Лекція 12

Інтерфейс RS-232.

Інтерфейс RS-232 був розроблений для забезпечення зв'язку між термінальним обладнанням та апаратурою передачі даних, використовуючи послідовний обмін двійковими даними.

Стандарт RS-232 був розроблений у 1969 році американською Асоціацією електронної промисловості, та після незначних поправок отримав назву RS-232C. У 1991 році була здійснена модифікація цього стандарту, після чого він отримав назву EIA/TIA-232E. Інша відома назва цього протоколу ITU V.24. Загально прийнятою назвою є EIA-232, або просто RS-232.

У стандарті передбачені асинхронний та синхронний режими обміну, однак переважна більшість пристроїв (наприклад ПК) працюють лише в асинхронному режимі.

Варто зазначити, що інтерфейс не забезпечує гальванічної розв'язки пристроїв.

Згідно інтерфейсу RS-232 (рис. 4,9), логічній «1» відповідає напруга на вході приймача в діапазоні від -12 до -3 В. Для ліній послідовних даних цей стан називається «MARK» (лог. «1»). Логічному «0» відповідає діапазон від +3 до +12 В. Для послідовних даних цей стан називається SPACE. Діапазон від -3 до +3 В – зона нечутливості, обумовлена гістерезисом приймача: стан ліній буде рахуватися зміненим лише після перетинання порогу чутливості. Рівні сигналів на виходах передавачів повинні бути в діапазонах від -12 до -5 В та від +5 до +12 В для представлення, відповідно, «1» та «0». Різниця потенціалів між схемними землями з'єднувальних пристроїв повинна бути меншою 2 В, інакше можливе неправильне сприйняття сигналів.

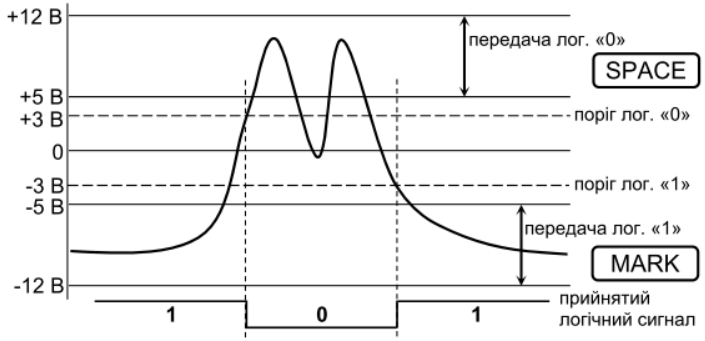


Рис. 4.9. Прийом сигналів RS-232

Формат кадру інтерфейсу RS-232 співпадає з форматом кадру модуля UART (п.4.9а, стр. 109). На рис. 4.10 зображено формат кадру для числа 82 у протоколі RS-232, що є подібним до представлення цього числа у TTL-рівнях модуля UART (рис. 4.3).

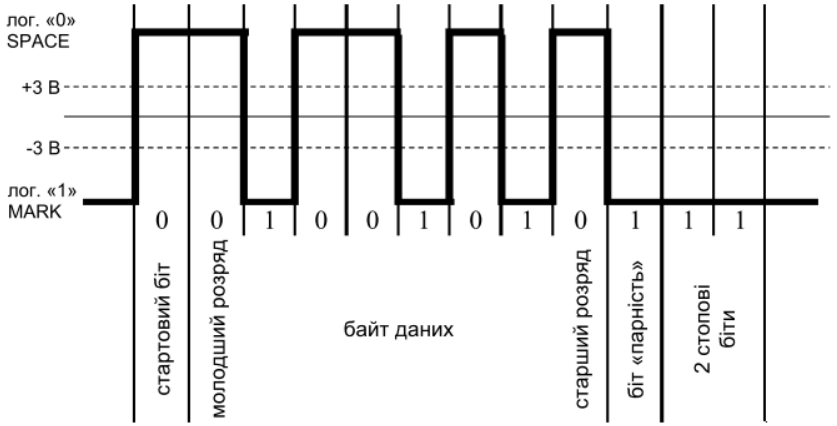


Рис. 4.10. Представлення числа 82 (01010010_2)
у кадрі 8n2 з перевіркою на парність у рівнях RS-232

Для підключення МК через модуль UART до COM-порту ПК необхідно виконати перетворення TTL-рівнів у рівні інтерфейсу RS-232 та навпаки за допомогою спеціалізованої мікросхеми, наприклад, MAX232.

