

Саратовский государственный университет им. Н.Г.Чернышевского

А.А.Семёнов

ОРГАНИЗАЦИЯ  
ПОСЛЕДОВАТЕЛЬНОГО  
ИНТЕРФЕЙСА

Учебное пособие  
для студентов факультета компьютерных наук  
и информационных технологий  
и факультета nano- и биомедицинских технологий

Издательство Саратовского университета  
2007

УДК 621.382-181(075.8)

ББК 32.844.1 я73

С30

**Семёнов А.А.**

С30 Организация последовательного интерфейса: Учеб. пособие для студ. фак. компьютерных наук и информационных технологий и фак. нано- и биомедицинских технологий – Саратов: Изд-во Сарат. ун-та, 2007. – 51 с.: ил.

ISBN \_ - \_ - \_ - \_ - \_

Учебное пособие представляет собой руководство к практическим занятиям по курсам "Устройство и применение микропроцессоров" и "Микропроцессорные системы". Содержит теоретическое описание материала, знание которого необходимо при выполнении лабораторных и практических работы по изучению основ программной и аппаратной организации последовательного интерфейса микропроцессорных систем.

Для студентов, обучающихся по направлениям "Электроника и микроэлектроника", "Физика полупроводников и диэлектриков" специальностям "Вычислительные машины, комплексы, системы и сети", "Микроэлектроника и полупроводниковые приборы".

Рекомендуют к печати:

Кафедра физики твердого тела факультета  
нано- и биомедицинских технологий  
Саратовского государственного университета  
Доктор физико-математических наук А.В. Скрипаль

УДК 621.382-181 (075.8)

ББК 32.844.1 я73

ISBN \_ - \_ - \_ - \_ - \_

© Семёнов А.А., 2007

## ОГЛАВЛЕНИЕ

<b>В в е д е н и е</b> .....	4
<b>1. Теоретическая часть</b> .....	6
1.1. Организации ввода-вывода в последовательном коде.....	6
1.2. Архитектура БИС адаптера последовательного интерфейса K580VB51.....	13
1.3. Подключение УСАПП к магистралям микропроцессорной системы.....	15
1.4. Начальная инициализация адаптера.....	18
1.5. Режимы работы адаптера.....	20
1.6. Программирование БИС УСАПП.....	22
1.7. Программная модель БИС УАПП 8250.....	25
1.8. Основы программирования микропроцессорных систем на языке ассемблера .....	28
1.8.1. Формулировка задачи.....	29
1.8.2. Решение задачи.....	30
1.8.3. Проектирование программы.....	31
1.8.4. Проектирование модулей.....	31
1.8.5. Логическая структура ассемблерной программы.....	32
1.8.6. Отладка программы и исправление ошибок.....	36
<b>2. Экспериментальная часть</b> .....	39
2.1. Программно-аппаратные средства.....	39
2.2. Практические задания.....	41
2.3. Порядок выполнения заданий.....	45
<b>Список литературы</b> .....	46
<b>Приложение. Система команд микропроцессора K580VM80</b> .....	47

## ВВЕДЕНИЕ

В практике управления микропроцессорной системой (МПС) периферийными устройствами встречаются ситуации, когда устройства, с которыми МПС обменивается информацией, находятся на значительном расстоянии или принципиально не имеют возможности для приёма и передачи данных в параллельном коде. Простейшим примером является передача цифровой информации по радиоканалу, телефонным или телеграфным линиям. Перечисленные каналы физически не могут одновременно передавать все разряды информационного слова МПС. В подобных ситуациях целесообразно организовывать обмен данными по одной линии связи в последовательном коде. Совокупность программных и аппаратных средств управления обменом по информационному каналу в таком случае образует последовательный интерфейс, определяющий последовательный способ обмена данными МПС с внешним устройством (ВУ).

В качестве аппаратных средств интерфейса в самом простом случае могут применяться универсальные сдвиговые регистры с параллельной записью информации, включенные в состав МПС как устройства ввода/вывода (УВВ). Сигнал записи микропроцессора (МП), подключенный к такому регистру в качестве стробирующего в моменты параллельной загрузки, позволяет зафиксировать в нём состояние бит отдельных разрядов магистрали данных (МД). После этого регистр переключается в режим последовательного сдвига, и по тактовым импульсам синхросигнала МПС, определяющего частоту появления битов данных на выходе регистра, осуществляется передача данных внешнему устройству в последовательном коде. В приёмном устройстве передаваемые биты "вдвигаются" в регистр по тактовым импульсам, и по завершению процесса (поступлению тактовых импульсов в количестве, равном числу передаваемых бит) данные могут быть считаны в параллельном коде по сигналу чтения МП.

В более сложных ситуациях, когда приёмник не синхронизирован непосредственно сигналом передатчика, процесс установления связи наряду с сигналами данных может потребовать введения в поток данных специальных служебных битов, а также участия в процедуре обмена дополнительных сигналов (сигналов квитирования), определяющих направление передачи, подтверждение истинности данных, состояние ВУ и т.д. Сложный процесс асинхронного обмена данными между МПС и ВУ, а также требования наиболее рационального распределения времени работы системы

приводят к необходимости разработки специальных интерфейсных устройств для организации последовательного обмена данными. Задача таких устройств состоит в приеме данных по сигналам управления и преобразовании их в параллельный код, формировании сигналов, указывающих на наличие данных в устройстве, выдаче данных в последовательном коде по сигналам управления, а также формировании сигналов, указывающих на готовность устройства к приему новых данных.

Технология производства больших интегральных схем (БИС) позволяет создавать универсальные многофункциональные интерфейсные устройства ввода/вывода последовательной информации. Такие устройства могут быть программируемыми и применяться для двунаправленной передачи данных и сигналов управления. МПС информируется о готовности устройства к обмену данными, как правило, по сигналам прерывания. Обмен данными осуществляется путем обращения микропроцессора к устройству ввода/вывода как к ячейке памяти или ВУ. Таким образом, интерфейсные устройства ввода/вывода последовательной информации позволяют согласовать во времени процесс обмена данными между МПС и ВУ при рациональном использовании времени работы микропроцессора.

БИС для аппаратной организации последовательного интерфейса выпускаются различными производителями, но, благодаря широкому распространению компьютеров IBM PC и их аналогов, значительную популярность завоевали интегральные схемы (ИС), совместимые с **INS8250**. Микросхема, называемая **UART (Universal Asynchronous Receiver–Transmitter** или **УАПП — Универсальный Асинхронный Приемопередатчик**), осуществляет преобразование параллельного кода в последовательный при передаче и последовательного в параллельный – при приёме, контроль ошибок, формирование запросов прерывания и ряд других сервисных функций.

В отечественных ЭВМ, разработанных на основе микропроцессорного комплекта К580, функции программируемого устройства ввода/вывода последовательной информации выполняет БИС *адаптера последовательного интерфейса К580ВВ51 (КР580ВВ51А, КР580ИК51)*, так называемый *универсальный синхронно-асинхронный приёмопередатчик (УСАПП)*. Микросхема предназначена для организации ввода/вывода информации в последовательном коде и позволяет реализовать обмен данными как в синхронном, так и в асинхронном режимах.

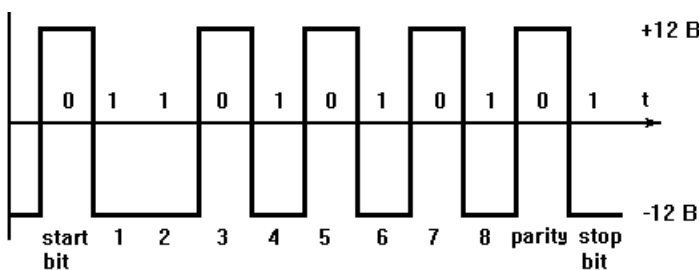
БИС УАПП и УСАПП обычно используются для сопряжения микропроцессорной системы с периферийным оборудованием, имеющим стандартный последовательный интерфейс (модемом, манипулятором "мышь", цифровым дисплеем, принтером, телетайпом, различными внешними накопителями). Микросхемы могут быть использованы для организации обмена в режиме прерывания, а также для сопряжения с устройствами, поддерживающими нестандартные протоколы последовательного обмена данными (при специальном низкоуровневом программировании).

## 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

### 1.1. ОРГАНИЗАЦИИ ВВОДА-ВЫВОДА В ПОСЛЕДОВАТЕЛЬНОМ КОДЕ

Последовательный интерфейс использует одну сигнальную линию для передачи данных в одном направлении. Информационные биты передаются последовательно друг за другом, что и предопределило название интерфейса. В соответствии со сложившейся практикой первым обычно передается младший разряд.

Последовательная передача данных может осуществляться в синхронном и асинхронном режимах. При асинхронной передаче каждому байту данных предшествует старт-бит, за ним следуют биты данных, после них может передаваться бит паритета (четности) и, в завершение посылки, передается стоп-бит, гарантирующий определенную выдержку времени между соседними посылками (рис. 1). Старт-бит следующей посылки может передаваться в любой момент времени, начиная с момента окончания стоп-бита предыдущей посылки, таким образом, между передачами могут быть паузы произвольной длительности.



**Рис. 1.** Временная диаграмма передачи 1 байта количеством передаваемых бит в секунду.

Внутренний генератор синхронизации приемника содержит счетчик-делитель частоты, обнуляемый в момент начала старт-бита от передатчика. Этот счетчик формирует внутренние стробы, по которым приемник фиксирует принимаемые биты. В идеальном случае, эти внутренние стробы должны приходиться на середину битовых интервалов, однако, из-за разностей скоростей приемника и передатчика возникает рассогласование, накапливающееся с количеством принятых бит.

Для асинхронного режима принят ряд стандартных скоростей обмена: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бит/сек. Очевидно, что чем больше скорость передачи, тем строже требования к синхронизации приемника и передатчика, кроме того, с ростом скорости обмена усиливается влияние фронтов импульсов.

Старт-бит позволяет организовать простую синхронизацию приемника по сигналу от передатчика. При этом подразумевается, что приемник и передатчик работают на одной и той же скорости обмена (т.е. имеют одну и ту же тактовую частоту), измеряемую

Количество передаваемых бит данных задается программно и может составлять 5, 6, 7 и 8 бит. Количество стоп-бит может быть 1, 1.5, 2 (по длительности). Бит паритета может отсутствовать.

Таким образом, для установления связи при асинхронном режиме необходимо установить одинаковыми следующие параметры приемника и передатчика: скорость обмена, количество бит данных, наличие бита паритета, тип паритета (четность - нечетность), длительность стоп-бита.

Синхронный режим передачи данных предполагает постоянную активность канала связи. Посылка начинается с передачи синхробайта, за которым вплотную следует последовательность передаваемых бит. Если у передатчика нет данных для передачи, то он заполняет паузу непрерывной передачей старт-бит. Очевидно, что при синхронном обмене можно добиться *больших* скоростей обмена, поскольку не требуется передавать старт-бит, стоп-бит, бит паритета (в асинхронном на каждые 8 бит данных приходится по 3-4 служебных бита). При синхронном обмене необходима внешняя синхронизация, так как передается большой массив данных. При синхронизации только в начале передачи блока данных даже небольшая разность частот приведет к появлению быстро накапливающейся ошибки.

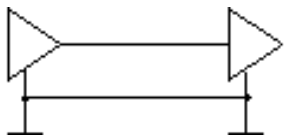
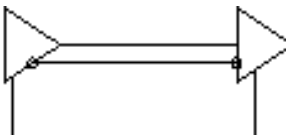
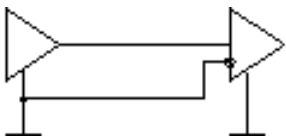
Внешняя синхронизация может обеспечиваться как за счет использования отдельной линии для передачи сигнала синхронизации, так и за счет применения самосинхронизирующихся методов кодирования данных, при использовании которых синхросигнал выделяется приемником из потока принимаемых данных.

Емкость передающей линии и токоформирующая способность источника сигналов ограничивают длину линии, при которой возможна передача. Так, формирователь транзисторно-транзисторной логики (ТТЛ) может надежно активизировать линию, длина которой не превышает 70 см. Для расширения диапазона передачи используются специализированные ИС, которые называются линейными формирователями и служат для активизации линии передачи, и соответствующие линейные приёмники, подключаемые к этой линии. Таким образом, БИС адаптеров последовательного интерфейса выполняют операции по преобразованию параллельного кода в последовательный при передаче и последовательного в параллельный – при приёме, контролю ошибок, формированию запросов прерывания, а линейные формирователи и приёмники обеспечивают нагрузочную способность адаптера и согласование с передающими линиями.

Чтобы удовлетворить требованиям различных классов стандартов, установленных Ассоциацией электронной промышленности (EIA), выпускаются разнообразные типы линейных формирователей или приемников. Так RS-232C (Reference Standard № 232 Revision C, отечественный аналог – Стык С2) является наиболее распространенным стандартом на интерфейс для микрокомпьютерных систем, тогда как стандарты RS-422A и RS-423A применяются лишь в тех случаях, когда интерфейс типа RS-232C не может

удовлетворить требованиям системы. На основе этих стандартов строятся однопроводной, симметрично-дифференциальный и несимметрично-дифференциальный интерфейсы. Характеристики этих интерфейсов сведены в таблицу 1.

**Таблица 1. Характеристики последовательных интерфейсов**

Тип интерфейса	Схема интерфейса	Параметры интерфейса: L (длина линии), V (скорость передачи)
<b>RS-232C</b> (однопроводной)		L=15 м, V=20 Кбит/с
<b>RS-422A</b> (симметрично-дифференциальный)		L=12 м, V=10 Мбит/с L=120 м, V=1 Мбит/с L=1200 м, V=100 Кбит/с
<b>RS-423A</b> (несимметрично-дифференциальный)		L=9 м, V=100 Кбит/с L=91 м, V=10 Кбит/с L=1200 м, V=1 Кбит/с

Линии интерфейсов RS-232C и RS-423A несимметричны, имеют самую низкую защищенность от синфазной помехи. RS-423A имеет приемник с дифференциальным входом, что несколько повышает его помехозащищенность. Лучшими параметрами обладает симметричный интерфейс RS-422A. Приемник и передатчик RS-422A имеют дифференциальные входы и, следовательно, обладают высокой защищенностью от синфазных помех.

