2 . Возможно ли организовать работу передающего и приемного узлов с определением состояния линии в дуплексном режиме ?

3 . Адаптеры передающей и приемной станций настроены на разные скорости обмена . Каким образом будет происходить между ними обмен данными ?

4 . Какие дополнительные коммутации в разъеме DB9 необходимо выполнить , чтобы можно было использовать коммуникационные функции BIOS ?

5 . Предложите алгоритм обмена управляющими сигналами между устройствами DTE и DCE при установлении связи .

6 . Каким образом в интерфейсе RS - 232 используются сигналы RI , CTS , RTS , DTR , DSR ?

3. 4 . Содержание отчета

Листинги программ передающего и приемного узлов с подробными комментариями .

ЛАБОРАТОРНАЯ РАБОТА № 4
ПЕРЕДАЧА И ПРИЕМ КАДРОВ С ИСПОЛЬЗОВАНИЕМ МЕХАНИЗМА ПРЕРЫВАНИЙ

Цель работы : изучение принципов функционирования и программирование передающего и приемного узлов станции при обмене кадрами с использованием механизма прерываний .

4. 1 . Индивидуальное задание

Разработайте на языке TC или TASM и выполните отладку программ , описывающих работу передающего и приемного узлов станции с использованием механизма прерываний . Настройку адаптера последовательной связи произвести в соответствии с исходными данными к лабораторной работе № 1 .

Данная лабораторная работа рассчитана на восемь часов аудиторных занятий . Из них четыре часа отводятся на разработку алгоритмов и программ передающего узла станции , а следующие четыре часа - приемного узла станции и проведение экспериментальных исследований . Работа выполняется бригадой из двух студентов .

4. 2 . Методические указания

Рассматривается дуплексная передача кадров в виде нескольких байтов между передающим и приемным узлами станции сети с

использованием механизмов прерывания , буферизации и форматирования кадров [1 , 2] . Сетевые порты обслуживаются исключительно ISR - программами (Interrupt Service Routine), вызываемые прерываниями .

Кадр из нескольких байтов вводится с клавиатуры терминала передатчика во входной буфер buf _ s . Введенный символ возврата каретки CR (код ASCII 13) означает конец ввода кадра . В этот момент передатчик выдает на экран своего монитора кадр и начинает его передачу байт за байтом в сетевую среду . Приемник принимает кадр из среды как некоторую последовательность байтов и размещает их в своем буфере buf _ r . После приема заключительного символа EOT полученный кадр выводится на зкран монитора приемного узла .

Чтобы дать возможность приемнику распознавать начало и конец кадра данных , передатчик строит кадры следующего формата :

PRE , . . . данные . . . , EOT

где PRE - преамбула ;

EOT - признак конца передачи .

Символы PRE и EOT - это передний и задний разделители кадра . В качестве PRE используется байт , состоящий из одних нулей (он никогда не должен включаться в поле данных какого - либо кадра) . Приход байта PRE означает начало поступления нового кадра . В коде ASCII EOT принимают равным коду знака доллара $ (в 16 - й системе код 24) .

Будем полагать , что последовательность символов ограничена по размеру одной строкой , то есть 80 символами . Клавиша Enter будет использоваться для завершения сообщения при его вводе . Чтобы сделать понятными сообщения на экранах передающего и приемного узлов во время ввода и отображения кадра , последние два символа в поле данных всегда будут комбинацией кода для CR и LF (перевод строки) . Тогда формат кадра будет иметь вид :

PRE , . . . данные . . . , CR , LF , EOT

Буфер кадра строится как некоторый массив оперативной памяти , определенный своим начальным адресом buf _ org :

buf _ org → PRE
* * *
CR
LF
EOT

Доступ к буферу кадра осуществляется с помощью двух указателей , один из которых указывает на следующую ячейку буфера buf _ nxt , а

другой - на конец буфера buf_end. Размер кадра данных может изменяться в пределах границ одной строки. Поэтому значение указателя конца буфера меняется от кадра к кадру.

Во время инициализации первый байт кадра всегда содержит код преамбулы. Он записывается в первый байт буфера и далее не меняется. Данные вводятся в буфер с клавиатуры. Комбинация кода ASCII первой нажатой клавиши помещается во вторую ячейку буфера. Поэтому инициализировать указатель buf_nxt нужно так, чтобы он адресовал второй байт.

Алгоритмы работы узлов станции

Алгоритм работы передающего узла показан на рис. 4.1. Он включает основную часть (Send) и обработчик аппаратного прерывания от порта COM1 (Isr_s). В начале основной части происходит инсталляция обработчика прерываний. Затем в буфер передатчика загружается преамбула. Далее происходит загрузка буфера передачи buf_s байтами кадра, вводимыми с клавиатуры терминала, отображение кадра на экране и запуск ISR - программы на передачу кадра в линию связи.

Завершение работы основной части программы передающего узла станции происходит по нажатию клавиши Esc.

В обработчике прерываний Isr_s кадр побайтно выводится в линию до момента, когда поступит байт EOT. При этом запрещаются прерывания передатчика и работа передающего узла завершается.

Буфер кадра в приемном узле не содержит поля PRE и EOT. Эти байты используются для синхронизации работы передающего и приемного узлов сети. Они не являются полезной информацией и поэтому выбрасываются приемником из принятого кадра. Алгоритм работы приемного узла представлен на рис. 4.2.

В основной части программного модуля Receive выполняется инсталляция обработчика прерываний, происходит настройка указателя приема байтов buf_nxt_r на начало буфера приема buf_org_r и дается разрешение на прерывания от приемного узла адаптера.

Завершение основной части происходит после нажатия на терминале приемного узла клавиши Esc. При этом происходит восстановление прежнего состояния аппаратуры, которое имело место перед инсталляцией, и завершение работы программы возвратом в DOS.

