслуживающих сопряжение машины с внешней средой. В ПЭВМ с архитектурой IBM PC адаптер последовательного интерфейса встроен в чипсет. В то же время на рынке электронных компонентов активно продаются специальные микросхемы для поддержки создания последовательных интерфейсов в специализированных микропроцессорных системах. Логика управления этими отдельными микросхемами во многом совпадает с логикой встроенных в чипсеты адаптеров.

## 2.1. Адаптер 8251

На рис. 2.1 представлен типичный микрокомпьютерный интерфейс RS232 на базе ИМС 8251 (российский аналог К580ВВ51/К580ИК51), рассмотренный в работе [4]. Программируемая микросхема 8251 представляет собой универсальный синхронно-асинхронный приемо-передатчик (УСАПП: USART – Universal Synchronous-Asynchronous Receiver/Transmitter). Она осуществляет параллельно-последовательные и последовательно-параллельные преобразования данных.

Справа от микросхемы 8251 на этом рисунке расположены преобразователи уровня, выполненные на дискретных компонентах с использованием микросхемы 7406 (К155ЛП3) и части микросхемы 74LS04, используемой в задающем генераторе. Слева изображены компоненты, связывающие адаптер с линиями системного интерфейса Multibus (И41 в России).

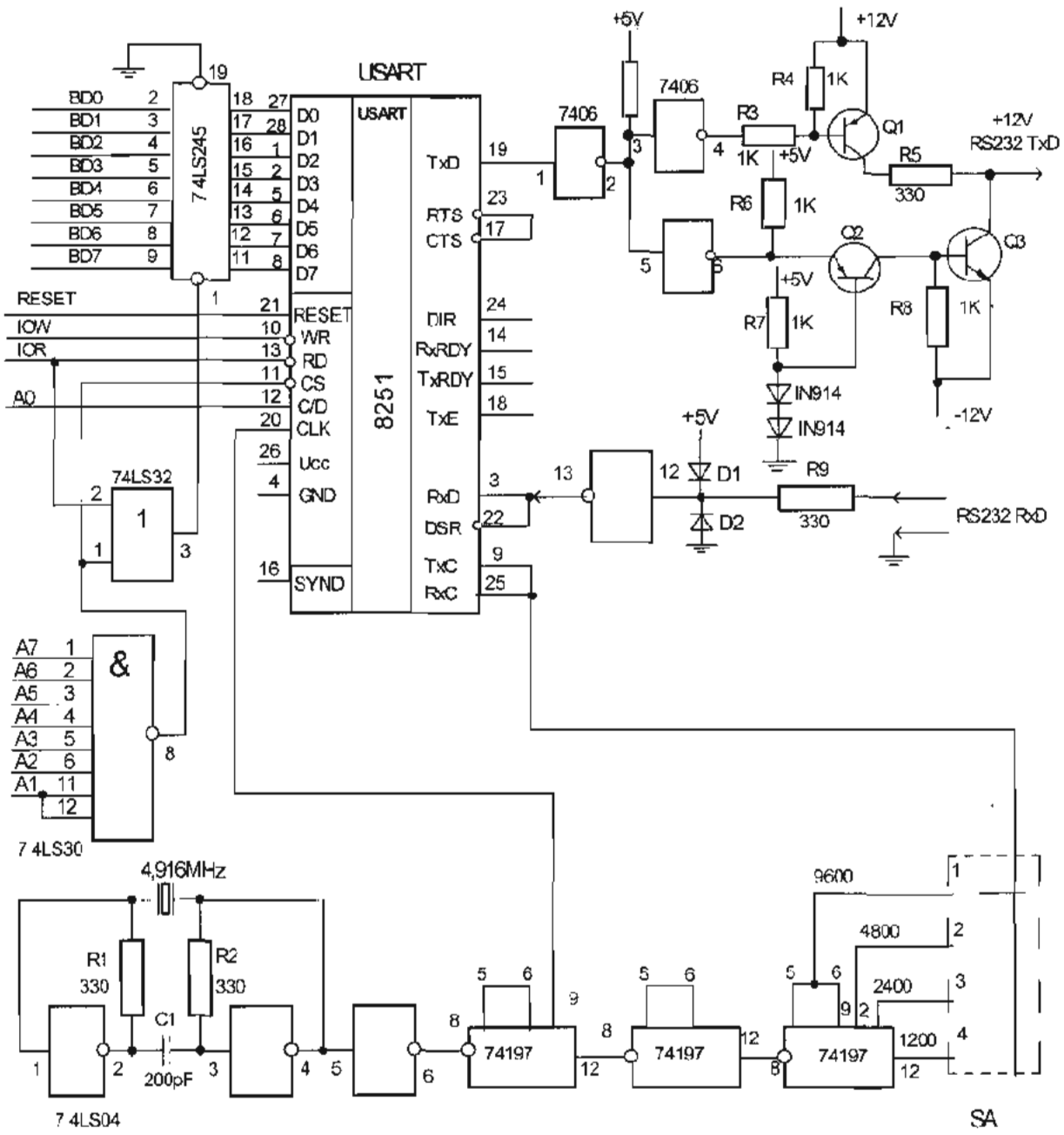



Рис. 2.1. Организация адаптера последовательного интерфейса на ИМС 8251

Обращение к адаптеру организуется по адресам 0FEh и 0FFh, благодаря чему в качестве дешифратора адреса используется всего лишь одна 7-входовая схема И-НЕ (74LS30, российский аналог К555ЛА2), обрабатывающая разряды байта адреса порта A1-A7. Выход дешифратора адреса поступает на вход CS (Crystal Select – выбор кристалла) ИМС 8251, обеспечивая ее стробирование в случае обращения программы МП к портам 0FEh и 0FFh. Младший разряд адресной шины A0 поступает на вход C/D, что позволяет логике управления ИМС 8251 различать обращения к регистрам данных и регистрам состояния-управления. Линии байта данных D0-D7 отделяются от системной шины дан-

ных BD0-BD7 двунаправленным приемо-передатчиком 74LS245 (К555АП6), направление передачи которого задается сигналом управления чтением IOR в случае, когда дешифратор адреса выделяет обращение к адаптеру (четвертая часть ИМС 74LS32 / К555ЛЛ1). Сигналы управления чтением-записью RD и WR ИМС 8251 подключаются стандартным образом к линиям IOR и IOW интерфейса MultiBus. Если адаптер включить не в адресное пространство портов ввода-вывода, а в адресное пространство памяти, то линии RD и WR адаптера необходимо подключить к линиям системного интерфейса MEMR и MEMW, а дешифратор адреса, вырабатывающий сигнал CS, должен обрабатывать 15 старших разрядов адресной шины, т. е. сигналы A1-A15.

