МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

«ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ»

для студентів усіх форм навчання
напряму підготовки 6.050803 «Акустотехніка»

Електронне видання

ЗАТВЕРДЖЕНО
кафедрою МІРЕС
протокол № 4 від 4.11.2017

Харків 2017

Методичні вказівки до лабораторних робіт з дисципліни «Основи мікропроцесорної техніки» для студентів усіх форм навчання напряму підготовки 6.050803 «Акустотехніка» / Упоряд. О.В. Зубков. – Харків: ХНУРЕ, 2017. – 32 с.

Упорядник О.В. Зубков

Рецензент:

ЗМІСТ

ЗАГАЛЬНІ ВКАЗІВКИ

У сучасному світі високих технологій і електроніки мікропроцесори відіграють важливу роль. Вони застосовуються повсюдно – починаючи від дитячих іграшок і закінчуючи складними електронними комплексами й системами. У відмінності від DSP-процесорів вони виконують більш скромні функції, їх тактові частоти обмежені, як правило, 10-100 МГц, вони є однопотоковими, обсяг оперативної пам'яті не перевищує 100 кбайт і т.і. Однак, комбінація невисокої ціни із широкими функціональними можливостями, дозволяє використовувати мікропроцесори в багатьох електронних пристроях і призводить до їх повсюдного поширення.

Метою лабораторного практикума з дисципліни “ОМкТ” є навчання студентів програмуванню сучасних мікропроцесорів Atmega16, симуляції їх роботи за допомогою сучасних середовищ налагодження програмного забезпечення мікропроцесорів.

Відповідно до вимог до дисципліни в даних методичних вказівках описано п’ять лабораторних робіт, тематика яких охоплює всі основні розділи курсу. Усі лабораторні роботи виконуються з використанням комп'ютерів і середовища моделювання. На комп'ютері студенти працюють у спеціальних середовищах IAR Embedded Workbench4.16 і Proteus, призначених для програмування і налагодження розроблювальних програм. Схеми цифрових пристроїв з використанням мікропроцесорів розробляються у програмному симуляторі Proteus. Роботи виконуються фронтальним способом. Їх послідовність підібрана таким чином, що в кожній наступній роботі студенти вирішують більш складне і об'ємне завдання, але при цьому вони частково можуть використовувати програмні методи та розв'язки (наприклад, підпрограми), які розроблялися в попередніх роботах. Це дозволяє більш глибоко засвоїти досліджуваний матеріал і закріпити отримані навички програмування avr-контролерів.

Порядок виконання всіх лабораторних робіт однаковий і складається з наступних пунктів:

- розробити алгоритм, програму для контролера Atmega16 і ввести її у вихідний файл середовища IAR Embedded Workbench4.16;
- запустити компілятор, при необхідності виправити виявлені помилки;
- за допомогою програмного емулятора середовища Proteus переконатися в правильності роботи розробленої програми. Для цього виконати програму в покроковому режимі, контролюючи при цьому стан усіх задіяних регістрів і портів вводу-виводу;
- робота в лабораторії може вважатися закінченої тільки після пред'явлення викладачеві правильно зібраною схемою пристрою та правильно працюючим запрограмованим avr-контролером.

Опис кожної лабораторної роботи містить наступні основні підрозділи: мета роботи, методичні вказівки по організації самостійної роботи студентів і

формулювання завдання, яке повинен розв'язати студент у ході виконання роботи, контрольні запитання і завдання.

Кожна лабораторна робота розрахована на одне чотиригодинне заняття.

Перед початком циклу лабораторних занять кожний студент зобов'язаний:

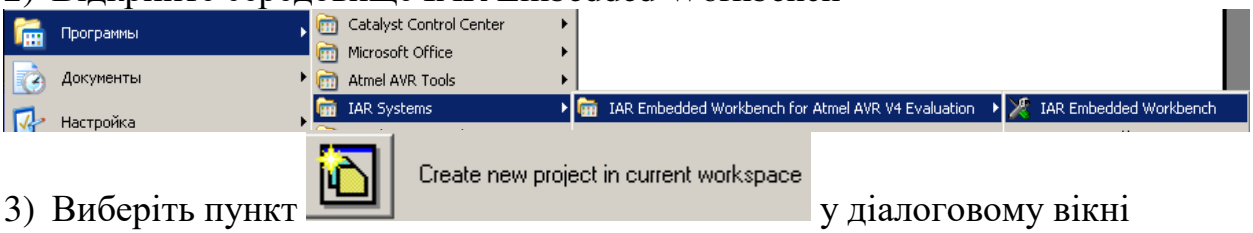
засвоїти «Правила техніки безпеки», викладені нижче і розписатися в груповому лабораторному журналі, пред'явити звіт про попередню роботу. Залік по виконаних раніше роботах здається протягом часу, виділеного для лабораторних занять.

При роботі в лабораторії врахувати, що для живлення ПЕОМ використовується мережа змінного струму напругою 220В, і хоча передбачений ряд заходів, що забезпечують безпечну роботу в лабораторії, можлива поразка електричним струмом у випадку порушення правил поведінки в лабораторії і техніки безпеки.

ОПИС СЕРЕДОВИЩ ПРОГРАМУВАННЯ IAR Embedded Workbench, PROTEUS 7

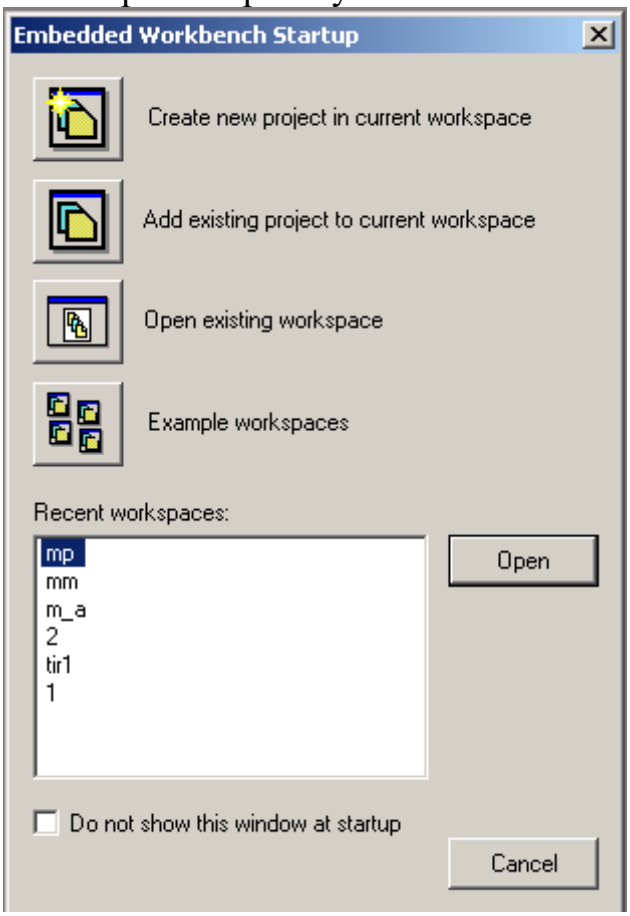
Створення проекту в середовищі IAR Embedded Workbench.

1) Створіть на диску D порожню папку з назвою англійською мовою
2) Відкрийте середовище IAR Embedded Workbench