Интерфейс RS-232C использует несимметричные приемники и передатчики, сигнал передается относительно общего провода (схемной земли). Интерфейс не обеспечивает гальванической развязки устройств. Уровни сигналов, подаваемых на внешний разъём через специальные буферные формирователи, лог."1" (MARK) и лог."0" (SPACE) составляют:  $-15 \dots -3$  В и  $+3 \dots +15$  В соответственно, что обеспечивает высокий уровень помехо-



защищенности интерфейса, поскольку между уровнями  $+3 \dots -3$  В существует зона нечувствительности, обуславливающая гистерезис приемника. Состояние на выходе приемника изменяется только при пересечении напряжением порога  $+3$  или  $-3$  В.

Интерфейс предполагает наличие защитного заземления обоих устройств. Присоединение и отключение устройств с автономным питанием должно производиться при отключенном питании, иначе разность не выровненных потенциалов устройств в момент коммутации может превысить допустимые пределы и вывести из строя микросхемы порта.

Различают два вида устройств, участвующих в последовательном обмене: **DTE** (**Data Terminal Equipment** или аппаратура канала данных) и **DCE** (**Data Communication Equipment** или аппаратура передачи данных). Устройство **DTE** — терминальное устройство (обычно компьютер); его задачей является преобразование данных из параллельного кода в последовательный при передаче информации и обратное преобразование при приеме. Устройство **DCE** — устройство связи (модем); его задача — преобразование сигнала к виду, удобному для организации связи при передаче и восстановлении формы исходного сигнала при приеме. Поскольку устройства **DTE** и **DCE** работают достаточно самостоятельно, предусмотрены дополнительные сигналы (сигналы квитирования) для их взаимодействия:

- **DTR** – (**Data Terminal Ready**) сигнал готовности терминала, формируемый **DTE** при необходимости организовать обмен с **DCE**.
- **DSR** – (**Data Set Ready**) сигнал готовности связного оборудования, который формирует **DCE**, как ответ на появление активного сигнала **DTR**, а также – указание на работоспособность оборудования.
- **RTS** – (**Request To Send**) сигнал запроса передачи; его формирует **DTE** при необходимости передать данные к **DCE**.
- **CTS** – (**Clear To Send**) сигнал сброса передачи, формируется **DCE**, как ответ на активность сигнала **RTS**, и обозначает временные интервалы, когда **DTE** может передавать на **DCE** данные.

В некоторых случаях при связи двух **DTE** обходятся без **DCE**, при этом функции **DCE** выполняет специальный кабель, называемый "нуль-модем". Этот кабель фактически выполняет функции модема и осуществляет физическое переназначение пар сигналов **DTR** и **DSR**, **RTS** и **CTS**, а также сигналов передачи и приема информации **TxD** и **RxD**.

Для стандартизации оборудования обычно со стороны **DTE** (например, в компьютере) устанавливается разъем типа "вилка", со стороны **DCE** (например, в модеме) – разъем типа "розетка". Тип применяемых разъемов – DB-25S или DB-9S. Распределение сигналов по контактам 25– или 9–штырькового разъема последовательного интерфейса в IBM-совместимом компьютере и их назначение приведено в таблице 2.

Таблица 2. Назначение линий интерфейса RS-232C

Сигнал	DB25S	DB9S	Назначение линии
PG	1	-	<b>(Power Ground)</b> Защитная земля. Соединяется с корпусом устройства и экраном кабеля.
SG	7	5	<b>(Signal Ground)</b> Сигнальная земля (относительно нее действуют линии сигналов)
TxD	2	3	<b>(Transmitted Data)</b> Выход передатчика
RxD	3	2	<b>(Received Data)</b> Вход приемника
RTS	4	7	<b>(Ready To Send)</b> Выход запроса передачи. Состояние "включено" уведомляет модем о том, что у терминала есть данные для передачи.
CTS	5	8	<b>(Clear To Send)</b> Вход разрешения передачи данных терминалу. Состояние "выключено" аппаратно запрещает передачу данных. Применяется для аппаратного управления потоками данных.
DTR	20	4	<b>(Data Terminal Ready)</b> Выход: "терминал готов к передаче данных"
DSR	6	6	<b>(Data Set Ready)</b> Вход сигнала готовности от аппаратуры передачи данных
DCD	8	1	<b>(Data Carrier Detect)</b> Вход сигнала обнаружения несущей удаленного модема
RI	22	9	<b>(Ring Indicator)</b> Вход индикатора вызова (звонка)

В таблице 3 приведена распайка кабеля для подключения модема. Такой кабель иногда называют прямым (из-за того, что в нем соединяются одноименные контакты). Если устройства **DTE** соединяются между собой без аппаратного модема, то используются так называемые нуль-модемные кабели, имеющие на обоих концах розетки. Контакты таких кабелей соединяются перекрестно, причем некоторые линии могут не соединяться друг с другом (так называемый минимальный кабель, недостающие линии эмулируются перемычками на разъемах). Пример распайки нуль-модемного кабеля приведен в таблице 4.

**Таблица 3. Распайка кабеля для подключения модема**

Сигнал	DB-9S	DB-25S	Направление	DB-25S	DB-9S	Сигнал
TxD	3	2	⇒	2	3	TxD
RxD	2	3	⇐	3	2	RxD
DTR	4	20	⇒	20	4	DTR
DSR	6	6	⇐	6	6	DSR
RTS	7	4	⇒	4	7	RTS
CTS	8	5	⇐	5	8	CTS
DCD	1	8	⇐	8	1	DCD
RI	9	22	⇐	22	9	RI
SG	5	7	↔	7	5	SG

**Таблица 4. Распайка нуль-модемного кабеля для соединения устройств DTE**

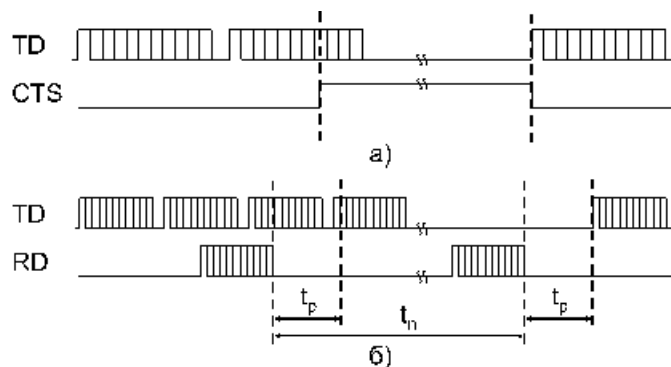
Сигнал	DB-9S	DB-25S	Направление	DB-25S	DB-9S	Сигнал
TxD	3	2	⇒	3	2	RxD
RxD	2	3	⇐	2	3	TxD
RTS	7	4	⇒	5	8	CTS
CTS	8	5	⇐	4	7	RTS
DTR	4	20	⇒	6	6	DSR
				8	1	DCD
DSR	6	6	⇐	20	4	DTR
DCD	1	8				
SG	5	7	↔	7	5	SG

Поскольку приёмная и передающая части адаптера работают независимо друг от друга, существует возможность проверки последовательного интерфейса, заключающаяся в том, что передаваемые им сигналы направляются непосредственно на вход собственного приемника. Для осуществления такого режима выход данных соединяется со входом (TxD → RxD), а

линии квитирования соединяются следующим образом: **RTS** → **CTS**; **DTR** → (**DCD** + **DSR**).

Для управления потоком данных могут использоваться два вида протоколов — аппаратный и программный. Аппаратный — предполагает обмен дополнительными сигналами для согласованной работы приёмника и передатчика, программный же с этой целью использует вставку в информационный поток специальных служебных байт.

**Аппаратный протокол управления RTS/CTS.** Использует сигнал **CTS**, который позволяет остановить передачу данных, если приемник не готов к работе. Передача данных по этому протоколу показана на рис. 2 а.



**Рис. 2.** Временные диаграммы обмена: а) аппаратный протокол управления б) программный протокол управления

Байт, передаваемый на момент прихода **CTS**, будет передан, однако с момента окончания его передачи передатчик переходит к ожиданию готовности приемника (т.е. снятия **CTS**). Обеспечивает самую быструю реакцию передатчика на состояние приемника, позволяет организовать обмен, не прибегая к буферизации. При соединении двух ПК необходимо перекрестное соединение выводов **RTS** → **CTS**. Если аппаратный протокол обмена не используется, то на линию **CTS** ПК необходимо подать сигнал "включено", что обычно достигается соединением **CTS** ПК с его же **RTS** перемычкой на разъеме. Аппаратный обмен через минимальный нуль-модемный кабель, в котором не задействованы сигналы квитирования, невозможен.

**Программный протокол XON/XOFF.** Предполагает наличие двунаправленного канала обмена. Временные диаграммы обмена показаны на рис. 2 б. Предполагает наличие у приемника буфера, так как время реакции передатчика  $t_p$  может оказаться достаточно большим. Когда буфер приемника заполняется до определенного уровня (обычно 80-90%), он передает на приемник команду **XOFF** (байт с кодом 13H). Приняв этот код, передатчик прекращает передачу и переходит в состояние ожидания до прихода команды **XON** (байт с кодом 11H), по которому передатчик возобновляет передачу.

**Программный протокол EXT/ACK.** При обмене по этому протоколу приемник посылает передатчику команду **ACK** (байт с кодом 06H) для со-

общения о готовности к приёму информации. В ответ передатчик посылает приемнику один байт или пакет байт определенного размера. Управляющий код **ЕХТ** (байт с кодом 03Н) передается приемником для запроса паузы в передаче.

## 1.2. АРХИТЕКТУРА БИС АДАПТЕРА ПОСЛЕДОВАТЕЛЬНОГО ИНТЕРФЕЙСА K580BB51

БИС адаптера последовательного интерфейса K580BB51 представляет собой универсальный синхронно–асинхронный приемопередатчик и предназначена для организации обмена между микропроцессором и внешними устройствами в последовательном формате. УСАПП может принимать данные с 8-разрядной шины данных МП и передавать их в последовательном формате периферийным устройствам, а также получать последовательные данные от периферии и преобразовывать их в параллельную форму для передачи в МП. Обмен данными производится в асинхронном режиме со скоростью передачи до 9600 бит/с или в синхронном – со скоростью до 56 Кбит/с. Длина передаваемых символов составляет от 5 до 8 бит. При передаче в МП символов длиной менее 8 бит неиспользуемые биты заполняются нулями. Формат символа включает также служебные биты и необязательный бит контроля по четности (нечетности).

Структурная схема адаптера последовательного интерфейса представлена на рисунке 3.

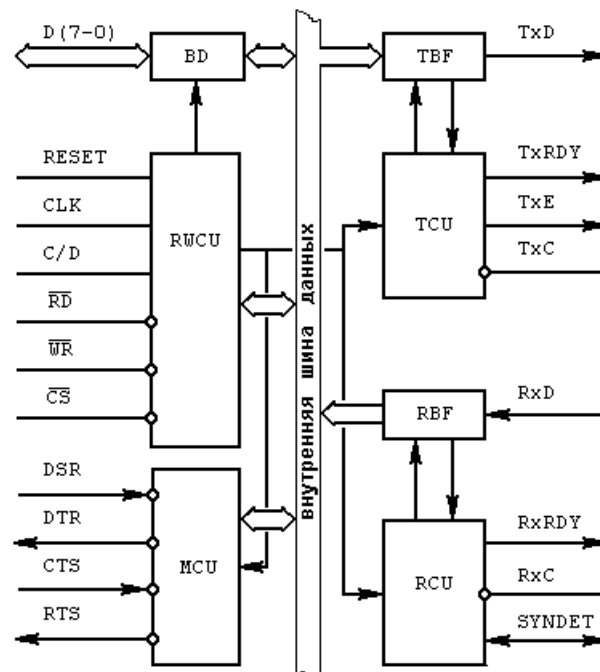


Рис. 3. Структура адаптера последовательного интерфейса K580BB51

В состав БИС входят:

- буфер передатчика (TBF) со схемой управления передатчиком (TCU), предназначенные для приема данных от МП и выдачи их в последовательном формате на выход TxD;
- буфер приемника (RBF) со схемой управления приемником (RCU), выполняющие прием последовательных данных со входа RxD и передачу их в МП в параллельном формате;
- буфер данных (BD), представляющий собой параллельный 8-разрядный двунаправленный регистр с трехстабильными каскадами и служащий для обмена данными и управляющими словами между МП и УСАПП;
- блок управления записью/чтением (RWCU), принимающий управляющие сигналы от МП и генерирующий внутренние сигналы управления;
- блок управления модемом (MCU), обрабатывающий управляющие сигналы, предназначенные для ВУ.

Назначение входных, выходных и управляющих сигналов УСАПП приведено при описании выводов микросхемы в таблице 5.

**Таблица 5. Описание выводов УСАПП**

Обозначение	Номер контакта	Назначение вывода
- 1 -	- 2 -	- 3 -
<b>D(7 - 0)</b>	8;7;6;5;2;1;28;27	Канал данных
<b>RESET</b>	21	Установка "0"(исходное состояние)
<b>CLK</b>	20	Синхронизация
<b>C/D</b>	12	Управление/данные - напряжение - L-уровня указывает на запись (чтение) данных в(из) БИС; напряжение H-уровня указывает на запись управляющих сигналов или чтение слова-состояния в(из) БИС
<b>RD</b>	13	Чтение-разрешение вывода данных или слова-состояния из УСАПП на шину данных МП
<b>WR</b>	10	Запись-разрешение ввода информации с шины данных в УСАПП
<b>CS</b>	11	Выбор микросхемы - подключение УСАПП к шине данных МП
<b>DSR</b>	22	Готовность передатчика терминала
<b>DTR</b>	24	Запрос передатчика терминала
<b>CTS</b>	17	Готовность приемника терминала
<b>RTS</b>	23	Запрос приемника терминала
<b>RxC</b>	25	Синхронизация приемника (по входу TxD)
<b>RxRDY</b>	14	Готовность приемника
<b>RxD</b>	3	Вход приемника

- 1 -	- 2 -	- 3 -
<b>SYNDET</b>	16	Вид синхронизации: для синхронного режима выходное напряжение Н-уровня - признак внутренней синхронизации; для синхронного режима с внешней синхронизацией сигнал является входным; в асинхронном режиме сигнал является выходным
<b>TxC</b>	9	Синхронизация передатчика (по входу TxD)
<b>TxE</b>	18	Конец передачи: напряжение Н-уровня признак окончания посылки данных
<b>TxRDY</b>	15	Готовность передатчика
<b>TxD</b>	19	Выход передатчика
<b>Ucc</b>	26	Напряжение питания (+5 В ±5%)
<b>GND</b>	4	Общий вывод (0 В)

Основные сигналы управления работой УСАПП подаются на блок **RWCU** от МП и определяют вид обрабатываемой информации и направление передачи в соответствии с таблицей 6.