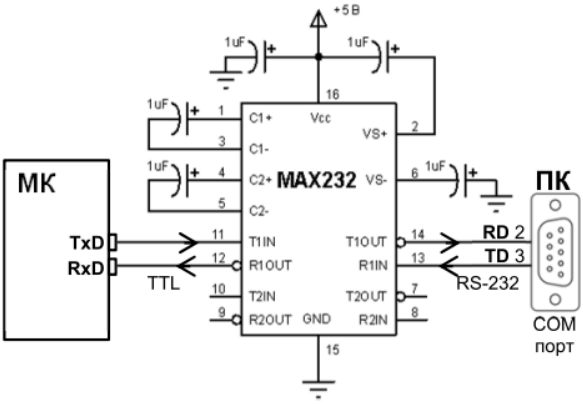


Рис. 4.10. Підключення модуля UART до ПК

Якщо на ПК відсутній фізичний COM - порт, тоді необхідно або використовувати плати розширення PCI з підтримкою RS-232 чи, скажімо, адаптери USB/RS-232, або спеціалізовані мікросхеми на зразок FT232RL.

4.12. Інтерфейс RS-485.

RS-485 (інша назва EIA/TIA-485) – найпоширеніший стандарт фізичного рівня зв'язку (канал зв'язку + спосіб передачі сигналу). Цей інтерфейс забезпечує обмін даними між декількома пристроями по одній двопровідній лінії зв'язку в напівдуплексному режимі. Для каналу зв'язку вибирається вита пара.

В основі інтерфейсу RS-485 лежить принцип диференціальної (балансної) передачі даних. По одному дроті (умовно А) іде оригінальний сигнал, а по іншому (В) – його інверсна копія. Тобто, якщо на одному дроті «1», то на іншому «0», і навпаки. Тому між двома дротами витої пари завжди є різниця потенціалів»: при логічній «1» вона позитивна, а при «0» – негативна (рис. 4.11). Такий спосіб передачі забезпечує високу стійкість до синфазних перешкод (що діють на два дроти лінії одночасно).

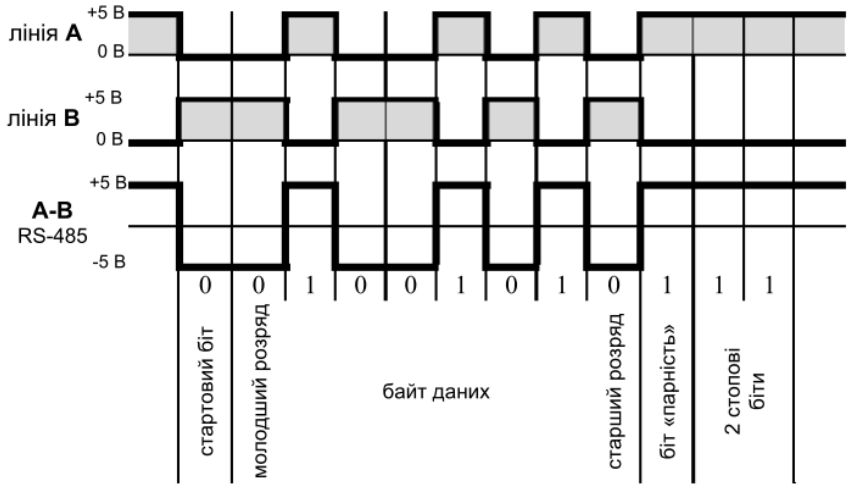


Рис. 4.11. Представлення числа 82 (01010010_2)
у кадрі 8n2 з перевіркою на парність у рівнях RS-485

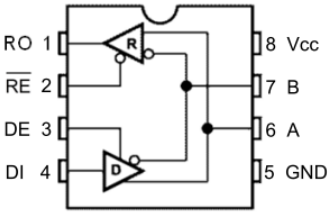
Стандарт RS-485 описує лише фізичний рівень процедури обміну даними. Його основні задачі це:

– перетворення вхідної послідовності «1» та «0» у диференціальний сигнал;
– передача диференціального сигналу в симетричну лінію зв'язку;
– підключення чи відключення передавача драйвера згідно сигналу верхнього протоколу обміну;
– прийом диференціального сигналу з лінії зв'язку.