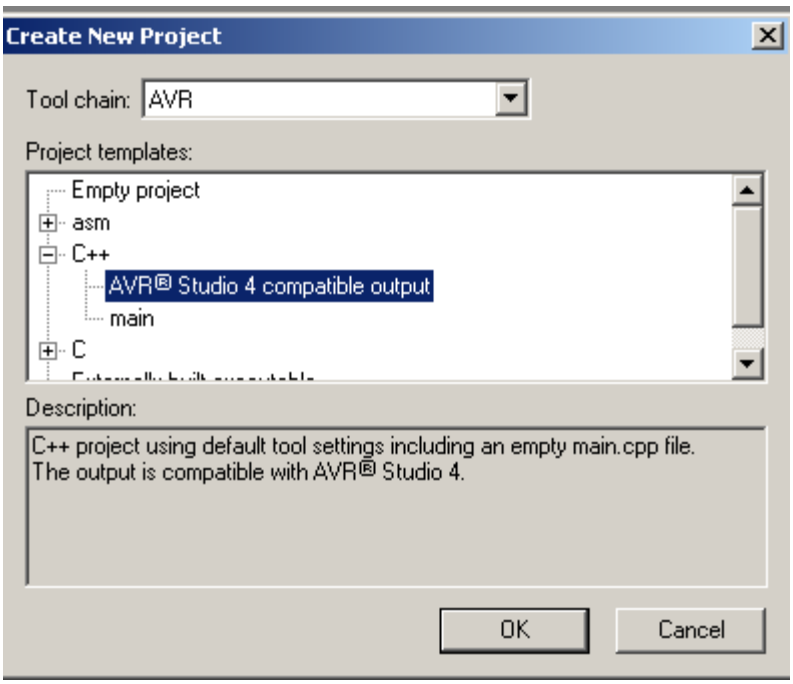


3) Виберіть пункт

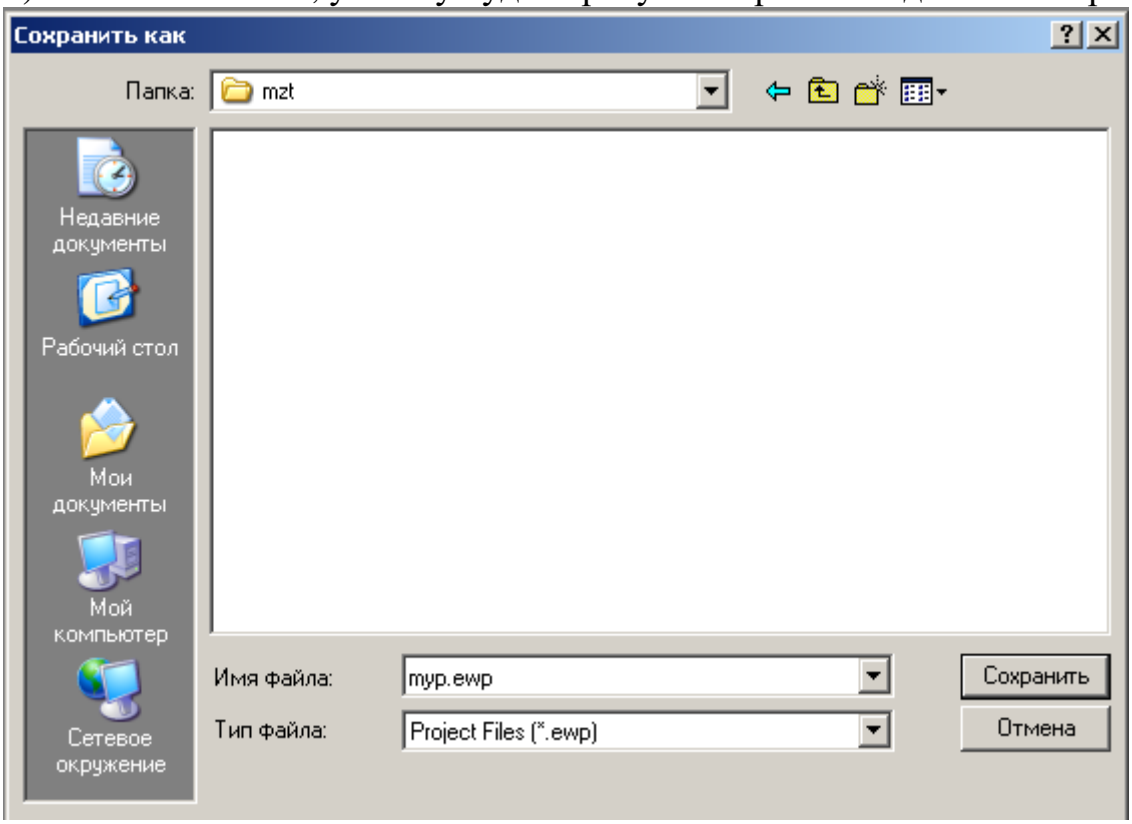
у діалоговому вікні створення проекту і натисніть кнопку **Open**



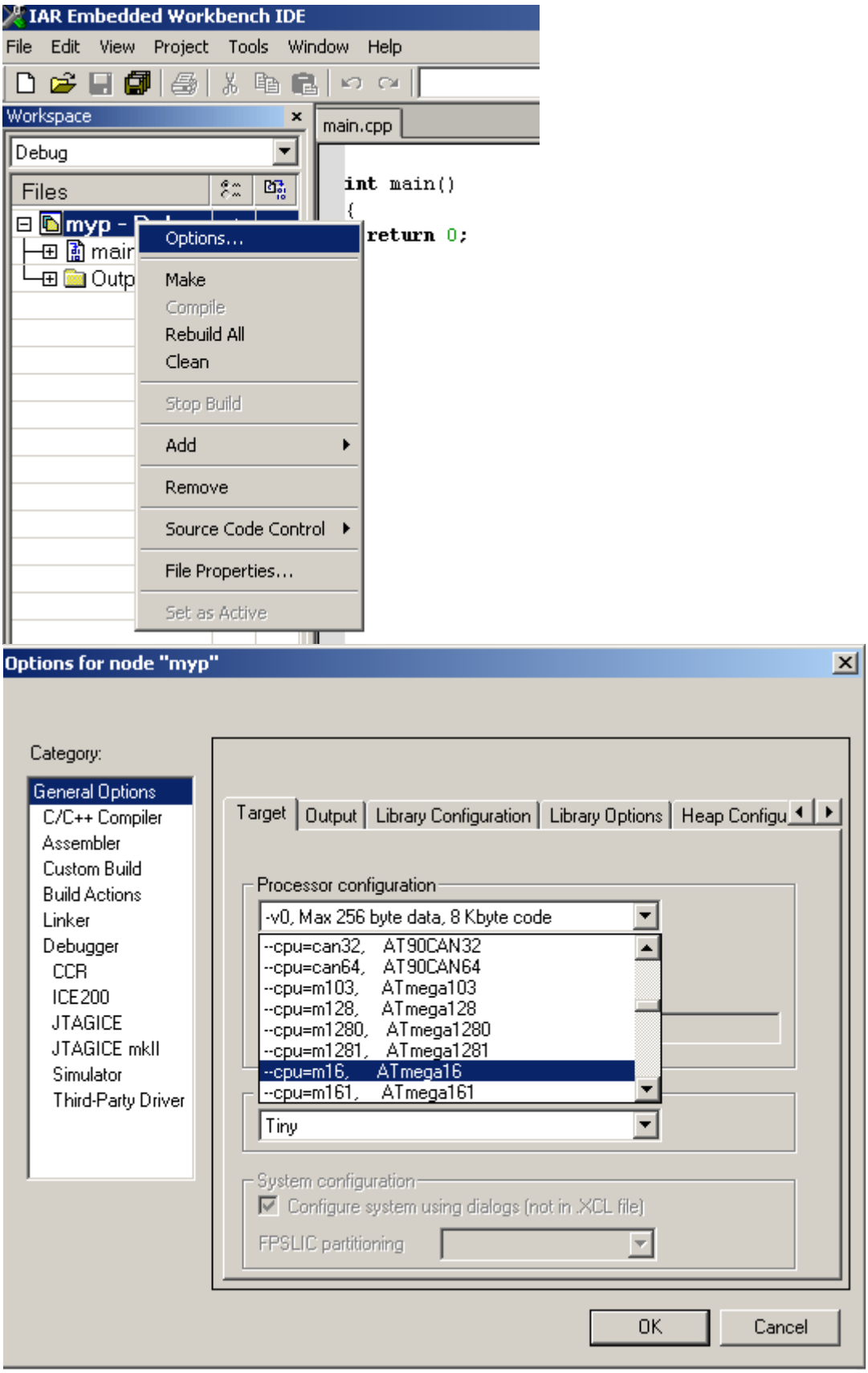
4) Визначте тип проекту в наступному діалоговому вікні і натисніть кнопку ОК



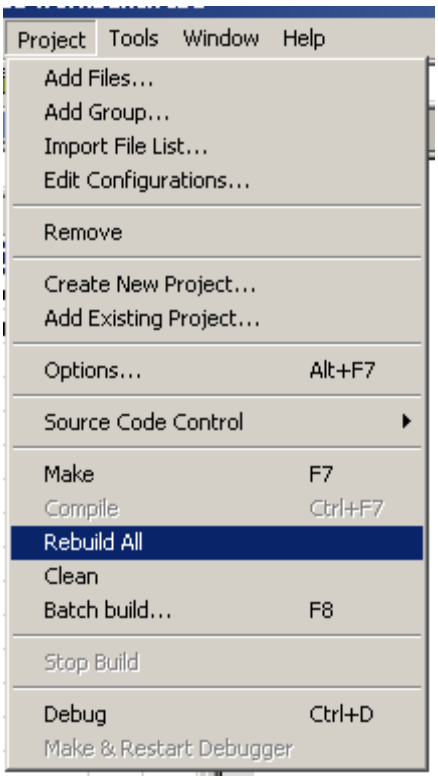
5) Вкажіть каталог, у якому буде перебувати проект і задайте ім'я проекту



6) Задайте тип процесора

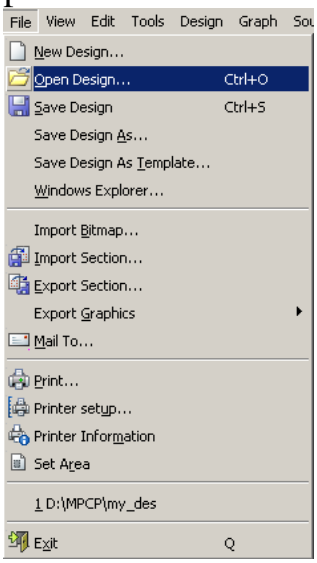


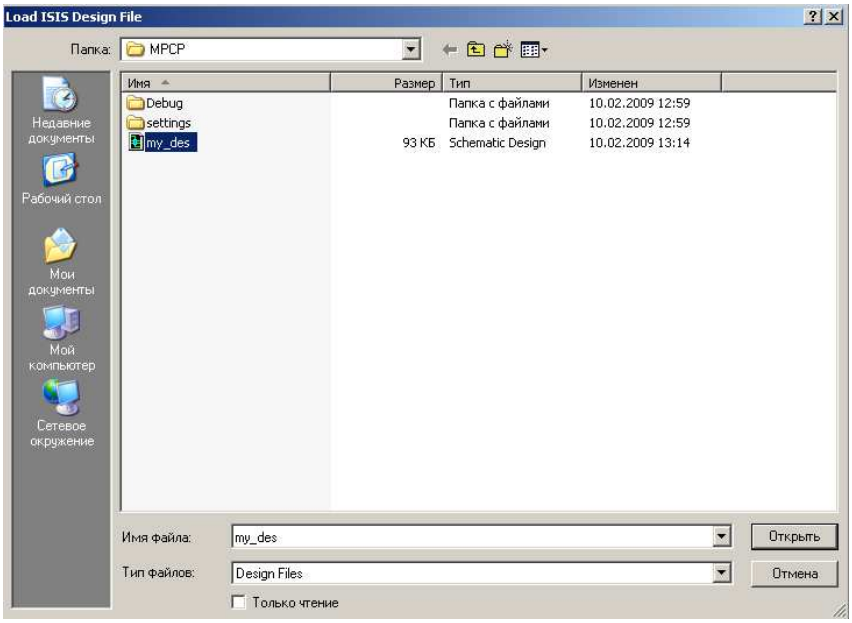
7) Напишіть програму. Збережете програму і відкомпілюйте її. Для цього виберіть пункт меню



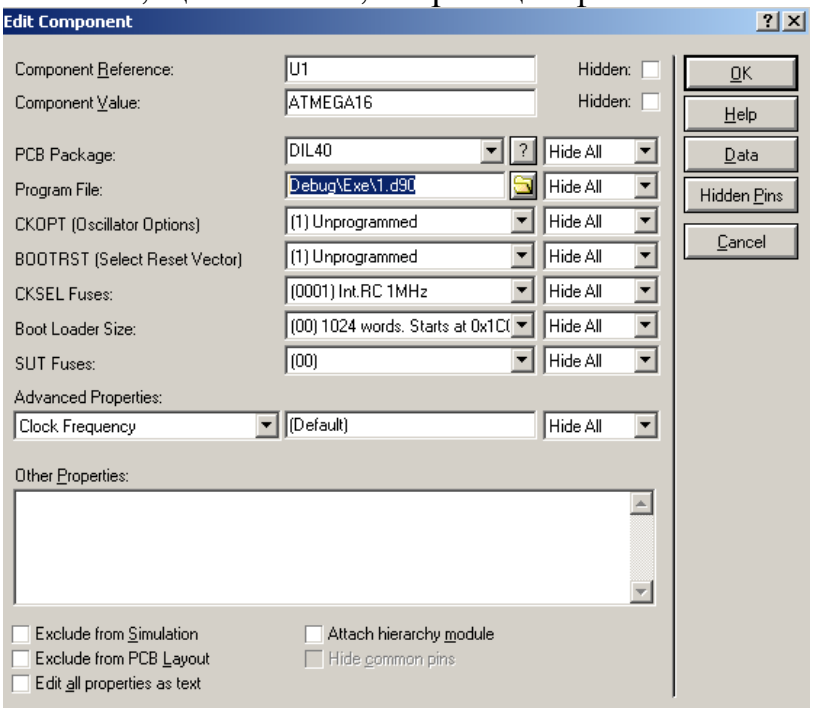
8) Після компіляції в створеному Вами каталозі з'явиться папка Debug, а в ній папка Exe. У цій папці перебуває файл проекту з расширением.d90, який може бути завантажений у середовище Proteus 7.