Синхронизация внутренних процессов ИМС 8251 задается двумя внешними сигналами: CLK – это сигнал общей синхронизации (частота $F$ в рассмотренных выше расчетных формулах), а TxC и RxC – сигналы внешних генераторов отсчетов для передатчика и приемника соответственно (частота $F_P$). Сигналы внешних генераторов могут внутри ИМС делиться на 16 или 64 (коэффициент $k$ в расчетных формулах). Если $k = 1$, то требуется выполнить соотношение: $F/F_P >= 30$. Если $k > 1$, то соотношение меняется: $F/F_P >= 5$.

В нижней части рис. 2.1 представлен генератор, образованный задающим генератором на базе ИМС 74LS04 (К555ЛН1) и двухкаскадным делителем частоты на базе ИМС 74197 (К555ИЕ15). Первый каскад делителя формирует вход CLK, а второй – входы TxC и RxC. Второй каскад содержит набор перемычек SA, позволяющих аппаратно задавать различные скорости передачи.

В адресном пространстве портов ввода-вывода для этого адаптера выделяются всего два порта: при A0 = 0 адресуются внутренние входной и выходной параллельные регистры данных. Выбор одного из них выполняется значениями сигналов RD (чтение, активный низкий уровень RD) и WR (запись, активный низкий уровень WR). При A0 = 1 происходит либо загрузка управляющих данных в адаптер, либо чтение его слова состояния в микропроцессор. На самом деле в адаптере имеются два управляющих регистра, содержимое которых определяет функционирование. Самый первый байт, загружаемый по адресу A0=1 после сигнала сброса RES (вход 21 ИМС 8251), попадает в так называемый регистр режима. Разряды этого байта при асинхронном обмене задают следующее управление:

(B1,B0): программирование делителя частоты – 01: $k$ =1, 10: $k$=16, 11: $k$=64;

(B3,B2): длина символа – 00: $n$=5, 01: $n$=6, 10: $n$=7, 11: $n$=8;

(B5,B4): управление контролем – 00,10: нет, 01: нечетность, 11: четность;

(B7,B6): число стоповых бит; 01 – 1, 10 – 1.5, 11 – 2.

Любой следующий байт по адресу A0=1 будет загружаться в регистр команды. Формат этого регистра таков:

B0: разрешение передачи;

B1: запрос флага готовности передатчика;

B2: разрешение приема;

B3: признак сбоя передачи символа; 0 – норма, 1 – сбой;

B4 – сброс ошибки; 1 – сброс;

B5 – запрос передачи; 1 – на выходе RTS логический 0;

B6 – внутренний сброс; 1 – сбросить i8251 в исходное состояние (аналогично RES);

B7 – в асинхронном режиме должно быть равным 0.

При чтении регистра состояния по адресу A0=1 в микропроцессор поступает байт следующего формата:

B0 – готовность передатчика;

B1 – готовность приемника;

B2 – буфер передатчика пуст;

B3 – флаг ошибки паритета;

B4 – флаг ошибки переполнения;

B5 – флаг ошибки стопового сигнала;

B6, B7 – в асинхронном режиме не используются.

Программирование обмена данными предполагает, что микросхема адаптера уже настроена на определенный режим путем вывода в регистр режима соответствующего байта. Это делается в фазе инициализации адаптера и такое действие называется программированием режима. Следует различать два случая: однорежимное и многорежимное использование адаптера.

Однорежимное использование обычно используется в узко специализированных вычислительных машинах. Фаза инициализации адаптера размещается в программе инициализации всех устройств. Эта программа запускается при включении питания и нажатии на кнопку «Сброс». Предполагается, что режим задается жестко для всех случаев обмена данными через адаптер, которые могут встретиться в ходе работы машины. При этом для программирования режима нужно вывести в порт, за которым закреплен регистр режима, всего один байт.

Многорежимное использование предполагает, что в различных фазах функционирования машины обмен данными через адаптер может быть организован в различных режимах. Это происходит либо в связи с тем, что устройст-

ва, подключаемые к последовательному интерфейсу, могут меняться по ходу эксплуатации машины, либо в связи с тем, что к последовательному интерфейсу подключается устройство, способное перестраивать свой режим, например, с целью поддержки максимально возможной скорости обмена в условиях изменения уровня помех в линиях связи, либо последовательный интерфейс через дополнительный коммутатор может подключаться к одному из нескольких каналов последовательной связи. При многорежимном использовании для инициализации адаптера в регистр управления выводятся два байта – сначала байт программного сброса 040h (01000000b), а затем байт режима.

Рассмотрим два примера настройки адаптера К580ВВ51 в фазе инициализации, считая, что регистр управления закреплен за портом с адресом 0FFh, а регистр данных за портом с адресом 0FEh.

Пример 2.1.

Пусть требуется организовать вывод массива байтов через рассматриваемый адаптер из специализированной ЭВМ на базе МП К1821ВМ85 или К580ВМ80 (однорежимное использование адаптера) при следующих параметрах: $k$ = 1 (В1,В0 = 01), длина слова – 8 бит (В3,В2 = 11), нет контроля (В5,В4 = 00), два стоповых бита (В7,В6 = 11). Программа инициализации имеет вид:

```
INI51_80: MVI   A, 0CDH  ; формирование байта режима 11001101
          OUT   0FFH     ; вывод байта режима
```

Пример 2.2.

