МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

**К. П. Вонсевич, М. О. Безуглий**

# МІКРОПРОЦЕСОРНА ТЕХНІКА
## КОМП'ЮТЕРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського*
*як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою*
*«Комп'ютерно-інтегровані системи та технології в приладобудуванні»*
*спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»*

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти *Жученко О.А*., д.т.н., професор, професор кафедри технічних та програмних засобів автоматизації, КПІ ім. Ігоря Сікорського
*Павловський О.М.*, к.т.н., доцент, доцент кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем, КПІ ім. Ігоря Сікорського

Відповідальний редактор *Тимчик Г.С.,* д.т.н., професор

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 1 від 16.09.2021 р.) за поданням Вченої ради Приладобудівного факультету (протокол № 7/21 від 30.08.2021 р.)*

Електронне мережне навчальне видання

*Вонсевич Костянтин Петрович,* канд. техн. наук
*Безуглий Михайло Олександрович,* д-р. техн. наук, доцент

# МІКРОПРОЦЕСОРНА ТЕХНІКА
## КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Мікропроцесорна техніка: Комп'ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / К. П. Вонсевич, М. О. Безуглий ; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 3,53 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 94 с.

Навчальний посібник включає стислі теоретичні відомості та інструкції необхідні для виконання практичних робіт з дисципліни «Мікропроцесорна техніка». У посібнику розглянуто інформацію про типову структуру мікропроцесорної системи і функції її складових частин, особливості роботи з портами введення/виведення інформації, таймерами, зовнішніми та внутрішніми перериваннями, окремими інтерфейсами передачі інформації. Представлено особливості налаштування модулів мікропроцесорних систем, наведено приклади блоків коду для реалізації типових задач. Кожен практикум закінчується переліком вимог та завдань, що повинні бути реалізовані студентом, контрольними запитаннями до вивченого матеріалу.

Виконання практичних робіт з дисципліни дозволить закріпити теоретичні знання, набуті студентом, сприятиме розвитку прикладних навичок та практичних вмінь по роботі з мікропроцесорною технікою, вдосконаленню самостійних рішень та творчої складової у студентських проектах.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

I/O-порт – порт введення/виведення інформації

SPI – серійний послідовний інтерфейс

ЗП – зовнішнє переривання

ЛКМ – ліва кнопка миші

МК – мікроконтролер

МП – мікропроцесор

МПС – мікропроцесорна система

ПК – персональний комп'ютер

ПКМ – права кнопка миші

ПМ – постійний магніт

ТД – тактове джерело

Т/Л – таймер/лічильник

ШД – шина даних

ШІМ – широтно-імпульсна модуляція

# ВСТУП

Комп’ютерні практикуми, представлені у цьому навчальному посібнику, містять інструкції та завдання до виконання п'яти практичних робіт у двох програмних середовищах: CodeVisionAVR та Proteus. Теоретичний матеріал, представлений у кожному практикумі, є доповненням до лекційної інформації з дисципліни «Мікропроцесорна техніка», а його структура визначає мету роботи та її змістовність. У практикумах наведено теоретичні відомості, приклади налаштування параметрів робочих програм, приклади фрагментів коду та електричних схем для моделювання роботи мікропроцесорної техніки, математичні викладки. Кожен практикум містить опис завдання для виконання студентом, а також набір неповторюваних даних для різних варіантів. Для виконання практикумів студенти повинні знати: принципи роботи у середовищі Proteus, сутність роботи розглянутих складових частин мікропроцесорної техніки, послідовність та принцип вибору її налаштувань, а також сутність окремих математичних обчислень.

# КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 1
# ТИПОВА СТРУКТУРА МІКРОПРОЦЕСОРНОЇ СИСТЕМИ. ПОРТИ ВВОДУ ТА ВИВОДУ ІНФОРМАЦІЇ

**Мета:** Ознайомитись з типовою структурою мікропроцесорної системи та навчитись використовувати порти вводу та виводу інформації у мікроконтролерах серії AVR.

**Змістовність роботи:** Структура типової МПС. Звернення до регістрів, використання портів вводу та виводу інформації у МК. Принципи побудови типової програми для МК на мові програмування високого рівня. Використання функцій середовища Proteus для моделювання роботи електричної схеми із МК.

## 1.1. Теоретичні відомості

### 1.1.1. Структура типової МПС

При проектуванні та виробництві новітніх приладів, з метою автоматизації, підвищення ефективності і автономності роботи апаратури, активно використовуються мікропроцесорні системи (МПС), а також вимірювальні або керуючі модулі, побудовані на їх основі. Існує широке різноманіття МПС з різними вбудованими можливостями, функціями, що визначають особливості їх застосування. Однак, загальні принципи функціонування таких систем залишаються подібними.

Основним діючим елементом сучасної МПС є мікропроцесор (МП). **Мікропроцесор** – це програмно-керований пристрій (побудований на одній або декількох інтегральних мікросхемах), який здійснює процес обробки цифрової інформації та управління нею. Проте, на практиці МП не працює сам по собі, а є лише складовою частиною тієї або іншої **мікропроцесорної системи**. Окрім мікропроцесора, до складу МПС входять й інші, не менш важливі, елементи, що показані на рис. 1.1. А саме:

- RAM (Random Access Memory) – оперативна пам'ять (або оперативний запам’ятовуючий пристрій).

- ROM (Read Only Memory) – пам'ять призначена тільки для читання (або постійний запам’ятовуючий пристрій).
- Port I/O (Port Input/Output) – порти введення-виведення інформації.
- CPU (Central Processing Unit) – центральний процесорний пристрій (або мікропроцесор).

Інформація, записана в **RAM-пам'ять,** зберігається там лише доти, поки подано напругу живлення. Як тільки живлення буде відключено, інформація, записана до RAM, одразу втрачається. **ROM-пам'ять** призначена для довготривалого зберігання інформації і не втрачає записані в неї дані навіть після вимкнення живлення.

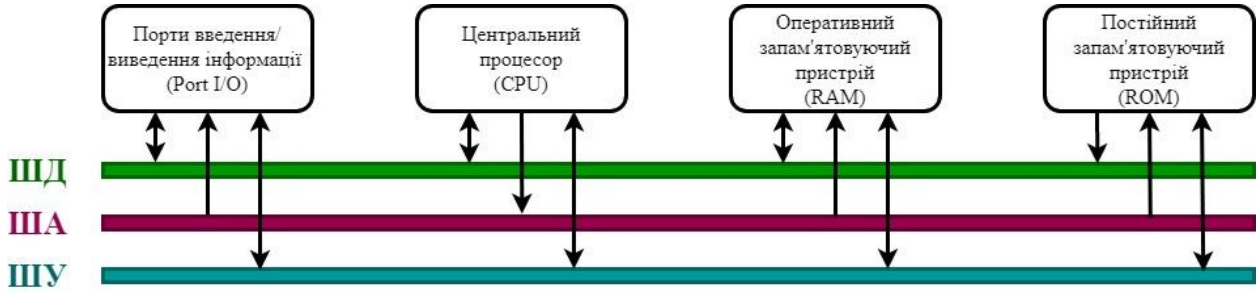



Рис. 1.1. Узагальнена структурна схема типової мікропроцесорної системи

**Порти введення/виведення інформації** – це спеціальні пристрої, за допомогою яких МПС може взаємодіяти із зовнішнім світом. Через **порти введення інформації** мікропроцесор отримує зовнішні впливи або керуючі сигнали (інформацію з датчиків, сигнали від кнопок, ключів тощо). А за допомогою **портів виведення інформації** мікропроцесор керує зовнішніми пристроями (реле, моторами, світловими індикаторами, дисплеями, тощо).

Загалом, робота з портами зводиться до того, що процесор виконує зчитування числа з порту введення інформації, або записує його у порт виводу інформації. Для того щоб МП міг керувати МПС, він з'єднаний з усіма її елементами за допомогою цифрових шин. **Шина** – це набір паралельних провідників, по яких передається цифровий сигнал. У свою чергу, провідники називаються **лініями шини**. У кожен момент часу по шині МПС передається одне двійкове число, а по кожній лінії шини передається один розряд цього числа. У будь-якій МПС є, принаймні, три основні шини, а саме:

- ШД - шина даних (DATA bus).
- ША - шина адреси (ADDR bus).
- ШУ - шина управління (CONTROL bus).

Всі три шини поєднані разом (ШД, ША, ШУ) формують так звану «системну шину». **Шина даних** призначена для передачі інформації від мікропроцесора до периферійних пристроїв, а також у зворотному напрямку. Розрядність шини даних визначається типом процесора, який застосовується у системі, але найчастіше у простих мікропроцесорах шина даних має 8 розрядів.

Подібно до ШД, **шина адреси** є набором провідників, по яких здійснюється передача двійкових чисел в електронній формі. Однак, двійкові числа, що передаються по шині адреси, є адресою комірки пам'яті або номеру порту введення/виведення інформації, до якого в конкретний момент часу звертається мікропроцесор.

У той же час, для управління процесами обміну інформацією МПС повинна мати набір ліній, що передають спеціальні керуючі сигнали. Ці лінії і прийнято об'єднувати в так звану «**шину управління**». Приблизний набір ліній, що входять до складу стандартної шини управління, містить:

- Лінію RD (Read) - сигнал читання.
- Лінію WR (Write) - сигнал запису.
- Лінію MREQ - сигнал ініціалізації пристроїв пам'яті (RAM чи ROM).
- Лінію IORQ - сигнал ініціалізації портів введення-виведення.
- Лінію READY - сигнал готовності.
- Лінію RESET - сигнал скидання.

У типовій МПС, зображеній на рис. 1.1, використовується окрема мікросхема процесора, окремі мікросхеми пам'яті і окремі порти введення/виведення інформації. Однак, стрімкий розвиток мікропроцесорної техніки вимагає все більшої інтеграції декількох складових МПС на одній мікросхемі. Саме тому було розроблено мікросхеми, які об'єднують в собі одразу всі елементи мікропроцесорної системи. Такі мікросхеми отримали назву «**мікроконтролерів**» (МК).

### 1.1.2. Звернення до регістрів портів вводу/виводу інформації у МК серії AVR

Мікроконтролери серії AVR мають у своєму складі від одного до семи портів введення/виведення інформації (I/O-портів). Кожен розряд такого порту приєднаний до одного з виводів (контактів) мікросхеми. Їх кількість для кожної конкретної мікросхеми може варіюватись, однак усі порти мікроконтролерів AVR є восьми-розрядними. Окрім того, I/O-порти можуть бути **повні** і **неповні**.

**Повний** порт містить та використовує всі вісім розрядів, а у **неповних** портах задіяні не всі розряди, а лише деяка їх кількість. Однак, для мікропроцесора, порти завжди залишаються восьми-розрядними. МП завжди записує у такі порти (і зчитує з них) повноцінний байт інформації. Водночас, невживані біти при записі просто втрачаються, а при читанні байту з порту – невживані розряди прирівнюють до нуля. Причиною виникнення повних та неповних портів є з обмежена кількістю виводів (ніжок) у корпусах мікросхем.

Кожен I/O-порт має своє **ім'я**. Імена портів позначаються латинськими літерами від A до G. Для управління кожним портом вводу/виводу інформації використовується три спеціальних **регістри**. Це регістри **PORT*x***, **DDR*x*** і **PIN*x***. Символом «***x***» тут позначено конкретну літеру – ім'я порту (наприклад, для порту A імена регістрів управління будуть: «PORTA, DDRA і PINA», для порту B: «PORTB, DDRB і PINB тощо). Розглянемо детальніше призначення кожного з цих регістрів:

- PORTx - регістр даних (використовується для виведення інформації з I/O-порту).
- DDRx - регістр напрямку передачі інформації.
- PINx - регістр введення інформації в I/O-порт.

Окремі **розряди регістрів** також мають свої імена. Розряди регістра PORT*x* зазвичай іменуються як **P*x*n**. Де «**n**» – це номер розряду. Наприклад, розряди регістру PORTA будуть іменуватися як: PA0, PA1, PA2-PA7. Розряди регістру DDR*x* іменуватимуться, як **DD*x*n** (для порту A – DDA0, DDA1 – DDA7), а розряди регістру PIN*x* матимуть назви **PIN*x*n** (для порту A – PINA0, PINA1 –

PINA7). Для інших портів літера A замінюється відповідно на B, C, D, E, F, G.

Будь-який порт введення/виведення інформації у мікроконтролері серії AVR влаштований таким чином, що кожен його розряд може працювати як на *введення*, так і на *виведення* інформації. Тобто він може бути як *входом*, так і *виходом*. Для перемикання режимів роботи у I/O-портах служить регістр **DDR*x***. Кожен розряд регістру DDR*x* керує своїм розрядом I/O-порту. Якщо в будь-якому розряді регістру DDR*x* записано *логічний нуль*, то відповідний розряд порту працює як *вхід*. Якщо ж у цьому розряді присутнє значення *логічної одиниці*, то розряд порту працює як *вихід*.

Для того, щоб **видати інформацію** на зовнішній вивід мікросхеми, потрібно у відповідний розряд DDR*x* записати логічну одиницю, а потім – виконати запис байту даних у регістр PORT*x*. Вміст відповідного біта цього байту одразу з'явиться на зовнішньому виводі мікросхеми і буде присутній там постійно, поки його не буде замінено іншим значенням (або доти, поки ця лінія порту не переключиться на режим введення інформації).

Для того щоб **прочитати інформацію** із зовнішнього виводу мікросхеми МК, необхідно спочатку перевести потрібний розряд I/O-порту в режим введення інформації. Тобто записати у відповідний розряд регістра DDR*x* логічний нуль. Тільки після цього на відповідний вивід мікроконтролера можна подавати цифровий сигнал від зовнішнього пристрою, а МК виконуватиме зчитування байту даних з регістра PIN*x*. При цьому, вміст відповідного біта із прочитаного байту відповідатиме рівню сигналу, що надійшов на зовнішній вивід мікросхеми та, як наслідок, – відповідного I/O-порту МК.

Порти введення/виведення інформації у мікроконтролерах AVR мають ще одну корисну функцію. У режимі вводу інформації вони можуть підключати до кожної ніжки I/O-порту в мікросхемі **внутрішній навантажувальний резистор**. Такий резистор обмежує вихідний струм з ніжки МК для зовнішніх пристроїв, що підключені між виводом порту і загальним проводом. Внутрішній резистор дозволяє значно розширити можливості порту адже завдяки цьому значно спрощується підключення зовнішніх контактів і кнопок до мікросхеми контролеру. Зазвичай контакті елементи вимагають додаткового підключення зовнішнього резистора, однак, використовуючи внутрішній резистор портів МК,

можна обійтись і без зовнішніх резистивних компонентів. Якщо відповідний порт МК знаходиться у режимі *введення інформації*, то включенням і відключенням внутрішніх резисторів керує регістр PORT*x*. Це добре видно з табл. 1.1, у якій показано всі режими роботи I/O-порту.

Таблиця 1.1. Конфігурації портів введення/виведення інформації

| DDxn | Pxn | Режим | Резистор | Нотатки |
|---|---|---|---|---|
| 0 | 0 | Введення | Вимкнено | Вивід відключено від схеми |
| 0 | 1 | Введення | Ввімкнено | Вивід працює як джерело струму |
| 1 | 0 | Виведення | Вимкнено | На виході лог. «0» |
| 1 | 1 | Виведення | Вимкнено | На виході лог. «1» |

На рис. 1.2. представлено узагальнену електричну схему для одного розряду порту введення/виведення інформації у МК серії AVR.

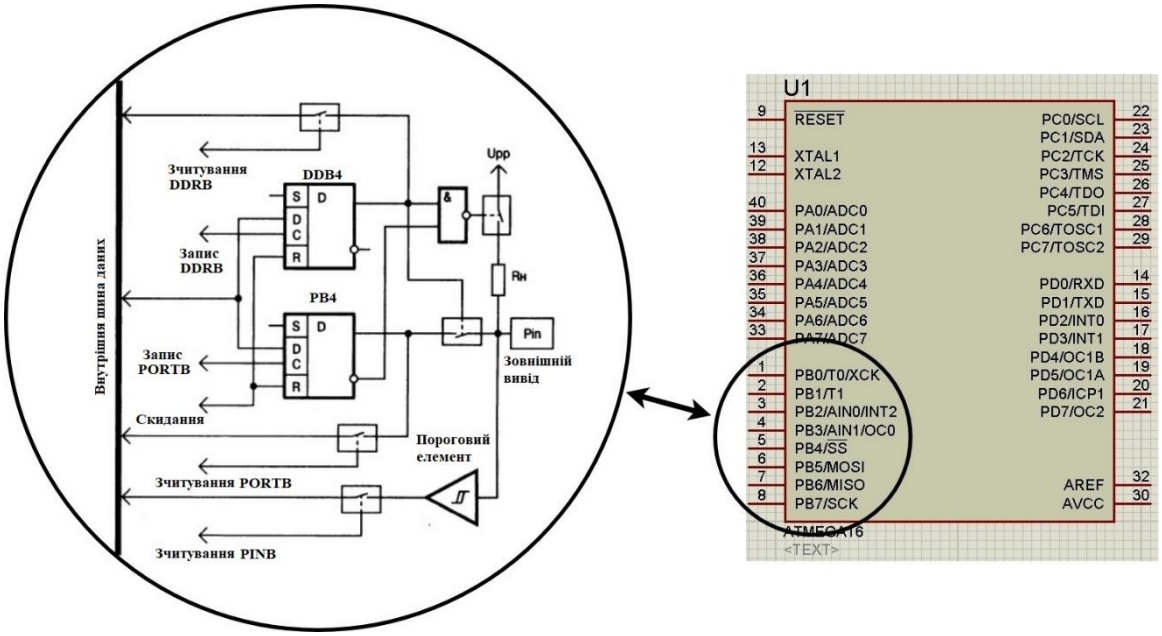



Рис. 1.2. Узагальнена електрична схема одного розряду I/O-порту

Варто зазначити, що представлена схема є спрощеним варіантом реальної схеми I/O-порту адже будь-який порт МК окрім основних функцій має ще ряд додаткових. Як наслідок, реальна схема I/O-порту виглядатиме значно складніше. До кожної реальної схеми додаються елементи, які реалізують усі додаткові функції, що є характерними для конкретного порту.

## 1.2. Завдання для виконання

Створити робочу папку, яка буде мати наступну назву: **MP1_PR1_Шифр групи_Прізвище студента_№ варіанту** (де PR1 розшифровується як: «Практична робота №1»). **Назва папки задається латиницею (англомовними символами)!**

**Завдання 1.1.** Розробити електричну схему для подальшого моделювання роботи портів введення/виведення інформації МК у відповідності до початкових умов, що наведені у табл. 1.2.

Розрахувати номінали резисторів (їх опір та потужність), які необхідно використати у розробленій схемі для обмеження робочого струму на світлодіодах 10-ти сегментного індикатору. Розрахунок проводиться відповідно до робочих параметрів використаних електронних компонентів (а саме: робочої напруги у схемі, вихідного струму що генерується ніжками МК, номінальної робочої напруги світлодіоду в індикаторі). При цьому параметри вихідного струму, який генерується ніжками конкретного МК залежно від напруги живлення у схемі, студент знаходить самостійно (у технічній документації до мікроконтролеру, що надається його виробником).

**Порядок виконання Завдання 1.1.**

1. Відкрити робоче вікно середовища Proteus та обрати з бази електронних компонентів необхідний перелік (у відповідності до номеру варіанту в таблиці 1.2).

2. Розмістити обрані компоненти на робочому полі та з'єднати їх провідниками з метою подальшого моделювання схеми. Місце розташування компонентів на схемі обирається студентом самостійно.

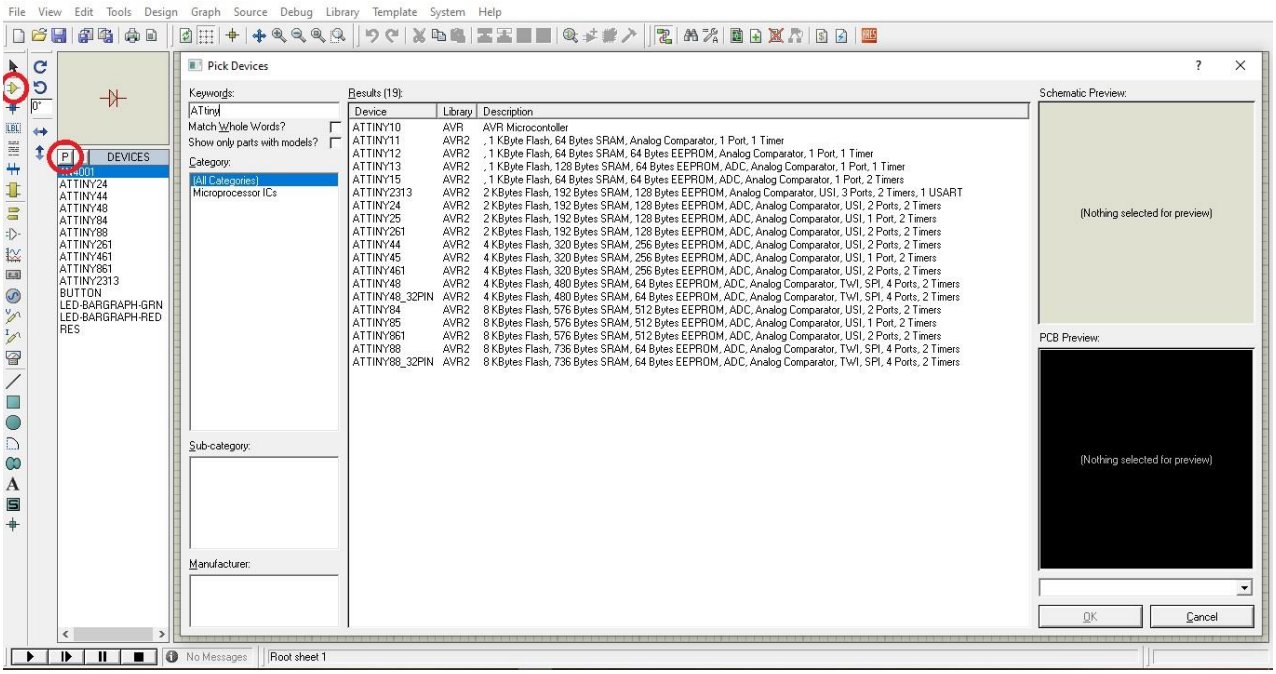



Рис. 1.3. Приклад вибору електронних компонентів із бази середовища Proteus

3. Зберегти проект розробленої схеми у раніше створеній робочій папці із назвою: «PR1_Шифр групи_№ варіанту». **Назва файлу задається лише латиницею (англомовними символами)!**

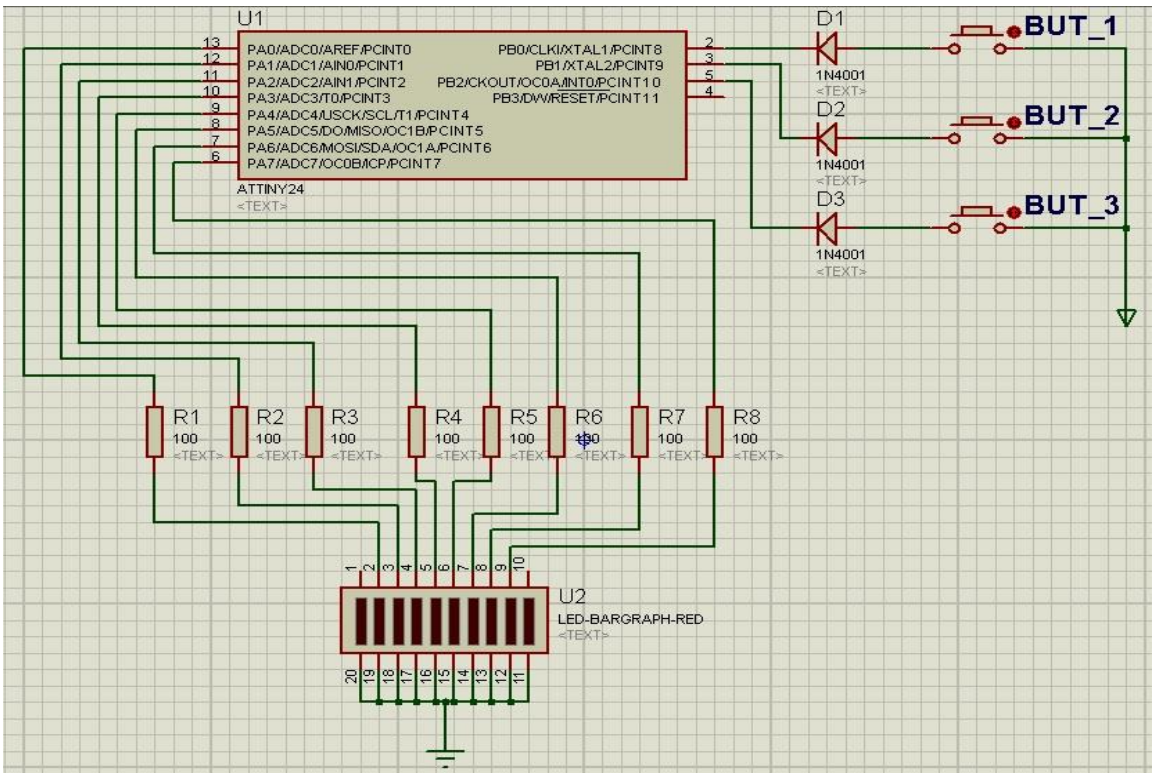



Рис. 1.4. Приклад створеної електричної схеми
та розташування компонентів обраних з бази

**Завдання 1.2.** Провести налаштування та створити програмний код для МК відповідно до індивідуального завдання та початкових умов, наведених у табл. 1.2.

**Порядок виконання Завдання 1.2.**

1. Відкрити робоче вікно середовища CodeVisionAVR, створити новий файл проекту та провести налаштування МК у відповідності до параметрів, вказаних у табл. 1.2. Для цього необхідно виконати наступні кроки:
   - На верхній робочій панелі середовища CodeVisionAVR натиснути кнопку «Create a new file or project» → «New Project» → «Yes» → «AVR8(ATtiny, ATmega, AT90)» → «Ok».
   - У вкладці «Chip», обрати зі списку модель МК відповідно до індивідуального завдання (рядок «Chip») та робочу частоту МК (рядок «Clock»).
   - У вкладці «Ports», провести налаштування I/O-портів на вхід або вихід у залежності від індивідуального завдання.
2. Після завершення налаштувань натиснути кнопку «Generate program, save and exit» та зберегти три файли проекту (усі з однаковою назвою) у раніше створеній робочій папці (у якій вже розміщено Proteus-файл). **Назва файлу задається лише латиницею (англомовними символами)!**
3. Після збереження проекту у основному вікні (яке відкриється програмою CodeVisionAVR автоматично) написати програмний код, який повинен забезпечити виконання наступних умов (див. рис. 1.4):
   - Якщо жодну із кнопок («BUT1»/«BUT2»/«BUT3») не активовано, то всі світлодіоди 10-ти сегментного індикатору «U2» повинні бути вимкнені.
   - Якщо активовано кнопку «BUT1», то мікроконтролер «U1» повинен надіслати сигнал логічної «1» на відповідні ніжки 10-ти сегментного індикатору «U2» і, як наслідок, подати живлення на світлодіоди, підключені до створеної схеми.