В обработчике прерывания Isr_r приемного узла станции с помощью флага flag и кода преамбулы PRE устанавливается момент начала поступления кадра. Если принят байт с ошибкой (Error), происходит ее

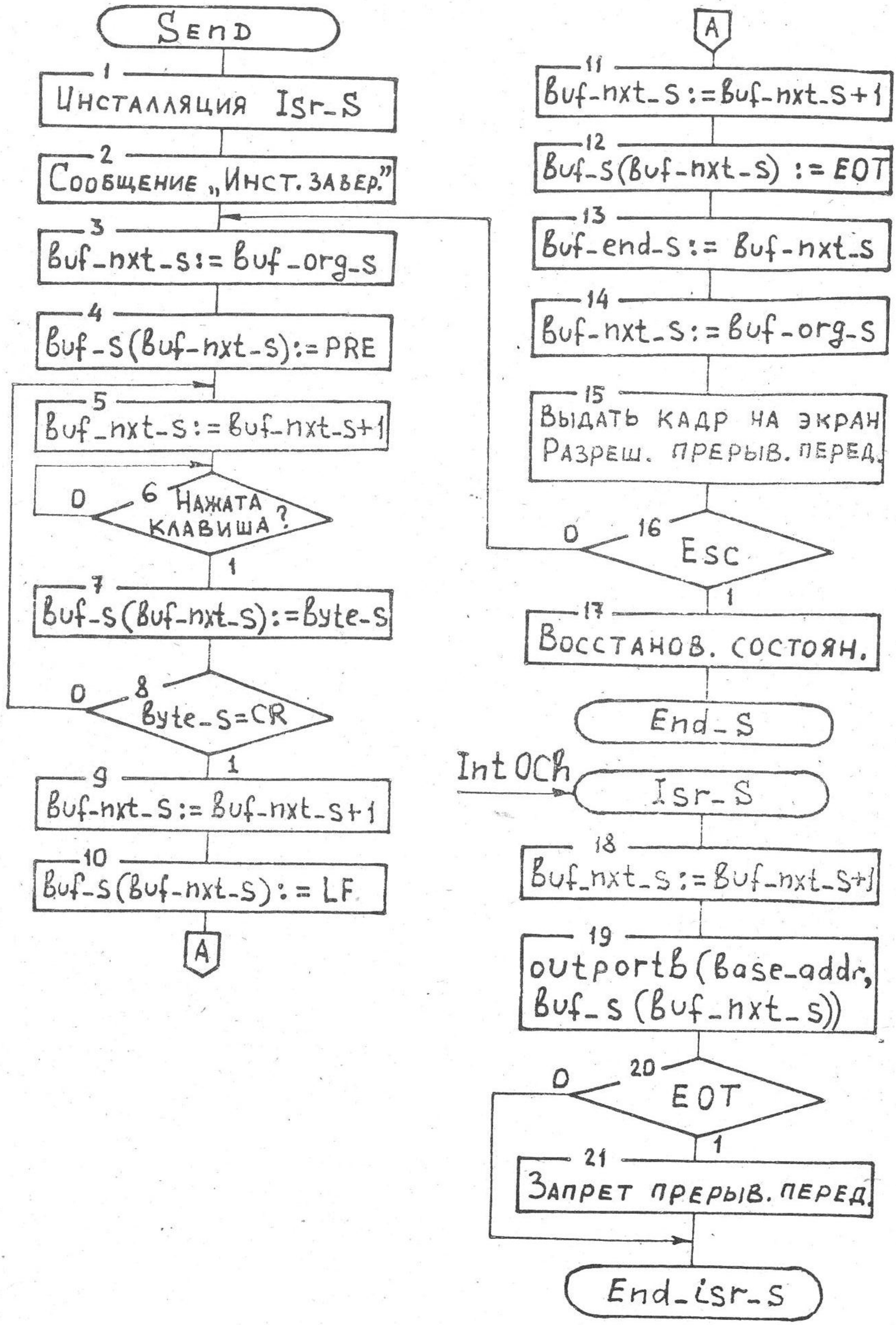

Рис. 4.1. Алгоритм работы передающего узла станции

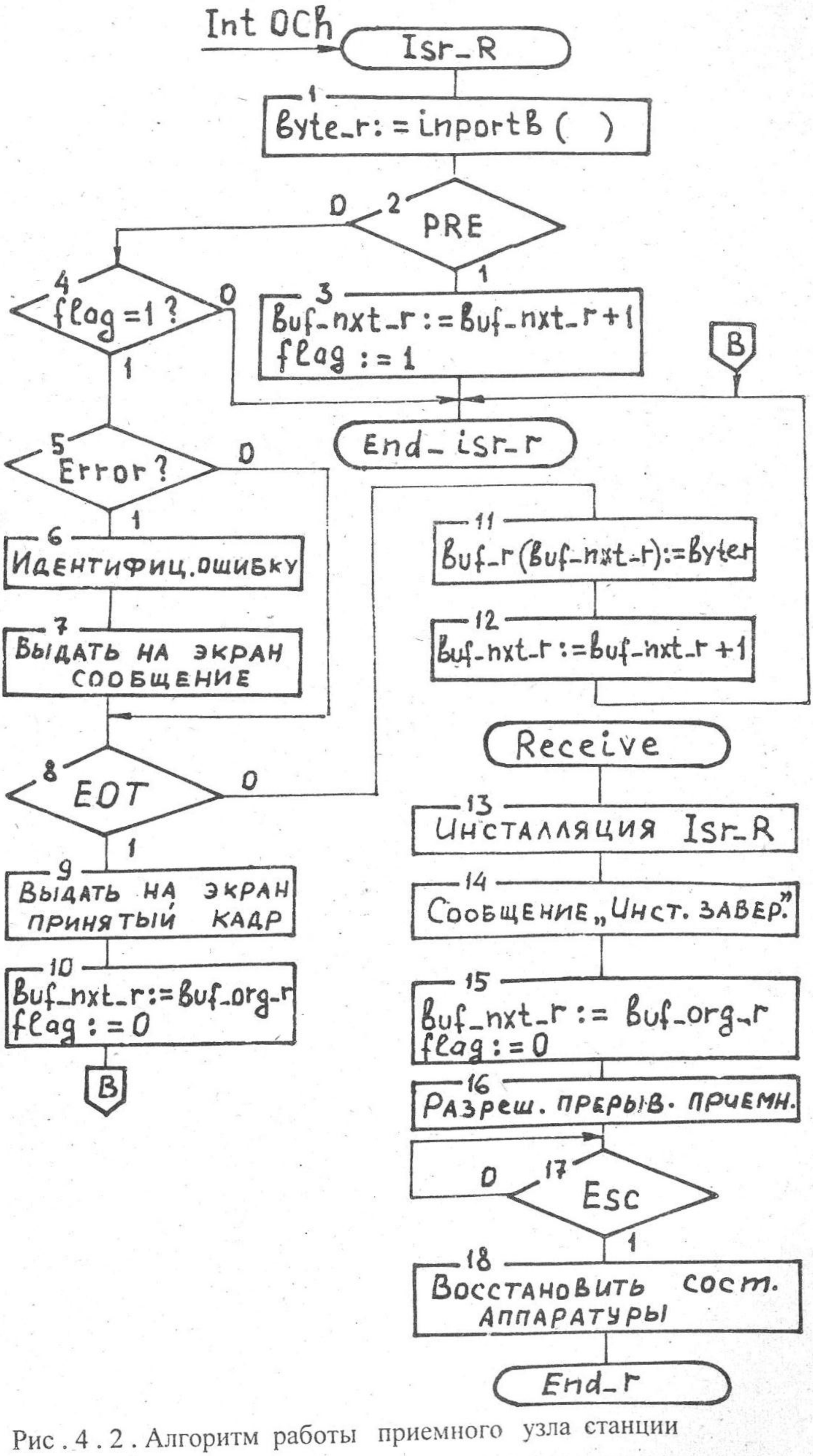

Рис . 4 . 2 . Алгоритм работы приемного узла станции

идентификация и на экран выдается соответствующее сообщение . Принятые байты кадра запоминаются в приемном буфере . По завершению приема , когда поступит байт PRE , на экран терминала приемника выводится принятый кадр . После этого происходит восстановление состояния переменных flag и buf _ nxt _ r для настройки приемного узла на получение очередного кадра.

Настройка адаптера на генерацию прерываний

В данной лабораторной работе организован обмен данными с использованием механизма прерываний . Прием или передача байтов кадра вызывают в линиях интерфейса IRQ3 или IRQ4 запроса на аппаратное прерывание . Линия запроса IRQ3 в машинах IBM PC AT соответствует порту COM2 , линия IRQ4 - порту COM1 . Этим линиям соответствуют номера прерываний 0Bh (для IRQ3) и 0Ch (для IRQ4) . Прерывания от COM - портов обслуживает ISR - программа обработчик , которая принимает символы из сетевой среды и помещает их в приемный буфер , либо передает очередной байт в сетевой интерфейс .

Для организации передачи информации , управляемой механизмом прерываний , необходимо выполнить инициализацию адаптера , которая включает несколько шагов [1] :

- маскируются аппаратные прерывания , так как вся процедура инициализации адаптера должна выполняться без прерываний ;
- сохраняется предыдущий вектор прерывания : 0Ch для COM1 , 0Bh для COM2 в ПЭВМ типа IBM AT ;
- на место сохраненного вектора прерывания записывается адрес входа в свой собственный обработчик прерывания ;
- в адаптере последовательной связи i8250 деблокируются необходимые прерывания (для этого маску разрешения прерываний записывают во внутренний регитр управления прерываниями - ICR) ;
- производится инициализация адаптера последовательной связи (устанавливается скорость обмена данными , задается формат слова и т . п .) ;
- разрешаются прерывания во внутреннем регистре управления модемом MCR (необходимо установить в единицу бит D3 - сигнал OUT2 разрешает прерывания от адаптера) ;