Решта особливостей обміну, синхронізації та квітування покладається на верхній протокол обміну, наприклад RS-232 чи ModBus.

RS-485 забезпечує передачу даних зі швидкістю до 10 Мбіт/сек. Максимальна дальність залежить від швидкості: при швидкості 10 Мбіт/сек максимальна довжина лінії – 120 метрів, при швидкості 100 Кбіт/сек – 1200 метрів.

Апаратно інтерфейс реалізовується за допомогою спеціалізованих мікросхем прийомопередавачів з диференціальними входами/виходами (до лінії зв'язку) та цифровими портами (до портів UART МК), наприклад, MAX485(Maxim), ST485 (STMicroelectronics) (рис. 4.12).



Умовні позначання:
D (driver) – передавач
R (receiver) – приймач
RO (receiver output) – цифровий вихід приймача
$\overline{RE}$ (receiver output enable) – дозвіл роботи приймача
DE (driver output enable) – дозвіл роботи передавача
DI (driver input) – цифровий вхід передавача
A – прямий диференціальний вхід/вихід
B – інверсний диференціальний вхід/вихід

Рис. 4.12. Умовні позначення виводів MAX485, ST485

Цифровий вихід RO підключається до RxD приймача UART, а цифровий вхід DI до TxD передавача UART (рис. 4.13). Оскільки на диференціальній стороні приймач та передавач об'єднані, то на час приймання необхідно відключати передавач, а на час передачі – приймач. Для цього призначені керуючі входи $\overline{RE}$ (дозвіл приймача) та DE (дозвіл передавача). Вхід дозволу роботи приймача є інверсним, тому його можна об'єднати з входом дозволу роботи передавача, та одним сигналом з будь-якого порту МК перемикати між собою приймач та передавач. Якщо рівень «1» – працює передавач, якщо ж «0» – приймач.

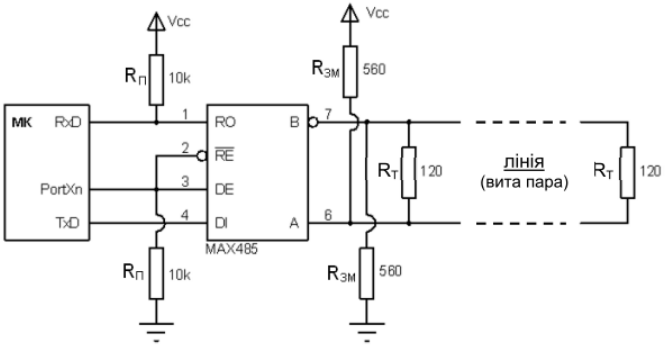


Рис. 4.13. Підключення прийомопередавача RS-485 до МК

Під час роботи прийомопередавача RS-485 на передачу вихід приймача RO переводиться у третій стан і вивід RxD МК «повисає у повітрі». Тому, замість високого рівня «1» відсутності передачі, будь-яка завада може бути прийнята за вхідний сигнал. Для цього необхідно на час передачі відключати приймач UART (через керуючий регістр), або підтягувати вивід RxD за допомогою резистора до «1». Також можна на МК задіяти внутрішній підтягуючий резистор.

При подачі живлення на МК пройде певний час, поки вивід керування дозволами роботи драйвера RS-485 буде проініціалізований на вихід. До цього моменту він буде функціонувати як високоімпедансний вхід. Тому існує можливість, що якоюсь завадою буде активований передавач RS-485, і у лінію буде відправлене «сміття». Щоб уникнути цього, рекомендується виводи керування роботою драйвера підтягнути резистором до «землі».

Чутливість приймача RS-485 може бути різною, але гарантований граничний діапазон розпізнавання сигналу становить ±200 мВ. Тобто, коли U_{AB}>+200 мВ – приймач визначає «1», якщо U_{AB}<-200 мВ, тоді приймач визначає «0». Якщо різниця потенціалів є меншою за визначені допустимі границі, тоді правильне розпізнавання сигналу не гарантується. Таке може трапитися або при від'єднанні приймача від лінії, або при відсутності в лінії активних передавачів, коли ніхто не задає рівень. Щоб уникнути видачі помилкових сигналів на вхід UART, необхідно на входах A-B гарантувати різницю потенціалів U_{AB}>+200 мВ. Цей зсув, при відсутності вхідних сигналів, забезпечує на виході приймача лог. «1», підтримуючи стан відсутності передачі. Для цього прямий вхід A необхідно підтягнути резистором до живлення, а інверсний вхід B до «землі» (рис. 4.13).

