Лабораторна робота №7. Моделювання модуля SPI

Тема: Моделювання модуля SPI

Мета: Користуючись пакетом PROTEUS дослідити моделювання модуля SPI

1. Порядок виконання роботи

- створити модель пристрою в пакеті Proteus 8.6
- розробити схему алгоритму роботи моделі та робочу програму
- створити hex-файл та підключити його до мікроконтролера
- запустити модель та виконати її дослідження згідно методичних вказівок
- зробити відповідні висновки.

2. Стислі теоретичні відомості

2.1 Опис моделі

Схему моделі дослідження послідовного периферійного інтерфейсу SPI наведено на рисунку 1. Ліворуч на схемі розташовано мікроконтролер, що працює в режимі «Master», а праворуч - мікроконтролер, який працює в режимі «Slave». В якості як веденого, так і ведучого мікроконтролера використано мікроконтролери ATMEGA8. Блок кнопок «Master Data To Send» призначено для задання байта даних для передачі від ведучого до веденого. Усі вісім кнопок цього блоку підключено до ліній порту D. Нижнє положення кнопки відповідає подачі на відповідну лінію порту D мікроконтролера напруги низького рівня, тобто логічного нуля, а верхнє положення кнопки - подачі на відповідну лінію порту напруги високого рівня, тобто логічної одиниці. Резистори R1...R8, що підтягують, призначені для формування напруги високого рівня на відповідній лінії порту мікроконтролера, коли контакти відповідної кнопки розімкнені.

