

МИНИСТЕРСТВО ОБРАЗОВАНИЯ УКРАИНЫ
ДОНЕЦКИЙ
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

и задания к лабораторным работам
по дисциплине «Сети ЭВМ»

Часть I

МИНИСТЕРСТВО ОБРАЗОВАНИЯ УКРАИНЫ

ДОНЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

У т в е р ж д е н о

на заседании кафедры

электронных вычислительных

машин .

Протокол № 1 от 31.08.96

М Е Т О Д И Ч Е С К И Е У К А З А Н И Я

и задания к лабораторным работам по дисциплине

“ Сети ЭВМ “

Часть I

(для студентов специальности 7.09.15.01

дневной и вечерней форм обучения)

Рекомендовано к изданию

методической комиссией

специальности 7.09.15.01

в количестве 250 экземпляров

Протокол № 7 от 24.06.96.

УДК 681 . 142

Методические указания и задания к лабораторным работам по дисциплине "Сети ЭВМ", часть I (для студентов специальности 7.09.15.01 дневной и вечерней форм обучения) / Сост. Ю. В. Губарь - Донецк : ДГТУ, 1996 - 68 с.

Приведены методические указания и задания к работам, входящим в цикл лабораторных работ по дисциплине "Сети ЭВМ". В первой части цикла рассмотрены вопросы программирования адаптера последовательной связи и таймера, передачи и приема кадров в локальных вычислительных сетях с различным методом доступа, механизмами опроса состояния и прерываний, различными способами адресации сетевых станций, а также многоуровневый сетевой интерфейс.

Составитель

доц. Губарь Ю. В.

Ответственный

за выпуск

проф. Святный В. А.

Рецензент

доц. Ладыженский Ю. В.

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Цикл лабораторных работ по курсу "Сети ЭВМ" предназначен для выработки у студентов практических навыков разработки и исследования алгоритмов и программ управления станцией локальной вычислительной сети (ЛВС).

Лабораторные работы проводятся на компьютерах типа IBM PC, соединенные между собой в локальную сеть по стандарту Ethernet, а также согласно интерфейса RS-232 C.

Последовательность выполнения лабораторных работ построена по принципу нарастания сложности и постепенного использования в программах средств и методов, рассматриваемых в лекционном курсе. Конкретные рекомендации по выполнению лабораторных работ приведены в методических указаниях по каждой работе.

В результате проведения лабораторных работ студент должен:

- изучить алгоритмы взаимодействия сетевых станций при различных сетевых топологиях и режимах работы;
- разработать на языке ассемблера или C++ программу, реализующую заданный протокол связи;
- выполнить экспериментальные исследования в лаборатории с использованием сетевого компьютера и составить протокол испытаний.

При подготовке к лабораторным работам кроме настоящих методических указаний рекомендуется пользоваться следующими литературными источниками:

1. Касаткин А. В. Профессиональное программирование на языке Си. Системное программирование. - Минск: Выш. шк., 1993 г.

2. Райс Л. Эксперименты с локальными сетями микроЭВМ. - М.: Мир, 1990 г.

3. Фролов А. В., Фролов А. Г. Программирование модемов. - М.: "Диалог - МИФИ", 1993 г.

4. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT. - М.: Радио и связь, 1992 г.

5. Самофалов К. Г., Виктор О. В. Микропроцессоры. - Библиотека инженера. - К.: Техника, 1989 г.

6. Personal Computer Hardware Reference Library Serial / Parallel Adapter. - IBM Corp., USA, 1986.

7. Кардышев С. В., Капкин А. М. Интерфейс программиста Турбо - Си. - М.: Радио и связь, 1992 г.

ЛАБОРАТОРНАЯ РАБОТА № 1

АНАЛИЗ , ПРОГРАММИРОВАНИЕ И ИССЛЕДОВАНИЕ АДАПТЕРА ПОСЛЕДОВАТЕЛЬНОЙ СВЯЗИ

Цель работы : изучение функциональной схемы , принципов работы и программирования адаптера последовательной связи на основе микросхемы i8250 [6 , 1 , 2].

1.1. Индивидуальное задание

1. Изучите по литературным источникам и материалам приложения I состав и функционирование БИС i8250 .

2 . Рассчитайте систему синхронизации БИС i8250 при частоте задающего генератора $F = 1,8432 \text{ МГц}$. 16 - битное значение делителя N соответствует номеру студента в журнале группы .