**Таблица 6. Операции, определяемые сигналами управления от МП**

Операция	C/D	RD	WR	CS
Чтение данных из УСАПП на D(7 - 0)	0	0	1	0
Запись данных с D(7 - 0) в УСАПП	0	1	0	0
Чтение слова-состояния из УСАПП на D(7 - 0)	1	0	1	0
Запись управляющего слова с D(7 - 0) в УСАПП	1	1	0	0
Отключение УСАПП от D(7 - 0)	X	1	1	0
Отключение УСАПП от D(7 - 0)	X	X	X	1

*Примечание. X- безразличное состояние сигнала*

### 1.3. ПОДКЛЮЧЕНИЕ УСАПП К МАГИСТРАЛЯМ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

Условное графическое изображение УСАПП и схема подключения к магистралям микропроцессорной системы показаны на рисунке 3. Применение программируемого таймера КР580ВИ53 в режиме делителя частоты позволяет выбирать любую скорость передачи из полного стандартного ряда с неточностью не более 1%.

Через выводы **D0÷D7** буфера канала данных УСАПП подключается к одноименным линиям магистрали данных непосредственно или через буферный элемент типа К580ВА86 в том случае, если нагрузочной способности буфера канала данных недостаточно для работы с шиной данных сис-

темы. Вход  $C/D$ , как правило, подключается к младшим разрядам магистрали адреса (МА)  $A0$  (или  $A1$ ), а поступивший на них код задает следующий порядок подключения буфера канала данных: 1 – к управляющему регистру, 0 – к регистру данных. Код на остальных адресных линиях определяет выбранную микросхему.

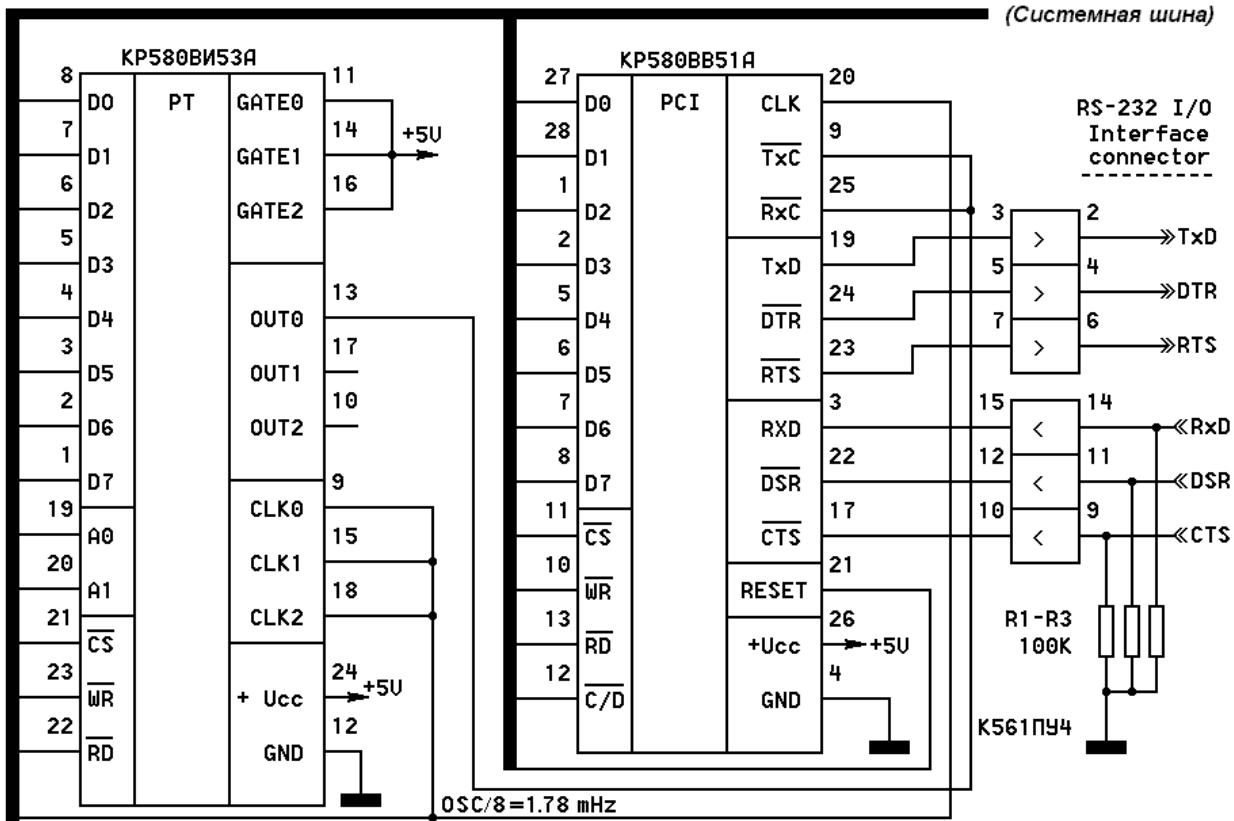


Рис. 2. Схема подключения УСАПП к магистралям микропроцессорной системы

Сигналы выборки  $\overline{CS0}, \overline{CS1}$  таймера и УСАПП с низким активным уровнем формируются схемой дешифратора  $DC$ , который может быть выполнен на микросхемах комбинационной логики, компараторах кодов и дешифраторах в интегральном исполнении. В МПС, имеющих в своем составе контроллер прямого доступа к памяти (ПДП), работу дешифратора выборки следует блокировать на время циклов ПДП по активному уровню сигнала  $AEN$  (Address Enable), поскольку в этот промежуток времени сигналы магистрали управления (МУ) формируются контроллером ПДП, а не центральным процессором. В МПС, построенных на МП  $K580BM80$ , дешифрация устройств ввода/вывода может быть упрощена в том случае, если их количество не превышает шести. Поскольку МП  $K580BM80$  при обращении к внешним устройствам командами  $IN$  и  $OUT$  дублирует на линиях  $A15 \div A8$  код, выставяемый по линиям  $A7 \div A0$ , входы  $\overline{CS}$  микросхем внешних устройств могут подключаться непосредственно к линиям



**A15÷A10** MA без дополнительной дешифрации и однозначно адресоваться кодами 111110XXb, 111101XXb, 111011XXb, 110111XXb, 101111XXb, 011111XXb.

Входы  $\overline{WR}$  и  $\overline{RD}$  определяют направление передачи данных и подключаются к линиям  $\overline{IOWR}$ ,  $\overline{IORD}$  МУ в том случае, если УСАПП подключен как устройство ввода/вывода, или к линиям  $\overline{MEMWR}$ ,  $\overline{MEMRD}$  – если регистры УСАПП включены как ячейки памяти. В простых микропроцессорных системах, в которых пространство ячеек памяти и пространство устройств ввода/вывода не разделены, входы  $\overline{WR}$  и  $\overline{RD}$  подключаются непосредственно к соответствующим выводам микропроцессора.

При включении УСАПП в пространство устройств ввода/вывода программирование и обмен данными с ним осуществляется командами ассемблера IN и OUT. При работе с регистрам УСАПП как с ячейкам оперативного запоминающего устройства (ОЗУ), в полном объеме применимы команды обращения к памяти. В том случае, если пространство ячеек памяти и пространство устройств ввода/вывода не разделены, доступ к регистрам УСАПП возможен как с помощью команд обращения к памяти, так и с помощью команд ввода/вывода с учетом специфики работы последних применительно к конкретному типу процессора. Так команды IN 35H и OUT 35H МП К580ВМ80 выполняются в этом случае эквивалентно командам LDA 3535H и STA 3535H.

Вход установки БИС в исходное состояние **RESET** подключается к линии начального сброса **RESET** магистрали управления МПС.

Для связи с периферийными устройствами используются 2 линии обмена данными, а также сигналы управления, состояние которых определяется и опрашивается программным способом. Нагрузочная способность выводов УСАПП такова, что позволяет подключать к каждому выходу только один вход ТТЛ элементов, поэтому для согласования с передающей линией необходимо вводить специальные буферные элементы, роль которых в данной схеме выполняют элементы ИС 561ПУ4.

#### 1.4. НАЧАЛЬНАЯ ИНИЦИАЛИЗАЦИЯ АДАПТЕРА

Режим работы УСАПП задается программно путем загрузки в него управляющих слов из МП. Различаются управляющие слова 2-х видов: инструкции режима и команды. Формат инструкции режима для асинхронного обмена представлен на рисунке 3.

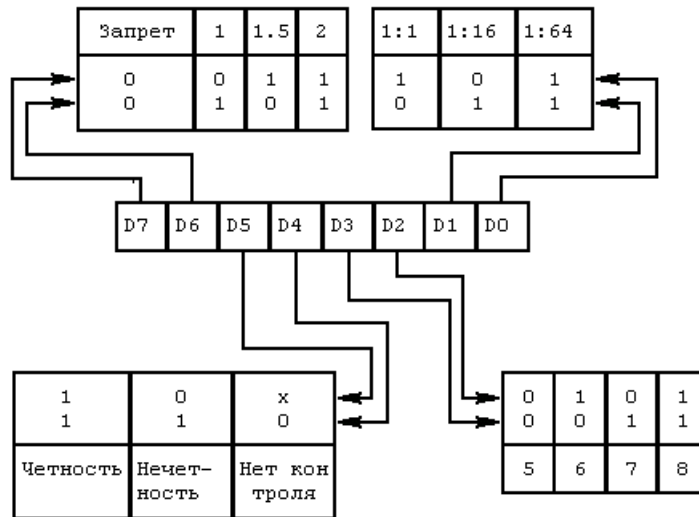


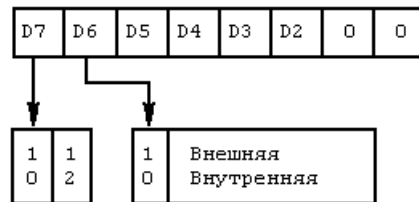
Рис. 3. Формат инструкции режима для асинхронного обмена

Инструкция режима задает синхронный или асинхронный режим работы, формат данных, скорость приема или передачи, необходимость контроля. Инструкция заносится сразу после установки УСАПП в исходное состояние программно или по сигналу RESET и заменяется лишь при смене режима. Команда осуществляет управление установленным режимом обмена и может многократно задаваться в процессе обмена, управляя различными его этапами. При асинхронном обмене команда загружается сразу же после инструкции режима, а при синхронном обмене перед ней располагаются один или два синхросимвола. Ограничения на последовательность загрузки управляющих слов связаны с внутренней организацией УСАПП. В асинхронном режиме работы формат данных включает нулевой старт-бит, биты данных, контрольный бит и стоп-биты. Число битов данных и стоп-битов, а также наличие или отсутствие бита контроля задаются инструкцией режима.

Разряды D0 и D1 определяют три разновидности асинхронного режима по частоте сигналов синхронизации (с частотой сигналов синхронизации,  $1/16$  и  $1/64$  частоты синхронизации). Разряды D3 и D2 определяют число битов данных. Режим контроля задается разрядами D5 и D4: при D4=0 контроль по четности запрещен; значение разряда D5 устанавливает

вид контроля - по четности или нечетности. Разряды D7 и D6 определяют число передаваемых стоп-битов.

При синхронном обмене данные передаются в виде массивов слов, а для синхронизации запуска при приеме данных используются один или два символа синхронизации. Формат инструкции режима для синхронного обмена представлен на рисунке 4.



**Рис. 4.** Формат инструкции режима для синхронного обмена

Разряды D1 и D0 для синхронного режима должны иметь нулевое значение. Разряд D6 устанавливает вид синхронизации (внешняя или внутренняя). Разряд D7 определяет использование одного (D7=1) или двух (D7=0) символов синхронизации. Назначение разрядов D3, D2 и D5, D4 - такое же, как при асинхронном обмене.

Команды подаются на УСАПП после инструкции режима и управляют выполнением конкретных операций. Назначение отдельных разрядов команд управления УСАПП поясняется в таблице 7.

**Таблица 7.** Назначение разрядов команд управления УСАПП

Разряд	Назначение разряда (обозначение)	Пояснение
D0	Разрешение передачи (TxEN)	Передача информации невозможна при D0=0 и возможна при D0=1
D1	Запрос о готовности передатчика	Запись 0 на выходе с DTR при D1=1 терминала к передаче (DTR)
D2	Разрешение приема (RxE)	Прием информации невозможен при D2=0 и возможен при D2=1
D3	Конец передачи (SBRK)	При D3=0 нормальная работа канала передачи, при D3=1 H-уровень на TxD
D4	Установка ошибок (ER)	При D4=1 установка разрядов ошибок в исходное состояние
D5	Запрос о готовности приемника	Запись 0 на выходе RTS при D5=1 терминала к приему (RTS)
D6	Программный сброс схемы в исходное состояние (IR)	При D6=1 УСАПП установлен в исходное состояние и готов к приему инструкции режима
D7	Режим поиска синхросимволов (EH)	При D7=1 установлен режим поиска символов синхронизации

Для контроля состояния УСАПП в процессе обмена данными МП может с помощью команды ввода считывать слово-состояние БИС из специального внутреннего регистра состояний. Значение управляющих сигналов при чтении слова-состояния указано в таблице 6. Формат слова-состояния приведен на рисунке 5.