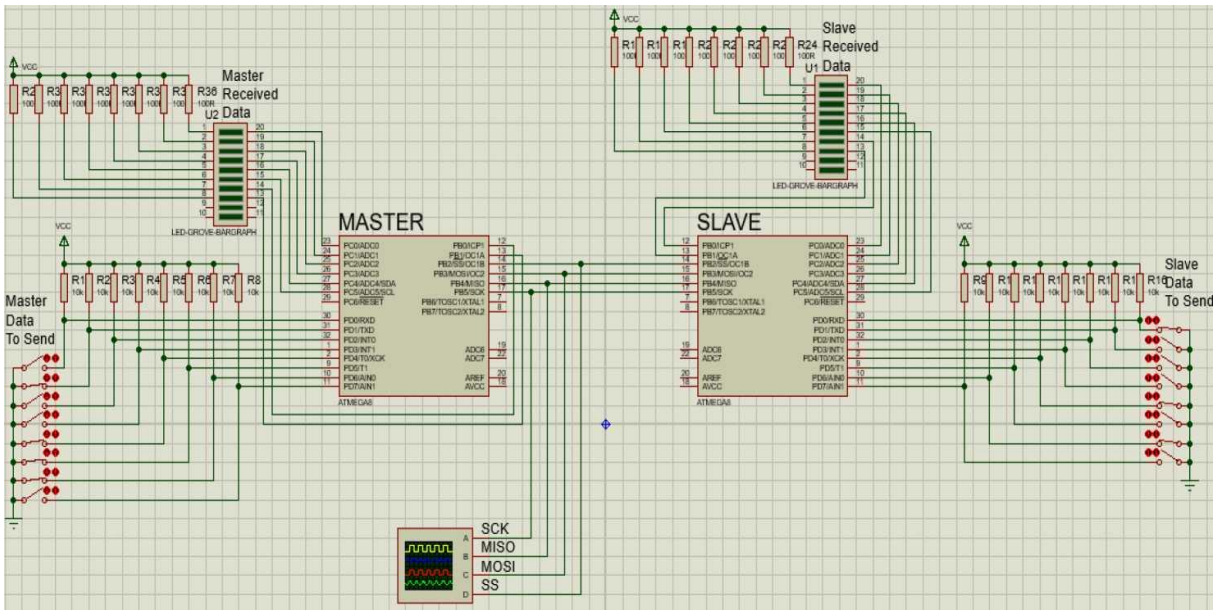

Рисунок 1 – Схема робочої моделі

Світлодіодна лінійка U2 «Master Received Data» призначена для відображення байта даних, прийнятого ведучим мікроконтролером від веденого. Верхній за схемою сегмент лінійки відображає молодший біт прийнятого байта даних, а нижній - старший. Світіння сегмента відповідає тому, що було прийнято логічний нуль, тобто натиснутій відповідній кнопці з боку передавача. Катоди сегментів світлодіодної лінійки, що відображають 6 молодших біт прийнятого байта підключено до ліній PC0...PC5 порту C мікроконтролера, а катоди сегментів, що відображають 2 старші біти - до ліній PB0, PB1 порту B. Аноди всіх сегментів світлодіодної лінійки через обмежуючі резистори R25...R32 підключено до лінії живлення +5ВДо ліній SCK, 77, MOSI та MISO ведучого мікроконтролера підключено відповідні лінії веденого мікроконтролера та входи відповідних каналів віртуального осцилографа.

Підключення веденого мікроконтролера виконано аналогічно. Кнопки «Slave Data To Send» призначені для вводу байта даних для передачі від веденого до ведучого. А світлодіодна лінійка U1 «Slave Received Data» відображає байт даних, прийнятий веденим від ведучого. Резистори R17...R24

обмежують струм через світлодіоди лінійки U1, а резистори R9...R16 призначені для формування напруги високого рівня на відповідних лініях порту D веденого мікроконтролера, коли контакти відповідної кнопки розімкнені.

2.2. Порядок моделювання

Для запуску моделювання натискаємо кнопку «Run simulation» « ▶ ». Після чого кнопками «Master Data To Send» задаємо байт для передачі ведучим МК веденому та одночасно спостерігаємо за відображенням його прийому на лінійці світлодіодів U1 «Slave Received Data». Також спостерігаємо за зміною сигналів, що відображаються у вікні віртуального осцилографа. Далі кнопками «Slave Data To Send» задаємо байт для передачі веденим МК ведучому та одночасно спостерігаємо за відображенням його прийому на лінійці світлодіодів U2 «Master Received Data» та за зміною сигналів, що відображаються у вікні віртуального осцилографа. Задамо, наприклад, байт «10001101» кнопками «Master Data To Send» для передачі від ведучого мікроконтролера («Master») до веденого («Slave»). Після чого спостерігаємо прийом даного байта веденим мікроконтролером та відображення його прийому на лінійці світлодіодів U1 «Slave Received Data». Відповідні фрагменти схеми моделі наведено на рисунку 2. Заданий байт «10001101» від ведучого веденому передається послідовно по лінії MOSI, що видно на відповідному каналі віртуального осцилографа (рисунок 3).

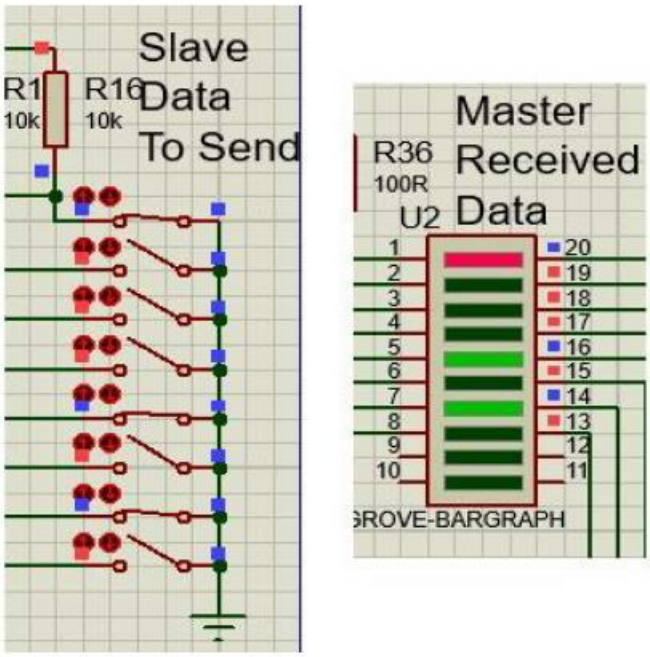

Рисунок 2 – Фрагменти схеми моделі для відображення передачі від ведучого до веденого

