Лекція 11

Організація передачі даних через UART/USART.

Універсальний синхронний/асинхронний послідовний прийомопередавач (USART) забезпечує обмін даними МК AVR з зовнішніми пристроями по послідовному каналу в повнодуплексному режимі. При цьому передача даних може бути як асинхронна так і синхронна.

При **синхронному** послідовному вводі/виводі передача окремих бітів даних синхронізується за допомогою тактового сигналу, який передається одночасно з даними. Синхронна послідовна передача даних використовується, основним чином, на рівні друкованих плат, у тому числі, для обміну даними між різними інтегрованими блоками у складі схеми МК та різними периферійними схемами.

При **асинхронній** передачі даних синхронізація виконується у часі за допомогою стартових та стопових бітів, що визначають початок та кінець передачі слова даних. Асинхронна передача даних використовується для комунікації блоків, розділених у просторі та які мають певну автономність один від одного. Наприклад, між ПК та принтером, між пристроєм на базі МК та комп'ютером.

Модуль USART підтримує як синхронний, так і асинхронний режими роботи. Однак на практиці його найчастіше використовують саме в асинхронному режимі, а синхронний режим реалізують за допомогою модуля SPI. Тому в курсі лекцій ми розглядатимемо лише асинхронний режим, і надалі модуль називатимемо UART.

а. Формат передачі кадру даних UART. За своєю структурою він ідентичний інтерфейсу RS-232, з тією лиш відмінністю, що в інтерфейсі RS-232 логічні рівні формуються напругами від ±3 до ±12В, в модулі UART логічні рівні відповідають TTL-рівням (0 та 5 В).

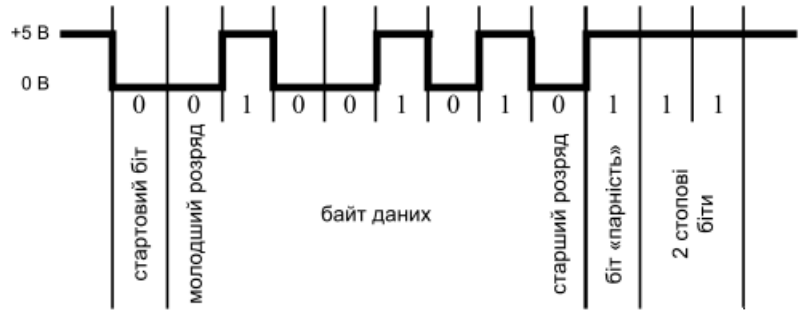

Рис. 3. Представлення числа 82 (01010010 2)
у кадрі 8n2 з перевіркою на парність

Початок кадру даних завжди фіксується низьким рівнем стартового біту (рис. 4.3). Після цього йде байт даних (5-9 бітів) з молодшими розрядами спереду. Якщо дозволена перевірка на парність, то далі йде біт парності, що доповнює байт даних «1» чи «0» так, щоб кількість «1» байту даних була парною (при опції «Парність») чи непарною (при опції «Непарність»). Цей простий засіб дає можливість виявляти непарну кількість спотворених бітів. Останніми передаються стопові (1 чи 2) біти, що представлені високим рівнем. Якщо на лінії передача даних відсутня, тоді на ній завжди присутній високий рівень.

Швидкість передачі вимірюється у бодах (baud), бітів за секунду (bps), і на рис. 3 кадр даних складається з 12 бодів.

Формат кадру задається відповідними бітами регістрів керування UCSRB та UCSRC.

Таблиця 5. Визначення розміру байту даних

	Кількість біт у байті даних				
	5 біт	6 біт	7 біт	8 біт	9 біт
UCSZ0	0	1	0	1	1
UCSZ1	0	0	1	1	1
UCSZ2	0	0	0	0	1

Таблиця 6. Керування контролем парності

	немає	парність	непарність
UPM0	0	0	1
UPM1	0	1	1

Вибір кількості стоп-бітів здійснюється за допомогою розряду USBS у регістрі керування UCSRC. Якщо цей розряд скинутий в «0», тоді передавач формує 1 стоп-біт у кінці посилки. Якщо ж встановлений в «1», тоді – 2 стоп-біти. Варто зазначити, що приймачем другий стоп-біт ігнорується, і відповідно, помилки кадрування виявляються лише для першого стоп-біта. Найбільш популярним є формат кадру 8n1 (1 старт, 8 біт даних, 1 стоп) без контролю парності.