Требуется организовать ввод массива байтов через рассматриваемый адаптер в ЭВМ на базе МП К1810ВМ86 для случая многорежимного использования адаптера. Пусть устанавливается такой режим: $k$ = 16 (В1,В0 = 10), длина слова – 8 бит (В3,В2 = 11), контроль четности (В5,В4 = 11), 1 стоповый бит (В7,В6 = 01). Поскольку инициализация возможна во многих местах программы ЭВМ, имеет смысл организовать инициализацию в виде подпрограммы. Пусть эта подпрограмма называется .INI51_86 и байт режима передается в нее через регистр AH. Тогда подпрограмма инициализации будет иметь вид:

```
INI51_86  PROC NEAR
          MOV  AL, 040H  ; формирование байта сброса
          OUT  0FFH      ; программный сброс адаптера
          MOV  AL, AH    ; формирование байта режима
          OUT  0FFH, AL  ; вывод байта режима
          RET
.INI51_86 ENDP
```

Для инициализации заданного режима с помощью этой подпрограммы необходимо выполнить следующие команды:

```
          MOV  AH, 07EH ; формирование байта режима
          CALL INI51_86 ; настройка адаптера на этот режим
```

### 2.2. Адаптер 16550

Теперь рассмотрим основные элементы структуры адаптера 16550, который может использоваться как отдельная микросхема и в то же время является составной частью чипсета ПЭВМ, совместимой с IBM PC. На рис. 2.2 представлена простейшая схема организации интерфейса RS232 на этой микросхеме для специализированной системы на базе МП I8088 (российский аналог К1810ВМ88). Данная схема приводится на странице 20 технического описания [7] адаптера 16550D и приводится ниже без изменений.

В правой части схемы расположены преобразователи уровней TTL-RS232, причем используется 25-контактный разъем DB25P (см. рис. 1.7).

МП 8088 имеет совмещенные в младшем байте линии адреса и данных. По стробу ALE младший байт адреса буферизуется в регистре DP8212 (российский аналог – К589ИР12). Три младших разряда этого байта используются для адресации внутренних регистров объектов адаптера (входы A0-A2). Для данных используется приемопередатчик DP74038, направление передачи которого задается сигналом DT/R МП (1 – выдача, 0 – прием), а сам момент готовности МП выдавать или принимать данные задается выходом DEN (активный 0).

В схеме отсутствует дешифратор адреса для выбора кристалла. Высокий уровень сигнала на выходе $\overline{IO}/M$ МП означает, что происходит обращение к порту ввода-вывода, а низкий уровень задает обращение к памяти. В схеме этот выход непосредственно связан со входом выбора кристалла CS адаптера. Тем самым исключается возможность включать в пространство портов ввода-вывода еще какие-либо устройства. Если такое ограничение недопустимо, то необходимо добавить в схему дешифратор адреса.

*Для получения более полного представления о внешних контактах* адаптера 16550 на рис. 2.3 изображен ее корпус. Как мы видим, микросхема имеет три адресных входа $A0-A2$, что в сочетании с сигналами, различающими ввод и вывод по адресу, позволяет адресовать более чем 8 регистров. Кроме того, интерпретацию некоторых адресных значений меняет так называемый бит DLAB (Divisor Latch Access Bit), расположенный в разряде 7 регистра управления линией. При единичном значении этого бита кодами 000 и 001 на входах $A0-A2$ адресуется двухбайтовый регистр программируемого делителя частоты (коэффициент деления $k$). При нулевом значении DLAB этими кодами адресуются буферный регистр данных и регистр разрешения прерываний.

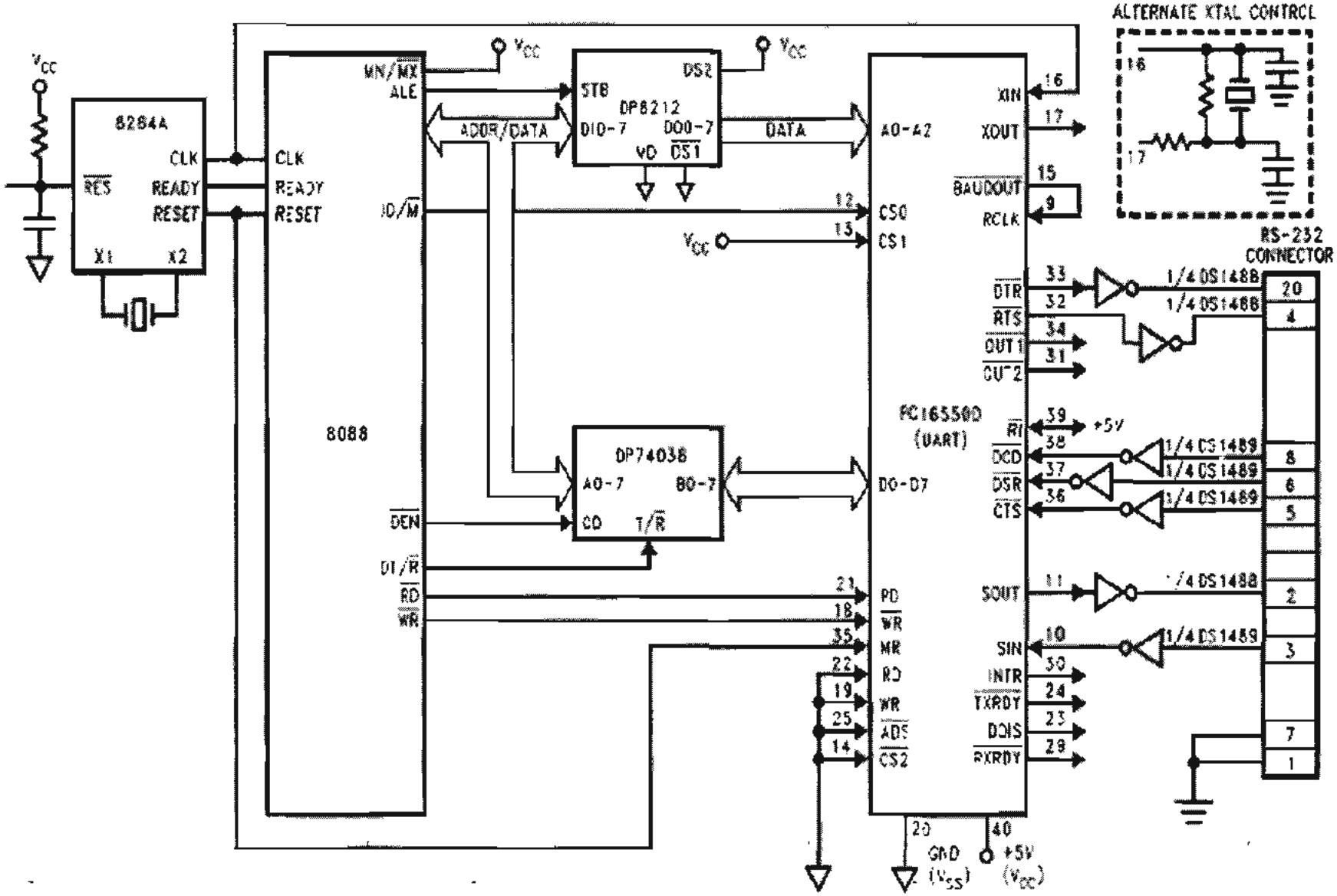



Рис. 2.2. Базовая схема подключения УСАРТ 16550 к МП

Несколько входов выбора кристалла $CS0, CS1, \overline{CS2}$ позволяют упростить построение дешифратора адреса. Например, для порта COM1 ПЭВМ, совместимой с IBM PC, базовый адрес которого равен 0F8h=11111000b, для выбора кристалла адаптера достаточно иметь схему 4И, на входы которой подаются адресные входы A5-A7 и сигнал IO/M, а выход замыкается на вход адаптера CS0. Разряды адреса A4,A3 *подаются при этом на входы* CS1, CS2.