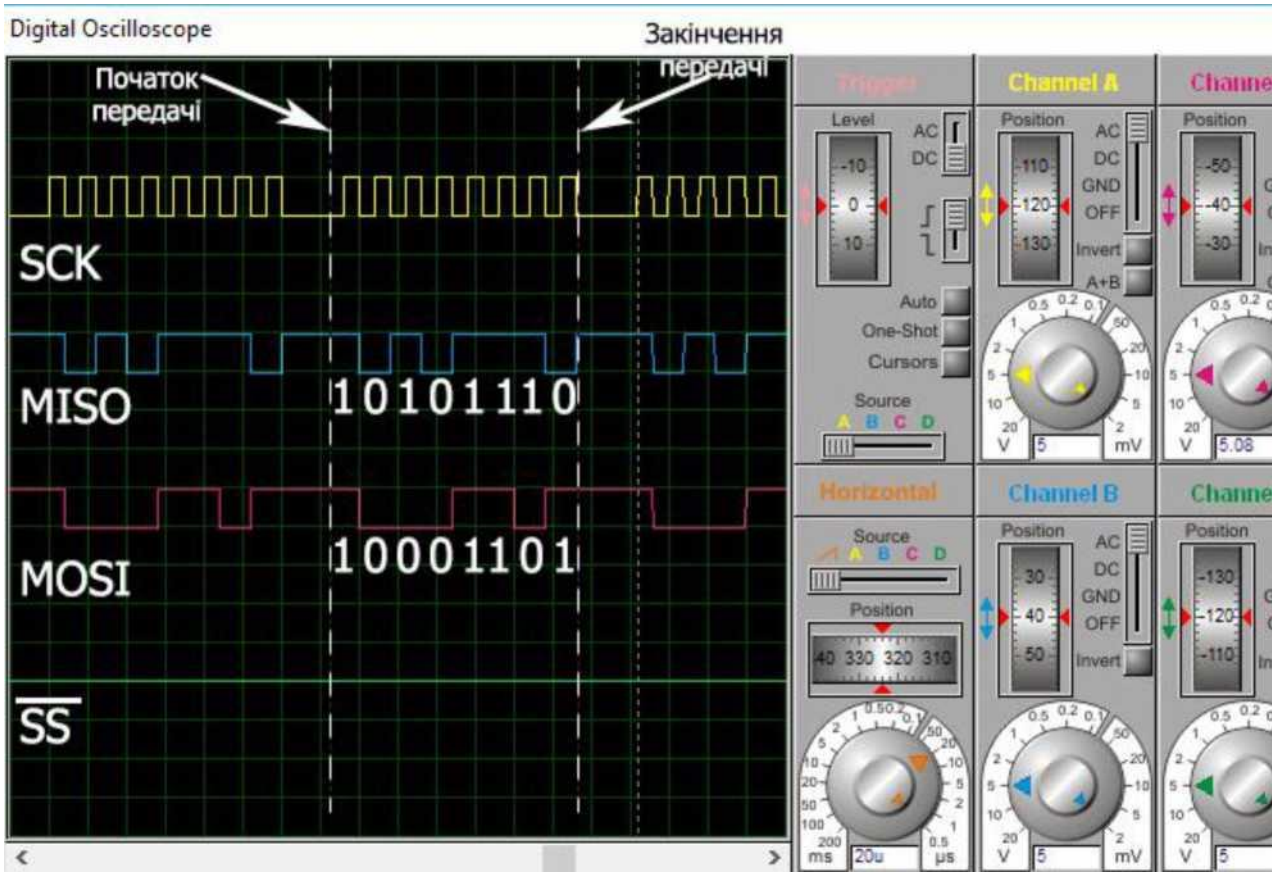

Рисунок 3 – Вікно віртуального осцилографа

Кнопками «Slave Data To Send» для передачі від веденого мікроконтролера до ведучого задамо, байт «10101110». Після чого спостерігаємо прийом даного байта ведучим мікроконтролером та відображення його прийому на лінійці світлодіодів U2 «Master Received Data». Відповідні фрагменти схеми моделі наведено на рисунку 4 Заданий байт «10101110» від веденого ведучому передається послідовно по лінії MISO, що видно на відповідному каналі віртуального осцилографа (рисунок 3).

