Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра комп’ютерних наук

**Русу О.П.**

# ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ ТА РОБОТОТЕХНІКА В ЗАКЛАДАХ ОСВІТИ

Методичні вказівки до практичних робіт

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

Русу О.П. Програмування мікроконтролерів та робототехніка в закладах освіти. Методичні вказівки до практичних робіт [Електронне видання]. Кафедра комп’ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 69 с.

Дисципліна «Програмування мікроконтролерів та робототехніка в закладах освіти» присвячена вивченню принципів побудови та функціонування мікроконтролерів, а також здобуттю навичок їх практичного використання. Особливу увагу у курсі приділяється створенню програмного забезпечення на мовах асемблеру та принципам побудови пристроїв Інтернету речей.

## ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Дисципліна «Програмування мікроконтролерів та робототехніки в закладах освіти» присвячена вивченню мікроконтролерів як інструмента реалізації STEM-освіти та формуванню навичок їх використання у навчальному процесі. Особливу увагу приділено застосуванню мікроконтролерів для розвитку алгоритмічного, технічного та інженерного мислення учнів, програмуванню мікроконтролерів сімейства PIC на мові асемблеру, а також використанню робототехнічних систем у навчальній і проєктній діяльності закладів загальної середньої освіти.

**МЕТА ДИСЦИПЛІНИ.** Метою дисципліни є формування знань з принципів побудови мікроконтролерних застосунків, основ програмування мікроконтролерів сімейства PIC, а також здатності застосовувати робототехнічні засоби у навчальному процесі закладів освіти.

**ПРЕРЕКВІЗИТИ.** Передумовами для вивчення дисципліни є знання і вміння, отримані здобувачем при вивченні попередніх навчальних дисциплін бакалаврської підготовки: «Комп'ютерна схемотехніка та архітектура комп'ютерів», «Дискретна математика», «Алгоритмізація та програмування», «Алгоритми та структури даних».

**ПОСТРЕКВІЗИТИ.** Знання і вміння, отримані під час вивчення дисципліни, можуть бути використані у подальшій професійній діяльності вчителя інформатики або технологій, зокрема при організації навчальних занять, гурткової роботи та STEM-проєктів, а також при виконанні кваліфікаційної роботи.

## ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

**Тема 1. Знайомство з мікроконтролерами**

Призначення мікроконтролера. Застосування мікроконтролерів. Особливості роботи з мікроконтролерами. Мікроконтролери загального призначення провідних виробників електронних компонентів: Microchip Technology, STMicroelectronics, Texas Instruments, Renesas Electronics та інших. Мікроконтролер PIC16F84A.

Апаратні засоби для створення застосунків на мікроконтролерах: плати для розробки, макетні плати, плати для оцінки, опорні проєкти. Екосистеми для розробки застосунків на мікроконтролерах провідних виробників електронних компонентів. Платформа Arduino.

Засоби для програмування мікроконтролерів: зовнішні програматори та налагоджувачі, засоби та внутрішньосхемного програмування та налагодження.

Інтегровані середовища для розробки програмного забезпечення для мікроконтролерів: MPLAB, Microchip Studio, Atmel Studio, IAR Embedded Workbench, STM32CubeIDE, Code Composer Studio та інші. Елементи інтегрованих систем: редактори коду, компілятори, симулятори, програмне забезпечення для внутрішньосхемного налагодження.

Технічна документація мікроконтролерів: Reference Manual, Datasheet, Errata.

Типова електрична частина застосунків на мікроконтролерах. Виводи загального та спеціального призначення.

Порти введення-виведення загального призначення. Принципи формування електричних сигналів за допомогою портів введення-виведення. Принцип програмного керування портами введення-виведення та формування зовнішніх сигналів.

Основні елементи архітектури мікроконтролера: арифметико-логічний пристрій, акумулятори, пам'ять програм, оперативна пам'ять, стек.

Пам'ять програм. Призначення, розрядність. Технології виготовлення. Принципи адресації. Сторінки пам'яті програм.

Оперативна пам'ять. Регістри загального та спеціального призначення. Принципи адресації оперативної пам'яті. Сторінки оперативної пам'яті.

Стек. Призначення та принцип роботи стеку. Обмеження стеку та наслідки його переповнення. Передавання параметрів в підпрограми на мові асемблеру.

Машинний цикл і тривалість виконання команд. Принцип роботи командного конвеєра та наслідки його порушення. Команда nop та її використання для формування часових інтервалів. Принцип формування часових інтервалів у мікроконтролерах. Розрахунок тривалості виконання програмного коду. Формування затримок за допомогою вкладених циклів.

Тактовий генератор та його вплив на швидкодію і енергоспоживання мікроконтролера. Варіанти тактових генераторів мікроконтролерів та їх характеристики. Кварцові та керамічні резонатори, RC-генератори.

Слово конфігурації мікроконтролера та його призначення.

Типова структура програмного забезпечення мікроконтролера.

Допоміжні таймери та їх роль у підвищенні надійності програмного забезпечення: таймери затримки запуску мікроконтролера, сторожові таймери.