б. Підключення UART. У МК AVR протокол передачі даних UART реалізований апаратно. На деяких моделях навіть є реалізовано 2 модулі UART. Приймач даних під'єднаний до виводу з надписом RxD, а передавач до виводу TxD. При з'єднанні між собою двох модулів UART для передачі даних, необхідно з'єднати навхрест між собою виводи модулів передачі та приймання, як на рис. 4.

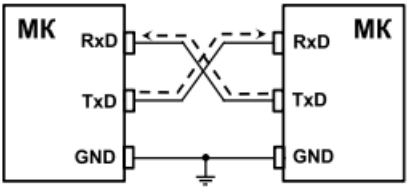

Рис. 4. З’єднання модулів UART для передачі даних

в. Апаратна частина UART. Модуль складається з 3-х основних частин (рис. 4.5): тактового генератора (контролера) швидкості передачі, блоку приймача та блоку передавача.

Блок передавача містить однорівневий буфер, зсувний регістр, схему формування біта парності та схему керування. Блок ж приймача містить схеми відновлення тактового сигналу та даних, схему контролю парності, дворівневий буфер, зсувний регістр та схему керування.

Буферні регістри приймача та передавача розміщуються за єдиним адресом простору вводу/виводу та позначаються як регістр даних UDR. У цьому регістрі зберігаються молодші 8 розрядів даних, що приймаються чи передаються. При читанні UDR виконується звертання до буферного регістра приймача, а при записі – до буферного регістра передавача.

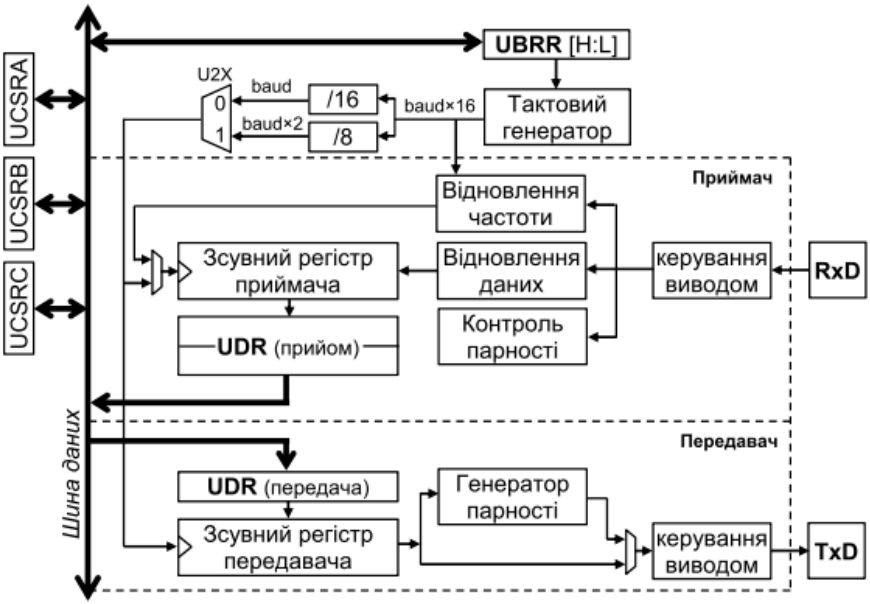

Рис. 5. Спрощена структурна схема модуля UART

У модулях USART буфер приймача дворівневий (FIFO-буфер). При будь-якому звертанні до регістра UDR цей буфер змінює свій стан. Тому необхідно спершу зчитати дані з цього регістра, а потім вже виконувати необхідні маніпуляції над ним.

Регістр контролера швидкості UBRR задає необхідний коефіцієнт поділу для системного тактового сигналу, після чого цей сигнал ще поступає на додаткові дільники, вибір яких здійснюється за допомогою додаткового біта U2X.