3 . Составьте диаграмму сигналов последовательных данных в коде ASCII при передаче собственной фамилии (использовать контроль по четности и два стоповых бита) .

4 . Разработайте на языке TC или TASM и выполните отладку на ПЭВМ программы , которая выполняет следующие действия :

- определяет число COM - портов в компьютере ;
- определяет базовый адрес заданного COM - порта (если этот порт отсутствует в компьютере , то выбирается ближайший к нему меньший по номеру порт) ;
- производит инициализацию регистров заданного COM - порта ;
- считывает состояние регистров RBR , IIR , LSR , MSR , BAUD0 , BAUD1 порта и отображает состояния отдельных разрядов на экране .

Исходные данные.

Номер COM - порта :	0	1	2	3						
	COM1	COM2	COM3	COM4	(N) mod 4					
Число стоповых бит :	0	1								
	1 бит	2 бита			(N) mod 2					
Длина слова :	0	1	2	3						
	5 бит	6 бит	7 бит	8 бит	(N) mod 4					
Контроль :	0	1	2							
	нет	по чет .	по нечет .		(N) mod 3					
Частота передачи (в бодах) :										
0	1	2	3	4	5	6	7	8	9	10
50	75	110	134,5	150	300	600	1200	1800	2000	2400

11	12	13	14	15	16	17	
3800	4800	7200	7600	19200	38400	56000	(N) mod 18

Разрешены прерывания : по установленному разряду регистра LSR .

0	1	2	3	4	5	6	
D6	D5	D4	D3	D2	D1	D0	(N) mod 7

1.2. Методические указания

Последовательно - параллельный адаптер IBM PC имеет параллельный и последовательные порты [6]. Он устанавливается в разьеме функционального расширения системной панели .

Последовательная часть адаптера на основе БИС i8250 является полностью программируемой , поддерживает асинхронную связь . Она добавляет (или удаляет) к последовательному потоку данных и передает (принимает) стартовый бит , стоповые биты и бит четности , обеспечивает возможность программирования генератора частоты передачи по последовательному каналу связи в бодах . Кроме того , она формирует сигналы управления модемом (приложение 1) .

На этапе инициализации компьютера BIOS тестирует имеющиеся порты последовательной связи и инициализирует их . В зависимости от версии BIOS инициализируются первые два или четыре порта. В табл. 1.1 приведены основные данные по каждому порту для IBM PC AT .

Таблица 1.1

Адаптер	Область хранения базового адреса	Базовый адрес	Диапазон адресов	Линия прерыв.	Номер прерыв.
COM1	40 : 00 H	3F8H	3F8H - 3FFH	IRQ4	0CH
COM2	40 : 02 H	2F8H	2F8H - 2FFH	IRQ3	0BH
COM3	40 : 04 H	3E8H	3E8H - 3EFH	IRQ4	0CH
COM4	40 : 06 H	2E8H	2E8H - 2EFH	IRQ3	0BH

Каждый адаптер управляет 10 внутренними регистрами i8250 , доступ к которым осуществляется по номерам портов . Номера портов отсчитываются относительно базового адреса (base_addr) . Если в области хранения базового адреса записан 0 , это свидетельствует об отсутствии адаптера [1] .

Регистры буфера передатчика (THR) и приемника (RBR) расположены по базовому адресу порта . Для передачи данных в линию связи в регистр THR необходимо записать передаваемый байт данных . После приема данных в регистр RBR принятый байт можно прочитать в CPU .

В зависимости от состояния старшего бита регистра управления линией (LCR) , расположенного по адресу `base_addr + 3` , назначение регистра данных может меняться . Если `D[7] = 0` (`DLAB = 0`) , регистр по адресу `base_addr` используется для записи передаваемых данных , либо приема поступающих из последовательной линии данных . Если `DLAB = 1` , регистр используется для ввода младшего байта частоты тактового генератора . Старший байт делителя частоты записывается в регистр BAUD1 , расположенный по адресу `base_addr + 1` .

Ниже приводится описание на языке Си отдельных регистров , входящих в состав UART i8250 [3] . Битовые поля описаны как беззнаковые (unsigned) . Каждому полю выделяется ровно столько бит , сколько задается в поле “ ширина ” .