Таймери загального призначення: джерела тактового сигналу, передподільники тактової частоти, основні регістри.

Механізм переривань: принцип роботи, джерела переривань, прапори та біти дозволу переривань. Обробка переривань: вектори обробників переривань, збереження та відновлення контексту, асемблерні команди та типові алгоритми обробки переривань.

**Тема 2. Програмування на мові асемблеру**

Набори асемблерних команд мікроконтролерів різних виробників (PIC12, PIC16, PIC18, AVR, STM8, STM32, MSP430 та інших).

Системи числення, що використовуються при написанні програмного коду на мові асемблеру: двійкова, десяткова, шістнадцяткова. Формати інтерпретації чисел.

Змінні. Правила формування імен змінних. Розміщення змінних в оперативній пам’яті. Варіанти призначення імен змінним.

Константи. Правила формування імен констант. Розміщення констант к пам’яті програм. Варіанти призначення імен змінним.

Мітки. Мітки як різновид констант. Правила призначення та розташування міток.

Особливості роботи з багатобайтними змінними та константами. Директиви Low, High, Upper.

Принципи копіювання інформації: копіювання констант із пам’яті програм в оперативні пам’ять, копіювання інформації між комірками оперативної пам’яті. Особливості використання акумулятора.

Організація арифметико-логічних операцій. Принципи виконання арифметико логічних операцій: константа-змінна, змінна-змінна. Алгоритми виконання арифметико-логічних операцій із багатобайтними числами.

Прапори арифметико-логічних операцій: прапор нуля, прапор переносу-запозичення, прапор переносу-запозичення між тетрадами байту.

Арифметичні операції на мові асемблеру: додавання, віднімання, множення, ділення.

Логічні операції: унарні (інверсія, обмін тетрадами, побітове зміщення) та бінарні (побітове І, АБО, виключне АБО).

Умовні операції. Умовна операція з двома гілками. Умовна операція із однією гілкою. Умовні операції з кількістю гілок більше двох.

Організація умовних операції за допомогою асемблерних команд btfss та btfsc.

Організація умовних операції за допомогою асемблерних команд incfsz, decfsz.

Організація умовних операції за допомогою непрямої адресації та обчислювальних безумовних переходів. Покажчики на мові асемблеру. Програмний лічильник. Особливості та ризики використання непрямої адресації та обчислювальних безумовних переходів.

Організація циклів.

Алгоритми роботи з векторами та матрицями.

Алгоритми роботи з символьними даними.

Алгоритми обчислення математичних функцій (квадратний корінь, експонента, тригонометричні функції тощо).

**Тема 3. Керування апаратними вузлами електронної техніки**

Керування потужним навантаженням. Схемотехніка підсилювачів сигналів мікроконтролера. Підсилювачі на біполярних та польових транзисторах. Спеціалізовані електронні компоненти із логічними рівнями керування (транзистори, тиристори, симистори тощо). Особливості формування сигналів для різних схем підсилювачів.

Контроль стану механічних перемикачів. Варіанти електричного підключення механічних перемикачів. Брязкіт контактів. Гістерезіс. Тригери Шмідта. Алгоритми контролю стану механічних перемикачів.

Динамічна індикація. Принципи керування багаторозрядними індикаторами. Матричні індикатори.

Рідкокристалічні індикатори. Принципи керування індикаторами на рідких кристалах. Алгоритм формування сигналів для керування рідкокристалічними індикаторами.

Колекторні двигуни. Схема підключення до мікроконтролера. Драйвери колекторних двигунів. Алгоритм On-Off. Алгоритм керування зі зміною напрямку обертання. Алгоритм керування зі зміною швидкості. Широтно-імпульсна та частотно-імпульсна модуляція.

Крокові двигуни. Різновиди крокових двигунів. Двигуни із однією обмоткою (двигун Лав'є) та з двома обмотками. Алгоритми формування сигналів для керування кроковими двигунами.

Сервоприводи. Різновиди сервоприводів. Алгоритми формування сигналів для керування сервоприводами.

Периферійні модулі для забезпечення людино-машинного інтерфейсу (контролери клавіатур, матричних та рідкокристалічних індикаторів, звукових сигналізаторів тощо).

Периферійні модулі для комунікації по стандартним інтерфейсам (UART, IIC, SPI, CAN, USB та інших).

Принципи побудови типових застосунків на мікроконтролерах (ялинкові гірлянди, побутова техніка, інформаційні панелі, роботи, безекіпажні транспортні засоби, та інші).

**Тема 4. Створення пристроїв Інтернету речей**

Особливості пристроїв Інтернету речей: функціональність, інформаційна безпека, енергоощадність.

Апаратна основа пристроїв Інтернету-речей: спеціалізовані мікроконтролери, датчики, компоненти людино-машинного інтерфейсу.

Особливості живлення від автономних джерел енергії, монітори живлення. Методи зменшення енергоспоживання: зниження тактової частоти, енергоощадні режими, монітори живлення та енергоспоживання, модулі регульованих стабілізаторів напруги.