Дублирование входов управления записью $WR, \overline{WR}$ и чтения $RD, \overline{RD}$ также упрощает схемы управления режимами и селекции адаптера. Адресные сигналы стробируются низким уровнем сигнала на входе ADS. В схеме на рис. 2.2 дешифратор адреса *вырожден*.

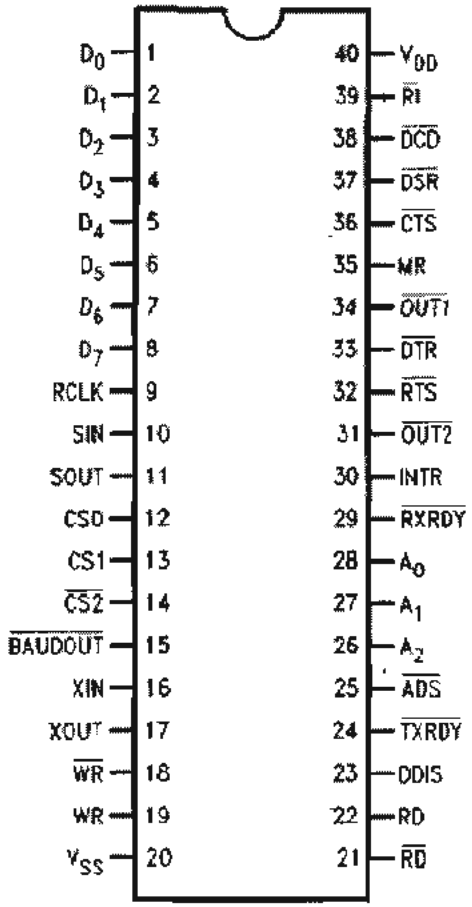



Рис. 2.3. Корпус ИМС 16550

В технической документации принято описывать регистры адаптера, отталкиваясь от их внутренних адресов. Так в документе [7] приводится таблица с описанием регистров, которая в переводе на русский язык представлена ниже в табл. 2.1.

Приводимое ниже рассмотрение адаптера касается, прежде всего, случая, когда он включен в чипсет ПЭВМ, т. е. используется для реализации внешних коммуникаций через COM1 и COM2.

Первый адаптер COM1 имеет базовый адрес 3F8h и занимает диапазон адресов от 3F8h до 3FFh. Второй адаптер COM2 занимает адреса 2F8h – 2FFh. Асинхронные адаптеры могут вырабатывать прерывания: COM1 – IRQ4 (INT 0Ch), COM2 – IRQ3 (INT 0Bh). Все следующие рассуждения будет касаться только COM1. Их легко перенести на COM2, если поменять в адресах портов первую цифру 3 на 2 и заменить адрес вектора прерываний.

Таблица 2.1.

Регистры адаптера 16550

| DLAB | A2 | A1 | A0 | Регистр |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | При чтении – буферный регистр приемника<br>При записи – буферный регистр передатчика |
| 0 | 0 | 0 | 1 | *Регистр разрешения прерываний (только запись)* |
| X | 0 | 1 | 0 | При чтении – регистр идентификации прерывания |
| X | 0 | 1 | 0 | При записи – регистр установки параметров FIFO и DMA |
| X | 0 | 1 | 1 | Регистр управления линией |
| X | 1 | 0 | 0 | Регистр управления модемом |
| X | 1 | 0 | 1 | Регистр состояния линии, только чтение |
| X | 1 | 1 | 0 | Регистр состояния модема, только чтение |
| X | 1 | 1 | 1 | Рабочий регистр |
| 1 | 0 | 0 | 0 | Младший байт делителя частоты (только запись) |
| 1 | 0 | 0 | 1 | Старший байт делителя частоты (только запись) |

Буферу вводимых и выводимых данных соответствует адрес регистра адаптера (АРА), равный нулю, или порт 3F8h ПЭВМ. Если записать единицу в старший бит управляющего регистра порта 3FBh (АРА=3), то вывод в порт 3F8h будет задавать младший байт делителя частоты. Старший байт делителя записывается при этом в порт 3F9h (АРА=1). Частота задающего генератора $F$ = 1 843 200 Гц. Коэффициент предделителя $q$ = 16. Таким образом, коэффициент деления нужно выбирать по формуле

$$k = F/q/BAUD_T = 115200/BAUD_T.$$

Тот факт, что имеет место соотношение F/q = 1 843 200/16 = 115 200, предопределил именно значение 115 200 в качестве максимальной скорости передачи данных через COM-порт ПЭВМ.

Порт 3F9h используется в качестве регистра старшего разряда делителя только после вывода в порт 3FBh байта с установленным старшим битом (бит DLAB). Иначе он используется как регистр управления прерываниями. В этом режиме используются 4 младших разряда порта:

B0 – разрешение прерывания при готовности принимаемых данных;

B1 – разрешение прерывания после передачи байта, когда выходной буфер передачи пуст;

B2 – разрешение прерывания по обнаружению состояния BREAK или при возникновении ошибки;

B3 – разрешение прерывания по изменению состояния входных линий на разъеме RS232 (CTS, DSR, RI, DCD).