Для уникнення ефекту довгої лінії, коли сигнали мають здатність відбиватися від відкритих кінців лінії передачі та її відгалужень, необхідно на віддалених кінцях лінії між провідниками витої пари включати узгоджувальні резистори (термінатори). Номінал резисторів має бути рівним хвильовому опорі лінії передачі. Як правило він становить 120 Ом. Для коротких ліній (кілька десятків метрів) та при низьких швидкостях (менше 38 400 біт/сек) узгодження можна не робити взагалі.

Лінія зв'язку повинна представляти собою один кабель витої пари. До цього кабелю приєднуються всі прийомопередавачі RS-485. Відстань від лінії до драйверів RS-485 повинна бути як можна коротшою, оскільки довгі відгалуження вносять неузгодженість та викликають відбиття.

Це має і свій великий недолік: підвищене споживання струму від передавача, оскільки в лінію включається низькоомне навантаження, а також зменшене число підключених прийомопередавачів RS-485. Тому можна використати більш економне схемне рішення (рис. 4.14).

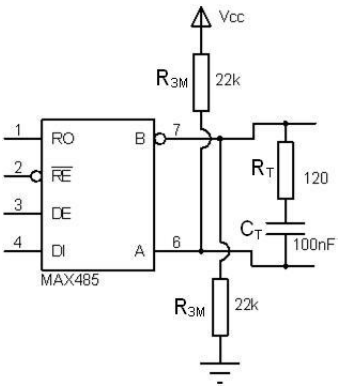


Рис. 4.14. Економне підключення прийомопередавача RS-485 до МК

Для нормального функціонування мережі RS-485 необхідно, щоб сигнальні «землі» пристроїв були з'єднані між собою. Рекомендується таке з'єднання виконувати за допомогою резистора 100 Ом для кожного драйвера (рис. 4.15) . Сигнальні «землі» та захисні заземлення мають також бути з'єднані між собою резистором 100 Ом.

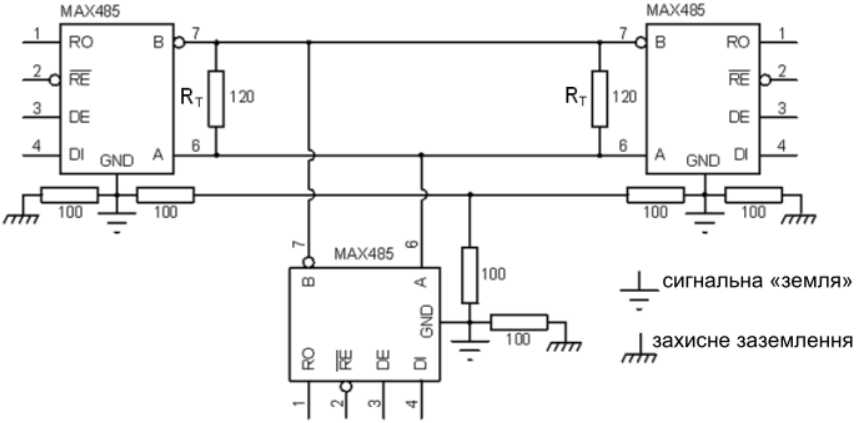


Рис. 4.15. Реалізація кіл сигнального заземлення

Згідно зі специфікацією RS-485, на лінії можуть знаходитися до 32 приймопередавачів, враховуючи узгоджуючі резистори (120 Ом). Це обумовлено вхідним опором приймача 12 КОм з боку лінії. Деякі мікросхеми драйверів мають підвищений вхідний опір, і тому дають можливість підключати до лінії більшу кількість пристроїв.

4.13. Мультипроцесорний режим модуля UART.

Цей режим дає можливість ефективно організувати обмін даними між основним (master) та рядом підлеглих (slave) МК по протоколу RS-485. При цьому, кожному підлеглому призначається своя унікальна адреса, згідно якої основний МК буде звертатися до нього.

Для цього необхідно налаштувати модулі UART усіх МК у мережі RS- 485 на 9-ти бітний обмін даними та усі підлеглі МК переключити у режим мультипроцесорного обміну, встановивши в «1» розряд MPCM у регістрі керування UCSRA.