Регистр управления линией (LCR)

```
#define LCR_N 3 /* смещение относительно base_addr */
typedef union _LCR_ {
    struct {
        unsigned char len : 2 ; /* длина передаваемых данных */
        unsigned char stop : 1 ; /* число стоповых бит */
        unsigned char parity : 2 ; /* контроль по паритету */
        unsigned char stuck_parity : 1 ; /* фиксация четности */
        unsigned char en_break_ctl : 1 ; /* установка BREAK */
        unsigned char dlab : 1 ; /* управление доступом к регистрам */
    } bit_reg ;
    unsigned char byte ;
} LCR ;
```

Регистр состояния линии (LSR)

```
#define LSR_N 5 /* смещение относительно base_addr */
typedef union _LSR_ {
    struct {
        unsigned char in_ready : 1 ; /* данные получены */
        unsigned char overflow : 1 ; /* ошибка переполнения */
        unsigned char parity : 1 ; /* ошибка четности */
        unsigned char synxr : 1 ; /* ошибка синхронизации */
    }
}
```



```

unsigned char break_detekt : 1 ; /* обнаружен break */
unsigned char out_ready : 1 ; /* регистр передатчика пуст */
unsigned char shift_ready : 1 ; /* рег. сдвига перед. пуст */
unsigned char taim_aut : 1 ; /* тайм - аут */
} bit_reg ;
unsigned char byte ;
} LSR ;

```

Регистр управления прерываниями (ICR)

```

#define ICR_N 1 /* смещение относительно base_addr */
typedef union _ICR_ {
    struct {
        unsigned char in_ready : 1 ; /* разр. прер. по готов. приема */
        unsigned char out_ready : 1 ; /* прер. после передачи байта */
        unsigned char err_1 : 1 ; /* разр. прерывания по ошибке */
        unsigned char change : 1 ; /* разр. прерыв. по измен. сост. */
        unsigned char reserv : 4 ; /* не используются */
    } bit_reg ;
    unsigned char byte ;
} ICR ;

```

Регистр идентификации прерывания (IIR)

```

#define IIR_N 2 /* смещение относительно base_addr */
typedef union _IIR_ {
    struct {
        unsigned char no_inter : 1 ; /* нет прерыв., ожид. обслуж. */
        unsigned char inter_id : 2 ; /* идентификация причины прер. */
        unsigned char reserv : 5 ; /* должны быть равны нулю */
    } bit_reg ;
    unsigned char byte ;
} IIR ;

```

Инициализация адаптера последовательной связи

Рассмотрим пример инициализации адаптера последовательной связи, имеющего базовый адрес `base_addr [ 1 ]`. При инициализации устанавливается частота передачи `baud_rate` и формат слова `format` (длина слова, способ контроля и т.п.). Для вывода в порт байта используется функция Турбо - Си :

```

outportb ( int port , unsigned char value ) ;

```


Это макрокоманда , которая записывает значение параметра value в выходной порт , определяемый параметром port .

```
#include < dos . h >

void set_baud ( long unsigned baud_rate , char format )
{
    extern unsigned base_addr ;
    union {
        int baud_divisor ;
        char bytes [ 2 ] ;
    } word ;
    word . bytes [ 0 ] ;
    /* Вычисление значения частоты передачи для записи
       его в регистры base_addr + 0 и base_addr + 1 */
    word . baud_divisor = 1843200L / ( long ) baud_rate >> 4 ;
    /* Установка бита DLAB в регистре LCR */
    outportb ( base_addr + 3 , format | 0x80 ) ; /* DLAB = 1 */
    outportb ( base_addr + 0 , word . bytes [ 0 ] ) ;
    outportb ( base_addr + 1 , word . bytes [ 1 ] ) ;
    /* Сброс бита DLAB в регистре формата слова */
    outportb ( base_addr + 3 , format ) ;
}
```

Для определения числа COM - портов , установленных в компьютере , можно использовать следующий фрагмент программы [2] :

```
int com_inst ( void )
{
    register unsigned _es * com = ( unsigned _es * ) 0x10 ;
    _ES = 0x40 ;
    return ( ( * com & 0x0E00 ) >> 9 ) ;
    /* Возвращает биты 11 , 10 и 9 слова по адресу 40 : 10h */
}
```

При доступе к объекту данных по значению указателя (т . е . при выполнении операции *) сегмент адреса помещается в сегментный регистр (как правило , это регистр ES) . Для указателя _es компилятор сгенерирует машинную инструкцию с префиксом переопределения сегмента , и при формировании адреса данных CPU будет использовать текущее значение в ES . Для доступа к сегментному регистру ES использована псевдопеременная _ES .