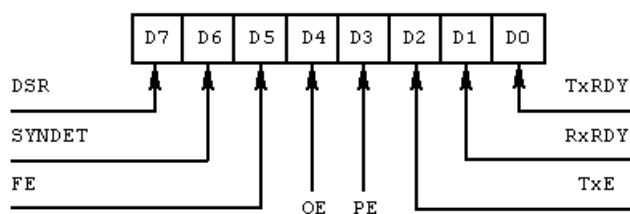


Рис. 5. Формат слова-состояния УСАПП

Кроме сигналов, рассмотренных в таблице 5, в слове-состоянии формируются три флага ошибок:

- разряд **D3** устанавливается при возникновении ошибки четности (**PE – Parity Error**);
- разряд **D4** устанавливается при возникновении ошибки переполнения (**OE – Overrun Error**), если МП не прочитал символ;
- разряд **D5** устанавливается при наличии ошибки стоп-бита (**FE – Framing Error**), если в конце посылки для асинхронного режима не обнаруживается стоп-бит.

## 1.5. РЕЖИМЫ РАБОТЫ АДАПТЕРА

После записи инструкции режима и команды УСАПП готов к выполнению обмена данными в одном из пяти режимов: синхронная передача; синхронный прием с внутренней синхронизацией; синхронный прием с внешней синхронизацией; асинхронная передача; асинхронный прием.

**При синхронной передаче** данных на выходе **TxD** с частотой сигнала синхронизации формируется последовательность, начинающаяся с синхросимволов, запрограммированных инструкцией режима. Затем передаются поступающие из МП коды символов, каждый из которых может заканчиваться битом контроля. Если МП не загрузил очередной символ к моменту передачи, то УСАПП вставляет в передаваемую последовательность синхросимволы, а на выходе **TxE** вырабатывается сигнал Н(High)-уровня, идентифицирующий пустую передачу.

**При синхронном приеме** с внутренней синхронизацией УСАПП начинает работу с поиска во входной последовательности синхросимволов. УСАПП сравнивает записанные в него при настройке синхросимволы с

принимаемыми символами. После обнаружения синхросимволов устанавливается сигнал Н-уровня на выводе **SYNDET** и начинается прием входных данных (рис. 6). Сигнал на выводе **SYNDET** автоматически сбрасывается при чтении слово-состояние УСАПП.

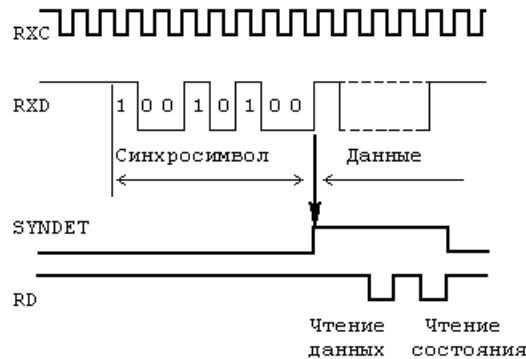


Рис. 6. Временные диаграммы синхронного обмена

*При синхронном приеме с внешней синхронизацией* на вывод **SYNDET** подается сигнал от внешнего устройства, который разрешает прием данных на входе **RxD** со скоростью сигналов синхронизации, поступающих на вход **RxC**.

Возможна организация приема данных в МП по прерыванию, если сигнал на выводе **SYNDET** использовать как запрос прерывания.

*При асинхронной передаче* последовательные данные формируются на выходе **TxD** по спаду сигнала синхронизации **TxC** с периодом, задаваемым инструкцией режима и равным 1, 16 или 64 периодам сигнала синхронизации. Если после передачи символа следующий символ отсутствует, то на выходе **TxE** устанавливается напряжение Н-уровня, пока новые данные не поступят от МП. В программе, реализующей алгоритм асинхронной передачи, запись очередного байта в УСАПП производится по команде вывода (**OUT**), если в слово-состоянии разряд  $D0=1$  что соответствует Н-уровню сигнала на выходе **TxRDY** используется как сигнал запроса прерывания.

*Асинхронный прием данных* начинается с поиска старт-бита который устанавливает на входе **RxD** напряжение L(Low)-уровня. Наличие этого бита вторично проверяется стробированием его середины внутренним строб импульсом. Если старт-бит найден то запускается внутренний счетчик битов который определяет начало и конец битов данных, бит контроля и стоп-биты. Прием стоп-бита идентифицирует окончание приема байта информации и сопровождается установкой сигнала Н-уровня на выходе **RxRDY**.

В программе реализующей алгоритм асинхронного приема передача очередного байта данных в МП может производиться по команде ввода (**IN**) если в слово-состоянии разряд  $D1=1$  что соответствует Н-уровню сигнала

на выходе **RxRDY** или по прерыванию если сигнал на выходе **RxRDY** используется как сигнал запроса прерывания.

## 1.6. ПРОГРАММИРОВАНИЕ БИС УСАПП

Программирование БИС УСАПП может вестись на физическом или логическом уровне. Программирование на физическом уровне производится на языках низкого уровня или в машинных кодах. Логический уровень программирования обеспечивается алгоритмическими языками высокого уровня, коммуникационными программами, некоторыми пакетами прикладных программ. Следует отметить, что в реальных микропроцессорных системах необходимость в низкоуровневое программирование УСАПП возникает крайне редко, поскольку подпрограммы управления им входят в большинство стандартных библиотек и часто бывают оформлены в виде инструкций языка высокого уровня.

Программу двухстороннего обмена можно рассматривать как три самостоятельных подпрограммы:

- настройка БИС УСАПП,
- прием данных,
- передача данных.

```

;+-----+
;|      Пример программирования УСАПП на языке ассемблер:      |
;|  ПОДПРОГРАММЫ ОБСЛУЖИВАНИЯ КОНТРОЛЛЕРА ПОСЛЕДОВАТЕЛЬНОГО  |
;|  ИНТЕРФЕЙСА, ПОДКЛЮЧЕННОГО К СИСТЕМНЫМ ШИНАМ (Рис. 2)    |
;+-----+
;
;          ORG   SETIT
;+-----+
;|          ПОДПРОГРАММА НАСТРОЙКИ КОНТРОЛЛЕРА                |
;+-----+
;
INIT:
        LXI    H, KDIV
        MVI    A, MI
INIT1:
        PUSH   PSW
;-----
        НАСТРАИВАЕМ ТАЙМЕР
        MVI    A, SCO+RLW+MODE3
        STA    CW53
        MOV    A, L
        STA    CTO
        MOV    A, H
        STA    CTO
;

```

```

; _____ УСТАНОВЛИВАЕМ УСАПП В ИСХ. СОСТОЯНИЕ
      MVI      A,01H
      STA      CW51
      STA      CW51
      MVI      A,IR
      STA      CW51
; _____ ЗАПИСЫВАЕМ ИНСТРУКЦИЮ РЕЖИМА
      POP      PSW
      STA      CW51
; _____ ЗАПИСЫВАЕМ ИНСТРУКЦИЮ КОМАНДЫ
      MVI      A, TXEN+DTR+RXE+RTS
      STA      CW51
;      RET

```

```

;+-----+
;|                ПОДПРОГРАММА ПЕРЕДАЧИ БАЙТА ИЗ РЕГИСТРА C                |
;+-----+
;

```

```

TXD:   PUSH     PSW
; _____ ЖДЕМ ГОТОВНОСТИ
TX1:   LDA      CW51
      ANI      TXRDY+DSR
      CPI      TXRDY+DSR
      JNZ      TX1
; _____ ПЕРЕДАЕМ БАЙТ
      MOV      A,C
      STA      DAT51
      POP      PSW
      RET

```

```

;
;+-----+
;|                ПОДПРОГРАММА ПРИЕМА БАЙТА В АККУМУЛЯТОР                |
;+-----+
;

```

```

RXD:
; _____ ПРОВЕРЯЕМ ГОТОВНОСТЬ
      LDA      CW51
      ANI      RXRRY
; _____ ВОЗВРАТ С ФЛАГОМ ПЕРЕНОСА,
; _____ ЕСЛИ ПРИЕМНИК НЕ ГОТОВ
      STC
      RZ
; _____ ЧИТАЕМ ПРИНЯТЫЙ БАЙТ
      LDA      DAT51
      SMC
      RET

```

```

;
;+-----+
;|----- Внешние метки и константы -----|
;+-----+

```

```

SETIT:      EQU      0D400H      ; НАЧ.АДРЕС ПОДПРОГРАММ

```

```

; _____ АДРЕСА РЕГИСТРОВ УСАПП:
DAT51: EQU 0A800H ; РЕГИСТР ДАННЫХ ВВ51
CW51: EQU 0A801H ; РЕГИСТР КОМАНД ВВ51

```

```

; _____ АДРЕСА РЕГИСТРОВ ТАЙМЕРА:
CT0: EQU 0A400H ; СЧЕТЧИК 0 ВИ53
CT1: EQU 0A401H ; СЧЕТЧИК 1 ВИ53
CT2: EQU 0A402H ; СЧЕТЧИК 2 ВИ53
CW53: EQU 0A403H ; РЕГИСТР КОМАНД ВИ53

```

```

; _____ УПРАВЛЯЮЩЕЕ СЛОВО ВИ53
BCD: EQU 01H ; ДВОИЧНО-ДЕСЯТИЧНЫЙ СЧЕТ
MODE0: EQU 00H
MODE1: EQU 02H
MODE2: EQU 04H ; ВЫБОР
MODE3: EQU 06H ; РЕЖИМА
MODE4: EQU 08H
MODE5: EQU 0AH

```

```

;
ROF: EQU 00H ; ЧТЕНИЕ НА ЛЕТУ
RLL: EQU 10H ; ЧТЕНИЕ/ЗАПИСЬ МЛ. БАЙТА
RLH: EQU 20H ; ЧТЕНИЕ/ЗАПИСЬ СТ. БАЙТА
RLW: EQU 30H ; ЧТЕНИЕ/ЗАПИСЬ СЛОВА
SCO: EQU 00H ; |
SC1: EQU 40H ; > ВЫБОР СЧЕТЧИКА
SC2: EQU 80H ; |

```

```

; _____ ИНСТРУКЦИЯ КОМАНДЫ ВВ51
TXEN: EQU 01H ; ПЕРЕДАТЧИК ВКЛЮЧЕН
DTR: EQU 02H ; УСТРОЙСТВО ГОТОВО
RXE: EQU 04H ; ПРИЕМНИК ВКЛЮЧЕН
SBRK: EQU 08H ; ПРЕРЫВАНИЕ ПЕРЕДАЧИ
ER: EQU 10H ; СБРОС ОШИБОК ПРИЕМА
RTS: EQU 20H ; ПЕРЕДАЧА РАЗРЕШЕНА
IR: EQU 40H ; ПРОГР. СБРОС УСАПП
EH: EQU 80H ; РАЗРЕШЕНИЕ ПОИСКА
; СИНХРОСИМВОЛА

```

```

; _____ РЕГИСТР СОСТОЯНИЯ ВВ51
TXRDY: EQU 01H ; ПЕРЕДАТЧИК ГОТОВ
RXRDY: EQU 02H ; ПРИЕМНИК ГОТОВ
TXE: EQU 04H ; ПЕРЕДАЧА ЗАКОНЧЕНА
PE: EQU 08H ; ОШИБКА ЧЕТНОСТИ
OE: EQU 10H ; ПЕРЕПОЛНЕНИЕ ПРИЕМНИКА
FE: EQU 20H ; ОШИБКА ФОРМАТА
SYNDET: EQU 40H ; СИНХРОСИМВОЛ НАЙДЕН
DSR: EQU 80H ; ПЕРЕДАТЧИК ДАННЫХ ГОТОВ

```

```

;
END

```

```

; *****

```



## 1.7. ПРОГРАММНАЯ МОДЕЛЬ БИС УАПП 8250

С БИС УАПП 8250 снизу–вверх совместимо подавляющее большинство микросхем – контроллеров СОМ порта современных ПК. Как уже было отмечено, эта микросхема осуществляет преобразование параллельного кода в последовательный при передаче и последовательного в параллельный – при приёме, контроль ошибок, формирование запросов прерывания и ряд других сервисных функций. Контроллеры, совместимые с БИС УАПП 8250 получили широкое распространение в схемотехнике современных микропроцессорных систем, что определяет необходимость обсуждения как её программной модели, так и вопросов, связанных с её программированием.

С точки зрения программиста контроллер последовательного интерфейса представляет собой набор регистров, занимающих смежные адреса в пространстве устройств ввода-вывода.

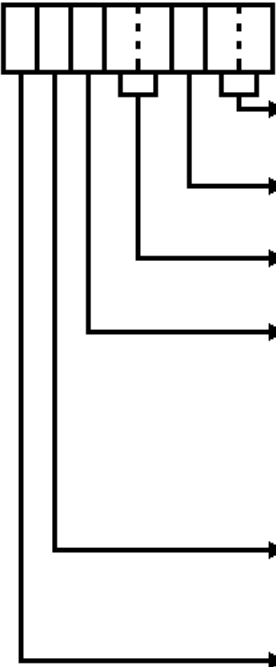


Базовый адрес СОМ порта определяется путем чтения переменных BIOS, которые последовательно занимают восемь смежных байт по смещению 0:0400H в области нулевого сегмента. Стандартным адресом порта СОМ1 является 3F8H, СОМ2 – 2F8H. Перед работой с портом рекомендуется определить его адрес путем чтения переменной BIOS, а не брать стандартное значение.