Поведінка підлеглих МК з активованим мультипроцесорним режимом залежить від значення старшого розряду даних. Якщо він встановлений в «1», то це означає, що основний МК надсилає адресу , і тоді усі підлеглі приймають цей адресний байт даних та порівнюють зі значенням своєї адреси. Якщо для якогось з них адреси співпадають, тоді цей підлеглий МК відключає для себе мультипроцесорний режим та переходить у звичайний режим прийому даних. Решта МК і далі знаходяться в мультипроцесорному режимі. Якщо основний МК надсилає дані зі скинутим в «0» старшим розрядом, то ці дані може прийняти лише той МК, який працює у звичайному режимі, тобто для якого попередня адреса була співпала з його власною. Решта підлеглі МК ігнорують ці дані, і завдяки цьому зменшується процесорне навантаження на них. По завершенню обміну даними з «базою» вибраний підлеглий МК знову активує мультипроцесорний режим та переходить у режим фільтрації даних.

Приклад . Реалізувати обмін даними по протоколу RS- 485 між базою (master) та двома клієнтами (slave). Адреса кожного клієнта задається за допомогою 8 -ми клавішного перемикача типу «піаніно», що підключене до порту C. Кожен клієнт також має підключені до портів A та B 4 - клавішні перемикачі, які умовно позначені як Left та Right. База має 3 кнопки: перша вибирає клієнта A чи B, друга визначає які значення (Left чи Right) необхідно надіслати клієнтові для бази, а третя, згідно вибраних параметрів за допомогою попередніх кнопок, надсилає запит вибраному клієнту. Вибір клієнта, перемикачів та отриманого значення від клієнта відображається на індикації (рис. 4.16).

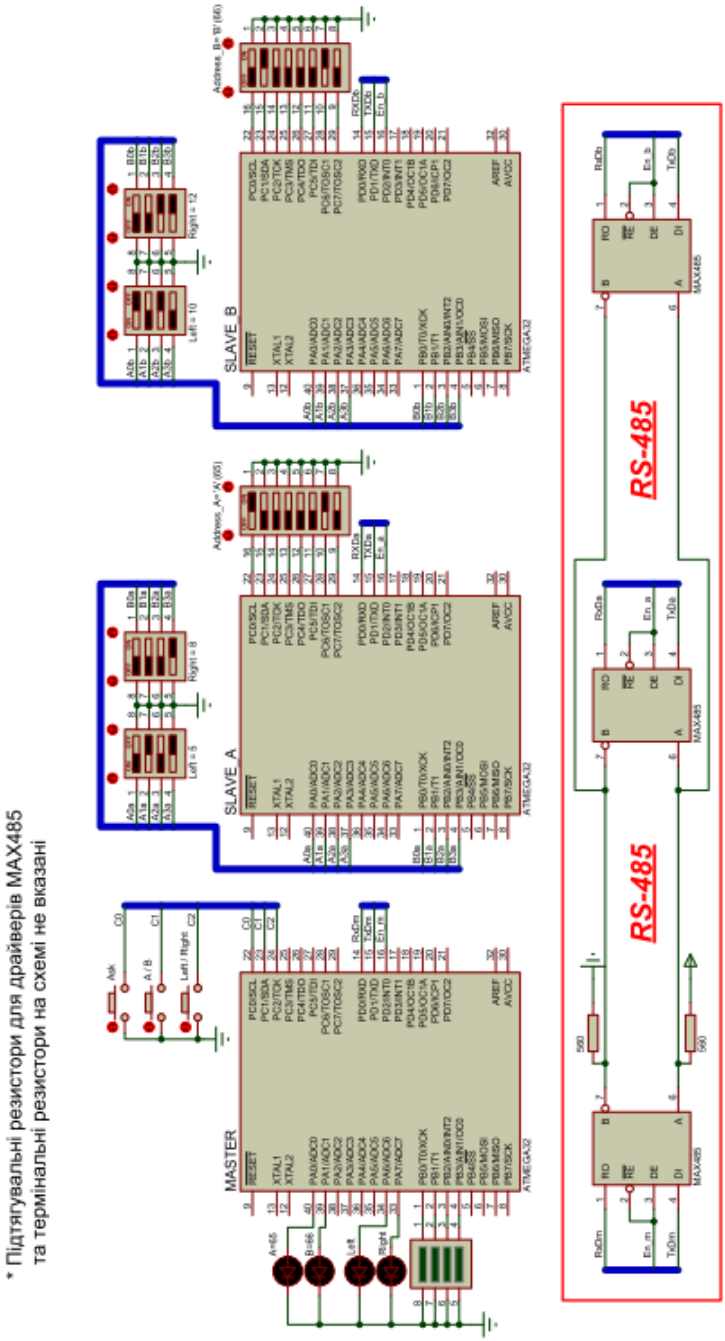


Рис. 4.16. Принципова схема для дослідження мультипроцесорного режиму модуля UART