Та же самая информация о составе подключенного оборудования может быть получена программой и через прерывание 11H BIOS .

Проверка наличия в системе запрошенного адаптера может быть выполнена следующим образом :

```
if ( ( port > 4 ) || ( port < 0 ) ) return ( - 1 ) ;
```

При выполнении этого фрагмента происходит возврат значения (- 1) если задан порт не COM1 - COM4 .

Считывание из области BIOS базового адреса данного COM - порта можно выполнить с использованием макроопределения МК_FP Турбо - Си . Оно создает дальний указатель из адреса сегмента и смещения :

```
base_addr = * ( ( unsigned far * ) МК_FP ( 0x40 , port * 2 ) ) ;
```

Содержимое отдельных регистров UART можно сосчитать с помощью макрокоманды `int inportb ( int port )` , содержащейся в файле `dos . h` . Эта макрокоманда считывает байт из порта , определенного параметром `port` .

В качестве примера рассмотрим фрагмент программы отображения на экране дисплея содержимого регистра идентификации прерывания [3] :

```
IIR_reg_iir ;
```

```
printf ( " \n \n Регистр идентификации прерывания = % Xh \n \n " ,
```

```
data = ( unsigned char ) inportb ( base_addr + IIR_N ) ) ;
```

```
reg_iir . byte = data ;
```

```
printf ( " Нет прерываний , ожидающих обслуживания : % d \n " ,
```

```
    " Идентификатор прерывания : % d \n " ,
```

```
    reg_iir . bit_reg . no_inter ,
```

```
    reg_iir . bit_reg . inter_id ) ;
```

```
printf ( " \n \n Нажмите любую клавишу для продолжения " ) ;
```

```
getch ( ) ;
```

1.3 . Контрольные вопросы

1. Где и как формируются стартовый и стоповый биты ?
2. Для чего служит бит DLAB в регистре управления линией UART ?
3. Каким образом осуществить проверку того , что в компьютере установлен заданный COM - порт ?
4. Что произойдет , если программа начнет работать с COM- портом , инициализация которого не была проведена ?
5. Какие ошибки могут возникнуть при асинхронной передаче ?
6. Объясните , каким образом организована система прерываний UART?

1.4 . Содержание отчета

- 1 . Расчет системы синхронизации UART i8250 .
- 2 . Диаграмма сигналов в линии связи , отображающая свою фамилию .
- 3 . Листинг программы с подробными комментариями .

ЛАБОРАТОРНАЯ РАБОТА № 2

АНАЛИЗ , ПРОГРАММИРОВАНИЕ И ИССЛЕДОВАНИЕ ТАЙМЕРА

Цель работы : изучение функциональной схемы , режимов работы и программирования интервального таймера на основе БИС i8253 .

2.1 . Индивидуальное задание

1 . Изучите по литературным источникам и материалам приложения 2 состав и функционирование блоков таймера i8253 .

2 . Выполните инициализацию таймера в заданном режиме работы . Вариант задания выберите как $(N) \bmod 6$. Приведите временную диаграмму работы таймера и схему его подключения к системной шине процессора при константе пересчета $n = 8$ и тактовой частоте $F_{clk} = 8 \text{ МГц}$.

3 . Разработайте и выполните отладку программы управления светофором . На перекрестке дорог 1 и 2 установлен светофор , который функционирует по жесткой программе (табл . 2 . 1) .

Таблица 2 . 1

Номер состояния	Дорога 1	Дорога 2	Время , сек .
1	Зеленый	Красный	$(N) \bmod 30$
2	Желтый	Красный и желтый	$(N) \bmod 5$
3	Красный	Зеленый	$(N) \bmod 15$
4	Красный и желтый	Желтый	$(N) \bmod 7$

Микропроцессорная система управляет светофором подачей на него управляющего слова : $D7 \leftarrow 0$, $D6 \leftarrow 0$; Дорога 1 : $D5 \leftarrow$ красный , $D4 \leftarrow$ желтый , $D3 \leftarrow$ зеленый ; Дорога 2 : $D2 \leftarrow$ красный , $D1 \leftarrow$ желтый , $D0 \leftarrow$ зеленый .