Порт 3FAh (АА=2) используется для идентификации причины прервания:

B0 – нет прерываний, ожидающих обслуживания;

(B2,B1) – 00: прерывание по состоянию приемника; возникает при переполнении приемника, ошибках четности или формата данных или при состоянии BREAK; сбрасывается после чтения из порта 3FDh;

01: данные приняты и доступны для чтения; сбрасывается после чтения данных из порта 3F8h;

10: буфер передатчика пуст; сбрасывается при записи новых данных в регистр данных передатчика по адресу 3F8h;

11: состояние модема; устанавливается при изменении состояния входных линий CTS, RI, DCD, DSR; сбрасывается после чтения состояния модема из порта 3FEh;

(B7÷B3) – должны быть равны 0.

Через порт 3FBh (АРА⁻3) задаются режимы последовательного обмена:

(B1,B0) – длина символа; 00: $n$=5, 01: $n$=6, 10: $n$=7, 11: $n$=8;

B2 – число стоповых бит; 0 – 1 бит, 1– 2 бит;

(B5,B4) – управление контролем; 00,10 – нет, 01 – нечетность, 11 – четность;

B5 – фиксация четности; при B5=1 бит четности всегда принимает значение 0, если задан контроль четности, и значение 1, если контроль нечетности;

B6 – установка перерыва; вызывает вывод строки нулей в качестве сигнала BREAK для подключенного устройства;

B7 – 1: порты 3F8h и 3F9h используются для загрузки делителя частоты; иначе используются как обычно.

Пример 2.3.

Требуется разработать подпрограмму INI16550, которая настраивает рассматриваемый адаптер в специализированной ЭВМ на базе МП I8086. В подпрограмму через регистр AX передается коэффициент деления базовой частоты $k$, через регистр DL – байт режима последовательного обмена. Пусть подпрограмма предназначена только для настройки режимов, а инициализация конкретных операций ввода-вывода выполняется в других частях управляющей системы. Это означает, что после настойки режимов необходимо запретить прерывания. С учетом всех вышепредставленных соглашений подпрограмма инициализации будет иметь такой вид:

```
INI16550  PROC NEAR
          ; устанавливаем DLAB для доступа к делителю частоты
          PUSH AX
          IN  AL, 3FBh
```

```
        OR  AL, 80h
        OUT 3FBh, AL
        ; программируем делитель
        POP  AX
        OUT 3F8h, AL   ; младший байт
        MOV AL, AH     ; старший байт
        OUT 3F9h, AL
        ; устанавливаем режимы
        MOV AL, DL
        OUT 3FBh, AL
        ; запрешаем прерывания
        MOV AL, 0
        OUT 3F9h, AL
        RET
INI16550  ENDP
```

Пример 2.4.

Пусть требуется установить с помощью подпрограммы INI16550 режим передачи со следующими параметрами: скорость передачи - 1200 бод, число информационных бит кадра - 8, метод контроля - нечетность, число стоповых бит - 2. Для этого необходимо выполнить следующие команды::

```
MOV   AX, 115200/1200  ; формирование коэффициента деления
MOV   DL, 01111b       ; формирование байта режима
CALL  INI16550         ; настройка адаптера на этот режим
```

# 3. Программная реализация процессов ввода-вывода

## 3.1. Базовые алгоритмы для ввода-вывода с опросом флага готовности

Вывод заданного числа байтов в режиме опроса готовности обычно реализуется на основе следующего алгоритма:

Алгоритм 3.1.
1) настройка режима;
2) *инициализация указателя массива выводимых данных;*
3) занести в счетчик байтов размер блока данных;
4) ввести байт состояния;
5) если признак готовности в байте состояния равен 0, то идти к 4;
6) вывести байт данных;
7) продвинуть указатель массива к следующему байту;
8) уменьшить счетчик байтов на единицу;
9) если счетчик байтов не равен 0, то идти к 4.

В этом алгоритме пункты 1÷3 инициализируют цикл вывода массива, а пункты 4÷9 образуют этот цикл. Внутри цикла через пункты 4 и 5 реализуются циклический процесс опроса флага готовности.

При вводе данных адаптер может обнаружить ошибку. Программа ввода должна зафиксировать этот факт и продолжить ввод, поскольку передатчик «не знает» об ошибке и не может прекратить передачу блока данных. Один из наиболее простых способов фиксации ошибки – подсчет числа ошибок. Если счетчик ошибок после приема блока данных равен нулю, то это означает нормальное завершение процесса ввода, иначе ПО принимающей стороны должно что-то предпринять, например, сообщить передающей стороне, что необходимо повторить передачу блока данных. Алгоритм ввода блока данных фиксированной длины с подсчетом числа ошибок обычно имеет такой вид:

Алгоритм 3.2.
1) настройка режима;
2) инициализация указателя массива памяти;
3) занести в счетчик байтов размер блока данных;
4) обнулить счетчик ошибок;
5) ввести байт состояния;
6) если признак готовности в байте состояния равен 0, то идти к 5;
7) если в байте состояния имеются признаки ошибки,
   то увеличить счетчик ошибок;

8) ввести байт данных и разместить в памяти;
9) продвинуть указатель массива к следующему байту;
10) уменьшить счетчик байтов на единицу;
11) если счетчик байтов не равен 0, то идти к 5.

В этом алгоритме пункты 1÷4 инициализируют цикл ввода, а пункты 5÷11 образуют этот цикл. Циклический процесс опроса флага готовности осуществляется в пунктах 5 и 6.

## 3.2. Программная реализация ввода-вывода с опросом флага готовности для адаптера 8251

Варианты реализации двух приведенных алгоритмов приведены в примерах 3.1 и 3.2.

Пример 3.1.

Пусть требуется организовать вывод массива байтов на основе алгоритма 3.1 при режимах и адресах регистров адаптера, рассмотренных в примере 2.1. Во время вывода на линию TxD не предполагается обслуживание ввода с линии RxD. Адрес выводимого массива помечен меткой BUF, длина массива равна 64 байта. Тогда программа вывода будет иметь такой вид:

```
    ; байт команды:
    ;   B0=1 - передача разрешена,
    ;   B1=1 - установка флага готовности передатчика
    MVI   A, 3      ; формирование байта команды:
    OUT   0FFH      ; вывод байта команды
    LXI   H, BUF    ; позиционировать HL на начало массива
    MVI   C, 64     ; инициализировать счетчик байтов
L:  IN    0FFH      ; проверка флага готовности
    ANI   1         ; флаг находится в младшем разряде
    JZ    L         ; если не готов, то вернуться к проверке
    MOV   A, M      ; взять очередной байт из массива
    OUT   0FEH      ; вывести байт
    INX   H         ; продвинуться к следующему байту
    DCR   C         ; счет байтов
    JNZ   L         ; продолжить, если не все выведено
```

Пример 3.2.