- очищаются все условия прерываний , которые могли уже возникнуть из - за включения механизма прерываний в работу ;
- вновь разрешаются аппаратные прерывания .

Приведенная последовательность состоит из минимально возможного числа шагов и при необходимости может дополнится сохранением внутренних регистров UART и контроллера прерываний . В табл . 4 . 1 приведены функции MS - DOS , наиболее часто использующиеся в ISR - программах .

Таблица 4 . 1

Имя функции	Вызов	Возврат
Завершиться и остаться резидентной	ah = 31h al = код возврата dx = размер резидентной части int 21h	нет
Установить вектор прерывания	ah = 25h al = номер прерывания ds : dx = адрес обработчика int 21h	нет
Получить вектор прерывания	ah = 35h al = номер прерывания int 21h	es : bx = адрес прог. обработчика

Функция 31h прерывания 21h требует , чтобы в dx было загружено число 16 - байтовых параграфов от начала PSP (префикс сегмента программы) [4] . PSP - это область памяти размером 100h байт , которая содержит информацию , необходимую DOS для работы программы .

Запрет аппаратных прерываний в программе на языке Си можно выполнить с помощью макро disable ( ) , а разрешение аппаратных прерываний - с помощью макро enable ( ) .

Ту или иную линию запроса аппаратного прерывания можно маскировать [1 , 4]. Маска запросов хранится во внутреннем регистре контроллера прерываний PIC (Programmable Interrupt Controller) i8259 и может быть прочитана или записана доступом к порту 21h. Каждый бит этого порта соответствует одной линии запроса прерывания . В случае , если бит маски равен нулю , запрос прерывания будет обслуживаться в соответствии с приоритетом . Бит 3 регистра маски запросов прерывания соответствует линии IRQ3 (COM2) , бит 4 - IRQ4 (COM1) .

Когда поступает запрос аппаратного прерывания , контроллер i8259 анализирует значение , записанное во внутреннем регистре обслуживания прерывания (порт 20h) . В этом регистре хранится уровень прерывания , обслуживаемого в текущий момент . Если он ниже уровня приоритета

нового запроса , начинается обслуживание этого запроса . Контроллер сообщает в CPU о необходимости выполнить прерывание и передает его номер . По завершении аппаратного прерывания его уровень приоритета должен быть сброшен в порте 20h . В противном случае менее приоритетные запросы прерывания будут проигнорированы несмотря на то , что более приоритетное прерывание уже обслужено .