При единичном значении бита соответствующий сигналу цвет загорается , при нулевом - гаснет . Для отсчета временных интервалов предусмотрен второй канал таймера (порт 42h) .

Для наблюдения за ходом выполнения программы отобразите на экране дисплея два небольших окна , имитирующих работу светофора .

2.2. Методические указания

Программирование таймера можно разделить на два этапа. На первом этапе нужно загрузить управляющие слова по всем используемым каналам таймера. На втором этапе необходимо осуществить загрузку самих счетчиков константами пересчета (младший байт , затем старший байт).

Пусть , например , сигнал тактовой частоты , подаваемый на вход CLK , равен 5 МГц ($T = 200$ нс). Выполним настройку таймера как счетчика событий , работающего в пятом режиме . Таймер выдает на выходах OUT сигналы через 1 , 2 мкс , 1 , 6 мкс и 2 мкс [5]. Следовательно , константы пересчета , загружаемые в 0 , 1 и 2 каналы счетчиков будут соответственно равны 6 , 8 и 10 .

Примем , что таймер имеет следующие адреса портов : 40h , 41h , 42h и 43h . Напомним , что в пятом режиме (аппаратно - запускаемый одновибратор) на выходе OUT устанавливается сигнал низкого уровня на время одного периода CLK после отсчета загруженной в счетчик CE константы . Загрузка константы происходит аппаратно по фронту сигнала GATE .

Фрагмент кода программы на языке ассемблера имеет вид :

; запись управляющего слова в счетчик 0

MVI A , 0011 1010B

OUT 43H

; запись управляющего слова в счетчик 1

MVI A , 0111 1010B

OUT 43H

; запись управляющего слова в счетчик 2

MVI A , 1011 1010B

OUT 43H

; загрузка младшего байта в счетчик 0

MVI A , 0000 0110B

OUT 40H

; загрузка старшего байта в счетчик 0

MVI A , 0000 0000B

OUT 40H

; загрузка младшего байта в счетчик 1

MVI A , 0000 1000B

OUT 41H

; загрузка старшего байта в счетчик 1

MVI A , 0000 0000B

OUT 41H

; загрузка младшего байта в счетчик 2

MVI A , 0000 1010B

OUT 42H

; загрузка старшего байта в счетчик 2

MVI A , 0000 0000B

OUT 42H

Все машины IBM PC имеют в своем составе микросхему таймера [4 , 5]. Каждый из трех каналов таймера i8253 (i8254 для IBM AT) состоит из трех регистров (прил. 2). Доступ к каждой группе регистров осуществляется через один из портов. Порты имеют номера от 40H до 42H и соответствуют каналам 0 , 1 и 2. Восьмибитовое управляющее слово загружается в регистр с адресом порта 43H.

Две входные линии для каждого канала таймера управляют его работой : это линия тактовых сигналов CLK и линия разрешения счета GATE , называемая " воротами " . Для нулевого и первого каналов системного таймера " ворота " всегда открыты. Для второго канала имеется возможность управлять " воротами " . " Ворота " закрываются установкой младшего бита порта с адресом 61H , который является регистром микросхемы i8255 [4]. Сброс этого бита снова открывает " ворота " .

Рассмотрим пример управления вторым каналом таймера .

; инициализация второго канала системного таймера

MOV AL , 1011 1010B ; управляющее слово

OUT 43H , AL

MOV AL , 0CH

OUT 42H , AL

; загрузка младшего байта в счетчик

MOV AL , 0E9H

OUT 42H , AL

; загрузка старшего байта в счетчик

; запуск таймера в работу

IN AL , 61H

OR AL , 01H

OUT 61H , AL

; установка GATE2

AND AL , 0FFH

OUT 61H , AL

; сброс GATE2 : " ворота " открыты

; проверка завершения счета

M1 : MOV AL , 0E8H

; код команды RBC

OUT 43H , AL

IN AL , 42H

; чтение слова состояния канала 2

TEST AL , 80H

; проверка состояния выхода OUT2

JNZ M1

2.3. Контрольные вопросы

1. Сформулируйте отличия в работе таймера в режимах 0 и 4, 1 и 5.
2. Каким образом и в какой последовательности осуществляется загрузка управляющих слов и констант пересчета в счетчики ?
3. Какие три способа чтения состояния счетчиков таймера вам известны ? Перечислите эти способы и определите работу таймера.
4. Нарисуйте схему подключения таймера к системной шине процессора .
5. Разработайте программные модели работы таймера в каждом из шести режимов .