9) Відкрийте середовище Proteus 7. (Пуск/Програми/Proteus 7 professional/ISIS 7 Professional). Відкрийте створений викладачем файл





11) Підключите до мікроконтролера симуляційний файл із розширенням *.d90, який був раніше створений у середовищі IAR. Для цього двічі клацніть маніпулятором «миша» по мікроконтролеру і у меню, що з'явиться, оберіть цей файл



Підтвердіть вибір.

12) Перевірте роботу програми в складі принципової схеми. Для цього підключите осцилограф у потрібні крапки принципової схеми і натисніть кнопку Play у нижньому лівому куті середовища розробки.

ЛАБОРАТОРНА РОБОТА №1

ВИВЧЕННЯ АРХІТЕКТУРИ І ПРИНЦИПІВ РОБОТИ ПОРТІВ ВВОДУ-ВИВОДУ AVR-МІКРОКОНТРОЛЕРІВ

Мета роботи: освоїти апаратні та програмні принципи конфігурування і роботи з портами вводу-виводу, зовнішніми перериваннями.

1.1 Теоретична частина

При підготовці до лабораторної роботи слід детально вивчити і проробити теми № 1-5 по конспекту лекцій, а також рекомендовану по даних темах літературу[1, с. 81-82, 137-145б 197-205; 2, с 320-328]. Особливу увага звернути на функціональне призначення системних регістрів PORTD, PORTB, PORTC, DDRD, DDRB, DDRC, PINB, PIND, PINC, GICR, MCUCR мікроконтролера Atmega16, які визначають конфігурацію його зовнішніх портів і зовнішніх переривань.

Кожен з 4-х портів (A, B, C, D) мікроконтролера Atmega16 складається із трьох регістрів. Регістри DDRA, DDRB, DDRC, DDRD визначають які з виводів контролера будуть входами, а які виходами. Якщо в якісь біти цих регістрів записані «1», то відповідні їм виводи будуть виходами. Якщо в якісь біти цих регістрів записані «0», то відповідні їм виводи будуть входами. На виводи, які сконфігуровани, як виходи – можна видавати логічний «0» або «1». Для цього необхідно змінювати стани відповідних біт у регістрах PORTA, PORTB, PORTC, PORTD.

Зчитувати стани логічних сигналів з виводів, які сконфігуровани як входи, можна опитуючи стан відповідних біт у регістрах PINA, PINB, PINC, PIND.

При зміні стану сигналів на виходах і конфігуруванні виводів можна працювати з регістром цілком або з його окремими розрядами. Наприклад

DDRA=15; //Конфігуруємо виводи PA0, PA1, PA2, PA3 як виходи

PORTA=3; //Видаємо на виходи PA0, PA1 логічну «1», а на інші логічний «0»

Для зміни сигналу на одному з виводів використовуються функції:

SETBIT(Ім'я_порту, номер_біта) - встановлює біт порту з номером *номер_біта* в стан «1».

CLEARBIT(Ім'я_порту, номер_біта) - скидає біт порту з номером *номер_біта* в стан «0»

Приклад

SETBIT(PORTB, 2); встановлення 2-го біта порту B у стан логічної «1»

При роботі із входами опитати сигнали на входах можна зчитавши стан вхідного регістру цілком

X= PINB; //Зчитуємо сигнали із усіх входів порту B

або один із входів функцією

TESTBIT(ім'я_регістру, номер_біта в регістрі);

Приклад

```
if (TESTBIT(PIND,0))
    {
       //Блок операторів
    }
```

Для організації тимчасових затримок можна використовувати 2 метода. Якщо точність тимчасової затримки не відіграє важливої ролі, то таку затримку можна організувати програмно за допомогою циклів for або while. При цьому затримку підбирають експериментально, умовно вважаючись, що на виконання однієї команди потрібно 2 такту процесора. Процес організації програмних затримок спрощується при використанні бібліотеки delay.h, у якій реалізовані функції секундних, мікросекундних і мілісекундних затримок. Для використання цієї бібліотеки її потрібно скопіювати в папку із проектом і підключити в програмі, як і інші бібліотеки. Після компіляції проекту, відкриваємо з IAR бібліотеку delay.h і змінюємо в ній значення тактової частоти (значення параметра XTALL, що задається в Мгц). У цій бібліотеці є 3 функції затримки delay_us(затримка в мкс),delay_ms(затримка в мс), delay_s(затримка в с), що дозволяють задати потрібну затримку. Приклад виклику функції затримки на 20 мс.

```
delay_ms(20);
```

Організація затримок з високою точністю виконується за допомогою таймера. Нижче приводиться опис 8-мі розрядного таймера.

Нульовий таймер-лічильник може працювати в режимах підрахунку зовнішніх, внутрішніх імпульсів і генератора ШИМ сигналів.

Налаштування 0-го таймер-лічильника здійснюються бітами регістру TCCR0

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

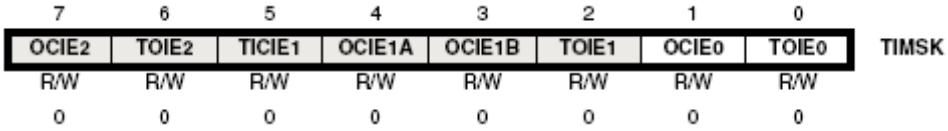
Біти CS00,CS01,CS02 задають коефіцієнт передділення частоти внутрішніх імпульсів, а також фронт сигналу зовнішніх імпульсів.

CS02,CS01,CS00	Налаштування
000	Таймер виключений
001	Коефіцієнт передділення частоти внутрішніх імпульсів 1
010	Коефіцієнт передділення частоти внутрішніх імпульсів 8
011	Коефіцієнт передділення частоти внутрішніх імпульсів 64
100	Коефіцієнт передділення частоти внутрішніх імпульсів 256
101	Коефіцієнт передділення частоти внутрішніх імпульсів 1024
110	Підраховуються зовнішні імпульси на вході T0 по спадаючому фронту
111	Підраховуються зовнішні імпульси на вході T0 по наростаючому фронту

WGM01, WGM00 визначають режим ШІМ модулятора, а також режим роботи лічильника

WGM01,WGM00	Налаштування
00	Нормальний режим
01	ШІМ з корекцією фази
10	Автоскидання таймера при досягненні граничного значення
11	Швидкий ШІМ (при рахуванні до переповнення)

Налаштування переривань 0-го таймер-лічильника здійснюється бітами регістру TIMSK



Біт OCIE0 при установці в 1 дозволяє переривання по збігові з порогом.
Біт TOIE0 при установці в 1 дозволяє переривання по переповненню.

При роботі в режимі збіга з порогом рахування значення порогу записується в регістр OCR0.

1.2 Завдання та порядок його виконання

При виконанні даної лабораторної роботи в середовищі Proteus 7 необхідно зібрати принципову електричну схему, розробити алгоритм, написати та налагодити програму пристрою, що має у своєму складі 4 кнопки і 8 світлодиодів. Перші 3 світлодиода повинні світитися або гаснути при натисканні або відпусканні відповідних їм кнопок. Даний процес повинен відбуватися в реальному масштабі часу. Вибір використовуваних для розв'язку даного завдання портів мікроконтролера, тактової частоти процесора і часових параметрів визначається даними таблиці 1.

Таблиця 1 – Вихідні дані до виконання лабораторної роботи

№ бригади	1	2	3	4	5	6	7	8
Входи з підключеними кнопками	PB0… PB3	PC2… PC5	PD1... PD4	PA0… PA3	PB2… PB5	PC4… PC7	PD3… PD6	PA2… PA5
Виходи	Порт C	Порт D	Порт A	Порт B	Порт A	Порт B	Порт C	Порт D
Тактова частота, МГц	2	1	8	6	10	16	4	12
Інтервал часу t,с	0,3	0,4	0,5	0,8	1	1,2	1,4	1,6

При натисканні на 4-у кнопку, необхідно виконати послідовне включення одного зі світлодиодів («вогонь, що біжить») через інтервал часу t. Програму необхідно реалізувати в 2-х варіантах: 1) інтервал часу t відраховується з використанням бібліотеки delay.h; 2) інтервал часу відраховується таймером.

Принципова схема підключення кнопок і світлодиодів до мікроконтролера наведено на рисунку 1.1.

