**Лабораторная работа №3**

**Интерфейс RS-232**

**Цель:**

1. Изучить принципы обмена информацией по интерфейсу RS232;

2. Получить навыки использования интерфейса RS232 для передачи пакетов данных.

**Задание:**

Реализовать обмен данными между двумя устройствами со скоростью 19200 бит/с.

В передаваемый пакет входит:

Старт байт = 55h

Длина поля данных от 1 до 20 (число, байт данных)

Контрольная сумма(дополняет сумму остальных элементов пакета до 0) =>

=> Необходимо принять пакет - инвентировать побайтово и отправить обратно с тем же форматом пакета.

Тактовая частота МК 8 МГц.

**Теория:**

RS —стандарт [физического уровня](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) для [асинхронного интерфейса (UART)](https://ru.wikipedia.org/wiki/UART). Устройство, поддерживающее этот стандарт, широко известно как [последовательный порт](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%80%D1%82) [персональных компьютеров](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). Исторически стандарт имел широкое распространение в [телекоммуникационном](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8) оборудовании. В настоящее время используется для подключения к компьютерам широкого спектра оборудования, нетребовательного к скорости обмена, особенно при значительном удалении его от компьютера и отклонении условий применения от стандартных. В компьютерах, занятых офисными и развлекательными приложениями, практически вытеснен интерфейсом [USB](https://ru.wikipedia.org/wiki/USB).

RS-232 обеспечивает передачу данных и некоторых специальных сигналов между терминалом ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Data Terminal Equipment, DTE) и коммуникационным устройством ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Data Communications Equipment, DCE) на расстояние до 15 метров на максимальной скорости (115200 [бод](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%B4)). Так как этот интерфейс известен не только простотой программирования, но и неприхотливостью, в реальных условиях это расстояние увеличивается во много раз с примерно пропорциональным снижением скорости.

Протокол интерфейса предполагает два режима передачи данных — [синхронный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [асинхронный](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), а также два метода управления обменом данных: аппаратный и программный. Каждый режим может работать с любым методом управления. В протоколе также предполагается вариант управления передачей данных по специальным сигналам, устанавливаемым хостом (DSR — сигнал состояния готовности, DTR — сигнал готовности передачи данных).

Для передачи данных по интерфейсу RS-232 используется [код NRZ](https://ru.wikipedia.org/wiki/NRZ_(%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B9)), который не является самосинхронизирующимся, поэтому для синхронизации используются стартовый и стоповый биты, позволяющие выделить битовую последовательность и синхронизировать приёмник с передатчиком.

**Код:**

;RXCIE TXCIE UDRIE RXEN TXEN UCSZ2 RXB8 TXB8 UCSRB

.include "m16def.inc"

.def temp = R16

.def cnt\_bytes = R17

.def chck\_sum = R18

.def chck\_sum\_pos = R19; позиция байта контрольной суммы

.def data\_length = R20; количество байт данных

.def check\_all\_transmitted = R21;

jmp RESET ; Reset Handler

jmp EXT\_INT0 ; IRQ0 Handler

jmp EXT\_INT1 ; IRQ1 Handler

jmp TIM2\_COMP ; Timer2 Compare Handler

jmp TIM2\_OVF ; Timer2 Overflow Handler

jmp TIM1\_CAPT ; Timer1 Capture Handler

jmp TIM1\_COMPA ; Timer1 CompareA Handler

jmp TIM1\_COMPB ; Timer1 CompareB Handler

jmp TIM1\_OVF ; Timer1 Overflow Handler

jmp TIM0\_OVF ; Timer0 Overflow Handler

jmp SPI\_STC ; SPI Transfer Complete Handler

jmp USART\_RXC ; USART RX Complete Handler

jmp USART\_UDRE ; UDR Empty Handler

jmp USART\_TXC ; USART TX Complete Handler

jmp ADC\_C ; ADC Conversion Complete Handler

jmp EE\_RDY ; EEPROM Ready Handler

jmp ANA\_COMP ; Analog Comparator Handler

jmp TWSI ; Two-wire Serial Interface Handler

jmp EXT\_INT2 ; IRQ2 Handler

jmp TIM0\_COMP ; Timer0 Compare Handler

jmp SPM\_RDY ; Store Program Memory Ready Handler

;

RESET:

ldi temp,high(RAMEND) ; Main program start

out SPH,temp ; Set Stack Pointer to top of RAM

ldi temp,low(RAMEND)

out SPL,temp

;скорость 19200 бит в сек при 8мгц

ldi temp,0

out UBRRH,temp

ldi temp,25

out UBRRL,temp

; настройка на прием в ассинхроном режиме 8 бит данных

ldi temp,(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0)

out UCSRC,temp

ldi temp,(1<<RXCIE)|(1<<RXEN)|(0<<UCSZ2) ;прерывание по приему и приемник

out UCSRB,temp

clr cnt\_bytes ;очищаем счетчик принятых байт

sei ; Enable interrupts

LOOP:

in temp, UCSRA

sbrc temp, RXC ;

call USART\_RXC

in temp, UCSRA

sbrc temp, TXC ;

call USART\_TXC ;