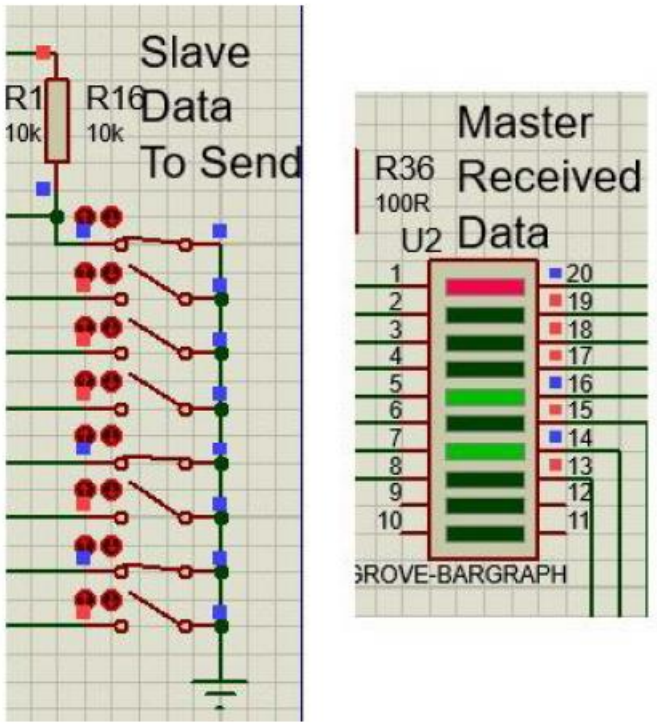

Рисунок 4 – Фрагменти схеми моделі для відображення передачі від веденого до ведучого

2.3. Схема алгоритму

На рисунках 5 та 6 наведено схеми алгоритмів роботи керуючої програми для ведучого та веденого мікроконтролерів відповідно.