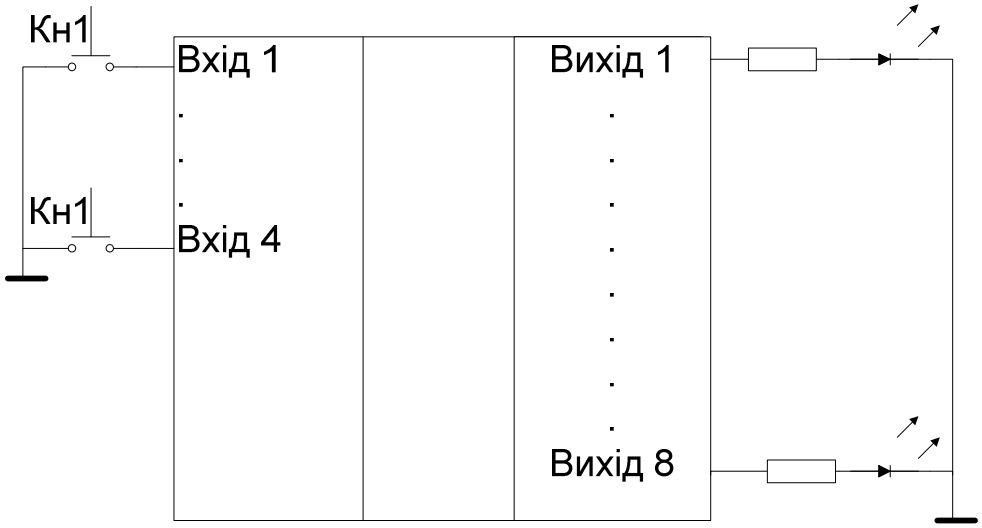


Рисунок 1.1 – Схема електрична принципова підключення кнопок і світлодиодів до мікроконтролера.

Відповідно до цієї схеми, якщо кнопка не натиснута, то на вході мікроконтролера сигнал логічної одиниці, а якщо кнопка натиснута, то – логічного нуля.

Порядок виконання роботи

1) Скласти алгоритм розв'язання поставленого завдання

2) Розробити програму мовою С++ для мікроконтролера Atmega16 за розробленим алгоритмом

3) Скомпілювати програму в середовищі IAR Embedded Workbench4.16

4) Завантажити скомпільований програмний код у середовищі Proteus 7. Для цього слід відкрити файл D://Mpcp/my_des.dsn і підключити до мікроконтролера скомпільований раніше файл з розширенням .d90. У відкритому файлі слід зібрати схему відповідну номеру бригади та рисунку 1.2.

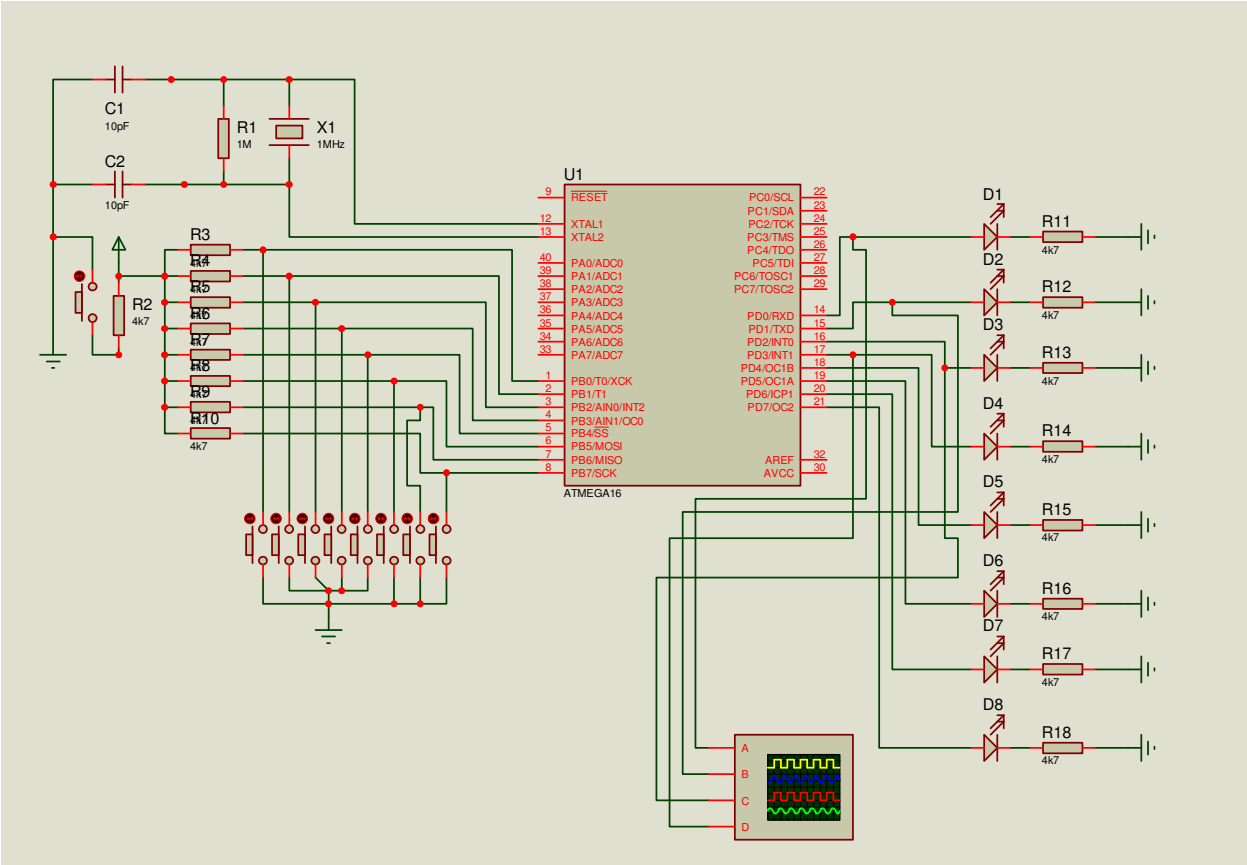


Рисунок 1.2 – Схема лабораторного макета в середовищі Proteus 7

Перевірте роботу програми в складі принципової схеми.

1.3 Зміст звіту

Звіт повинен містити:

1) Тему і мету роботи
2) Схему електричну принципову лабораторного макета
3) Лістинг розробленої програми мовою С++
4) Виводи по роботі

1.4 Перелік контрольних запитань

1) Загальна архітектура avr-контролерів
2) Внутрішня структура портів вводу-виводу
3) Системні регістри, що керують портами вводу-виводу
4) Організація пам'яті програм
5) Організація пам'яті даних

ЛАБОРАТОРНА РОБОТА № 2

ВИВЧЕННЯ ПРИНЦИПІВ ПОБУДОВИ МАТРИЦІ КЛАВІАТУРИ

Мета роботи: освоїти апаратні і програмні принципи організації матриці клавіатури.

2.1 Теоретична частина

При підготовці до лабораторної роботи слід детально вивчити і проробити теоретичний матеріал, що стосується основних схемотехнічних і програмних методів побудови матриці клавіатури ([1, с. 97-104, 206-212; 2, с 329-384] і теми № 1-6 по конспекту лекцій). Слід підкреслити, що застосування матриці клавіатури в реальному пристрої дозволяє суттєво скоротити необхідна кількість задіяних виводів мікроконтролера.

Приклад електричної принципової схеми підключення клавіатури до мікроконтролера наведено на рисунку 2.1. У даній схемі виводи PB0 і PB1 є виходами, а виводи PB2 і PB3 – входами. Тобто стовпці клавіатури підключені до виходів, а рядка до входів з підтягуючими резисторами. При натисканні однієї із клавіш відповідний вихід процесора виявляється з'єднаним із входом. Якщо тільки на цьому виході процесора зараз сигнал логічного «0», а на інших виходах сигнал логічної «1», то тільки на відповідному натиснутій кнопці вході виявляється сигнал логічного «0». У такому випадку ми однозначно визначаємо натиснуту кнопку.

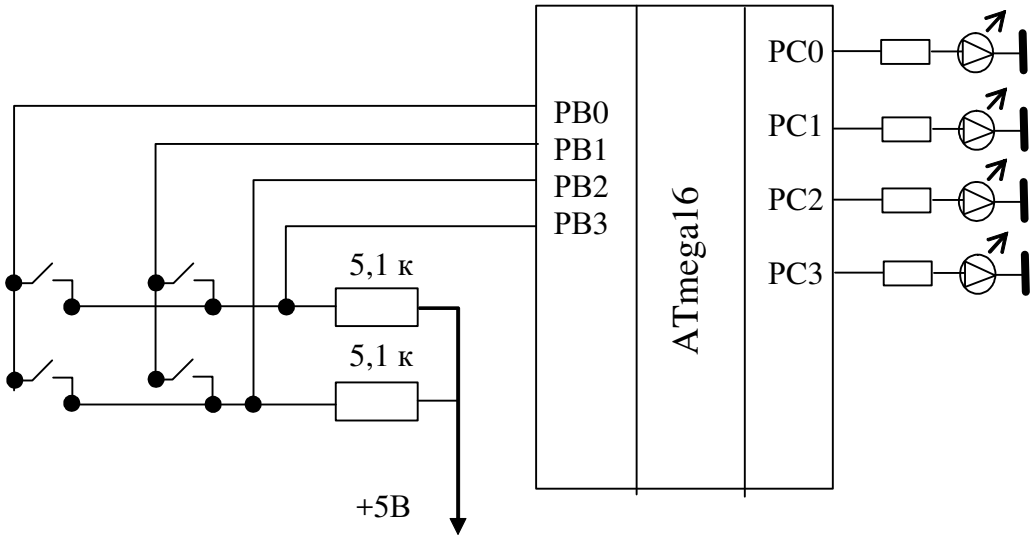


Рисунок 2.1 – Схема електрична принципова підключення клавіатури до мікроконтролера

У ході виконання лабораторної роботи в середовищі Proteus 7 необхідно зібрати електричну схему, розробити алгоритм і програму для пристрою на базі контролера Atmega16, що має у своєму складі матрицю клавіатури, організовану у вигляді 2x2. Кожній клавіші поставлений у відповідність світлодиод, підключений до порту X, який повинен засвітитися

при натисканні на відповідну клавішу. При повторному натисканні на ту ж клавішу світлодиод повинен гаснути. При опитуванні клавіатури необхідно організувати програмну боротьбу з деренчаням контактів.

При роботі із кнопками слід завжди пам'ятати, що при замиканні механічної кнопки виникає деренчання контактів, який триває від 10 мс до 100 мс, залежно від якості контактів. Якщо не боротися із цим ефектом, то замість одного натискання кнопки Ви зареєструєте серію з великої кількості натискань. Найефективнішим засобом боротьби є програмний метод – метод опитування кнопок 4 рази через рівні інтервали часу (протягом 40-100 мс).