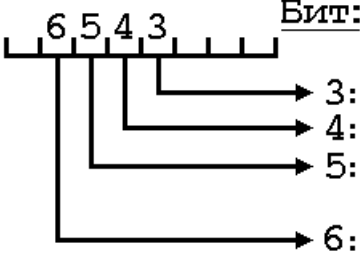
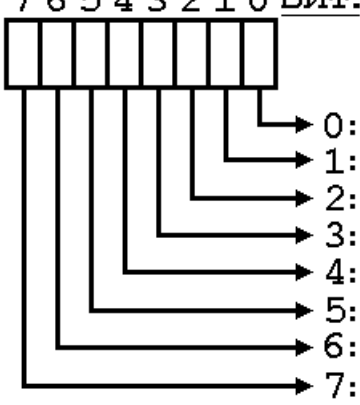
Ниже в таблице 8 приводится описание портов ввода/вывода для СОМ1, имеющего базовый адрес 3F8H. Микросхема УАПП 8250 имеет 10 программируемых однобайтовых регистров, с помощью которых управляется и контролируется порт коммуникации. Доступ к этим регистрам осуществляется через семь смежных адресов портов, начиная с базового. Следует обратить внимание, что порты 3F8H и 3F9H имеют разное назначение в зависимости от бита 7 порта 3F8H (так называемый бит DLAB – Divisor Latch Access Bit). Из десяти регистров только шесть необходимы для осуществления простой последовательной связи, при этом обычная коммуникационная процедура непрерывно проверяет состояние регистра статуса линии, ожидая вводимого символа или готовности к передаче байта.

**Таблица 8. Назначение регистров БИС УАПП 8250**

Порт	Операция	Описание
-1-	-2-	-3-
3F8h	Запись	<b>Регистр передатчика</b> (по этому адресу записывается байт для передачи)
3F8h	Чтение	<b>Регистр приемника</b> (по этому адресу читается принятый байт)

-1-	-2-	-3-																
3F8h	Запись	<p><b>Младший байт делителя скорости обмена</b>  (Если DLAB=1, то по этому адресу записывается младший байт значения делителя скорости обмена)</p>																
3F9h	Запись	<p><b>Старший байт делителя скорости обмена</b>  (Если DLAB=1, то по этому адресу записывается старший байт значения делителя скорости обмена).  <i>Скорость задается значением делителя, равным <math>115200/V</math>, где <math>V</math> – скорость в бодах.</i></p>																
3F9h	Запись	<p><b>Регистр управления прерываниями</b>  (1 в соответствующем бите — разрешить прерывание)</p>																
<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 20px;"> <p>7 6 5 4 3 2 1 0 Бит:</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 20px;">7</td><td style="width: 20px;">6</td><td style="width: 20px;">5</td><td style="width: 20px;">4</td><td style="width: 20px;">3</td><td style="width: 20px;">2</td><td style="width: 20px;">1</td><td style="width: 20px;">0</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td></td><td></td><td></td><td></td> </tr> </table> </div> <div> <p>0: прерывание по приему символа;  1: прерывание по завершению передачи символа;  2: прерывание по обрыву линии или по ошибке в линии;  3: прерывание по изменению состояния модема (любой из линий CTS, DSR, RI и DCD).</p> </div> </div>			7	6	5	4	3	2	1	0	0	0	0	0				
7	6	5	4	3	2	1	0											
0	0	0	0															
3FAh	Чтение	<p><b>Регистр идентификации прерывания</b>  (Отдельные биты регистра указывают причину вызова прерывания)</p>																
<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 20px;"> <p>7 6 5 4 3 2 1 0 Бит:</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 20px;">7</td><td style="width: 20px;">6</td><td style="width: 20px;">5</td><td style="width: 20px;">4</td><td style="width: 20px;">3</td><td style="width: 20px;">2</td><td style="width: 20px;">1</td><td style="width: 20px;">0</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td style="text-align: center;">⋮</td><td></td><td></td> </tr> </table> </div> <div> <p>0: = 0 — есть отложенные прерывания (= 1 — нет);  1-2: = 11 — ошибка или обрыв линии  (сбрасывается чтением регистра состояния линии — порта 3FDh);  = 10 — принят символ  (сбрасывается чтением регистра приемника — порта 3F8h);  = 01 — передан символ  (сбрасывается записью символа в регистр приемника — порт 3F8h);  = 00 — изменение состояния модема (линий CTS, DSR, RI или DCD)  (сбрасывается чтением регистра состояния модема — порта 3FEh).</p> </div> </div>			7	6	5	4	3	2	1	0	0	0	0	0	0	⋮		
7	6	5	4	3	2	1	0											
0	0	0	0	0	⋮													
-1-	-2-	-3-																

3FBh	Чтение/ Запись	Регистр управления линией:
<p>7 6 5 4 3 2 1 0 Бит:</p>  <p><b>число бит данных:</b> = 00 — 5, =01 — 6, =10 — 7, =11 — 8;</p> <p><b>число стоп-бит:</b> = 0 — 1, =1 — 1.5 при 5 битах данных (иначе — 2)</p> <p><b>тип четности:</b> = X0 — нет, =01 — нечетная, =11 — четная;</p> <p><b>постоянная четность</b> = 0 — отмена постоянной четности; = 1 — постоянный бит четности (зависит от битов 3-4): биты 3-4=01 - бит четности всегда 1; биты 3-4=11 - бит четности всегда 0; биты 3-4=X0 - без бита четности.</p> <p>6: = 1 — имитировать обрыв линии (посылка нулей);</p> <p><b>бит DLAB:</b> = 1 — порты 3F8h и 3F9h для загрузки скорости обмена; = 0 — порты 3F8h и 3F9h в обычном режиме.</p>		
3FCh	Запись	Регистр управления модемом
<p>7 6 5 4 3 2 1 0 Бит:</p>  <p>0: =1 — установить выход DTR;</p> <p>1: =1 — установить выход RTS;</p> <p>2: =1 — установить OUT1 (не используется);</p> <p>3: =1 — установить OUT2 (разрешить прерывания от RS-232);</p> <p>4: =1 — диагностический режим (перенаправить выход на вход контроллера).</p>		
3FDh	Чтение	Регистр состояния линии (биты 1-4 вызывают прерывания по ошибке <sup>1</sup> )
<p>7 6 5 4 3 2 1 0 Бит:</p>  <p>0: =1 — данные приняты (сбрасывается чтением приемника);</p> <p>1: =1 — потеря предыдущего символа;</p> <p>2: =1 — ошибка четности;</p>		

-1-	-2-	-3-
 <p>Бит:</p> <p>3: =1 — неверный стоп-бит;  4: =1 — обнаружен обрыв линии;  5: =1 — сдвиговый регистр передатчика пуст (можно передавать следующий символ);  6: =1 — регистр передатчика пуст (нет обрабатываемых данных).</p>		
3FEh	Чтение	<b>Регистр состояния модема</b> (биты 0-3 вызывают прерывание по изменению состояния модема <sup>1</sup> )
 <p>Бит:</p> <p>0: =1 — изменилось состояние линии<sup>2</sup> CTS;  1: =1 — изменилось состояние линии DSR;  2: =1 — изменилось состояние линии RI;  3: =1 — изменилось состояние линии DCD;  4: состояние линии CTS;  5: состояние линии DSR;  6: состояние линии RI;  7: состояние линии DCD.</p>		
<b>Примечания:</b>		<sup>1</sup> — если соответствующее прерывание разрешено; <sup>2</sup> — данная линия изменила свое состояние по сравнению с последним чтением этого регистра.

## 1.8. ОСНОВЫ ПРОГРАММИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ НА ЯЗЫКЕ АССЕМБЛЕРА

Ассемблерные программы обладают преимуществом компактности кода и максимальным быстродействием, вследствие этого они чаще всего используются при работе микропроцессорных систем в режиме реального времени, а также при необходимости управления внешними устройствами или обмена данными с ними. Операционные системы МПС обычно содержат низкоуровневые подпрограммы (драйверы) для работы со стандартными устройствами ввода/вывода. Потребность в написании ассемблерной подпрограммы может быть вызвана заменой стандартного устройства, оп-

тимизацией кода существующих драйверов, необходимостью обслуживания нестандартной периферии.

Рассмотрим в качестве примера задачу управления через адаптер последовательного ввода/вывода матричным принтером по протоколу RS-232C и выведем блок кодов ASCII заданного размера на печать.

### 1.8.1. Формулировка задачи

*Первым этапом решения задачи является составление алгоритма*, включающего в себя как словесное описание, так и любые виды формализации — таблицы, формулы, графики, принципиальные и блок-схемы. Этап формулирования задачи является одним из самых важных, поскольку ошибки, допущенные в процессе составления алгоритма, сводят на нет все дальнейшие усилия по составлению программы и кодированию.

Понятие интерфейса RS-232C относится к набору сигналов, протоколу обмена соединительному кабелю и разъемам, устанавливаемым в принтерах и ПК. Большинство современных принтеров наряду со стандартным параллельным интерфейсом поддерживают управление и по последовательному каналу. Выбор необходимого интерфейса обычно осуществляется настройкой принтера с помощью специальных переключателей или переключков ("джамперов"), назначение которых приводится в инструкции по эксплуатации к печатающему устройству.

Подготовка контроллера последовательного порта к работе сводится к выполнению следующих операций (для определенности считаем, что регистры УАПП расположены в пространстве устройств ввода-вывода, начиная с базового адреса 3F8H):

1. Установить бит DLAB порта 3FBH и записать значение делителя, задающего скорость обмена, в порты 3F8H и 3F9H.
2. Инициализировать регистр управления линией (порт 3FBH), при этом сбросить бит DLAB.
3. Инициализировать регистр управления модемом (порт 3FCH).
4. Инициализировать регистр управления прерываниями (порт 3F9H) и, если прерывания разрешены, установить адрес программы обработки прерываний от последовательного интерфейса. В случае если механизм прерываний не используется, следует опрашивать готовность контроллера программно.

Передача байта предполагает выполнение следующих операций:

1. Циклический опрос регистра состояния с проверкой бита готовности к передаче очередного байта.
2. Запись байта по адресу регистра передатчика.

**Определение задачи:** вывод блока кодов на печать через порт последовательного интерфейса по протоколу RS-232C.

**Устройства:** принтер, адаптер интерфейса, клавиатура.

**Датчики и сигналы:** матрица клавиш, опрашиваемая через порт клавиатуры; линии последовательного интерфейса, программно опрашиваемые и аппаратно обслуживаемые БИС УАПП.

**Суть задачи:** Настроить порт последовательного ввода/вывода, указав необходимую скорость обмена, длину передаваемых символов, длительность стоп-бита, вид контроля по четности (нечетности). Считать байт статуса УАПП, проверить его на готовность к передаче. Передать принтеру код инициализации. Считать байт статуса УАПП, проверить его на ошибку. В случае ошибки – выйти из программы с необходимым сообщением. В отсутствие ошибки – взять байт по начальному адресу блока и передать его на печать. Снова считать байт статуса УАПП и проверить его на ошибку. Если байт передан успешно, повторить процедуру передачи со следующим байтом пока не достигнут конец блока кодов. Обеспечить выход из процедуры по нажатию клавиши [ESC] матрицы клавиатуры.

Задача поставлена. Она содержит работу с внешними устройствами в реальном времени, программирование периферии и портов ввода/вывода.

### 1.8.2. Решение задачи

Следует определить для себя ответы на ряд вопросов следующего характера:

- а) какие данные вводятся или выводятся;
- б) какие порты используются и в какие режимы их запрограммировать;
- в) в каком виде осуществить диалог с оператором (если диалог нужен);
- г) какая информация и в какой форме необходима для вывода;
- д) возможное развитие задачи.

**Возможные ответы:** состояние линий интерфейса принтера преобразуется в установку соответствующих битов каналов УАПП. Начало блока данных определяет регистр SI, размер блока кодов определяет регистр CX. Если интерфейс реализован на БИС, совместимой с **INS8250**, осуществим следующую настройку:

**Скорость обмена 9600 бод, 1 стоп-бит;  
контроль четности, длина слова – 8 бит;  
протокол обмена – аппаратный.**

Поскольку принтер является медленным механическим устройством, выбранная сравнительно небольшая скорость обмена с ним вполне достаточна.

Для опроса клавиатуры с целью упрощения будем использовать стандартную подпрограмму опроса консоли, возвращающую код клавиши в случае её нажатия и код 00H – в противном случае (Клавиатура может опрашиваться разными программами, поэтому лучше не использовать опрос матрицы клавиш своей подпрограммой для совместимости). В качестве диалога с оператором или пользователем можно предусмотреть вывод на

экран сообщения типа: "Для прекращения печати нажмите клавишу [ESC]" ("Press ESC to terminate..."), но поскольку мы пишем процедуру низкого уровня, а 'ESC' – стандартная клавиша для отмены процесса, подобное предупреждение обычно выводится вызывающей программой. В качестве развития задачи можно предусмотреть подачу звукового сигнала, если принтер занят свыше определенного времени, что обычно свидетельствует об ошибке печати.

### 1.8.3 Проектирование программы

Существует множество различных методов проектирования программ. Метод “Сверху вниз” или “Нисходящее проектирование” является одним из наиболее простых и понятных. Суть метода заключается в том, что решаемая задача последовательно разбивается на части, каждая из которых – на более мелкие (называемые модулями). Разбиение прекращается, когда содержание модуля не вызывает видимых затруднений для программной реализации. Вполне вероятно, что отдельные модули могут в дальнейшем потребовать дополнительно и более мелкого разбиения.

Детализация разбиения и размер модулей величина субъективная, но предпочтительно, чтобы модули были небольшими, что связано с особенностями зрительного восприятия человека, и, по возможности – независимыми, для оперативной замены при внесении изменений в программу.

### 1.8.4. Проектирование модулей

При правильном проектировании программы простота функций и логики работы каждого модуля позволяют сразу же записать код на языке ассемблера. В противном случае можно описать алгоритм работы модуля на “псевдокоде” (то есть, на языке, близком к естественному). Указатель комментария “ ; ” позволяет реализовать такой подход в программах на ассемблере.

Правила “хорошего тона” в программировании предъявляют к оформлению и написанию отдельного модуля следующие требования:

- а) наличие заголовка, включающего имя вызова модуля, краткое описание выполняемой функции, а также входных и выходных данных;
- б) небольшой размер (для отладки в экранном режиме – не более 40 строк);
- в) возвращать управление по месту вызова;
- г) наличие одного входа и одного выхода;
- д) возможность обращения к другим модулям (универсальность отдельных модулей);
- е) по возможности меньшая взаимная зависимость (для оперативной замены);
- ж) наличие необходимых комментариев по тексту программы.