- Якщо активовано кнопку «BUT2», мікроконтролер «U1» повинен забезпечити одночасне ввімкнення та вимкнення восьми світлодіодів 10-ти сегментного індикатору «U2» з частотою Fm (що вказана у табл. 1.3).
- Якщо активовано кнопку «BUT3», мікроконтролер «U1» повинен забезпечити почергове ввімкнення та вимкнення кожного з восьми розрядів 10-ти сегментного індикатору «U2», створюючи тим самим візуальний ефект «біжучого рядка».

4. Провести компіляцію створеного коду, натиснувши кнопку «Build all project files» на верхній панелі середовища CodeVisionAVR. Якщо у створеному коді немає критичних помилок (див. поле «Errors» у нижньому вікні CodeVisionAVR), то компілятор створить у робочій папці файл з розширення «*.cof», який можна імпортувати до електричної схеми розробленої у середовищі Proteus.

5. Імпортувати згенерований «*.cof» файл до МК «U1» (див. рис. 1.4). Для цього необхідно двічі натиснути на символ МК лівою кнопкою миші (ЛКМ), та у відкритому вікні («Edit Component») перейти до поля «Program File», (обравши символ теки, розташований праворуч).

6. У цьому ж вікні («Edit Component») задати робочу частоту МК (згідно з номером варіанту в табл. 1.2), вказавши її значення у полі «Clock Frequency» замість фрази «Default».

7. Перевірити достовірність роботи створеного коду шляхом запуску схеми на моделювання (кнопка «Play» у нижньому лівому куті робочого вікна Proteus).

8. Зберегти раніше створений Proteus-файл та результати моделювання.

9. Оформити звіт до практичної роботи.

## 1.3. Вимоги до оформлення звіту

1. Звіт повинен бути представлений в електронному вигляді. Вихідний файл звіту має зберігатись у форматі **«*.pdf».**

2. Назва електронного файлу зі звітом задається у форматі: «МП1_ПР1_Шифр групи_Прізвище студента_№варіанту».

3. Структура звіту повинна містити наступні елементи:

- Титульний аркуш.
- Мету роботи.
- Номер варіанту та індивідуальне завдання (у відповідності до табл. 1.2).
- Електричну схему, розроблену студентом у середовищі Proteus.
- Формули та результати розрахунку **номіналів резисторів (параметрів опору та потужності),** які використані у розробленій електричній схемі.
- Блок-схему алгоритму для робочого коду МК.
- Текст з робочим кодом (повністю, включно з налаштуваннями МК).
- Зображення з результатами моделювання схеми у середовищі Proteus.
- Висновки по роботі.
- Перелік контрольних запитань.

4. Якість зображення та розмір шрифту на схемах, рисунках і таблицях повинні чітко відображати представлену в них інформацію. Усі рисунки, таблиці, схеми та розділи у звіті повинні бути підписані (вимоги див. у додатку А).

5. Файл звіту додається у загальну робочу папку (див. п. 1.2), яка надсилається викладачеві на перевірку. Для проходження перевірки робоча папка повинна містити: Proteus – файл (у форматі «*.DSN») зі створеною електричною схемою, усі файли проекту, створеного у середовищі CodeVisionAVR, звіт до практичної роботи.

**Якщо оформлення звіту і робочої папки не відповідають вимогам, практична робота до захисту не допускається!**

## 1.4. Контрольні запитання

1. Що таке мікропроцесорна система. Які основні складові типової МПС?

2. У чому полягає різниця між мікропроцесором та мікроконтролером?

3. Що таке I/O-порти та для чого вони призначені?

4. Що таке повний та неповний I/O-порт?

5. Які основні регістри використовуються для роботи із I/O-портами у AVR-мікроконтролерах? Яке їхнє призначення?

6. Що таке шина у МПС та яке її призначення?

7. Які основні види шин входять до складу типової МПС?

8. Від чого залежить максимальне число яке можна передати по шині у МПС?

9. З чого складається приблизний набір функціональних ліній, що входять до складу стандартної шини управління у МПС?

Таблиця 1.2. Початкові умови до завдання 1.2

| № варіанту | Модель МК | Тактова частота | Порт для кнопок | Порт для індикатору | Модель індикатору | Струм для сегментів індикатору | Робоча напруга індикатору | Напруга живлення у схемі | Модель діоду |
|---|---|---|---|---|---|---|---|---|---|
| **1** | ATtiny24 | 2 Mhz | B | A | LED-BARGRAPH-GRN | 20 mA | 2 V | 5 V | 1N4001 |
| **2** | ATtiny44 | 1 Mhz | B | A | LED-BARGRAPH-RED | 10 mA | 2.2 V | 4.8 V | 1N4002 |
| **3** | ATtiny48 | 1 Mhz | C | D | LED-BARGRAPH-GRN | 30 mA | 2.4 V | 4.9 V | 1N4003 |
| **4** | ATtiny84 | 4 Mhz | B | A | LED-BARGRAPH-RED | 15 mA | 2.5 V | 5.2 V | 1N4004 |
| **5** | ATtiny88 | 1 Mhz | D | B | LED-BARGRAPH-GRN | 10 mA | 1.8 V | 5.0 V | 1N4005 |
| **6** | ATtiny261 | 2 Mhz | A | B | LED-BARGRAPH-RED | 20 mA | 2 V | 5.1 V | 1N4006 |
| **7** | ATtiny461 | 3 Mhz | B | A | LED-BARGRAPH-GRN | 25 mA | 1.9 V | 4.95 V | 1N4007 |
| **8** | ATtiny861 | 1 Mhz | A | B | LED-BARGRAPH-RED | 12 mA | 1.85 V | 4.85 V | 1N4004 |
| **9** | ATtiny2313 | 2 Mhz | D | B | LED-BARGRAPH-GRN | 15 mA | 2.4 V | 5.0 V | 1N4005 |

Продовження таблиці 1.2

| № варіанту | Модель МК | Тактова частота | Порт для кнопок | Порт для індикатору | Модель індикатору | Струм для сегментів індикатору | Робоча напруга індикатору | Напруга живлення у схемі | Модель діоду |
|---|---|---|---|---|---|---|---|---|---|
| **10** | ATtiny84 | 1 Mhz | B | A | LED-BARGRAPH-GRN | 22 mA | 2.1 V | 5.0 V | 1N4001 |
| **11** | ATtiny2313 | 1 Mhz | B | D | LED-BARGRAPH-RED | 20 mA | 1.9 V | 5.1 V | 1N4003 |
| **12** | ATtiny88 | 3 Mhz | C | D | LED-BARGRAPH-GRN | 14 mA | 3 V | 5.25 V | 1N4007 |
| **13** | ATtiny261 | 3 Mhz | B | A | LED-BARGRAPH-RED | 10 mA | 2.2 V | 5.2 V | 1N4006 |
| **14** | ATtiny48 | 2 Mhz | B | D | LED-BARGRAPH-GRN | 38 mA | 2 V | 4.82 V | 1N4003 |
| **15** | ATtiny24 | 1 Mhz | A | B | LED-BARGRAPH-GRN | 10 mA | 2.8 V | 5.5 V | 1N4005 |
| **16** | ATtiny88 | 1 Mhz | C | D | LED-BARGRAPH-GRN | 21 mA | 3.4 V | 5.0 V | 1N4002 |
| **17** | ATtiny44 | 2 Mhz | A | B | LED-BARGRAPH-RED | 40 mA | 2.95 V | 5.0 V | 1N4004 |
| **18** | ATtiny2313 | 2 Mhz | D | B | LED-BARGRAPH-RED | 80 mA | 3.3 V | 4.8 V | 1N4007 |

Продовження таблиці 1.2

| № варіанту | Модель МК | Тактова частота | Порт для кнопок | Порт для індикатору | Модель індикатору | Струм для сегментів індикатору | Робоча напруга індикатору | Напруга живлення у схемі | Модель діоду |
|---|---|---|---|---|---|---|---|---|---|
| **19** | ATtiny84 | 3 Mhz | A | B | LED-BARGRAPH-GRN | 17 mA | 2.7 V | 4.7 V | 1N4003 |
| **20** | ATtiny261 | 1 Mhz | A | B | LED-BARGRAPH-RED | 19 mA | 2.9 V | 4.9 V | 1N4001 |
| **21** | ATtiny24 | 1 Mhz | A | B | LED-BARGRAPH-GRN | 24 mA | 2.4 V | 5.4 V | 1N4004 |
| **22** | ATtiny44 | 3 Mhz | A | B | LED-BARGRAPH-RED | 11 mA | 2.1 V | 4.1 V | 1N4004 |
| **23** | ATtiny48 | 3 Mhz | D | C | LED-BARGRAPH-GRN | 30 mA | 2.9 V | 4.3 V | 1N4001 |
| **24** | ATtiny84 | 2 Mhz | A | B | LED-BARGRAPH-RED | 25 mA | 1.8 V | 5.0 V | 1N4007 |
| **25** | ATtiny261 | 1 Mhz | B | A | LED-BARGRAPH-RED | 45 mA | 2.5 V | 5.5 V | 1N4003 |

# КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2
# ТАЙМЕРИ. ПЕРЕРИВАННЯ ПО ТАЙМЕРУ. СЕКУНДОМІР НА МІКРОКОНТРОЛЕРІ AVR

**Мета:** Ознайомитись із принципом роботи таймерів/лічильників та навчитись використовувати переривання по таймеру в МК серії AVR.

**Змістовність роботи:** Базові поняття, тактовий генератор, тактова частота. Таймер/лічильник, регістри і тактове джерело таймеру. Переддільник частоти, переривання по таймеру. Типові налаштування таймерів у середовищі CodeVisionAVR. Написання коду програми для МК і використання функцій середовища Proteus для моделювання роботи електричної схеми.

## 2.1. Теоретичні відомості

### 2.1.1. Базові поняття. Такт, тактовий генератор, тактова частота

Кожен мікроконтролер або мікропроцесорна система працює на певній частоті (наприклад: 1 МГц, 4 МГц тощо), а більшість інструкцій у МК виконується за один такт. Тому, перш ніж розглядати поняття таймера, варто визначитись з такими поняттями як «такт» і «частота» у МПС.

**Такт** процесора (або такт ядра процесора) – це проміжок між двома імпульсами тактового генератора.

**Тактовий генератор** (або генератор тактової частоти) синхронізує виконання всіх операцій процесора і призначений для синхронізації різних дій у цифрових пристроях. Він виробляє електричні імпульси заданої частоти, яку часто використовують як «еталонну частоту». Підраховуючи кількість таких імпульсів (що, зазвичай, мають прямокутну форму) можна вимірювати конкретні часові інтервали.

**Тактова частота** МП – це кількість тактових імпульсів, що надходять до процесора з метою синхронізації його роботи. Тактова частота визначає кількість команд, які процесор може виконати за 1 секунду. Наприклад, якщо за одну секунду двічі плеснути у долоні, то частота ударів буде дорівнювати 2 Гц. Якщо плеснути тричі – 3 Гц і т.д. Тобто, чим вищою є частота роботи мікроконтролера (або тактова частота МП), тим швидше він працює. Для того

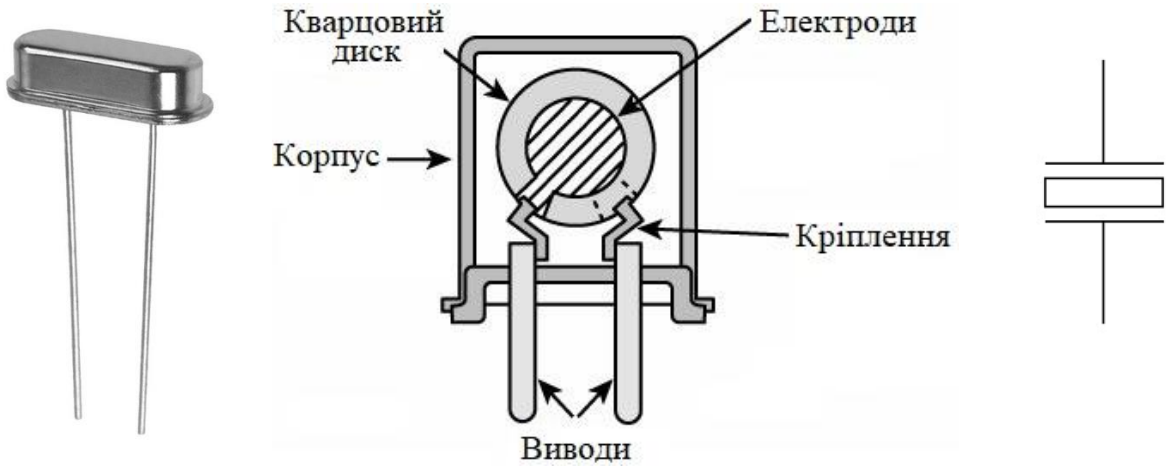
щоб МК працював на деякій тактовій частоті потрібне тактове джерело (тактовий генератор), адже у випадку його відсутності МПС не працюватиме. Тактове джерело у мікропроцесорній системі може бути **внутрішнім** та **зовнішнім**.

**Внутрішній генератор** тактових імпульсів (RC-генератор) передбачений у більшості сучасних мікроконтролерів серії AVR і застосовується у випадку відсутності зовнішнього тактового джерела (ТД). Як правило, нові МК налаштовані на роботу саме від внутрішнього ТД. Однак, внаслідок температурних процесів частота внутрішнього ТД може змінюватися. Саме тому, такий метод синхронізації вважається непридатним для складних проектів на МК, а як стабільне джерело тактової частоти використовується **зовнішнє** джерело – кварцовий генератор або програмована мікросхема генерації.

Головною складовою частиною **кварцового генератора** є кварцовий резонатор (рис. 2.1.а) – це п'єзоелектричний резонатор, основою якого є кристалічний елемент з кварцу. Для створення резонаторів підходить низькотемпературний кварц, що володіє яскраво вираженими п'єзоелектричними властивостями.

Варто зазначити, що для стабільної роботи резонатору важливим є технологічний процес виготовлення його складових частин. Наприклад, залежно від кута зрізу кварцової пластини відрізнятимуться і її електромеханічні властивості (робоча частота, температурна стабільність, стійкість до резонансу, відсутність або наявність паразитних резонансних частот тощо). Саме тому шматок кварцу, з якого виготовлятиметься пластина резонатору, повинен відрізатись під чітко заданим кутом. У процесі виготовлення генератору на кварцову пластинку з обох сторін наносять шари металів (нікель, платину, срібло або золото), після чого жорстким дротом кріплять пластинку в основу герметичного корпусу (рис. 2.1.б).

У процесі монтажу на друковану плату кварцовий резонатор не можна перегрівати, адже перегрів елементу може призвести до деформації електродів, а при температурі вище 573 °С, п'єзоелектричні властивості кварцу втрачаються повністю.




а) б) в)

Рис. 2.1. Кварцовий резонатор: а – загальний вигляд; б – конструкція; в – позначення на електричній схемі

Текстове позначення кварцового резонатора на електричній схемі зазвичай супроводжується надписом «ZQ», «Z», або «X», а графічне позначення показано на рис. 2.1.в.

### 2.1.2. Таймер/лічильник. Регістри і тактове джерело таймеру

Розглянемо властивості таймеру. **Таймер/лічильник (Т/Л)** – це один з модулів МК AVR, за допомогою якого можна вести підрахунок часу і вхідних імпульсів, організовувати переривання або ШІМ-сигнал тощо. Таймер, як складова частина мікроконтролера, здатен генерувати сигнал у конкретний момент часу, що задається користувачем. Коли цей момент настає, викликається відповідне переривання мікроконтролера, нагадуючи йому про необхідність виконати певне завдання або фрагмент коду. Залежно від моделі МК кількість Т/Л може відрізнятись. Більш того, різні таймери можуть мати різний функціонал і розрядність. Наприклад, у мікроконтролері Atmega8 розміщено три Т/Л: два 8-ми розрядних і один 16-ти розрядний, а у МК Atmega640x, 1280x, 2560x – шість таймерів/лічильників (два з розрядністю 8 біт та чотири з розрядністю 16 біт).

Таймер – це пристрій, який виконує постійний підрахунок до певного заданого числа або величини, яка дорівнює розрядності таймеру. Цей процес відбувається за рахунок послідовного збільшення величини змінної, яка

називається **рахунковим регістром**. Таймер збільшує значення свого лічильника доти, доки не буде досягнута максимальна величина, яка може бути записана у рахунковий регістр. У цій точці лічильник переповнюється, а таймер «скидається» у нульове значення. Щоб користувач мав змогу відслідкувати момент «скидання» після перенаповнення рахункового регістру, таймер встановлює відповідний біт у так званий «**прапор переривань».** Варто зазначити, що у випадку 8-ми розрядного (8-ми бітного) таймеру рахунковий регістр може зберігати число до 255, а у випадку 16-ти розрядного, максимальне значення складатиме 65535.

Для того щоб значення лічильника у таймері збільшувалось через точні інтервали часу, його також підключають до тактового джерела, що генерує деякий постійно повторюваний сигнал. Кожного разу, коли на Т/Л надходить тактовий сигнал, таймер збільшує значення лічильника на одиницю. Як тактове джерело для таймеру може використовуватись зовнішнє ТД (у такому випадку говорять про «асинхронний» режим роботи, а ТД підключається до ніжок TOSC1 і TOSC2 на рис. 2.2), але у більшості випадків використовується внутрішнє джерело самого чіпу (внутрішній тактовий генератор).

Окрім того, Т/Л може здійснювати підрахунок імпульсів із «рахункових входів» (позначення T0 і T1 на рис. 2.2). У цьому випадку, при відповідних налаштуваннях, таймер буде підраховувати або передній (перепад з логічного 0 в 1), або задній (перепад логічної 1 в 0) фронт імпульсів, які прийшли на ці входи. При цьому важливо щоб частота вхідних імпульсів не перевищувала тактову частоту процесора, інакше він не встигне обробити вхідний імпульс.

Для того щоб використовувати таймери у МК серії AVR існує набір спеціальних **регістрів налаштувань**. Такі регістри є 8-ми розрядними і мають певну структуру (рис. 2.3). Зокрема, регістр налаштувань має 8 біт, а кожен біт – свою назву та свою установчу (конфігураційну) змінну. Наприклад, для налаштування роботи Т/Л Timer 1 на різній тактовій частоті використовуються два регістри управління TCCR. Три біта (CS12, CS11 і CS10) в регістрі TCCR1B визначають тактову частоту таймера. Обираючи їх у різних комбінаціях, можна отримати для таймеру різну швидкість роботи.

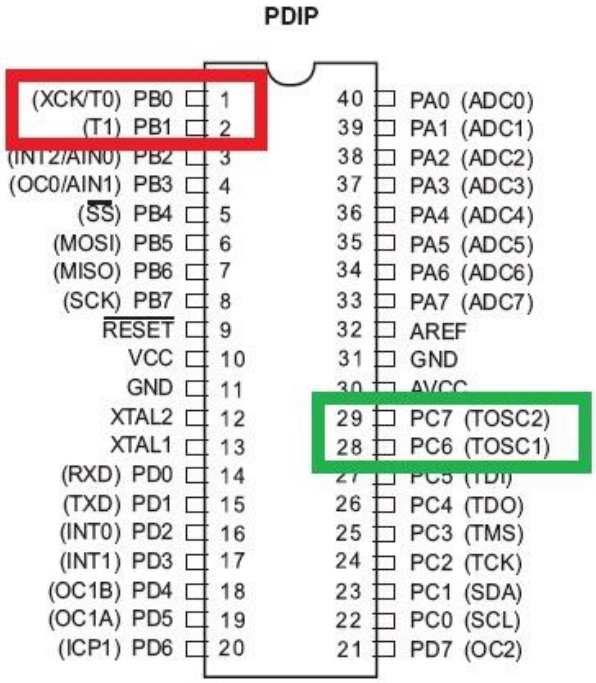



Рис. 2.2. Позначення рахункових входів та входів тактового джерела для таймеру МК AVR


Рис. 2.3. Приклад структури регістру налаштувань Timer 2 у МК AVR

Розглянемо призначення основних регістрів, що використовуються при налаштуванні таймеру в МК серії AVR. Для прикладу налаштуємо «Timer 2» МК ATmega8 на роботу від зовнішнього кварцового резонатора з частотою 32.768 кГц, при тактовій частоті МК у 1 МГц та переддільником на 128. Тоді, код налаштувань для таймеру МК матиме вигляд, показаний на рис. 2.4.

```
64  // Timer/Counter 2 initialization
65  // Clock source: TOSC1 pin
66  // Clock value: PCK2/128
67  // Mode: Normal top=0xFF
68  // OC2 output: Disconnected
69  ASSR=1<<AS2;
70  TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) | (1<<CS22) | (0<<CS21) | (1<<CS20);
71  TCNT2=0x00;
72  OCR2=0x00;
73
74  // Timer(s)/Counter(s) Interrupt(s) initialization
75  TIMSK=(0<<OCIE2) | (1<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<TOIE0);
76
```

Рис. 2.4. Код налаштувань для Timer 2 у середовищі CodeVision AVR

На рис. 2.4 використано позначення:

**ASSR** – регістр статусу асинхронного режиму

- ASSR=1<<AS2; – таймер переведено у асинхронний режим роботи.

**TCCR** – регістр швидкості та режимів рахунку

- TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) | (1<<CS22) | (0<<CS21) | (1<<CS20); – встановлено переддільник на 128.

**TCNT** – регістр підрахунку імпульсів

- TCNT2=0x00; – початкове значення рахункового регістра дорівнює 0x00.

**OCR** – регістр порівняння

- OCR2=0x00; - значення регістру порівняння 0x00.

### 2.1.3. Переддільник частоти. Переривання по таймеру

Рахунок у таймері відбувається з однією постійною швидкістю, що дорівнює тактовій частоті мікроконтролера (адже таймер працює на тій самій частоті, що і мікроконтролер). Однак, іноді така швидкість роботи таймеру може бути занадто високою, тому використовують так звані «переддільники», які зменшують кількість «рахункових імпульсів» у таймері в 8/64/256/1024 разів. Увімкнення переддільника для таймеру МК відбувається програмно.

Припустимо, що при налаштуванні робочої програми було обрано переддільник на $P = 1024$. Якщо частота мікроконтролера $F_M = 8МГц$, то після переддільника частота таймера складатиме:

$$F_{MP} = \frac{F_M}{P} = \frac{8000000}{1024} = 7813Гц \qquad (2.1)$$

Інакше кажучи, після дільника таймер за одну секунду підрахує 7813 імпульсів. Таймери, як і зовнішні переривання, працюють незалежно від основної програми. Тому, за необхідності, Т/Л можна налаштувати так, щоб замість виконання основного циклу він паралельно виконував власний функціонал, який буде прописано у відповідному фрагменті коду. Наприклад, виникла необхідність один раз за 0.5 секунди виконувати деякий код. Тоді, якщо $F_{MP} = 7813 Гц$, за одну секунду буде виконано 7813 цокань таймеру, а за 0.5 секунди – у 2 рази менше (3906 цокань). Для реалізації поставленого завдання достатньо значення 3906 занести до відповідного регістру порівняння у таймері, а з надходженням кожного імпульсу Т/Л перевіряти чи достатньою є вже підрахована кількість імпульсів.

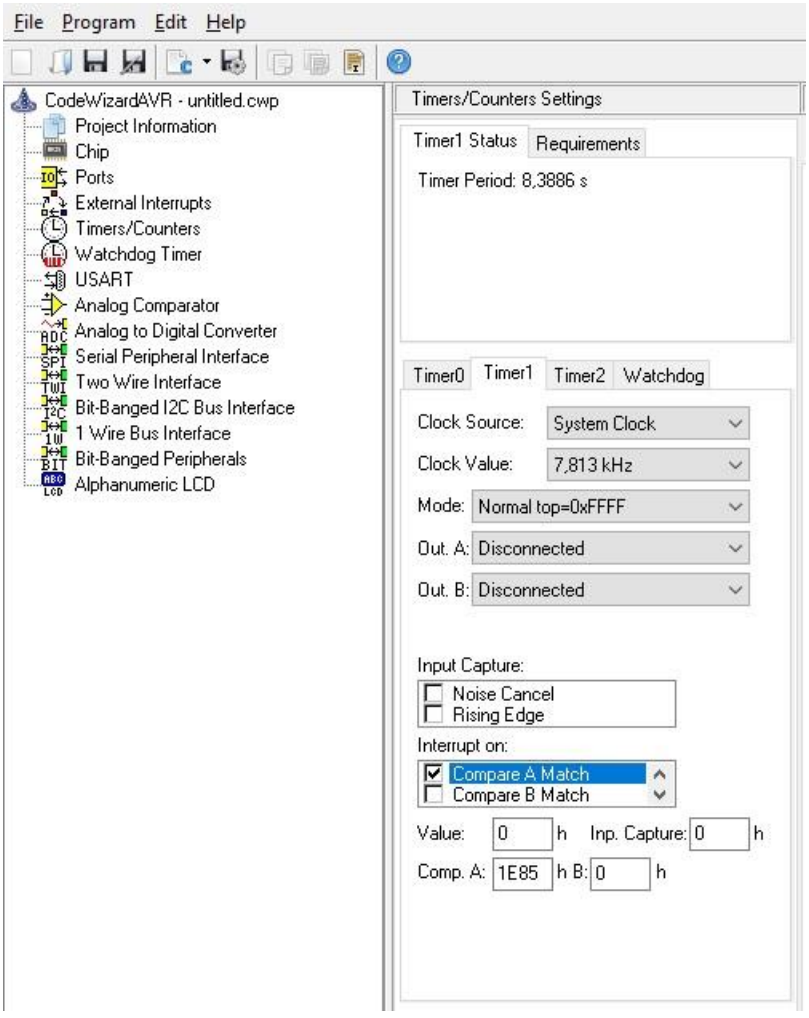



Рис. 2.5. Вікно налаштувань переривання за збігом по Timer_1 у МК AVR

У випадку коли значення таймеру та регістру порівняння співпадуть – відбудеться «**переривання за збігом**». Це означає, що поточна програма

зупиниться та виконається ділянка коду, який прописано у окремому блоці обробки переривань, що не пов'язаний з основною програмою та не впливає на неї.

