Міністерство освіти і науки України
Одеська національна академія зв'язку О.С. Попова
Кафедра комп'ютерно-інтегрованих технологічних процесів та виробництв

Методичні вказівки
до лабораторних робіт № 6…10
з дисципліни «Обчислювальна техніка та мікропроцесори»

Одеса 2017

УДК 004.2+004.3+004.4 План НМВ 2017 р.
ББК 3973.26-018я73

Рецензент: доц. каф. ІТ Ларін Д.Г.
С.Ю. Манаков, С.П. Главацький, І.С. Кушнір

Методичні вказівки до лабораторних робіт № 6-10 з дисципліни «Обчислювальна техніка та мікропроцесори»

Розглядаються питання програмування мікроконтролерів AVR ATmega16 фірми Atmel, робота з таймерами/лічильниками, модулем АЦП та інтерфейсами обміну даними з зовнішніми пристроями.

Ухвалено
на засіданні кафедри КІТ П і В та
рекомендовано до друку.
Протокол №4 від 10.11.2016 р.

Затверджено
методичною радою
академії зв'язку.
Протокол №5 від 28.02.2017 р.

Зміст

Лабораторна робота № 6. Таймери-лічильники мікроконтролера ATmega16 3

Лабораторна робота № 7. Модуль АЦП МК ATmega16 11

Лабораторна робота № 8. Універсальний синхронний/асинхронний приймач/передавач USART 19

Лабораторна робота № 9. Інтерфейс SPI МК ATMEGA16. 28

Лабораторна робота № 10. Інтерфейс TWI (I2C) МК ATmega16 38

Лабораторна робота № 6

ТАЙМЕРИ-ЛІЧИЛЬНИКИ МІКРОКОНТРОЛЕРА ATmega16

1 Мета роботи

Вивчення функцій таймерів-лічильників МК ATmega16. Набуття навичок роботи з таймерами в заданому режимі і використання переривань від таймерів.

2 Основні положення

2.1 Тактування мікроконтролера ATmega16

Максимальна тактова частота роботи МК ATmega16 становить 16 МГц. За даної частоти він здатний виконувати 16 мільйонів інструкцій за секунду, тобто його продуктивність досягає 16 MIPS.

Джерелом тактових імпульсів можуть бути:

- зовнішній кварцовий резонатор;
- зовнішній генератор імпульсів;
- вбудований генератор імпульсів.

МК ATmega16 містить 4 внутрішніх джерела тактових імпульсів: 1, 2, 4 і 8 МГц.

Як зовнішні джерела тактового сигналу використовують кварцові резонатори або генератори частотою до 16 МГц (рис. 2.1).

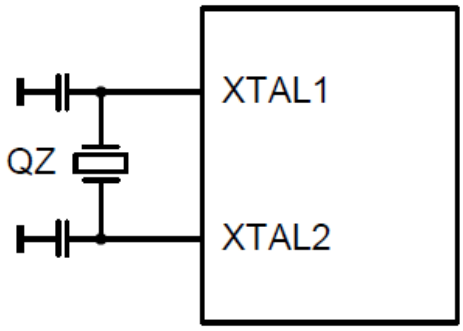

Рисунок 2.1 – Підключення зовнішнього джерела тактового сигналу

Вибір джерела тактового сигналу і частоти досягаються за допомогою настроювальних бітів (fuse bits) і регістра калібрування частоти OSCCAL. Чим вище тактова частота МК, тим швидше працює МК, проте зростає споживання енергії та тепловиділення.

2.2 Таймери-лічильники МК ATmega16

Таймер-лічильник є одним із найбільш часто використовуваних ресурсів мікроконтролерів AVR. Його основне призначення – відраховувати задані часові інтервали. Крім того, таймери-лічильники можуть виконувати низку додаткових функцій: формування сигналів ШІМ (широтно-імпульсна модуляція) заданої щілинності, підрахунок тривалості і кількості вхідних імпульсів.

МК ATmega16 оснащений трьома таймерами-лічильниками: два 8-розрядних таймера-лічильника T0 і T2, і один 16-розрядний T1.

Таймер-лічильник T0 використовує два виводи МК ATmega16. Вивід T0 (PB0) – вхід зовнішнього тактового сигналу – може застосовуватися для підрахунку імпульсів. Вивід OC0 (PB3) – вихід схеми порівняння таймера-лічильника. На цьому виводі за допомогою таймера можна формувати меандр або сигнал ШІМ. Також він може змінювати рівень сигналу при спрацьовуванні схеми порівняння. Висновки T0 і OC0 задіюються тільки при відповідних настройках таймера і в звичайному стані є висновками загального призначення.

Далі розглянемо регістри таймера-лічильника T0, структура якого показана на рис. 2.2.

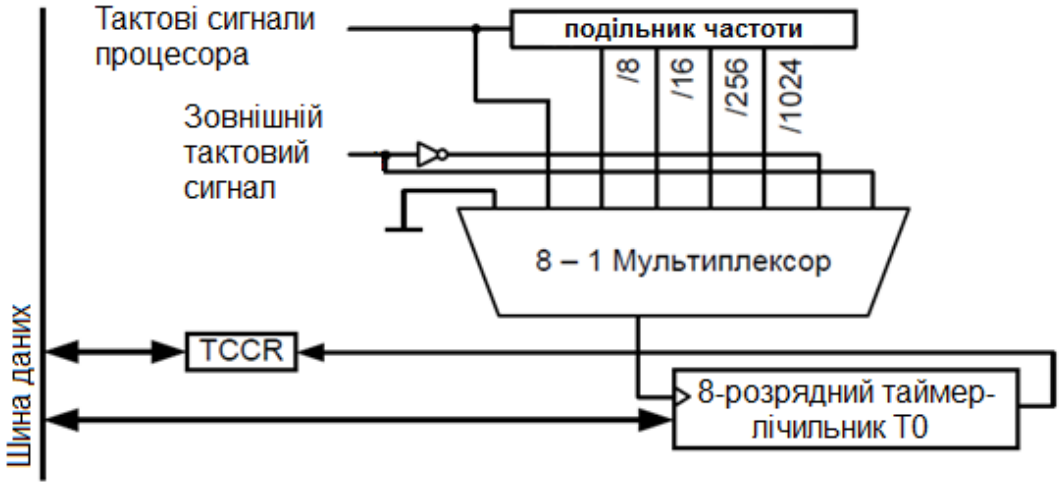

Рисунок 2.2 – Структура таймера-лічильника T0

2.2.1 Регістри таймера-лічильника T0

Таймер T0 має у своєму складі три регістри:

- лічильний регістр TCNT0 (рис. 2.3);
- регістр порівняння OCR0 (рис. 2.4);
- конфігураційний регістр TCCR0 (рис. 2.5).

Крім того, існують три регістри, що відносяться до всіх трьох таймерів ATmega16:

- конфігураційний регістр TIMSK;

- статусний регістр TIFR;
- регістр спеціальних функцій SFIOR.

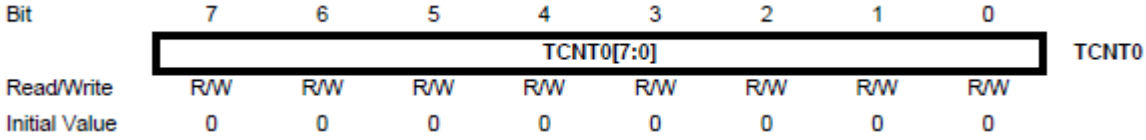

Рисунок 2.3 – Лічильний регістр TCNT0

Під час роботи таймера за кожним імпульсом тактового сигналу значення TCNT0 змінюється на одиницю. Залежно від режиму роботи таймера, вміст лічильного регістра може як збільшуватися, так і зменшуватися.

Регістр TCNT0 можна як читати, так і записувати. Останнє використовується коли потрібно задати його початкове значення. Коли таймер працює, змінювати його вміст TCNT0 не рекомендується, тому що це блокує схему порівняння на один наступний такт.

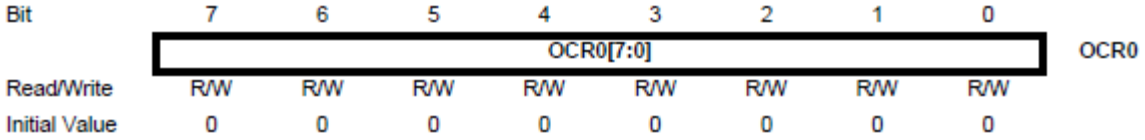

Рисунок 2.4 – Регістр порівняння OCR0

Значення регістра порівняння OCR0 постійно порівнюється з лічильним регістром TCNT0, і, в разі збігу, таймер може викликати переривання, змінювати стан виведення OC0 і т.ін. в залежності від режиму роботи. Значення OCR0 можна як читати, так і записувати.

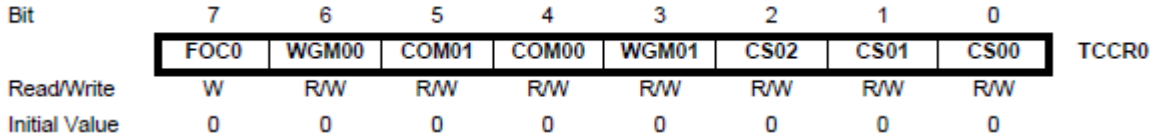

Рисунок 2.5 – Конфігураційний регістр TCCR0

Конфігураційний регістр TCCR0 таймера-лічильника T0 визначає джерело тактування таймера, коефіцієнт подільника частоти, режим роботи таймера-лічильника T0 і поведінку виводу OC0.

Біти CS02, CS01, CS00 (Clock Select) визначають джерело тактової частоти для таймера T0 і задають коефіцієнт подільника (табл. 2.1).

Таким чином, таймер-лічильник може бути зупинений, може тактуватися від внутрішньої частоти і також від сигналу на виводі T0.

Біти WGM10, WGM00 (Wave Generator Mode) визначають режим роботи таймера-збігу T0 (табл. 2.2):

- нормальний режим (Normal);
- скидання при співпадання (CTC);
- два режими ШІМ (FastPWM і Phase Correct PWM).

Таблиця 2.1 – Вибір тактової частоти для таймера T0

CS02	CS01	CS00	Опис
0	0	0	Джерела тактування немає. Таймер зупинений
0	0	1	Тактова частота МК
0	1	0	Тактова частота МК/8
0	1	1	Тактова частота МК/64
1	0	0	Тактова частота МК/256
1	0	1	Тактова частота МК/1024
1	1	0	Зовнішнє джерело на виході T0. Спрацювання на заднім фронті
1	1	1	Зовнішнє джерело на виході T0. Спрацювання на переднім фронті

Таблиця 2.2 – Вибір режиму таймера T0

WGM01	WGM00	Режим роботи таймера/лічильника
0	0	Normal
0	1	Phase Correct PWM
1	0	CTC
1	1	Fast PWM

Біти COM01, COM00 (Compare Match Output Mode) визначають поведінку вивода OC0. Якщо хоча б один із цих бітів установлений в 1, то вивід OC0 перестає функціонувати як звичайний вивід загального призначення і підключається до схеми порівняння таймера-лічильника T0. При цьому він також повинен бути налаштований в режим виходу.

Поведінка виведення OC0 залежить від режиму роботи таймера-лічильника T0. У режимах Normal і CTC вивід OC0 поводиться однаково, а в режимах ШІМ його поведінка відрізняється.

Біт FOC0 (Force Output Compare). Цей біт призначений для примусової зміни стана виводу OC0. Він працює тільки для режимів Normal і CTC. При установці біта FOC0 в 1 стан виводу змінюється відповідно до значень бітів COM01, COM00. FOC0 біт не викликає переривання і не скидає таймер в режимі CTC.

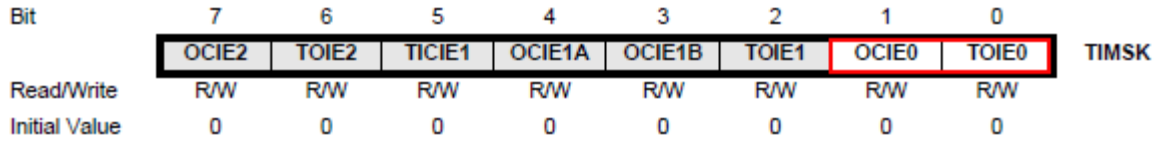

Рисунок 2.6 – Регістр маскування переривань таймерів, TIMSK

TIMSK (Timer/Counter Interrupt Mask Register) – загальний регістр для всіх трьох таймерів ATmega16 містить біти дозволу переривань (рис. 2.6). Таймер T0 може викликати переривання при переповненні лічильного регістра TCNT0 і при збігу лічильного регістра з регістром порівняння OCR0. Відповідно для таймера T0 в регістрі TIMSK зарезервовані два біти – TOIE0 і OCIE0. Решта бітів відноситься до інших таймерів.

TOIE0 – при установці в 0 забороняє переривання за переповненням, 1 – дозволяє.

OCIE0 – при установці в 0 забороняє переривання за випадковим збігом, 1 – дозволяє.

Природно переривання будуть викликатися тільки якщо установлений біт глобального дозволу переривань – біт I регістра SREG.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 2.7 – Регістр прапорців переривань, TIFR

TIFR (Timer/Counter0 Interrupt Flag Register) – загальний регістр для всіх трьох таймерів-лічильників (рис. 2.7). Містить статусні прапорці, які установлюються при виникненні переривань від таймерів. Для таймера T0 це є *переповнення* лічильного регістра TCNT0 або *збіг* його з регістром порівняння OCR0.

Якщо в ці моменти в регістрі TIMSK дозволені переривання і встановлений біт I, мікроконтролер викличе оброблювач відповідного переривання.

Прапорці автоматично скидаються при запуску оброблювача переривання. Також це можна зробити програмно, записавши 1 до відповідного прапорця.

TOV0 – установлюється в 1 при переповненні лічильного регістра.

OCF0 – установлюється в 1 при збігу лічильного регістра з регістром порівняння OCR0.

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 2.8 – Регістр SFIOR

Один з розрядів регістра SFIOR (Special Function IO Register) скидає 10-розрядний двійковий лічильник, який ділить вхідну частоту для таймерів T0 і T1. Скидання подільника частоти здійснюється за умови установлення біта PSR10 (Prescaler Reset Timer/Counter1 і Timer/Counter0) в 1.

2.3 Приклад програми

Наступна програма реалізує схему «рухомі вогні» на 8-ми світлодіодах, що підключені до порту B (рис. 2.9).