Инсталляция обработчика прерывания

Ниже из [1] приведены фрагменты программ на языке Си , выполняющие инсталляцию и обработку прерываний от COM - портов . Функция install _ com ( ) инсталлирует обработчик аппаратного прерывания com _ handler для заданного адаптера последовательной связи (0 - COM1 , 1 - COM2) . Вектор прерывания com _ interrupt (0Bh или 0Ch) зависит от линии запроса прерывания . Для упрощения процесса восстановления функция install _ com ( ) сохраняет во внешних структурных переменных по шаблонам struct COM _ STATE и struct ISR соответственно состояние аппаратуры и предыдущий вектор прерывания .

```
                    /***    Функция install _ com ( )    ***/
# include  < dos . h >
# define   NO _ COM      -1    /* отсутствует запрошенный адаптер */
# define   OK             0    /* инсталляция успешно завершена   */
int  com _ inst ( void ) ;      /* возвращает число адаптеров  */
struct  ISR                     /* вектор прерывания */
{
unsigned  isr _ offset ;        /*  смещение точки входа обработчика */
unsigned  isr _ segment ;       /*  сегмент точки входа обработчика */
unsigned  isr _ number ;        /*  номер вектора прерывания */
} ;
/* ---------------------------------------------------------------------*/
struct  COM _ STATE             /* состояние аппаратуры */
{
char  port _ 21 ;               /* значение порта 21 контроллера i8259 */
char  mcr       ;               /* значение регистра управления модемом */
char  icr       ;               /* значение регистра управления прерыван .*/
char  lcr       ;               /* значение регистра управления линией  */
char  baud0     ;               /* младший байт частоты передачи */
char  baud1     ;               /* старший байт частоты передачи */
} ;
/* ------------------Внешние переменные ----------------------------------*/
```

```
extern struct ISR old_isr ;
extern struct COM_STATE old ;
extern unsigned base_addr ;
/********** Основная часть инсталлятора ********************/
install_com ( int com_interrupt , void interrupt ( * com_handler ( ) ,
              int com_adapter )
{
struct REGPACK r ;        /* структура , объявленная в dos.h */
unsigned far *bios_addr = ( unsigned far* ) 0x00400000L ;
int i = 0xff ; /* отработка задержки */
if (( com_inst ( ) < com_adapter ) || ( com_adapter > 1 ) )
return NO_COM ;
/* ---------- Определение базового адреса адаптера ---------- */
base_addr = *( bios_addr + com_handler ) ;
if ( !base_addr ) return NO_COM ;
disable ( ) ;                    /* запрет аппаратных прерываний */
/* ---------- Сохранение старого вектора прерываний ----------*/
r.r_ax = ( ox35 << 8 ) | com_interrupt ;
intr ( 0x21 , &r ) ;             /* вектор прерывания в ex : bx */
old_isr.isr_offset = r.r_bx ;
old_isr.isr_segment = r.r_es ;
old_isr.isr_number = com_interrupt ;
/*------- Сохранение предыдущего состояния аппаратуры ---------- */
old.port_21 = inportb ( 0x21 ) ;
old.icr     = inportb ( base_addr + 1 ) ;
old.lcr     = inportb ( base_addr + 3 ) ;
old.mcr     = inportb ( base_addr + 4 ) ;
outportb ( base_addr + 3 , old.lcr | 0x80 ) ; /* Установка DLAB */
old.buud0   = inportb ( base_addr ) ;
old.baud1   = inportb ( base_addr + 1 ) ;
outportb ( base-addr + 3 , old.lcr ) ;
/* -------- Установка обработчика прерываний ----------------*/
r.r_ax = ( 0x25 << 8 ) | com_interrupt ;
r.r_dx = FP_OFF ( com_handler ) ;
r.r_ds = FP_SEG ( com_handler ) ;
intr ( 0x21 , &r ) ;
/* --- Демаскирование прерываний от COM1 или COM2 в i8259 -- */
outportb ( 0x21 , inportb ( 0x21 ) & ~ ( 0x10 >> com_adapter )) ;
outportb ( base_addr + 1 , 0x07 ) ; /* разрешить прерывания */
```

```
/********** Инициализация UART i8250 **********************/
outportb ( base_addr + 4 , 0x04 ) ; /* Выдать сигнал OUT2   */
/*---------- Сброс причины возможных прерываний в адаптере ---*/
outportb ( base_addr , 0x00 ) ; /* пуст регистр передатчика  */
inportb  ( base - addr ) ;          /* готовность данных */
inportb  ( base_addr + 6 ) ;    /* изменилось состояние модема */
inportb  ( base_addr + 5 ) ;    /* изменилось состояние линии  */
/*---------------- Разрешить аппаратные прерывания ----------------*/
enable ( ) ;
return  OK ;
}
```

Функция из библиотеки Турбо - Си

```
void  intr ( int intr_num , struc REGPACK  * preg ) ;
```

поддерживает альтернативный интерфейс для выполнения программных прерываний . Прототип функции находится в файле dos . h . Функция intr ( ) генерирует прерывание CPU , заданное аргументом intr . num .

Функция com _ inst ( ) возвращает число адаптеров последовательной связи , установленных в компьютере .

```
int  com_inst ( void )
{
register  unsigned  _es  * com = ( unsigned  _es  * ) 0x10 ;
_ES = 0x40 ;
return ( ( * com  & 0x0E00 ) >> 9 ) ;
}
```

Обработчик аппаратного прерывания

Передача управления обработчику прерывания происходит по вектору прерывания . Получив управление , обработчик com _ handler ( ) выполняет следующие действия :

1) уточняет причину возникновения прерывания чтением содержимого регистра идентификации прерывания IIR ;

2) определив причину , обработчик устраняет ее и выполняет все необходимые действия ;

3) перед завершением своей работы обработчик прерывания анализирует наличие еще не обработанных прерываний , которые могут возникнуть одновременно . При наличии необработанного прерывания (бит D0 = 0 в регистре IIR) продолжает их обработку ;

4) программа работы обработчика должна завершаться инструкцией очистки внутреннего регистра обслуживания прерываний контроллера i8259

```
outportb ( 0x20 , 0x20 ) ;
```

```
/****** COM_ISR.C - модель обработчика прерываний ******/
# include < dos.h >
void interrupt com_handler ( )
{
 extern unsigned base_addr ;
 static char result_isr , line_status , modem_status , received_byte ;
 result_isr = inportb ( base_addr + 2 ) ; /* Чтение регистра IIR */
 do { if ( !( result_isr & 0x01 ) )//Есть ли активное прерывание?
    { switch ( ( result_isr & 0x06 ) >> 1 {
case 0 : /* Изменение регистра состояния модема */
        modem_status = inportb ( base_addr + 6 ) ;
        break ; /* Сброс причины прерывания */
case 1 : /* Регистр данных передатчика пуст */
        /*** Алгоритм работы передающего узла ***/
case 2 : /* Готовность данных в приемном регистре */
        /*** Алгоритм работы приемного узла ***/
case 3 : /* Изиенение регистра состояния линии */
        line_status = inportb ( base_addr + 5 ) ;
          if ( line_status & 0xE0 )
            /*** Секция обработки ошибок ***/
        }
    }
    else break ; /* Выход из цикла обработки прерываний */
    outportb ( 0x20 , 0x20 ) ; /* Очистка регистра прерываний */
}
```

Если обычная Си - функция объявлена с модификатором interrupt , компилятор автоматически добавляет секции сохранения всех регистров CPU в начале кода функции , восстанавливает их в конце этого кода и вместо RET подставляет IRET , обеспечивая правильный возврат из прерывания [1]. Все функции типа interrupt считаются far - функциями .