Виконаємо налаштування таймеру для поставленого завдання. Для цього у вкладці «Timers» у генераторі налаштувань середовища CodeVisionAVR обираємо Timer 1 (рис. 2.5). Задаємо тактову частоту таймера 7.813 кГц і обираємо пункт *Interrupt on: Compare A Match*. Таким чином вказується, що при збігу значення у регістрі порівняння «Comp A» необхідно виконати переривання. Оскільки за умовою переривання повинно виконуватись 1 раз в секунду, то при заданих налаштуваннях необхідно підрахувати 7813 імпульсів. Розрахувавши число 7813 у шістнадцятковій системі, отримаємо значення «1e85». Саме його і необхідно записати у 16-ти бітний регістр порівняння «Comp A». Якщо всі налаштування виконано вірно, у програмному коді МК у середовищі CodeVisionAVR з'явиться такий текст:

```
// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr (void)
{

}
```

Це і є блок обробки переривань по збігу, у «тілі» якого необхідно написати код, що повинен виконуватись за певний проміжок часу. За умовою – величина такого проміжку – це одна секунда. Отже, залишається лише створити змінну, значення якої буде збільшуватись 1 раз за секунду (тобто 1 раз за переривання). Для цього ініціалізуємо змінну «int sek = 0»;. А у перериванні будемо збільшувати її значення від 0 до 59. Приклад фрагменту до створеного програмного коду виглядатиме так:

```
#include <mega8.h>
  int sek = 0; // змінна для зберігання секунд

// Блок обробки переривання за збігом
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
        sek++;          // інкрементувати змінну кожну секунду
        if(sek>59)      // обнулити відлік після 59 секунд
        {
                sek=0;
        }
        TCNT1=0;        // обнулити таймер (регістр підрахунку імпульсів)
}

void main(void)
{
// Declare your local variables here

….

// Налаштування таймеру
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 7,813 kHz
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer Period: 8,3886 s
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
```

```
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (1<<CS12) | (0<<CS11) |
(1<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
// Значення регістру порівняння, що записане у старшому та молодшому розрядах регістру
OCR1AH=0x1E;
OCR1AL=0x85;
OCR1BH=0x00;
OCR1BL=0x00;
  // Запуск таймеру
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (1<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<TOIE0);

…

#asm("sei")

while (1)
    {
        // Код основної програми
    };
}
```

### 2.1.4. Типові налаштування в середовищі CodeVisionAVR та регістри таймеру

Створимо новий проект у CodeVision AVR та відкриємо вікно налаштувань МК у Code WizardAVR. Обравши вкладку «Timers» можна побачити робоче вікно, показане на рис. 2.6. Розглянемо основні пункти меню 8-ми розрядного Т/Л «Timer 2» для МК ATmega8 з тактовою частотою 8 MHz:

**Clock Source** – джерело тактового сигналу.

- *System Clock* – таймер тактується частотою, на якій працює мікроконтролер.

- *TOSC1 pin* – таймер буде працювати від зовнішнього кварцу на ніжках TOSC1, TOSC2.

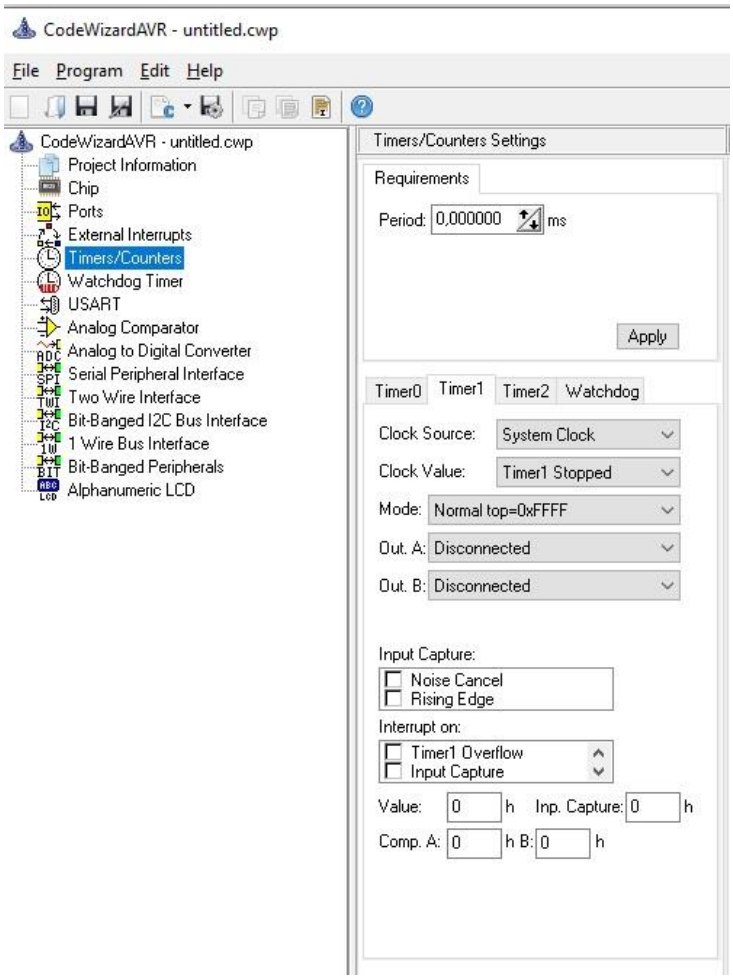



Рис. 2.6. Вікно налаштувань таймерів/лічильників у середовищі CodeVision AVR

**Clock Value** – значення тактової частоти таймеру.

- *Timer№ Stopped* – таймер зупинено.
- *8 000, 000 кГц …7,183 кГц* – тактова частота таймеру залежно від переддільника (на 8, 32, 64, 128, 256 або 1024), Див. п. 2.1.4.

**Mode** – режим функціонування таймеру.

- *Normal top* – таймер рахуватиме від 0 до 255, а після переповнення повертається до 0 і рахунок повторюється.
- *Phase Correct PWM* – таймер циклічно здійснює рахунок від 0 до 255, а потім від 255 до 0. При першому збігу вивід OCxx переводиться у стан логічного 0, а при другому – у стан логічної 1.

- *CTC top* – режим скидання таймеру за збігом: коли значення в рахунковому регістрі досягає значення в регістрі порівняння, то рахунковий регістр повертається у значення 0 і рахунок починається спочатку.
- *Fast PWM top* – таймер рахуватиме від 0 до 255, а після переповнення повертається до 0 і рахунок повторюється. Коли значення в рахунковому регістрі досягає значення в регістрі порівняння (задається в рядку «Compare»), то таймер виставляє певний логічний рівень (який обирається у випадаючому списку «Output») на ніжці OCxx (див. рис. 2.2).

**Output** – вихідна ніжка таймеру.

**Overflow interrupt** – переривання при переповненні таймеру.

**Compare Match Interrupt** – переривання при збігу числа в регістрі порівняння.

**Timer Value** – початкове значення рахункового регістра.

**Compare** – значення регістра порівняння.

## 2.2. Завдання для виконання

Створити робочу папку, яка буде мати наступну назву: **MP1_PR2_Шифр групи_Прізвище студента_№варіанту** (де PR2 розшифровується як: «Практична робота №2»). **Назва папки задається латиницею (англомовними символами)!**

**Завдання 2.1.** Розробити електричну схему для моделювання роботи портів введення/виведення інформації і таймера/лічильника МК у відповідності до початкових умов, що наведені у табл. 2.1.

Розрахувати номінали резисторів (їх опір та потужність), які необхідно використати у розробленій схемі для обмеження робочого струму на світлодіодах 14-ти сегментного індикатору. Розрахунок проводиться відповідно до робочих параметрів використаних електронних компонентів (а саме: робочої напруги у схемі (5В) та струму світлодіодів для сегментів індикатору).

**Порядок виконання Завдання 2.1.**

1. Відкрити робоче вікно середовища Proteus та обрати з бази електронних компонентів необхідний перелік (відповідно до номеру варіанту в таблиці 2.1).

2. Розмістити обрані компоненти на робочому полі, з'єднати їх провідниками та задати робочі параметри (у відповідності до індивідуального завдання) з метою подальшого моделювання схеми. Місце розташування компонентів на схемі обирається студентом самостійно.

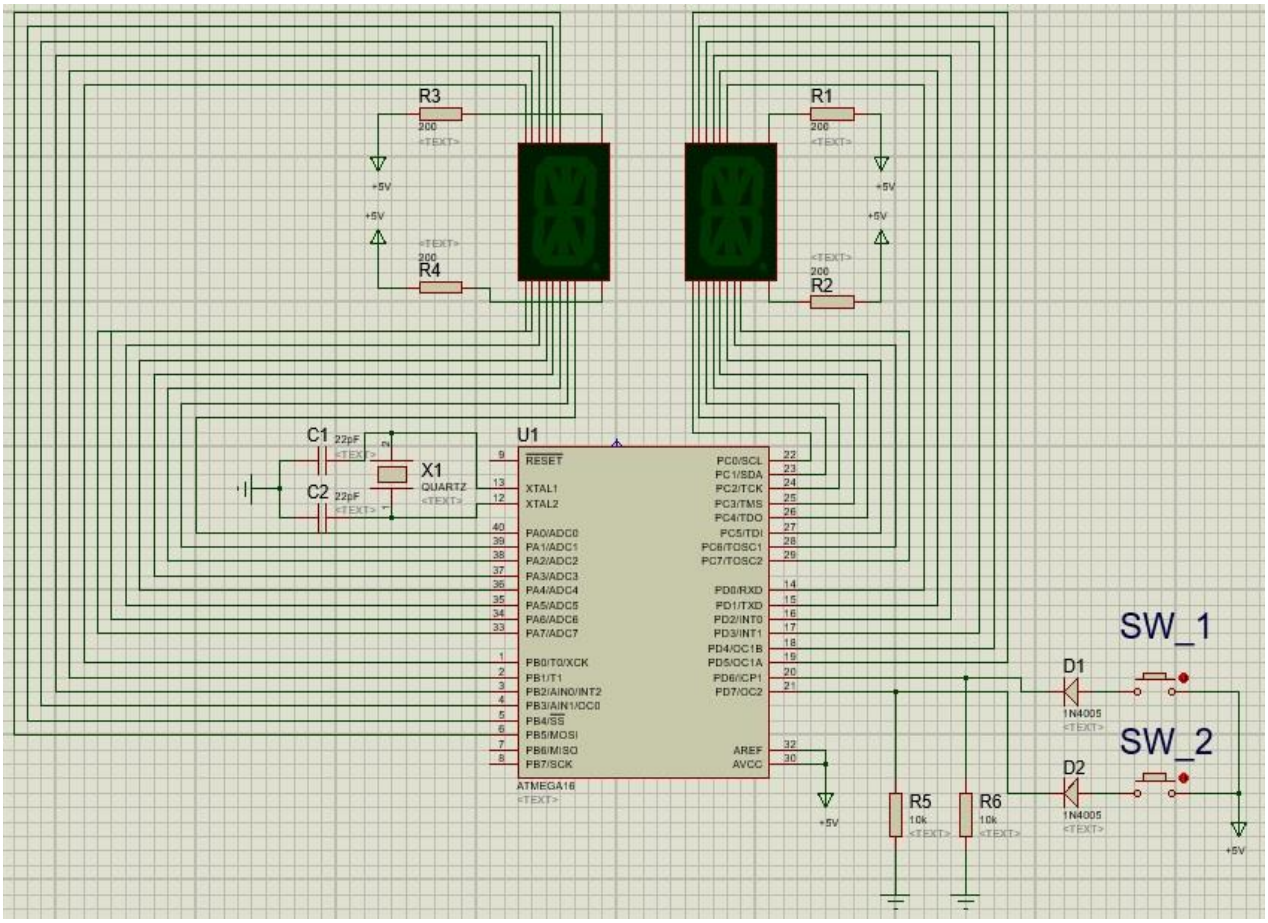



Рис. 2.7. Приклад створеної електричної схеми
та розташування компонентів обраних із бази

3. Зберегти проект розробленої схеми у раніше створеній робочій папці із назвою: «PR2_Шифр групи_№ варіанту». **Назва файлу задається лише латиницею (англомовними символами)!**

**Завдання 2.2.** Провести налаштування та створити програмний код для МК відповідно до індивідуального завдання та початкових умов, наведених у табл. 2.1.

**Порядок виконання Завдання 2.2.**

1. Відкрити робоче вікно середовища CodeVisionAVR, створити новий файл проекту та провести налаштування МК у відповідності до параметрів, вказаних у табл. 2. Для цього необхідно виконати наступні кроки:

- На верхній робочій панелі середовища CodeVisionAVR натиснути кнопку «Create a new file or project» → «New Project» → «Yes» → «AVR8(ATtiny, ATmega, AT90)» → «Ok».
- У вкладці «Chip» обрати зі списку модель МК відповідно до індивідуального завдання (рядок «Chip») та робочу частоту МК (рядок «Clock»).
- У вкладці «Ports» провести налаштування I/O-портів на вхід або вихід, залежно від індивідуального завдання. При цьому початковий стан розрядів порту, до якого підключено сегменти індикаторів, потрібно налаштувати на вихід та перевести їх у стан логічної «1», а не логічного «0», як це задано у системі автоматично (див. рис. 2.8).
- У вкладці «Timers» обрати режим роботи і задати параметри відповідного таймеру, залежно від індивідуального завдання.

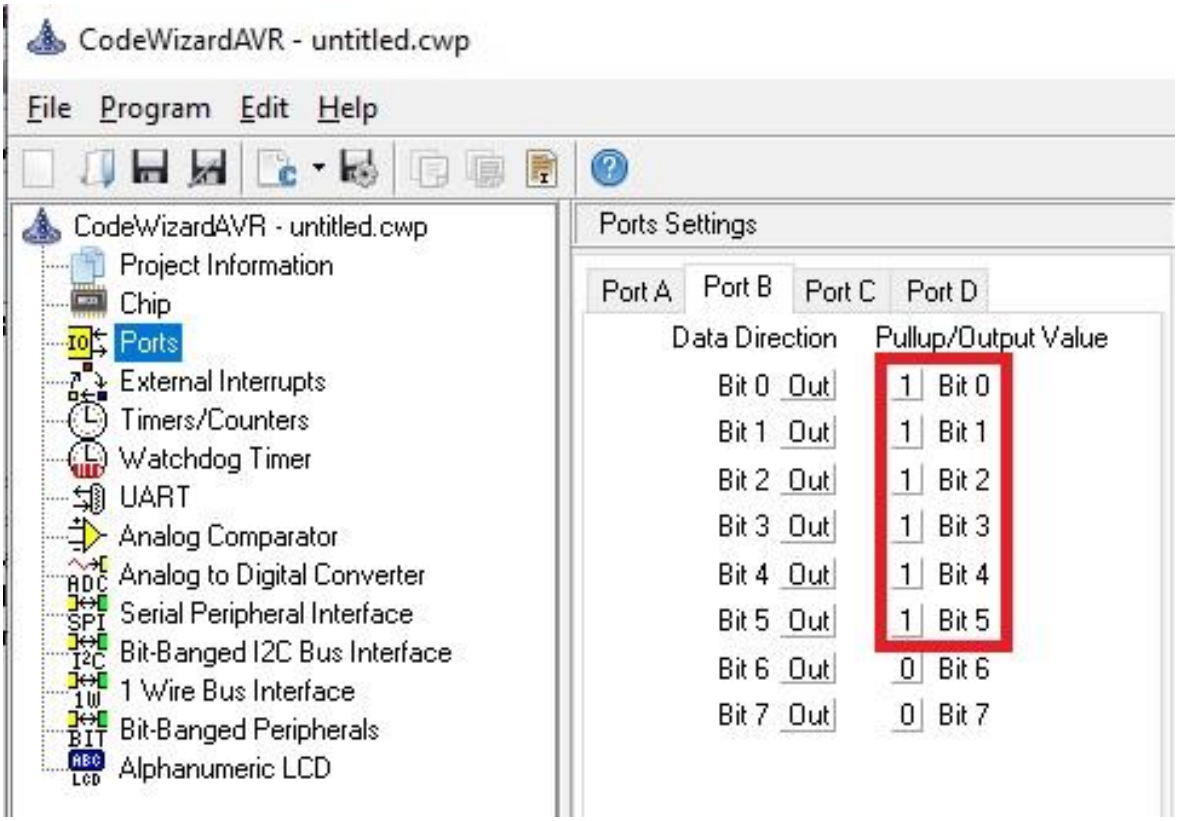



Рис. 2.8. Приклад налаштувань портів у CodeVisionAVR
для другої практичної роботи

2. Після завершення налаштувань натиснути кнопку «Generate program, save and exit» та зберегти три файли проекту (усі з однаковою назвою) у раніше створеній робочій папці (у якій вже розміщено Proteus-файл). **Назва файлу задається лише латиницею (англомовними символами)!**

3. Після збереження проекту в основному вікні (яке відкриється програмою CodeVisionAVR автоматично) написати програмний код, який би забезпечував виконання наступних умов (див. рис. 2.7):

- Якщо жодну з кнопок («SW_1»/«SW_2») не активовано, то всі світлодіоди 14-ти сегментних індикаторів «LI_1» та «LI_2» повинні бути вимкнені.
- Якщо активовано кнопку «SW_1», то мікроконтролер «U1» повинен впродовж $t_{d1}$ секунд виводити на індикатори «LI_1» та «LI_2» фразу «ОК», а після цього – відобразити на індикаторах номер варіанту студента у форматі «N#». У подальшому номер варіанту повинен відображатись на індикаторах увесь час, поки натиснуто кнопку «SW_1».
- Якщо активовано кнопку «SW_2», то мікроконтролер «U1» повинен активувати відповідний таймер/лічильник і вести постійний підрахунок часу, що пройшов від моменту натиснення кнопки «SW_2». Підрахунок часу проводити у секундах та відображати на індикаторах «LI_1» та «LI_2». Після того, як кнопку «SW_2» буде деактивовано, потрібно зупинити таймер, вивести на індикатори фразу «XX» та моргнути ними $N$ разів із затримкою $t_{d2}$. Після цього – відобразити значення у секундах, до якого дорахувала система, поки кнопку «SW_2» було натиснуто. Розраховане значення повинно відображатись на індикаторах впродовж часу $t_{d3}$, а після цього всі світлодіоди 14-ти сегментних індикаторів повинні бути вимкнені.

**Звернути увагу:** оскільки у електричній схемі використовуються 14-ти сегментні індикатори із загальним анодом, то для активації конкретних сегментів МК повинен надсилати на відповідні ніжки сигнал логічного «0». Для підрахунку часу, що пройшов від моменту натискання кнопки, необхідно використати переривання по таймеру/лічильнику. Номер таймеру і режим його роботи вказані у індивідуальному завданні (табл. 2.1).

4. Провести компіляцію створеного коду, натиснувши кнопку «Build all project files» на верхній панелі середовища CodeVisionAVR.

5. Імпортувати згенерований «*.cof» файл до МК «U1» та задати робочу частоту МК (у відповідності до номеру варіанту в табл. 2.1). Для цього обрати у вкладці «CKSEL Fuses» параметр «Ext.RC» у діапазоні, на якому працює кварцовий генератор, що використаний у схемі. Після необхідно вказати конкретне значення робочої частоти МК у полі «Clock Frequency» замість фрази «Default».

6. Перевірити достовірність роботи створеного коду шляхом запуску схеми на моделювання.

7. Зберегти раніше створений Proteus-файл та результати моделювання.

8. Оформити звіт до практичної роботи.

## 2.3. Вимоги до оформлення звіту

1. Звіт повинен бути представлений в електронному вигляді. Вихідний файл звіту має зберігатись у форматі **«*.pdf»**.

2. Назва електронного файлу зі звітом задається у форматі: «МП1_ПР2_Шифр групи_Прізвище студента_№варіанту».

3. Структура звіту повинна містити наступні елементи:

- Титульний аркуш.
- Мету роботи.
- Номер варіанту та індивідуальне завдання (відповідно табл. 2.1).
- Електричну схему, розроблену студентом у середовищі Proteus.
- Формули та результати розрахунку **номіналу резисторів (параметри опору та потужності), значення частоти таймеру і значення регістру порівняння для таймеру.**
- Блок-схему алгоритму для робочого коду МК.
- Текст із робочим кодом (повністю, включно з налаштуваннями).
- Зображення із результатами моделювання схеми у середовищі Proteus.
- Висновки по роботі.
- Перелік контрольних запитань.

4. Якість зображення та розмір шрифту на схемах, рисунках і таблицях повинні чітко відображати представлену в них інформацію. Усі рисунки, таблиці, схеми та розділи у звіті повинні бути підписані (вимоги див. у додатку А).

5. Файл звіту додається у загальну робочу папку (див. п. 2.2), яка надсилається викладачу на перевірку. Для проходження перевірки робоча папка повинна містити: Proteus – файл (у форматі «*.DSN») зі створеною електричною схемою, усі файли проекту створеного у середовищі CodeVisionAVR, звіт до практичної роботи.

**Якщо оформлення звіту та робочої папки не відповідають вимогам, практична робота до захисту не допускається!**

## 2.4. Контрольні запитання

1. Що таке такт?
2. Що таке тактова частота?
3. Що тактове джерело МК? Які бувають види тактових джерел?
4. Як працює кварцовий генератор і для чого він потрібен?
5. Що таке таймер/лічильник у МК?
6. На що впливає розрядність таймеру? Які таймери використовуються у МК серії AVR?
7. Яка структура регістру налаштувань для таймеру МК AVR?
8. Що таке рахунковий регістр та прапор переривань?
9. Що таке переривання по таймеру, які вони бувають і для чого потрібні?
10. Що таке переддільник тактової частоти таймеру і для чого він потрібен?
11. Які основні режими роботи таймерів у МК серії AVR? Охарактеризуйте їх.
12. Від яких тактових джерел може працювати таймер у МК серії AVR?

Таблиця 2.1. Початкові умови до завдання 2.2

| № варіанту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | Таймер | Режим роботи таймеру | $t_{d1}$ (сек) | $N$ | $t_{d2}$ (мс) | $t_{d3}$ (мс) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | ATmega16 | 8 Mhz | 22 pf | 14SEG-MPX1-CA-BLUE | 20 mA | 1N4001 | Timer_1 | Normal top | 2 | 5 | 800 | 800 |
| **2** | ATmega32 | 4 Mhz | 15 pf | 14SEG-MPX1-CA-GRN | 10 mA | 1N4002 | Timer_2 | CTC top | 3 | 3 | 300 | 1000 |
| **3** | ATmega64 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-RED | 30 mA | 1N4003 | Timer_2 | Normal top | 1 | 2 | 400 | 900 |
| **4** | AT90S8535 | 4 Mhz | 18 pf | 14SEG-MPX1-CA-BLUE | 15 mA | 1N4004 | Timer_2 | CTC top | 2 | 3 | 250 | 800 |
| **5** | ATmega8535 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-GRN | 10 mA | 1N4005 | Timer_1 | CTC top | 3 | 2 | 350 | 700 |
| **6** | ATmega8535 | 8 Mhz | 22 pf | 14SEG-MPX1-CA-RED | 20 mA | 1N4006 | Timer_1 | Normal top | 2 | 4 | 850 | 600 |
| **7** | AT90S8535 | 1 Mhz | 15 pf | 14SEG-MPX1-CA-RED | 25 mA | 1N4007 | Timer_2 | CTC top | 4 | 5 | 300 | 500 |
| **8** | ATmega64 | 8 Mhz | 16 pf | 14SEG-MPX1-CA-GRN | 12 mA | 1N4004 | Timer_2 | CTC top | 3 | 2 | 800 | 1000 |
| **9** | ATmega32 | 2 Mhz | 22 pf | 14SEG-MPX1-CA-BLUE | 15 mA | 1N4005 | Timer_2 | Normal top | 2 | 3 | 250 | 800 |
| **10** | ATmega16 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-GRN | 15 mA | 1N4002 | Timer_1 | CTC top | 1 | 4 | 300 | 1000 |
| **11** | ATmega16 | 4 Mhz | 18 pf | 14SEG-MPX1-CA-BLUE | 10 mA | 1N4003 | Timer_2 | Normal top | 2 | 5 | 400 | 900 |
| **12** | AT90S8535 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-RED | 20 mA | 1N4004 | Timer_1 | Normal top | 3 | 2 | 250 | 800 |
| **13** | AT90S8535 | 1 Mhz | 15 pf | 14SEG-MPX1-CA-BLUE | 25 mA | 1N4007 | Timer_2 | Normal top | 3 | 2 | 350 | 700 |
| **14** | ATmega8535 | 4 Mhz | 20 pf | 14SEG-MPX1-CA-RED | 15 mA | 1N4006 | Timer_2 | CTC top | 2 | 5 | 500 | 1000 |

Продовження таблиці 2.2

| № варіанту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | Таймер | Режим роботи таймеру | $t_{d1}$ (сек) | $N$ | $t_{d2}$ (мс) | $t_{d3}$ (мс) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | ATmega16 | 1 Mhz | 20 pf | 14SEG-MPX1-CA-GRN | 15 mA | 1N4004 | Timer_2 | CTC top | 3 | 4 | 500 | 900 |
| **16** | ATmega64 | 8 Mhz | 18 pf | 14SEG-MPX1-CA-BLUE | 18 mA | 1N4003 | Timer_2 | Normal top | 4 | 2 | 500 | 1200 |
| **17** | ATmega64 | 6 Mhz | 21 pf | 14SEG-MPX1-CA-GRN | 32 mA | 1N4002 | Timer_1 | Normal top | 2 | 3 | 600 | 800 |
| **18** | ATmega16 | 6 Mhz | 20 pf | 14SEG-MPX1-CA-BLUE | 17 mA | 1N4007 | Timer_1 | CTC top | 1 | 5 | 450 | 1100 |
| **19** | ATmega16 | 4 Mhz | 19 pf | 14SEG-MPX1-CA-RED | 16 mA | 1N4001 | Timer_2 | Normal top | 4 | 5 | 550 | 800 |
| **20** | ATmega32 | 8 Mhz | 19 pf | 14SEG-MPX1-CA-BLUE | 22 mA | 1N4003 | Timer_2 | CTC top | 3 | 2 | 450 | 750 |
| **21** | ATmega32 | 6 Mhz | 17 pf | 14SEG-MPX1-CA-RED | 28 mA | 1N4005 | Timer_1 | Normal top | 2 | 4 | 700 | 850 |
| **22** | ATmega8535 | 1 Mhz | 15 pf | 14SEG-MPX1-CA-RED | 10 mA | 1N4002 | Timer_1 | Normal top | 1 | 3 | 500 | 1300 |
| **23** | ATmega8535 | 3 Mhz | 22 pf | 14SEG-MPX1-CA-GRN | 19 mA | 1N4007 | Timer_2 | CTC top | 3 | 2 | 650 | 1400 |
| **24** | AT90S8535 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-BLUE | 14 mA | 1N4001 | Timer_2 | Normal top | 3 | 2 | 200 | 1200 |
| **25** | AT90S8535 | 2 Mhz | 15 pf | 14SEG-MPX1-CA-GRN | 11 mA | 1N4004 | Timer_2 | CTC top | 4 | 4 | 500 | 1000 |

# КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3
# ПЕРЕРИВАННЯ ЗА ЗБІГОМ. WATCHDOG ТАЙМЕР. КЕРУВАННЯ КРОКОВИМ ДВИГУНОМ

**Мета:** Ознайомитись з принципом роботи таймерів/лічильників, сторожового таймеру і навчитись використовувати мікроконтролер AVR для керування крокoвим двигуном.