Схема відновлення тактового сигналу (у приймачі) призначена для синхронізації внутрішнього тактового сигналу, що формується контролером швидкості передачі, та пакетів з даними, що поступають на вивід RxD. Схема відновлення даних виконує зчитування та фільтрацію кожного розряду для отримуваного пакету.

г. Швидкість прийому/передачі. Швидкість обміну задається контролером швидкості передачі, що функціонує як подільник системного тактового сигналу з програмованим коефіцієнтом поділу, значення якого знаходиться у регістрі UBRR. Регістр UBRR є 12-розрядний та фізично розміщується у 2-х регістрах UBRRH та UBRRL.

В асинхронному режимі швидкість обміну визначається не лише значенням регістра UBRR, але і станом розряду U2X у регістрі керування UCSRA. Якщо цей біт встановлений в «1», то коефіцієнт поділу подільника зменшується у 2 рази, а швидкість, відповідно, подвоюється. Швидкість обміну в асинхронному режимі визначається за такими формулами:

$$\text{при U2X} = 0: \quad \mathbf{BAUD} = \frac{\text{XTAL}}{16(\text{UBBR} + 1)}; \quad \mathbf{UBRR} = \frac{\text{XTAL}}{16 \cdot \text{BAUD}} - 1$$

$$\text{при U2X} = 1: \quad \mathbf{BAUD} = \frac{\text{XTAL}}{8(\text{UBBR} + 1)}; \quad \mathbf{UBRR} = \frac{\text{XTAL}}{8 \cdot \text{BAUD}} - 1$$

Прийняті такі стандартні швидкості обміну даними: 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200, 230400 бод.

Для уникнення виникнення помилок передачі рекомендується використовувати стабілізований кварцовий тактовий генератор. Також має значення і величина частоти, на якій працює кварцовий кристал (табл. 5). На деяких частотах можна отримати нульову похибку при передачі даних відносно ряду стандартних швидкостей. Похибка передачі обчислюється за такою формулою:

$$\text{Error}[\%] = \left(\frac{\text{BAUD}_{\text{розрах.}}}{\text{BAUD}} - 1\right) \cdot 100\%.$$

Розрахуємо похибку для стандартної швидкості 9600 бод при частоті тактового генератора 8МГц.

$$\text{UBRR} = \frac{8 \cdot 10^6}{16 \cdot 9600} - 1 = 51{,}083 = 51;$$

$$\text{BAUD}_{\text{розрах.}} = \frac{8 \cdot 10^6}{16(51+1)} = 9615{,}38 \text{ Бод};$$

$$\text{Error}[\%] = \left(\frac{9615{,}38}{9600} - 1\right) \cdot 100\% = 0{,}16\%.$$

Рекомендується використовувати значення регістра UBRR, при яких отримана швидкість передачі відрізняється від необхідного значення менше, аніж на 0,5%.

Таблиця 7. Оцінка похибки передачі при швидкості 9600 бод

	1 МГц		3.6864 МГц		4 МГц		6 МГц	
	UBRR	**Error**	**UBRR**	**Erro r**	**UBRR**	**Error**	**UBRR**	**Error**
U2X=0	0x06	-7,0%	0x17	0%	0x19	0,16%	0x26	0,16%
U2X=1	0x0C	0,16%	0x2F	0%	0x33	0,16%	0x4D	0,16%

	7.3728 МГц		8 МГц		9.216 МГц		10 МГц	
	UBRR	**Erro r**	**UBRR**	**Error**	**UBRR**	**Erro r**	**UBRR**	**Error**
U2X=0	0x2F	0%	0x33	0,16%	0x3B	0%	0x40	0,16%
U2X=1	0x5F	0%	0x67	0,16%	0x77	0%	0x81	0,16%

	11.0592 МГц		12 МГц		14.7456 МГц		16 МГц	
	UBRR	**Erro r**	**UBRR**	**Error**	**UBRR**	**Erro r**	**UBRR**	**Error**
U2X=0	0x47	0%	0x4D	0,16%	0x5F	0%	0x67	0,16%
U2X=1	0x8F	0%	0x9B	0,16%	0xBF	0%	0xCF	0,16%

*****_похибка визначає відхилення швидкості передачі_

Варто звернути увагу, що на частотах тактового генератора рівних 3.6864, 7.3728, 11.0592 та 14.7456 МГц похибка передачі є рівною нулю для усієї лінійки стандартних швидкостей.