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR (TIMER0_OVF_vect)
```

```
{
     if (PORTB != 128) PORTB<<1;
     else PORTB = 0x01;
}
int main(void)
{
     DDRB |= 0xff;
     PORTB |= 0x01;
     TCCR0 |= (1<<CS00);
     TCCR0 &= ~(1<<CS01);
     TCCR0 |= (1<<CS02);
     TCCR0 &= ~(1<<WGM00);
     TCCR0 &= ~(1<<WGM01);
     TIMSK |= (1<<TOIE0);
     OCR0 = 255;
     asm("sei");
     while(1)
    {
        asm("nop");
    }
}
```

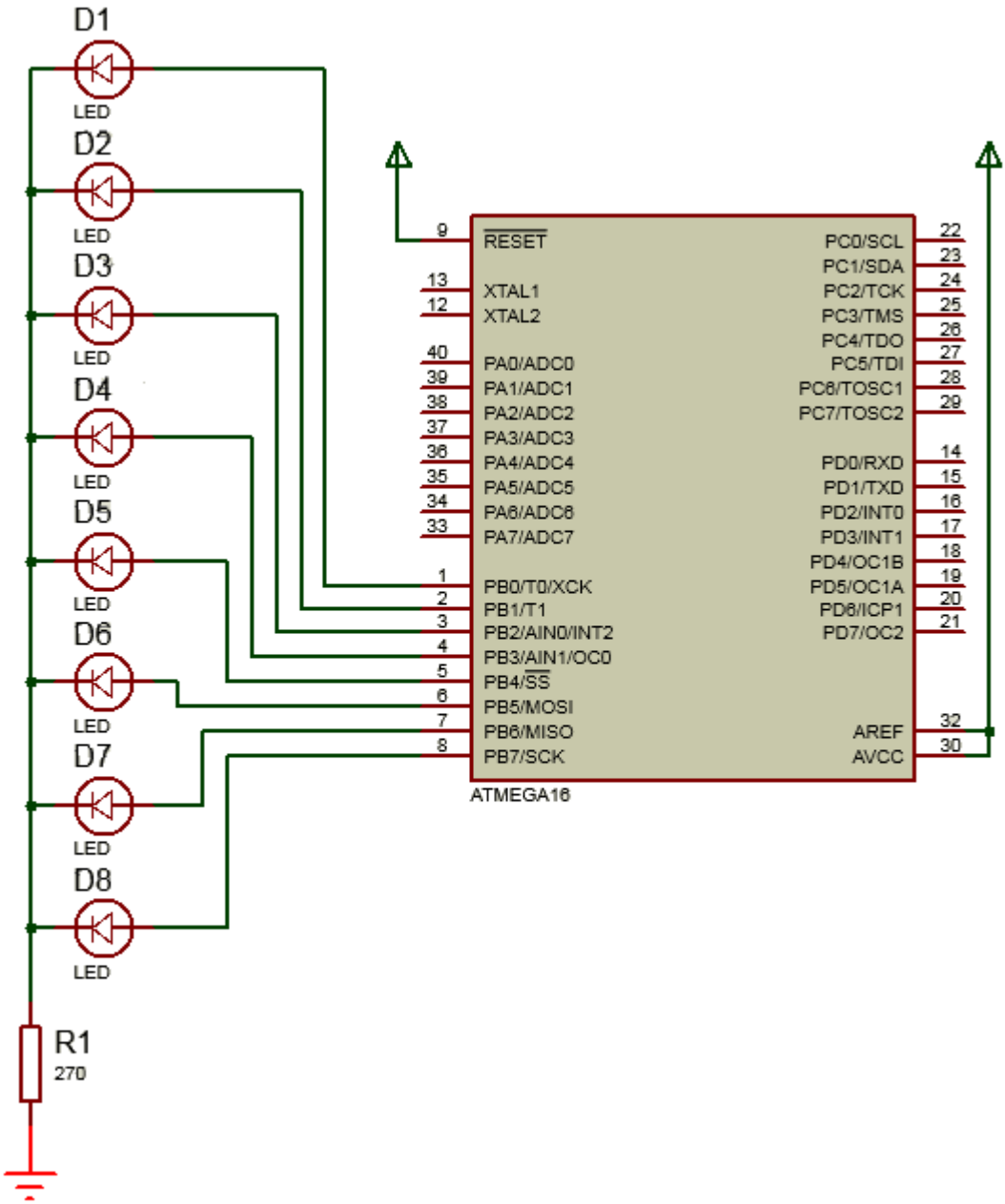

Рисунок 2.9 – Схема «рухомі вогні»

3 Контрольні питання

1. Які джерела тактової частоти може використовувати МК сімейства AVR?
2. Які таймери-лічильники є в МК ATmega16?
3. Регістри таймерів-лічильників МК AVR.
4. Режим Normal. Яке переривання передбачене.
5. Режим CTC, та його переривання.

4 Домашнє завдання

1. Вивчити розд. 2 (Основні положення).
2. Підготувати протокол лабораторної роботи.
3. Відповісти на контрольні питання.

5 Лабораторне завдання

1. Скласти схему рис. 2.6 у середовищі Proteus.
2. Створити проект в AVR Studio для програми з п. 2.3.
3. Скомпілювати програму й отримати файл з розширенням .hex в папці \Debug проекту.
4. Установити отриманий файл в якості прошивки мікроконтролера на схемі в Proteus і перевірити працездатність схеми.
5. Розставити коментарі в програмі. Визначити обраний режим таймера та значення всіх конфігураційних бітів.
6. Обчислити частоту перемикання світлодіодів у схемі.

6 Оформлення протоколу

Протокол виконання лабораторної роботи включає: тему, мету роботи, виконання домашнього і лабораторного завдання, висновки.

7 Список рекомендованої літератури

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL – [5-е изд.] / Евстифеев А.В. – М.: Издательский дом «Додэка-XXI», 2008. – 560 с.

Лабораторна робота № 7

МОДУЛЬ АЦП МК ATmega16

1 Мета роботи

Вивчення характеристик і режимів роботи вбудованого АЦП МК ATmega16. Набуття навичок програмування АЦП.

2 Основні положення

2.1 Вбудований модуль АЦП ATmega16

Мікроконтролер ATmega16 містить у своєму складі 10-розрядний, 8-канальний АЦП (аналого-цифровий перетворювач). 8 входів АЦП (рис. 2.1) підключені до єдиного модуля за допомогою аналогового мультиплексора, тобто АЦП є спроможний перетворювати тільки один сигнал в поточний момент часу.

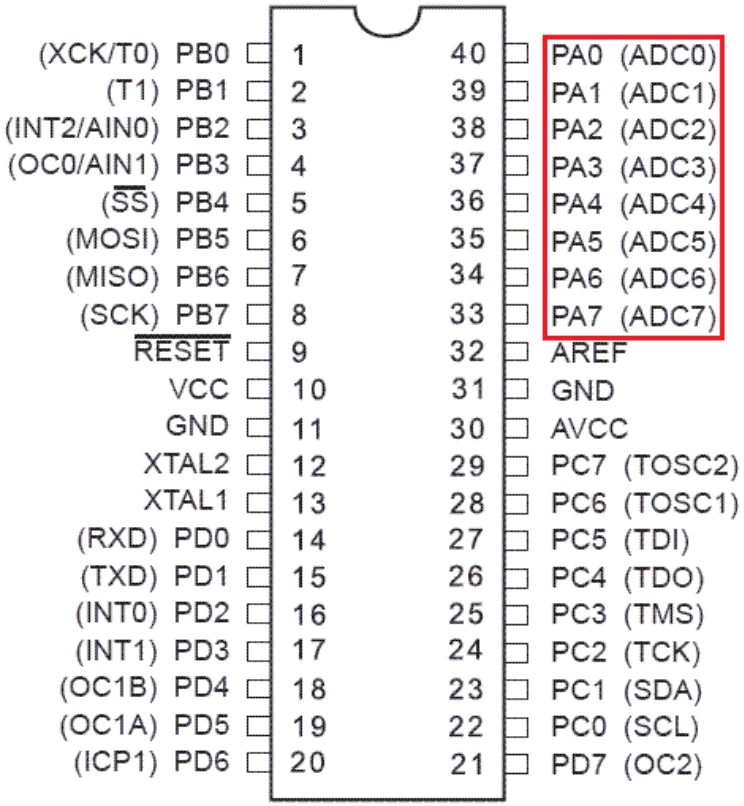

Рисунок 2.1 – Виводи АЦП МК ATmega16

Якщо необхідно перетворювати декілька сигналів одночасно, то вони перетворюються послідовно. На вивід AREF подається опорна напруга (верхня межа для перетворювача), нижня межа – завжди 0 В (GND). Як

верхня межа може бути використана напруга живлення V_{CC} або вбудоване джерело опорної напруги *2,56 В*. АЦП в мікроконтролері так само, як і всі інші пристрої, працює від тактової частоти МК. Частоту, що подається на АЦП можна зменшити за допомогою *подільника частоти* АЦП. Перетворення при цьому буде виконуватися повільніше, але більш якісно.

2.2 Регістри модуля ADC

ADMUX (ADC Multiplexer Selection Register) – регістр мультиплексора АЦП.

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

REFS1:0 (Reference Selection Bits)**.** Ці біти визначають величину верхньої межі опорної напруги для АЦП (табл. 2.1). Якщо дані біти змінити під час перетворення – зміна напруги відбудеться тільки за наступного перетворення.

Таблиця 2.1 – Вибір опорної напруги для АЦП

REFS1	REFS0	Вибір опорного навантаження
0	0	AREF (вихід МК), внутр. V_{ref} вимкнене
0	1	AVCC із зовнішнім конденсатором на виході AREF
1	0	Зарезервовано
1	1	Внутрішнє джерело 2,56 В з конденсатором на AREF

ADLAR (ADC Left Adjust Result). Біт відповідає за подання результату перетворення в регістрі даних АЦП. Установка біта в 1 призведе до вирівнювання результату ліворуч, 0 – праворуч. Зміна біта призводить до негайної зміни вирівнювання результату АЦП (для поточного і наступних перетворень). Детальніше про вирівнювання результату описано нижче (регістри даних ADCL і ADCH).

MUX4:0 (Analog Channel and Gain Selection Bits). Значення даних бітів визначають комбінацію підключених аналогових сигналів (табл. 2.2), а також коефіцієнт підсилення для диференціальних режимів включення. Якщо біти змінити під час перетворення, це відіб'ється на наступному перетворенні.

ADCSRA (ADC Control and Status Register A) – регістр налаштування і статусу АЦП.

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Біти регистра:

ADEN (ADC Enable). Даний біт вмикає/вимикає АЦП. Скидання даного біта в 0 перериває поточне перетворення.

ADSC (ADC Start Conversion). У режимі одиничного перетворення необхідно установлювати даний біт в 1 для запуску кожного перетворення. ADSC залишатиметься в 1 до тих пір, поки триває перетворення. По закінченню, він автоматично скидається в 0.

Таблиця 2.2 – Вибір вхідного каналу АЦП

MUX4:0	Вибір канала АЦП
00000	ADC0
00001	ADC1
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7
01000 … 11111	Інші режими вхідних сигналів для АЦП

ADATE(ADC Auto Trigger Enable). Цей біт відповідає за режим автозапуску перетворення за сигналом. Для відключення даного режиму цей біт повинен бути скинутий.

ADIF(ADC Interrupt Flag). Автоматично установлюється в 1 після закінчення перетворення. Оброблювач переривання після закінчення перетворення буде викликаний, якщо переривання для нього дозволені біт ADIE і біт I в регістрі SREG установлений в 1. Після виконання оброблювача, даний біт автоматично скидається в 0.

ADIE(ADC Interrupt Enable). Коли даний біт і прапорець I в регістрі SREG є установлені в 1, переривання після закінчення перетворення дозволені.

ADPS2:0(ADC Prescaler Select Bits). Дані біти визначають подільник робочої частоти МК для подачі її на АЦП. Тобто вони визначають швидкість роботи АЦП (частоту дискретизації). Значення даних бітів надані у табл. 2.3.

Таблиця 2.3 – Вибір вхідного каналу АЦП

ADPS2	ADPS1	ADPS0	Подільник
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Пара 8-бітних регістрів **ADCL і ADCH** утворюють 16-бітний *регістр даних* АЦП. При цьому виникає питання – як буде розміщений 10-бітний результат перетворення АЦП у 16 бітах регістра даних? Тут розробники передбачили два варіанті – вирівнювання праворуч, або ліворуч, в залежності від значення біта *ADLAR*. При цьому, зайві біти даних не враховуються.

при ADLAR = 0:

	7	6	5	4	3	2	1	0
ADCH	–	–	–	–	–	–	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

при ADLAR = 1:

	7	6	5	4	3	2	1	0
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	–	–	–	–	–	–

Якщо обране вирівнювання *ліворуч*, тоді можна *відкинути два молодші біти результата* у регістрі ADCL і користуватися тільки 8-бітним значенням з регістра *ADCH*. Це призводить до втрати точності перетворення (у $2^2 = 4$ разів), але працювати з таким результатом (8 бітів = 1 байт) набагато зручніше, ніж з 10-бітовим. На практиці, ці два молодших біта є найбільш схильними до перешкод і не мають цінності. Саме з цього приводу, внутрішнє джерело опорної напруги АЦП обране як 2,56 В ($2^8 = 256$).

При читанні регістрів, спочатку повинен бути зчитаний ADCL, потім ADCH. Якщо значення молодших бітів ігнорується – зчитується тільки ADCH.

2.3 Приклад програми

Програма реалізує вимірювач напруги в діапазоні 0…5 В. Зміна вимірюваної величини – джерела живлення МК, 5 В – відбувається вручну за допомогою підстроювального резистора 0…10 кОм. Отримане значення відображається в реальному часі на символьному ЖК-дисплеї LCD HD44780 (рис. 2.2).

Текст програми:

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

#define RS 0    //RS на 0-му розряді порту - сигнал керування LCD,
                //0=команда, 1=дані.
#define E  1    //E на 1-му разряді порту  - сигнал керування LCD,
                //1=строб даних.
```

```
void lcd_com(char pCom) //Функція передачі команд у LCD
{
	PORTB &= ~(1<<RS); //Запис команд
	PORTB |= (1<<E);
	_delay_us(50);
	PORTD = pCom; //Надсилання команди
	PORTB &= ~(1<<E);
	_delay_us(50);
}

void lcd_putchar(char pKey) //Функція виведення символів на LCD
{
	PORTB |= (1<<RS); //Запис даних
	PORTB |= (1<<E);
	_delay_us(50);
	PORTD = pKey; // Відобразити символ
	PORTB &= ~(1<<E);
	_delay_us(50);
}

void lcd_puts( const char *str) //Функція виводу рядка на LCD
{
	while(*str) lcd_putchar(*str++);
}

void lcd_init(void)
{
	//Ініціалізація дисплея:
	_delay_ms(20);		//Очікування включення LCD
	DDRB = 0b00000011;	//до входів RS та E LCD
	DDRD = 0b11111111;	//Виводи для команд і даних LCD

	lcd_com(0b00110000); //8-бітний режим
	_delay_ms(5);
	/*lcd_com(0b00110000); //8-бітний режим
	_delay_us(150);*/
	lcd_com(0b00110000); //8-бітний режим
	_delay_ms(1);
	lcd_com(0b00111000); //8 біт, 2 рядка
	_delay_ms(1);
	lcd_com(0b00000110); //Уст.інкремент лічильника адреси
	_delay_ms(1);
	//lcd_com(0b00001110); //Вкл. LCD, з курсором
	lcd_com(0b00001100); //Вкл. LCD, без курсора
	_delay_ms(1);
```

```
	lcd_com(0b00000001); //Очищення LCD
	_delay_ms(2);
}

//Функція переміщення курсора LCD
void lcd_movcur(unsigned char x, unsigned char y)
{
	if((x<16)&&(y<2))
	{
		lcd_com(128 + x + y*64); // y=0/1 - номер рядка
	}
}

//Функція ініціалізації LCD
void adc_init(void)
{
	ADMUX = 0b01100000;
	SFIOR &= (0b00011111);
	ADCSRA = 0b11101111;
}
//Оброблювач переривання за завершенням перетворення АЦП
ISR (ADC_vect)
{
	unsigned int ACP;
	ACP = ADCH*1.96; // 5 : 2,56 = 1.96
	unsigned char ACPmod100 = ACP % 100;
	unsigned char ACPmod10 = ACP % 10;
	asm("cli");
	lcd_movcur(0,1);
	lcd_putchar(0x30+(ACP-ACPmod100)/100);
	lcd_putchar(',');
	lcd_putchar(0x30+((ACPmod100-ACPmod10)/10));
	lcd_putchar(0x30+ACPmod10);
	lcd_putchar('V');
	_delay_ms(200);
	asm("sei");
}
//Головна функція
int main(void)
{
	asm("cli");
	lcd_init();
	adc_init();

	lcd_puts("ADC voltage:");
```

```
        asm("sei");

        while(1)
        {

        }
}
```