**Змістовність роботи:** Переривання за збігом, генерація звука. Сторожовий таймер, принцип функціонування Watchdog. Принцип роботи крокового двигуна та керування драйвером двигуна за допомогою мікроконтролеру. Написання коду програми для МК і використання функцій середовища Proteus для моделювання роботи електричної схеми.

## 3.1. Теоретичні відомості

### 3.1.1.Переривання за збігом. Генерація звука

Розглянемо як за допомогою таймеру МК серії AVR згенерувати звук та вивести його на динамік. Для приклад, використаємо мікроконтролер ATMega8, стандартну обв'язку (у вигляді кварцового резонатору на 8 МГц і двох конденсаторів на 22 пФ) та п'єзовипромінювач LS1 без внутрішнього генератора (рис. 3.1).

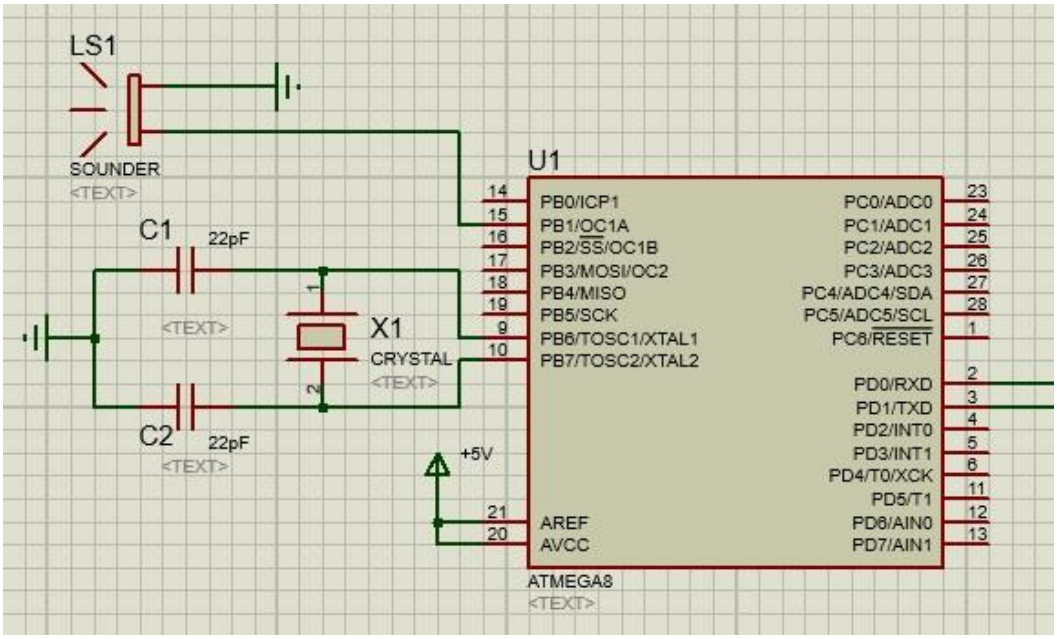



Рис. 3.1. Приклад схеми для генерації звукових коливань на МК AVR

Для створення звуку на п'єзовипромінювачі необхідно забезпечити коливання його мембрани з певною частотою. Кожній звуковій ноті відповідає своя частота коливання, наприклад, частота коливання у 261.6 Гц відповідає ноті «до» першої октави, 293.7 Гц – ноті «ре» першої октави, 329.6 Гц – ноті «мі» тощо. Тобто, якщо підключити певну ніжку МК до випромінювача і перемикати її зі швидкістю 261 раз за секунду, то можна отримати звук ноти «до», 293 рази за секунду – ноти «ре» тощо. У загальному випадку звук з частотою вище 1 кГц буде більш «писклявим», а нижче 300 Гц – більш «басовим».

Для перемикання ніжки мікроконтролеру з необхідною частотою використовують таймер/лічильник. Для цього достатньо порівнювати поточне значення таймеру з деяким завчасно заданим числом (що відповідає частоті певної ноти), та у момент, коли ці значення співпадають, – виконувати переривання. Для цього будемо використовувати внутрішній 16-ти розрядний таймер/лічильник (Т/Л1) МК, основи роботи з яким було розглянуто у практичній роботі №2. Принцип формування частоти за допомогою Т/Л1 полягає у наступному: таймер підраховує імпульси доти, доки його значення не буде дорівнювати числу, яке записане у регістр порівняння OCR1A. У момент, коли значення у OCR1A співпадатиме з поточним значенням таймера, – відбувається переривання, у якому поточний стан порту (а також ніжки МК, що під'єднана до нього) повинен змінюватись на протилежний (інвертуватись).

Приклад налаштувань таймеру показано на рис. 3.2. Тут число «107», записане у регістрі порівняння, – це значення, що відповідає частоті у 263 Гц, яке записане у шістнадцятковій системі числення. Для генерування сигналу на певній частоті в блоці коду програми МК, що відповідає за переривання за збігом, необхідно записати наступне:

```
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
 PORTB.1=!PORTB.1;   // Інвертування значення порта. Генерування меандру
}
```

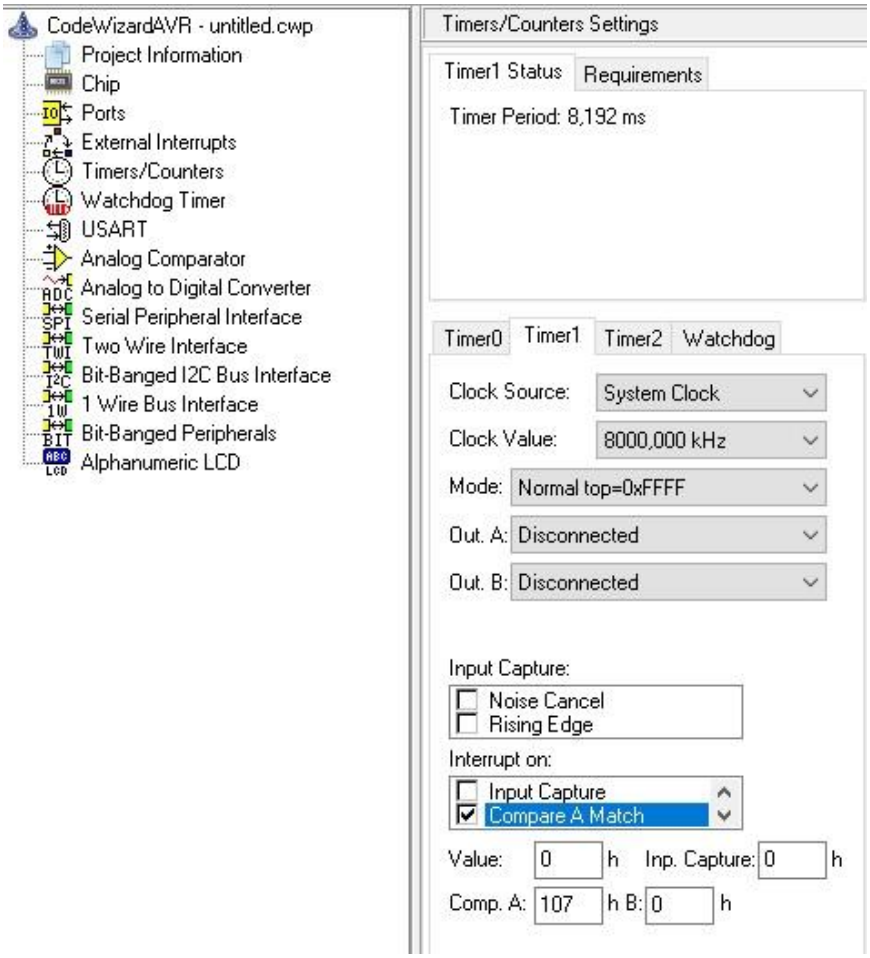



Рис. 3.2. Приклад налаштування переривань по таймеру
для генерації звукових коливань

Як наслідок, ніжка МК до якої підключено п'єзовипромінювач, буде вмикатись/вимикатись кожного разу, коли відбувається переривання, а на виході ніжки генеруватиметься «пульсуючий» сигнал (меандр) з певною частотою. Змінюючи значення у регістрі OCR1A, можна регулювати частоту сигналу і, як наслідок, частоту генерації звукових коливань.

### 3.1.2. Сторожовий таймер. Принцип функціонування Watchdog

При роботі з мікроконтролерами можуть виникати ситуації, при яких виконувана програма може «зависнути». У таких випадках необхідно підключати схему до відповідної ніжки МК (вивід RESET), яка відповідає за скидання чіпу, або вимикати живлення системи повністю, щоб відновити її роботу. Альтернативою цим рішенням є застосування так званого «Сторожового таймера» (Watchdog), що реалізований у МК серії AVR.

**Watchdog** – це таймер, який здійснює підрахунок імпульсів незалежно від основної програми та призначений для відслідковування моменту її «зависання». Суть його роботи полягає у наступному: якщо Watchdog увімкнено і основна програма працює стабільно, то через певні проміжки часу в ній повинен виконуватись код, який скидає цей таймер. Якщо таймер постійно скидається, то це означає, що система працює стабільно. Однак, якщо у певний момент часу код скидання не надійде до Watchdog-таймеру, це означатиме, що частина основної програми не виконалась і МК припинив свою належну роботу («завис»). У цьому випадку Watchdog дорахує до певного значення і повністю перезапустить роботу мікроконтролера (як наслідок, основна програма почне виконуватися з самого початку). Тобто, Watchdog через певні проміжки часу (при настанні події **Time-Out**) примусово перезавантажує роботу основної програми. Нормально працююча система для того щоб уникати перезавантаження, повинна постійно онуляти (скидати) таймер через проміжки часу, які є меншими ніж **Time-Out** період цього таймеру.

Розглянемо основні налаштування Watchdog-таймеру (базуючись на мікроконтролері ATMega8) і приклад його застосування. На рис. 3.3. показано вікно налаштувань Watchdog у CodeVisionAVR.

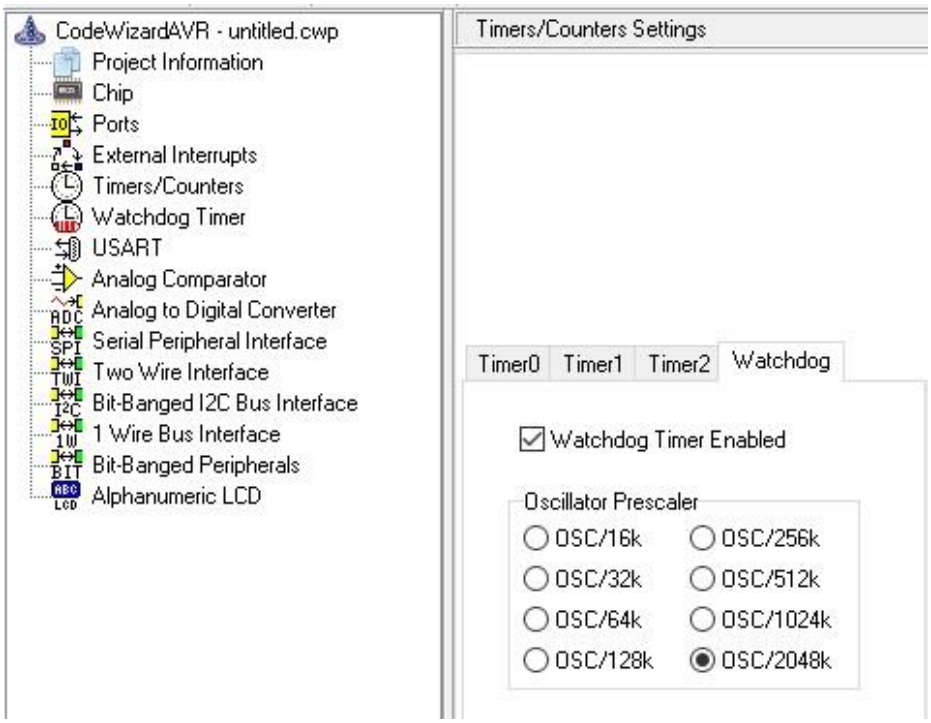



Рис. 3.3. Приклад налаштувань для Watchdog таймеру

Для активування цього таймеру достатньо налаштувати лише два основних параметри:

- Увімкнути таймер, обравши пункт «Watchdog Timer Enabled» на панелі налаштувань.
- Задати параметри переддільника частоти, на якій буде працювати таймер (див. п. 2.1.3 у практичній роботі №2).

У технічній документації до мікроконтролеру вказано, що Watchdog – таймер тактується від окремого вбудованого генератора, частота якого складає 1 МГц при напрузі живлення номіналом у 5 В. Однак, необхідно враховувати, що робоча частота може змінюватись при не стабільному живленні МК (табл. 3.1).

*Таблиця 3.1.*

**Значення параметру Time-Out залежно від переддільника Watchdog**

| WDP2 | WDP1 | WDP0 | Кількість циклів Watchdog генератора | Типове значення Time-Out при $V_{CC} = 3.0V$ | Типове значення Time-Out при $V_{CC} = 5.0V$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 16K (16 384) | 17.1 мс | 16.3 мс |
| 0 | 0 | 1 | 32K (32 768) | 34.3 мс | 32.5 мс |
| 0 | 1 | 0 | 64K (65 536) | 68.5 мс | 65 мс |
| 0 | 1 | 1 | 128K (131 072) | 0.14 с | 0.13 с |
| 1 | 0 | 0 | 256K (262 144) | 0.27 с | 0.26 с |
| 1 | 0 | 1 | 512K (524 288) | 0.55 с | 0.52 с |
| 1 | 1 | 0 | 1024K (1 048 576) | 1.1 с | 1.0 с |
| 1 | 1 | 1 | 2048K (2 097 152) | 2.2 с | 2.1 с |

Наприклад, для переддільника «16 К» при 3 В **Time-Out** складає 17.1 мс, а при 5 В – вже 16.3 мс тощо. Налаштуємо проект для переддільника 2048 К, (враховуючи, що напруга живлення мережі складає 5 В) і запустимо його в середовищі Proteus. Згенерований код МК виглядатиме наступним чином:

```
#include <mega8.h>
void main(void)
{
// Watchdog Timer initialization
// Watchdog Timer Prescaler: OSC/2048k
```

```
#pragma optsize-
#asm("wdr")
WDTCR|=(1<<WDCE) | (1<<WDE);
WDTCR=(0<<WDCE) | (1<<WDE) | (1<<WDP2) | (1<<WDP1) | (1<<WDP0);
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif
// Globally enable interrupts
#asm("sei")
while (1)
    { // Place your code here }
}
```

Враховуючи, що основний код програми порожній і в ньому не прописано команду для скидання Watchdog-таймеру, то після 2.1 секунди Proteus згенерує повідомлення про те, що: «Watchdog не скинуто і мікроконтролер буде перезавантажено». Після чого відбудеться фактичне перезавантаження системи (рис. 3.4). Скидання таймеру відбувається командою: **#asm («wdr»)**. При роботі з Watchdog-таймером необхідно також враховувати і те, що у бібліотеці «delay.h» вже реалізовано скидання Watchdog.

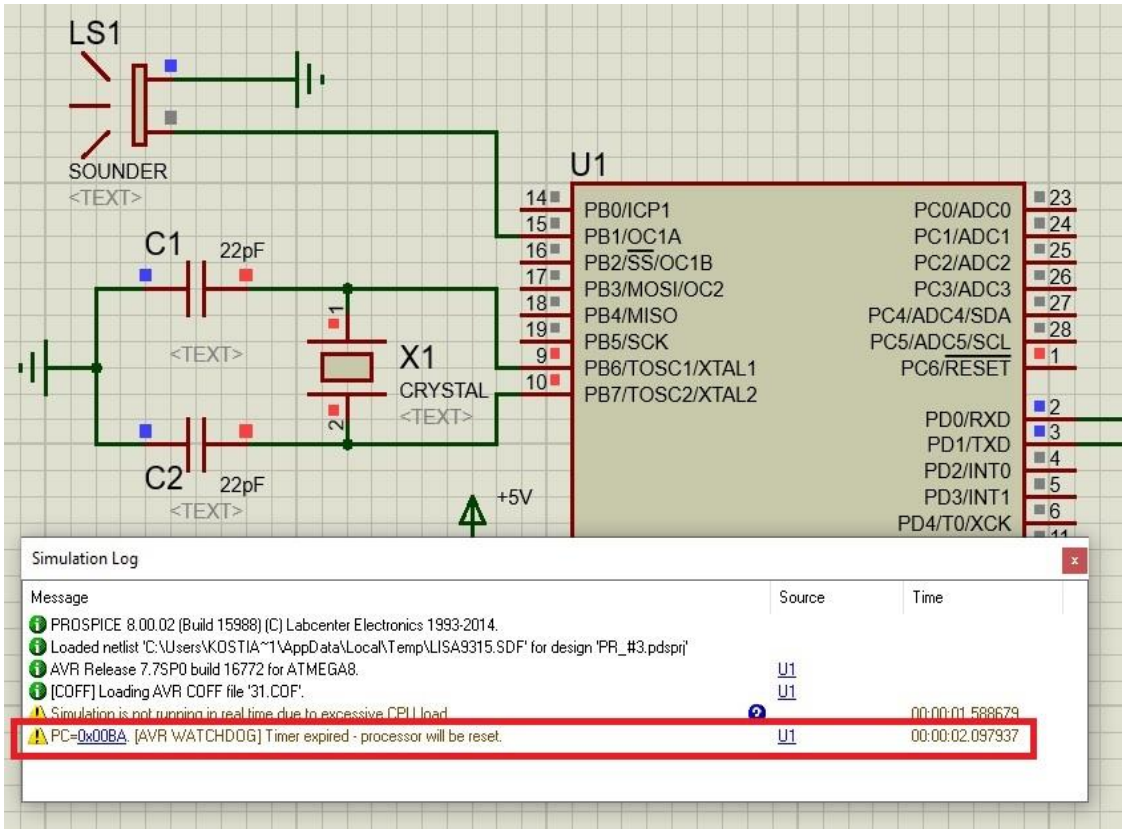



Рис. 3.4. Результат моделювання роботи Watchdog-таймеру в середовищі Proteus

Якщо увімкнути сторожовий таймер за допомогою Fuse-бітів МК, то він буде задіяний у роботі постійно, без можливості повторно відключити його програмним шляхом.

### 3.1.3. Принцип роботи крокового двигуна. Керування драйвером двигуна

Розглянемо принцип роботи крокового двигуна. Уявімо собі постійний магніт (ПМ), що закріплений на вертикальній осі, яка проходить через його центр. Цей магніт може вільно обертатись навколо осі, а біля нього розташований нерухомо-закріплений електромагніт. У відповідності до правила свердлика північ електромагніту буде ліворуч, а південь – праворуч. Якщо на початок обмотки електромагніту подати мінус, а на кінець – плюс, то постійний магніт повернеться до електромагніту північним полюсом (синім кінцем). Якщо ж змінити полярність електромагніту на протилежну, ПМ почне рух у зворотну сторону та повернеться до електромагніту південним полюсом (червоним кінцем). Як наслідок, у залежності від полярності електромагніту, ПМ матиме два стійких стани, а крок його руху буде дорівнювати 180° (рис. 3.5).

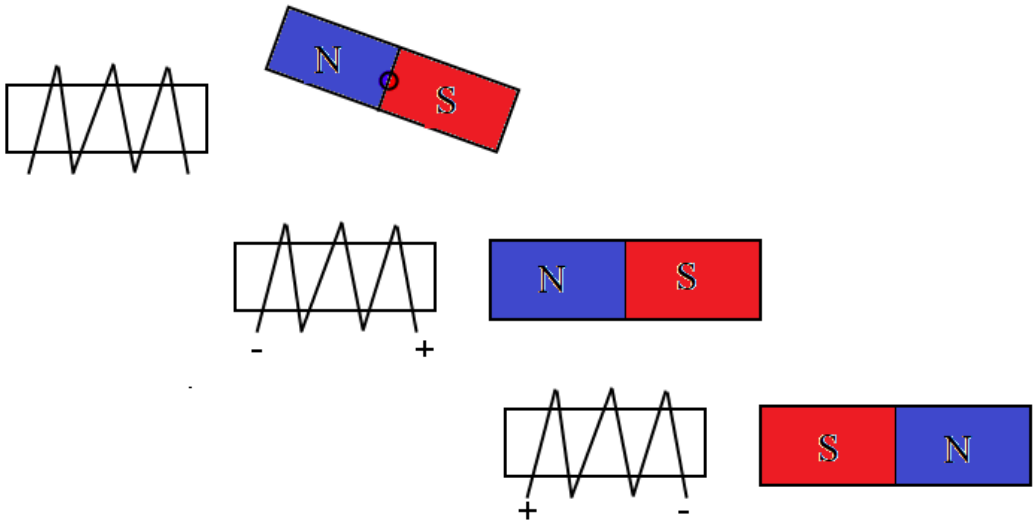



Рис. 3.5. До пояснення принципу роботи крокового двигуна

Якщо доповнити таку схему ще одним електромагнітом, розташованим вертикально, то постійний магніт матиме вже чотири стійких стани, а його крок складатиме 90° (рис. 3.6). За аналогічним принципом обертаються і магніти, що

розташовані в середині крокового двигуна, та відбувається обертання валу двигуна на певний кут.

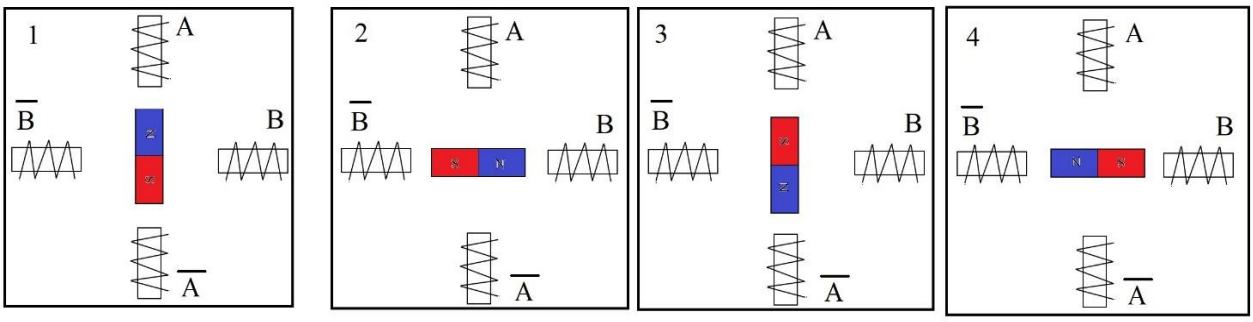



Рис. 3.6. До пояснення принципу роботи крокового двигуна

Якщо двигун має дві незалежні обмотки не з'єднані в центрі, то він називається біполярним. Окрім біполярних крокових двигунів розрізняють також уніполярні та чотири-обмоткові. Розглянемо як за допомогою МК можна керувати обертанням валу біполярного двигуна. Для прикладу використаємо двигун моделі «ST-PM35-15» та мікроконтролер ATMega8. Номінальна напруга такого двигуна складає 12 В, опір обмотки 4 Ом, а кут кроку дорівнює 7.5° (тобто на один повний оберт валу необхідно здійснити 360° / 7.5 = 48 кроків). У технічній документації двигуна пропонується таблиця, що відображає послідовність перемикання обмоток двигуна (табл. 3.2).

*Таблиця 3.2.*

**Послідовність перемикання обмоток двигуна «ST-PM35-15»**

| **Lead Wire Color** | **Terminal Code** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| Black | 1 | – | – | | |
| Orange | 2 | | – | – | |
| Brown | 3 | | | – | – |
| Yellow | 4 | – | | | – |

У відповідності до таблиці для керування двигуном достатньо почергово подавати напругу на відповідні дроти. Символ «–» відповідає рівню логічної одиниці, а пуста клітинка в таблиці – рівню логічного нуля. Створимо елементарний код програми, яка циклічно виконуватиме такий алгоритм дій: 1) здійснити перший крок двигуном, 2) зачекати одну секунду, 3) здійснити другий крок двигуном, тощо.

```
#include <mega8.h>
#include <delay.h>
void main(void)
{
PORTD=0x00;
DDRD=0x0F;
while (1)
    {
    PORTD=0b00000011; //+a +b
    delay_ms(1000);
    PORTD=0b00000110; //+b -a
    delay_ms(1000);
    PORTD=0b00001100; //-a -b
    delay_ms(1000);
    PORTD=0b00001001; //-b +a
    delay_ms(1000);
    }
}
```

Результат виконання такої програми показано на рис. 3.7. Як видно з рисунку, МК здійснює почергове перемикання обмоток двигуна, і, як наслідок, – його покрокове обертання на заданий кут.