Особливості забезпечення необхідного рівня захисту інформації та кібербезпеки пристроїв Інтернету-речей. Спеціалізовані периферійні модулі: модулі шифрування, апаратні генератори випадкових чисел, модулі для обчислення контрольних сум та інші.

Засоби зменшення енергоспоживання: програмована система тактування, режими енергозбереження. Периферійні модулі для забезпечення бездротових інтерфейсів. Системи-на-кристалі. Екосистеми для створення пристроїв Інтернету речей.

## ПРАКТИЧНІ ЗАНЯТТЯ

**Тема 1. Мікроконтролери як основа робототехніки**

1. Ознайомлення з апаратною платформою мікроконтролерів та їх використанням у STEM-освіті.
2. Вивчення структури та основних характеристик мікроконтролерів сімейства PIC.
3. Налаштування середовища розробки для програмування мікроконтролерів (MPLAB або аналогів).

4. Розробка простих програм для керування світлодіодами (базові операції введення-виведення).
5. Аналіз прикладів застосування мікроконтролерів у навчальних STEM-проєктах.
6. Розробка фрагмента навчального заняття з використанням мікроконтролерів (визначення мети, результатів навчання).

**Тема 2. Програмування на мові асемблеру**

1. Ознайомлення з архітектурою мікроконтролерів PIC та системою команд.
2. Написання та налагодження програм керування портами введення-виведення.
3. Реалізація алгоритмів з використанням розгалужень і циклів на мові асемблеру.
4. Програмування таймерів та дослідження їх роботи на практиці.
5. Використання переривань у програмах для мікроконтролерів.
6. Налагодження програм у середовищі MPLAB: аналіз помилок та оптимізація коду.
7. Розробка методичних матеріалів для пояснення основ програмування учням.

**Тема 3. Керування апаратними вузлами електронної техніки**

1. Дослідження роботи електронних компонентів (транзисторів, симисторів) у схемах керування.
2. Підключення та програмування індикаторів (LED, LCD) і звукових пристроїв.
3. Організація людино-машинного інтерфейсу: робота з кнопками, клавіатурами, дисплеями.
4. Налаштування та використання інтерфейсів зв'язку (UART, I2C, SPI).
5. Розробка простих пристроїв на мікроконтролерах (світлові ефекти, сигналізація, інформаційні панелі).
6. Моделювання та реалізація навчального мікроконтролерного проєкту.

**Тема 4. Використання робототехніки в закладах освіти**

1. Ознайомлення з освітніми робототехнічними платформами (Arduino, Micro:bit, LEGO Education, VEX).
2. Розробка простих програм для керування робототехнічними пристроями (Arduino IDE, Scratch, MakeCode).
3. Створення базових робототехнічних проєктів (рухомі механізми, сенсори, керування).
4. Інтеграція робототехніки у навчальні предмети (інформатика, фізика, математика).
5. Організація STEM-проєктів і гурткової роботи з робототехніки.
6. Підготовка навчального проєкту для участі у конкурсах або олімпіадах з робототехніки.
7. Розробка сценарію уроку або позакласного заходу з використанням робототехнічних платформ.

# ПРАКТИЧНІ ЗАВДАННЯ

## 1. «ВІТАЮ, СВІТЕ!»

### 1. Мета роботи

1. Встановити програмне забезпечення MPLAB.
2. Створити перший проект на мікроконтролері PIC16F84.
3. Перевірити його працездатність.

### 2. Вказівки до виконання роботи

### 2.1 Встановлення середовища розробки MPLAB

Створити програмне забезпечення для мікроконтролерів PIC можна в інтегрованих середовищах для розробки (Integrated Development Environment, IDE) різних виробників.

Найбільш поширеним IDE для розробки програмного забезпечення для мікроконтролерів PIC є безоплатне IDE MPLAB X, яке має широкий набір інструментів та підтримує більшу кількість мікросхем, виробництва компанії Microchip Technology, у тому числі і мікроконтролери PIC та AVR. Завантажити останню версію MPLAB X можна з офіційного сайту Microchip Technology за посиланням https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide. Крім того, компанія Microchip Technology підтримує технологію хмарної розробки за допомогою хмарного IDE MPLAB Xpress, доступного за посиланням https://www.microchip.com/en-us/tools-resources/develop/mplab-xpress.

Однак для першого знайомства із особливостями мікроконтролерів PIC краще спершу освоїти більш просте програмне забезпечення MPLAB, яке є попередньою версією MPLAB X та MPLAB Xpress. Наразі компанія Microchip Technology припинила розвиток цього програмного продукту, проте усі його версії доступні для завантаження за посиланням:

https://www.microchip.com/en-us/tools-resources/archives/mplab-ecosystem

Завантажуємо останню версію IDE MPLAB (Версія 8.92) (рис. 1).