Алгоритм програмної обробки клавіатури

1. Визначаємо кількість клавіш у клавіатурі і формат цієї клавіатури. Створюємо в програмі масив з довільним іменем і кількістю елементів у масиві рівним числу клавіш. Вважаємо, що кожній клавіші відповідає один з елементів масиву.
2. Будемо вважати, що стовпці клавіатури підключені до виходів контролера, а рядка – до входів. На всіх входах включені підтягуючи резистори, а на всі виходи подані логічні «1».
3. Запускаємо таймер процесора, який повинен відраховувати інтервал часу порядку 20мс. В оброблювачі переривання від цього таймера повинна перебувати програма обробки клавіатури.
4. В оброблювачі переривання від таймера пишемо програму почергового опитування стовпців клавіатури. Спочатку, подаємо логічний «0» на перший вихід (перший стовпець клавіатури). На інші виходи подається логічна «1». Опитуємо по черзі входи. Сигнали на них відповідають станам кнопок першого стовпця. Якщо на вході логічна «1» (кнопка не натиснута), то відповідний до цієї кнопки елемент масиву обнуляється. Якщо на вході логічний «0» (кнопка натиснута), то відповідний до цієї кнопки елемент масиву збільшується на 1. Далі, перевіряємо, чи досягло значення елементів масиву 4, тобто 4 опитування підряд кнопка зареєстрована натиснутої. Якщо так, то включаємо відповідний до цієї кнопки світлодиод. Інші світлодиоди гасимо.
5. Подаємо на другий вихід логічний «0» (на другий стовпець клавіатури), а на перший логічну «1». Опитуємо по черзі входи. Сигнали на них відповідають станам кнопок другого стовпця. Якщо на вході логічна «1» (кнопка не натиснута), то відповідний до цієї кнопки елемент масиву обнуляється. Якщо на вході логічний «0» (кнопка натиснута), то відповідний до цієї кнопки елемент масиву збільшується на 1. Далі, перевіряємо, чи досягло значення елементів масиву 4, тобто 4 опитування підряд кнопка зареєстрована натиснутої. Якщо так, то включаємо відповідний до цієї кнопки світлодиод. Інші с світлодиоди гасимо.

2.2 Завдання та порядок його виконання

Вихідні дані для написання програми наведено в таблиці 2.1

Таблиця 2.1 – Вихідні дані для виконання роботи

№ бригади	1	2	3	4	5	6	7	8
Входи	PB0, PB1	PD2, PD3	PB1, PB2	PD4, PD5	PA0, PA1	PC0, PC1	PD1, PD2	PC5, PC6
Виходи	PB5, PB6	PD6, PD7	PC3, PC4	PA1, PA2	PA6, PA7	PA0, PA1	PC1, PC2	PB4, PB5
Світлодиоди (порт)	C	A	D	C	B	B	A	D
Частота, МГц	4	1	2	8	2	4	1	10

Алгоритм опитування клавіатури наведено на рисунку 2.2

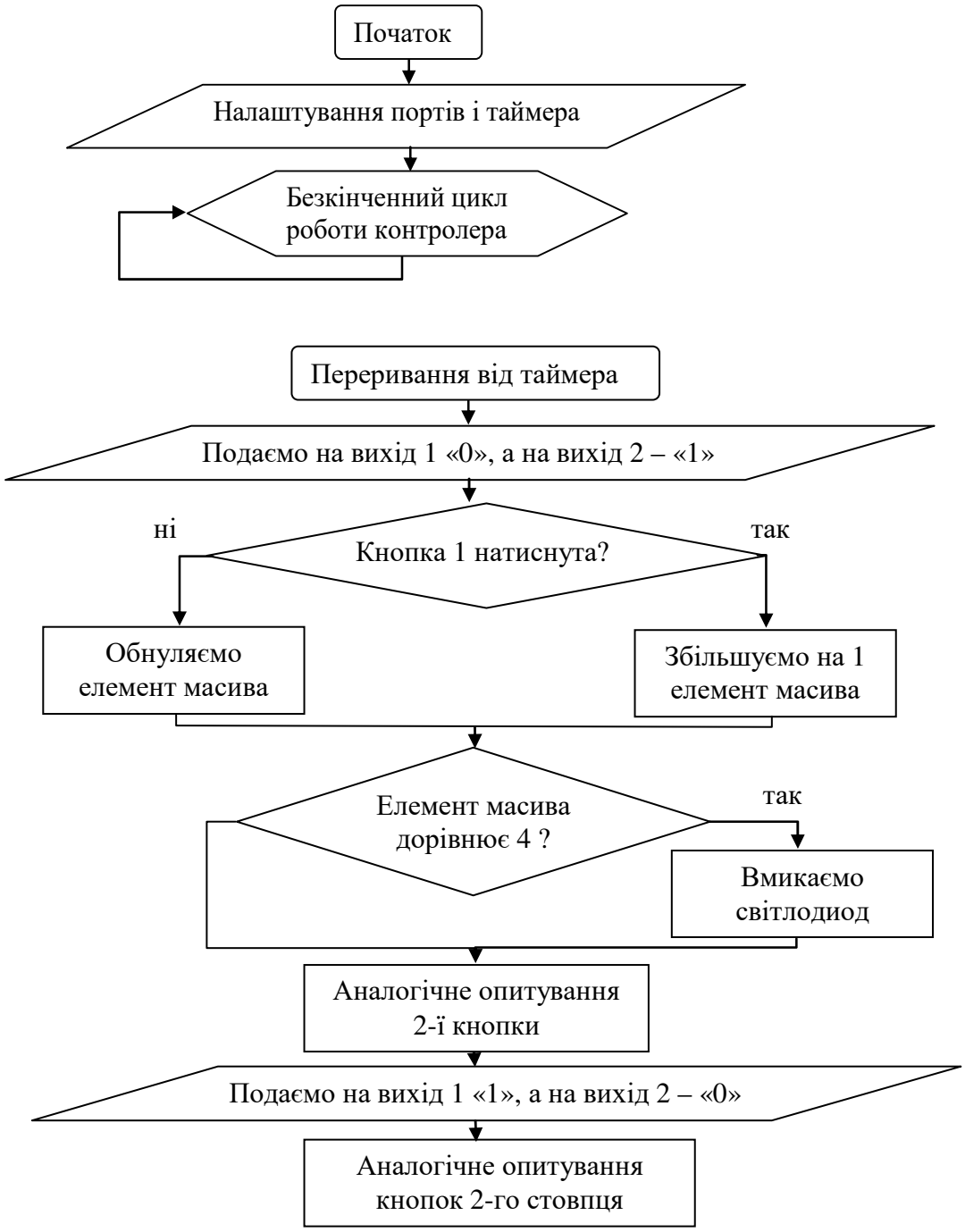


Рисунок 2.2 – Алгоритм опитування клавіатури

Порядок виконання роботи:
1) Скласти алгоритм розв'язку поставленого завдання.
2) Розробити програму мовою С++ для мікроконтролера Atmega16, що реалізує розроблений алгоритм.
3) Скомпілювати програму в середовищі IAR Embedded Workbench4.16.
4) Завантажити скомпільований програмний код у середовищі Proteus 7. Для цього слід відкрити файл D://Mpcp/klava.dsn і підключити до мікроконтролера скомпільований раніше файл з розширенням .d90. У відкритому файлі необхідно зібрати принципову схема клавіатури і світлодиодів, що підключені до мікропроцесору відповідно до варіанта завдання з таблиці 2.1.
5) Перевірити працездатність розробленої програми.

2.3 Зміст звіту

Звіт повинен містити:
1) Тему та мету роботи
2) Схему електричну принципову лабораторного макета
3) Лістинг розробленої програми мовою С++
4) Виводи по роботі

2.4 Перелік контрольних запитань

1) Поясніть, як конфігурується 1-й таймер-лічильник при рахуванні до переповнення або до збігу із пороговим значенням?
2) Чи існує різниця між конфігуруванням 1-го таймер-лічильника при рахуванні до порогу А та до порогу В? Чи можна працювати до збігу з обома порогами?
3) Поясните, як конфігурується 0-й таймер-лічильник при рахуванні до переповнення або до збігу із пороговим значенням?
4) У чому сутність ефекту деренчання контактів і які існують фізичні та програмні методи боротьби із цим ефектом?
5) Поясніть алгоритм обробки клавіатури.
6) Навіщо необхідно фіксувати факт натискання кнопки при обробці клавіатури?

ЛАБОРАТОРНА РОБОТА №3

ВИВЧЕННЯ ПРИНЦИПІВ ПРИЙМАННЯ ПАКЕТІВ ДАНИХ

Мета роботи: вивчити зовнішні переривання і принципи приймання пакетів даних.

3.1 Теоретична частина

При підготовці до лабораторної роботи слід детально вивчити і проробити теми № 1-8 по конспекту лекцій, а також рекомендовану по даних темах літературу[1, с. 95-96; 2, с 308-319].