Зауваження . Для деякий моделей AVR адреса старшого байта UBRRH суміщена з адресою регістра керування UCSRC. Тому при записі необхідний регістр визначається станом старшого розряду записуваного значення. Якщо старший біт скинутий в «0», то значення записується у регістр UBRRH, а якщо встановлений в «1», тоді у регістр керування UCSRC.

д. Передача та прийом даних, переривання модуля UART.

Для активації прийому/передачі модуля UART необхідно надати дозволи на роботу передавача та приймача, встановивши відповідні біти TXEN та RXEN у регістрі керування UCSRB. Тоді відповідні виводи МК, позначені як TxD та RxD, підключаються до модуля UART та працюють на прийом і передачу, незалежно від налаштувань регістрів керування портом, до якого вони належать.

Для відправки байту даних необхідно записати його значення у регістр даних UDR. Після цього ці дані пересилаються із UDR у зсувний регістр передавача. Якщо в регістр UDR відправити одразу ще один байт даних , то ці дані будуть відправлені у зсувний регістр лише після того, як у зсувному регістрі буде відправлений останній біт з кадру. Отже, частота запису даних в UDR визначається швидкістю обміну даними модуля UART.

Прийом даних починається з моменту виявлення приймачем коректного старт-біту. Далі, кожен наступний біт кадру зчитується зі швидкістю, заданою для модуля UART, та розміщується у зсувному регістрі, аж поки не буде виявлений перший стоп-біт. Після цього вміст зсувного регістра пересилається у буфер приймача UDR, звідки прийняте значення має бути зчитаним.

Якщо формат кадру передбачає 9 біт даних, тоді перед записом в регістр UDR молодших 8 біт необхідно виставити у потрібне значення біт TXB8 (регістр UCSRB). Аналогічно і при прийомі даних, спершу необхідно прочитати значення біту RXB8 (регістр UCSRB), а потім вже читати значення молодших 8-ми бітів у регістрі UDR.

При прийомі даних також можемо виконати перевірку прапорів помилок (регістр UCSRA), які мають бути перевіреними ще перед читанням регістру даних UDR:

UPE – прапор помилки контролю парності, який виставляється при виявлені помилки парності у прийнятих даних.

DOR – прапор переповнення, який виставляється при виявленні нового старт-біта у зсувному регістрі, а буфер приймача у цей момент є заповнений (2 значення).

FE – прапор помилки кадрування, який виставляється при виявленні у прийнятому кадрі «0» на місці першого стоп-біта.

Для сповіщення про події: прийнято новий байт даних, завершено передачу даних, регістр даних UDR порожній передбачені відповідні прапорці RXC, TXC, UDRE (регістр UCSRA).

На основі цих прапорців також можуть бути згенеровані переривання для обробки цих подій. Дозвіл на переривання визначаються відповідними прапорами дозволів (регістр UCSRB):

RXCIE – дозвіл на переривання по завершенню прийому;

TXCIE – дозвіл на переривання по завершенню передачі;

UDRIE – дозвіл на переривання при спорожненні регістра UDR.

Переривання по завершенню передачі даних використовуються лише в окремих випадках. Наприклад, для переключення кінцевого пристрою у режим прийому по завершенню передачі даних в протоколі передачі даних RS-485.

Приклад. Реалізувати передачу даних через модуль UART із зовнішнім пристроєм. Згідно рис. 4.6 підключаються до МК дві кнопки для надсилання кодів символів ‘A’ та ‘B’, та три світлодіоди, які засвічуються при надсиланні на вхід модуля UART кодів символів ‘1’, ‘2’ та ‘3’. При прийомі інших значень світлодіоди гаснуть.