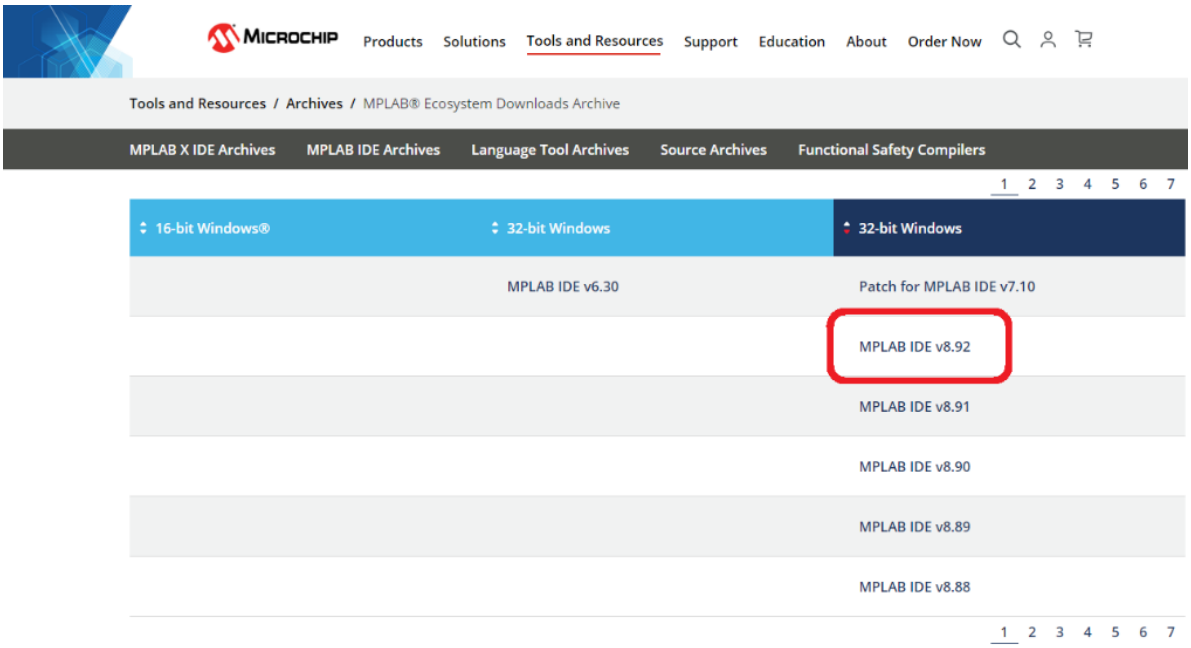


Рисунок 1 – Встановлення IDE MPLAB (крок 1)

Розпаковуємо архів та запускаємо програму встановлення (рис. 2)

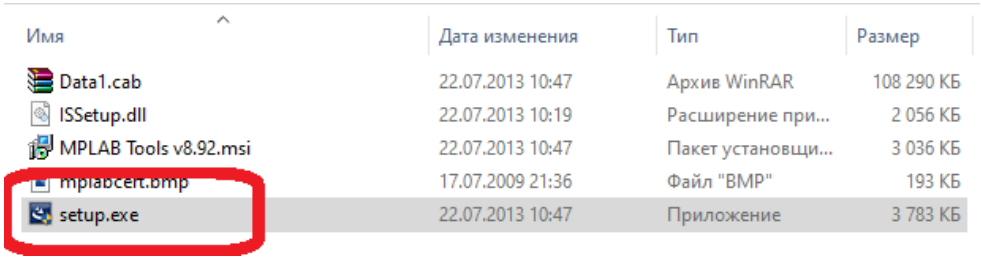


Рисунок 2 – Встановлення IDE MPLAB (крок 2)

У першому вікні тиснемо Next (рис 3).

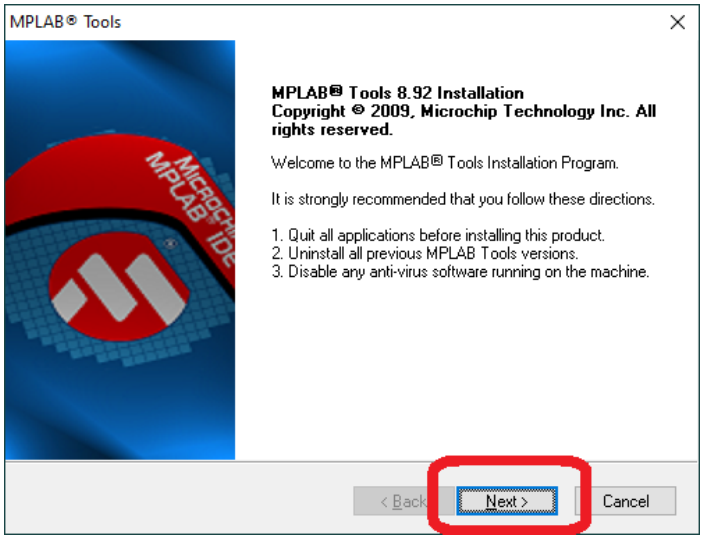


Рисунок 3 – Встановлення IDE MPLAB (крок 3)

Читаємо та приймаємо ліцензійні умови (рис. 4).