Схема електрична принципова лабораторної установки представлено на рисунку 3.1

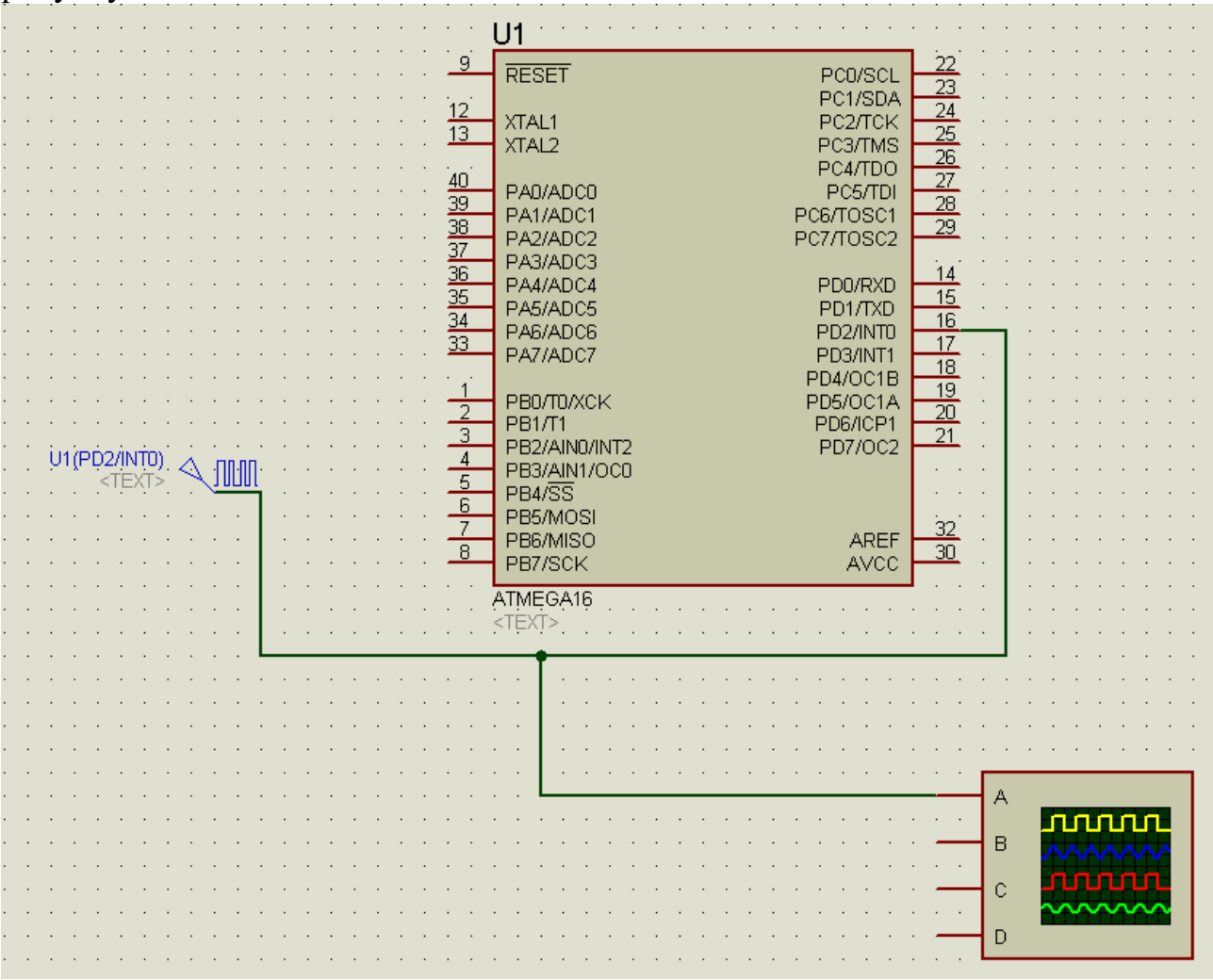
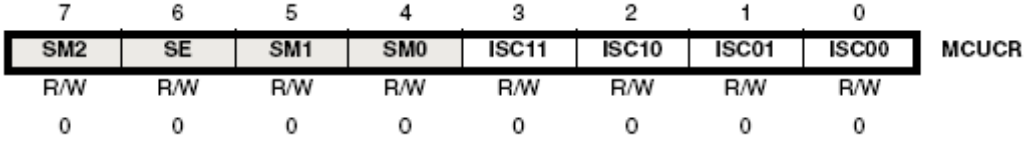


Рисунок 3.1 – Схема електрична принципова лабораторної установки

Зовнішні переривання дозволяються бітами регістру GICR

7	6	5	4	3	2	1	0	
INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
R/W	R/W	R/W	R	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

Біти INT0, INT1, INT2 – дозволяють відповідні переривання INT0, INT1, INT2.
Для визначення рівня сигналу або фронту, по якому відбувається переривання, використовуються біти регістру MCUCR



Біти ISC01, ISC00 задають ознака виникнення переривання INT0.

Стан ISC01, ISC00	Переривання
00	По низькому рівню вхідного сигналу
01	По зміні вхідного сигналу
10	По спадаючому фронту вхідного сигналу
11	По наростаючому фронту вхідного сигналу

Біти ISC11, ISC10 задають ознака виникнення переривання INT1.

Стан ISC11, ISC10	Переривання
00	По низькому рівню вхідного сигналу
01	По зміні вхідного сигналу
10	По спадаючому фронту вхідного сигналу
11	По наростаючому фронту вхідного сигналу

3.2 Завдання та порядок його виконання

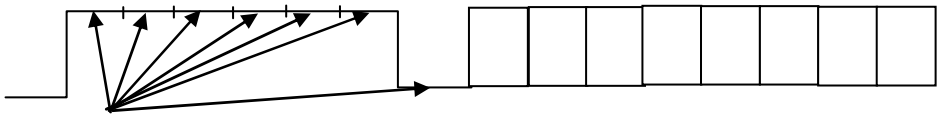
У модемному зв'язку та інших комунікаційних протоколах не підтримуваних досліджуваним контролером використовується пакетна передача даних. Пакет починається з унікальної послідовності біт (у лабораторній роботі це послідовність 01111110), далі передається корисна інформація. Початкова унікальна послідовність біт 01111110 призначена для того, щоб відрізнити прийняті дані від перешкод. Це означає, що якщо прийнята така послідовність біт, те можна продовжувати зчитування інформації, що приймається. Вихідні дані для виконання лабораторної роботи наступні

Параметр	№ бригади					
	1	2	3	4	5	6
Швидкість передачі	1 кбіт/с	2 кбіт/с	4 кбіт/с	4 кбіт/с	0,5 кбіт/с	8 кбіт/с
Тактова частота процесора	1МГц	4МГц	8МГц	2МГц	10МГц	16МГц

Кількість байт у пакеті – 2.

Приймання пакета виконується без використання вбудованих інтерфейсів, тому що реалізований формат передачі не підтримується жодним з таких інтерфейсів. Тому, алгоритм приймання має такий вигляд.

1) Цифровий сигнал подається на вхід PD2 (INT0). Необхідно налаштувати зовнішнє переривання INT0, щоб з його допомогою очікувати зміни логічного сигналу з «0» в «1» на вході;
2) при виникненні такої зміни в оброблювачі переривання від INT0 слід запустити таймер-лічильник на час, що дорівнює половині тривалості імпульсу. Це пов'язане з тим, що ухвалювати рішення щодо приймання логічного «0» або «1» необхідно посередині біта, щоб звести до мінімуму вплив завалених фронтів імпульсу на результат ухвалення рішення. Переривання від INT0 на час роботи таймера слід заборонити



Моменты принятия решения

3) Після відліку інтервалу в 0,5 тривалості біта слід переконфігурувати таймер, щоб він відраховував інтервали, рівні тривалості 1 біта. Щораз, коли поточне значення у таймері буде досягати порога, ми повинні зчитувати із входу INT0 стан логічного сигнала і записувати значення прийнятого біта на відповідну позицію змінної типу unsigned char. Коли будуть прийняті перші 7 біт ми повинні перевірити отриманий байт на відповідність преамбулі 01111110. Якщо відповідності немає, то це перешкода і робота таймера повинна бути припинена, а для входу INT0 знову дозволені переривання. Якщо прийнятий байт відповідає 01111110, то ми продовжуємо приймання ще 8 біт. По завершенню приймання цих 8 біт слід скопіювати отриманий байт даних у внутрішню змінну, заборонити роботу таймера і дозволити переривання від входу INT0, щоб знову очікувати приймання чергового пакета.

Порядок виконання роботи.

1) Скласти алгоритм розв'язку поставленого завдання.
2) Розробити програму мовою С++ для мікроконтролера Atmega16 за складеним алгоритмом.
3) Скомпілювати програму в середовищі IAR Embedded Workbench4.16.
4) Завантажити скомпільований програмний код у середовищі Proteus 7.4.

3.3 Зміст звіту

Звіт повинен містити:

1) Тему і мету роботи
2) Схему електричну принципову лабораторного макета
3) Лістинг розробленої програми мовою С++

4) Виводи по роботі

3.4 Перелік контрольних запитань

1) Поясніть, як конфігурується 1-й таймер-лічильник при рахуванні до переповнення або до збігу із пороговим значенням?
2) Поясніть, як конфігуруються зовнішні переривання по входах INT0,INT1?
3) Чи існує різниця між конфігурування зовнішніх переривань по входах INT0,INT1 та INT2?
4) Поясните, як конфігурується 0-й таймер-лічильник при рахунку до переповнення або до збігу із граничним значенням?
5) Поясните алгоритм асинхронного приймання пакета даних. Чому слід зчитувати біти даних по середині біта?

ЛАБОРАТОРНА РОБОТА №4

ВИВЧЕННЯ ПРОГРАМУВАННЯ АНАЛОГО-ЦИФРОВОГО ПЕРЕТВОРЮВАЧА МІКРОКОНТРОЛЕРА ATMEGA16 І ВЗАЄМОДІЇ З ІНДИКАТОРОМ НА БАЗІ КОНТРОЛЕРА HD44780.

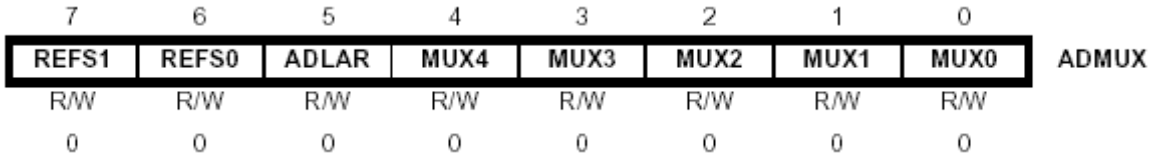
Мета роботи: вивчити принципи конфігурування та автоматичного запуску вбудованого в Atmega16 АЦП і вивчити принципи програмної взаємодії з жидкокриталічними символьними індикаторами на базі контролера HD44780.

4.1 Теоретична частина

4.1.1 Програмування вбудованого АЦП

При підготовці до лабораторної роботи слід детально вивчити і проробити теми № 1-11 по конспекту лекцій, а також рекомендовану по даних темах літературу[1, с. 106-107; 2, с 401-418].

Для наналаштування вбудованого АЦП використовується ряд регістрів процесора.

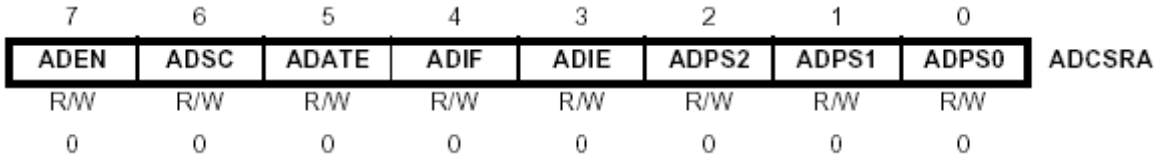


Регістр АЦП ADMUX дозволяє обрати вхід порту А, що буде підключений до АЦП, орієнтування результату і вибір опорної напруги. Якщо біт ADLAR встановлений в «1», то результат ліво-орієнтований. Опорна напруга від внутрішнього генератора задається встановленими в «1» бітами REFS1 і REFS0. Якщо обидва біта скинуті, то опорна напруга береться від входу AREF. У випадку, якщо REFS1=0, а REFS0=1, опорна напруга береться від AVCC із зовнішнім конденсатором, підключеним до AREF. Вибір входу порта А, що підключається до АЦП задається наступною таблицею.