2.4. Содержание отчета

1. Временная диаграмма работы , схема подключения к системной шине и инициализация таймера по второму пункту задания .
2. Листинг программы управления светофором с подробными комментариями .

ЛАБОРАТОРНАЯ РАБОТА № 3 АСИНХРОННАЯ ПЕРЕДАЧА БАЙТОВ С ИСПОЛЬЗОВАНИЕМ МЕХАНИЗМА ОПРОСА СОСТОЯНИЙ

Цель работы : изучение принципов функционирования и программирования передающего и приемного узлов станции сети при реализации симплексной передачи данных с использованием механизма опроса состояний.

3.1. Индивидуальное задание

Разработайте и выполните отладку программ работы передающего и приемного узлов станции , работающих по следующему алгоритму :

- инициализация адаптеров станций ;
- установление связи между станциями путем обмена символами R (прием) и S (передача) ;
- передача и прием символов (фамилия , имя , отчество) и отображение их на левой и правой половинах экрана дисплея ;
- идентификация ошибки в коммуникациях и выдача на экран соответствующего сообщения ;
- завершение работы после нажатия на клавишу Enter .

Исходные данные (номер COM - порта , число стоп - бит , контроль и частота передачи) принимаются равными значениям из лабораторной работы № 1 . Длину слова принять равной 8 бит . Прерывания запрещены .

3.2. Методические указания

В данной лабораторной работе рассматриваются вопросы соединения рабочих станций локальной вычислительной сети (ЛВС) через последовательный интерфейс RS - 232 C . Этот интерфейс подразумевает наличие оборудования двух типов : терминального (DTE) и связного (DCE) . В большинстве случаев компьютер является терминальным устройством , а модем , принтер , графопостроитель - всегда являются устройствами связи . Терминальное оборудование может посылать и (или) принимать данные по последовательному интерфейсу . Связное оборудование позволяет упростить последовательную передачу данных совместно с терминальным оборудованием . Наилучшим примером связного оборудования служит модем .

При выполнении лабораторной работы использовано подключение рабочих станций ЛВС через интерфейс RS - 232 C без использования модема (так называемое немодемное соединение) . Минимальное число линий для организации связи двух станций равно трем : для приема данных RXD , передачи данных TXD и для сигнального заземления GND . Связь между станциями при использовании 9 - контактного разъема организуется по схеме рис . 3 . 1 .

При выполнении трехпроводной схемы соединения перенос информации может быть выполнен только непосредственным доступом к портам , так как в этом случае функции BIOS не работают [1] . Однако адаптер последовательной связи на основе БИС i8250 сохраняет способность генерировать запросы прерывания и при правильной его настройке и наличии собственного обработчика прерывания может быть организована симплексная , полудуплексная или дуплексная связь .