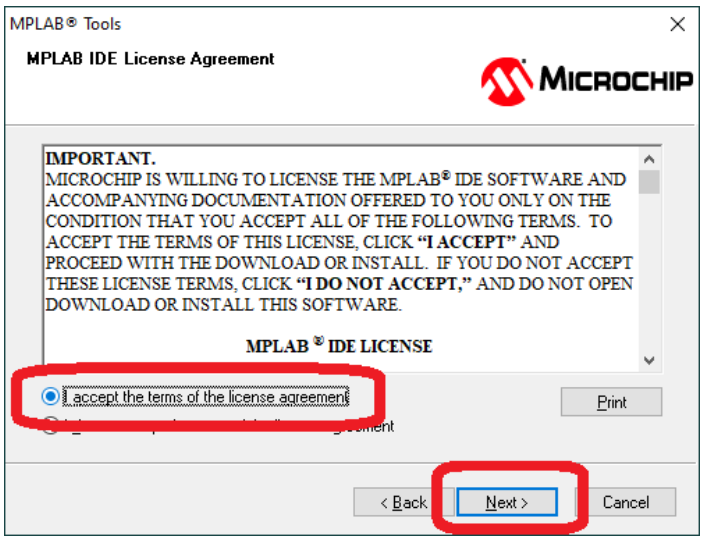


Рисунок 4 – Встановлення IDE MPLAB (крок 4)

На перших порах обираємо типовий варіант конфігурації (рис. 5).

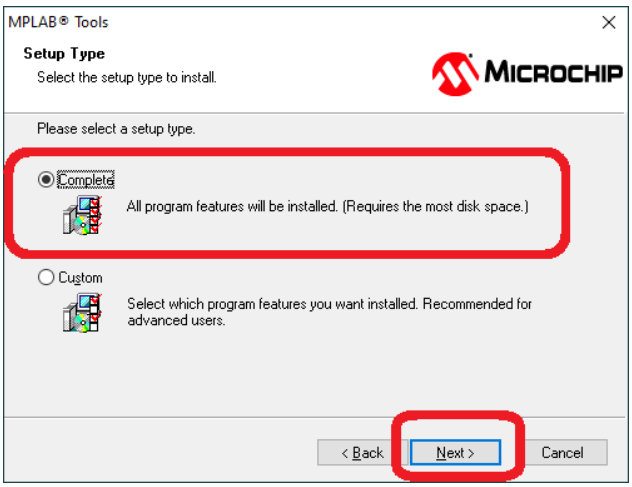


Рисунок 5 – Встановлення IDE MPLAB (крок 5)

Якщо потрібно, вказуємо каталог, куди встановити програму (рис. 6).

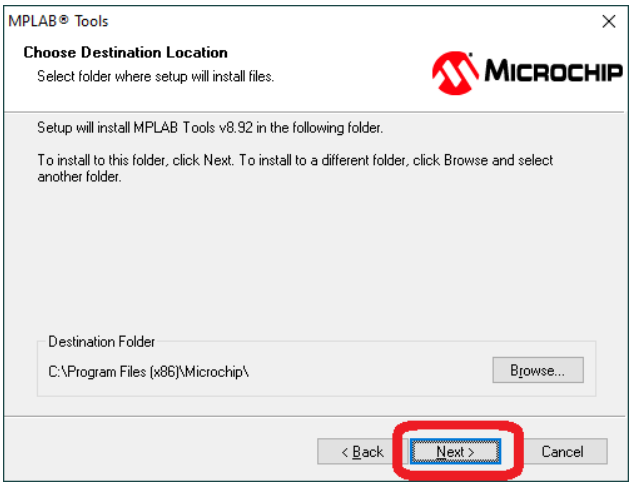


Рисунок 6 – Встановлення IDE MPLAB (крок 6)

Ще раз приймаємо ліцензійні умови (рис. 7).

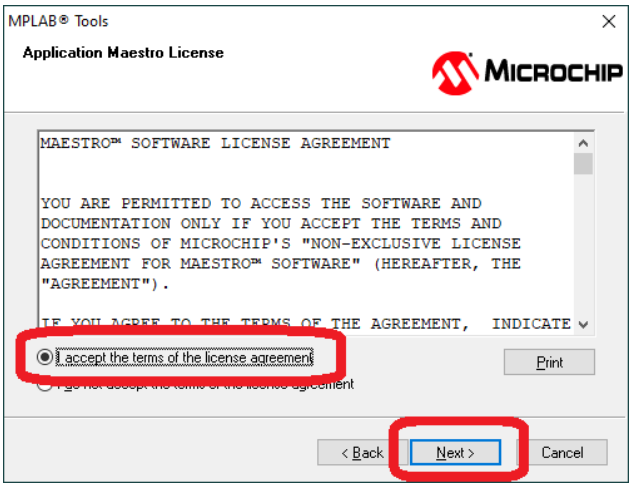


Рисунок 7 – Встановлення IDE MPLAB (крок 7)

І ще раз приймаємо ліцензійні умови (рис. 8).