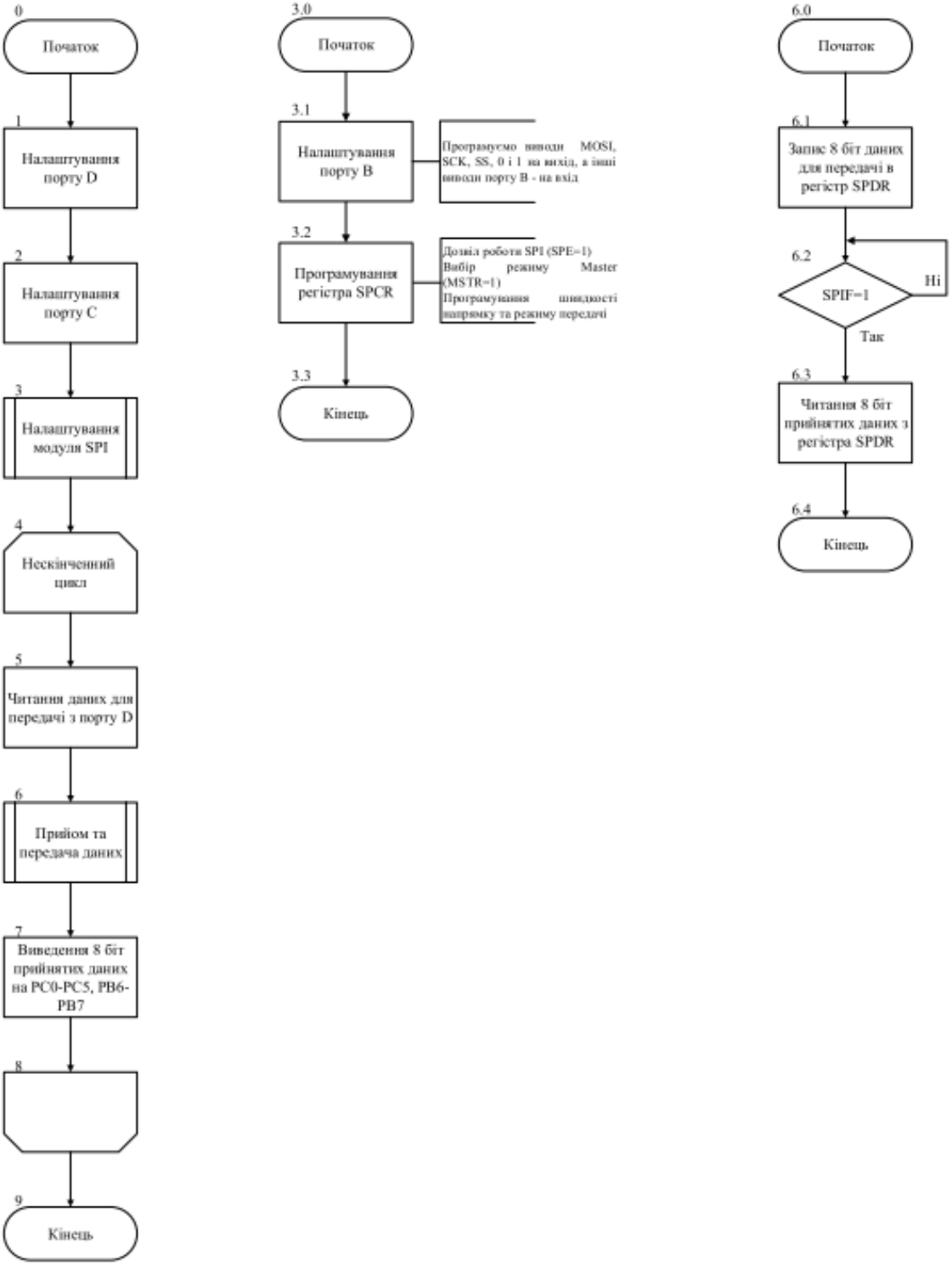

Рисунок 5 – Схема алгоритму керуючої програми ведучого МК

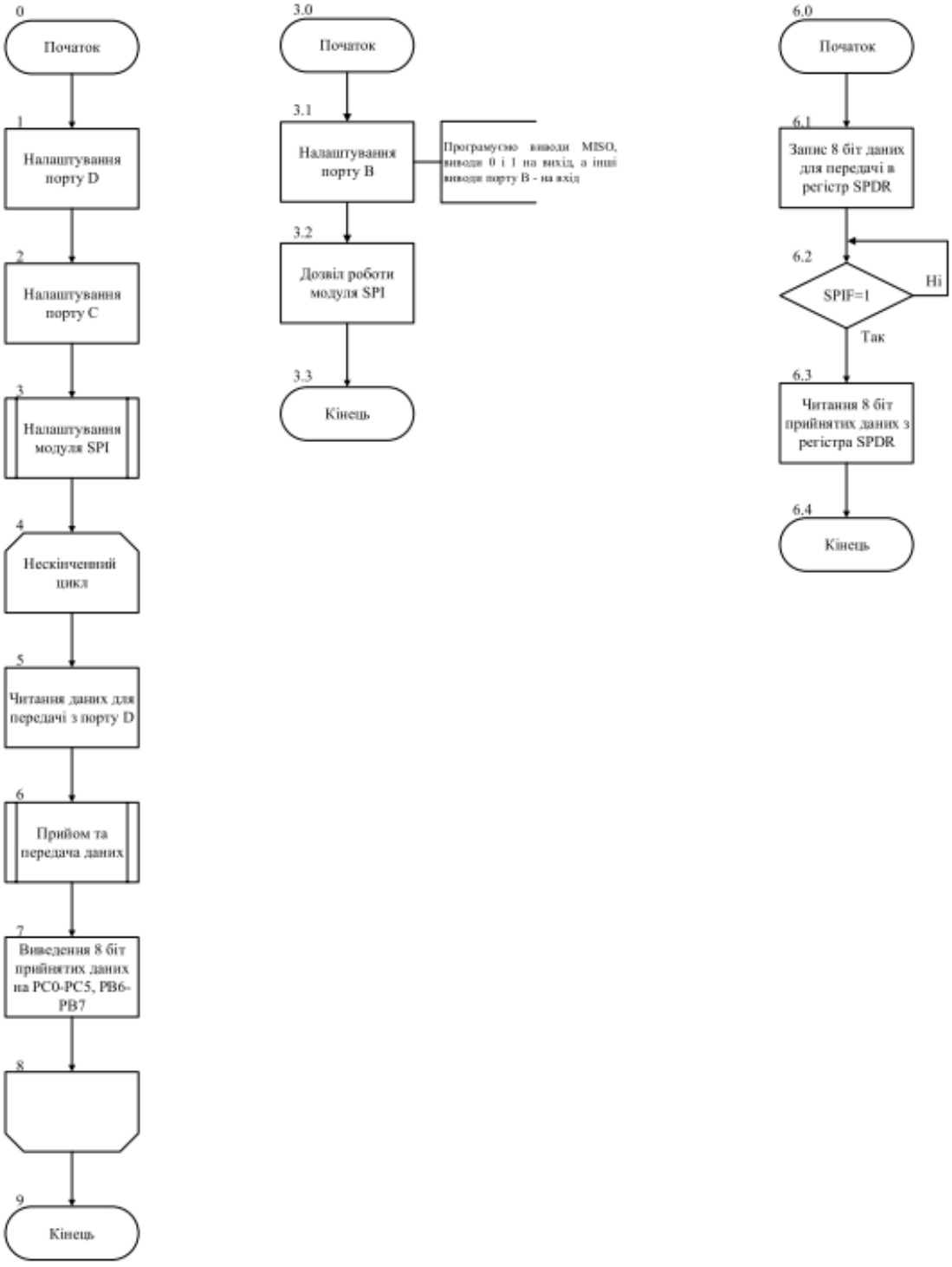

Рисунок 6 – Схема алгоритму керуючої програми веденого МК

2.4. Лістинг керуючої програми

Відповідно до схем алгоритмів, наведених на рисунках 5 та 6, розроблено керуючі програми для ведучого та веденого мікроконтролерів. Лістинги розроблених програм наведено нижче.

SPI Master:

```
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>

void SPI_MasterInit(void)
{
        DDRB = 0b00101111;                                  // 3.1 Програмуємо виводи MOSI та SCK, SS,
                                                                    виводи 0 і 1 на вихід, а інші
на вхід
        SPCR =  (1<<SPE)|(1<<MSTR)|(1<<SPR0);               // 3.2 Дозвіл роботи SPI, режим Master,
                                                                    встановлення швидкості передачі
(fck/16)
}
char SPI_MasterTransmit_Receive(char cData)
{
        char received_Data;
        /* Початок передачі */

        SPDR = cData;
                                                            //6.1 Запис 8біт даних для передачі в SPDR
        while(!(SPSR & (1<<SPIF)))
        ;                                                   // 6.2
                                                            //6.2 Чекаємозавершення передачі
        received_Data=SPDR;
SPDR                                                        //6.3 Читанняприйнятих даних з регістра

        return received_Data;
}

int main(void)
{
        DDRD=0x00;
        PORTD=0x00;                                         // 1 Програмуємо порт D на вхід.
                                                            // 1 Відключаємо внутрішні підтягуючі
схемі.                                                              резисторри, оскільки вони є на


        DDRC=0xff;
        PORTC=0xff;                                         //2 Програмуємо виводи порта C на вихід.
                                                            //2 Виводи порта C встановлюємо в 1.
        SPI_MasterInit();
        char data;                                          //3 Налаштування моуля SPI

     while (1)                                              // 4
     {
              data = PIND;                                  // 5 Читання даних для передачі з порту D

              char r_Data = SPI_MasterTransmit_Receive(data); // 6 Прийом/передача даних
```

```
                PORTC = r_Data;  // 7 Виведення 6 молодших біт прийнятих
                                                                    даних на PC0-PC5

                /* Виведення 2 старших біт прийнятих даних на PB6-PB7:*/ char      temp =
                PORTB;//                                7
                temp &=  0b11111100;                   //  7
                temp | = ((r_Data>>6) & 0b00000011);// 7
                PORTB =  temp;                         //  7
        }                                              //  8
}                                                      // 9
```