Выполнение последнего требования, безусловно, необходимо при коммерческой разработке программ и передаче программного обеспечения заказчику. Тем не менее, выполнение этого требования облегчает работу и с личными программами, позволяя всякий раз не затрачивать время на чтение кода и выяснение алгоритмов.

Перечисленные требования несут в себе некоторые элементы структурного программирования и являются в большей мере желательными, но не обязательными. Программа, написанная без учета данных требований, может быть вполне работоспособна, но трудоёмка в сопровождении и отладке.

### 1.8.5. Логическая структура ассемблерной программы

Элементы подпрограммы на языке ассемблера легче понять, если они расположены в той последовательности, в которой встречаются. Логической структуру ассемблерной программы в наиболее общем случае удобно представить в виде пяти вложенных друг в друга частей:

**Уровень 1:** общая надстройка ассемблера.

**Уровень 2:** надстройка ассемблерной подпрограммы.

**Уровень 3:** входной код.

**Уровень 4:** получение значений параметров вызывающей программы.

**Уровень 5:** выполнение кода программы, вызов служб ROM BIOS.

**Уровень 4:** передача результатов обратно вызывающей программе.

**Уровень 3:** выходной код.

**Уровень 2:** завершение надстройки ассемблерной подпрограммы.

**Уровень 1:** завершение общей надстройки ассемблера.

Данной базовой структуры можно придерживаться в большинстве интерфейсных программ, написанных для обращения к системным службам или просто в виде стандартных подпрограмм на ассемблере, однако следует иметь в виду, что конкретное кодирование будет видоизменяться в зависимости от языка программирования, используемого в вызывающей программе.

Для программ, написанных на языке ассемблера для процессора K580BM80 общий вид программы может быть примерно следующим:

```

;+-----+
;|          ОСНОВНЫЕ ИНСТРУКЦИИ ПРОГРАММЕ АССЕМБЛЕР          |
;+-----+
;
; Col.1 _____ - Поле меток и имен переменных
; |   Col.2 _____ - Поле (псевдо)оператора (директивы)
; |   |   Col.3 _____ - Поле операндов
; |   |   |   Col.4 _____ - Поле комментариев
;+--1--+ +--2--+ +--3--+ +--4-----+

```



```

;+1-+ +2-+ +3--+ +4-----+
      ORG 100H
;      | +---+          256 (100H или другое количест-
;      | +----- во байт) резервируются перед
;      |          началом программы...
;      +----- этой директивой, она же связы-
;              вает начало программы с абсо-
;              лютным адресом 100H
;+-----+
;|              Начало основной процедуры          |
;+-----+
;
begin:
; |          Метка (произвольна), сообщает
; +----- ассемблеру о начале выполнения
;          программы с данного адреса или
;          просто фиксирует начальный ад-
;          рес (может отсутствовать).

;+----- Тело основной процедуры -----+
;|
;  ...
;  ...
;|
;+-----+
;|              Объявление и инициализация данных          |
;+-----+
;          DB 32H, 5BH; (Размер: 1 байт, байты)
;          DW 0FFE1H; (Размер: 1 слово)
;          |_____ псевдооператор резервирования
;
;+-----+
;----- Внешние метки и константы -----
;
;      +----- псевдооператор присваивания
;      |
SYST: EQU 0F800H; _____ выход в систему
KLAV: EQU 0F81BH; _____ опрос клавиатуры без остановки программы
PRTA: EQU 0FFE4H; _____ адрес порта А
PRTB: EQU 0FFE5H; _____ адрес порта В
PRTC: EQU 0FFE6H; _____ адрес порта С
RGYS: EQU 0FFE7H; _____ адрес регистра управляющего слова

;+-----+
;|              Секция конца программы          |
;+-----+
;
;          END; <----- Директива конца основной процедуры
;+-----+

```

Оформим рассматриваемый нами алгоритм печати блока кодов на псевдокоде с элементами ассемблерной программы. При переводе всего алгоритма на язык ассемблера строки псевдокода автоматически становятся комментариями.

```

;----- Вывод блока кодов на печать -----
; [SI] - начало блока, [CX] - размер блока, [BL] - вывод. байт
;-----
SEG_A      SEGMENT BYTE PUBLIC
           ASSUME  CS:SEG_A, DS:SEG_A

           ORG 100H; _____ укажем транслятору начальный адрес

BLOUT      PROC FAR

BEGIN:
           JMP  START

AD_COM1:   DW 0000H; адрес последовательного порта COM1
BLEN:      DW 1000H; размер блока (вычисляется заранее)

START:
           CALL INIT; _____ осуществим начальную настройку УАПП
           PUSH CS
           POP  ES; _____ ES указывает на сегмент кода

;----- Программа вывода блока данных на печать -----
;
BLOCK:
           MOV AX,WORD PTR OFFSET INIP; коды инициализации
           MOV SI,AX; _____ занесем в SI адрес начала блока
           MOV CX,WORD PTR BLEN; занесем в CX размер блока
           ADD CX,02H; _____ учтем два кода инициализации
CYCLE:
           ; _____ цикл вывода байтов на печать
           MOV BL,ES:[SI]; в BL - очередной байт буфера
           CALL BOUT; _____ вывод на печать очередного байта
           CMP AL,1BH; _____ сравнить код с 1BH (была нажата [Esc]?)
           JZ  SYST; _____ нажата [Esc] при выводе байта на печать
           INC SI; _____ укажем следующий байт
           LOOP CYCLE; _____ возврат в цикл печати блока кодов
SYST:
           CMP AL,1BH; _____ сравнить код выхода с 1BH
           JNE SYS1; _____ не было выхода по нажатию [Esc]
           MOV AL,01H; _____ AL=01H - код завершения по нажатию [Esc]
           JMP SYST2; _____ выход с ERRORLEVEL = 1
SYS1:
           XOR AX,AX; _____ AL=00H - код успешного завершения
SYS2:
           MOV AH,4CH
           INT 21H; _____ выход в ОС с кодом завершения
;-----

```

```

;----- Подпрограмма вывода байта [BL] на печать -----
;----- опрос клавиатуры проводить для выхода из цикла -----
;
BOUT PROC NEAR
    MOV DX,WORD PTR AD_COM1; DX - базовый адрес порта COM1
    ADD DX,05H; _____ DX - адрес регистра состояния УАПП
WAIT:
    CALL CONS; _____ опрос консоли
    CMP AL,1BH; _____ сравнить код с 1BH
    JZ AWAY; _____ выход с кодом 1BH (нажата [Esc])
    IN AL,DX; _____ чтение байта состояния
    TEST AL,1FH; _____ проверка на ошибки
    JZ WAIT; _____ нет ошибок
    MOV AL,1BH; _____ ошибку не анализируем,
    JMP AWAY; _____ выходим с кодом 1BH (прервать передачу)
WAIT1:
    TEST AL,20H; _____ бит 5="1" (буфер передатчика пуст ?)
    JZ WAIT; _____ если нет - ждем когда будет пуст
    SUB DX,05H; _____ DX-адрес регистра передатчика
    MOV AL,BL; _____ байт для печати
    OUT DX,AL; _____ передача байта
    XOR AL,AL; _____ выход с кодом 00H (передача успешна)
AWAY:
    RETN; _____ возврат из п/программы с кодом завершения
BOUT ENDP

;----- Подпрограмма опроса консоли - Z=1 - нажата -----
;----- на выходе: код клавиши или 0, если клавиша не нажата --
;
CONS PROC NEAR
    MOV AH,01H
    INT 16H; _____ опросить клавиатуру, есть ли нажатие ?
    JNZ CON1; _____ есть нажатие, узнать код клавиши
CON1:
    XOR AX,AX; _____ нет нажатия, выходим с AL=00H
    JMP CON2
    MOV AH,00H
    INT 16H; _____ получить код клавиши с клавиатуры в AL
CON2:
    RETN; _____ если нажата то выход с кодом в AL
CONS ENDP

;----- Подпрограмма настройки УАПП в заданный режим -----
;--- (скорость 9600 бод, 1 стоп-бит, контроль четности, -----
;--- длина слова - 8 бит)
INIT PROC NEAR
    PUSH ES
    MOV AX,40H
    MOV ES,AX; _____ ES указывает на сегмент данных BIOS=0040H
    MOV DX,ES:[00];_ в DX - базовый адрес порта COM1
    POP ES

```

```

;-----
MOV WORD PTR AD_COM1,DX;_запомним его в ячейках памяти
ADD DX,03H;_____в DX - адрес регистра управления
MOV AL,80H;_____установим бит DLAB - настройка делителя
OUT DX,AL;_____скорость - 9600 бод
DEC DX;_____делитель = 115200/9600 = 12d = 000CH
DEC DX;_____DX-адрес старшего байта делителя скорости
MOV AL,00H;_____установка старшего байта делителя
OUT DX,AL
DEC DX;_____DX-адрес младшего байта делителя скорости
MOV AL,0CH;_____установка младшего байта делителя
OUT DX,AL
ADD DX,03H;_____DX-адрес регистра управления
MOV AL,00011011B;DLAB=0, четность, 1 стоп-бит, 8 бит
OUT DX,AL
DEC DX
DEC DX;_____DX-адрес регистра разрешения прерываний
MOV AL,00H;_____прерывания запрещены
OUT DX,AL
RETN;_____вернемся из подпрограммы.
INIT ENDP

INIP DB 1BH, 40H;_____коды инициализации принтера
BUFF DB 8000 DUP (41H);_____буфер блока кодов (заполнен 'A')

BLOUT ENDP
SEG_A ENDS
END BEGIN;_____укажем транслятору конец программы.
;-----

```

### 1.8.6. Отладка программы и исправление ошибок

Текст программы набирается в одном из текстовых редакторов, поддерживающих работу с Ассемблером (Edit, Z-Edit, Multi Edit и т.д.). Очень удобны в этой связи появившиеся в последнее время редакторы и интегрированные среды для работы с программами на Ассемблере, имеющие синтаксическую подсветку текста. Выделение разным цветом операторов, операндов, меток и комментариев позволяет нагляднее представить себе логическую структуру программы и уже на этапе набора избежать множества ошибок (ошибочно набранный оператор, к примеру, будет выделен редактором цветом операнда, что визуально хорошо заметно). Текстовые процессоры типа Microsoft Word менее пригодны для работы, поскольку обычно расставляют в тексте невидимые коды управления, не понятные транслятору. В крайнем случае, для работы может быть применена любая программа работы с текстами, имеющая опцию "Сохранить, как текст DOS".

Написанная на языке Ассемблер программа подлежит *трансляции* при помощи одной из *программ-трансляторов* в коды микропроцессора. Итогом работы транслятора с языка "ассемблер" (assemble (англ.) – собирать) в общем случае является либо объектный код в программном буфере (обычно указываются размер и адрес оттранслированного блока), либо файл объектного кода типа **\*.obj**. Распространенные программы-трансляторы различных фирм зачастую носят похожие названия типа ASMED, MASM, TASM.

В ходе компиляции транслятор выявляет *синтаксические ошибки* (то есть, по сути дела, проверяет грамматику – неизвестный оператор, недопустимый код, двойное определение метки, отсутствует метка и т.д.), и выдает сообщение в виде текста на дисплее или в соответствующем файле о числе и характере обнаруженных ошибок.

Код или текстовое сообщение о характере ошибки обычно приводятся с указанием строки программы, где эта ошибка была обнаружена. Интегрированные среды чаще всего указывают на ошибку непосредственно в тексте программы, что удобно для дальнейшего её исправления. Простые программы-компиляторы также позволяют сформировать текстовый файл программы, так называемый "листинг" (**\*.lst**), в котором, наряду с текстом программы и кодами микропроцессора, в соответствующих строках приводятся сообщения об ошибках, снабженные комментарием типа "\*\*\*Error...".

Если ошибок не выявлено, или они найдены и исправлены, программа обрабатывается редактором связей (LINK, TLINK), что может быть необязательной операцией, если транслятор ассемблера способен сам (и ему предписано это) сформировать машинный код.

Далее программа в машинных кодах может быть загружена в рабочую область и подлежит отладке для выявления *семантических ошибок*, к числу которых можно отнести неправильное логическое построение программы, не инициализированные стек или регистры, неверно заданные переменные счетчиков, то есть, все ошибки, допущенные самим программистом в ходе написания программы. Отладка программы осуществляется в *пошаговом режиме* с помощью программ-отладчиков (SID, DEBUG, Turbo Debugger, Code View, Soft Ice и др.). В том случае, если программа в кодах является управляющей для какого-либо законченного устройства, отладка осуществляется на инструментальной машине, а не на самом устройстве, при этом обычно используются кросс-средства, эмулирующие само устройство или прибор.

В процессе отладки на дисплей выдается протокол пошагового прохождения программы с указанием состояния всех флагов признаков, регистров, стека, соответствующих им ячеек памяти и их содержимого. Процесс позволяет проследить логику работы программы и выявить несоответствия разработанному алгоритму (счетчик считает больше чем задано, неверные условные переходы, подпрограмма искажает содержимое стека, ошибочные операнды и т.д.).

Следует отметить, что не все программы позволяют провести отладку в *статическом режиме*. Большинство программ, написанных на ассемблере, являются драйверами аппаратных устройств и поэтому возможны (и нередко) случаи, когда отладку приходится вести в *динамическом режиме*. Критические участки программы должны выполняться в режиме реального времени. Большинство отладчиков позволяют осуществить проверку программы в таком режиме при помощи соответствующих функций ("Выполнить до метки", "Выполнить подпрограмму"), вызываемых нажатием специальных клавиш или их комбинаций. Полностью динамическая отладка обычно ведется при помощи специальных приборов, таких как частотомер, осциллограф, логический анализатор, которые позволяют визуально проконтролировать результаты работы программы. В сложных случаях возможно написание специальных отладочных модулей, которые могут проверить необходимые временные задержки по встроенному таймеру или имитировать работу аппаратной части. Хорошие результаты даёт расстановка по ходу программы контрольных точек, перед остановом в которых по заданному признаку (прерывание от клавиатуры или по таймеру), программа сохраняет состояние рабочих регистров и необходимых операндов, после чего возможен их просмотр.