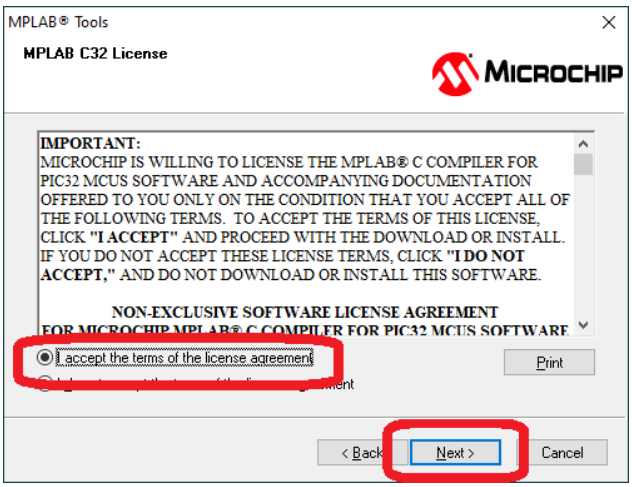


Рисунок 8 – Встановлення IDE MPLAB (крок 8)

Майже все готово для початку встановлення програми (рис. 9).

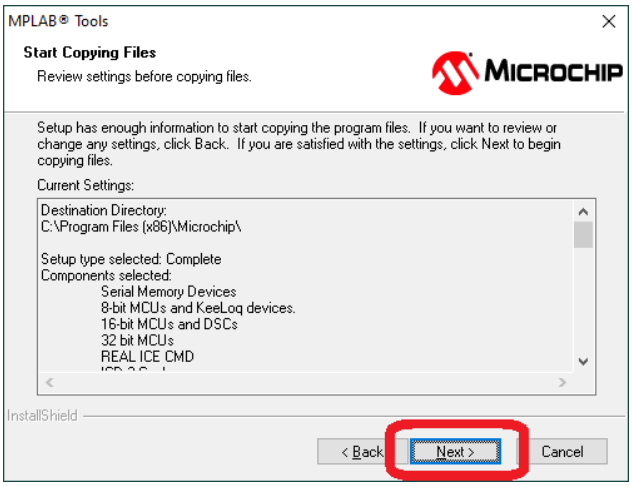


Рисунок 9 – Встановлення IDE MPLAB (крок 9)

Трішки чекаємо

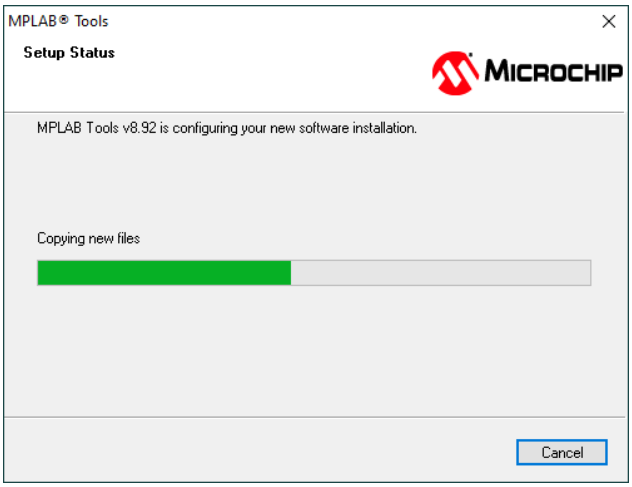


Рисунок 10 – Встановлення IDE MPLAB (крок 10)

Інсталяція завершена (рис. 11).

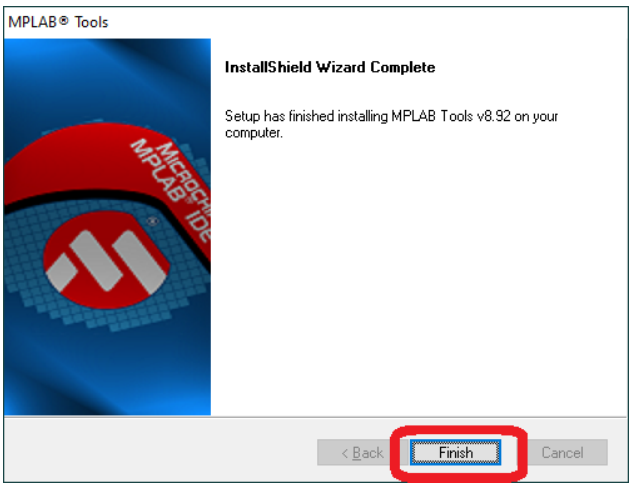


Рисунок 11 – Встановлення IDE MPLAB (крок 11)

За необхідності можна переглянути деяку технічну документацію. Якщо це не потрібно просто закриваємо вікно (рис. 12).