Требуется организовать ввод массива из 10 байтов на основе алгоритма 3.2 для специализированной ЭВМ на базе МП К1810ВМ86 (адреса регистров адаптера и режимы как в примере 3.1). Адрес области памяти для вводимого массива помечен меткой BUF. В регистре DL подсчитываются ошибки, обнаруживаемые контролем четности. Программа ввода будет иметь такой вид:

```
    ; байт команды:
    ;   B0=1 - передача разрешена,
    ;   C
    ;   B1=1 - установка флага готовности передатчика

    MOV   AL, 21H    ; формирование байта команды
    OUT   0FDH       ; вывод байта команды
    LEA   DI, BUF    ; позиционировать DI на начало массива
    MOV   CX, 10     ; инициализировать счетчик байтов
    XOR   DL, DL     ; очистка счетчика ошибок

    ; цикл ввода CX байтов
L:  IN    0FDH       ; проверка флага готовности
    TEST  AL, 2      ; флаг находится в разряде B1
    JZ    L          ; если не готов, то вернуться к проверке
    TEST  AL, 8      ; ошибка четности находится в разряде B3
    JZ    NE         ; перепрыгнуть через счет ошибок
    INC   DL
NE: IN    0FCH       ; ввести байт
    MOV   [DI], AL   ;   и разместить его в массиве
    INC   DI         ; продвинуться к следующему байту BUF
    LOOP  L          ; пока --CX != 0, продолжать цикл
    CMP   DL, 0
    JNE   ERR        ; идти к программе реагирования на ошибку
```

Рассмотрим пример организации вывода в случае реализации интерфейса RS232 на основе адаптера 16550 для специализированной ЭВМ, программно совместимой с IBM PC.

*Пример 3.3.*

Логику процесса вывода сохраняем как в примере для К1821ВМ85 и 8251, но процесс вывода оформляем как подпрограмму WRITE и считаем, что в ответ мы получаем квитанцию с нулевым байтом в случае безошибочной передачи. Считаем, что процесс нужно повторять до 5 попыток, после чего выход из подпрограммы с возвратом в AX значения квитанции.

```
        ; объявление буфера данных
        .DATA
BUF     DB  64 DUP (?)   ; сам буфер данных
        .CODE
```

```
; Подпрограмма вывода WRITE получает через SI
; адрес выводимого массива, а через CX - размер
WRITE    PROC NEAR
         PUSH  BP
         MOV   BP, SP
         PUSH  SI              ; сохраняем адрес буфера для повтора
         PUSH  CX              ; сохраняем длину буфер для повтора
         ; организуем внешний цикл не более 5 попыток
         MOV   CX, 5           ; число попыток
         ; i-я попытка
PROBE:   PUSH  CX              ; внутренний цикл выводит CX байтов
         MOV   SI, [BP-2]      ; восстановить адрес буфера
         MOV   CX, [BP-4]      ; восстановить длину буфера
LW:      IN    AL, 03FDh       ; проверка флага готовности
         TEST  AL, 100000b     ; флаг находится в разряде B5
         JZ    LW              ; если не готов, то вернуться
                               ;   к проверке
         MOV   AL, [SI]        ; взять очередной байт из массива
         OUT   3F8H, AL        ; вывести байт
         INC    SI             ; продвинуться к следующему байту
         LOOP  LW              ; счет байтов в CX и организация
                               ;   цикла
         ; ввод квитанции и проверка
LC:      IN    AL, 03FDh       ; проверка флага готовности приемника
         TEST  AL, 1           ; флаг находится в разряде B0
         JZ    LC              ; если не готов, то вернуться
                               ;   к проверке
         TEST  AL, 1110b       ; проверить ошибку приема
         JNZ   ERR             ; вывести байт
         IN    AL, 03F8h       ; ввести квитанцию
         CMP   AL, 0
         JE    WR_DONE
ERR:     POP   CX
         LOOP  PROBE
WR_DONE: MOV   AH, 0           ; квитанция в AL,
                               ; нужно представить ее в AX
         MOV   SP, BP          ; освобождаем стек
                               ;   от локальных данных
         POP   BP
RET
WRITE    ENDP
...
```

```
; Пример вызода массива BUF с помощью ГП WRITE
        LEA    SI, BUF
        MOV    CX, SIZE_BUF
        CALL   WRITE
```

### 3.3. Программная реализация ввода-вывода в режиме прерываний

Ввод или вывод заданного числа байтов в режиме прерываний предполагает, что в ПО имеется две программы: программа инициализации процесса ввода/вывода и программа обработки прерываний, которая активизируется по готовности адаптера передавать очередной байт. Для активизации прерываний ИМС 8251 вырабатывает два внешних сигнала готовности (на рис. 1.8 не показаны): RxRDY (вывод 14) – готовность приемника, TxRDY (вывод TxRDY) – готовность передатчика. Обычно в системе на базе МП имеется контроллер прерываний, обеспечивающий активизацию различных обработчиков прерываний для реагирования на готовность различных каналов ввода-вывода.

Прерываемая программа в общем случае никак не связана с процессом ввода-вывода и использует любые регистры процессора. В этой связи обработчик прерываний должен представлять указатель буфера и счетчик байтов в оперативной памяти. Загрузку стартовых значений этих переменных выполняет программа инициализации. После инициализации управление, как правило, передается какой-то другой задаче. Это делает обычно диспетчер процессов операционной системы. В какой-то момент может возникнуть ситуация, когда диспетчеру процессов требуется узнать, не завершен ли ввод-вывода, с тем чтобы возобновить исполнение процесса, запустившего ввод-вывод. В простейшем случае обработчик прерываний сообщает о состоянии процесса ввода-вывода в некотором байте статуса. В этом байте необходимо представлять по крайней мере три ситуации: процесс не завершен, процесс завершен нормально и процесс завершен с ошибкой. Будем считать, что эти три ситуации кодируются соответственно значениями 0, 1, -1. Значит, программа инициализации должна очистить байт статуса. Кроме того, программа инициализации должна настроить адаптер связи, а также контроллер прерываний таким образом, чтобы тот активизировал обработчик прерываний при возникновении запросов со стороны адаптера связи. В простейшем случае достаточно установить в контроллере прерываний бит разрешения прерываний по возникновению запроса на соответствующем входе запроса (IRQ  Interrupt Request ) и разрешить процессору

реагировать на запросы контроллера прерываний. Рассмотрим процесс вывода данных в режиме прерываний на примере.