Програмний код МК «master»

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define    F_CPU          8000000L
#define    BAUD           9600
#define    UBRRcalc       (F_CPU/(BAUD*16L)-1)
#define    BUF_SIZE       16
#define   BUF_MASK   (BUF_SIZE-1)
unsigned char BufferOUT[BUF_SIZE][2], StartBufOUT = 0, EndBufOUT = 0;
volatile unsigned char waitread = 0;          //
// ---------------  запис у буфер передачі ----------------------------------------------------------
void WriteBufOUT(unsigned char value, unsigned char bit8)
{
        BufferOUT[EndBufOUT][0] = value;
        BufferOUT[EndBufOUT++][1] = bit8;
        EndBufOUT &= BUF_MASK;
        cli();                                          // загальна заборона на переривання
        if( waitread == 0 )
                UCSRB |= 1<<UDRIE; // дозвіл на переривання спорожнення UDR
        sei();                                          // загальний дозвіл на переривання
}
 // ---------------       переривання при прийнятому байті даних  --------------------------------
ISR(USART_RXC_vect)
{
        PORTB = UDR;
        waitread = 0;
        // перевіряємо на спорожнення буфера передачі
        if( StartBufOUT != EndBufOUT )
                UCSRB |= 1<<UDRIE; // дозвіл на переривання спорожнення UDR
}
// ---------------  переривання при завершенні передачі ----------------------------------------
ISR(USART_TXC_vect )
{ PORTD &= ~(1<<PD2); }              //режим прийому
// ---------------  переривання при спорожненні буферу UDR ------------------------------
ISR(USART_UDRE_vect )
{
        PORTD |= 1<<PD2;                                //режим передачі
        if( BufferOUT[StartBufOUT][1] == 1 )   UCSRB |= 1<<TXB8;    //bit8=1
        else   { UCSRB &= ~(1<<TXB8); waitread = 1; }         //bit8=0; очік. прийому
        asm("nop");                                     // затримка в 1 такт
        UDR = BufferOUT[StartBufOUT++][0];              // читання з FIFO-буфера
        StartBufOUT &= BUF_MASK;
        // перевіряємо на спорожнення буфера передачі
        if( StartBufOUT == EndBufOUT || waitread == 1 )
                UCSRB &= ~(1<<UDRIE); // заборона на перер. спорожнення UDR
}
```

```
int main(void)
{
        Init();                 // ініціалізація периферії
        sei();                  // загальний дозвіл на переривання
        while (1)
        {
            if( bit_is_clear(PINC, 0) )         // надіслати запит
                {
                    if( bit_is_set(PINA, 0) )   WriteBufOUT('A', 1);        // адреса A
                    else                        WriteBufOUT('B', 1);        // адреса B
                    if( bit_is_set(PINA, 6) )   WriteBufOUT('L', 0);        // лівий
                    else                        WriteBufOUT('R', 0);        // правий
                    _delay_ms(500);             // затримка 0.5 сек.
                }
            if( bit_is_clear(PINC, 1) )         // вибір підлеглого
                {
                    if( bit_is_set(PINA, 0) )   {PORTA &= ~(1<<0); PORTA |= 1<<1;}
                    else                        {PORTA &= ~(1<<1); PORTA |= 1<<0;}
                    _delay_ms(500);             // затримка 0.5 сек.
                }

            if( bit_is_clear(PINC, 2) )         // з якого перемикача надіслати дані
                {
                    if( bit_is_set(PINA, 6) )   {PORTA &= ~(1<<6); PORTA |= 1<<7;}
                    else                        {PORTA &= ~(1<<7); PORTA |= 1<<6;}
                    _delay_ms(500);             // затримка 0.5 сек.
                }
    }
}