Преимуществом работы в статическом режиме отладки является возможность оперативно вмешиваться в работу программы, изменяя при необходимости содержимое регистров, ячеек памяти, стека. В процессе динамической отладки большинство этих функций доступны только в моменты остановки программы в контрольных точках.

Все обнаруженные *семантические ошибки* исправляются в исходном ассемблерном тексте, после чего программа снова транслируется и подвергается отладке. Работающий вариант программы, в котором все найденные ошибки уже исправлены, проверяется отладчиком на прохождение различных предельных случаев, которые при обычном выполнении могут не встретиться (достижение конца переносимых или передаваемых блоков данных, минимальные и максимальные конечные значения счетчиков, несоответствие версии управляющей операционной системы и т.п.).

Корректный вариант программы и исходный её текст сохраняются на внешнем носителе (магнитные и оптические диски, ленты, распечатки листинга) и при необходимости сопровождаются рабочей документацией.

Программы, разработанные для управления отдельными устройствами, записываются ("прошиваются") в микросхемы ПЗУ при помощи программаторов (отдельных устройств или приставок к ПК). Запрограммированные ПЗУ устанавливаются в разработанное устройство, после чего последнее готово к работе.

## 2. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

### 2.1. ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА

Порты последовательного ввода/вывода были первоначально разработаны для подсоединения информационных терминалов к большой ЭВМ через модемы. Последовательный интерфейс был введен в персональные компьютеры IBM для коммуникации с внешними устройствами, откуда и произошло его логическое название COM–порт (**C**ommunication port). Поскольку этот порт исторически поддерживает стандарт EIA — RS–232, его также называют RS–232 портом. В отдельных ранних версиях PC/XT и в некоторых отечественных IBM–совместимых персональных ЭВМ в этом качестве использовалась микросхема **UART INS8250** (или её отечественный аналог – KP1847BB2).

В ряде отечественных ЭВМ для аппаратной реализации последовательного интерфейса применялась БИС KP580BB51 (аналог i8251), представляющая собой УСАПП (USART) – универсальный синхронно-асинхронный приемопередатчик. Эта микросхема не совместима на уровне регистров с БИС **UART INS8250**, и в современных ПК она практически не применяется.

Дальнейшим развитием **UART INS8250** стали БИС 8250A(B)/16450/16450A/16550/16550A, совместимые на уровне регистров, но обладающие повышенным быстродействием, наличием буфера данных магазинного типа, а также поддержкой режимов работы с прямым доступом к памяти. Каждая из перечисленных микросхем совместима с предыдущей, но не наоборот. Это следует учитывать при составлении программы управления – если разработанная программа ориентирована на UART 8250, то она будет работать со всеми последующими модификациями УАПП.

Особенностью БИС 16550 и 16550A является то, что они имеют дополнительные, по сравнению с UART 8250, регистры. Многие биты, считающиеся в БИС 8250 резервными, в микросхеме 16550 задействованы для управления её новыми функциями. Однако все регистры UART 8250 совпадают с соответствующими регистрами БИС 16550, что обеспечивает совместимость.

Современные IBM–совместимые персональные ЭВМ поддерживают до четырех COM–портов. Во время процедуры POST BIOS тестирует и инициализирует COM1 и COM2, при этом адаптер COM1 обычно декодирует порты 3F8H ... 3FFH, а COM2 — 2F8H ... 2FFH. Адаптер может вызвать аппаратное прерывание по ряду условий, в зависимости от значений в регистре разрешения прерываний **IER** (3F9H или 2F9H), причем COM1 вы-

зывает прерывание IRQ 4 (обрабатывается вектором INT 0CH), а COM2 — IRQ 3 (обрабатывается вектором INT 0BH). Базовые адреса последовательных портов занимают 8 первых байт таблице COMМ, начинающейся с адреса 0:0400H.

В моделях IBM PC/AT порты последовательного интерфейса были интегрированы в отдельную БИС, в функции которой также входило обслуживание каналов параллельного ввода/вывода, дисковод, жесткого диска и GAME-порта. На современных системных платах эти функции берет на себя интегральный чипсет, сохраняя при этом совместимость регистров портов по адресам ввода/вывода.

Практические задания, предполагающие работу с УАПП, легко реализуются в среде программного эмулятора при отображении портов адаптера последовательного ввода/вывода виртуальной машины K580BM80 на регистры БИС последовательного интерфейса материнской платы. Под управлением программы-эмулятора при работе с портами осуществляется простой последовательный ввод/вывод через опрос, режим работы по прерываниям не поддерживается.

Программа — эмулятор, представляющая собой кросс-средство для выполнения команд процессора K580BM80 (i8080) на платформе x86, позволяет обратиться к регистрам последовательного порта COM2 как к ячейкам памяти по адресам 0FFF4H ÷ 0FFF7H, 0FFF1H ÷ 0FFF3H по следующему соглашению:

- 0FFF4H — (запись) регистр **TSR**; 0FFF4H — (чтение) регистр **RBR**;
- 0FFF5H — (запись) старший байт делителя;
- 0FFF5H — (запись) регистр разрешения прерываний **IER**;
- 0FFF6H — (чтение/запись) регистр управления линией **LCR**;
- 0FFF7H — (запись) регистр управления модемом **MCR**;
- 0FFF1H — (чтение) регистр идентификации прерываний **IIR**;
- 0FFF2H — (чтение) регистр статуса линии **LSR**;
- 0FFF3H — (чтение) регистр статуса модема **MSR**.

Последовательные порты компьютера также доступны из программы-эмулятора как устройства ввода/вывода по их реальным адресам посредством команд IN и OUT. Для достижения общности с платформой IBM PC/AT при программировании внешних устройств, команды IN N и OUT N интерпретируются программой эмулятора расширенным образом. Команды IN 0BCh (IN 0BDh) и OUT 0BCh (OUT 0BDh) выполняют обращение к устройству ввода/вывода, адрес которого указан в регистровой паре соответственно BC и DE.



## 2.2. ПРАКТИЧЕСКИЕ ЗАДАНИЯ

### Задание 3.1

Осуществить самотестирование адаптера последовательного ввода/вывода микропроцессорного контроллера или инструментальной ЭВМ (УСАПП или УАПП) посредством приема информации, переданной самим же адаптером. С этой целью использовать разъём–заглушку, осуществляющий следующие соединения линий адаптера: (TxD → RxD), RTS → CTS; DTR → (DCD + DSR). Для тестирования передать набор байт с кодами 20H÷7FH, сравнивая принятую информацию с переданной. Идентичность переданных и принятых байт будет свидетельствовать об исправности информационных каналов последовательного интерфейса.

По успешному завершению операций с приёмом/передачей набора байт, осуществить тестирование каналов данных адаптера в режиме эхопечати, когда байт, введенный с клавиатуры как код клавиши, передаётся через последовательный интерфейс, а принятое значение отображается на дисплее. Соответствие выведенного на дисплей символа символу нажатой клавиши также подтвердит исправность информационных каналов последовательного интерфейса.

Тестирование провести для ряда стандартных скоростей обмена: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 бит/сек, осуществляя их выбор по нажатию клавиш [1] ÷ [9], которым соответствуют коды 31H ÷ 39H.

Доступное пользователю ОЗУ расположено по адресам 0000H÷7FFFH. Частота тактирования адаптера последовательного ввода/вывода инструментальной ЭВМ – 115200 Гц. Регистры адаптера (если это не оговорено дополнительно) адресуются по соглашению, приведенному в разделе 2.1 "Программно–аппаратные средства".

При выполнении задания реализовать следующую настройку адаптера последовательного интерфейса:

**вид обмена - асинхронный,**  
**протокол обмена - аппаратный,**  
**длина слова - 8 бит,**  
**контроль четности - четность,**  
**количество стоп-бит - 1 стоп-бит.**

Для организации диалогового режима применить стандартные подпрограммы опроса клавиатуры и вывода символов и сообщений на дисплей. Использовать код 0DH ("Enter") для управления тестированием и код 1BH ("Esc") — для выхода в системный монитор по адресу 0F800H. Для начальной очистки видео–ОЗУ использовать стандартную системную подпрограмму 0C80FH, вызвав её дважды при наличии управляющего кода 1FH в регистре А микропроцессора.

### Задание 3.2

Осуществить приём блока кодов от учебной микро-ЭВМ комплекса УМПК-51 по интерфейсу RS-232C, используя адаптер последовательного ввода/вывода микропроцессорного контроллера или инструментальной ЭВМ (УСАПП или УАПП). Регистры адаптера (если это не оговорено дополнительно) адресуются по соглашению, приведенному в разделе 2.1 "Программно-аппаратные средства".

Доступное пользователю ОЗУ инструментальной ЭВМ расположено по адресам 0000H÷7FFFH. Доступное пользователю ОЗУ микро-ЭВМ комплекса УМПК-51 занимает адреса: 1000H÷2FFFH, в адресном пространстве 0000H÷0FFFH расположены подпрограммы тест-мониторной системы микро-ЭВМ. Частота тактирования адаптера последовательного ввода/вывода инструментальной ЭВМ – 115200 Гц. При выполнении задания реализовать следующую настройку адаптера последовательного интерфейса:

вид обмена - асинхронный,  
 протокол обмена - аппаратный,  
 скорость обмена - 2400 бод,  
 длина слова - 8 бит,  
 контроль четности - без контроля,  
 количество стоп-бит - 2 стоп-бита.

После запуска программы обмена нажатием клавишей [NL] и вывода сообщения (**rs-232**), микро-ЭВМ комплекса УМПК-51 ожидает управляющую посылку от инструментальной ЭВМ, которая инициализирует передачу блока данных. Формат **управляющей посылки** следующий:

первый байт: 52H ("R") - передача из ОЗУ УМПК-51 в ЭВМ;  
 второй байт: младший байт начального адреса ОЗУ УМПК-51;  
 третий байт: старший байт начального адреса ОЗУ УМПК-51;  
 четвертый байт: младший байт конечного адреса ОЗУ УМПК-51;  
 пятый байт: старший байт конечного адреса ОЗУ УМПК-51;  
 шестой байт: (CRC) контрольная сумма управляющей посылки.

В случае безошибочного приема управляющей посылки микро-ЭВМ комплекса УМПК-51 выдает байт ответа (**06H**) и начинает передачу блока данных из ячеек адресного пространства, начиная с указанных в посылке. Данные оканчиваются второй контрольной суммой (**CRC<sub>2</sub>**), которая подсчитывается как продолжение контрольной суммы управляющей посылки (**CRC**) без учета самого шестого байта CRC путем циклического сложения всех байт без учета переносов и переполнений.

Во время передачи информации на дисплей микро-ЭВМ комплекса УМПК-51 выводится адрес текущей ячейки адресного пространства в формате:

**XXXX**

По завершению передачи блока данных микро-ЭВМ комплекса УМПК-51 выходит из подпрограммы обмена, что индицируется следующим сообщением:

### **rEADY**

В ходе приема байтов **управляющей посылки** микро-ЭВМ комплекса УМПК-51 осуществляет занесение их в ячейки внутренней памяти данных с подсчетом **контрольной суммы (КС)** путем циклического сложения всех байтов без учета переносов и переполнений. По окончании приема **управляющей посылки** микро-ЭВМ принимает байт контрольной суммы (**CRC**), вычисленный уже инструментальной ЭВМ, который сравнивается с **КС**. Если значения **CRC** и **КС** идентичны, микро-ЭВМ делает вывод о безошибочном приёме данных посылки и начинает передачу блока данных, подсчитывая его **контрольную сумму**. В случае несовпадения значения **CRC** и **КС** процедура обмена прекращается с индикацией сообщения об ошибке.

Для управления обменом использовать опрос состояния разряда 1 канала В параллельного периферийного адаптера, обслуживающего матрицу клавиатуры, установка которого в нулевое состояние через подключенную к нему замыкающую кнопку (нажатие клавиши [**Shift**]) прерывает процедуру обмена и инициализирует выход в системный монитор по адресу 0F800H. Адрес канала В ППА в совмещенной карте памяти – 0FFE1H.

Для создания файла использовать стандартные подпрограммы файловых операций инструментальной ЭВМ. Для организации диалогового режима использовать стандартные подпрограммы опроса клавиатуры и вывода символов и сообщений на дисплей.

### **Задание 3.3**

Осуществить передачу блока кодов учебной микро-ЭВМ комплекса УМПК-51 по интерфейсу RS-232C, используя адаптер последовательного ввода/вывода микропроцессорного контроллера или инструментальной ЭВМ (УСАПП или УАПП). Регистры адаптера (если это не оговорено дополнительно) адресуются по соглашению, приведенному в разделе 2.1 "Программно-аппаратные средства".

Доступное пользователю ОЗУ инструментальной ЭВМ расположено по адресам 0000H÷7FFFH. Доступное пользователю ОЗУ микро-ЭВМ комплекса УМПК-51 занимает адреса: 1000H÷2FFFH, в связи с чем размер передаваемого блока кодов не должен превышать 4096 байт. Частота тактирования адаптера последовательного ввода/вывода инструментальной ЭВМ – 115200 Гц.

При выполнении задания реализовать следующую настройку адаптера последовательного интерфейса:

вид обмена - асинхронный,  
 протокол обмена - аппаратный,  
 скорость обмена - 2400 бод,  
 длина слова - 8 бит,  
 контроль четности - без контроля,  
 количество стоп-бит - 2 стоп-бита.

После запуска программы обмена нажатием клавишей [NL] и вывода сообщения (rs-232), микро-ЭВМ комплекса УМПК-51 ожидает управляющую посылку от инструментальной ЭВМ. Формат управляющей посылки следующий:

первый байт: 57H ("W") - передача из ЭВМ в ОЗУ УМПК-51;  
 второй байт: младший байт начального адреса ОЗУ УМПК-51;  
 третий байт: старший байт начального адреса ОЗУ УМПК-51;  
 четвертый байт: младший байт конечного адреса ОЗУ УМПК-51;  
 пятый байт: старший байт конечного адреса ОЗУ УМПК-51;  
 шестой байт: (CRC) контрольная сумма управляющей посылки.