SPI Slave:

```
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>

void SPI_SlaveInit(void)
{
        DDRB = 0b00010011;                     // 3.1 Програмуємо виводи MISO, виводи 0 і 1
на
                                                    вихід, а інші - на вхід
        SPCR =  (1<<SPE);                      // 3.2 Дозвіл роботи модуля SPI
}
char SPI_SlaveTransmit_Receive(char cData)
{
        char received_Data;
        /* Початок передачі */

        SPDR = cData;
                                               // 6.1Запис 8 біт даних для передачі в SPDR
        while(!(SPSR & (1<<SPIF))) // 6.2 ;

        received_Data=SPDR;                    // 6.2 Чекаємо завершення передачі
SPDR
                                               // 6.3 Читання прийнятих даних з регістра
        return received_Data;
}

int main(void)
{
        DDRD=0x00;
        PORTD=0x00;
                                               // 1 Програмуємо порт О на вхід.
схемі.                                         // 1 Відключаємо внутрішні підтягуючі
                                                        резисторри, оскільки вони є на

        DDRC=0xff;
        PORTC=0xff;
                                               //2 Програмуємо виводи порта С на вихід.
        SPI_SlaveInit();                       //2 Виводи порта С встановлюємо в 1.
        char data;
                                               //3 Налаштування модуля SPI
        while (1)                              // 4
        {
                data = PIND;                   //  5 Читання даних для передачі з порту
                D

                char r_Data = SPI_SlaveTransmit_Receive(data); // 6 Прийом/передача даних

                PORTC = r_Data;                //  7 Виведення 6 молодших біт прийнятих
даних на                                                PC0-PC5

                /* Виведення 2 старших біт прийнятих даних на PB6-PB7:*/
                char temp = PORTB;             // 7

         temp &=  0b11111100;              //7
```

```
            temp | = ((r_Data>>6) & 0b00000011);// 7
            PORTB =  temp;                      //7
        }                                       //8
}                                              //9
```

3. Зміст звіту

Звіт по роботі повинен містити:

- схему моделі;
- схему алгоритму роботи моделі;
- робочу програму;
- формули за потребою.

Контрольні запитання

1. Який режим обміну використовується в модулі SPI: асинхронний чи синхронний? Відповідь пояснити.
2. На яку відстань і з якою швидкістю можна здійснювати обмін даними через інтерфейс SPI?
3. Поясніть структурну схему модуля SPI.
4. В яких режимах може працювати мікроконтролер при обміні даними інтерфейсом SPI?
5. Скільки та які виводи мікроконтролера використовує модуль SPI?
6. Як виконується перепризначення режиму роботи виводів модуля SPI?
7. Поясніть схему з'єднання двох мікроконтролерів інтерфейсом SPI.
8. Які пристрої знаходяться перед входами восьмирозрядних регістрів зсуву ведучого та веденого мікроконтролерів? Яку функцію вони виконують?
9. Який регістр призначено для керування модулем SPI?
10. Який регістр використовується для контролю стану модуля, а також для додаткового керування швидкістю обміну?
11. Як задати режим роботи мікроконтролера в режимі «MASTER»?
12. В який спосіб здійснюється передача даних в модулі SPI?
13. Як визначити кінець передачі байта?
14. За якої умови при передачі даних від ведучого до веденого можлива передача у зворотному напрямі?
15. Що означає те, що у модулі реалізовано одинарну буферизацію при передачі та подвійну при прийомі?
16. Поясніть структура SPI-мережі мікроконтролерів.
17. Як обирається потрібний ведений мікроконтролер в SPI-мережі?

18. Поясніть вплив сигналу $\overline{SS}$ на початок обміну даними в різних режимах роботи модуля.
19. Чи можна до інтерфейсу підключати декілька периферійних пристроїв?
20. Назвіть кількість режимів передачі даних та принцип їх роботи.
21. Що є джерелом тактового сигналу при роботі модуля? Як формується тактовий сигнал?
22. При яких частотах гарантується функціонування мікроконтролера в режимі «Slave»?
23. З яких етапів складається обмін в режимі SPI?
24. Поясніть схему включення мікроконтролерів в режимі програмування послідовним каналом при використанні інтерфейсу SPI.