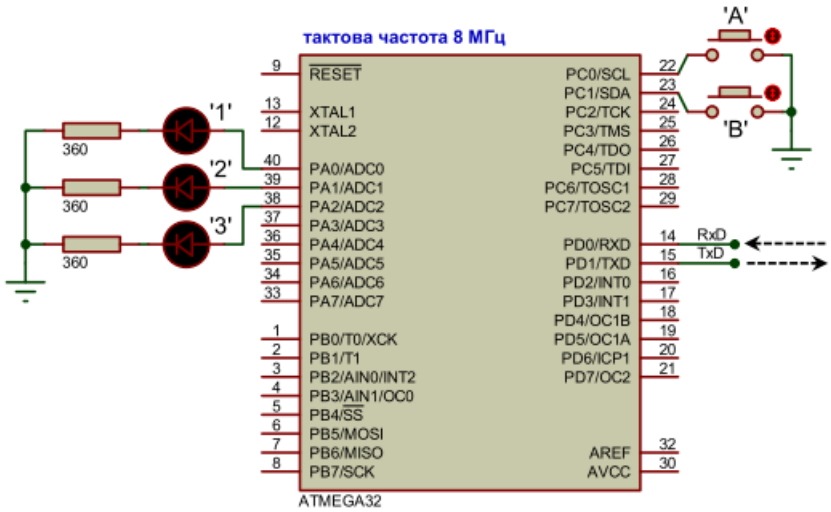

Рис. 6. Схема підключення МК для прийому/передачі даних по UART

У програмному коді прикладу буде задіяна лише підпрограма переривання при прийнятті байту даних. Макроси для підпрограм переривань при завершенні передачі та спорожненні буферу UDR наводяться для наочності, але не є реалізовані та не задіяні.

```
#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)

ISR(USART_RXC_vect)             // переривання при прийнятому байті даних
{
        switch(UDR)
        {
                case '1': PORTA = 1<<0; break;
                case '2': PORTA = 1<<1; break;
                case '3': PORTA = 1<<2; break;
                default: PORTA = 0;
        }
}

ISR(USART_TXC_vect )            // переривання при завершенні передачі
{}
ISR(USART_UDRE_vect )           // переривання при спорожненні буферу UDR
{}

int main(void)
{
        Init();                 // ініціалізація периферії
        sei();                  // загальний дозвіл на переривання
        while (1)
   {
                if(bit_is_clear( PINC, 0) )         // якщо натиснута кнопка A
                { UDR = 'A'; _delay_ms(500); }      // передача 'A' + затримка

                if(bit_is_clear( PINC, 1) )         // якщо натиснута кнопка B
                { UDR = 'B'; _delay_ms(500); }      // передача 'B' + затримка
   }
}

void Init()
{
        DDRA=0xFF; PORTA=0x00;                      // на вихід
        DDRC=0x00; PORTC=0xFF;                      // на вхід
        // ініціалізація USART (асинхр. режим)
        // швидкість (регістр UBRR)


        // формат кадру 8n2 без перевірки парності
        UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0)|(1<<USBS);

        // дозвіл прийому-передачі + перериав.прийому(регістр UCSRB)
        UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
}
```

10. Реалізація кільцевого буфера FIFO.

Обмін даними через модуль UART має суттєвий недолік: не можливо відправити на передачу одразу пакет байтів. Необхідно постійно очікувати спорожнення буфера UDR, а тоді вже відправляти черговий байт на передачу . Наприклад, для такого коду буде відправлено лише один-два байта:

```
for(int i=1; i<=10; i++)
    UDR =i;
```

Для спрощення організації обміну даними між різними пристроями чи процесами використовуються програмні кільцеві (циклічні) буфери FIFO (First Input First Output). Їх можна використовувати як для організації прийому/передачі модуля UART, так і, наприклад, для запису пакету даних в пам'ять EEPROM.

Кільцевий буфер є зручний тим, що дає можливість дуже просто виконувати заповнення буфера, перевірку наявності даних в буфері та вибірку даних з нього. При цьому не потрібно особливо турбуватися за границю буфера, переповнення пам'яті і т.п. Кільцевий буфер дозволяє коректно обмінюватися даними між обробником переривань та основною програмою.