void Init()     // ініціалізація периферії
{
        DDRA=0xFF; PORTA=0x00;                      // на вихід - 0В
        DDRB=0xFF; PORTB=0x00;                      // на вихід - 0В
        DDRC=0x00; PORTC=0xFF;                      // на вхід - Rпідт.
        DDRD=0b11111110; PORTD=0b00000001;          // pin0 - Rпідт
        // ініціалізація USART (асинхр. режим)
        // швидкість (регістр UBRR)
        UBRRL = (unsigned char)(UBRRcalc);
        UBRRH = (unsigned char)(UBRRcalc>>8);
        //обнулення регістра UCSRA
        UCSRA = 0;
        // формат кадру 9n2 без перевірки парності
        UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0)|(1<<USBS);
        // дозвіл прийому-передачі+перериВ.прийому, заверш.перед.+ 9n
        UCSRB = (1<<UCSZ2)|(1<<RXEN)|(1<<TXEN)|(1<<RXCIE)|(1<<TXCIE);
}
```

Пояснення до програми. Для організації передачі даних через модуль UART використовуємо кільцевий 16-елементний буфер розмірністю 16×2. Один байт для 8 біт даних, другий для старшого розряду 9-бітної посилки, який буде вказувати чи посилка адресна (1), чи містить дані (0).

База надсилає дві посилки (адреса+дані) та очікує відповіді від вибраного клієнта. За допомогою змінної waitread організовується заборона на пересилання даних з кільцевого буфера до передавача UART до моменту відповіді від клієнта.

Оскільки на принциповій схемі не передбачено зовнішніх підтягуючих резисторів до високого рівня входів RxD модулів UART, то виводи RxD конфігуруються на внутрішній підтягуючий резистор.

Термінальні резистори 120 Ом також відсутні на принциповій схемі (через них пакет Proteus не коректно працював).

Чотири світлодіоди призначені для індикації вибраного клієнта (A чи B) та вказаного перемикача (Left чи Right), значення якого має бути надісланим від клієнта. На стовпцевий індикатор з 4-х сегментів, що підключений до порту B, виводиться значення, отримане від клієнта.

Програмний код МК «slave»

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define     F_CPU     8000000L
#define     BAUD      9600
#define     UBRRcalc  (F_CPU/(BAUD*16L)-1)

unsigned char Address;
// ---------- переривання при прийнятому байті даних ----------
ISR(USART_RXC_vect)
{
    {
        if( UDR == Address )
            UCSRA &= ~(1<<MPCM);          //відключ.мультипроц.режим
    }
    else
    {
        PORTD |= 1<<PD2;                  // режим передачі
        if( UDR == 'L' )      UDR = ~PINA;  // надсил. знач. Left
        else                  UDR = ~PINB;  // надсил. знач. Right
        UCSRA |= (1<<MPCM);               // включ.мультипроц.режим
    }
}
```

```
// ---------- переривання при завершенні передачі ----------------------------------
ISR(USART_TXC_vect )
{ PORTD &= ~(1<<PD2); }                    //режим прийому

int main(void)
{
        Init();                                    // ініціалізація периферії
        sei();                                     // загальний дозвіл на переривання
        while (1)
        {     }
}

void Init()       // ініціалізація периферії
{
        DDRA=0x00; PORTA=0xFF;                     // на вхід - Rпідт.
        DDRB=0x00; PORTB=0xFF;                     // на вхід - Rпідт.
        DDRC=0x00; PORTC=0xFF;                     // на вхід - Rпідт.
        DDRD=0b11111110; PORTD=0b00000001; // pin0 - Rпідт
        //ініціалізація USART (асинхр. режим) //швидкість
        (регістр UBRR)


        // мультипроцесорний режим
        UCSRA = (1<<MPCM);
        //формат кадру 9n2 без перевірки парності
        UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0)|(1<<USBS);
        //дозвіл прийому-передачі+перерив.прийому,заверш.перед.+9n
        UCSRB = (1<<UCSZ2)|(1<<RXEN)|(1<<TXEN)|(1<<RXCIE)|(1<<TXCIE);
        // читання адреси МК з піаніна
        Address = ~PINC;
}
```

Пояснення до програми. Значення, що зчитуються з перемикачів, інвертуються для отримання нормального вигляду, оскільки їхні виводи під'єднані до «землі». Адреса клієнта зчитується на етапі ініціалізації, і тому її нове значення буде задіяне лише після рестарту МК.