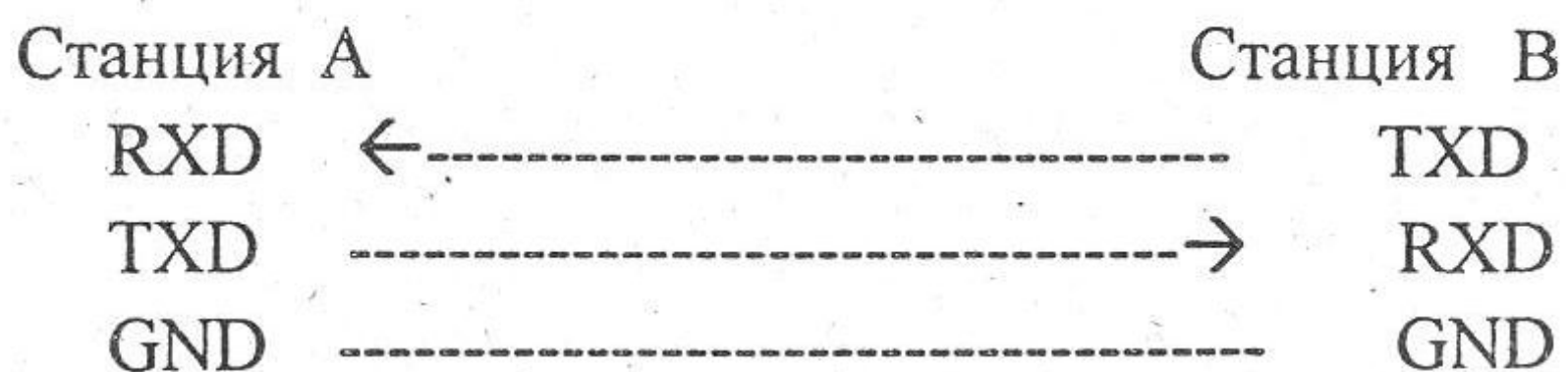


Рис . 3 . 1 . Трехпроводная схема соединения станций

Программирование рабочей станции

В лабораторной работе используются сетевые узлы двух типов : передающий и приемный [1]. Передающий узел должен непосредственно и безусловно передавать по линии TXD все символы , вводимые с его клавиатуры . Приемный узел должен принимать любой символ , появившийся из сетевой шины по линии RXD и отображать его на своем экране .

Таким образом , каждый символ , вводимый с клавиатуры передатчика , с некоторой задержкой передачи должен появиться на экране приемника . Необходимо также организовать эхо - контроль на экран передатчика .

Рассмотрим более подробно алгоритмы работы передающего и приемного узлов станции .

Передающий узел

Ниже приведена программа работы передающего узла станции , оформленная в виде COM - файла на языке ассемблера [4]. Напомним , что COM - файлы хранятся уже в таком виде , в котором загружаемая программа должна быть в памяти машины . Данные и стек в такой программе хранятся в кодовом сегменте , поэтому программный код по длине не может превышать 64 К байт . COM - программы начинаются с адреса 0100H , а завершаются - прерыванием 20 H для возврата управления DOS .

Перед выполнением операций обмена производится инициализация регистров микросхемы UART i8250 . Это регистры делителей скорости обмена BAUD0 и BAUD1 (настраиваются на скорость обмена 2400 бод) , регистр управления линией LCR (настраивается на 8 бит данных , один стоп - бит и контроль по четности) и регистр управления прерываниями ICR (настраивается на запрет всех прерываний) .

Основная часть программы носит циклический характер . В каждом цикле проверяется бит D5 регистра статуса линии LSR. Этот бит показывает , свободен ли регистр хранения передатчика для приема данных от CPU . Если бит D5 = 1 , в регистр хранения передатчика THR посылается очередной байт , введенный с клавиатуры . Этот байт далее пересылается в последовательную линию , а также выводится на экран дисплея . Программа завершает работу после нажатия на клавишу Enter . Если в коммуникациях произошла ошибка , то на экран выводится соответствующее сообщение .

; COM - программа для станции - передатчика

cseg segment

assume cs : cseg , ds : cseg , ss : cseg


```

        org      0100h

start :
        jmp      short begin
        error_message db 'Ошибка в коммуникациях ', oah, odh,
        base_addr    dw ? ; базовый адрес станции

begin :
;
;----- инициализация адаптера последовательного порта
;
;-----получаем базовый адрес COM1
        mov ax , 40h
        mov es , ax ; es указывает на область данных BIOS
        mov dx , es : [ 0 ]
        mov base_addr , dx
;-----устанавливаем регистры BAUD0 и BAUD1
add dx , 3 ; адрес регистра LCR
mov al , 10000000B ; устанавливаем бит DLAB
out dx , al ; посылаем в порт
dec dx
dec dx ; указатель на BAUD1
mov al , 0
out dx , al ; посылаем старший байт делителя
dec dx ; указатель на BAUD0
mov al , 30h
out dx , al ; посылаем младший байт делителя
;----- инициализация регистра LCR
mov al , 0 ; обнуляем al
or al , 11B ; длина данных 8 бит
or al , 000B ; один стоп - бит
or al , 1000B ; контроль по паритету
or al , 10000B ; контроль по четности
add dx , 3 ; адрес регистра LCR
out dx , al ; инициализация LCR
;----- инициализация регистра ICR
dec dx
dec dx ; адрес регистра ICR
mov al , 0 ; запрет прерываний
out dx , al ; посылаем байт в ICR
;