Таблиця 4.1 – Конфігурування входу АЦП

Біти MUX4…MUX0	Вхід АЦП
00000	PA0
00001	PA1
00010	PA2
00011	PA3
00100	PA4
00101	PA5
00110	PA6
00111	PA7

Регістр контролю і статусу АЦП ADCSRA:



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

Біт ADEN=1 вмикає модуль АЦП. Запис одиниці в ADSC запускає цикл перетворення. У режимі "вільного польоту" запис одиниці запускає перше перетворення, наступні запускаються автоматично.

ADATE – вибір джерела початку перетворення сигналу на вході АЦП. Якщо цей біт дорівнює «1», то джерело початку перетворення задається конфігурацією регістру SFIOR.

ADIF - прапор переривання АЦП. Цей біт встановлюється в «1», коли АЦП завершене перетворення і у регістрах ADCL та ADCH перебувають актуальні дані. Цей прапор встановлюється навіть у тому випадку, якщо переривання заборонені. Це необхідно для випадку програмного опитування АЦП. Якщо використовуються переривання, то прапор скидається автоматично. Якщо використовується програмне опитування, то прапор може бути скинутий записом «1» у цей біт.

ADIE - якщо в цьому біті встановлена одиниця, і переривання дозволені глобально, то при закінченні перетворення буде виконаний перехід по вектору переривання від АЦП.

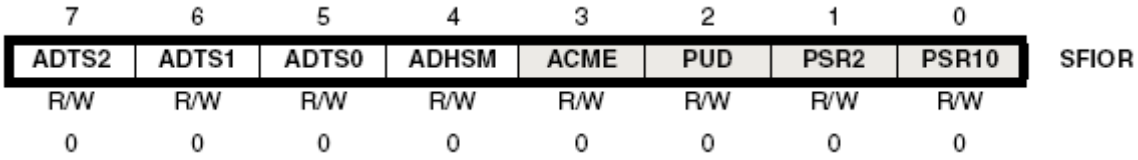
Для одержання максимально точного результату, модуль АЦП повинен тактуватися частотою в межах від 50 до 200 кГц. Якщо необхідний результат з точністю менш 10 біт, то можна використовувати частоту більше 200 кГц. Модуль АЦП містить дільник частоти, щоб одержувати потрібну тактову частоту для перетворення із частоти процесора.

Біти ADPS2..0 задають коефіцієнти преддільника частоти.

Таблиця 4.2 – Конфігурування біт ADPS2..0

ADPS2	ADPS1	ADPS0	Коефіцієнт преддільника частоти
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

Для конфігурування АЦП у режимі автоматичного запуску використовується регістр SFIOR



7	6	5	4	3	2	1	0	
ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10	SFIOR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Призначення біт регістра SFIOR наведено в таблиці 4.3

Таблиця 4.3 – Конфігурування біт регістра SFIOR

ADTS2…ADTS0	Режим
000	Звичайний пуск
001	По перериванню від аналогового компаратора
010	По зовнішньому перериванню INT0
011	По збігу з порогом 0-го Т/Л
100	По переповненню 0-го Т/Л
101	По збігу з порогом В 1-го Т/Л
110	По переповненню 1-го Т/Л
111	По захвату 1-го Т/Л

4.1.2 Теоретичні основи керування індикаторами на базі контролера HD44780.

При підключенні LCD на базі HD44780 до AVR мікроконтролеру є два режима шини даних —8 біт і 4 біта. У восьмибітному режимі простіше завантажувати байти — не потрібно зрушувати байт, але в 4-бітному скорочується число використовуваних виводів контролера на 4. За замовчуванням індикатор сконфігурован на роботу з 4-х бітною шиною.

- Виводи DB7…DB0 це шина даних/адреси.
- E — стробуючий вхід. Зміною напруги на цій лінії ми даємо зрозуміти дисплею що потрібно забирати/віддавати дані з/на шину даних.
- RW — визначає в якому напрямку передаються дані. Якщо 1 — то на читання з дисплея, якщо 0 те на запис у дисплей.

RS — визначає що в нас передається, команда (RS=0) або дані (RS=1).
DDRAM — пам'ять дисплея. Усе, що запишеться в DDRAM буде виведено на екран. Тобто, наприклад, якщо завантажити код 0×31 — на екрані відобразиться символ «1» тому, що 0x31 – це ASCII код цифри 1.
Схемне підключення дисплея має вигляд

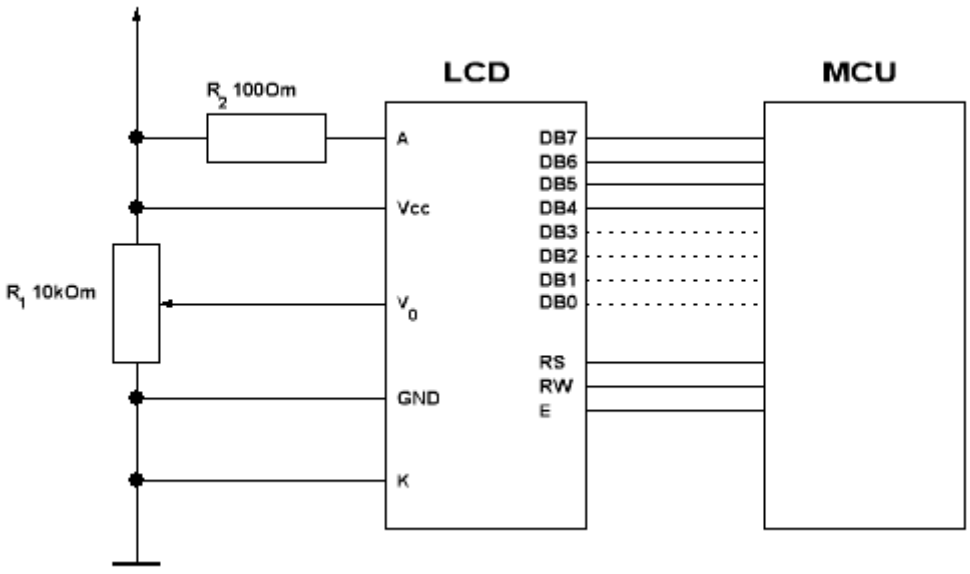


Рисунок 4.1 – Схемне підключення дисплея з HD44780

Система команд. Про те, що передається команда контролеру дисплея повідомляє сигнал на вході **RS**=0. Сама команда складається зі старшого біта, що визначає, за що відповідає дана команда і бітов параметрів, що вказують контролеру HD44780, що зробити.

Таблиця 4.4 – Таблиця команд HD44780

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Значення
0	0	0	0	0	0	0	1	Очищення екрана. Лічильник адреси на 0 позицію DDRAM
0	0	0	0	0	0	1	-	Адресація на DDRAM скидання зрушень, Лічильник адреси на 0
0	0	0	0	0	1	I/D	S	Налаштування зрушення екрана і курсору
0	0	0	0	1	D	C	B	Налаштування режиму відображення
0	0	0	1	S/C	R/L	-	-	Зрушення курсору або екрана, залежно від біт команди
0	0	1	DL	N	F	-	-	Вибір числа ліній, ширини шини і розміру символу
0	1	AG	AG	AG	AG	AG	AG	Перемкнути адресацію на SGRAM і задати адреса в SGRAM
1	AD	AD	AD	AD	AD	AD	AD	Перемкнути адресацію на DDRAM і задати адреса в DDRAM

Пояснимо призначення окремих біт:

I/D — інкремент або декремент лічильника адреси. По дефолту завантажен «0» — декремент. Тобто кожний наступний байт буде записаний в n-1 комірку пам’яті. Якщо завантажити «1» — буде інкремент.

- **S** - зрушення екрана, якщо завантажили «1», то з кожним новим символом буде зрушуватися вікно екрана, поки не досягнемо кінця DDRAM.
- **D** — включити дисплей. Якщо завантажити туди «0», то зображення зникне, а в цей час можемо створювати у відеопам'яті користувацькі символи. Щоб з'явилось зображення на екрані, необхідно записати «1» у цей біт.
- **C** - включити курсор у вигляді прочерку (у біт завантажуємо «1»).
- **B** — зробити курсор у вигляді миготливого чорного квадрата;
- **S/C** зрушення курсору або екрана. Якщо завантажити «0», то зрушується курсор. Якщо «1», то екран. По одному разу за команду;

- **R/L** — визначає напрямок зрушення курсору і екрана: 0 — вліво, 1 — вправо;
- **D/L** — біт визначальний ширину шини даних: 1-8 біт, 0-4 біта;
- **N** — число рядків: 0 — один рядок, 1 — два рядки;
- **F** - розмір символу: 0 — 5x8 крапок, 1 — 5x10 крапок (зустрічається вкрай рідко);
- **AG** - адреса в пам'яті CGRAM;
- **AD** — адреса в пам'яті DDRAM.