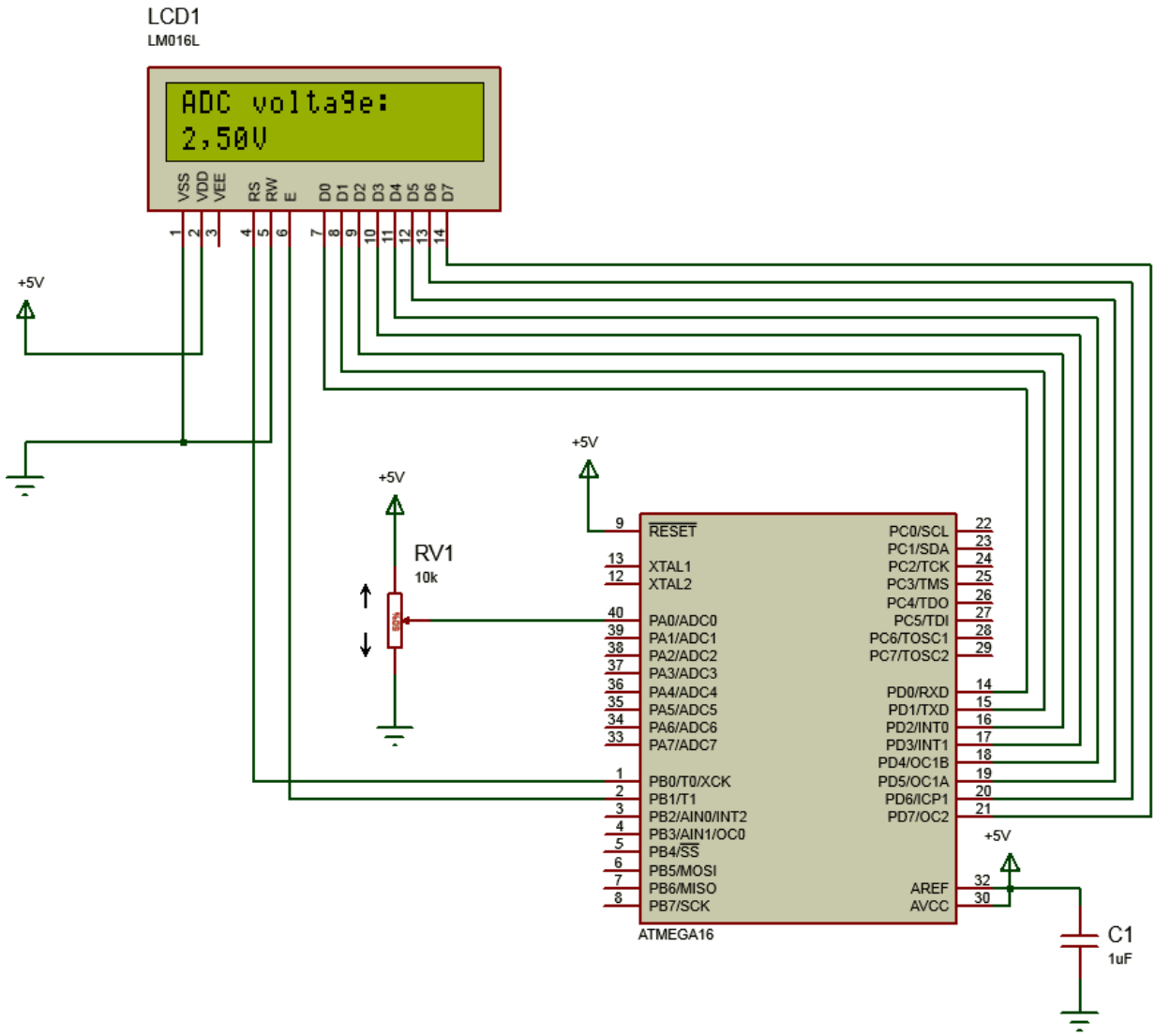

Рисунок 2.2 – Схема лабораторної роботи

3 Контрольні питання

1. Що таке АЦП? Області застосування.
2. Що вимірює АЦП – струм, напругу або опір?
3. На що впливає розрядність АЦП?
4. Що характеризує частота дискретизації АЦП?
5. Загальні характеристики модуля ADC у складі МК ATmega16.
6. Регістри ADC.

7. На що впливає опорна напруга АЦП? Джерела опорної напруги АЦП ATmega16.
8. Переривання які підтримує модуль ADC.

4 Домашнє завдання

1. Вивчити розділ 2 (Основні положення).
2. Підготувати протокол лабораторної роботи.
3. Відповісти на контрольні питання.

5 Лабораторне завдання

1. Скласти схему рис. 2.6 у середовищі Proteus.
2. Створити проект в AVR Studio для програми з п. 2.4.
3. Скомпілювати програму й отримати файл з розширенням .hex у папці \Debug проекту.
4. Установити отриманий файл в якості прошивки мікроконтролера на схемі в Proteus і перевірити працездатність схеми.
5. Розставити коментарі в рядках функції adc_init програми.

6 Оформлення протоколу

Протокол виконання лабораторної роботи включає: тему, мету роботи, виконання домашнього і лабораторного завдання, висновки.

7 Список рекомендованої літератури

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL – [5-е изд.] / Евстифеев А.В. – М.: Издательский дом «Додэка-XXI», 2008. – 560 с.

Лабораторна робота № 8

УНІВЕРСАЛЬНИЙ СИНХРОННИЙ/АСИНХРОННИЙ ПРИЙМАЧ/ПЕРЕДАВАЧ USART

1 Мета роботи

Вивчення характеристик, режимів роботи й ініціалізації модуля USART у складі МК ATmega16. Набуття навичок передавання даних.

2 Основні положення

USART (Universal Synchronous and Asynchronous Serial Receiver and Transmitte) – універсальний синхронний/асинхронний послідовний приймач/передавач.

USART – внутрішній устрій МК призначене для обміну інформацією з іншими пристроями за правилами стандарту RS-232.

RS-232 (Recommended Standard 232) – в телекомунікаціях, стандарт послідовного синхронного й асинхронного передавання двійкових даних між двома пристроями. Найчастіше використовується в промисловому і вузькоспеціальному обладнанні, вбудованих пристроях. Присутній в застарілих ПК як COM-порт. Швидкості обміну, підтримувані портом вимірюються в Бодах (1 Бод = 1 біт/с): 1200; 2400; 4800; 9600; 14400; 19200; 28800; 38400; 57600; 76800; 115200. Для сучасних пристроїв, максимальна швидкість може досягати й більших значень.

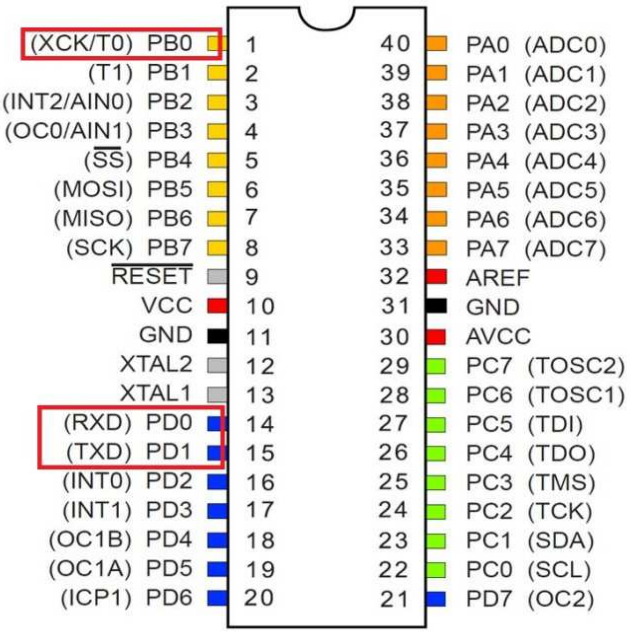

Рисунок 2.1 – Виводи модуля USART МК ATmega16

Виводи ATmega16, що пов'язані з USART (рис. 2.1):

- PD0 (RXD) – вхід USART;
- PD1 (TXD) – вихід USART;
- PB0 (XCK) – вхід/вихід зовнішнього тактового сигналу.

Основні властивості модуля:

- повнодуплексний обмін по послідовному каналу;
- синхронний і асинхронний режими роботи;
- синхронізація як від ведучого, так і від веденого пристрою;
- швидкість передавання може варіюватися в досить великих межах;
- слово даних довжиною 5…9 розрядів (в асинхронному режимі 8…9 розрядів), найчастіше – 8;
- апаратна підтримка генерації і контролю *біта парності кількості одиниць у слові даних* для виявлення помилок на прийомі;
- три переривання: «передавання завершено», «регістр даних передавача – порожній», «прийом завершено».

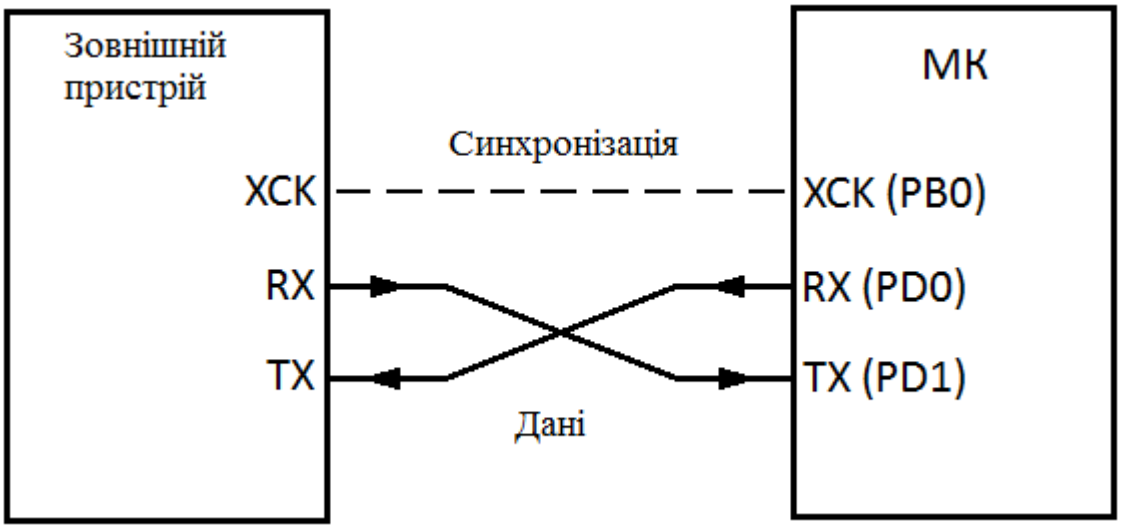

Рисунок 2.2 – Підключення зовнішнього пристрою до МК через USART

Приймач з передавачем з'єднуються хрест-навхрест (рис. 2.2). В асинхронному режимі вивід тактування XCK не використовується.

Формат кадру показано на рис. 2.3. Під кадром у даному випадку розуміється сукупність одного слова даних і супутньої інформації. Кадр починається зі старт-біта, за яким слідує молодший розряд слова даних. Після старшого розряду слова даних слідує один або два стоп-біти. Якщо включена схема формування біта парності P (визначає парність або непарність *кількості одиниць* у слові даних), він поміщується між старшим розрядом слова даних і першим стоп-бітом.

Формат кадру визначається розрядом UCSZ2 регістра UCSRB і розрядами UCSZ1, UCSZ0 регістра UCSRC. Вибір кількості стоп-бітів (в USART) здійснюється за допомогою розряду USBS регістра UCSRC.

Рисунок 2.3 – Формат кадру стандарта RS-232

Розряди UPM1:UPM0 регістра UCSRC визначають функціонування схеми контролю парності кількості одиниць у розрядах даних, які передаються через USART.

Значення біта парності обчислюється шляхом виконання логічної операції «Виключає АБО» над усіма розрядами переданого слова даних. Якщо використовується перевірка на непарність (odd parity), отриманий результат інвертується.

У МК ATmega16 для роботи з модулем USART використовуються наступні регістри:

- 3 керуючі регістри UCSRA, UCSRB, UCSRC;
- 2 регістри швидкості передавання UBRRL і UBRRH;
- регістр даних UDR.

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Рисунок 2.4 – Керуючий регістр UCSRA

RXC – Прапорець завершення прийому. Встановлюється в 1 за наявності непрочитаних даних в буфері приймача (регістр даних UDR). Скидається апаратно після спустошення буфера.

TXC – Прапорець завершення передачі. Установлюється в 1 після передачі всіх розрядів посилки з зсувного регістра передавача, за умови, що в регістр даних UDR не було завантажено нового значення. Скидається апаратно при виконанні підпрограми обробки переривання або програмно, записом в нього логічної 1.