Головная функция main ( )

Основные задачи , выполняемые функцией main ( ) следующие :

- вызов инсталлятора обработчика прерываний ;
- организация работы передающего и приемного узлов станции ;
- ожидание нажатия клавиши Esc для завершения программы ;
- восстановление состояния адаптера .

```
/******** Модель головной функции main ( ) ***********/
# include < dos.h , bios.h , stdio.h >
```

```
/* ---------- Прототипы используемых функций ----------*/
void interrupt com_handler ( ) ;
int install_com ( int , void interrupt ( ) , int ) ;
int uninstall_com ( int ) ;
/* ---------- Описание переменных и структур ----------*/
struct ISR old_isr ;
struct COM_STATE old ;
. . . . . .
void main ( void )
{
  int ret ; ret = install_com ( 0x0C , com_handler , 0 ) ;
  /* Настройка узлов передающего и приемного узлов станции */
  /* Ожидание нажатия клавиши Esc для завершения программы */
  uninstall_com ( 0 ) ; /* Восстановление состояния адаптера */
}
```

4. 3 . Контрольные вопросы

1. Каким образом осуществляется установка обработчика прерываний ?
2. Чем отличается аппаратное прерывание от программного ?
3. Почему аппаратное прерывание завершается командой IRET ?
4. Каким образом разработать резидентную программу обработки прерываний от COM - портов ?
5. Измените алгоритм работы приемного узла так , чтобы он мог отвергать пустые кадры в полях данных (кадры из символов CR и LF) .

4. 4 . Демонстрации в лаборатории

1. Введите и пошлите из узла S в узел R несколько однострoковых кадров : (фамилия , имя , отчество) .
2. Измените байт PRE и продемонстрируйте эффект несовпадения.
3. Организуйте между узлами разные скорости обмена (например , 1200 бод и 2400 бод) . Проверьте , может ли осуществляться такая связь ?
4. Организуйте дуплексный обмен кадрами между узлами .

4. 5 . Содержание отчета

1. Блок - схемы программ передающего и приемного узлов с подробными комментариями .
2. Листинги разработанных программ .

ЛАБОРАТОРНАЯ РАБОТА № 5
ИССЛЕДОВАНИЕ МНОГОУРОВНЕВОГО СЕТЕВОГО ИНТЕРФЕЙСА

Цель работы : получить навыки разработки и исследования многоуровневого сетевого интерфейса .

5. 1 . Индивидуальное задание

Разработайте и выполните отладку программы , реализующей алгоритмы функционирования сетевого и прикладного уровней станции ЛВС . Настройка адаптера производится в соответствии с исходными данными к лабораторной работе № 1 . При проведении экспериментов организуйте на экране небольшое окно , в которое выведите состояние флагов каждого уровня и межуровневого интерфейса .

5. 2 . Методические указания

В лабораторной работе исследуется многоуровневая сетевая архитектура следующего вида : терминал -- прикладной уровень -- межуровневый интерфейс -- сетевой уровень -- физический уровень [2] .

Кадры (PRE , . . . данные . . . , CR , LF , EOT) создаются с помощью клавиатуры терминала . Прикладной уровень (ПУ) принимает кадры , поступающие из терминала , и делает запрос сетевому уровню на их передачу физическому уровню . В противоположном направлении прикладной уровень принимает кадры по запросу из сетевого уровня и отображает их на экране терминала .

Сетевой уровень (СУ) принимает кадры , поступающие по кабелю из физического уровня и делает запрос прикладному уровню на их обработку и отображение на экране терминала . В противоположном направлении СУ по запросу от ПУ осуществляет передачу кадров физическому уровню и далее в моноканал .

Физический уровень реализован аппаратно и представляет собой трехпроводную последовательную связь между сетевыми компьютерами с помощью UART i8250 . Прикладной и сетевой уровни реализуются программно . Обмен кадрами между СУ и ПУ осуществляется с помощью межуровневого интерфейса (рис. 5.1).

Этот интерфейс имеет буферы приема (RBUF) и передачи (TBUF), а также ряд флагов , доступ к которым производится обоими уровнями . Назначение флагов межуровневого интерфейса следующее :

TBF - флаг “ буфер передачи заполнен “ ;

RBF - флаг “ буфер приема заполнен “ ;

SP - флаг запроса от ПУ к СУ на передачу кадра ;