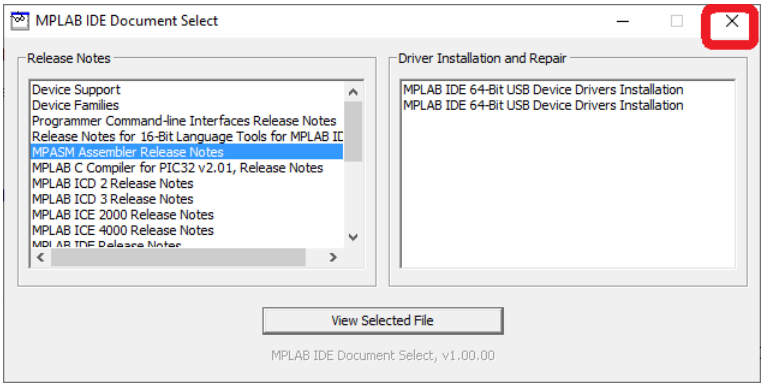


Рисунок 12 – Встановлення IDE MPLAB (крок 12)

### 2.2 Створення нового проекту

Запускаємо MPLAB IDE, відкривається 2 порожніх вікна Untitled Workspace та Output (рис. 13).

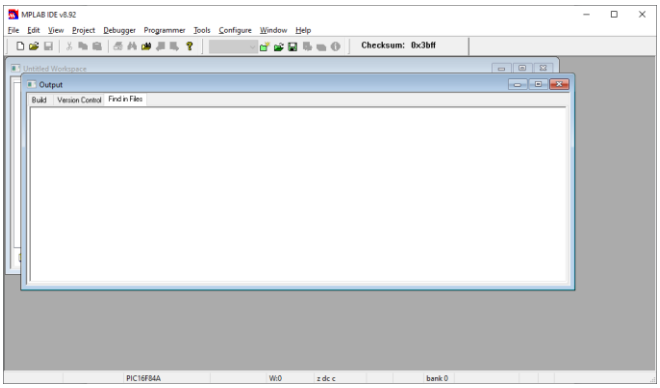

Рисунок 13 – Створення нового проєкту (крок 1)

Запускаємо помічника для створення нового проекту Project → Project Wizard (рис. 14).

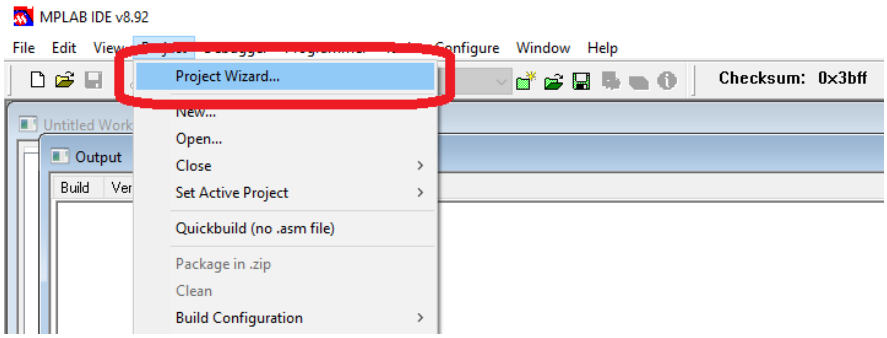


Рисунок 14 – Створення нового проєкту (крок 2)

У першому вікні тиснемо «Далі» (рис. 15).

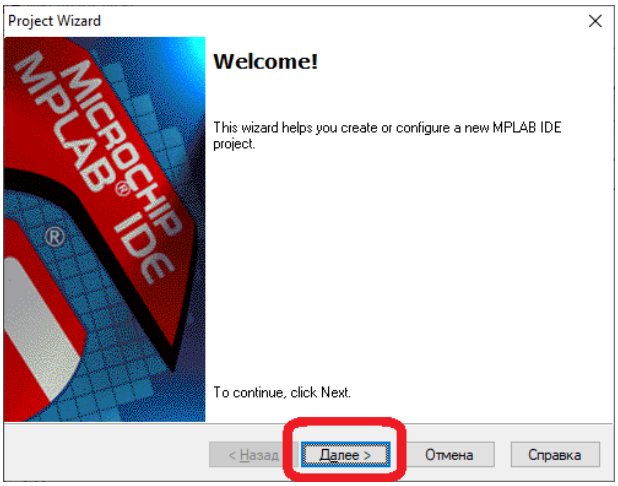


Рисунок 15 – Створення нового проєкту (крок 3)

Обираємо мікроконтролер. Перші проекти будемо робити на мікроконтролері PIC16F84A, що є оновленою новою версією мікроконтролера PIC16F84. Тому на даному етапі обираємо PIC16F84 або PIC16F84A (рис. 16).

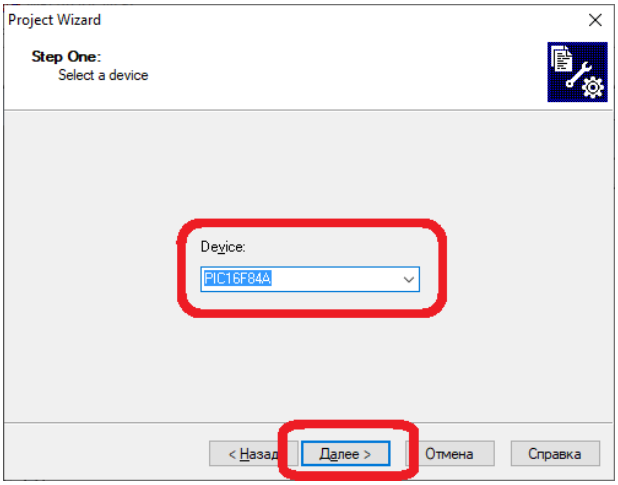


Рисунок 16 – Створення нового проєкту (крок 4)

Обираємо інструменти для створення файлу прошивки (тут можна залишити параметри, що рекомендуються) (рис. 17).