UDRE – Прапорець спустошення регістра даних. Установлюється в 1 при порожньому буфері передавача (після пересилання байта з регістра даних

UDR у зсувний регістр передавача). Установлений прапорець означає, що в регістр даних можна завантажувати нове значення. Якщо розряд UDRIE регістра UCR (UCSRB) установлено, генерується запит на переривання «регістр даних – порожній». Скидається апаратно, при записі в регістр даних.
FE – Прапорець помилки кадрування. Установлюється в 1 при виявленні помилки кадрування, тобто, якщо перший стоп-біт прийнятої посилки дорівнює 0. Скидається при прийомі стоп-біта, якщо він дорівнює 1.
DOR – Прапорець переповнення. Установлюється в 1, якщо в момент виявлення нового старт-біта у зсувному регістрі приймача знаходиться останнє прийняте слово, а буфер приймача заповнений (два значення). Установлюється в 1, якщо новий кадр буде поміщено у зсувний регістр приймача до того, як з регістра даних буде прочитане попереднє слово. Прапорець скидається при пересиланні прийнятих даних зі зсувного регістра приймача в буфер.
PE – Прапорець помилки контролю парності. Установлюється в 1, якщо в даних, що знаходяться в буфері приймача, виявлена помилка контролю парності. При відключеному контролі парності цей розряд постійно читається як 0.
U2X – Подвоєння швидкості обміну. Якщо цей розряд установлений в 1, коефіцієнт ділення передподільника частоти контролера зменшується з 16 до 8, подвоюючи тим самим швидкість асинхронного обміну по послідовному каналу. У USART розряд U2X використовується тільки при асинхронному режимі роботи. У синхронному режимі він повинен бути скинутий.
MPCM – Режим мультипроцесорного обміну. Розряд MPCM використовується в режимі мультипроцесорного обміну. Якщо він установлений в 1, ведений мікроконтролер очікує на прийом кадру, що містить адресу. Кадри, що не містять адреси пристрою, ігноруються.

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 2.5 – Керуючий регістр UCSRB

RXCIE – Дозвіл переривання по завершенню прийому. Якщо даний розряд установлений в 1, то при установленні прапорця RXC регістра UCSRA генерується переривання.
TXCIE – Дозвіл переривання по завершенню передачі. Якщо даний розряд установлений в 1, то при установленні прапорця TXC регістра UCSRA генерується відповідне переривання.
UDRIE – Дозвіл переривання при очищенні регістра даних UART. Якщо даний розряд установлений в 1, при установленні прапорця UDRE в регістрі UCSRA генерується переривання.

RXEN – Дозвіл прийому. При установленні цього розряду в 1 дозволяється робота приймача USART/UART і перевизначається функціонування виводу RXD.
TXEN – Дозвіл передачі. При установленні цього розряду в 1 дозволяється робота передавача UART і перевизначається функціонування виводу TXD.
UCSZ2 – Формат посилок. Цей розряд використовується для завдання розміру слів даних, які передаються по послідовному каналу. У модулях USART він використовується спільно з розрядами UCSZ1 і UCSZ0 регістра UCSRC.
RXB8 – 8-й розряд отриманих даних. При використанні 9-розрядних слів даних цей розряд містить значення старшого розряду прийнятого слова. У разі USART, вміст цього розряду має бути зчитаний до читання регістра даних UDR.
TXB8 – 8-й розряд переданих даних. При використанні 9-розрядних слів даних, вміст цього розряду є старшим розрядом переданого слова. Необхідне значення має бути занесено в цей розряд до завантаження байта даних в регістр UDR.

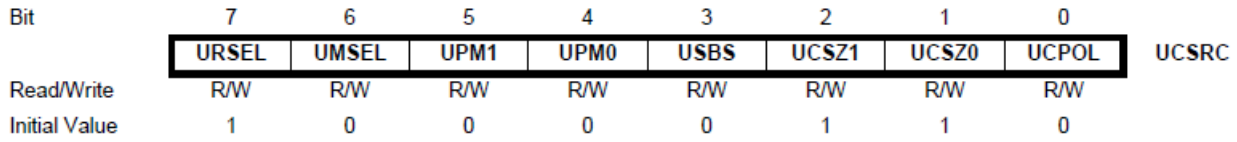

Рисунок 2.6 – Керуючий регістр UCSRC

URSEL – Вибір регістра. Цей розряд визначає, в який з регістрів модуля проводиться запис. Якщо розряд установлений в 1, звернення проводиться до регістра UCSRC. Якщо ж розряд скинутий в 0, звернення проводиться до регістра UBRRH.
UMSEL – Режим роботи USART. Якщо розряд скинутий в 0, модуль USART працює в асинхронному режимі. Якщо розряд установлено в 1, то модуль USART працює в синхронному режимі.
UPM1 і UPM0 – Режим роботи схеми контролю та формування парності. Ці розряди визначають функціонування схем контролю та формування парності (табл. 2.1).

Таблиця 2.1 – Біти контролю парності

UPM1	**UPM0**	**Режим контролю парності**
0	0	Відсутній
0	1	Зарезервовано
1	0	Перевірка на парність
1	1	Перевірка на непарність

USBS – Кількість стоп-бітів. Цей розряд визначає кількість стоп-бітів, що посилаються передавачем. Якщо розряд є скинутий в 0, передавач посилає 1

стоп біт, якщо установлений в 1, то 2 стоп біта. Для приймача вміст цього розряду є байдужим.
UCSZ1 і UCSZ0 – Формат посилок. Спільно з розрядом UCSZ2 ці розряди визначають кількість розрядів даних у посилках (табл. 2.2).

Таблиця 2.2 – Кількість розрядів даних у посилках

UCSZ2	UCSZ1	UCSZ0	Кількість розрядів даних
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	Зарезервовано
1	0	1	Зарезервовано
1	1	0	Зарезервовано
1	1	1	9

UCPOL – Полярність тактового сигналу. Значення цього розряду визначає момент видачі та зчитування даних на виводах модуля. Розряд використовується тільки при роботі у синхронному режимі. При роботі в асинхронному режимі він повинен бути скинутий в 0.

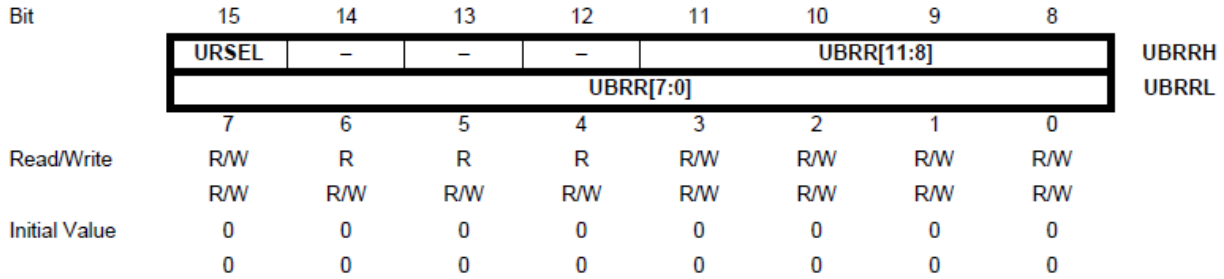

Рисунок 2.7 – Регістри швидкості передавання UBRRL і UBRRH

USEL – Вибір регістра. Цей розряд визначає, в який з регістрів модуля проводиться запис. Якщо розряд установлений в 1, звернення проводиться до регістра UCSRC. Якщо ж розряд є скинутий в 0, звернення проводиться до регістра UBRRH.
UBRR11-UBRR0 – Значення швидкості передавання в бодах. При роботі в асинхронному режимі швидкість обміну визначається не тільки вмістом регістра UBRR, але і станом розряду U2X регістра UCSRA. Якщо цей розряд установлено в 1, коефіцієнт ділення передподільника зменшується в два рази, а швидкість обміну відповідно подвоюється. При роботі в синхронному режимі цей розряд повинен бути скинутий.

Отже, швидкість обміну визначається наступними формулами:

• асинхронний режим (звичайний, U2X = 0):

BAUD = FCPU/(16*(UBRR + 1));

• асинхронний режим (прискорений, U2X = 1):

BAUD = FCPU/(8*(UBRR + 1));

• синхронний режим ведучого:

BAUD = FCPU/(2*(UBRR + 1)),

де BAUD – швидкість передавання в бодах; FCPU – тактова частота МК; UBRR – вміст регістра контролера швидкості передачі (0 ... 4095).

Кількість змін інформаційного параметра несучого періодичного сигналу за секунду вимірюється в бодах. 1 Бод дорівнює одній зміні інформаційного параметра за секунду. Наприклад, якщо такт передавання інформації дорівнює 0,1 секунди, то сигнал змінюється зі швидкістю 10 Бод.

Приклад

Здійснимо передавання даних по інтерфейсу USART між двома МК. Складемо дві програми: для передавача і приймача. Передавач буде кожну секунду відправляти веденому МК число, яке збільшується на 1. Приймач буде приймати це значення і виводити його розряди на PORTA (схема для Proteus на рис. 2.8).

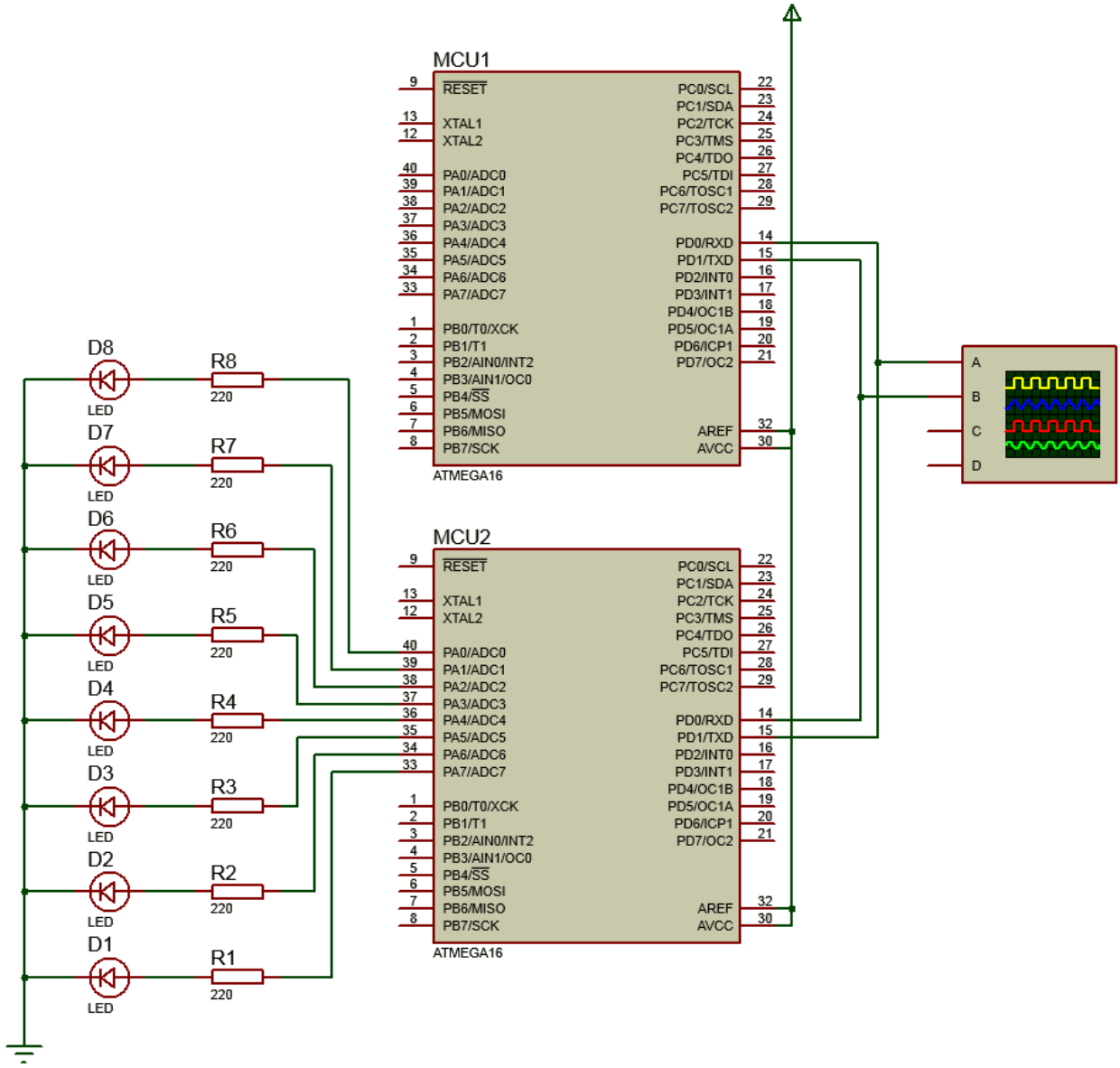

Рисунок 2.8 – Схема експерименту в Proteus

Лістинг програми МК-передавача (MCU1):

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
//установлення швидкості обміну
#define BAUD 9600
#define UBRR_VAL F_CPU/16/BAUD-1
void usart_init(unsigned int speed)
{
	//швидкість 9600 Бод
	UBRRH=(unsigned char) (speed>>8);
	UBRRL=(unsigned char) speed;
	UCSRA=0x00;
	//формат посилки: 8 біт даних, 1 стоп-біт
	UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
	//дозвіл роботи передавача
	UCSRB=(1<<TXEN);
}
int main(void)
{
	usart_init(UBRR_VAL);//ініціалізація модуля
	char i=0;//змінна для відправки
	while(1)
	{
		UDR=i; //запис в регістр даних UART і передавання
		_delay_ms(1000);
		i+=1;
	}
}
```

Лістинг програми МК-приймача (MCU2):

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
//установлення швидкості обміну
#define BAUD 9600
#define UBRR_VAL F_CPU/16/BAUD-1
void usart_init(unsigned int speed)
{
	//швидкість 9600 Бод
	UBRRH=(unsigned char) (speed>>8);
	UBRRL=(unsigned char) speed;
	UCSRA=0x00;
	UCSRB|=(1<<RXEN);//дозвіл роботи приймача
	UCSRB|=(1<<RXCIE);//дозвіл переривань за прийманням
	//формат посилки: 8 біт даних, 1 стоп-біт
	UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
}
//оброблювач переривання за завершенням приймання:
ISR(USART_RXC_vect)
{
```

```
      PORTA=UDR; //виведення прийнятого байта в порт
}
int main(void)
{
      DDRA=0xFF; //порт А в режим виходa
      PORTA=0x00;
      usart_init(UBRR_VAL);//ініціалізація модуля
      sei(); //дозволити всі переривання
      while(1);
}
```

3 Контрольні питання

1. Опишіть формат кадра стандарту RS-232.
2. Властивості вбудованого інтерфейсу USART МК AVR.
3. Які основні настройки виконуються в регістрах контролю та статусу USART при ініціалізації модуля?
4. Як встановлюється швидкість обміну даними?
5. Які переривання передбачені для модуля USART?