Пример 3.4.

Пусть требуется организовать с использованием прерываний вывод массива байтов в специализированной ЭВМ на базе МП К1821ВМ85 при режимах и адресах регистров адаптера, рассмотренных в примере 3.1. Мы должны учитывать, что передатчик не знает, имели ли место ошибки в передаче данных. Обнаружение ошибок - дело приемника. В то же время продолжать программу, инициализировавшую вывод массива, без уверенности, что данные переданы верно, нельзя. Пусть протокол обмена данными построен таким образом, что сразу после получения массива принимающая сторона посылает в ответ передающей так называемую квитанцию. Для простоты будем организовывать прием квитанции в режиме опроса готовности непосредственно в обработчике прерывания передатчика после завершения передачи всего массива. Будем считать, что квитанция содержит либо байт со значением количества принятых байт (считаем, что выводимый массив имеет длину 1..255 байт), либо со значением 0, говорящем о том, что была ошибка. Для надежности будем сравнивать значение байта квитанции с длиной выведенного массива. Это означает, что обработчик прерываний по ходу вывода должен работать с изменяемым значением счетчика выведенных элементов массива, а по окончанию процесса вывода должен будет сравнить начальное значение этого счетчика с содержимым квитанции. С учетом данного соглашения данные, используемые обработчиком прерываний для управления процессов вывода, могут быть объявлены следующим образом:

```
ADDR   DW 0 ; указатель буфера вывода
NUM    DB 0 ; количество выводимых байтов
COUNT  DB 0 ; изменяемый счетчик выведенных байт
STAT   DB 0 ; статус процесса вывода
```

Пусть подпрограмма инициализации получает через HL адрес выводимого массива, через C – длину массива, а бит разрешения прерывания канала вывода адаптера связи находится в младшем разряде регистра с адресом порта 0F0h. Тогда эта подпрограмма будет иметь вид:

```
WR_INIT:
    ; Настройка ИМС 8251 через байт команды <B0,B1>=11
    MVI    A, 3        ; формирование байта команды:
    OUT    0FFH        ; вывод байта команды
    SHLD   ADDR        ; инициализировать указатель буфера вывода
    MOV    A, C        ; инициализировать
    STA    NUM         ;   байт длины выводимого массива
    STA    COUNT       ;   и счетчик байтов
```

```
    XRA   A          ; очистка
    STA   STAT       ;   байта состояния
    IN    0F0h       ; разрешение прерывания
    ORI   1          ;   передатчику ИМС 8251
    EI               ; разрешение прерывания МП К1821ВМ85
    RET
```

Пусть номер вектора прерываний равен 3, что обязывает разместить стартовую точку обработчика прерываний по адресу 18h. Поскольку следующий стартовый адрес обработчика прерываний равен 20h, то стартовая часть рассматриваемой программы обработки должна занять не более 8 байт и передать управление на точку продолжения. Вариант программы обработчика может быть таким:

```
    ORG   18h
    PUSH  PSW        ; сохранить флаги и A
    PUSH  H          ; сохранить HL для использования его в качестве
                     ;   адреса буфера
    JMP   TX_I
    ORG   20h        ; начало обработчика другого прерывания
    ...

    ; Продолжение обработчика
TX_I:
    LDA   COUNT
    ORA   A
    JE    TX_DONE
    INR   A
    STA   COUNT
    LHLD  ADDR       ; загрузить адрес в HL
    MOV   A, M       ; взять очередной байт из массива
    OUT   0FEH       ; вывести байт
    INX   H          ; продвинуться к следующему байту
    SHLD  ADDR       ; счет байтов
TX_FIN:
    POP   H
    POP   PSW
    EI
    RET

TX_DONE:
    ; Сначала запрет прерывания от передатчика
    IN    0F0h       ; ввод байта управления контроллера прерывания
    ANI   0FEH       ; очистка младшего бита
    ; Настройка процесса ввода квитанции
```

```
    MVI   A, 21H      ; формирование байта команды
    OUT   0FDH        ; вывод байта команды

    ; Цикл ожидания готовности приемника
L:  IN    0FDH        ; проверка флага готовности
    ANI   A, 2        ; флаг находится в разряде B1
    JZ    L           ; если не готов, то вернуться к проверке.

    ; Анализ верности приема квитанции и ее содержимого
    IN    0FDH        ; еще раз вводим статус для анализа ошибки
    ANI   A, 8        ; ошибка четности находится в разряде B3
    JNZ   TX_ERR      ; ошибка приема квитанции
    LDA   NUM         ; сохранить
    MOV   L, A        ;   длину массива в L
    IN    0FCH        ; ввести байт квитанции
    CMP   L           ;
    JE    TX_ERR      ; была ошибка передачи массива

    ; Нормальное завершение
    MVI   A, 1
SET_STAT:
    STA   STAT
    JMP   TX_FIN
TX_ERR:
    MVI   A, -1
JMP SET_STAT
```

Пример вызова программы инициализации вывода рассмотрим, исходя из того, что выводимые данные определены следующим образом:

```
    WR_BUF      DS      400 ; буфер данных
    WR_SIZE     EQU     400 ; константный размер буфера
```

Для инициализации процесса вывода необходимо вызвать ПП WR_INIT, предварительно загрузив в HL и DE адрес буфера и его размер. Если это делает прикладная программа, то после вызова ПП она должна будет сообщить монитору, чтобы тот приостановил исполнение прикладного процесса. Например, так:

```
; WRITE(WR_BUF, WR_SIZE)
    LXI  H, WR_BUF
    LXI  D, WR_SIZE
    CALL WR_INIT
    LXI  H, WR_STAT
    PUSH H
```

```
        RST  7
```

Другой вариант заключается в том, что инициализацию выполняет монитор, но ему передаются параметры WRITE:

```
        LXI  H, WR_BUF
        PUSH H
        LXI  H, WR_SIZE
        PUSH H
        MVI  A, WRITE
        RST  7
```

Необходимо отметить, что процедуры вывода, приведенные в предыдущем примере, обладают одним недостатком. Если скорость приема данных невысока, то цикл опроса готовности может занять время, в течение которого процессор мог бы выполнить сотни команд обработки данных вместо того, чтобы ожидать готовность приемника. Если такая потеря времени нежелательна, то необходимо организовать ввод квитанции также в режиме прерывания. Передать управление прерванной программе можно, только выйдя из обработчика прерываний. Поскольку прерывания от передатчика запрещены, то после выхода из обработчика мы не сможем в него вернуться, чтобы анализировать квитанцию. Это означает, что организацию ввода квитанции и ее обработки нужно перенести в какую-то другую процедуру, которая тесно связана с диспетчером процессов. Сам диспетчер процессов активизируется не только программами, желающими запустить какой-то процесс, но и прерываниями от таймера. Это позволяет периодически проверять байты состояния служебных процессов ввода-вывода.