Для реалізації кільцевого буфера необхідно в оперативній пам'яті виділити неперервний блок Buffer[BufSize] (рис. 4.7), розміром кратним степені 2 та два індекси StartB і EndB. Індекс StartB вказує на місце читання даних з буфера, а EndB на місце для запису у буфер. Розмір буфера вибирається кратним степені 2 для того, щоб було зручно маніпулювати цими індексами за допомогою інкремента та накладання маски за допомогою порозрядного AND. Наприклад, якщо розмір буфера дор. 16 (0b10000), тоді маска вибирається BufSize-1, тобто 15 (0b01111). Діапазон значень індексів для масиву з 16 елементів складає [0; 15]. Тому накладання цієї маски за допомогою порозрядного AND на значення індексів від 0 до 15 не змінює їх. А от якщо до максимального значення 15 додати 1, тобто вийде 16, та накласти нашу маску, то тоді значення обнулиться, та індекс буде вказувати на нульовий елемент масиву даних. При такому принципі роботи з індексом відбувається автоматичний перехід на початок буфера, як тільки ми досягнемо його кінця. Найоптимальніші 256-байтні буфери, так як у якості індексу можна використати цілий 1 байт, і тоді вже немає потреби накладати на них маску при інкременті їхніх значень.

Якщо буфер порожній, тоді індекси StartB та EndB мають однакове значення.

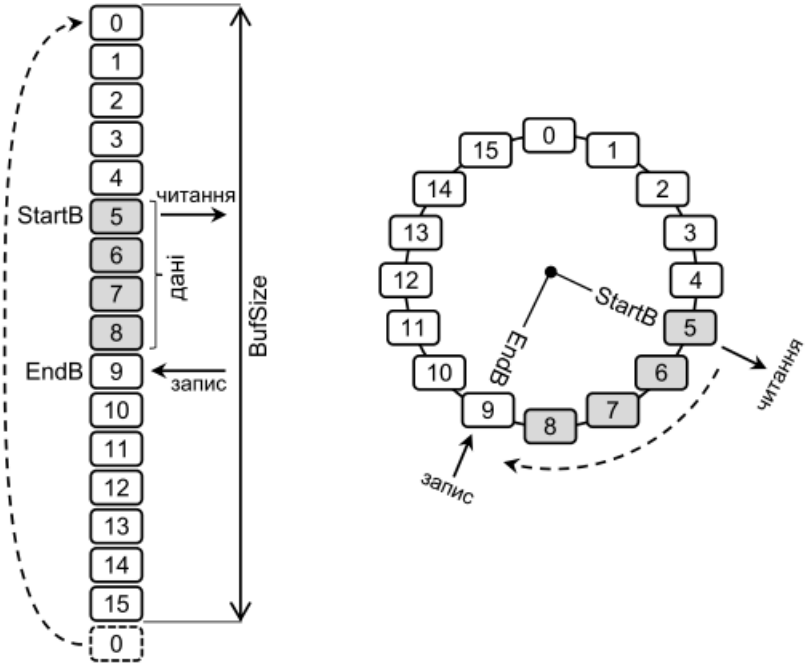

Рис. 7. Структура кільцевого буфера FIFO

Запис значення у буфер виконується за допомогою такого коду:

```
Buffer[EndB++] = value;
EndBuf &= BufMask;
```

Зчитування значення з буфера виконується аналогічно:

```
value = Buffer[StartB++];
StartB &= BufMask;
```

Про наявність даних у буфері свідчить відмінність значень індексів StartB та EndB:

```
while( StartB != EndB )    // поки буфер не порожній
{
// зчитуємо дані з буфера та опрацьовуємо їх
}
```

Для очистки буфера прирівнюємо між собою робочі індекси:

```
StartB = EndB;
```

Приклад. Реалізувати 2 кільцеві буфери для прийому та передачі даних через модуль UART. Згідно рис. 4.8, одна з кнопок має надсилати на вихід UART одразу 5 байтів даних. Періодично МК завантажується тривалою роботою, під час якої прийняті дані резервуються у вхідному буфері. По завершенню цієї роботи усі прийняті дані відсилаються назад, і МК знову переходить на виконання основної тривалої роботи. Два світлодіоди сигналізують про заповнення наших буферів, а окрема кнопка виконує скид помилки заповнення буферів.