4 Домашнє завдання

1. Вивчити розд. 2 (Основні положення).
2. Підготувати протокол лабораторної роботи.
3. Відповісти на контрольні питання.

5 Лабораторне завдання

1. Скласти схему рис. 2.6 у середовищі Proteus.
2. Створити проект в AVR Studio для програми з п. 2.4.
3. Скомпілювати програму й отримати файл з розширенням .hex у папці \Debug проекту.
4. Установити отриманий файл в якості прошивки мікроконтролера на схемі в Proteus і перевірити працездатність схеми.
5. Занести у протокол дані про обраний у програмі формат передавання.

6 Оформлення протоколу

Протокол виконання лабораторної роботи включає: тему, мету роботи, виконання домашнього і лабораторного завдання, висновки.

7 Список рекомендованої літератури

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL – [5-е изд.] / Евстифеев А.В. – М.: Издательский дом «Додэка-XXI», 2008. – 560 с.

Лабораторна робота № 9

ІНТЕРФЕЙС SPI МК ATMEGA16

1 Мета роботи

Вивчення характеристик і режимів роботи послідовного синхронного інтерфейсу SPI МК ATmega16. Набуття навичок обміну даних між ведучим і веденим пристроями SPI.

2 Основні положення

2.1 Підключення пристроїв по інтерфейсу SPI

SPI (англ. Serial Peripheral Interface – послідовний периферійний інтерфейс) – чотирипровідний синхронний інтерфейс, призначений для послідовного обміну даними між мікросхемами. Інтерфейс є розроблений компанією Motorola, але наразі використовується всіма виробниками. Даний інтерфейс відрізняють простота використання і реалізації, висока швидкість обміну і мала дальність дії.

За будь-якого обміну даними по SPI один із пристроїв є ведучим (Master), а інший – веденим (Slave). Зазвичай (але не завжди, наприклад, при прошиванні) в ролі ведучого виступає мікроконтролер. Ведучий переводить периферійний пристрій в активний стан і формує тактовий сигнал і дані. У відповідь ведений пристрій передає ведучому свої дані. Передача даних в обидві сторони відбувається синхронно з тактовим сигналом.

Фізично SPI реалізується на основі *циклічного зсувного регістра* (об'єднує регістри даних пристроїв, які з'єднані по SPI), який виконує функції як передавача, так і приймача. Принцип обміну даними по SPI проілюстрований на рис. 2.1. Після побітової передачі одного байта вміст регістрів даних двох SPI-пристроїв обмінюється.

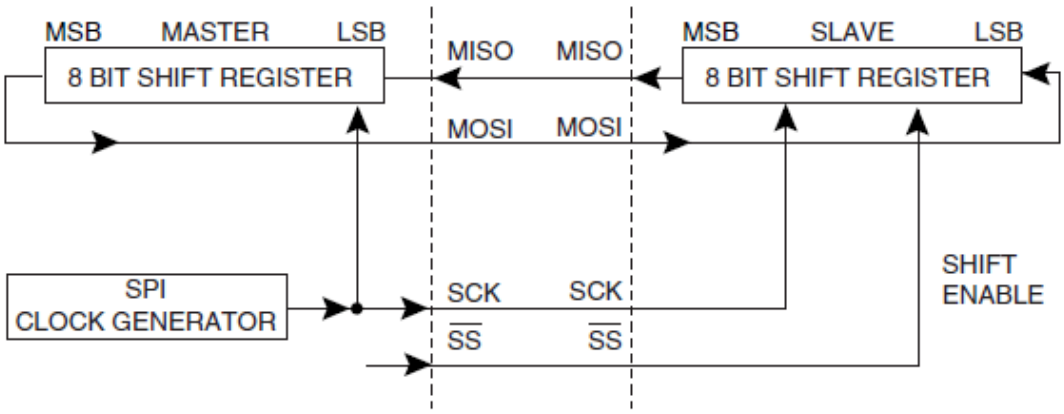

Рисунок 2.1 – Принцип обміну даними по інтерфейсу SPI

Сигнали, які використовуються даним інтерфейсом, мають таке призначення:
MOSI – Master Output/Slave Input. Вихід ведучого/вхід веденого. Служить для передавання даних від провідного пристрою до веденого.
MISO – Master Input/Slave Output. Вхід ведучого/вихід веденого. Служить для передавання даних від веденого пристрою до ведучого.
SLK – Serial Clock. Сигнал синхронізації. Служить для передавання тактового сигналу всім відомим пристроям.
SS – Slave Select. Вибір веденого. Служить для вибору веденого пристрою.

Виробники мікросхем часто використовують інші назви для цих сигналів. Альтернативні варіанти можуть бути наступними:
MOSI: DO, SDO, DOUT.
MISO: DI, SDI, DIN.
SCK: CLK, SCLK.
SS: CS, SYNC.

Типова схема з'єднання двох пристроїв по SPI показана на рис 2.2.

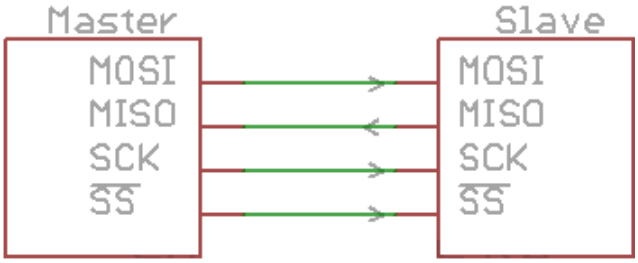

Рисунок 2.2 – Типова схема з'єднання двох пристроїв по SPI

Також, можливо підключення до ведучого пристрою декількох ведених пристроїв. Однак у будь-який момент часу обмін може відбуватися тільки з одним із них, інші повинні знаходитися в неактивному стані (рис. 2.3).

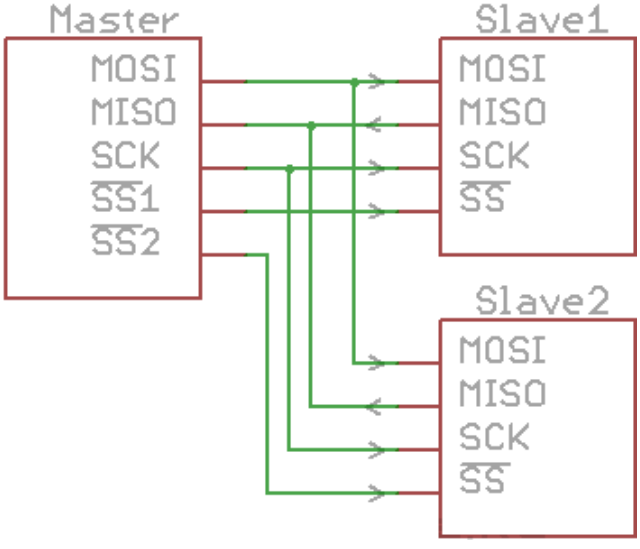

Рисунок 2.3 – Підключення до ведучого декількох ведених пристроїв

При каскадному з'єднанні по SPI (рис. 2.4) зсувні регістри пристроїв утворюють єдиний регістр, і кількість ліній SPI залишається рівною 4. Однак таке підключення підтримують не всі пристрої.

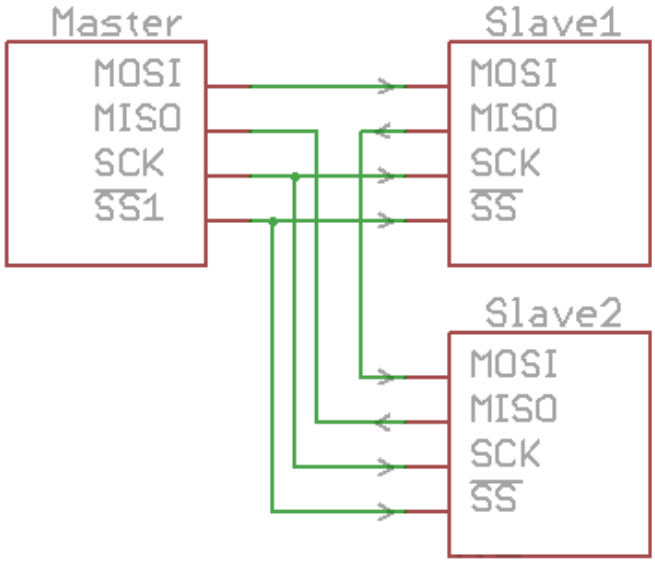

Рисунок 2.4 – Каскадне з'єднання

Також є можливим скорочений варіант схеми підключення, коли лінія MOSI або MISO не використовується (рис. 2.5). Тобто передавання даних здійснюється тільки в один бік. Така схема, наприклад, використовуються при підключенні до мікроконтролера зовнішніх мікросхем ЦАП і АЦП.

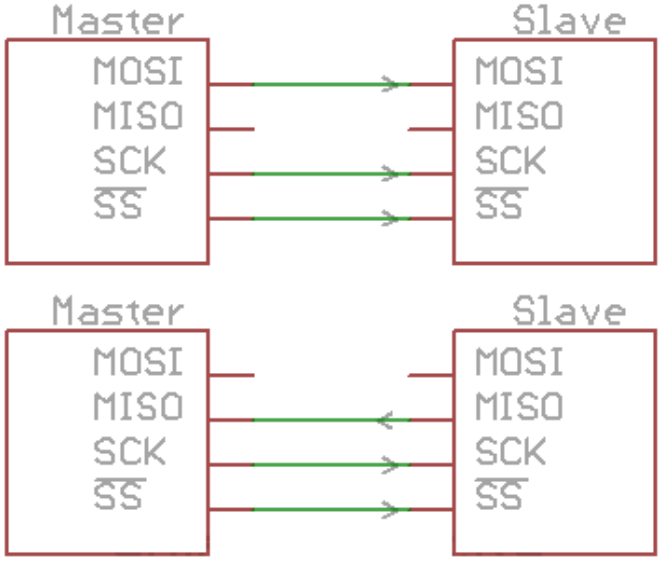

Рисунок 2.5 – Однобічне підключення

2.2 Протокол обміну SPI

Протокол обміну по інтерфейсу SPI є аналогічний логіці роботи зсувного регістра і полягає в послідовному побітовому введенні/виведенні даних за певними фронтами тактового сигналу. Установлення і вибирання даних здійснюється за протилежними фронтами тактового сигналу.

Специфікація SPI передбачає чотири режими передавання даних, які відрізняються між собою співвідношенням фази і полярності тактового сигналу і переданих даних.

Ці режими описуються двома параметрами:

CPOL – clock polarity. Полярність тактового сигналу визначає початковий рівень сигналу синхронізації.

CPHA – clock phase. Фаза тактового сигналу визначає послідовність установлення і вибирання даних.

Рис. 2.6…2.9 нижче ілюструють усі чотири режими обміну SPI.

SPI mode 0: CPOL = 0, CPHA = 0. Тактовий сигнал починається з рівня логічного 0. Защеплення даних виконується за наростаючим фронтом. Зміна даних відбувається за падаючим фронтом. Моменти защеплення даних показані на рис. 2.6…2.9 стрілками.

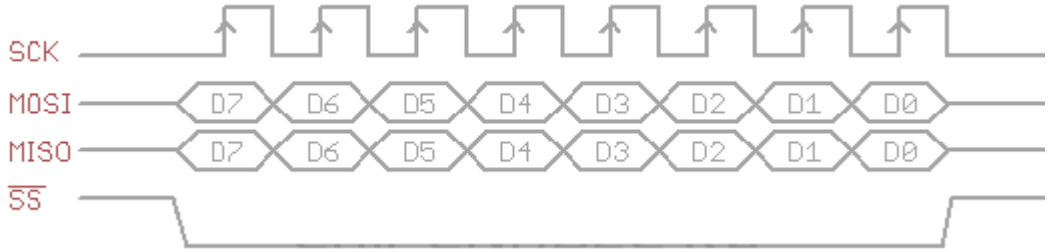

Рисунок 2.6 – SPI режим 0

SPI mode 1: CPOL = 0, CPHA = 1. Тактовий сигнал починається з рівня логічного 0. Зміна даних відбувається за наростаючим фронтом. Защеплення даних виконується за падаючим фронтом.

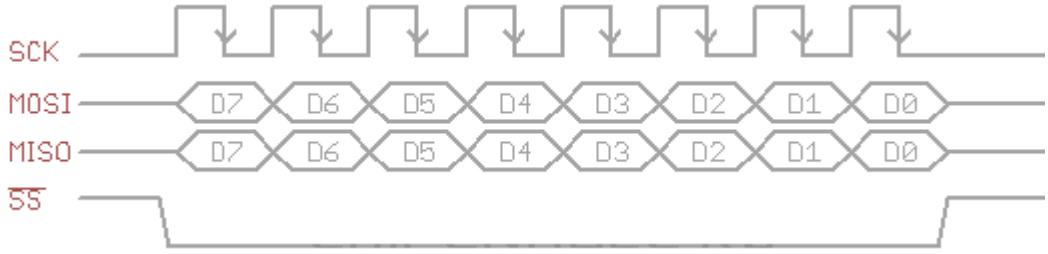

Рисунок 2.7 – SPI режим 1

SPI mode 2: CPOL = 1, CPHA = 0. Тактовий сигнал починається з рівня логічної 1. Защеплення даних виконується за падаючим фронтом. Зміна даних виконується за наростаючим фронтом тактового сигналу.

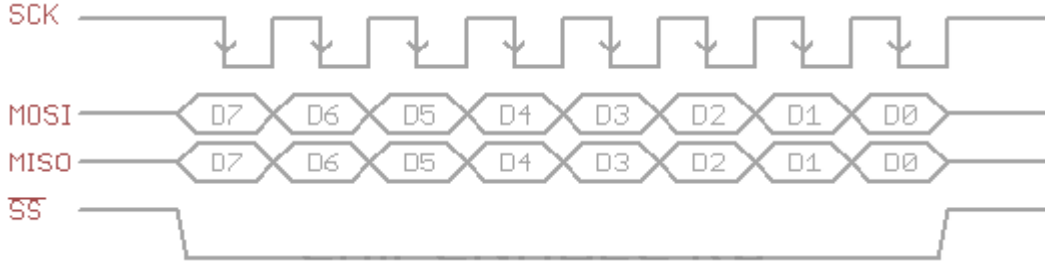

Рисунок 2.8 – SPI режим 2

SPI mode 3: CPOL = 1, CPHA = 1. Тактовий сигнал починається з рівня логічної 1. Зміна даних виконується за падаючим фронтом тактового сигналу. Защеплення даних виконується за наростаючим фронтом.