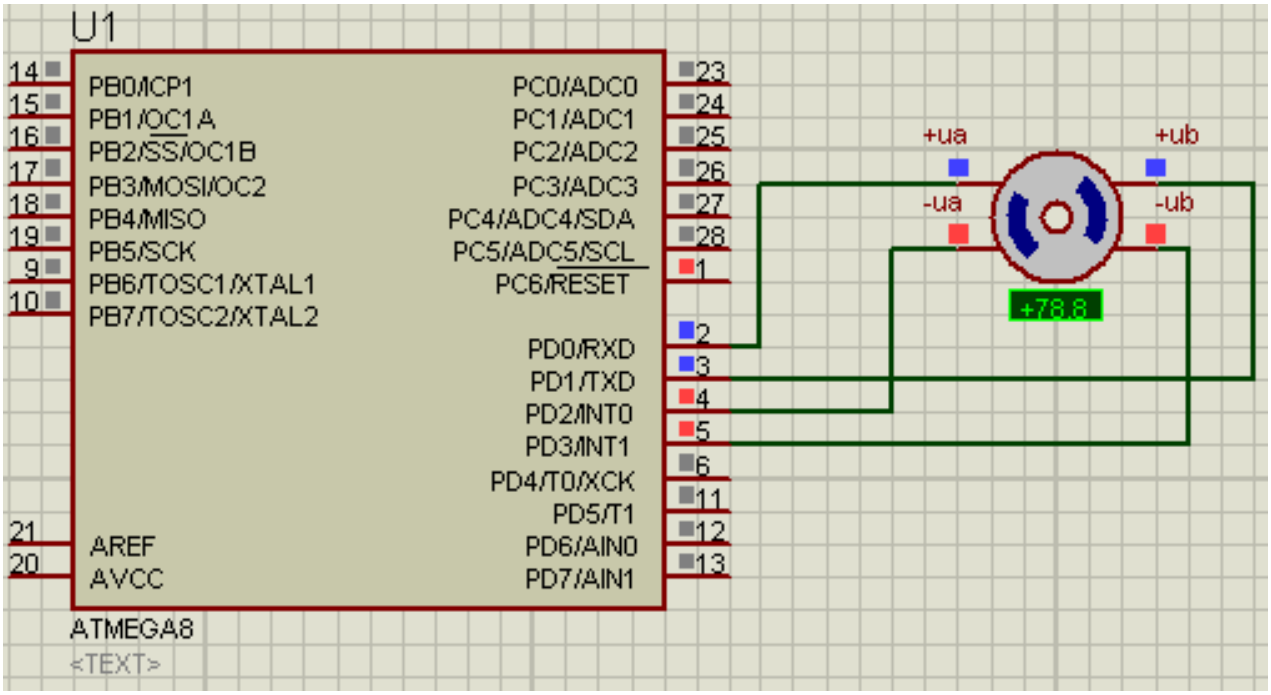



Рис. 3.7. Результат виконання коду програми у системі Proteus

Проте практично схема, показана на рис. 3.7 у такому виконанні, не зможе нормально працювати. Адже величини струму, яку зможе генерувати ніжка МК буде не достатньо для перемикання обмоток двигуна. У такому випадку для реалізації поставленого завдання до кожної ніжки МК, яка задіяна у керуванні двигуном, необхідно підключити додаткову схему з двома транзисторами, індуктивністю та захисним діодом.

Альтернативним рішенням є використання так званого «драйверу двигуна». Це спеціальна мікросхема, у конструкції якої вже реалізовано відповідні транзистори та діоди, а користувачу необхідно лише подавати набір керуючих імпульсів від МК для керування рухом двигуна в ту чи іншу сторону.

Прикладом такого драйверу є мікросхема L293D. Вона включає в собі одразу два драйвери для керування як кроковими двигунами, так і двигунами постійного струму та має такі основні характеристики:

- Окремі канали для логічної та силової частини напруги живлення двигунів (живлення логічної частини в межах від 4.5 В до 5 В (ніжка VSS на мікросхемі), а живлення силової частини двигунів від 4.5 В до 36 В вольт (ніжка VS на мікросхемі)).
- Генерування 600 мА на кожен вихідний канал.
- Піковий (максимальний) струм на виході до 1.2 А.
- Невибагливість мікросхеми до перепадів напруги вхідних сигналів, що подаються на ніжки INPUT. Логічний «0» розпізнається мікросхемою при вхідній напрузі менше 1.5 В. А логічна «1» при напрузі в діапазоні від 2.3 В до 7 В.
- Діапазон робочих температур від -40 ° С до + 150 ° С.
- Швидкість перемикання мікросхеми до 5 кГц.

Для подальшої зручності умовно позначимо драйвери мікросхеми як «правий борт» і «лівий борт» (рис. 3.8) та окреслимо принцип її функціонування. Розглянемо як працює «лівий борт» мікросхеми на прикладі двигуна постійного струму. Ніжка «ENABLE1» – це основний канал, що дозволяє роботу лівого борту мікросхеми (рис. 3.8).

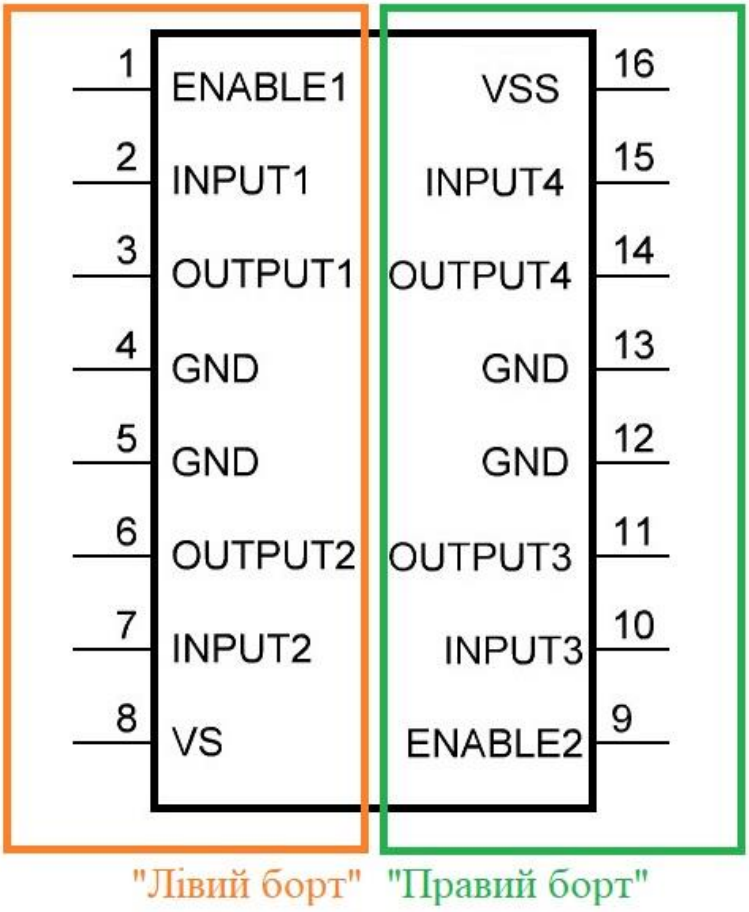



Рис. 3.8. До пояснення принципу роботи мікросхеми L293D

Без логічної одиниці на цьому виводі мікросхема не реагуватиме на жодну із комбінацій на входах «INPUT1» і «INPUT2» і, як наслідок, – не здійснюватиме керування двигуном. Ніжки «INPUT1» і «INPUT2» призначені для керування напрямом обертання валу двигуна. Умовно їх можна порівняти з кермом автомобіля: для повороту валу праворуч, необхідно подати логічну одиницю на «INPUT1», а на «INPUT2» – логічний нуль. Для зміни напрямку руху валу в зворотну сторону навпаки: на ніжку «INPUT1» – логічний нуль, а на «INPUT2» – логічну одиницю. При подачі однакових логічних рівнів (двох лог. «1», або двох лог. «0») на входи «INPUT1» та «INPUT2» двигун обертатися не буде, адже не можливо повернути кермо одразу в двох напрямках.

Варто зазначити, що обертання валу двигуна можна зупинити також подачею логічного нуля на ніжку «ENABLE1», при будь-якій конфігурації входів «INPUT1» та «INPUT2».

Виводи «OUTPUT1» і «OUTPUT2» мікросхеми служать для підключення мотора двигуна, а ніжки «GND» – під'єднуються до полюсу джерела живлення

(землі). «Правий канал» мікросхеми працює абсолютно ідентично. Аналогічним чином, підключивши до виходів «OUTPUT» обмотки крокового двигуна, можна керувати напрямом обертання його валу (рис. 3.9) шляхом комбінації логічних рівнів на входах «INPUT», регулюючи необхідну кількість кроків та швидкість перемикання обмоток.

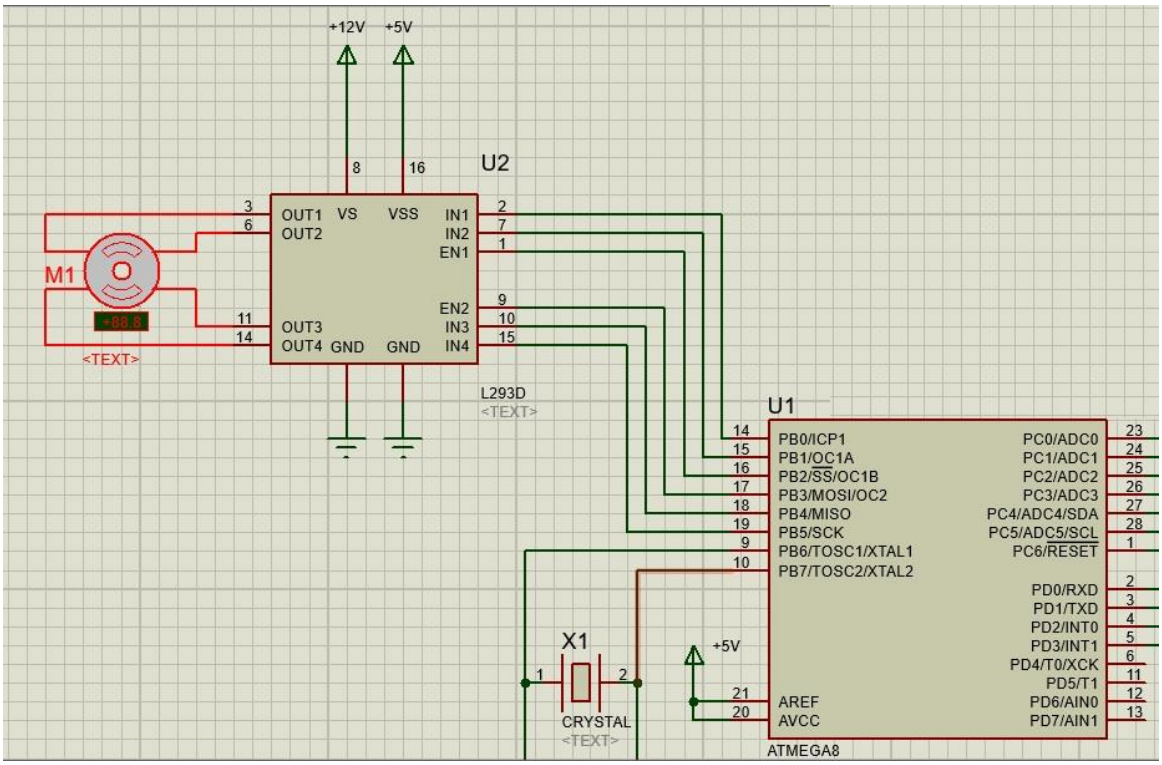



Рис. 3.9. До пояснення принципу роботи мікросхеми L293D

## 3.2. Завдання для виконання

Створити робочу папку, яка буде мати наступну назву: **MP1_PR3_Шифр групи_Прізвище студента_№варіанту** (де PR3 розшифровується як: «Практична робота №3»). **Назва папки задається латиницею (англомовними символами)!**

**Завдання 3.1.** Розробити електричну схему для подальшого моделювання роботи мікроконтролеру та периферійних пристроїв, відповідно з початковими умовами, що наведені у табл. 3.3.

Розрахувати номінали резисторів (їх опір та потужність), які необхідно використати у розробленій схемі для обмеження робочого струму на світлодіодах 14-ти сегментного індикатору. Розрахунок проводиться відповідно

до робочих параметрів використаних електронних компонентів (а саме: робочої напруги у схемі (5 В) та струму світлодіодів для сегментів індикатору).

Знайти і зазначити у протоколі значення робочих частот, на яких повинна відбуватись генерація звукового сигналу з мембрани п'єзорезонатору (відповідно до індивідуального завдання у табл. 3.3). Розрахувати кількість кроків, що необхідна для повного оберту валу двигуна (на 360°), базуючись на параметрах кута кроку $\varphi_A$ у табл. 3.3.

**Порядок виконання Завдання 3.1.**

1. Відкрити робоче вікно середовища Proteus та обрати з бази електронних компонентів необхідний перелік (у відповідності до номеру варіанту в таблиці 3.3). Розмістити обрані компоненти на робочому полі, з'єднати провідниками та задати їх робочі параметри (у відповідності до індивідуального завдання) з метою подальшого моделювання схеми. Місце розташування компонентів на схемі обирається студентом самостійно.

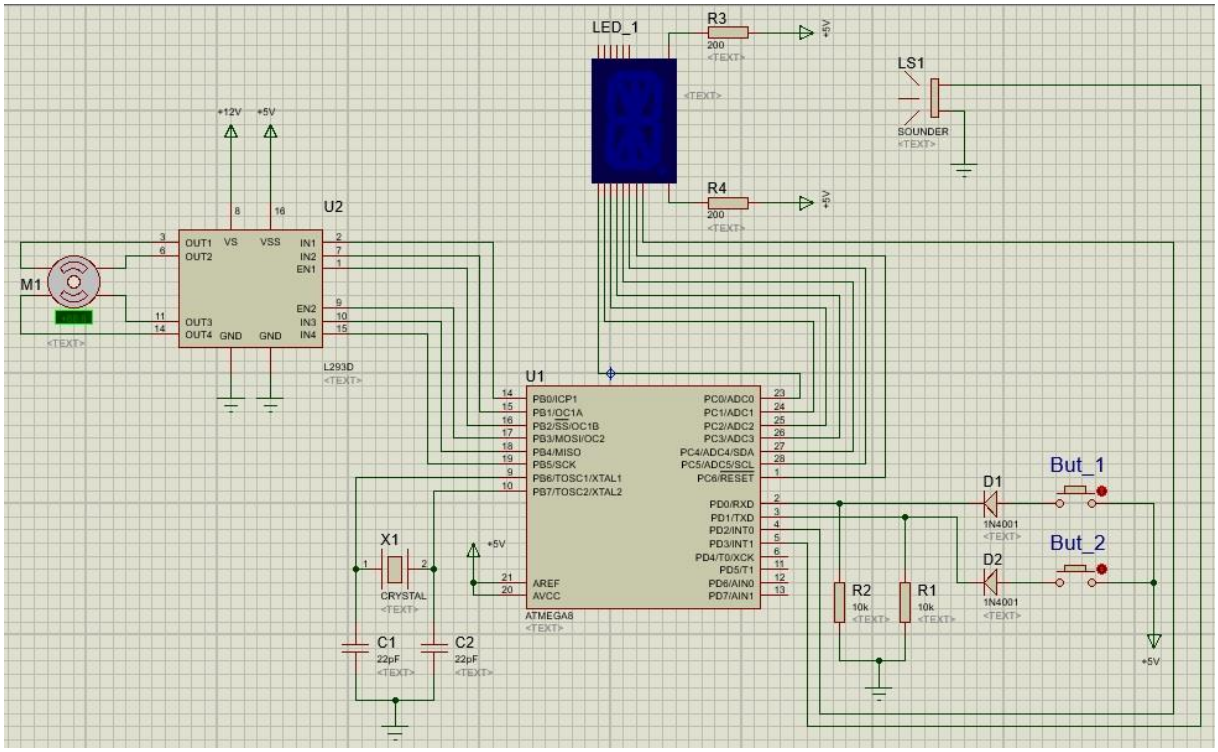



Рис. 3.10. Приклад створеної електричної схеми та розташування компонентів обраних з бази

2. Після завершення налаштувань натиснути кнопку «Generate program, save and exit» та зберегти три файли проекту (усі з однаковою назвою) у раніше

створеній робочій папці (в якій вже розміщено Proteus-файл). **Назва файлу задається лише латиницею (англомовними символами)!**

3. Задати кут кроку двигуна. Для цього: на схемі, створеній у середовищі Proteus, натиснути ЛКМ два рази по моделі двигуна та у вікні «Edit Component» знайти рядок «Step Angle», у якому і необхідно вказати значення параметру $\varphi_A$, відповідно до індивідуального завдання.

4. Зберегти проект розробленої схеми у раніше створеній робочій папці із назвою: «PR3_Шифр групи_№ варіанту». **Назва файлу задається лише латиницею (англомовними символами)!**

**Завдання 3.2.** Провести налаштування та створити програмний код для МК відповідно до індивідуального завдання та початкових умов, наведених у табл. 3.3.

**Порядок виконання Завдання 3.2.**

1. Відкрити робоче вікно середовища CodeVisionAVR, створити новий файл проекту та провести налаштування МК відповідно до параметрів, вказаних у табл. 3.3.

2. Після збереження проекту у основному вікні (яке відкриється програмою CodeVisionAVR автоматично) написати програмний код, який би забезпечував виконання наступних умов (див. рис. 3.10):

- Якщо жодну із кнопок («But_1» або «But_2») не активовано, то периферійні пристроїв повинні бути вимкнені.
- Якщо активовано кнопку «But_1», впродовж $t$ секунд генерувати на п'єзорезонаторі «LS1» звуковий сигнал ноти $NT_1$, після чого розпочати обертання валу двигуна «M1» праворуч, демонструючи на індикаторі «LED_1» напрям руху його. Після виконання $N$ повних обертів, зупинити двигун і вивести на індикатор кількість виконаних обертів.
- Якщо активовано кнопку «But_2», впродовж $t$ секунд генерувати на п'єзорезонаторі «LS1» звуковий сигнал ноти $NT_2$, після чого розпочати обертання валу двигуна «M1» ліворуч, демонструючи на

індикаторі «LED_1» напрям руху його. Після виконання $N$ повних обертів, зупинити двигун і вивести на індикатор кількість виконаних обертів.

- Впродовж всього робочого циклу схеми перевіряти стан МК за допомогою Watchdog-таймеру, Time-Out якого складає $t_{out}$.

**Звернути увагу:** оскільки у електричній схемі використовуються 14-ти сегментні індикатори із загальним анодом, то для активації конкретних сегментів МК повинен надсилати на відповідні ніжки сигнал логічного «0».

3. Провести компіляцію створеного коду, натиснувши кнопку «Build all project files» на верхній панелі середовища CodeVisionAVR.

4. Імпортувати згенерований «*.cof» файл до МК «U1» та задати робочу частоту МК (відповідно до номеру варіанту в табл. 3.3). Для цього: обрати у вкладці «CKSEL Fuses» параметр «Ext.RC» та вказати конкретне значення робочої частоти МК у полі «Clock Frequency» замість фрази «Default», задати відповідне значення частоти у параметрах кварцового резонатору «X1».

5. Перевірити достовірність роботи створеного коду шляхом запуску схеми на моделювання.

6. Зберегти раніше створений Proteus-файл та результати моделювання.

7. Оформити звіт до практичної роботи.

## 3.3. Вимоги до оформлення звіту

1. Звіт повинен бути представлений в електронному вигляді. Вихідний файл звіту має зберігатись у форматі **«*.pdf»**.

2. Назва електронного файлу зі звітом задається у форматі: «МП1_ПР3_Шифр групи_Прізвище студента_№варіанту».

3. Структура звіту повинна містити наступні елементи:

- Титульний аркуш.
- Мету роботи.
- Номер варіанту та індивідуальне завдання.
- Електричну схему, розроблену студентом у середовищі Proteus.
- Формули та результати розрахунку номіналу резисторів **(параметри**

**опору та потужності)**, **робочі частоти** (на яких повинна відбуватись генерація звукового сигналу з мембрани п'єзорезонатору), **формулу та результати розрахунку кількості кроків двигуна** (яка необхідна для повного оберту валу).

- Блок-схему алгоритму для робочого коду МК.
- Текст з робочим кодом (повністю, включно з налаштуваннями).
- Зображення з результатами моделювання схеми у середовищі Proteus.
- Висновки по роботі.
- Перелік контрольних запитань.

4. Якість зображення та розмір шрифту на схемах, рисунках і таблицях повинні чітко відображати представлену в них інформацію. Усі рисунки, таблиці, схеми і розділи у звіті повинні бути підписані (вимоги див. у додатку А).

5. Файл звіту додається до загальної робочої папки (див. п. 3.2), яка надсилається викладачу на перевірку. Для проходження перевірки робоча папка повинна містити: Proteus – файл (у форматі «*.DSN») зі створеною електричною схемою, усі файли проекту створеного у середовищі CodeVisionAVR, звіт до практичної роботи.

**Якщо оформлення звіту і робочої папки не відповідають вимогам, практична робота до захисту не допускається!**

## 3.4. Контрольні запитання

1. Що таке переривання? Які є види переривань?
2. Що таке «переривання за збігом» та як воно працює?
3. Як за допомогою МК AVR можна генерувати звукові коливання?
4. Що таке Watchdog – таймер і навіщо він потрібен?
5. Що таке Time Out у сторожовому таймері? На що впливає значення Time Out?
6. У чому полягає принцип роботи крокового двигуна?
7. Від чого залежить кут оберту валу у кроковому двигуні?
8. Що таке драйвер двигуна і навіщо він потрібен?
9. Яке призначення виводів у мікросхемі L293D? Розшифруйте їх.

10. Навіщо в мікросхемі L293D передбачено дві лінії живлення? Як вони позначаються?

11. Які значення напруги мікросхема L293D сприймає як логічну одиницю та логічний нуль?

Таблиця 3.3. Початкові умови до завдання 3.2

| № варі анту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | $t_{out}$ | $t$ | $NT_1$ | $NT_2$ | $N$ | $\varphi_A$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | ATmega16 | 4 Mhz | 19 pf | 14SEG-MPX1-CA-RED | 16 mA | 1N4001 | 0,13 с | 3 с | До | Сі | 5 | 7,5 ° |
| **2** | ATmega32 | 4 Mhz | 19 pf | 14SEG-MPX1-CA-BLUE | 22 mA | 1N4002 | 16,3 мс | 2 с | Ре | Фа | 3 | 9 ° |
| **3** | ATmega32 | 3 Mhz | 17 pf | 14SEG-MPX1-CA-RED | 28 mA | 1N4003 | 32,5 мс | 1 с | Мі | Ля | 2 | 15 ° |
| **4** | ATmega8535 | 1 Mhz | 15 pf | 14SEG-MPX1-CA-RED | 10 mA | 1N4004 | 65 мс | 4 с | Фа | Соль | 4 | 5 ° |
| **5** | ATmega8535 | 1 Mhz | 22 pf | 14SEG-MPX1-CA-GRN | 19 mA | 1N4005 | 0,26 с | 3 с | Сі | До | 6 | 20 ° |
| **6** | AT90S8535 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-BLUE | 14 mA | 1N4006 | 1,0 с | 1 с | Фа | Сі | 3 | 40 ° |
| **7** | AT90S8535 | 2 Mhz | 15 pf | 14SEG-MPX1-CA-GRN | 11 mA | 1N4007 | 2,1 с | 4 с | Ре | Мі | 8 | 45 ° |
| **8** | ATmega16 | 3 Mhz | 18 pf | 14SEG-MPX1-CA-BLUE | 10 mA | 1N4004 | 0,26 с | 2 с | Ля | Фа | 5 | 90 ° |
| **9** | AT90S8535 | 3 Mhz | 20 pf | 14SEG-MPX1-CA-RED | 20 mA | 1N4005 | 16,3 мс | 4 с | Мі | До | 4 | 60 ° |
| **10** | AT90S8535 | 1 Mhz | 15 pf | 14SEG-MPX1-CA-BLUE | 25 mA | 1N4002 | 2,1 с | 2 с | Сі | До | 2 | 30 ° |
| **11** | ATmega8535 | 1 Mhz | 15 pf | 14SEG-MPX1-CA-RED | 10 mA | 1N4003 | 0,52 с | 5 с | Ре | Соль | 10 | 15 ° |
| **12** | ATmega8535 | 4 Mhz | 22 pf | 14SEG-MPX1-CA-GRN | 19 mA | 1N4004 | 16,3 мс | 2 с | До | Ре | 7 | 5 ° |
| **13** | AT90S8535 | 4 Mhz | 20 pf | 14SEG-MPX1-CA-BLUE | 14 mA | 1N4007 | 0,13 с | 1 с | Мі | Сі | 3 | 12 ° |
| **14** | ATmega16 | 2 Mhz | 19 pf | 14SEG-MPX1-CA-RED | 16 mA | 1N4006 | 65 мс | 3 с | Ля | Соль | 5 | 18 ° |

Продовження таблиці 3.3

| № варі анту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | $t_{out}$ | $t$ | $NT_1$ | $NT_2$ | $N$ | $\varphi_A$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | ATmega16 | 2 Mhz | 22 pf | 14SEG-MPX1-CA-BLUE | 20 mA | 1N4005 | 0,26 с | 3 с | Сі | До | 6 | 40 ° |
| **16** | ATmega32 | 3 Mhz | 15 pf | 14SEG-MPX1-CA-GRN | 10 mA | 1N4006 | 1,0 с | 1 с | Фа | Сі | 3 | 45 ° |
| **17** | ATmega16 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-RED | 30 mA | 1N4007 | 0,13 с | 3 с | До | Сі | 5 | 90 ° |
| **18** | AT90S8535 | 1 Mhz | 18 pf | 14SEG-MPX1-CA-BLUE | 15 mA | 1N4005 | 65 мс | 3 с | Ля | Соль | 5 | 60 ° |
| **19** | AT90S8535 | 4 Mhz | 15 pf | 14SEG-MPX1-CA-RED | 25 mA | 1N4001 | 0,26 с | 3 с | Сі | До | 6 | 7,5 ° |
| **20** | ATmega32 | 4 Mhz | 16 pf | 14SEG-MPX1-CA-GRN | 12 mA | 1N4002 | 1,0 с | 1 с | Фа | Сі | 3 | 9 ° |
| **21** | ATmega16 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-GRN | 15 mA | 1N4003 | 32,5 мс | 1 с | Мі | Ля | 2 | 15 ° |
| **22** | ATmega16 | 2 Mhz | 20 pf | 14SEG-MPX1-CA-GRN | 15 mA | 1N4004 | 2,1 с | 4 с | Ре | Мі | 8 | 5 ° |
| **23** | ATmega8535 | 3 Mhz | 20 pf | 14SEG-MPX1-CA-RED | 15 mA | 1N4004 | 0,26 с | 3 с | До | Сі | 5 | 60 ° |
| **24** | ATmega32 | 3 Mhz | 15 pf | 14SEG-MPX1-CA-GRN | 10 mA | 1N4007 | 32,5 мс | 1 с | Ля | Соль | 5 | 30 ° |
| **25** | ATmega32 | 1 Mhz | 22 pf | 14SEG-MPX1-CA-BLUE | 15 mA | 1N4006 | 2,1 с | 1 с | Фа | Сі | 8 | 15 ° |

# КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4
# ЗОВНІШНІ ПЕРЕРИВАННЯ. РЕГІСТРИ ЗСУВУ. ЧАСТОТОМІР НА AVR

**Мета:** Ознайомитись з принципом роботи зовнішніх переривань у мікроконтролерах серії AVR та дослідити принципи застосування мікросхем регістрів зсуву.

**Змістовність роботи:** Зовнішні переривання, загальні відомості про зовнішні переривання. Регістри зсуву, мікросхема 74HC595. Реалізація частотоміру на мікроконтролері. Типові налаштування зовнішніх переривань у середовищі CodeVisionAVR. Написання коду програми для МК і використання вбудованих функцій середовища Proteus для моделювання роботи електричної схеми.