rjmp LOOP

ADC\_C: ; ADC Conversion Complete Handler

TIM1\_COMPA: ; Timer1 CompareA Handler

EXT\_INT0: ; IRQ0 Handler

EXT\_INT1: ; IRQ1 Handler

TIM2\_COMP: ; Timer2 Compare Handler

TIM2\_OVF: ; Timer2 Overflow Handler

TIM1\_CAPT: ; Timer1 Capture Handler

TIM1\_COMPB: ; Timer1 CompareB Handler

TIM1\_OVF: ; Timer1 Overflow Handler

TIM0\_OVF: ; Timer0 Overflow Handler

SPI\_STC: ; SPI Transfer Complete Handler

USART\_UDRE: ; UDR Empty Handler

EE\_RDY: ; EEPROM Ready Handler

ANA\_COMP: ; Analog Comparator Handler

TWSI: ; Two-wire Serial Interface Handler

EXT\_INT2: ; IRQ2 Handler

TIM0\_COMP: ; Timer0 Compare Handler

SPM\_RDY:; Store Program Memory Ready Handler

reti

USART\_RXC: ; USART RX Complete Handler

in temp,UDR ;получение данных от приемника

cpi cnt\_bytes,0

breq check\_start\_byte

cpi cnt\_bytes,1

breq check\_data\_length

cp cnt\_bytes,chck\_sum\_pos; проверка принятия последнего байта в пакете

breq check\_chck\_sum

st X+,temp ;сохранение байтов данных

inc cnt\_bytes

add chck\_sum,temp; учли принятый байт в контр сумме

reti

check\_start\_byte:

cpi temp,0x55

brne not\_start\_byte

inc cnt\_bytes

ldi chck\_sum,0x55;заносим в контр суму первое значение 55h

not\_start\_byte:

reti

check\_data\_length:

cpi temp,1

brlo Error

cpi temp,21

brsh Error

mov data\_length, temp ; занесли кол-во байт в data\_length

ldi XL,LOW(Data\_buffer)

ldi XH,HIGH(Data\_buffer)

inc cnt\_bytes

add chck\_sum,temp; учли принятый байт в контр сумме

ldi chck\_sum\_pos,2

add chck\_sum\_pos,temp; вычислили позицию байта контрольной суммы по формлуе (длина+2)

reti

Error:

clr cnt\_bytes ;для возвращения к началу приема

reti

check\_chck\_sum:

add chck\_sum,temp; учли принятый байт в контрольной сумме

cpi chck\_sum,0

breq transform\_packet

clr cnt\_bytes

reti

transform\_packet: ;цикл обработки данных (инвентируем принятые данные и возвращаем)

ldi XL,LOW(Data\_buffer)

ldi XH,HIGH(Data\_buffer) ;теперь Х адрес начало буффера

mov cnt\_bytes, Data\_length ;

invert:

ld temp, X ;загружаем в темп значение из ячейки памяти на которую ссылается указатель Х

com temp ;инвентируем значение (всех восьми битов по-битно)

st X+, temp; сохраняем в буффер обратно, а также инкрементируем указатель

dec cnt\_bytes ; уменьшили значение счетчика с последующей проверкой

brne invert ; не 0 - зацикливаемся, 0 - выходим из цикла

ldi temp,(1<<TXCIE)|(1<<TXEN)|(0<<UCSZ2);

out UCSRB,temp;переключаемся в режим передачи

ldi temp,0x55;стартовый байт

out UDR,temp;отправляем на UDR

mov chck\_sum, temp;подготавливаем для подсчета отправленных байтов (контрол суммы отпр пакета)

clr cnt\_bytes ;на всякий случай

clr check\_all\_transmitted; сброс признака передачи всего пакета

reti

USART\_TXC: ; USART TX Complete Handler

;прерывание по окончанию передачи байта

inc cnt\_bytes;

cpi cnt\_bytes,1;сравниваем с 1 и переходим

breq send\_length

;если не 1 - сюда

cp cnt\_bytes,chck\_sum\_pos; проверка отправки последнего байта в пакете

breq send\_chck\_sum

cpi check\_all\_transmitted, 0;проверка передан ли весь пакет

brne stop\_transmiting

;если не breq send\_length и не breq send\_chck\_sum и не все передано, то ниже

;передаем данные

ld temp, X+;извлекаем данные из буффера с последующим инкрементов указателя

out UDR,temp;отправляем данные

add chck\_sum, temp ;для учета в контрольной сумме

reti

stop\_transmiting:;перестройка приема передачи

ldi temp,(1<<RXCIE)|(1<<RXEN)|(0<<UCSZ2) ;прерывание по приему и приемник

out UCSRB,temp

clr cnt\_bytes

reti

send\_length:

ldi XL,LOW(Data\_buffer)

ldi XH,HIGH(Data\_buffer)

out UDR,data\_length; передаем длину поля данных

add chck\_sum, data\_length; учет в контрольной сумме

;end

reti

send\_chck\_sum:

;накопили сумму и надо '0 - сумма'

;clr temp

;sub temp, chck\_sum ;результат в темпе

NEG chck\_sum ;Two’s Complement Rd < $00 ? Rd;результат в chck\_sum

out UDR,chck\_sum

set check\_all\_transmitted

reti

.dseg

Data\_buffer: ;принятые данные

.byte 20

**Выводы.** На лабораторной работе изучили интерфейсы передачи данных и реализовали обмен данными между двумя устройствами.