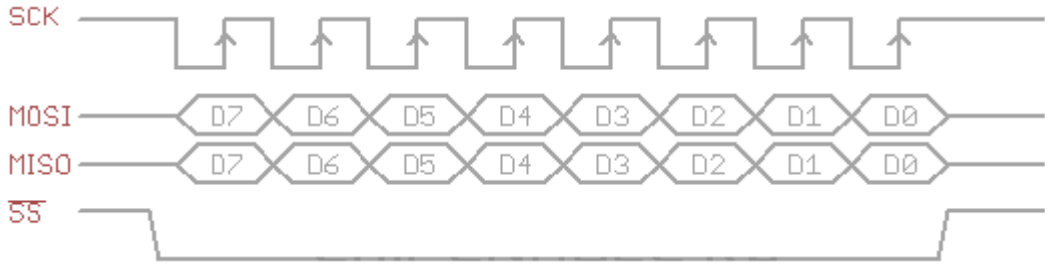

Рисунок 2.9 –SPI режим 3

Сучасні мікроконтролери підтримують усі чотири режими роботи SPI. Варто відзначити, що передавання даних по SPI може починатися не тільки зі старшого біта, але й з молодшого. Кількість байт переданих за час утримання сигналу вибору (SS) нічим не обмежена і визначається специфікацією використовуваного веденого пристрою. Також у специфікації на ведений пристрій вказуються підтримувані режими роботи SPI, максимальна частота тактового сигналу, вміст переданих або отриманих даних.

2.3 Модуль SPI МК ATmega16

Модуль SPI мікроконтролера AVR ATmega16 використовує для своєї роботи чотири виводи – MOSI, MISO, SCK і SS. Коли модуль є не задіяним, ці виводи працюють в якості ліній портів введення/виведення загального призначення. Коли модуль є включений, режими роботи цих виводів перевизначаються згідно з табл. 2.1.

Таблиця 2.1 – Режим роботи виводів SPI

Вихід	Напрям	
	Ведучий	Ведений
MOSI	*	Вхід
MISO	Вхід	*
SCK	*	Вхід
SS	*	Вхід

Якщо до МК підключено більше одного периферійного пристрою в якості додаткових виводів вибору веденого (SS), можна використовувати будь-які виводи загального призначення. При цьому штатний вивід SS повинен бути завжди правильно налаштований, навіть якщо він не використовується.

У мікроконтролері ATmega16 для роботи з модулем SPI використовуються три 8-розрядних регістри:

- керуючий регістр SPCR;

- статусний регістр SPSR;
- регістр даних SPDR.

Конфігурація модуля SPI установлюється за допомогою регістра SPCR (SPI Control Register).

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 2.10 – Регістр SPCR

SPIE – дозволяє/забороняє переривання від модуля SPI. Якщо біт установлено в 1, переривання від SPI дозволені.
SPE – вмикає/вимикає модуль SPI. Якщо біт установлено у 1, модуль SPI включений.
DORD – визначає порядок передавання даних. Коли біт установлений в 1, вміст регістра даних передається молодшим бітом вперед. Коли біт скинуто, то старшим бітом вперед.
MSTR – визначає режим роботи мікроконтролера. Якщо біт установлено в 1, мікроконтролер працює в режимі Master (ведучий). Якщо біт є скинутий – в режимі Slave (ведений). Зазвичай мікроконтролер працює в режимі Master.
CPOL і CPHA – визначають в якому режимі працює модуль SPI (табл. 2.2). Необхідний режим роботи залежить від периферійного пристрою, що використовується.

Таблиця 2.2 – Режими SPI, в залежності від розрядів CPOL і CPHA

Режим SPI	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

SPR1 і SPR0 – визначають частоту тактового сигналу модуля SPI, тобто швидкість обміну (табл. 2.3). Вона виходить шляхом розподілу тактової частоти МК. Максимально можлива швидкість обміну завжди вказується в специфікації периферійного пристрою.
Статусний регістр SPSR (SPI Status Register) призначений для контролю стану SPI модуля, крім того він містить додатковий біт керування швидкістю обміну.

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 2.11 – Регістр SPSR

SPIF – прапорець переривання від SPI. Він установлюється в 1 після закінчення передачі байта даних. Якщо переривання модуля є дозволеними, одночасно з установленням цього прапорця генерується переривання від SPI. Також цей прапорець встановлюється в 1 при переведенні МК з режиму Master в режим Slave за допомогою виводу SS. Скидання прапорця відбувається апаратно, при виклику підпрограми обробки переривання або після читання регістра SPSR з подальшим зверненням до регістра даних SPDR.
WCOL – прапор конфлікту запису. Прапор встановлюється в 1, якщо під час передавання даних відбувається спроба запису в регістр даних SPDR. Прапор скидається апаратно після читання регістра SPSR з подальшим зверненням до регістру даних SPDR.
SPI2X – біт подвоєння швидкості обміну. Установлення цього розряду в 1 подвоює частоту тактового сигналу SCK. Мікроконтролер при цьому повинен працювати в режимі master.

Взаємозв'язок між бітами SPR0, SPR1, SPI2X і частотою тактового сигналу SCK показано в табл. 2.3.

Таблиця 2.3 – Вибір частоти тактового сигналу SCK

SPI2X	**SPR1**	**SPR0**	**Частота SCK**
0	0	0	$f_{CPU}/4$
0	0	1	$f_{CPU}/16$
0	1	0	$f_{CPU}/64$
0	1	1	$f_{CPU}/128$
1	0	0	$f_{CPU}/2$
1	0	1	$f_{CPU}/8$
1	1	0	$f_{CPU}/32$
1	1	1	$f_{CPU}/64$
де f_{CPU} – тактова частота МК			

Для передавання і приймання даних служить регістр SPDR (SPI Data Register). Запис даних в цей регістр ініціює передавання даних. При читанні цього регістра, насправді зчитується вміст вхідного буфера зсувного регістра модуля SPI. Таким чином, за один цикл передачі даних вміст регістрів даних двох пристроїв обмінюється.

Інтерфейс SPI може застосовуватися для підключення до МК різних периферійних пристроїв, наприклад, мікросхем пам'яті з підтримкою SPI а також, SD/microSD Flash карт пам'яті. Перед підключенням слід узгодити режим SPI-модуля МК з підтримуваним режимом SPI пристрою.

Після ініціалізації модуля SPI МК в заданому режимі для підготовки до обміну даними з веденим пристроєм слід виконати:

1) Вибір веденого пристрою, подавши на його вхід SS логічний 0.

2) Подати тактовий сигнал на лінію SCK.

2.4 Приклад обміну даними між двома МК через інтерфейс SPI

Складемо дві програми для двох МК: ведучого і веденого. Ведучий буде кожну секунду відправляти веденому по інтерфейсу SPI число, яке збільшується на одиницю. Ведений буде приймати це значення і виводити його на PORTA. Схема показана на рис. 2.12.

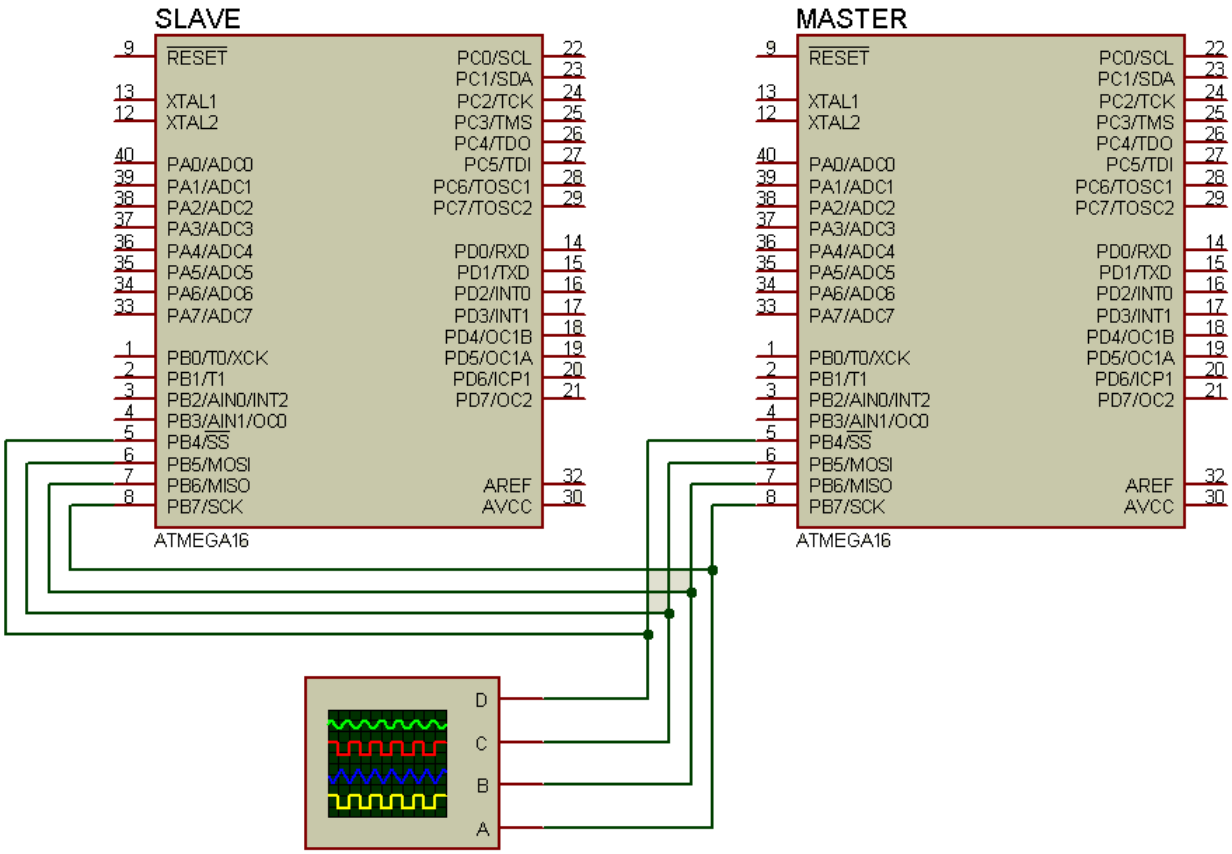

Рисунок 2.12 – Схема лабораторної роботи

Програма для Ведучого (Master):

```
define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
 // Функція ініціалізації ведучого МК
void SPI_MasterInit(void)
{
     // Установлення виводів SPI на вихід
     DDRB = 0xFF;
     //Включення SPI у режимі ведучого,
     //і установлення частоти тактування fclk/128
     SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);
}
 /* Функція передавання байта даних outData. Очікує
закінчення передавання та повертає прийнятий по MOSI байт */
```

```
unsigned char SPI_MasterTransmit(char outData)
{
     // Початок передачі
     SPDR = outData;
     // Очікування закінчення передачі
     while(!(SPSR & (1<<SPIF))) ;
     return SPDR; //повертаємо прийнятий байт
}

int main(void)
{
     char i=0;
     SPI_MasterInit();
     while(1)
     {
          i+=1;
          SPI_MasterTransmit(i);
          _delay_ms(1000);
     }
}
```

Програма для Веденого (Slave):

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

// Функція ініціалізації ведучого МК
void SPI_SlaveInit(void)
{
     // Установлення виходів MISO и SCK на вхід
     DDRB = 0x00;
     //Включення SPI, режим ведучого, дозвіл переривань
     SPCR = (1<<SPE)|(1<<SPIE);
}

ISR(SPI_STC_vect) //переривання SPI по прийому байта
{
     PORTA=SPDR; //виведення байта на PORTA
}