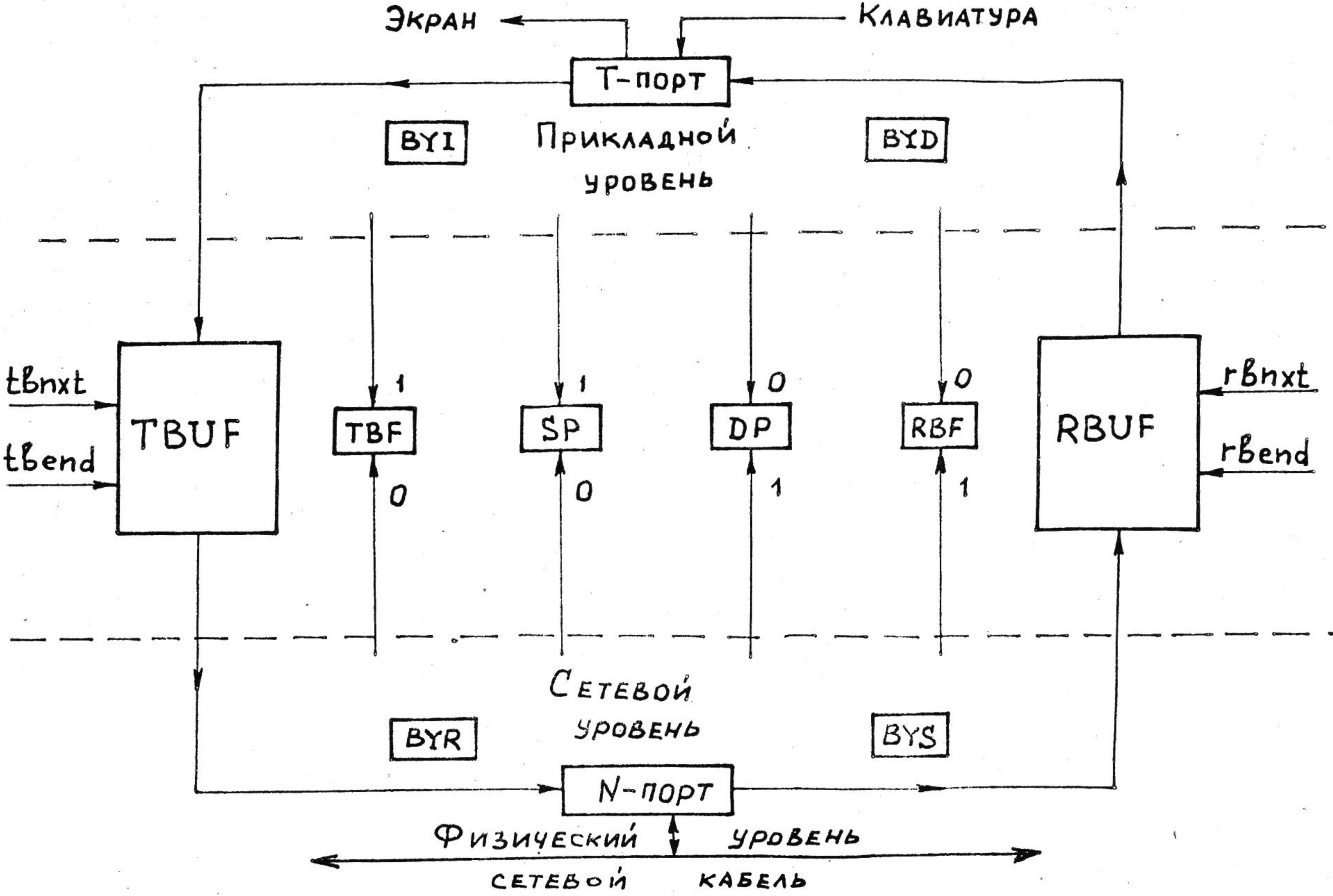

Рис . 5 . 1 . Структура межуровневого интерфейса

DP - флаг запроса от СУ к ПУ на отображение кадра.

Прикладной уровень принимает побайтно кадр с клавиатуры терминала и передает его в буфер передачи TBUF. После завершения приема ПУ устанавливает флаг SP передачи кадра, чтобы информировать СУ о том, что в сеть необходимо передать кадр. Как только СУ определит, что флаг SP взведен, он передает кадр из буфера TBUF в сетевой кабель.

Сетевой уровень в приемном узле принимает кадр из сети и помещает его в буфер приема RBUF. После завершения приема он устанавливает флаг DP, чтобы проинформировать ПУ о том, что из сети поступил новый кадр и его необходимо обработать. Когда ПУ определит, что флаг DP установлен, он извлекает кадр из буфера RBUF и отображает его на экране терминала.

Рассмотрим подробные алгоритмы работы прикладного и сетевого уровней.

Прикладной уровень

Работа ПУ может быть описана в виде конечного автомата, имеющего три состояния: "готов", "ввод" и "вывод" [2]. В состоянии "готов" ПУ не выполняет никаких действий. В состоянии "ввод" происходит заполнение буфера TBUF байтами кадра, вводимыми с клавиатуры терминала. В состоянии "вывод" ПУ извлекает кадр из буфера приема RBUF и отображает его на экране терминала (рис. 5.2). В табл.5.1 показана кодировка состояний ПУ с помощью флагов BYI и BYD. Работа конечного автомата описывается с помощью табл. 5.2.

Таблица 5.1

Кодирование состояний автомата ПУ

Состояние	Флаг BYI	Флаг BYD
Готов	0	0
Ввод	1	0
Вывод	0	1

Когда на клавиатуре терминала нажимается клавиша, в то время как ПУ находится в состоянии "готов", предполагается, что начинается создание (ввод) сообщения и ПУ меняет состояние на "ввод". Если обнаруживается, что установлен флаг DP, когда ПУ находится в состоянии "готов", считается, что сетевой уровень принял сообщение и оно должно быть отображено. ПУ меняет свое состояние на "вывод".

Таблица 5 . 2

Таблица переходов автомата ПУ

Тек . сост.	След. сост.	Условия	Действия
Готов	Ввод	Первый байт и TBF = 0	BYI : = 1
Готов	Вывод	DP = 1	DP : = 0 , BYD : = 1
Ввод	Готов	Ввод CR с клавиатуры	SP : = TBF : = 1 , BYI: =0
Вывод	Готов	Отображен послед. символ	BYD : = RBF : = 0

Если бы автомату ПУ было разрешено менять состояние “ ввод “ на “ вывод “ до того , как закончится ввод кадра , тексты обоих кадров (выводимого и отображаемого) на экране смешались бы.

Сетевой уровень

Конечный автомат , описывающий поведение СУ , имеет три состояния : “ готов “ , “ прием “ и “ передача “. В состоянии “ готов “ автомат не выполняет никаких действий . Он лишь анализирует каждый байт , поступающий из сети . Если принят байт преамбулы PRE , то автомат переключается в состояние “ прием “ . В этом состоянии происходит ввод кадра из сети в буфер приема RBUF (рис . 5 . 3).

Если автомат СУ находясь в состоянии “ готов “ обнаружит взведенный флаг SP (ПУ завершил формирование нового кадра) , он переходит в состояние “ передача “ . В этом состоянии из буфера TBUF извлекаются байты кадра и передаются в сетевой кабель (табл . 5 . 4).