## 4.1. Теоретичні відомості

### 4.1.1. Зовнішні переривання. Загальні відомості

Однією із особливостей мікроконтролерів AVR є те, що вони можуть тимчасово зупиняти виконання робочої програми внаслідок виникнення деякої зовнішньої події. Така подія називається **перериванням**, а залежно від того, виникає вона по команді від внутрішніх периферійних пристроїв чи від зовнішніх подій, розрізняють два види **джерел переривань**: зовнішні та внутрішні.

При подачі сигналу на відповідну ніжку МК, яка відповідає за зовнішні переривання (ЗП), виконання основної програми призупиниться, а натомість мікроконтролер почне обробляти програмний код, який буде записано у функції обробки зовнішніх переривань. Після повного виконання коду, розміщеного у тілі такої функції, МК продовжить виконання основної програми з місця, де було перервано роботу програми зовнішнім перериванням. Кількість ніжок МК, що призначені для обробки зовнішніх переривань та їх розташування відрізняється для кожного конкретного мікроконтролера. Наприклад, МК ATMega8 має у своєму складі дві таких ніжки, а МК ATMega16 – три.

Назви виводів, що призначені для ЗП, починаються з абревіатури INT (рис. 4.1). Наприклад, INT0, INT1 тощо.

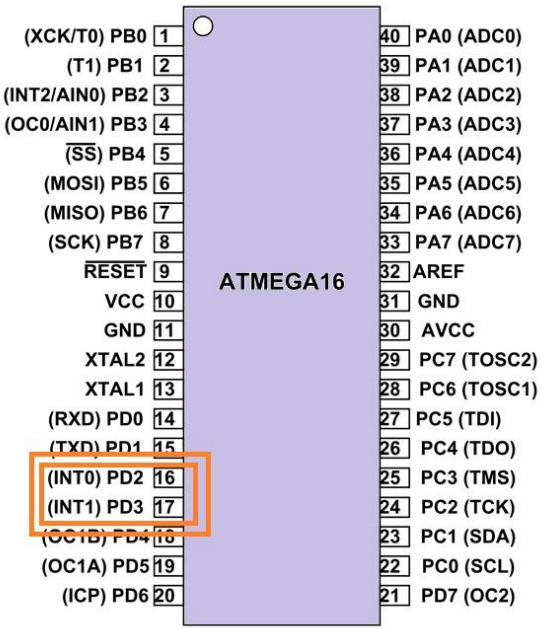



Рис. 4.1. Приклад позначення виводів МК ATMega16, призначених для обробки ЗП

Для активації зовнішнього переривання може використовуватись чотири види вхідних сигналів: Rising Edge, Falling Edge, Low level, Any change (рис. 4.2).

- Rising Edge – зростаючий фронт сигналу (перехід із лог. 0 в лог. 1).
- Falling Edge – спадаючий фронт сигналу (перехід із лог. 1 в лог. 0).
- Low level – сигнал низького рівня (логічного нуля).
- Any change – будь-які зміни фронту сигналу.

Кожне зовнішнє переривання має фіксований набір параметрів, що відповідають за його обробку та налаштування. А саме:

- **Вектор переривання** – комірка в області пам'яті програми, у якій зберігається адреса підпрограми для обробки переривання. Чим меншою є адреса вектору переривання, тим вищим буде його пріоритет. Тобто, наприклад, якщо у одній системі активовано зовнішні переривання INT0 та INT1, то спочатку МК обробить переривання INT0, а лише потім INT1. Адже адреса INT0 є меншою і, як наслідок, таке переривання має пріоритет над перериванням INT1.

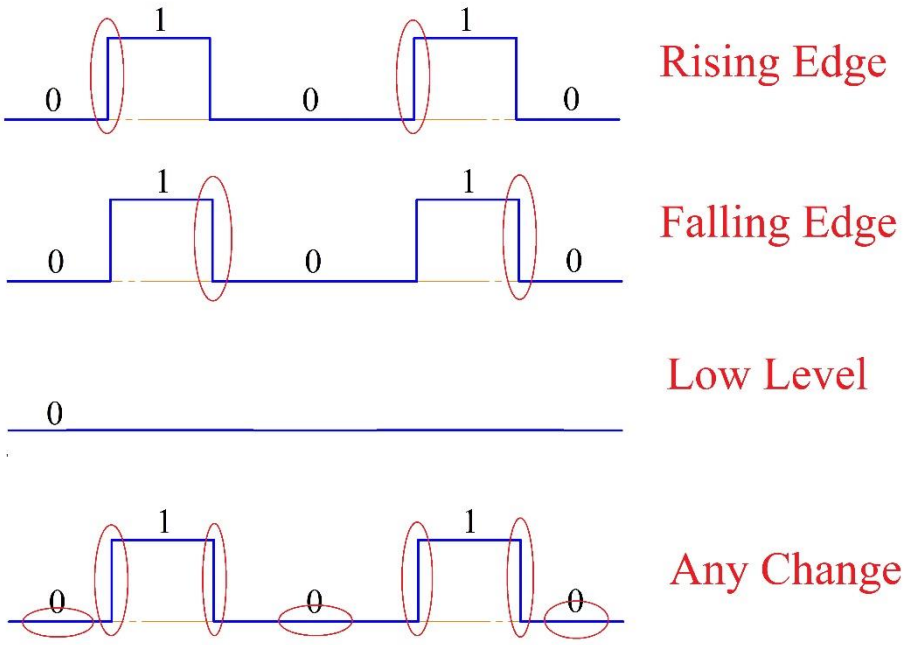



Рис. 4.2. Можливі види сигналів для активації зовнішніх переривань МК

Окрім того, варто зазначити, що зовнішні переривання у МК серії AVR мають вищий пріоритет, ніж переривання по таймеру/лічильнику або інші програмно створені затримки у коді МК. Тобто, якщо у мікроконтролері, у якому реалізовано обробку зовнішніх і внутрішніх переривань, буде одночасно активовано дві події, то робоча програма МК спочатку обробить зовнішнє переривання, а вже потім – внутрішнє.

- **Біт дозволу** – біт, який потрібно встановлювати у стан логічної 1, для отримання дозволу на обробку переривання.
- **Прапор переривання –** біт, який вказує на те, чи відбулося переривання.
- **Стек** – область пам'яті, яку центральний процесор МК використовує для зберігання і відновлення адреси повернення з підпрограм. Тобто після обробки переривання за допомогою даних, які записані у стек, відбувається повернення до ділянки коду основної програми, на якій зупинився МК перед обробкою ЗП.

Для налаштування зовнішніх переривань у середовищі CodeVisionAVR необхідно у вкладках вікна CodeWizard обрати пункт External IRQ та вказати номер зовнішнього переривання, яке повинно бути активовано (рис. 4.3). Після чого користувачу надається можливість обрати з випадаючого списку один із

чотирьох типів сигналів, що можуть бути використані для активації зовнішнього переривання (див. рис. 4.2).

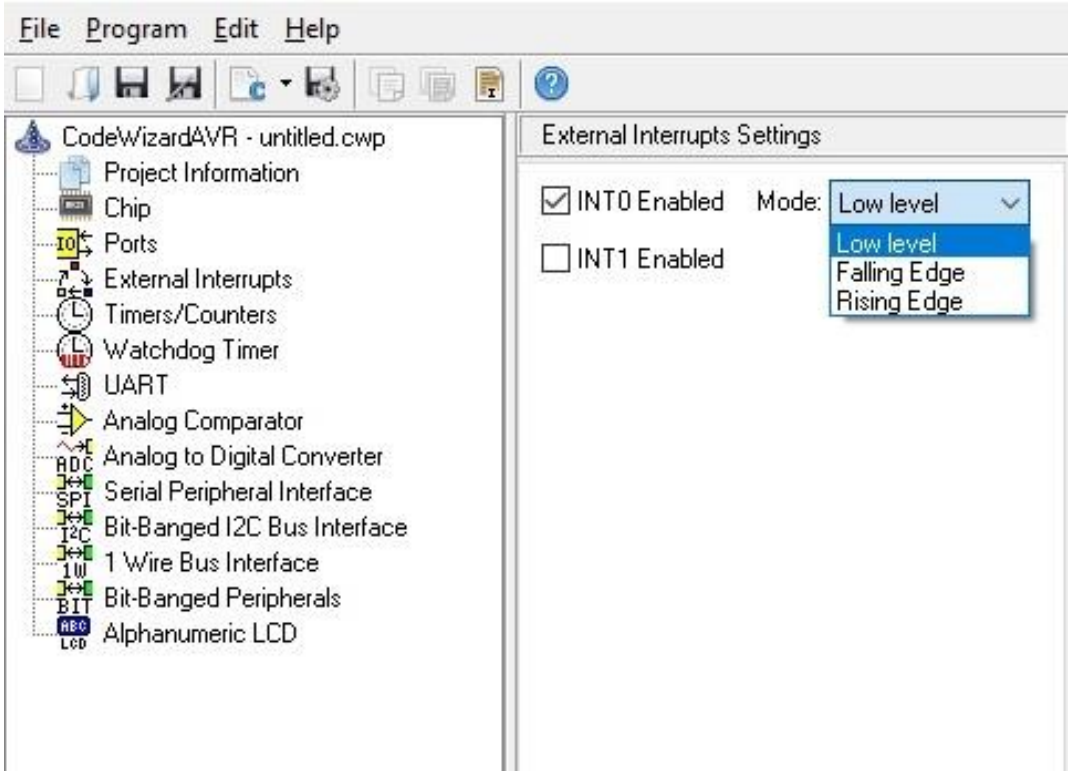



Рис. 4.3. Вікно налаштування зовнішніх переривань у CodeVisionAVR

Після вибору необхідних налаштувань у коді МК, який згенерував CodeWizard, з'явиться декілька нових блоків коду. Зокрема:

Блок коду для обслуговування зовнішніх переривань

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
}
```

Саме в середині такої функції повинен прописуватись код, який буде виконуватись МК після того, як на вхід INT0 надійде сигнал зовнішнього переривання.

Блок налаштування зовнішніх переривань

```
// External Interrupt(s) initialization
// INT0: On
```

```
// INT0 Mode: Rising Edge
// INT1: Off
GICR|=(0<<INT1) | (1<<INT0);
MCUCR=(0<<ISC11) | (0<<ISC10) | (1<<ISC01) | (1<<ISC00);
GIFR=(0<<INTF1) | (1<<INTF0);
```

Тут GICR (або GIMSK) – загальний регістр масок переривань, який забороняє (виконує маскування) або дозволяє зовнішні переривання, MCUCR – регістр керування ядром МК, GIFR – загальний регістр прапорів переривань, що містить прапори зовнішніх переривань. Також при налаштуванні зовнішніх переривань використовуються SREG – регістр стану, SPH та SPL – регістри, що вказують початок стека.

### 4.1.2. Регістри зсуву. Мікросхема 74HC595

При роботі з мікроконтролерами та мікропроцесорною технікою поширеними є ситуації, при яких стандартної кількості виводів (ніжок) МК не достатньо для реалізації поставлених завдань. У такому випадку постає питання, як можна збільшити кількість доступних виводів, не змінюючи при цьому модель контролеру. Одним з можливих рішень для такого завдання є використання регістрів зсуву.

Як відомо з теоретичного курсу, **регістр зсуву** – це пристрій, що складається з декількох послідовно з'єднаних тригерів, кількість яких визначає розрядність регістру. У мікропроцесорній техніці найчастіше застосовуються восьмирозрядні версії регістрів зсуву, основним призначенням яких є зміщення даних (які було подано на вхід регістру) з попереднього розряду на наступний по порядку. Розглянемо принцип роботи восьмирозрдяного регістру зсуву на прикладі мікросхеми 74HC595 (рис. 4.4).

Набір входів та виходів, що присутні у регістрі, має наступне призначення:

- SH_CP – тактовий сигнал.
- DS – шина даних.
- ST_CP – засув.
- $\overline{MR}$ – для скидання мікросхеми (активується низьким рівнем сигналу).

- $\overline{OE}$ – вхід, призначений для заборони або дозволу роботи мікросхеми (дозвіл - низький рівень сигналу, заборона – високий).
- Q0, Q1...Q7 – виходи регістру (його розряди).
- Q7' – вихід для каскадування.
- VCC і GND – лінії живлення.

Після ввімкнення мікросхеми на вхід DS надходить сигнал логічної одиниці або логічного нуля. Під час першої ітерації дані зі входу DS передаються на вихід Q0. Під час другої ітерації дані зі входу DS знову надійдуть на вихід Q0, але результат, який було записано в Q0 на попередньому етапі, переміститься на вихід Q1.

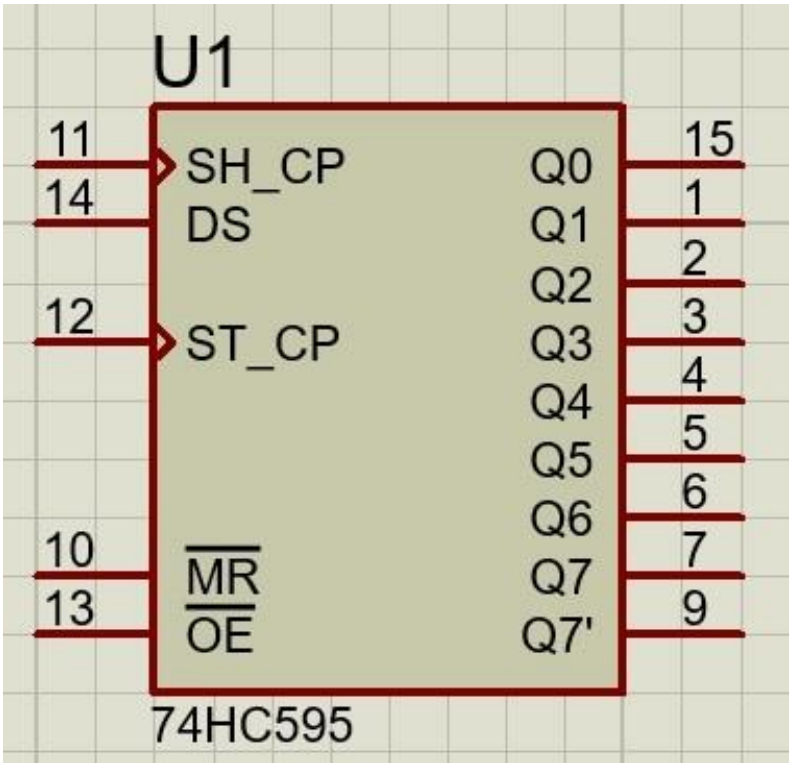



Рис. 4.4. Позначення мікросхеми 74HC595 в системі Proteus

Впродовж третьої ітерації дані зі виходу Q1 зсунуться на Q2, а дані зі виходу Q0 – на вихід Q1. Такий процес триватиме циклічно з постійним зсувом попереднього значення на наступний розряд. Саме тому такий регістр і називається зсувним.

Часові інтервали для зсуву розрядів задаються імпульсами, що повинні надсилатись на вивід SH_CP мікросхеми. У випадку коли в певний момент часу дані в регістрі не потрібно зсовувати на наступний розряд, то достатньо

припинити подачу імпульсів на ніжку SH_CP, зупинивши тим самим циклічний процес зсуву.

Ніжка ST_CP (або так звана «засувка») відповідає за зберігання на виходах Q0-Q7 мікросхеми одних і тих же значень незалежно від змін, що відбуваються на вході DS. Припустимо за умовою, що необхідно засвітити LED-індикатор, який підключено до виходу Q7 мікросхеми, але при цьому – не припустимо щоб на виходах Q0-Q6 з'являлась логічна одиниця. З підключенням «засувки» це питання вирішується легко. Доки на ніжку ST_CP подано сигнал логічної одиниці, «засувка» буде активною. Тобто дані, що будуть подаватись на вхід DS зсуватимуться порозрядно, але при цьому на виходах Q0-Q7 стан виводів не змінюватиметься. Але, як тільки на ніжку ST_CP надійде сигнал логічного нуля, «засувку» буде знято, а записані в розряди дані з'являться на виході.

Окрім того, у мікросхемі 74HC595 реалізовано можливість каскадного підключення. Якщо ніжку Q7' першої мікросхеми 74HC595 підключити до входу DS другого регістру 74HC595, то після зсуву даних з ніжки Q7 першого регістру, дані надійдуть на вихід Q0 другої мікросхеми. Загалом, з опису роботи зрозуміло, що швидкість роботи такої мікросхеми є невисокою. Однак, подібне рішення є зручним для збільшення кількості функціональних виводів контролеру та дозволяє реалізовувати управління значною кількістю сигнальних ліній за допомогою лише декількох ніжок МК.

**Приклад.** Припустимо, що для реалізації деякого завдання інженеру необхідно використати вісім семисегментних індикаторів. На чотири з них повинен виводитись час роботи установки, а на чотири наступні – виміряна температура в цеху.

Навіть при використанні мікроконтролерів типу ATMega16, 32, 64 тощо знадобиться як мінімум сім ніжок контролеру для виведення даних і ще вісім – для перемикання між індикаторами. У такому випадку ресурси мікроконтролеру використовуватимуться неефективно. Для поставленого завдання доцільніше використати простіший МК і всього три виводи його портів шляхом підключення регістру зсуву, який може керуватись за допомогою трьох сигнальних ліній, і, тим самим, зменшує навантаження на мікропроцесор.

### 4.1.3. Частотомір на AVR

Розглянемо як, використовуючи зовнішні переривання МК AVR та регістр зсуву, можна реалізувати схему для вимірювання частоти сигналу. Для прикладу використаємо мікроконтролер ATmega8, регістр 74HC595, а також генератор сигналу та індикатор лічильника у системі Proteus (рис. 4.5). Припустимо, що на один із входів МК, який відповідає за обробку ЗП (наприклад INT0), подається постійно пульсуючий сигнал з певною частотою. Якщо налаштувати вхід INT0 на перемикання по зростаючому фронту (Rising Edge), то з періодом сигналу T при появі кожного нового імпульсу в МК буде викликатись зовнішнє переривання. Як наслідок, якщо вести постійний підрахунок кількості переривань за секунду, можна визначити і частоту вхідного сигналу.

Для цього налаштуємо Timer1 мікроконтролера на рахунок з частотою один раз за секунду (див. п. 2.1.3 у практичній роботі №2). Якщо у момент, коли таймер спрацював, виконувати скидання лічильника та виводити результат на індикатори, то можна отримати систему, що буде вимірювати частоту вхідного сигналу. Водночас, для візуалізації розрахованого значення частоти вхідного сигналу можна використати декілька однорозрядних (або багаторозрядних) семисегментних індикаторів, що підключені до МК через регістр зсуву. Приклад програмного коду для частотоміра виглядатиме наступним чином:

```
#include <mega8.h>
unsigned long i=0, frequency = 0; // змінні для підрахунку кількості імпульсів та секунд

// Блок обробки зовнішнього переривання
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
 i++;
}

// Блок обробки переривання за збігом
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
```

```
{
frequency=i;
i=0;
TCNT1H=0x00;
TCNT1L=0x00; // скинути таймер (регістр підрахунку імпульсів)
}
void main(void)
{
…

#asm("sei")