```



```

;
;---основная часть программы по передаче символа
;
;----- ожидание , когда освободится сдвиговый регистр передатчика
;
wait_lab :
mov dx , base_addr      ; базовый адрес адаптера
add dx , 5               ; указатель на LSR
in al , dx               ; получаем байт статуса линии
test al , 00011110B      ; проверка на ошибку
jnz error_out            ; если ошибка , на обработку
test al , 00100000B      ; сдвиговый регистр пустой ?
jz wait_lab              ; если да , возврат на начало
;----- передача символа , принимаемого с клавиатуры
mov ah , 1               ; функция проверки нажатия
int 16h                  ; прерывание клавиатуры BIOS
jz wait_lab              ; возврат , если не было нажат.
mov ah , 0               ; функция получения кода клав.
int 16h                  ; введенный символ в al
cmp al , 13              ; нажата клавиша Enter ?
jne short continue       ; если нет, продолжить работу
int 20h                  ; если да , завершаем программу
continue :
sub dx , 5               ; адрес регистра передатчика
out dx , al              ; передача символа из al в THR
mov dl , al
mov ah , 2
int 21h                  ; вывод символа на экран
jmp short wait_out        ; продолжить цикл опроса
;----- вывод сообщения об ошибке
error_out :
mov ah , 9
lea dx , error_message
int 21h
jmp short wait_lab        ; продолжить цикл опроса
cseg ends
end start

```


Приемный узел

Инициализация приемного узла осуществляется аналогично как и передающего узла, поэтому код этого участка программы не приводится.

Основная часть программы выполнена в виде бесконечного цикла. Программа постоянно проверяет бит D0 регистра статуса линии LSR. Бит D0 этого регистра будет равен нулю, пока не будет получен символ в регистр данных приемника RBR. Когда бит D0 становится равным единице, нужно немедленно считать его из регистра RBR, с тем, чтобы на него не наложился следующий принимаемый символ. После того как символ считан и выведен на экран дисплея, бит D0 опять становится равным нулю и остается таковым пока из линии не поступит очередной символ.

Коммуникационные процедуры обычно создают циклические буферы для сбора поступающих символов [4]. Если принимаемые данные принимать на экран со скоростью 1200 бод, то процедура сдвига экрана BIOS не будет успевать и произойдет переполнение.

Завершение работы приемной станции произойдет после нажатия на клавишу Enter.

```
;  
;  
;  
;----- основная часть программы по приему символа  
;  
;----- ожидание получения символа из линии  
wait_lab :  
mov dx, base_addr  
add dx, 5 ; указываем на регистр LSR  
in al, dx ; получаем байт статуса  
test al, 00011110B ; проверяем на ошибку  
jnz error_out ; на обработку ошибки  
test al, 00000001B ; получены ли данные ?  
je resiv ; если да, на обработку  
mov ah, 1 ; клавиша нажата ?  
int 16h  
jz wait_lab ; перейти, если нет нажатия  
mov ah, 0 ; функция приема кода с  
int 16h ; клавиатуры  
cmp al, 13 ; нажата клавиша Enter ?  
jne short wait_lab ; если нет, возврат на начало  
int 20h ; завершение программы  
;----- обработка полученного символа
```



```

resiv :
mov dx , base _ addr
in  al , dx      ; получает байт из регистра RBR
mov dl , al
mov ah , 2
int  21h        ; вывод символа на экран
jmp short wait _ lab      ; возврат на начало цикла
;----- обработка коммуникационных ошибок
error _ out :
mov ah , 9
lea dx , error _ message
int  21h
jmp short wait _ lab      ; возврат на начало цикла
cseg  ends
      end  start

```

Проверка функционирования последовательной линии связи между двумя компьютерами может быть осуществлена с помощью процедуры Link (связь) , входящей в состав Norton Commander . Для использования этой процедуры выполните следующие действия :

- 1). Нажмите на клавишу F9 для вхождения в меню NC .
- 2). Для каждого компьютера выберите из меню Left или Right элемент Link . При этом на экране появится диалоговое окно Commander Link .
- 3). Далее необходимо выбрать режим работы : Master (ведущий) и Slave (ведомый) , а также последовательный порт , который используется для соединения . Для одного компьютера выбирается режим Master , для другого - Slave : При этом работать можно только на компьютере , находящемся в режиме Master . Ведомый компьютер можно рассматривать как дополнительную внешнюю память , работать на нем нельзя .
- 4). Работая на компьютере Master , можно копировать файлы на или с ведомого компьютера . При этом нет различия , в каком направлении копируются файлы .

3.3 . Контрольные вопросы

- 1 . Предложите способы решения проблемы переполнения регистра RBR при приеме данных из последовательной линии связи с большой скоростью .