Таблица 5 . 3

Кодирование состояний автомата СУ

Состояние	Флаг BYR	Флаг BYS
Готов	0	0
Прием	1	0
Передача	0	1

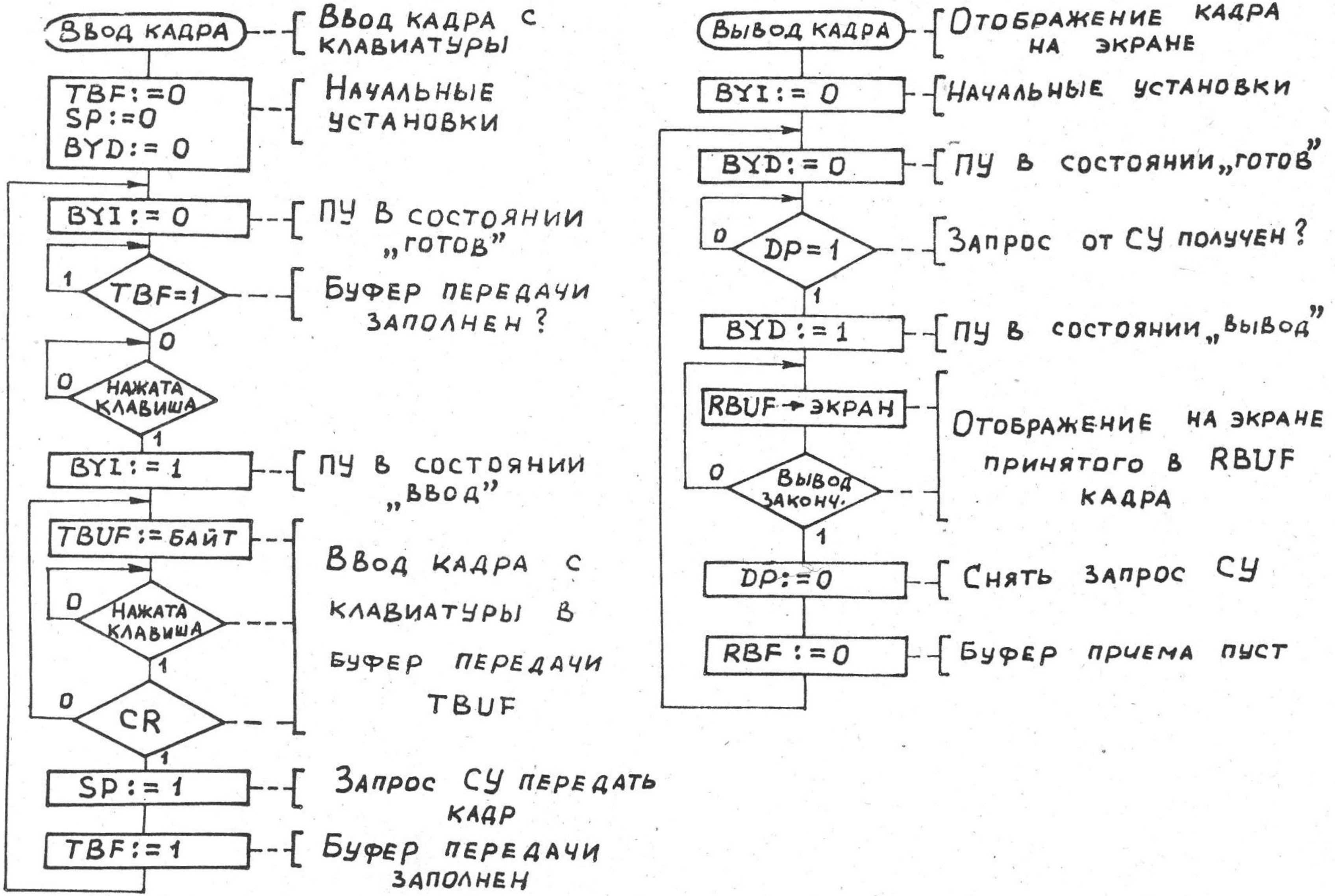

Рис . 5 . 2 . Алгоритмы работы прикладного уровня

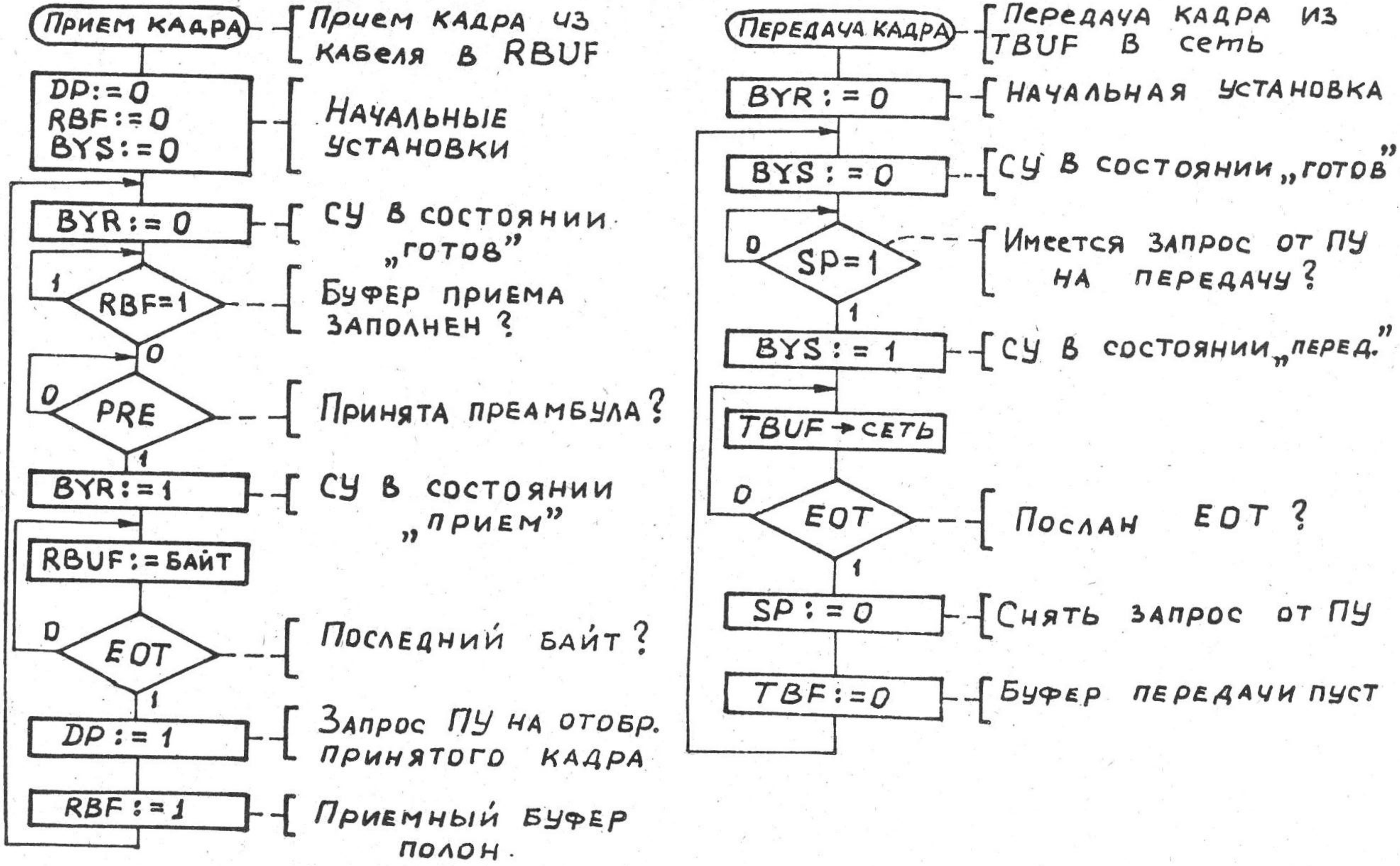

Рис . 5 . 3 . Алгоритмы работы сетевого уровня

Таблица 5 . 4

Таблица переходов автомата СУ

Тек. сост.	След. сост.	Условия	Действия
Готов	Прием	Получен PRE и RBF = 0	BYR : = 1
Готов	Передача	SP = 1	SP : = 0 , BYS : = 1
Прием	Готов	Получен EOT	RBF := DP : = 1 , BYR := 0
Передача	Готов	Послан EOT	BYS : = TBF : = 0

Переходы между состояниями “ прием “ и “ передача “ не разрешены.

Реализация алгоритмов

Все программные модули разделены на три части [2] :

Основная часть	Прикладной уровень	Сетевой уровень
Main	Tisr	Nisr
Clear _ flags	Input	Receive
Initialize	Echo _ byte	Transmit
Schedule	Output	Enable _ receive
Start _ send		
Start _ display		

Программные модули основной части

Модуль Main условно можно разбить на две части : последовательность шагов инициализации и цикла планирования . На последнем шаге разрешаются аппаратные прерывания , а CPU переходит в режим пользователя . Начиная с этого момента производится циклический вызов программы планирования Schedule , пока не произойдет изменение состояния системы нажатием на клавишу Esc .

```
const
   PRE , CR , LF , EOT : character     / *  служебные символы  * /
   rborg , tborg : integer             / * начальные указатели буферов  * /
var
   SP , DP , TBF , RBF , BYS , BYR , BYD , BYI : logical /* флаги * /
RBUF , TBUF : array   / * буферы приема и передачи  * /
rbnxt , tbnxt , rbend , tbend : pointer  / * текущие указатели буферов */
byte : character              / *  байт символа  * /
```

```
begin
   Clear _ flags                    / * сбросить все флаги  * /
    Initialize                       / * инициализация UART  * /
    enable CPU interrupt             / * разрешить прерывания CPU  * /
    repeat Schedule until Esc / * повторять планирование до Esc  * /
end .
```

Модуль Clear _ flags устанавливает в нулевое положение все флаги .

В модуле Initialize инициализируются указатели буферов передачи TBUF и приема RBUF , настраиваются вектора прерываний терминального (T - порт) и сетевого (N - порт) портов , а также инициализируется UART i8250 .

Первый байт буфера TBUF всегда содержит байт PRE , поэтому этот байт необходимо загрузить при инициализации . Ожидается , что первой операцией , выполняемой с буфером TBUF , является ввод данных уровнем ПУ . Поэтому указатель tbnxt буфера TBUF устанавливается при инициализации в tborg + 1. Принятый байт PRE нет необходимости помещать в буфер RBUF . Поэтому указатель rbnxt после инициализации должен указывать на первый байт в буфере RBUF .

Модуль планирования Schedule с помощью флагов SP и DP активизирует уровневые процессы . Эти процессы также активизируются непосредственно прерываниями портов .