while (1)
 {
        // Код основної програми
        // Вивести значення змінної для підрахунку кількості імпульсів на індикатор
 };
}
```

## 4.2. Завдання для виконання

Створити робочу папку, яка буде мати наступну назву: **MP1_PR4_Шифр групи_Прізвище студента_№варіанту** (де PR4 розшифровується як: «Практична робота №4»). **Назва папки задається латиницею (англомовними символами)!**

**Завдання 4.1.** Розробити електричну схему для подальшого моделювання роботи частотоміру, використовуючи мікроконтролер AVR, регістр зсуву 74HC595 та інші вбудовані елементи системи Proteus, у відповідності до початкових умов, що наведені у табл. 4.1.

Розрахувати номінали резисторів (їх опір та потужність), які необхідно використати у розробленій схемі для обмеження робочого струму на світлодіодах семисегментного індикатору. Розрахунок проводиться відповідно до робочих параметрів використаних електронних компонентів (а саме: робочої напруги у схемі (5 В) та струму світлодіодів для сегментів індикатору).

Провести попередній розрахунок частоти вхідного сигналу відповідно до індивідуальних завдань, наведених у табл. 4.1.

**Порядок виконання Завдання 4.1.**

1. Відкрити робоче вікно середовища Proteus та обрати з бази електронних компонентів необхідний перелік (у відповідності до номеру варіанту в таблиці 4.1).

2. Розмістити обрані компоненти на робочому полі, з'єднати їх провідниками та задати робочі параметри (у відповідності до індивідуального завдання) для подальшого моделювання схеми. Місце розташування компонентів на схемі обирається студентом самостійно. Приклад схеми наведено на рис. 4.5.

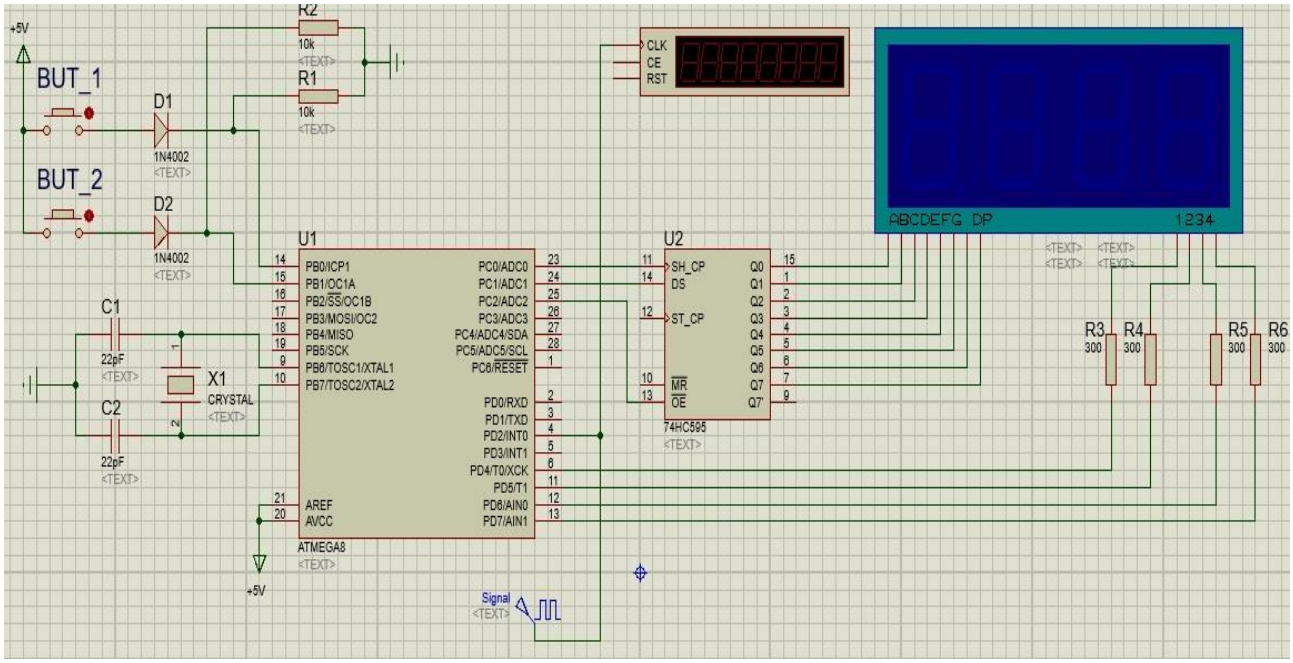



Рис. 4.5. Приклад створеної електричної схеми та розташування компонентів, обраних з бази

3. Зберегти проект розробленої схеми у раніше створеній робочій папці із назвою: «PR4_Шифр групи_№ варіанту». **Назва файлу задається лише латиницею (англомовними символами)!**

**Завдання 4.2.** Провести налаштування та створити програмний код для МК відповідно до індивідуального завдання та початкових умов, наведених у табл. 4.1.

**Порядок виконання Завдання 4.2.**

1. Відкрити робоче вікно середовища CodeVisionAVR, створити новий

файл проекту та провести налаштування МК у відповідності до заданих параметрів. Для цього необхідно виконати наступні кроки:

- На верхній робочій панелі середовища CodeVisionAVR натиснути кнопку «Create a new file or project» → «New Project» → «Yes» → «AVR8(ATtiny, ATmega, AT90)» → «Ok».
- У вкладці «Chip» обрати зі списку модель МК відповідно до індивідуального завдання (рядок «Chip») та робочу частоту МК (рядок «Clock»).
- У вкладці «Ports» провести налаштування I/O-портів на вхід або вихід, залежно від індивідуального завдання.
- У вкладці «Timers» обрати режим роботи і задати параметри першого таймера/лічильника.
- У вкладці «External Interrupts» обрати номер зовнішнього переривання і тип сигналу «Rising Edge».

2. Після завершення налаштувань натиснути кнопку «Generate program, save and exit» та зберегти три файли проекту (усі з однаковою назвою) у раніше створеній робочій папці (в якій вже розміщено Proteus-файл). **Назва файлу задається лише латиницею (англомовними символами)!** Визначити формулу для розрахунку частоти вхідного сигналу, базуючись на прикладі, що показаний у п.п.4.1.3.

3. Після збереження проекту у основному вікні (яке відкриється програмою CodeVisionAVR автоматично) написати програмний код, який би забезпечував виконання наступних умов (див. рис. 4.5):

- Якщо жодну з кнопок («BUT_1» / «BUT_2») не активовано, то на розряди семисегментного індикатору повинен виводитись надпис: «-.-.-.-.».
- Якщо активовано кнопку «BUT_1», то мікроконтролер «U1» повинен виводити на семисегментний індикатор номер варіанту студента (у форматі «НВ.ХХ», де ХХ – номер варіанту. Наприклад, для варіанту №1, надпис матиме вигляд «НВ.01»), що миготить із частотою 1 раз у $t_{d1}$.

- Якщо активовано кнопку «BUT_2», то мікроконтролер «U1» повинен здійснювати підрахунок і виводити на семисегментний індикатор значення частоти вхідного сигналу (параметр $N$ у табл. 4.1).

4. Провести компіляцію створеного коду, натиснувши кнопку «Build all project files» на верхній панелі середовища CodeVisionAVR.

5. Імпортувати згенерований «*.cof» файл до МК «U1» та задати параметри вхідного сигналу (на рис. 4.5 позначено як «Signal») і робочу частоту МК (у відповідності до номеру варіанту в табл. 4.1). Для цього обрати у вкладці «CKSEL Fuses» параметр «Ext.RC» у діапазоні, на якому працює кварцовий генератор, що використаний у схемі. Після цього вказати конкретне значення робочої частоти МК у полі «Clock Frequency» замість фрази «Default».

6. Перевірити достовірність роботи створеного коду шляхом запуску схеми на моделювання.

7. Зберегти раніше створений Proteus-файл та результати моделювання.

8. Оформити звіт до практичної роботи.

## 4.3. Вимоги до оформлення звіту

1. Звіт повинен бути представлений в електронному вигляді. Вихідний файл звіту має зберігатись у форматі **«*.pdf»**.

2. Назва електронного файлу зі звітом задається у форматі: «МП1_ПР4_Шифр групи_Прізвище студента_№варіанту».

3. Структура звіту повинна містити наступні елементи:

- Титульний аркуш.
- Мету роботи.
- Номер варіанту та індивідуальне завдання (відповідно до табл. 4.1).
- Електричну схему, розроблену студентом у середовищі Proteus.
- Формули та результати розрахунку номіналу резисторів **(параметри опору та потужності)**.
- Блок-схему алгоритму для робочого коду МК.
- Текст з робочим кодом (повністю, включно з налаштуваннями МК).
- Зображення з результатами моделювання схеми у середовищі Proteus.
- Висновки по роботі.

- Перелік контрольних запитань.

4. Якість зображення та розмір шрифту на схемах, рисунках і таблицях повинні чітко відображати представлену в них інформацію. Усі рисунки, таблиці, схеми і розділи у звіті повинні бути підписані (вимоги див. у додатку А).

5. Файл звіту додається у загальну робочу папку (див. п. 4.2), яка надсилається викладачу на перевірку. Для проходження перевірки, робоча папка повинна містити: Proteus – файл (у форматі «*.DSN») зі створеною електричною схемою, усі файли проекту, створеного у середовищі CodeVisionAVR, звіт до практичної роботи.

**Якщо оформлення звіту і робочої папки не відповідають вимогам, практична робота до захисту не допускається!**

## 4.4. Контрольні запитання

1. Що таке переривання. Які є види джерел переривання?
2. Яке позначення мають виводи МК AVR призначені для зовнішніх переривань? Яка кількість таких виводів у мікроконтролері?
3. Які існують види сигналів зовнішніх переривань?
4. Що таке стек, біт дозволу, вектор і прапор переривання?
5. Що таке регістр зсуву? Для чого використовуються мікросхеми регістрів зсуву?
6. Яке призначення виводів SH_CP, DS, ST_CP, $\overline{MR}$ та $\overline{OE}$ у мікросхемі 74HC595?
7. Що таке частота і період сигналу?
8. Що розуміють під поняттям «зростаючий фронт» сигналу?
9. Що розуміють під поняттям «спадаючий фронт» сигналу?

Таблиця 4.1. Початкові умови до завдання 4.2

| № варіанту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | $t_{d1}$ | $N$ |
|---|---|---|---|---|---|---|---|---|
| **1** | AT90S8535 | 2 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 25 mA | 1N4001 | 400 мс | 10 Гц |
| **2** | ATmega16 | 3 Mhz | 18 pf | 7SEG-MPX4-CA | 10 mA | 1N4002 | 300 мс | 100 Гц |
| **3** | ATmega8 | 3 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 19 mA | 1N4003 | 800 мс | 300 Гц |
| **4** | ATmega8535 | 1 Mhz | 22 pf | 7SEG-MPX4-CA | 10 mA | 1N4004 | 200 мс | 50 Гц |
| **5** | AT90S8535 | 4 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 19 mA | 1N4005 | 550 мс | 20 Гц |
| **6** | ATmega8 | 2 Mhz | 19 pf | 7SEG-MPX4-CA | 14 mA | 1N4006 | 600 мс | 100 Гц |
| **7** | ATmega8535 | 1 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 16 mA | 1N4007 | 850 мс | 15 Гц |
| **8** | ATmega8 | 2 Mhz | 22 pf | 7SEG-MPX4-CA | 22 mA | 1N4004 | 300 мс | 20 Гц |
| **9** | AT90S8535 | 2 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 28 mA | 1N4005 | 350 мс | 40 Гц |
| **10** | AT90S8535 | 1 Mhz | 15 pf | 7SEG-MPX4-CA | 20 mA | 1N4002 | 400 мс | 25 Гц |
| **11** | ATmega8535 | 1 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 10 mA | 1N4003 | 200 мс | 15 Гц |
| **12** | ATmega32 | 4 Mhz | 15 pf | 7SEG-MPX4-CA | 30 mA | 1N4004 | 100 мс | 70 Гц |
| **13** | ATmega8 | 2 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4007 | 150 мс | 30 Гц |
| **14** | ATmega16 | 1 Mhz | 20 pf | 7SEG-MPX4-CA | 16 mA | 1N4006 | 1000 мс | 10 Гц |

Продовження таблиці 4.1

| № варіанту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | $t_{d1}$ | $N$ |
|---|---|---|---|---|---|---|---|---|
| **15** | ATmega16 | 2 Mhz | 22 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4005 | 1200 мс | 90 Гц |
| **16** | AT90S8535 | 1 Mhz | 18 pf | 7SEG-MPX4-CA | 10 mA | 1N4006 | 300 мс | 95 Гц |
| **17** | ATmega8 | 4 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4007 | 400 мс | 15 Гц |
| **18** | ATmega16 | 2 Mhz | 19 pf | 7SEG-MPX4-CA | 12 mA | 1N4005 | 200 мс | 60 Гц |
| **19** | ATmega8535 | 1 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 25 mA | 1N4001 | 100 мс | 70 Гц |
| **20** | ATmega32 | 2 Mhz | 16 pf | 7SEG-MPX4-CA | 10 mA | 1N4002 | 500 мс | 10 Гц |
| **21** | ATmega16 | 4 Mhz | 19 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4003 | 600 мс | 10 Гц |
| **22** | ATmega8 | 1 Mhz | 19 pf | 7SEG-MPX4-CA | 12 mA | 1N4004 | 200 мс | 20 Гц |
| **23** | AT90S8535 | 4 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 12 mA | 1N4004 | 200 мс | 20 Гц |
| **24** | ATmega8 | 2 Mhz | 20 pf | 7SEG-MPX4-CA | 15 mA | 1N4007 | 300 мс | 15 Гц |
| **25** | ATmega32 | 3 Mhz | 17 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4006 | 350 мс | 30 Гц |

# КОМП'ЮТЕРНИЙ ПРАКТИКУМ №5
# РОБОТА ІЗ ЗОВНІШНЬОЮ ПАМ'ЯТТЮ. ОСНОВИ ФУНКЦІОНУВАННЯ SPI ІНТЕРФЕЙСУ

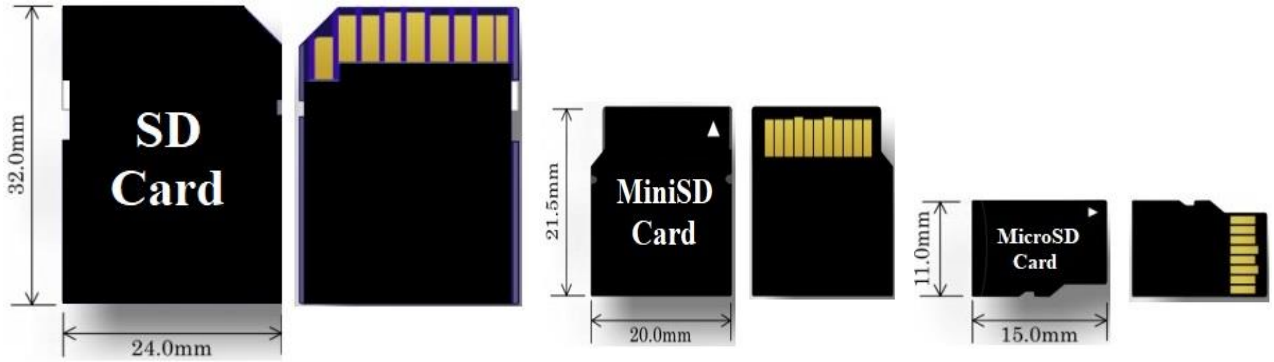
**Мета:** Ознайомитись з принципом роботи MMC/SD карт зовнішньої пам'яті та дослідити особливості організації SPI інтерфейсу.

**Змістовність роботи:** MMC/SD карти пам'яті, їх види та принципи роботи. SPI інтерфейс, основи його функціонування, призначення сигнальних ліній, визначення режимів роботи. Використання MMC/SD з мікроконтролерами AVR, бібліотека «sdcard.h». Написання коду програми для МК і використання вбудованих функцій середовища Proteus для моделювання роботи електричної схеми з зовнішньою картою пам'яті.

## 5.1. Теоретичні відомості

### 5.1.1. MMC/SD карти пам'яті. SPI інтерфейс

Мультимедійна карта захищеної цифрової пам'яті (MMC/SD) є стандартною картою пам'яті для мобільного обладнання, у якому можуть використовуватись як звичайні MMC/SD карти (рис. 5.1), так і їх версії зменшеного розміру (такі як miniSD та microSD тощо).




а) б) в)

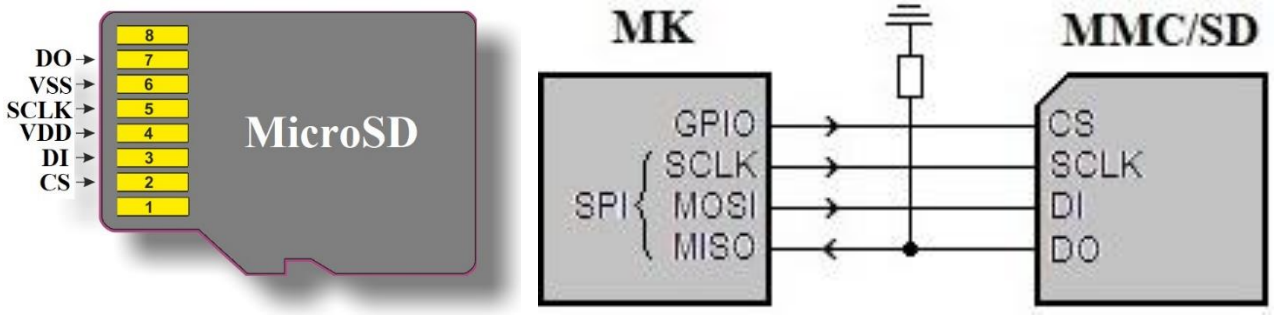
Рис. 5.1. Зовнішній вигляд та габаритні розміри карт пам'яті: MMC/SD (а); MiniSD (б); MicroSD (в)

Зазвичай MMC/SD має у своєму складі мікроконтролер, який забезпечує керування внутрішньою флеш-пам'яттю карти (передача розміру блоку даних,

аналіз рівня зносу карти, виправлення помилок тощо) та передачу інформації між картою пам'яті та зовнішнім хост-контролером. Передача інформації здійснюється у вигляді блоків даних розміром в 512 байт. Як наслідок, такий тип зовнішньої пам'яті відносять до категорії **блокових** пристроїв (принцип функціонування яких подібний до загального жорсткого диску на ПК). Для передачі інформації в MMC/SD пам'яті передбачено два режими роботи карти:

- за допомогою власного хост-інтерфейсу.
- через SPI інтерфейс.

Протокол зв'язку карти через SPI інтерфейс з зовнішнім МК є дещо простішим ніж за допомогою власного хост-інтерфейсу і, як наслідок, використовується для більшості проектів на мікроконтролерах. Позначення основних сигнальних ліній та виводів карти пам'яті (на прикладі MicroSD виконання) показано на рис. 5.2.а і описано в табл. 5.1. Схема підключення сигнальних ліній карти до зовнішнього мікроконтролера (за допомогою SPI інтерфейсу) показана на рис. 5.2.б.




а) б)

Рис. 5.2. Позначення і підключення сигнальних ліній карти пам'яті: позначення основних сигнальних ліній MicroSD (а); схема підключення сигнальних ліній MMC/SD до зовнішнього мікроконтролера (б)

Варто зазначити, що обов'язковим є «підтягування» лінії карти DO до навантажувального резистора, адже без виконання таких дій окремі моделі MMC/SD карт можуть не пройти процес ініціалізації. У SPI інтерфейсі передбачено чотири режими роботи (від 0 до 3), залежно від фази та полярності тактової частоти схеми. Для роботи з MMC/SD пам'яттю прийнято використовувати нульовий режим роботи, але в деяких випадках застосовують і третій режим SPI інтерфейсу.

*Таблиця 5.1.*

**Призначення виводів MicroSD карти**

| Номер | Позначення | Опис |
|---|---|---|
| 1 | – | Підключити до «підтягувального» резистору |
| 2 | CS | Вибір карти |
| 3 | DI | Лінія для запису даних в карту |
| 4 | VDD | Напруга живлення 2,7-3,3v |
| 5 | SCLK | Синхроімпульси SPI |
| 6 | VSS | Заземлення |
| 7 | DO | Лінія для зчитування даних із карти |
| 8 | – | Підключити до «підтягувального» резистору |

Для того, щоб краще зрозуміти алгоритм роботи з MMC/SD у режимі SPI варто розглянути короткий опис принципу функціонування цього режиму. SPI (Serial Parallel Interface) – це синхронна послідовна лінія передачі даних, яка працює в повнодуплексному режимі (тобто сигнали даних надсилаються по лінії одночасно в двох напрямках). Водночас, пристрої що підключені до SPI, обмінюються даними за принципом «Ведучий» (Master) та «Підлеглий» (Slave).

Ведучий пристрій створює «**кадр даних**» (характерний як для власних команд, так і для відповіді підлеглих пристроїв), генерує тактові імпульси та здійснює вибір підлеглого пристрою, з яким буде проводитись обмін інформацією (у прямому чи зворотному напрямку, або в двох напрямках одночасно). Для вибору конкретного підлеглого пристрою ведучий пристрій генерує відповідний логічний сигнал (розміром в один біт) або використовує дискретні вхідні / вихідні контакти загального призначення. Фактично, по SPI дані завжди передаються в обох напрямках, але ведучий та підлеглий пристрої здійснюють відбір значущих байтів отриманих з кадру даних за принципом: «викинути отриманий байт і лише передавати інформацію» чи «згенерувати байт-відповідь і лише приймати інформацію».

Межі тактового сигналу, з допомогою якого здійснюється синхронізація передачі даних, визначаються набором з двох параметрів: тактовою полярністю (CPOL) і тактовою частотою (CPHA). Кожен з цих параметрів може приймати два можливих стани і, як наслідок, визначає чотири можливих режими роботи

SPI (несумісні один з одним). Таким чином, пара пристроїв «ведучий / підлеглий» (рис. 5.3) повинна працювати в однаковому режимі та, як наслідок, використовувати однакові параметри тактового сигналу. Фактично, у SPI немає механізму підтвердження прийому даних і підлеглі пристрої можна розглядати як інструменти введення / виведення інформації з ведучого пристрою. Під час транзакції (передачі даних) по SPI надсилається пакет даних, що складається з: маркеру, блоку даних (корисної інформації) та додаткової CRC-команди. Варто зазначити, що процес передачі блоку даних розпочинається тільки після отримання маркеру (команди-відповіді) з підлеглого або ведучого пристрою (залежно від напрямку передач інформації).

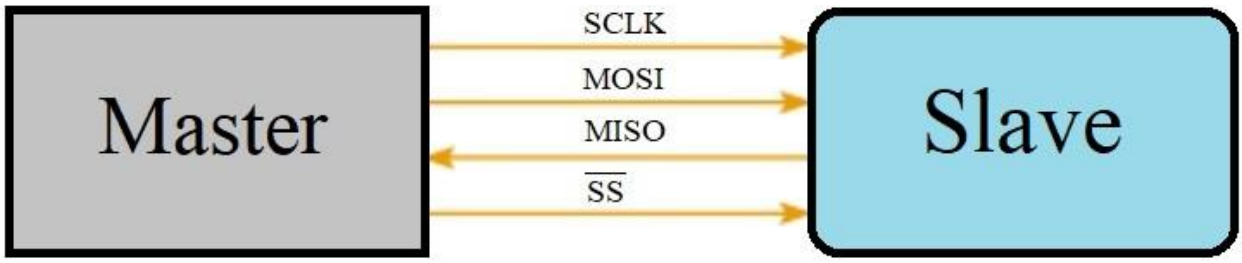



Рис. 5.3. Приклад схеми зв'язку в парі пристроїв через SPI-інтерфейс

Сигнал синхронізації, який генерується ведучим пристроєм, повинен передаватись по лінії SCLK. Лінія MOSI відповідає за передачу даних від ведучого пристрою до підлеглого, MISO – транслює інформацію від підлеглого пристрою до ведучого. А лінія SS забезпечує вибір конкретного підлеглого пристрою. У випадку, коли через SPI підключається декілька підлеглих пристроїв до одного ведучого, Master генерує окремий сигнал вибору для кожного з підлеглих пристроїв, підключених до лінії SS (див. рис. 5.4).

Алгоритм роботи MMC/SD у режимі SPI полягає в наступному: оскільки у випадку SD карти пакет інформації, що повинен передаватись між зовнішнім МК та зовнішньою пам'яттю, є «пакетом» завжди фіксованої довжини (розміром 512 байт), а дані надсилаються послідовно і побайтово (з фіксованим напрямком передачі інформації), то про готовність карти до роботи повинна свідчити поява сигналів на лініях передачі даних карти (DO та DI, відповідно). Після того, як на карту надіслано деякий кадр даних (Command frame), вона повинна обов'язково надіслати у зворотному напрямку до МК маркер (команду-відповідь).

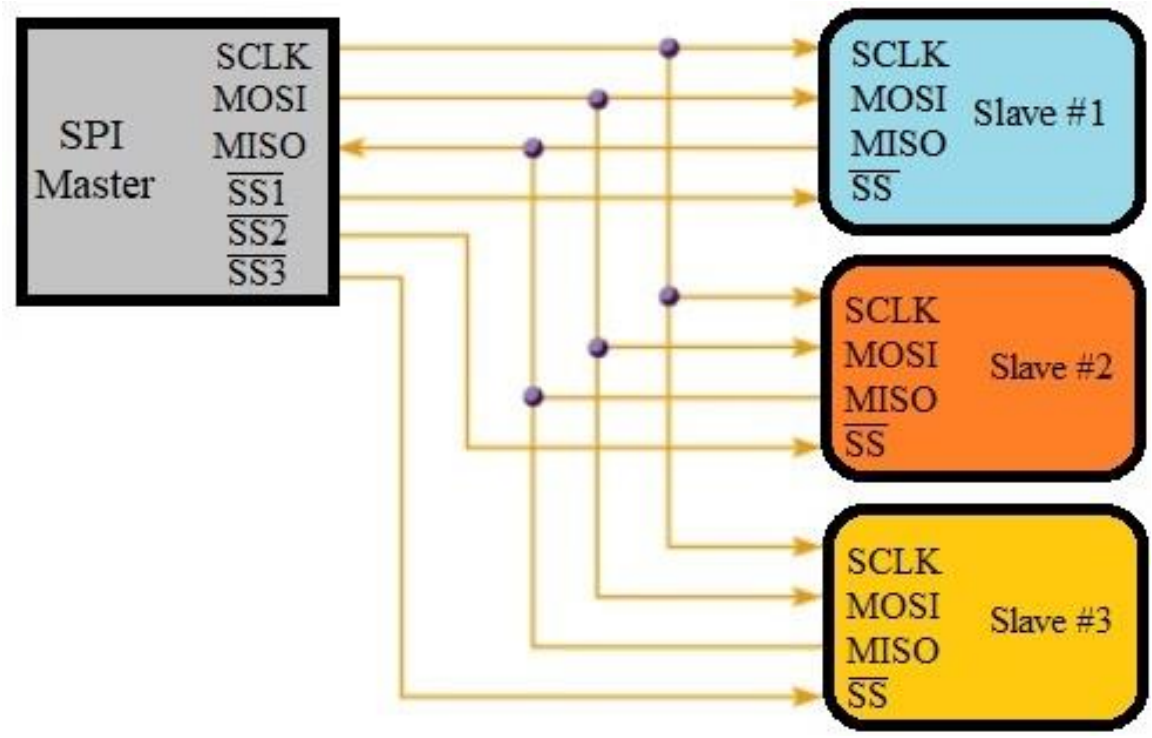



Рис. 5.4. Приклад схеми зв'язку декількох підлеглих пристроїв через SPI-інтерфейс

Оскільки передача даних між зовнішнім мікроконтролером і картою здійснюється за допомогою послідовного тактового сигналу, який генерує сам МК, то він повинен зчитувати дані, відправляти і отримувати байти доти, поки не буде виявлено необхідну зворотну команду-відповідь з карти.

У випадку, коли SD-карта готова отримати дані на вхід, внутрішній контролер карти надсилає на лінію DO сигнал логічної одиниці. У процесі передачі пакету інформації на карту (та читання інформації з неї) лінія DI також повинна постійно перебувати в стані логічної одиниці. Перед початком передачі інформації на лінію CS надсилається сигнал типу Falling Edge, що переводить її стан з логічної одиниці в нуль та утримує в стані логічного нуля впродовж всього процесу транзакції (команда-запит, команда-відповідь, процес передачі даних (якщо такі існують)). Після того як на карту надіслано кадр даних, нею надсилається зворотна відповідь на команду (рис. 5.5) з індексом R (номер якого залежить від того, наскільки успішним був процес передачі даних).

Тут передача команди з індексом R1 свідчить про успішний процес передачі даних, а R2, R3 або R7 – про певний тип помилки, що виник під час передачі інформації. Зворотня відповідь з карти до зовнішнього контролеру

надсилається назад протягом певного «часу команди-відповіді» (NCR), що складає від 0 до 8 байт для SD-карт, та від 1 до 8 байт для MMC-карт пам'яті.

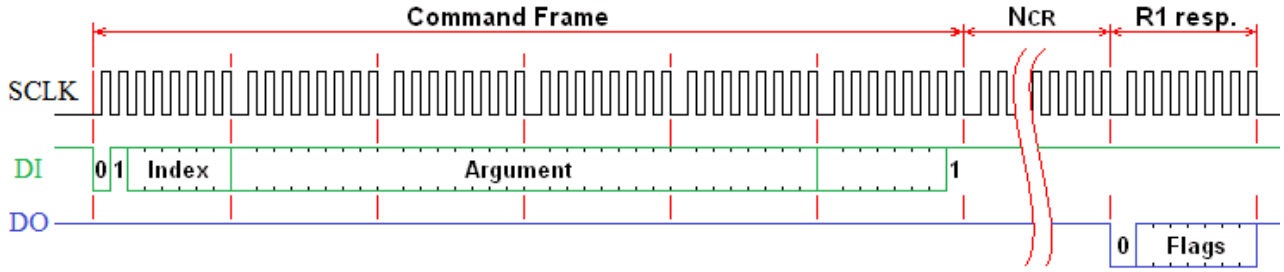



Рис. 5.5. Приклад візуалізації процесу обміну даних між MMC/SD картою і керуючим пристроєм

Для спрощення підключення МК до зовнішньої пам'яті у середовищі CodeVisionAVR передбачено спеціальну бібліотеку, що має назву «sdcard.h». Ця бібліотека підтримує роботу з декількома типами карт пам'яті, зокрема: MMC, SD, SD HC тощо. Розглянемо приклад підключення зовнішньої карти пам'яті до мікроконтролера AVR.

### 5.1.2. Підключення зовнішньої пам'яті до AVR

Як приклад зовнішнього накопичувача для нового проекту використаємо стандартний MMC / SD тип карти пам'яті. Робоча напруга живлення таких пристроїв варіюється в діапазоні від 2.7 В до 3.6 В. Однак, для оптимальної роботи карти необхідно використовувати напругу номіналом у 3.3 В та струмом не менше 100 мА (адже при виконанні операцій запису даних на карту, споживання нею струму може сягати до 100 мА і вище). Варто зауважити, що **при використанні робочої напруги номіналом 5 В і більше**, карта пам'яті практично одразу виходить з ладу.

Як і у попередніх практичних роботах для належної роботи схеми необхідно задати конкретні параметри тактової частоти МК, провести налаштування портів вводу/виводу інформаціїї, таймерів, переривань та інших модулів, що будуть використані в системі, а також підключити бібліотеку «sdcard.h» (у якій вже реалізовано необхідні налаштування SPI-інтерфейсу МК) для роботи з MMC/SD картами. Тобто, користувачу не потрібно додатково

проводити налаштування SPI-інтерфейсу, а лише фізично підключити карту пам'яті до ніжок SCLK, MOSI, MISO, SS на МК).

Подальшу роботу із зовнішньою пам'яттю у середовищі CodeVisionAVR можна поділити на два важливих етапи:

- Ініціалізація.
- Безпосередній запис і зчитування інформації з карти.

Розглянемо процес ініціалізації зовнішньої пам'яті. З метою моделювання роботи MMC/SD карти використаємо електричну схему, створену в системі Proteus для практичної роботи №4, і додамо до неї новий компонент «MMC» (позначення «M1» на рис. 5.6).

**Звернути увагу:** в індивідуальному завданні для студента модель МК, що використовується у схемі до практичної роботи №5 може не співпадати з моделлю МК, що використовувався для практичної роботи №4 (табл. 5.2).

Як видно з рис. 5.6 використана анімована модель карти пам'яті має чотири лінії керування (CS, DI, DO і CLK) та набір додаткових параметрів (рис. 5.7), до вікна налаштування яких можна перейти шляхом комбінації клавіш ПКМ –> Edit Properties. Для проведення процесу ініціалізації карти в полі «Card Image File» необхідно задати шлях до образу файлу з розширенням «*.mmc», який надано викладачем.

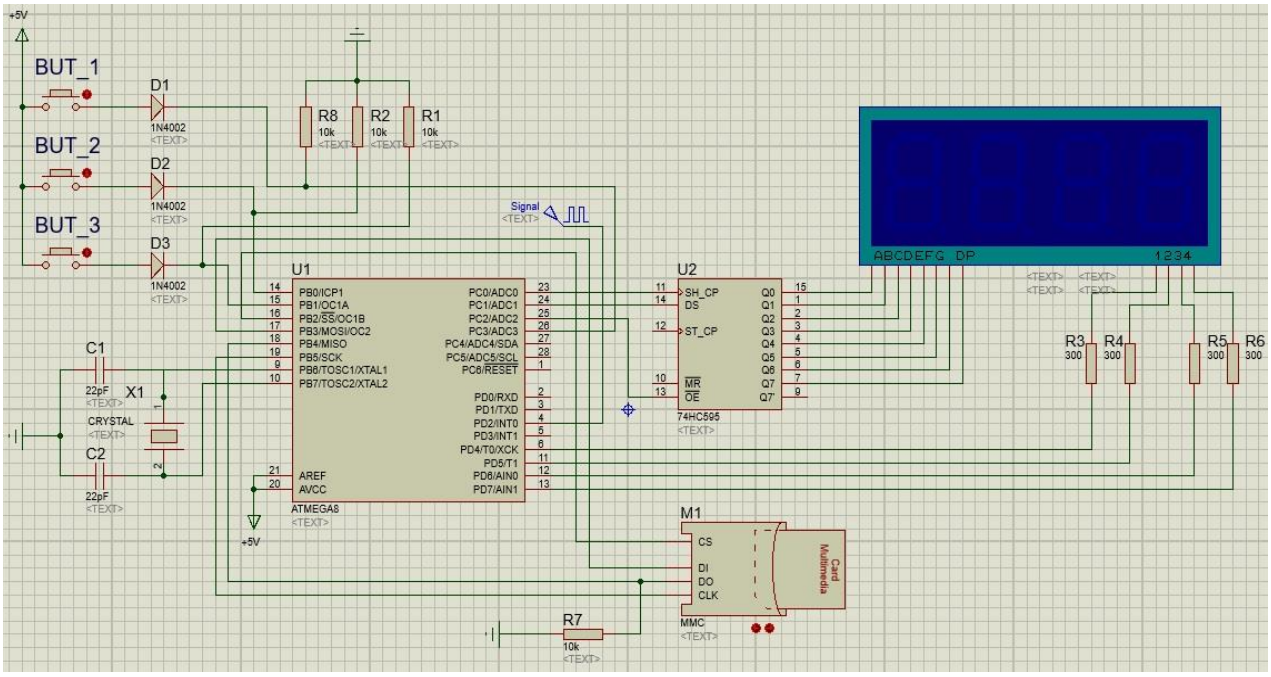



Рис. 5.6. Приклад електричної схеми для моделювання роботи зовнішньої пам'яті у системі Proteus

Якщо образ файлу присутній, то під час моделювання роботи схеми карта пройде ініціалізацію, якщо ж ні – то на МК буде надіслано один з кодів помилки (який, при бажанні, можна розшифрувати користуючись технічною документацією до програми CodeVisonAVR), наприклад: помилка ініціалізації, захист від запису, карту не знайдено тощо.

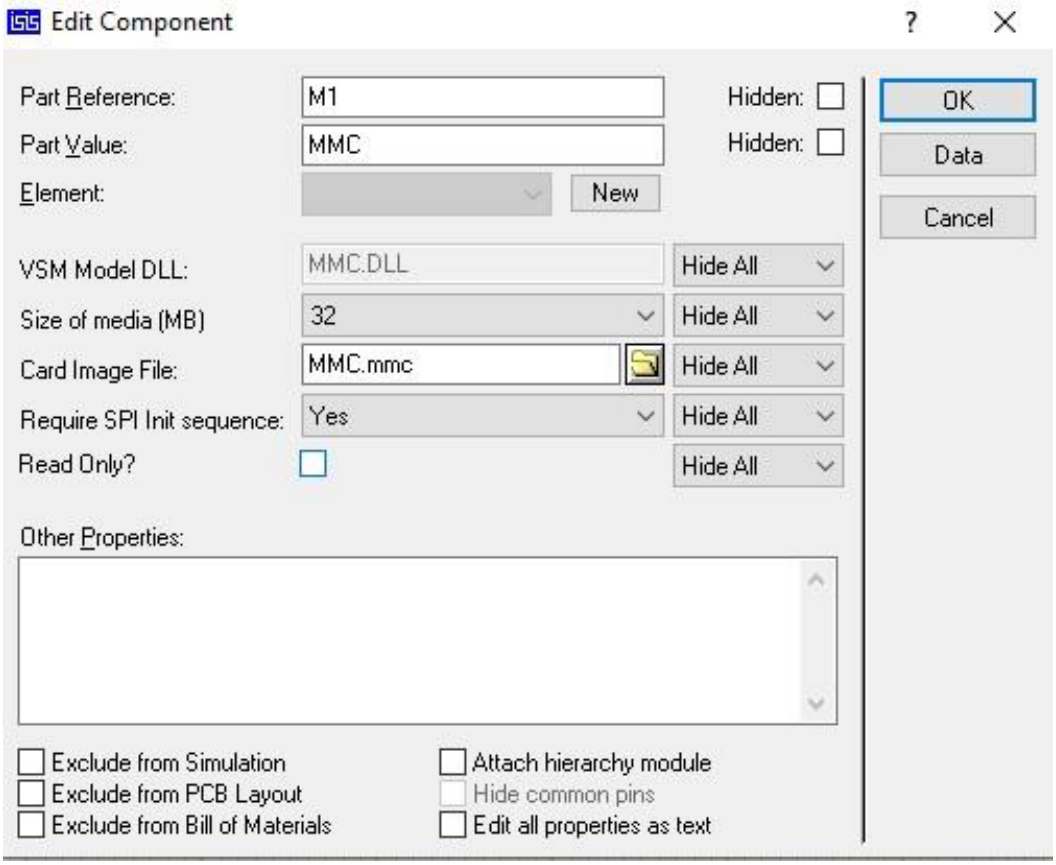



Рис. 5.7. Вікно налаштувань додаткових параметрів карти пам'яті у системі Proteus

Для роботи з картою пам'яті у CodeVisonAVR передбачено такі основні функції:

- «disk_timerproc()».
- «disk_initialize()».
- «disk_read()».
- «disk_write()».

Функція «disk_timerproc()» використовується для синхронізації роботи карти і повинна викликатись мікроконтролером кожні 10 мс, впродовж усього періоду роботи з зовнішньою пам'яттю. У свою чергу, «disk_initialize()» відповідає за ініціалізацію карти та як аргумент приймає її порядковий номер.

Якщо значення такої функції дорівнює нулю, то це означає, що MMC/SD успішно пройшла ініціалізацію та готова до роботи. Будь-які інші значення цієї функції означатимуть помилку ініціалізації. Функції «disk_read()» та «disk_write()» призначені для зчитування і запису даних в карту, відповідно. Такі функції приймають на вхід чотири аргументи: адресу пристрою, назву масиву для роботи з даними, з якого блоку почати зчитування/запис даних, яку кількість блоків даних потрібно прочитати/записати.

Наприклад, для функції «disk_read (0, Buffer, 0, 1)» перший аргумент (0) означає адресу пристрою, Buffer – назву масиву для запису даних, третій аргумент (0) – номер блоку, з якого буде починатись зчитування інформації, четвертий (1) – скільки блоків інформації потрібно прочитати.

Варто зазначити, що оскільки при роботі з MMC/SD будь-яка інформація записується і зчитується з карти блоками лише **по 512 байт**, то навіть якщо виникне необхідність прочитати лише 1 байт інформації з карти, за умовою МК все-одно повинен зчитувати цілий блок розміром 512 байт. Як наслідок, для коректної роботи коду програми для МК розмір масиву, до якого будуть записуватись дані (наприклад, Buffer у попередньому абзаці), повинен складати 512 значень. А для компіляції проекту та коду в CodeVisonAVR потрібно збільшити робочий розмір стеку в налаштуваннях проекту до 650 байт. Розглянемо приклад коду МК для ініціалізації MMC/SD карти пам'яті.

Приклад коду для ініціалізації MMC/SD карти пам'яті