Теперь рассмотрим пример организации вывода в режиме прерываний для адаптера на базе ИМС 16550, считая, что адреса регистров соответствуют порту COM1 ПЭВМ, совместимой с IBM PC.

Пример 3.5.

```
; Управляющая структура для вывода
WR_ADDR  DW 0   ; указатель буфера вывода
WR_COUNT DW 0   ; изменяемый счетчик выведенных байт
WR_STAT  DB 0   ; статус процесса вывода: 0 - не завершено,
                ; 1 нормально завершено,
                ; -1 - завершено с ошибкой
```

```
PUBLIC   R_ADDR, WR_COUNT, WR_STAT
```

Пусть подпрограмма инициализации получает через DX адрес выводимого массива, через AX – длину массива, а бит разрешения прерывания в контроллере прерываний находится в разряде 4 (IRQ4) регистра управления этого контроллера, причем адрес регистра управления – 21h. Тогда подпрограмма инициализации будет иметь вид:

```
        .CODE
WR_INIT: MOV  WR_ADDR, DX  ; инициализируем указатель буфера вывода
        MOV  WR_COUNT, AX ; инициализируем счетчик байтов
        MOV  BYTE PTR WR_STAT, 0  ; очищаем байт состояния
        MOV  AL, 10b  ; флаг разрешения прерываний в разряде B1
        OUT  03F9h    ; разрешение прерывания адаптеру
        MOV  AL, 10000b   ; флаг разрешения прерываний
                          ;  в разряде B4 контроллера прерываний
        OUT  21h          ; разрешение прерывания
                          ;  в контроллере прерываний
        STI               ; разрешение прерывания МП К1810ВМ86
        RET
```

Пусть номер вектора прерываний равен 0Ch (как у COM1 IBM PC). Тогда вектор прерываний может быть объявлен, например, так:

```
      .CODE
      ORG 4*0Ch
      DD  WR_INT; точка входа
```

Сам обработчик представляется без обработки ошибок, предполагая, что организацию ввода квитанции будет делать монитор, используя режим прерываний:

```
        .CODE
EXTERN   R_ADDR, WR_COUNT, WR_STAT

    ; Обработчик прерываний
WR_INT   PROC FAR
        DEC     WR_COUNT; если это прерывание после вывода
                        ; последнего байта, то идти на завершение
        JE      WR_DONE
        PUSH    SI
        MOV     SI, WR_ADDR
```

```
        MOV     AL, [SI]
        INC     WR_ADDR
        POP     SI
        JMP     WR_FIN
        IRET
WR_DONE: INC WR_STAT
        ; запрет прерываний адаптеру
        MOV     AL, 0
        OUT     03F9h
        IRET
WR_INT  ENDP
```

Рассмотренные подпрограммы инициализации режима прерываний должны вызываться из управляющего монитора. Программирование мониторинга процессов существенно зависит от применяемой операционной системы (ОС). Когда создается специализированная система контроля и управления, то в качестве базовой ОС используется типовая система реального времени (*ОС* РВ).

## Библиографический список


1. Гук, М. Интерфейсы ПК: справочник / М. Гук – СПб: Питер, 1999. 416 с.
2. Олссон, Г., Пиани, Д. Цифровые системы автоматизации и управления / Г. Олссон, Д. Пиани – СПб.: Невский Диалект, 2001. 557 с.
3. Мячев, А.А. Интерфейсы систем обработки данных / А. А. Мячев, В. Н. Степанов, В. Г. Щербо – М.: Радио и связь, 1989. – 416 с.
4. Коффрон, Дж., Лонг, В. Расширение микропроцессорных систем / Дж. Коффрон, В. Лонг – М.: Машиностроение, 1987. – 320 с.
5. Практичсский курс программирования микропроцессорных систем / В. Г. Майоров, А. И. Гаврилов – М.: Машиностроение. 1989. – 272 с.
6. Левенталь, Л. Программирование на языке ассеблера для микропроцессоров 8080 и 8085 / Л. Левенталь, У. Сэйвилл – М.: Радио и связь, 1987. – 448 с.
7. PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs / Technical Reference – National Semiconductor Corporation, 1995. – 22 p.
8. Фролов, А.В. Аппаратное обеспечение персонального компьютера / А. В. Фролов, В. Г. Фролов – М.: Диалог-МИФИ, 1997. – 304 с.
9. Баранов, В.Н. Применение микроконтроллеров AVR: Схемы, алгоритмы, программы / В. Н. Баранов – М.: Додэка, 2004. – 288 с.
10. Евстифеев, А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL» / А. В. Евстифеев – М.: Изд. дом «Додэка-XXI», 2004. – 560 с.
11. Предко, М. Справочник по PIC-микроконтроллерам: Пер. с англ. / М. Предко – М.: ДМК Пресс, 2004; ООО «Изд. дом «Додэка-XXI», 2004. – 512 с.
12. Микроконтроллеры. Выпуск 1: Однокристальные микроконтроллеры PIC17C4x, PIC17C75x, M3820. – М.: Додэка, 1998. – 384 с.

Учебное издание

**Технические решения для сопряжения специализированных ЭВМ с объектами ввода-вывода через последовательный интерфейс**

Учебно-методические указания

Составитель НЕГОДА Виктор Николаевич
Редактор Н. А. Евдокимова
Подписано в печать 30.09.2006. Формат 60x84 1/16. Бумага офсетная.
Печать трафаретная. Усл. печ. л. 2,63. Уч.-изд.л. 2,00. Тираж 100 экз.
Заказ № 1212
Ульяновский государственный технический университет,
432027, г.Ульяновск, ул. Северный Венец, 32.
Типография УлГТУ, 432027, г.Ульяновск, ул. Сев. Венец, 32.