В случае безошибочного приема управляющей посылки микро-ЭВМ комплекса УМПК-51 выдает байт ответа (06H) и начинает прием блока данных по указанным адресам. Данные оканчиваются второй контрольной суммой (CRC<sub>2</sub>), которая подсчитывается как продолжение контрольной суммы управляющей посылки (CRC) без учета самого шестого байта CRC путем циклического сложения всех байт без учета переносов и переполнений.

Во время приема информации на дисплей микро-ЭВМ комплекса УМПК-51 выводится адрес текущей ячейки ОЗУ в формате:

**XXXX**

В случае безошибочного приёма данных микро-ЭВМ комплекса УМПК-51 выдает байт ответа (86H), подтверждающий приём, и выходит из подпрограммы обмена, что индицируется сообщением

**rEADY**

В ходе приема байтов управляющей посылки микро-ЭВМ комплекса УМПК-51 осуществляет занесение их в ячейки внутренней памяти данных с подсчетом контрольной суммы (КС) путем циклического сложения всех байтов без учета переносов и переполнений. По окончании приема управляющей посылки микро-ЭВМ принимает байт контрольной суммы (CRC), вычисленный уже инструментальной ЭВМ, который сравнивается с КС. Если значения CRC и КС идентичны, микро-ЭВМ делает вывод о безошибочном приёме данных посылки и начинает прием блока данных. В противном случае процедура обмена прекращается с индикацией сообщения об ошибке.

Для управления обменом использовать опрос состояния разряда 1 канала В параллельного периферийного адаптера, обслуживающего матрицу клавиатуры, установка которого в нулевое состояние через подключенную к нему замыкающую кнопку (нажатие клавиши [**Shift**]) прерывает процедуру обмена и инициализирует выход в системный монитор по адресу 0F800H. Адрес канала В ППА в совмещенной карте памяти – 0FFE1H.

Для запроса файла использовать стандартные подпрограммы файловых операций инструментальной ЭВМ. Для организации диалогового режима использовать стандартные подпрограммы опроса клавиатуры и вывода символов и сообщений на дисплей.

**Примечание.** При выполнении заданий с использованием подпрограмм-имитаторов внешних устройств, а также в отдельных случаях специальной настройки конфигурации программы эмулятора адреса регистров УАПД могут отличаться от рекомендуемых в тексте задания.

### 2.3. ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЙ

1. Подготовить компьютер для выполнения соответствующего задания.
2. Ввести в редакторе подготовленные тексты программ, скомпилировать объектный код, отладить и выполнить программу. При статической отладке, следует в пошаговом режиме убедиться в правильности загрузки управляющих слов и рабочих значений в соответствующие регистры интерфейсной БИС, проконтролировать вывод в порты многоканальным логическим пробником. В режиме динамической отладки форму сигналов и правильность выполнения алгоритма следует проверить при помощи осциллографа или многоканального логического анализатора.
3. Оформить индивидуальный отчет, включающий необходимые иллюстративные материалы (функциональные схемы исследуемых подсистем, алгоритмы разработанных программ, временные диаграммы) и листинги отлаженных программ.

## СПИСОК ЛИТЕРАТУРЫ

1. Гук М. Аппаратные средства IBM-PC. Энциклопедия, 2-е изд. – СПб.: Питер 2003. – 928 с.
2. Микропроцессоры и микропроцессорные комплекты интегральных микросхем. Под ред. В.А. Шахнова. – М.: "Радио и связь", 1988. Т.1, – 368 с.
3. Микропроцессоры. В 3-х кн. / П.В. Нестеров, В.Ф. Шаньгин, В.Л. Горбунов и др.; Под ред. Л.Н. Преснухина. – М.: "Высшая школа", 1986.
4. Гутников В.С. Интегральная электроника в измерительных устройствах. – Ленинград.: "Энергоатомиздат", 1988. – 304 с.
5. Васильев Б.И., Гусев Ю.М., Миронов В.Н. Электронные промышленные устройства. – М.: "Высшая школа", 1988. – 304 с.
6. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC: Пер. с англ. / Под ред. У. Томпкинса и Дж. Уэбстера. – М.: Мир, 1992. – 592 с.
7. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT – М.: "Финансы и статистика", 1992. – 544 с.
8. МикроЭВМ: В 8 кн. Практик. Пособие / Под ред. Л.Н. Преснухина. Кн.7. Учебные стенды / Ю.И. Волков, В.Л. Горбунов, Д.И. Панфилов, С.Г. Шаронин. – М.: "Высшая школа", 1988. – 224 с.
9. Нортон П., Соухэ Д. Язык ассемблера для IBM PC. – М.: "Компьютер", 1993. – 352 с.
10. Рудаков П.И., Финогенов К.Г. Программируем на языке ассемблера IBM PC: В 4-х частях. Ч.1. (Ч.2., Ч.3–4.) – М.: "Энтроп", 1995. – 164 с. (164 с., 320 с.).

## ПРИЛОЖЕНИЕ

### СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА K580BM80

Флаги МП образуют регистр флагов (признаков) F: S.Z.0.AC.0.P.1.C

Условные обозначения: Z (ZERO) - флаг нулевого результата

S (SIGN) - флаг знака результата,

C (CARRY) - флаг переноса,

AC (AXIAL CARRY) - флаг осевого переноса,

P (PARITY) - флаг четности результата,

+ - команда изменяет флаг,

- - команда не изменяет флаг,

0 - команда устанавливает флаг в 0,

1 - команда устанавливает флаг в 1,

R - регистр общего назначения,

RP - регистровая пара,

# - однобайтовый операнд,

## - двухбайтовый операнд,

XX - адрес.

КОМАНДА	ДЛИНА (БАЙТ)	ФУНКЦИЯ	ПРИЗНАКИ				
			Z	S	C	AC	P
1	2	3					
MOV R,R	1	передача содержимого одного регистра в другой	-	-	-	-	-
MVI R,#	2	загрузка регистра вторым байтом команды	-	-	-	-	-
INR R	1	увеличить содержимое регистра на 1	+	+	-	+	+
DCR R	1	уменьшить содержимое регистра на 1	+	+	-	+	+
ADD R	1	к содержимому аккумулятора прибавить содержимое регистра	+	+	+	+	+
ADC R	1	к содержимому аккумулятора прибавить содержимое регистра с учетом флага переноса	+	+	+	+	+
SUB R	1	вычесть из аккумулятора содержимое регистра	+	+	+	+	+
SBB R	1	вычесть из аккумулятора содержимое регистра и флаг переноса	+	+	+	+	+
ANA R	1	логическое умножение аккумулятора и регистра	+	+	0	+	+
XRA R	1	выполнить операцию неравнозначности аккумулятора и регистра	+	+	0	0	+
ORA R	1	выполнить операцию логического сложения аккумулятора и регистра	+	+	0	0	+
CMP R	1	сравнить содержимое аккумулятора и регистра	+	+	+	+	+
ADI #	2	к содержимому аккумулятора прибавить второй байт команды	+	+	+	+	+
ACI #	2	к содержимому аккумулятора прибавить второй байт команды с учетом флага переноса	+	+	+	+	+

1	2	3	Z	S	C	AC	P
SUI #	2	из содержимого аккумулятора вычесть второй байт команды	+	+	+	+	+
SBI #	2	из содержимого аккумулятора вычесть второй байт команды	+	+	+	+	+
ANI #	2	с учетом флага переноса выполнить операцию логического умножения аккумулятора и второго байта команды	+	+	0	0	+
XRI #	2	выполнить операцию неравнозначности аккумулятора и второго байта команды	+	+	0	0	+
ORI #	2	выполнить логическое сложение аккумулятора и второго байта команды	+	+	0	0	+
CPI #	2	сравнить содержимое аккумулятора и второго байта команды	+	+	+	+	+
RLC	1	сдвинуть циклически влево содержимое аккумулятора	-	-	+	-	-
RRC	1	сдвинуть циклически вправо содержимое аккумулятора	-	-	+	-	-
RAL	1	сдвинуть циклически влево содержимое аккумулятора через флаг переноса	-	-	+	-	-
RAR	1	сдвинуть циклически вправо содержимое аккумулятора через флаг переноса	-	-	+	-	-
JMP XX	3	безусловный переход по указанному адресу	-	-	-	-	-
JC XX	3	условный переход при единичном состоянии флага переноса	-	-	-	-	-
JNC XX	3	условный переход при нулевом состоянии флага переноса	-	-	-	-	-
JZ XX	3	условный переход по нулевому значению результата	-	-	-	-	-
JNZ XX	3	условный переход по ненулевому значению результата	-	-	-	-	-
JP XX	3	условный переход по положительному значению результата	-	-	-	-	-
JM XX	3	условный переход по отрицательному значению результата	-	-	-	-	-
JPE XX	3	условный переход по четности кода результата	-	-	-	-	-
JPO XX	3	условный переход по нечетности кода результата	-	-	-	-	-
CALL XX	3	безусловный вызов подпрограммы по адресу XX	-	-	-	-	-
CC XX	3	вызов подпрограммы при единичном состоянии флага переноса	-	-	-	-	-
CP XX	3	вызов подпрограммы при положительном значении результата	-	-	-	-	-
CZ XX	3	вызов подпрограммы при нулевом значении результата	-	-	-	-	-



1	2	3	Z	S	C	AC	P
CNZ XX	3	вызов подпрограммы при нулевом значении результата	-	-	-	-	-
CM XX	3	вызов подпрограммы при отрицательном значении результата	-	-	-	-	-
CPE XX	3	вызов подпрограммы по четности кода результата	-	-	-	-	-
CPO XX	3	вызов подпрограммы по нечетности кода результата	-	-	-	-	-
RET	1	безусловный возврат из подпрограммы	-	-	-	-	-
RC	1	возврат из подпрограммы по единичному состоянию флага переноса	-	-	-	-	-
RP	1	возврат из подпрограммы по положительному результату	-	-	-	-	-
RM	1	возврат из подпрограммы по отрицательному результату	-	-	-	-	-
RPE	1	возврат из подпрограммы по четности кода результата	-	-	-	-	-
RNC	1	возврат из подпрограммы по ненулевому состоянию флага переноса	-	-	-	-	-
RZ	1	возврат из подпрограммы по нулевому результату	-	-	-	-	-
RNZ	1	возврат из подпрограммы по ненулевому результату	-	-	-	-	-
RPO	1	возврат из подпрограммы по нечетности кода результата	-	-	-	-	-
RST N	1	безусловный вызов подпрограммы по фиксированному адресу (N*8)H	-	-	-	-	-
LXI RP,##	3	загрузка регистровой пары вторым и третьим байтами команды	-	-	-	-	-
PUSH RP	1	запись регистровой пары в стек	-	-	-	-	-
POP RP	1	загрузка регистровой пары из стека	-	-	-	-	-
POP PSW	1	загрузка аккумулятора и регистра признаков из стека	+	+	+	+	+
STA XX	3	запись содержимого аккумулятора по указанному адресу	-	-	-	-	-
LDA XX	3	загрузка аккумулятора содержимым ячейки с указанным адресом	-	-	-	-	-
XCHG	1	поменять местами содержимое регистровых пар [HL] и [DE]	-	-	-	-	-
XTHL	1	поменять местами содержимое верхней ячейки стека и [HL]	-	-	-	-	-
SPHL	1	загрузить в указатель стека [SP] содержимым [HL]	-	-	-	-	-
PCHL	1	загрузить программный счетчик [PC] содержимым [HL]	-	-	-	-	-

1	2	3	Z	S	C	AC	P
DAD RP	1	к содержимому [HL] прибавить содержимое другой регистра- вой пары или указателя стека и результат поместить в [HL]	-	-	+	-	-
STAX RP	1	записать содержимое аккумуля- тора по адресу, указанному в регистравой паре	-	-	-	-	-
LDAX RP	1	загрузить аккумулятор содер- жимым ячейки, адрес которой указан в регистравой паре	-	-	-	-	-
INX RP	1	увеличить на 1 содержимое ре- гистравой пары или указателя стека	-	-	-	-	-
DCX RP	1	уменьшить на 1 содержимое ре- гистравой пары или указателя стека	-	-	-	-	-
CMA	1	инвертировать содержимое аккумулятора	-	-	-	-	-
STC	1	установить флаг переноса в 1	-	-	1	-	-
CMC	1	инвертировать флаг переноса	-	-	+	-	-
DAA	1	десятичная коррекция результата	+	+	+	+	+
SHLD XX	3	записать содержимое [HL] в две ячейки последовательно с указанного адреса: в первую - L, во вторую - H	-	-	-	-	-
LHLD XX	3	загрузить пару [HL] содержи- мым двух последовательных ячеек с указанного адреса, из младшей - L, из старшей - H	-	-	-	-	-
NOP	1	пустая операция	-	-	-	-	-
HLT	1	останов процессора	-	-	-	-	-
IN #	2	загрузка аккумулятора содер- жимым порта с номером #	-	-	-	-	-
OUT #	2	запись содержимого аккумуля- тора в порт с номером #	-	-	-	-	-
DI	1	запретить прерывания	-	-	-	-	-
EI	1	разрешить прерывания	-	-	-	-	-

Учебное издание

**Семёнов Андрей Андреевич**

ОРГАНИЗАЦИЯ  
ПОСЛЕДОВАТЕЛЬНОГО  
ИНТЕРФЕЙСА

Учебное пособие  
для студентов факультета компьютерных наук  
и информационных технологий  
и факультета nano- и биомедицинских технологий

Редактор В.А. Т р у ш и н а  
Технический редактор Л.В. А г а л ь ц о в а  
Корректор Ю.И. А с т а х о в а

---

Подписано в печать \_\_.\_\_.2007.

Формат 60×84 1/16. Бумага офсетная. Гарнитура Times. Печать офсетная.  
Усл.печ.л.1,16(1,25). Уч.-изд.л.0,9. Тираж 150 экз. Заказ

---

Издательство Саратовского университета.  
410012, Саратов, Астраханская, 83.  
Типография Издательства Саратовского университета.  
410012, Саратов, Астраханская, 83.