```
#include <mega8.h>
#include <sdcard.h>
#include <alcd.h>
 // Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
disk_timerproc(); // Кожні 10мс викликаємо функцію синхронізації карти пам'яті
}
 void main(void)
{
// Timer/Counter 1 initialization
TCCR1A=0x00;
```

```
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x4E;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;
// Global enable interrupts
#asm("sei")
// Ініціалізація карти
if((disk_initialize(0))==0)
{
        // Ініціалізацію пройдено. Вивести текст «CARD» на семисегментні індикатор
        …
}
else
{
        // Ініціалізацію не пройдено. Вивести текст «FAIL» на семисегментні індикатор
        //…
}
while (1)
        {
        // Код основної програми
        …
        }
}
```

Як видно з коду, для синхронізації роботи карти шляхом запуску функції «disk_timerproc()» через кожні 10 мс було налаштовано переривання за збігом на першому таймері/лічильнику МК. У випадку, якщо карта пройшла ініціалізацію, то за умовою при симуляції роботи схеми на індикаторах буде виведено надпис «CARD», а якщо ні – «FAIL». Розглянемо приклад коду МК для ініціалізації та зчитування інформації з MMC/SD карти пам'яті.

Приклад коду для зчитування інформації з MMC/SD карти пам'яті

```
#include <mega8.h>
#include <delay.h>
```

```
#include <sdcard.h>
#include <alcd.h>
#include <stdio.h>
// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
disk_timerproc(); // Кожні 10мс викликаємо функцію синхронізації карти пам'яті
}
void main(void)
{
unsigned char Buffer[512]; // Ініціалізація масиву в який будуть записуватись дані, що зчитані з карти
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 7,813 kHz
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x4E;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;
// Global enable interrupts
#asm("sei")
// Ініціалізація карти
if((disk_initialize(0))==0)
{
        // Ініціалізацію пройдено. Вивести текст «CARD» на семисегментні індикатор
        …
}
else
{
        // Ініціалізацію не пройдено. Вивести текст «FAIL» на семисегментні індикатор
        …
}
delay_ms(1000); // Витримати паузу
                // Очистити індикатор
```

```
…
while (1)
    {
        disk_read (0, Buffer, 0, 1); // Зчитати дані з SD карти під номером «0»,
                                     // записати зчитані дані у масив «Buffer»,
                                     // дані зчитувати з першого блоку карти,
                                     // зчитати один блок інформації/

        //Вивести зчитану інформацію на семисегментні індикатор
        …
    }
}
```

Запис інформації в карту проводиться аналогічно шляхом використання функції «disk_write()».

## 5.2. Завдання для виконання

Створити робочу папку, яка буде мати наступну назву: **MP1_PR5_Шифр групи_Прізвище студента_№варіанту** (де PR5 розшифровується як: «Практична робота №5»). **Назва папки задається латиницею (англомовними символами)!**

**Завдання 5.1.** Розробити електричну схему для подальшого моделювання роботи частотоміру, використовуючи мікроконтролер AVR, регістр зсуву 74HC595, MMC карту пам'яті та інші вбудовані елементи системи Proteus, у відповідності до початкових умов, що наведені у табл. 5.2.

Розрахувати номінали резисторів (їх опір та потужність), які необхідно використати у розробленій схемі для обмеження робочого струму на світлодіодах семисегментного індикатору. Розрахунок проводиться відповідно до робочих параметрів використаних електронних компонентів (а саме: робочої напруги у схемі (5 В) та струму світлодіодів для сегментів індикатору).

Провести попередній розрахунок частоти вхідного сигналу відповідно до індивідуальних завдань, наведеним у табл. 5.2.

**Порядок виконання Завдання 5.1.**

1. Відкрити робоче вікно середовища Proteus та обрати з бази електронних компонентів необхідний перелік (у відповідності до номеру варіанту в таблиці 5.2).

2. Розмістити обрані компоненти на робочому полі, з'єднати їх провідниками та задати робочі параметри (звіривши з індивідуальним завданням) для подальшого моделювання схеми. Місце розташування компонентів на схемі обирається студентом самостійно. Приклад схеми наведено на рис. 5.6.

3. Зберегти проект розробленої схеми у раніше створеній робочій папці із назвою: «PR5_Шифр групи_№ варіанту». **Назва файлу задається лише латиницею (англомовними символами)!**

**Завдання 5.2.** Провести налаштування та створити програмний код для МК відповідно до індивідуального завдання та початкових умов, наведених у табл. 5.2.

**Порядок виконання Завдання 5.2.**

1. Відкрити робоче вікно середовища CodeVisionAVR, створити новий файл проекту та провести налаштування МК у відповідності до заданих параметрів. Для цього необхідно виконати наступні кроки:

- На верхній робочій панелі середовища CodeVisionAVR натиснути кнопку «Create a new file or project» → «New Project» → «Yes» → «AVR8(ATtiny, ATmega, AT90)» → «Ok».
- У вкладці «Chip», обрати зі списку модель МК відповідно до індивідуального завдання (рядок «Chip») та робочу частоту МК (рядок «Clock»).
- У вкладці «Ports» провести налаштування I/O-портів на вхід або вихід, залежно від індивідуального завдання.
- У вкладці «Timers» обрати режим роботи і задати параметри першого таймера/лічильника.
- У вкладці «External Interrupts» обрати номер зовнішнього переривання і тип сигналу «Rising Edge».

2. Після завершення налаштувань натиснути кнопку «Generate program, save and exit» та зберегти три файли проекту (усі з однаковою назвою) у раніше створеній робочій папці (у якій вже розміщено Proteus-файл). **Назва файлу задається лише латиницею (англомовними символами)!**

3. Визначити формулу для розрахунку частоти вхідного сигналу, базуючись на прикладі, що показаний у п.п.4.1.3 практичної роботи №4.

4. Після збереження проекту у основному вікні (яке відкриється програмою CodeVisionAVR автоматично) написати програмний код, який би забезпечував виконання наступних умов (див. рис. 5.6):

- Якщо жодну із кнопок («BUT_1» / «BUT_2» / «BUT_3») не активовано, то на розряди семисегментного індикатору повинен виводитись надпис: «-.-.-.-.».
- Якщо активовано кнопку «BUT_1», то мікроконтролер «U1» повинен виводити на семисегментний індикатор впродовж часу $t_{d1}$ номер варіанту студента (у форматі «НВ.ХХ», де ХХ – номер варіанту. Наприклад, для варіанту №1 напис матиме вигляд «НВ.01»), після чого – перевірити процес ініціалізації зовнішньої карти пам'яті (якщо карта пройшла ініціалізацію, то вивести на індикатор надпис «CARD», якщо при ініціалізації відбулась помилка – надпис «FAIL»).
- Якщо активовано кнопку «BUT_2», то мікроконтролер «U1» повинен здійснювати підрахунок і виводити на семисегментний індикатор значення частоти вхідного сигналу (параметр $N$ у табл. 5.2), яке водночас повинно записуватись у зовнішню карту пам'яті (позначення «M1» на рис. 5.6).
- Якщо активовано кнопку «BUT_3», то мікроконтролер «U1» повинен відобразити поточне значення частоти вхідного сигналу, яке записане у зовнішню карту пам'яті.

**Звернути увагу:** початково, до образу файлу для зовнішньої пам'яті, який надано викладачем, записано фразу «.HI.». Як наслідок, значення виміряної частоти сигналу, початково, у зовнішню пам'ять не записано і буде доступним

лише після здійснення запису нових даних у карту студентом.

5. Провести компіляцію створеного коду, натиснувши кнопку «Build all project files» на верхній панелі середовища CodeVisionAVR.

6. Імпортувати згенерований «*.cof» файл до МК «U1», «*.mmc» файл до карти пам'яті «M1», задати параметри вхідного сигналу (на рис. 5.6 позначено як «Signal») і робочу частоту МК (у відповідності до номеру варіанту в табл. 5.2). Для цього обрати у вкладці «CKSEL Fuses» параметр «Ext.RC» у діапазоні, на якому працює кварцовий генератор, що використаний у схемі. Після цього вказати конкретне значення робочої частоти МК у полі «Clock Frequency» замість фрази «Default».

7. Перевірити достовірність роботи створеного коду шляхом запуску схеми на моделювання.

8. Зберегти раніше створений Proteus-файл та результати моделювання.

9. Оформити звіт до практичної роботи.

## 5.3. Вимоги до оформлення звіту

1. Звіт повинен бути представлений в електронному вигляді. Вихідний файл звіту має зберігатись у форматі **«*.pdf»**.

2. Назва електронного файлу зі звітом задається у форматі: «МП1_ПР5_Шифр групи_Прізвище студента_№варіанту».

3. Структура звіту повинна містити наступні елементи:

- Титульний аркуш.
- Мету роботи.
- Номер варіанту та індивідуальне завдання.
- Електричну схему, розроблену студентом у середовищі Proteus.
- Формули та результати розрахунку номіналу резисторів **(параметри опору та потужності)**.
- Блок-схему алгоритму для робочого коду МК.
- Текст з робочим кодом (повністю, включно з налаштуваннями МК).
- Зображення з результатами моделювання схеми у середовищі Proteus.
- Висновки по роботі.

- Перелік контрольних запитань.

4. Якість зображення та розмір шрифту на схемах, рисунках і таблицях повинні чітко відображати представлену в них інформацію. Усі рисунки, таблиці, схеми і розділи у звіті повинні бути підписані (вимоги див. у додатку А).

5. Файл звіту додається у загальну робочу папку (див. п. 5.2), яка надсилається викладачу на перевірку. Для проходження перевірки, робоча папка повинна містити: Proteus – файл (у форматі «*.DSN») зі створеною електричною схемою, всі файли проекту створеного у середовищі CodeVisionAVR, звіт до практичної роботи.

**Якщо оформлення звіту і робочої папки не відповідають вимогам, практична робота до захисту не допускається!**

## 5.4. Контрольні запитання

1. Що таке MMC/SD карта? Які варіації SD ви знаєте?
2. Яке призначення функціональних виводів microSD карти?
3. Що розуміють під поняттям внутрішнього та хост-контролеру при роботі з MMC/SD?
4. Що таке SPI та повнодуплексний режим роботи?
5. Що таке «кадр даних» і для чого він потрібен?
6. Опишіть принцип роботи SPI.
7. Що розуміють під поняттям Master та Slave?
8. Яке призначення функціональних ліній SPI? Як розшифровуються абревіатури назв функціональних ліній?
9. Скільки існує режимів роботи SPI та чим вони визначаються?
10. Які основні функції для роботи із MMC/SD у бібліотекці «sdcard.h» ви знаєте?
11. В чому полягає особливість зчитування інформації із MMC/SD карти?

Таблиця 5.2. Початкові умови до завдання 5.2

| № варіанту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | $t_{d1}$ | $N$ |
|---|---|---|---|---|---|---|---|---|
| **1** | ATmega8 | 2 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 25 mA | 1N4001 | 400 мс | 10 Гц |
| **2** | ATmega16 | 3 Mhz | 18 pf | 7SEG-MPX4-CA | 10 mA | 1N4002 | 300 мс | 100 Гц |
| **3** | ATmega8 | 3 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 19 mA | 1N4003 | 800 мс | 300 Гц |
| **4** | ATmega32 | 1 Mhz | 22 pf | 7SEG-MPX4-CA | 10 mA | 1N4004 | 400 мс | 50 Гц |
| **5** | ATmega16 | 4 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 19 mA | 1N4005 | 550 мс | 20 Гц |
| **6** | ATmega8 | 2 Mhz | 19 pf | 7SEG-MPX4-CA | 14 mA | 1N4006 | 600 мс | 100 Гц |
| **7** | ATmega32 | 1 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 16 mA | 1N4007 | 850 мс | 15 Гц |
| **8** | ATmega8 | 2 Mhz | 22 pf | 7SEG-MPX4-CA | 22 mA | 1N4004 | 300 мс | 20 Гц |
| **9** | ATmega16 | 2 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 28 mA | 1N4005 | 350 мс | 40 Гц |
| **10** | ATmega16 | 1 Mhz | 15 pf | 7SEG-MPX4-CA | 20 mA | 1N4002 | 400 мс | 25 Гц |
| **11** | ATmega8 | 1 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 10 mA | 1N4003 | 500 мс | 15 Гц |
| **12** | ATmega32 | 4 Mhz | 15 pf | 7SEG-MPX4-CA | 30 mA | 1N4004 | 400 мс | 70 Гц |
| **13** | ATmega8 | 2 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4007 | 450 мс | 30 Гц |
| **14** | ATmega16 | 1 Mhz | 20 pf | 7SEG-MPX4-CA | 16 mA | 1N4006 | 1000 мс | 10 Гц |

Продовження таблиці 5.2

| № варіанту | Модель МК | Тактова частота (кварц) | Конденсатори C1 і C2 | Модель індикатору | Струм для сегментів індикатору | Діод | $t_{d1}$ | $N$ |
|---|---|---|---|---|---|---|---|---|
| **15** | ATmega16 | 2 Mhz | 22 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4005 | 1200 мс | 90 Гц |
| **16** | ATmega32 | 1 Mhz | 18 pf | 7SEG-MPX4-CA | 10 mA | 1N4006 | 400 мс | 95 Гц |
| **17** | ATmega8 | 4 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4007 | 400 мс | 15 Гц |
| **18** | ATmega16 | 2 Mhz | 19 pf | 7SEG-MPX4-CA | 12 mA | 1N4005 | 350 мс | 60 Гц |
| **19** | ATmega32 | 1 Mhz | 15 pf | 7SEG-MPX4-CA-BLUE | 25 mA | 1N4001 | 350 мс | 70 Гц |
| **20** | ATmega32 | 2 Mhz | 16 pf | 7SEG-MPX4-CA | 10 mA | 1N4002 | 500 мс | 10 Гц |
| **21** | ATmega16 | 4 Mhz | 19 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4003 | 600 мс | 10 Гц |
| **22** | ATmega8 | 1 Mhz | 19 pf | 7SEG-MPX4-CA | 12 mA | 1N4004 | 400 мс | 20 Гц |
| **23** | ATmega32 | 4 Mhz | 20 pf | 7SEG-MPX4-CA-BLUE | 12 mA | 1N4004 | 400 мс | 20 Гц |
| **24** | ATmega8 | 2 Mhz | 20 pf | 7SEG-MPX4-CA | 15 mA | 1N4007 | 350 мс | 15 Гц |
| **25** | ATmega32 | 3 Mhz | 17 pf | 7SEG-MPX4-CA-BLUE | 15 mA | 1N4006 | 350 мс | 30 Гц |

# ДОДАТКИ

# ДОДАТОК А. ВИМОГИ ДО ОФОРМЛЕННЯ ПРАКТИЧНИХ РОБІТ

У цьому додатку наведено вимоги та представлено приклади оформлення основних складових елементів протоколу до практичної роботи. Порядок нумерації рисунків, таблиць, формул представлено у додатку лише як приклад. Тобто, наскрізна нумерація у кожній конкретній роботі повинна відслідковуватись студентом самостійно.

## Оформлення основного тексту в протоколі

**Шрифт:** Times New Roman 14 пт, **інтервал:** 1.5, **абзац:** перший рядок 1.25 см. **Вирівнювання основного тексту:** по-ширині, **вирівнювання заголовків:** по-середині, **вирівнювання підзаголовків:** по-ширині (з абзацу). Назви заголовків, підзаголовків, розділів та підрозділів виконуються жирним шрифтом.

## Оформлення рисунків та підписів до них

**Розміщення рисунку:** по-середині, **розміщення підпису рисунку:** знизу, по-середині. Рисунок підписується у форматі: «Рис. № практичної роботи . номер рисунку. Назва рисунку», звичайним шрифтом. Наприклад:

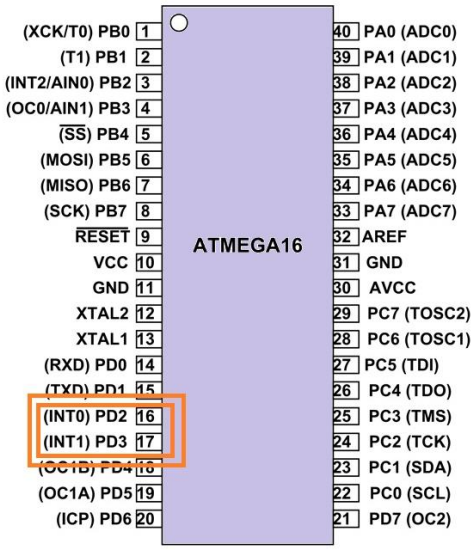



Рис. 1.1. Загальний вигляд мікросхеми

## Оформлення таблиць та підписів до них

**Розміщення таблиці:** зверху, вирівнювання по правому краю. Текст у таблиці виконується шрифтом Times New Roman 14 пт, або Times New Roman 12 пт, **інтервал:** одиничний.

Підписи таблиць виконуються у форматі: «Таблиця № практичної роботи. номер таблиці. Назва таблиці», звичайним шрифтом. Наприклад:

Таблиця 1.1. Конфігурації портів

| | | | | |
|---|---|---|---|---|
| 0 | 0 | Введення | Вимкнено | Вивід відключено від схеми |
| 0 | 1 | Введення | Ввімкнено | Вивід працює як джерело струму |
| 1 | 0 | Виведення | Вимкнено | На виході лог. «0» |
| 1 | 1 | Виведення | Вимкнено | На виході лог. «1» |

## Оформлення нумерації формул

**Розміщення формул:** по-середині листа. **Нумерація**: наскрізна, розміщується по правому краю листа, у форматі: «(Номер практичної роботи. номер формули)». Наприклад:

$$F_{MP} = \frac{F_M}{P} = \frac{8000000}{1024} = 7813 Гц \quad (1.1)$$

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Шерц Пауль, Монк Саймон «Электроника. Теория и практика» БХВ-Петербург, 2020 год, 1168 стр., ISBN: 978-5-9775-3847-3; 4-е издание, (78,5 мб. djvu).

2. Схемотехніка: Пристрої цифрової електроніки [Електронний ресурс] : в 2 т. : підручник для студентів, що навчаються за спеціальності «Електроніка» / В. М. Рябенький, В. Я. Жуйков, Ю. С. Ямненко, А. В. Заграничний ; НТУУ «КПІ». – Електронні текстові дані (2 файли: 5,06 Мбайт, 5,46 Мбайт). – Київ, 2016. – 757 с. – Назва з екрана.

3. Основи схемотехніки електронних систем: Підручник / В.І. Бойко, А.М.Гуржій, В.Я. Жуйков та ін. – К.: Вища шк., 2004. – 527с.: іл.

4. Белов А.В. Разработка устройств на микроконтроллерах AVR: шагаем от «чайника» до профи. — СПб.: Наука и Техника, 2013. — 528 с.

5. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательский дом «Додэка-XXI», 2004. – 288 с.: ил. (серия «Мировая электроника») ISBN 5-94120-075-7

6. Трамперт В. Измерение, управление и регулирование с помощью AVR микроконтроллеров.: Пер. с нем. – К.: «МК-Пресс», 2006. – 208 с., ил. ISBN 966-8806-14-X (рус.)

7. Лебедев М.Б. CodeVisionAVR: пособие для начинающих. – М.: Додэка-XXI, 2008. – 592 с.: ил. ISBN 978-5-94120-192-1

8. Мікропроцесорна техніка [Електронний ресурс] : навчальний посібник для студентів усіх форм навчання та студентів-іноземців напряму підготовки 6.050701 “Електротехніка та електротехнології” / НТУУ «КПІ» ; уклад. В. В. Кирик. – Київ : Політехніка, 2014. – 184 с.

9. Матвієнко М. П., Розен В. П., Закладний О. М. Архітектура комп'ютера. Навчальний посібник. — К: Видавництво Ліра-К, 2013. — 264 с.