```
if SP = 1 and BYS = 0 then          / * закончен ввод кадра и СУ готов */
        Start _ send
else if DP = 1 and BYD = 0 and BYI = 0 then / * принят кадр и ПУ готов * /
                Start _ display .
```

Для посылки кадра из буфера TBUF в сеть модуль Start _ send активизирует сетевой уровень . Флаг BYS устанавливается в единицу , чтобы отметить изменение состояния СУ , а чтобы гарантировать одноразовость передачи кадра , производится сброс флага SP . Наконец , в сетевом порте разрешается прерывание передачи и начинается сама передача.

Модуль Start _ display активизирует ПУ для отображения кадра из буфера RBUF на экране терминала . Работа этой программы аналогична работе программы Start _ send . Флаги BYD и DP заменяются на флаги BYS и SP , а вместо сетевого порта используется терминальный порт .

Программные модули прикладного уровня

Все операции ПУ запускаются прерываниями терминального порта (по нажатию клавиши) , которые обрабатываются программой Tisr . В зависимости от состояния ПУ в момент , когда запускается Tisr , производится вызов программ Input или Output . Программа Input за один раз вводит из терминального порта (клавиатуры) в буфер TBUF один байт . Принятые без ошибок байты кадра отображаются на экране программой Echo _ byte . Когда с клавиатуры вводится символ CR , ввод завершается и делается запрос на передачу кадра . Программа вывода Output за один раз выводит на экран один байт из буфера RBUF . После отображения всех байтов вывод заканчивается . В этот момент происходит запрет прерываний терминального порта , чтобы сделать возможным операции ввода .

if BYD = 1 then Output else Input .

Программные модули сетевого уровня

Операции сетевого уровня запускаются прерываниями сетевого порта , которые обрабатываются программой Nisr. В зависимости от текущего состояния вызывается либо программа приема Receive , либо программа передачи Transmit . Программа Receive за один раз вводит один байт из сетевого порта в буфер RBUF . Ввод принятых байтов в RBUF начинается после приема байта PRE при условии , что RBUF свободен , а СУ уже не занят , и завершается после приема байта EOT . В этот момент делается запрос на отображение кадра .

Программа Transmit за один раз осуществляет вывод одного байта из буфера TBUF в сетевой порт . Вывод заканчивается после того , как переданы все байты из TBUF . Затем вызывается программа разрешения приема Enable _ receive , чтобы очистить сетевой порт от байтов , случайно принятых во время передачи .

if BYS = 1 then Transmit else Receive .

5. 3 . Контрольные вопросы

1. Что выгоднее : хранить байты PRE и EOT в буфере TBUF или присоединить их “ по ходу дела “ во время передачи кадра ?
2. Как реализовать режим отбрасывания кадров с ошибками CRC ?

5. 4 . Демонстрации в лаборатории

1.Используйте два дуплексных узла (A и B) . Пошлите кадры (Ф , И , О) из A в B и из B в A .