int main(void)
{
```

```
    SPI_SlaveInit();
    sei();
    while(1);
}
```

3 Контрольні питання

1. Призначення сигналів і схеми підключення по SPI.
2. Принцип обміну даними по SPI.
3. Регістри модуля SPI МК ATmega16.
4. Як задається і чим обмежена швидкість обміну по SPI?

4 Домашнє завдання

1. Вивчити розд. 2 (Основні положення).
2. Підготувати протокол лабораторної роботи.
3. Відповісти на контрольні питання.

5 Лабораторне завдання

1. Скласти схему рис. 2.6 у середовищі Proteus.
2. Створити проект в AVR Studio для програми з п. 2.4.
3. Скомпілювати програму й отримати файл з розширенням .hex в папці \Debug проекту.
4. Установити отриманий файл в якості прошивки мікроконтролера на схемі в Proteus і перевірити працездатність схеми.
5. Зарисувати характер показів осцилографа з позначенням назв сигналів.

6 Оформлення протоколу

Протокол виконання лабораторної роботи включає: тему, мету роботи, виконання домашнього і лабораторного завдання, висновки.

7 Список рекомендованої літератури

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL – [5-е изд.] / Евстифеев А.В. – М.: Издательский дом «Додэка-XXI», 2008. – 560 с.

Лабораторна робота № 10

Інтерфейс TWI (I2C) МК ATmega16

1 Мета роботи

Вивчення послідовної шини I2C. Набуття навичок передавання даних між пристроями.

2 Основні положення

Інтерфейс I2C (Inter-Integrated Circuit) – послідовна шина даних для зв'язку інтегральних схем, що використовує дві двонаправлені лінії зв'язку SDA і SCL. По SDA передаються дані, по SCL – тактовий сигнал. Обидві лінії підтягнуті через резистори 4,7 кОм до живлючої напруги. Допустима ємність ліній з пристроями – до 400 пФ. Схема підключення – на рис. 2.1. Автор, фірма Philips, за використання назви цього інтерфейсу вимагає ліцензійних відрахувань, тому в мікроконтролерах Atmel використовується власна назва TWI – two-wire interface («двухпроводной інтерфейс»).

Переваги шини I2C:

1. Потрібно тільки дві лінії – лінія даних (SDA) і лінія синхронізації (SCL). Кожен пристрій, підключений до шини, може бути програмно адресований за унікальною адресою. У кожен момент часу існує просте відношення ведучий/ведений: ведучі можуть працювати як ведучий-передавач і ведучий-приймач.
2. Шина дозволяє мати декілька ведучих, надаючи засоби для визначення колізій і арбітраж, щоб запобігти пошкодженню даних в ситуації, коли два або більше ведучих одночасно починають передавати дані. У стандартному режимі забезпечується передавання послідовних 8-бітних даних зі швидкістю до 100 кбіт/с і до 400 кбіт/с у «швидкому» режимі.
3. Вбудований в мікросхеми фільтр придушує сплески, забезпечуючи цілісність даних.

2.1 Формат передавання даних

Інтерфейс I2C є синхронним, тому кожен біт даних на лінії SDA супроводжується імпульсом на лінії синхронізації SCL (рис. 2.2). Рівень даних повинен бути стабільним, коли на лінії синхронізації присутня логічна 1. Винятком для цього правила є генерація умов (СТАРТ/СТОП).

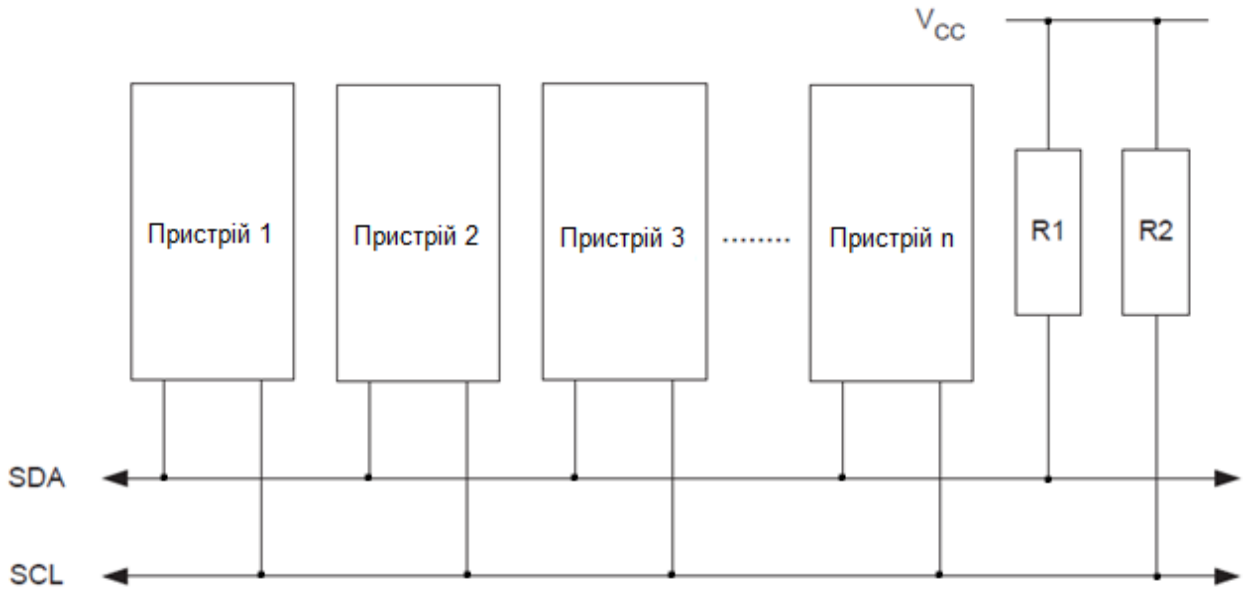

Рисунок 2.1 – Схема підключення пристроїв по шині I2C

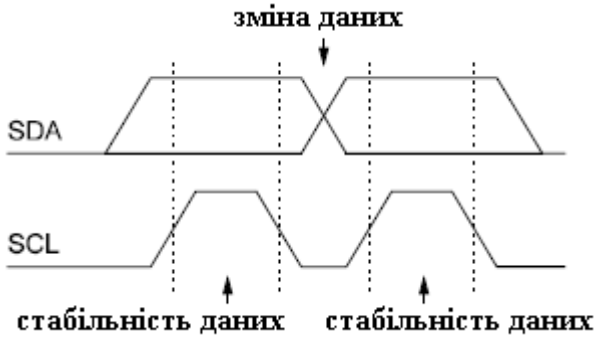

Рисунок 2.2 – Умови стабільності даних

Ведучий пристрій ініціює і закінчує передавання даних. Передавання ініціюється, коли ведучий формує умову СТАРТ на шині, і припиняється, коли ведучий формує на шині умову СТОП (рис. 2.3). Між умовами СТАРТ і СТОП шина вважається зайнятою і в цьому випадку ніякої інший ведучий не може здійснювати керуючий вплив на шині. Існують особливі випадки, коли нова умова СТАРТ виникає між умовами СТАРТ і СТОП. Даний випадок іменується як "Повітряний старт" і використовується при необхідності ініціювати майстром новий сеанс зв'язку, не втрачаючи за цьому керування шиною. Після "Повторного старту" шина вважається зайнятою до наступного СТОП. Це ідентично поведінці після СТАРТУ.

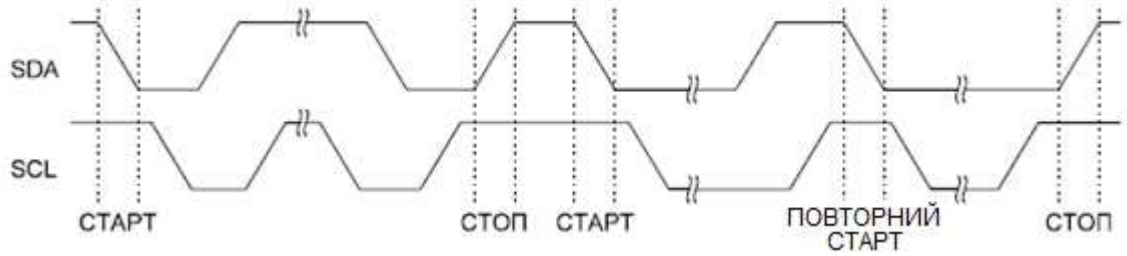

Рисунок 2.3 – Умови СТАРТ і СТОП на шині I2C

Розділяють чотири режими роботи кожного окремого модуля I2C:

– ведучий передавач;
– ведучий приймач;
– ведений передавач;
– ведений приймач.

2.2 Формат адресного пакета

Всі адресні пакети, що передаються по шині I2C (рис. 2.4), складаються з 9 біт: 7 біт адреси пристрою до якого йде звернення, один біт для завдання типу операції читання або запису (R/W) і один біт підтвердження (ACK). Якщо біт R/W = 1, то буде виконана операція читання, інакше – запис. Якщо підлеглий розпізнає, що до нього відбувається адресація, то він повинен сформувати низький рівень на лінії SDA на 9-му циклі SCL (формування біта підтвердження – ACK). Якщо адресується підлеглий пристрій зайнято або з яких-небудь інших причин не може обслужити ведучий пристрій, то на лінії SDA необхідно залишити високий рівень під час циклу підтвердження (NACK –no acknowledge). Ведучий після цього може передати умову СТОП або ПОВТОРНИЙ СТАРТ для ініціації нової передачі.

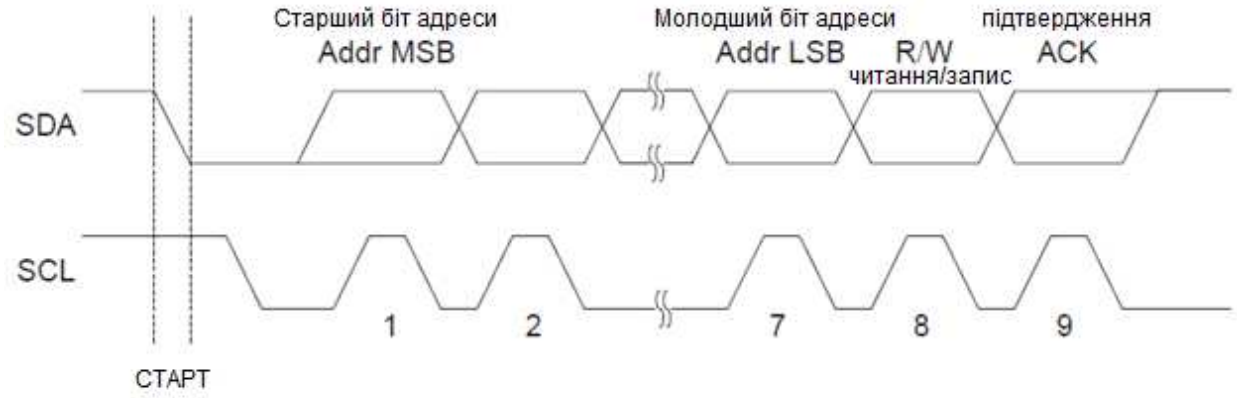

Рисунок 2.4 – Формат адресного пакета I2C

Слід зазначити, що 7-бітна адресація пристроїв обмежує їх кількість відомим співвідношенням: $2^7 = 128$. У разі I2C, кількість доступних адрес для пристроїв дорівнює 127, а адреса, що містить сім нулів, означає звернення до всіх підключених пристрох.

Пакети даних складаються з байта даних і біта квітування, тобто також мають довжину 9 біт (рис. 2.5). Після приймання кожного байта даних, приймаючий пристрій (приймач) відповідає передавальному пристрою (передавачу), установлюючи на лінії SDA низький рівень (це і є біт квітування). Якщо приймаючий пристрій отримав останній байт або більше не може продовжувати приймання даних, він повинен "залишити" на лінії SDA високий рівень.

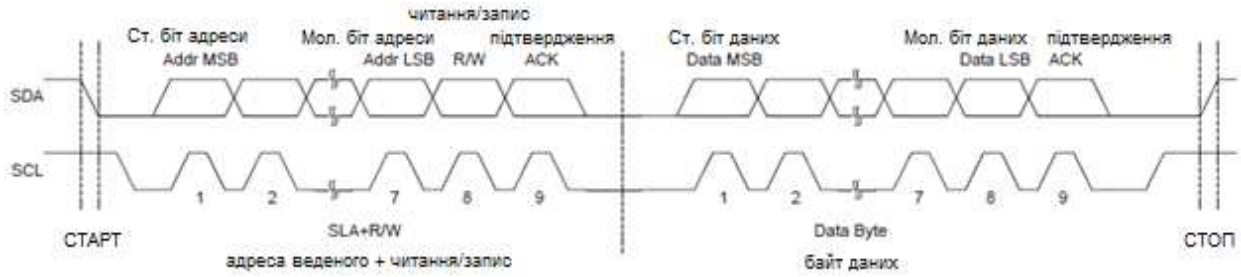

Рисунок 2.5 – Часова діаграма типового передавання даних по I2C

2.3 Регістри модуля TWI

Функціональна схема модуля TWI показана на рис. 2.6.

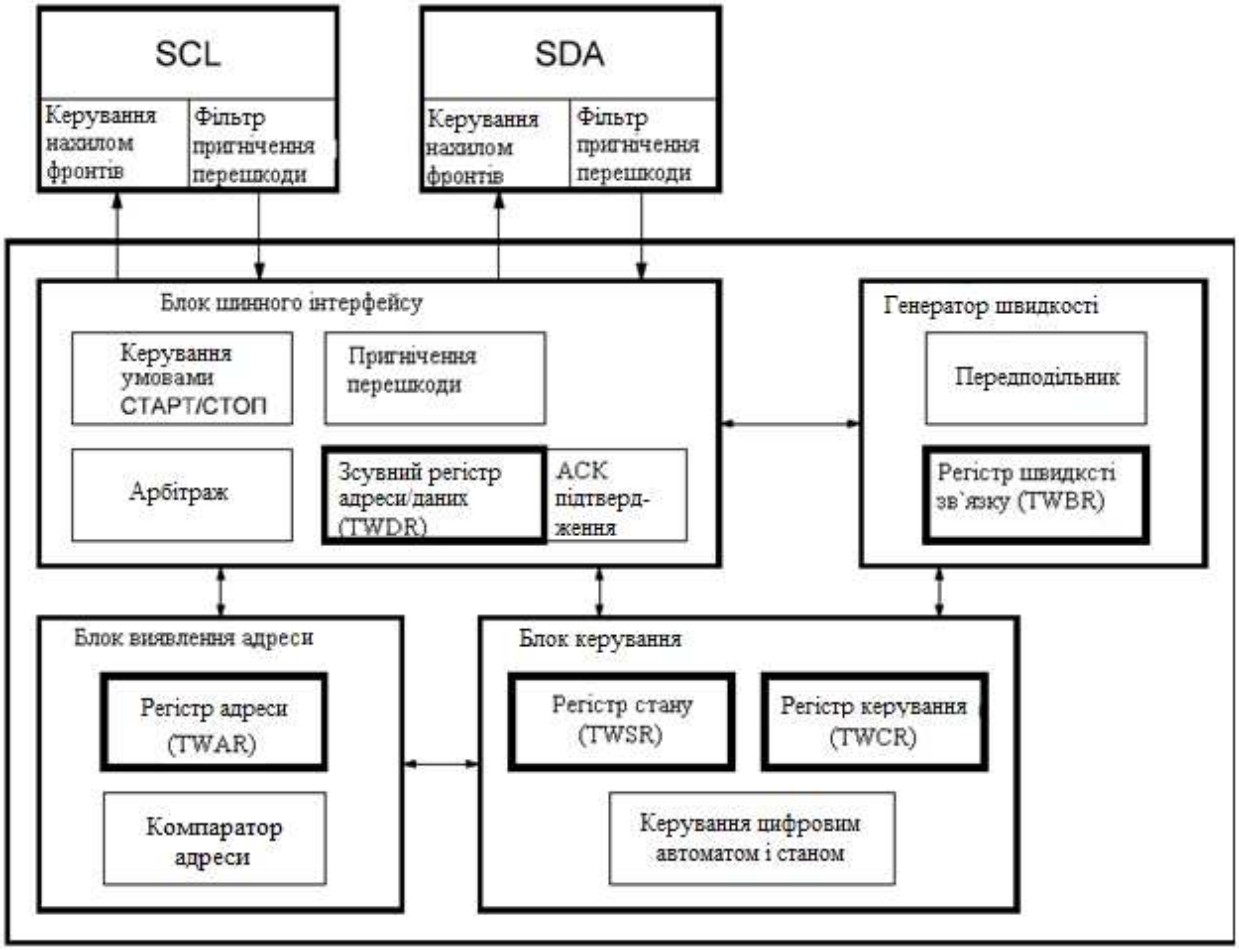

Рисунок 2.6 – Функціональна схема модуля TWI

У виділених жирною лінією прямокутниках знаходяться регістри настройки модуля I2C у мікроконтролері AVR. Далі описано регістри модуля I2C МК ATmega16. В інших мікроконтролерах можливі невеликі відмінності.

Регістр швидкості передавання TWBR (TWI Bit Rate Register)

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Біт 7: 0 – біти цього регістра визначають частоту роботи модуля I2C. Частота також залежить від тактової частоти роботи мікроконтролера, і значення в молодших бітах (TWPS0, TWPS1) регістра TWSR.

Частота SCL сигналу, тактова частота мікроконтролера і значення регістрів TWBR і TWSR пов'язані наступним співвідношенням:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}.$$

Регістр даних TWDR (TWI Data Register)

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

Біти 7:0 цього регістра зберігають дані, які ми або хочемо передати відомому, або отримали від ведучого.

Регістр адреси TWAR (TWI Address Register)

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

Біти 7:1 зберігають значення адреси, за якою буде відгукуватися мікроконтролер, перебуваючи в режимі веденого.

Біт 0 – біт дозволу на відгук під час спільних викликів.

Статусний регістр TWSR (TWI Status Register)

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

Біти 7:3 містять статусний код. Біти доступні тільки для читання, статусний код установлюється TWI модулем апаратно після виконання різних операцій. Наприклад, формування стана СТАРТ, передавання пакета даних і так далі. За значенням статусного коду можна судити про результат операції. Виконалася вона успішно чи ні.

Біт 2 – біт зарезервований і читається як 0.

Біт 1:0 впливають на частоту SCL (залежність наочна у формулі для обчислення SCL)

Регістр керування TWCR (TWI Control Register)

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Біт 7 – біт прапорця переривання TWI модуля. Цей біт установлюється апаратно, коли TWI модуль завершує поточну операцію (формування стана СТАРТ, передавання адресного пакета і так далі). При цьому, якщо установлено біт глобального дозволу переривань (біт I регістра SREG) і дозволені переривання TWI модуля, то викликається відповідний обробник.

Біт TWINT очищається програмно, записом одиниці. При виконанні обробника переривання цей біт не скидається апаратно, як в інших модулях. Скидання прапорця TWINT запускає роботу TWI модуля, тому всі операції з регістром даних, статусу або адреси, повинні бути виконані до його скидання. Поки біт TWINT установлено, на лінії SCL утримується низький рівень.

Біт 6 – біт дозволу підтвердження. Якщо біт TWEA установлений в 1, TWI модуль формує сигнал підтвердження (ACK) коли це потрібно. А потрібно це в трьох випадках: провідний або ведений пристрій отримав байт даних, ведений пристрій отримав загальний виклик, ведений пристрій отримав свою адресу.

Біт 5 – прапорець стану СТАРТ. Коли цей біт установлюється в 1, TWI модуль перевіряє чи не зайнята шина і формує стан СТАРТ. Якщо шина зайнята, він буде чекати появи на ній стана СТОП і після цього видасть стан СТАРТ. Біт TWSTA повинен бути очищений програмно, коли стан СТАРТ передано.

Біт 4 – прапорець стану СТОП. Коли цей біт установлюється в 1 в режимі ведучого, TWI модуль видає на шину стан СТОП і скидає цей біт. У режимі веденого установлення цього біта може використовуватися для відновлення після помилки. При цьому стан СТОП не формується, але модуль TWI повертається до початкового неадресованого стану.

Біт 3 – прапорець конфлікту запису. Цей прапор установлюється апаратно, коли виконується запис у регістр даних (TWDR) за низького значення біта TWINT. Тобто коли TWI модуль вже виконує якісь операції. Прапорець TWWC скидається апаратно, коли запис у регістр даних виконується при установленому прапорці переривання TWINT.

Біт 2 – біт дозволу роботи TWI модуля. Коли біт TWEN установлюється в 1, TWI модуль включається і бере на себе керування виводами SCL і SDA. Коли біт TWEN скидається, TWI модуль вимикається.

Біт 1 – біт зарезервований і читається як 0.

Біт 0 – дозвіл переривання TWI модуля. Коли біт TWIE і біт I регістра SREG установлені в 1 – переривання модуля TWI дозволені. Переривання будуть викликатися при установленні біта TWINT.

Рис. 2.7 описує порядок передавання даних по шині I2C більш докладно.

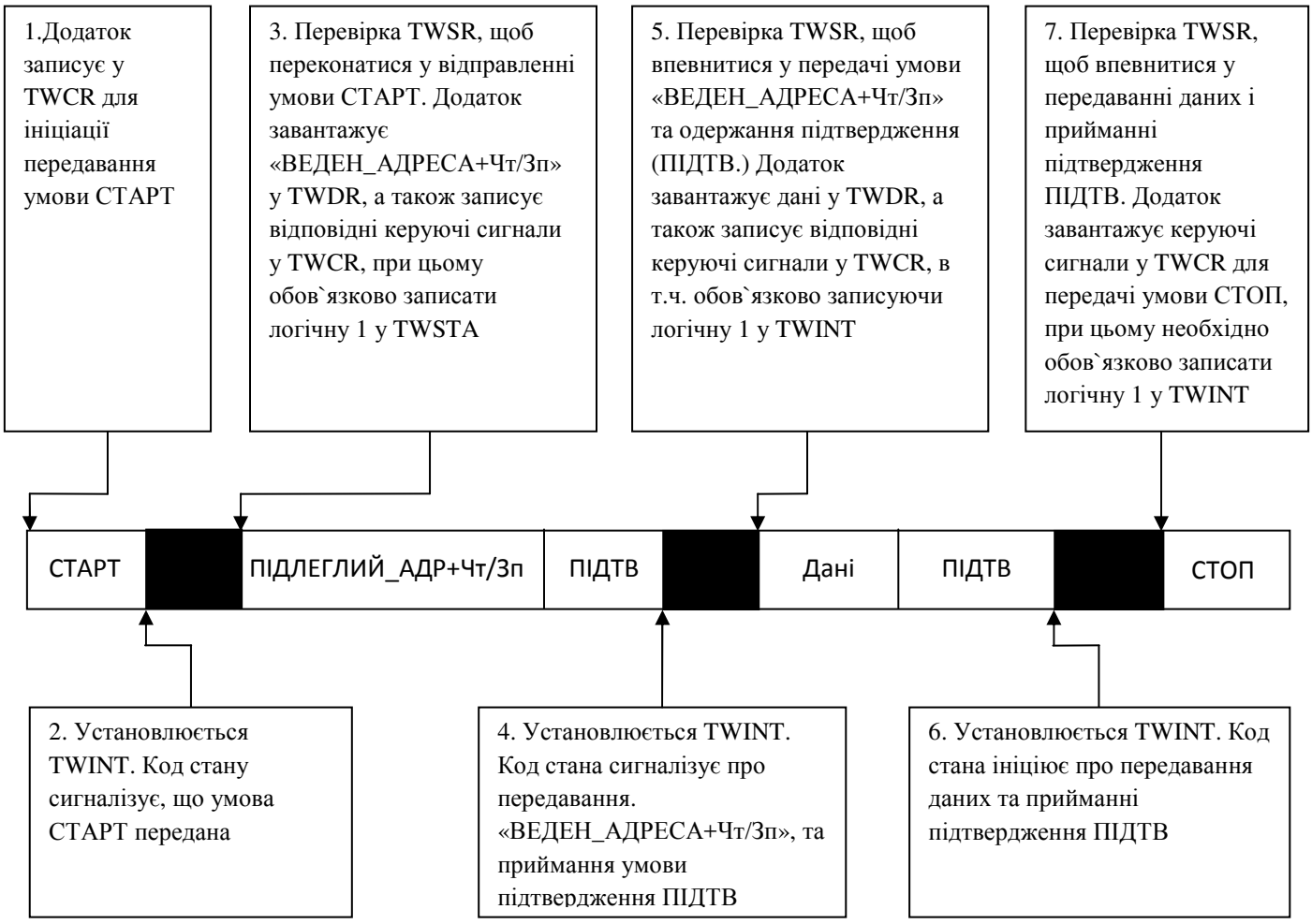

Рисунок 2.7 – Типове передавання даних (докладніше)

2.4 Приклад програми

Наступний код (дві програми для різних МК ATmega16) здійснює передавання даних від провідного передавача до веденого приймача (схема на рис. 2.8). Кожен прийнятий байт даних відображається на восьми світлодіодах.

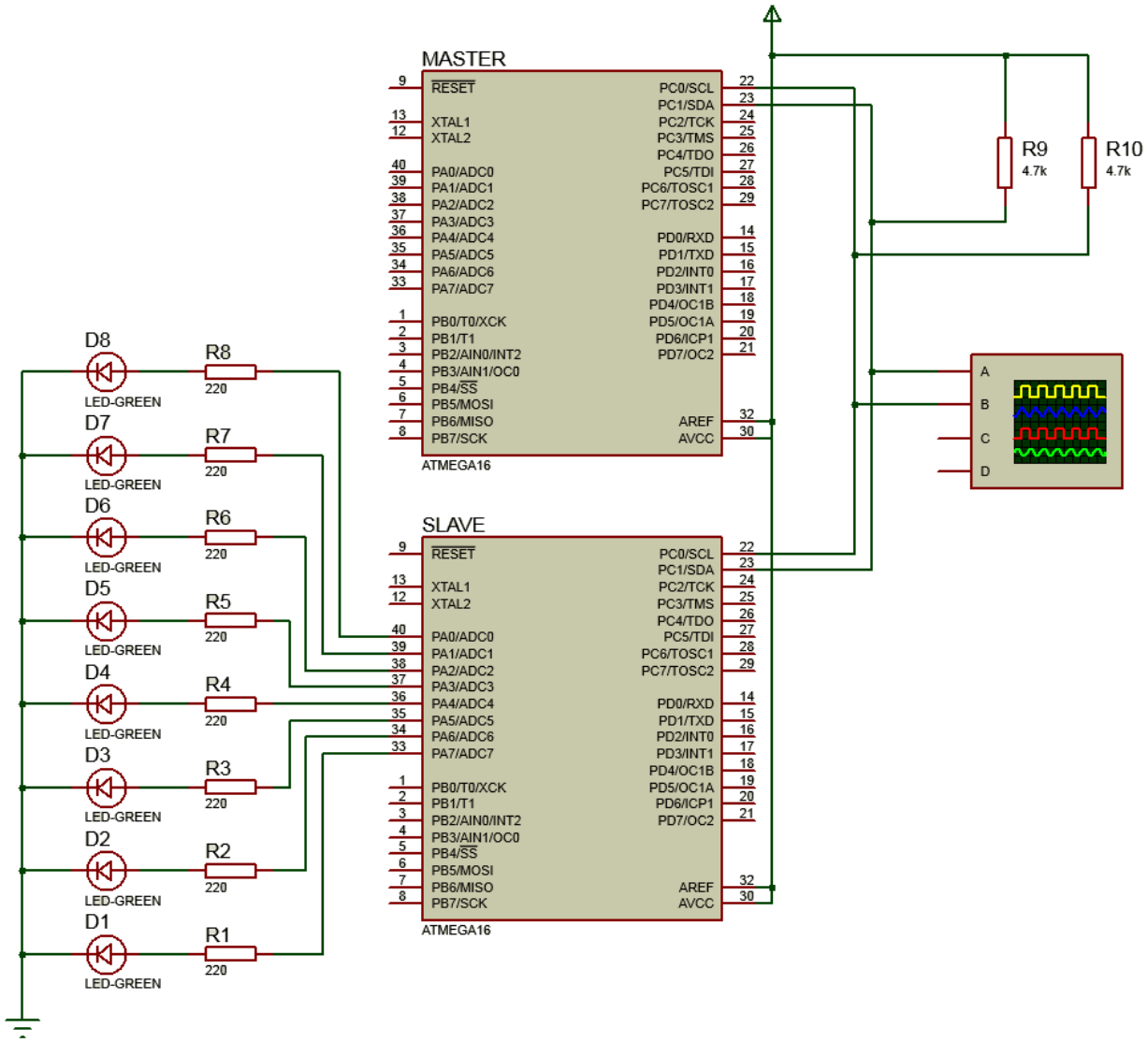

Рисунок 2.8 – Схема лабораторної роботи

Текст програми для ведучого МК (Master):

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

#define F_CPU 1000000UL
#define RTC_ADDR 0b11111111 //адреса ведучого
#define SLA_R   RTC_ADDR|0b00000001 //адрес + біт читання
#define SLA_W   RTC_ADDR&0b11111110 //адрес + біт запису

// відправлення команди СТАРТ
void I2C_StartCondition(void)
{
        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT))); //очікування установлення біта TWIN
}
```

```
// передавання СТОП
void I2C_StopCondition(void)
{
	TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
}

//передавання байта
void I2C_SendByte(unsigned char c)
{
	TWDR = c; //завантаження значення в регістр даних
	TWCR = (1<<TWINT)|(1<<TWEN); //початок передавання байта даних
	while (!(TWCR & (1<<TWINT))); //очікування установлення біта TWIN
}

//ініціалізація модуля I2C як передавача
void I2C_Init (void)
{
	TWBR=0xFF; //швидкість передавання
}

//передавання SLA_W + байт даних
void I2C_SendPocket (unsigned char value)
{
	I2C_StartCondition(); //умова СТАРТ
	I2C_SendByte(SLA_W); //передача адреси пристрою і біта запису
	I2C_SendByte(value); //передавання байта даних
	I2C_StopCondition(); //умова СТОП
}

int main(void)
{
	I2C_Init() //ініціалізація модуля
	char i=0; //змінна для передавання
	while (1);
{
	I2C_SendPacket (i);
	_delay_ms(1000);
	i+=1;
}
```

