**Лабораторна робота №3**

**Тема:** Інтерфейс RS-232

**Мета:** засвоїти принципи обміну інформацією за інтерфейсом RS232; Отримати навики використання інтерфейсу RS232 для передачі пакетів даних.

**Теоретичні відомості**

RS-232 - це інтерфейс передачі інформації між двома пристроями на відстань до 15м.

Використовується для підключення до обчислювальних машин самого різного устаткування, невибагливого до швидкості обміну.

В офісній техніці практично витіснений інтерфейсом USB. Зате активно використовується для підключення периферії і пристроїв, розташованих досить далеко від комп'ютера, і навіть працюють в складних умовах зовнішнього середовища. Також цей стандарт використовується для взаємодії мікроконтролерів різних архітектур, що мають інтерфейс UART, з іншими цифровими пристроями і периферією.

* Відстань 15 м.
* Швидкість передачі даних до 115200 біт / с.
* Тип точка-точка (master-slave).
* Можна підключити тільки один пристрій.
* Протокол інтерфейсу передбачає синхронний і асинхронний режим передачі даних.
* Апаратний і програмний метод управління обміном даних.
* Для організації RS232 використовуються роз'єми DB9M і DB25M.

Інтерфейс RS232 є послідовним асинхронним повнодуплексним інтерфейсом передачі даних, який часто використається для забезпечення сполучення МК та систем вищого рівня. Інтерфейс RS232 використовує дві однонаправлені лінії даних (TxD та RxD). Як правило, використовується топологія „точка-точка”.

В стані паузи на лініях TxD та RxD присутній рівень логічної 1 (рис. 1). Передача починається з видачі стартового біта (S), який потрібен для виявлення початку обміну приймачем. Стартовий біт завжди дорівнює логічному 0, тобто приймач починає прийом по першому спадаючому фронту сигналу. За ним передаються біти даних (D0-D7) молодшим бітом вперед. Після передачі бітів даних видається стоповий біт (P), який повинен дорівнювати логічній 1 (це забезпечує мінімально необхідну паузу між кадрами даних). За стоповим бітом може слідувати стан паузи або може починатися передача наступного байту.

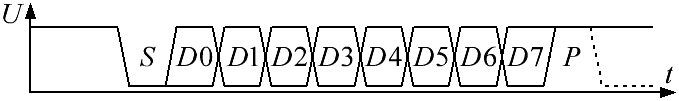


Рис. 1 Часова діаграма передачі 8-бітних даних

Швидкість обміну передавача повинна співпадати із швидкістю обміну приймача. Якщо ці швидкості не співпадають, то приймач буде отримувати некоректну інформацію. Для часткового контролю швидкості може використовуватися стоповий біт. Якщо приймач отримує на місці стопового біта логічний 0, то фіксується помилка кадру. Байт, що прийнято з помилкою кадру, повинен бути відкинутий. Швидкість обміну задається у вигляді дільника в регістрі швидкості.

Окрім передачі 8 біт даних в одному кадрі, існує можливість 9 бітної передачі (рис.2). Додатковий біт D8 може використовуватись для контролю передачі (наприклад, біт парності) або для зазначення певних ситуацій.

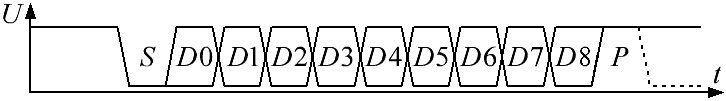


Рис. 2 Часова діаграма передачі 9-бітних даних

МК часто забезпечують підтримку апаратного детектування адреси за допомогою додаткового біту. Якщо додатковий біт дорівнює логічному 0, то прийнятий байт відкидається. Якщо додатковий біт дорівнює логічної 1, то генерується переривання, а прийнятий байт порівнюється з адресою пристрою. Такий режим дозволяє МК ефективно працювати з інтерфейсами RS-422 та RS-485, які також є асинхронними інтерфейсами, але вони використають лише одну двонаправлену диференційну лінію даних. Інтерфейси RS-422 та RS-485 використовують топологію «загальна шина».

**Завдання**

Реалізувати обмін даними між двома пристроями зі швидкістю 19200 біт / с.

В пакет, який передається входить:

Старт байт = 55h.

Довжина поля даних від 1 до 20 (число, байт даних)

Контрольна сума (доповнює суму інших елементів пакету до 0) =>

=> Необхідно прийняти пакет - інвертувати побайтово і відправити назад з тим же форматом пакета.

Тактова частота МК 8 МГц.

**Код програми**

;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) ; **Старт основной программы**

out SPH,temp ; **Установите указатель стека в верхнюю часть ОЗУ**

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 приём в асинхронном 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 приёму и приёмник**

out UCSRB,temp

clr cnt\_bytes ;**очищаем счетчик принятых байт**

sei ; **Разрешить прерывания**

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 пустой обработчик**

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 полный обработчик**

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 формуле (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 (инвертируем принятые данные и возвращаем)**

ldi XL,LOW(Data\_buffer)

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

mov cnt\_bytes, Data\_length ;

invert:

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

com temp ;**инвертируем значение (всех восьми битов побитно)**

st X+, temp; **сохраняем в буфер 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 полный обработчик**

;**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 буфера 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

**Висновок:** під час виконання лабораторної роботи ми засвоїли принципи обміну інформацією за інтерфейсом RS-232 та отримали навики використання інтерфейсу RS-232 для передачі пакетів даних.Реалізовано обмін даними між двома пристроями зі швидкістю 19200 біт / с.