4.1.3 Алгоритм читання/запису в LCD контролер HD44780

При вмиканні живлення індикатор необхідно ініціалізувати. Ініціалізація виконується за алгоритмом на рисунку 4.2

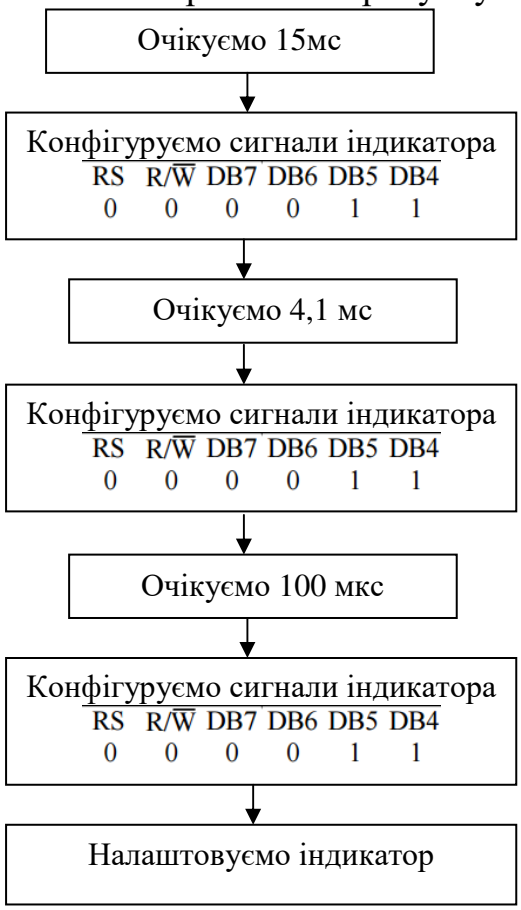


Рисунок 4.2 – Алгоритм ініціалізації індикатора WH1602

Напрямок, а також команда/дані визначаються лініями RS та RW, а читання та запис здійснюється по переходу строба (лінія E) з «1» в «0». При налаштуванні індикатора та виводі інформації необхідно реалізувати наступні алгоритми.

Запис команди в 4-х бітному режимі

Очікування готовності

1. RS=0 (команда);
2. RW=0 (запис);
3. E=1;

4. Завантажуємо в порт 4 старших біта коду команди;
5. Пауза 1,53мс;
6. E=0;
7. E=1;
8. Завантажуємо в порт 4 молодших біта коду команди;
9. E=0.

Запис байта в 4-х бітному режимі:

1. E=1;
2. Пауза 1,53мс;
3. Завантажуємо в порт старшу тетраду;
4. E=0;
5. Пауза;
6. E=1;
7. Пауза 1,53мс;
8. Завантажуємо в порт молодшу тетраду;
9. E=0.

4.2 Завдання та порядок його виконання

У лабораторній роботі за допомогою вбудованого в контролер АЦП необхідно виміряти напругу, що знімається з дільника, аналогічно тому, як показано в схемі на рисунку 4.3

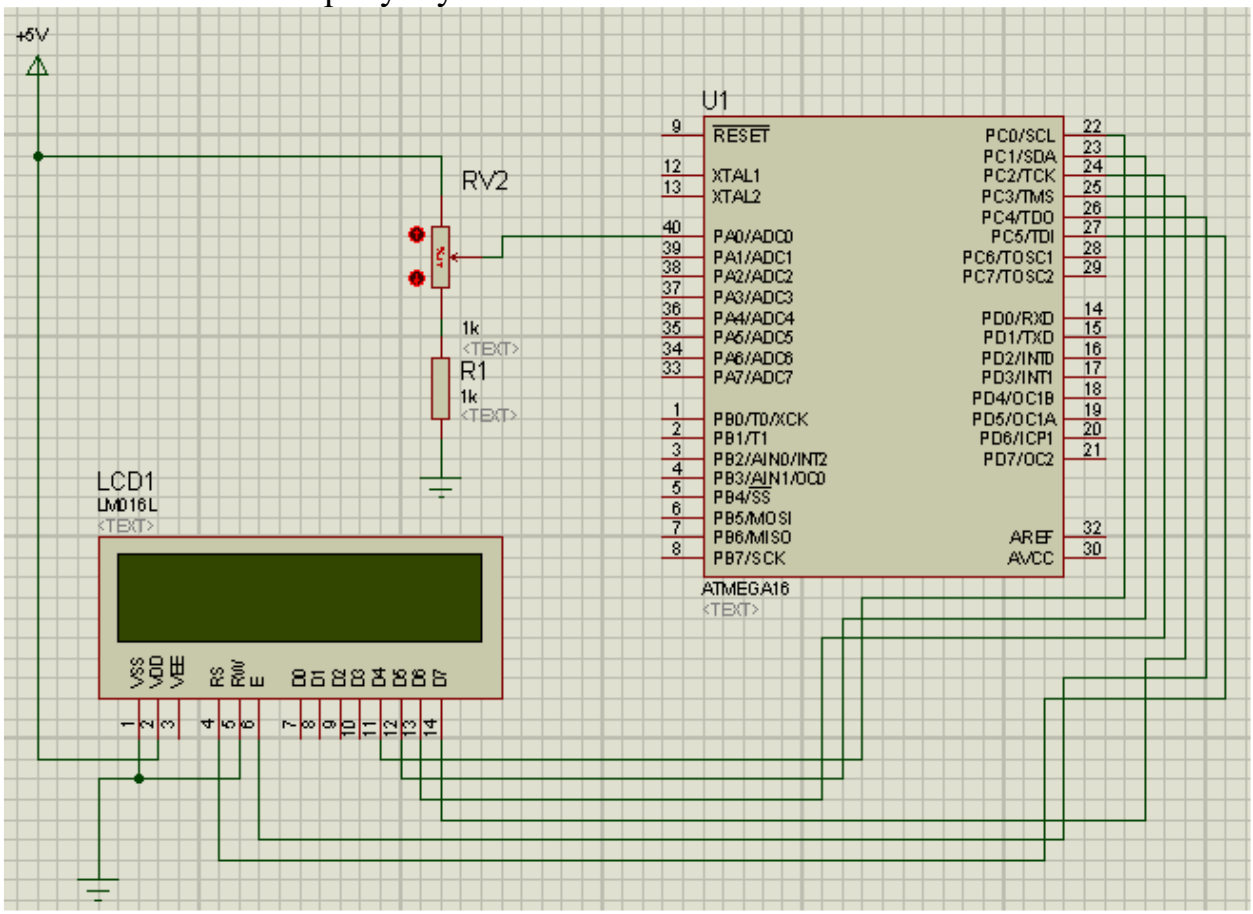


Рисунок 4.3 – Схема підключення індикатора

Результат вимірювання з точністю до десятих часток вольта необхідно видати на індикатор. Вихідні дані для написання програми наведені в таблиці 4.5

Таблиця 4.5 – Вихідні дані

№ бригади	1	2	3	4	5	6
Вхід АЦП	PA0	PA1	PA2	PA3	PA4	PA5
Частота дискретизації, Гц	10	15	20	25	30	35
Частота процесора, МГц	2	4	8	10	16	4
Порт підключення індикатора	PB	PC	PD	PB	PC	PD

При роботі з індикатором слід використовувати готові функції для передачі команд в індикатор і написати власну функцію виводу символу на індикатор.

На початку програми оголошується порт, до якого підключається індикатор

```
#define LCD_PORT PORTC
#define LCD_DDR DDRC
```

Далі оголошуються функції по керуванню портом

```
void Cursorpos(char y,char x); //Функція установки курсору в задану позицію (y – номер рядка: 1 або 2; x – номер символу в рядку починаючи з 0)
void Clear(void); //Функція очищення дисплея
void Sendbyte(char byte); //Функція відправлення байта команди на дисплей
void Sendnibble(char byte); //Функція початкового конфігурування дисплея
```

Перед нескінченним циклом виконання програми необхідно ініціалізувати дисплей

```
  LCD_DDR |= 0x3F;
  LCD_PORT &= 0xf0;
  CLEARBIT(LCD_PORT,4);
  delay_ms(50);
  Sendnibble(0x02); //Включаємо адресацію на DDRAM
  Sendbyte(0x06); //Новий байт пишеться в n+1 гніздо екрана, зрушується курсор
  Sendbyte(0x0D);//Включити дисплей, курсор у вигляді миготливого квадрата
Clear(); //Очищення дисплея
```

Далі, можна встановлювати курсор у потрібну позицію і виводити символи.

Після головної програми необхідно розташувати функції керування дисплеєм.

```
void Cursorpos(char y,char x)
{
   if(y == 1)
    y = 0x80;
     else
       y = 0xc0;
```

```
  CLEARBIT(LCD_PORT,5);
  Sendbyte(y + x);
}
void Clear(void)
{
  CLEARBIT(LCD_PORT,5);
  Sendbyte(0x01);
}
void Sendbyte(char byte)
{
  SETBIT(LCD_PORT,4);
  LCD_PORT &= 0xf0;
  LCD_PORT |= (byte >> 4);
  CLEARBIT(LCD_PORT,4);
  delay_us(1000);

  SETBIT(LCD_PORT,4);
  LCD_PORT &= 0xf0;
  LCD_PORT |= (byte & 0x0F);
  CLEARBIT(LCD_PORT,4);

  delay_us(3000);
}
void Sendnibble(char byte)
{
  SETBIT(LCD_PORT,4);
  LCD_PORT &= 0xf0;
  LCD_PORT |= (byte & 0x0F);
  CLEARBIT(LCD_PORT,4);
  delay_us(3000);
}
```

Крім наведених функцій необхідно розробити ще функцію void Sendchar(char byte), яка буде виводити символ у позицію, у якій перебуває курсор. Функцію слід розробити на підставі алгоритму передачі даних у дисплей і функції відправлення байта команди в дисплей.

4.3 Зміст звіту

Звіт повинен містити:
1) Тему і мету роботи
2) Схему електричну принципову лабораторного макета
3) Лістинг розробленої програми мовою С++
4) Виводи по роботі

4.4 Перелік контрольних запитань

1) Характеристики і внутрішній пристрій вбудованого в контролер АЦП.
2) Як ініціалізувати індикатор на базі вбудованого контролера HD44780?
3) Як керувати індикатором через 4-х провідну шину?
4) Як виводити символи на індикатор через 4-х провідну шину?

Перелік рекомендованої літератури

1. Белов А.В. AVR микроконтроллеры: от азов программирования до создания практических устройств. СПб: Наука и техника, 2016. – 544 с.
2. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. – М.:ДОДЭКА-XXI, 2007. – 592с.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни
«ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ»

для студентів усіх форм навчання
напряму 6.05803 «Акустотехніка»

Упорядник: Зубков Олег Вікторович

Відповідальний випусковий В.М. Олейніков

Редактор

План 200_, поз.

Підп. до друку Формат 60×84 1/16. Спосіб друку – ризографія.
Умов.друк.арк. Облік. вид.арк. Тираж прим.
Зам. № Ціна договірна.

ХНУРЕ. Україна. 61166, Харків, просп. Науки, 14

Віддруковано в навчально-науковому
видавничо-поліграфічному центрі ХНУРЕ
61166, Харків, просп. Науки, 14