Ведений (Slave):

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

```
#include <avr/interrupt.h>
//адреса на яку відповідає приймач
#define MY_ADDR 0b11111111

//читання статусу
unsigned char I2C_GetStatus(void)
{
	unsigned char status; //змінна зберігання статусу
	status = TWSR & 0xF8; //маска
	return status;
}

//ініціалізація модуля I2C як приймача
void I2C_Init (void)
{
	TWAR=MY_ADDR; //адреса для звертання
	TWCR=(1<<TWEA) //дозвіл підтвердження
	|(1<<TWEN) //включення модуля TWI
	|(1<<TWIE); // дозвіл переривання
}

ISR(TWI_vect)
{
	if(I2C_GetStatus()==0x80) //прийшли дані від ведучого
	PORTA=TWDR; //вихід значення на порт A
	TWCR|=(1<<TWEN); //скидання прапорця
}

int main(void)
{
	I2C_Init(); //ініціалізація модуля
	DDRA=0xFF; //налаштування порту на вихід
	sei(); //вирішення глобальних переривань
	while(1);
}
```

3 Контрольні питання

1. Опишіть схему підключення пристроїв по шині I2C.
2. Яка максимальна кількість пристроїв на шині I2C?
3. Швидкість передавання даних по I2C.
4. Формат адресного пакета I2C.
5. Формат передавання даних по I2C.
6. Регістри модуля TWI (I2C).
7. Які переривання підтримує модуль?

4 Домашнє завдання

1. Вивчити розд. 2 (Основні положення).
2. Підготувати протокол лабораторної роботи.
3. Відповісти на контрольні питання.

5 Лабораторне завдання

1. Скласти схему рис. 2.6 у середовищі Proteus.
2. Створити проект в AVR Studio для програми з п. 2.4.
3. Скомпілювати програму й отримати файл з розширенням .hex у папці \Debug проекту.
4. Установити отриманий файл в якості прошивки мікроконтролера на схемі в Proteus і перевірити працездатність схеми.
5. Замалювати характер показів віртуального осцилографа.

6 Оформлення протоколу

Протокол виконання лабораторної роботи включає: тему, мету роботи, виконання домашнього і лабораторного завдання, висновки.

7 Список рекомендованої літератури

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL – [5-е изд.] / Евстифеев А.В. – М.: Издательский дом «Додэка-XXI», 2008. – 560 с.