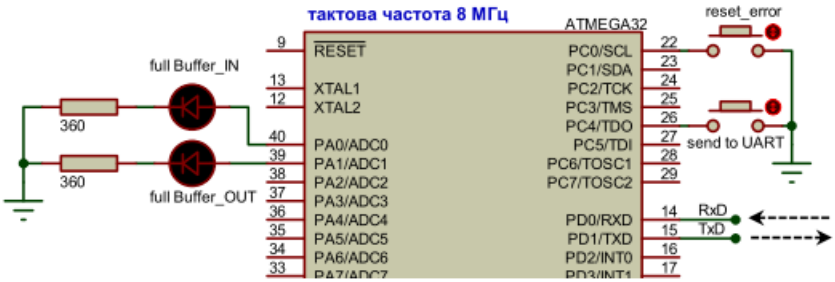

Рис. 8. Схема підключення МК для реалізації кільцевих буферів

```
#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    BufSize       16
#define    BufMask       (BufSize - 1)

unsigned char BufferIN[BufSize], StartBufIN=0, EndBufIN=0, BufINer=0; unsigned char BufferOUT[BufSize], StartBufOUT=0, EndBufOUT=0, BufOUTer=0;

//-------------------------------------------------------------------------------------------------------------
void WriteBufOUT(unsigned char value)    // запис у буфер передачі
{
        BufferOUT[EndBufOUT++] = value;
        EndBufOUT &= BufMask;
        // перевіряємо на переповнення
        буфера if( StartBufOUT == EndBufOUT )

        // даємо дозвіл на переривання спорожнення UDR
        UCSRB |= 1<<UDRIE;
}
```

```
//-----------------------------------------------------------------------------------------------------------
- unsigned char ReadBufIN(void) // читання з буферу прийому {

        unsigned char value = BufferIN[StartBufIN++];
        StartBufIN &= BufMask;
        return value;
}
//-----------------------------------------------------------------------------------------------------------
ISR(USART_RXC_vect)             // переривання при прийнятому байті даних
{// запис у буфер прийому
        BufferIN[EndBufIN++] =
        UDR; EndBufIN &= BufMask;
        // перевіряємо на переповнення
        буфера if( StartBufIN == EndBufIN )
                BufINer = 1;                    // помилка: буфер заповнений
}
//-----------------------------------------------------------------------------------------------------------
ISR(USART_UDRE_vect )           // переривання при спорожненні буферу UDR
{
        UDR = BufferOUT[StartBufOUT++];
        StartBufOUT &= BufMask;
        // перевіряємо на спорожнення буфера передачі
        if( StartBufOUT == EndBufOUT )
                UCSRB &= ~(1<<UDRIE); // заборона на перериВ. спорожн. UDR
}

int main(void)
{
        Init();                 // ініціалізація периферії
        sei();                  // загальний дозвіл на переривання
        while (1)
        {
                if( bit_is_clear(PINC, 0) ) // якщо натисн. кнопка «reset error» {
                    PORTA = 0; BufINer=0; BufOUTer=0; }

                if( bit_is_clear(PINC, 4) ) // якщо натисн. кнопка «send to UART»
                    for(int i=100; i<105; i++)

                                WriteBufOUT(i);  // запис у буфер передачі

                // тривала робота 2 сек.
                _delay_ms(2000);

                while( StartBufIN != EndBufIN )         // поки вх. буфер не
                    WriteBufOUT( ReadBufIN() );         порожній // BufOUT <- BufIN
                // індикація помилок
                if(BufINer) PORTA=1<<0;                 // заповнення буфера BufIN
                if(BufOUTer) PORTA=1<<1;                // заповнення буфера BufOUT
        }
}
```

```
void Init()      // ініціалізація периферії
{
        DDRA=0xFF; PORTA=0x00;                  // на вихід
        DDRC=0x00; PORTC=0xFF;                  // на вхід
        // ініціалізація USART (асинхр. режим)
        // швидкість (регістр UBRR)


        // формат кадру 8n2 без перевірки парності


        // дозвіл прийому-передачі + перериів.прийому(регістр UCSRB)
        UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
}
```

Зауваження: після запису даних у буфер передачі необхідно надати дозвіл на переривання для події спорожнення буфера UDR. Аналогічно, при завершенні передачі останнього байта з вихідного буфера необхідно заборонити це переривання.