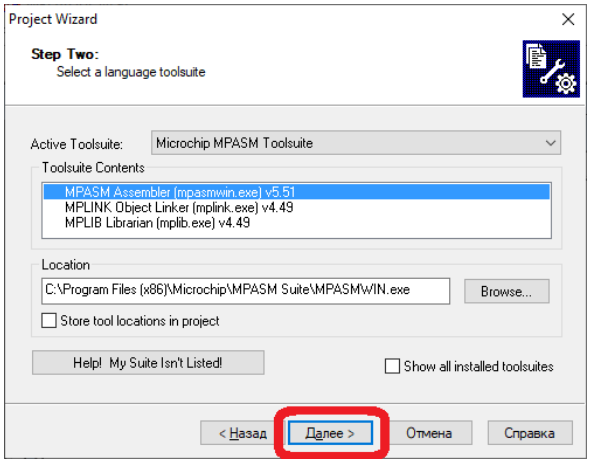


Рисунок 17 – Створення нового проєкту (крок 5)

Створюємо папку для файлів проекту, вигадуємо назву проекту, та зберігаємо усе те, що ви вигадали (рис. 18, 19).

**УВАГА!** MPLAB IDE не працює із файлами, що мають у свої назви літери кирилиці. Тому не розташовуйте папку проекту на робочому столі, або у папці «Мої документи». Краще створіть папку для проекту, наприклад, на диску D:.

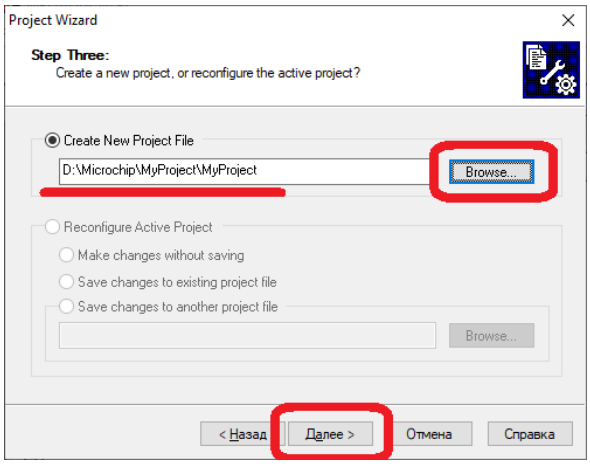


Рисунок 18 – Створення нового проєкту (крок 6)

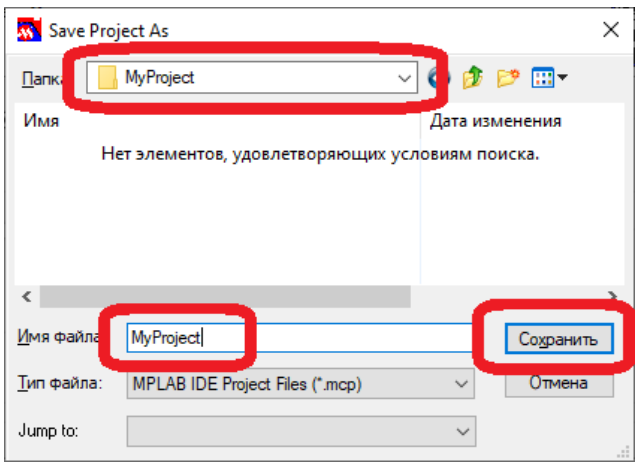


Рисунок 19 – Створення нового проєкту (крок 7)

На наступному етапі до проекту можна додати вже існуючі файли. Для першого проекту цей етап можна пропустити (рис. 20).

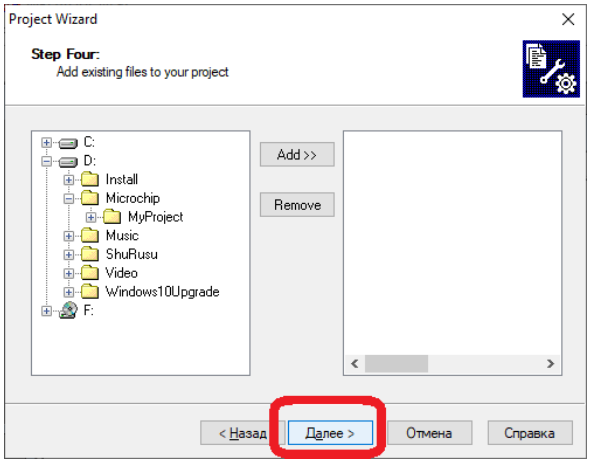


Рисунок 20 – Створення нового проєкту (крок 8)

Після цього робота по створенню нового проекту завершена (рис. 21).

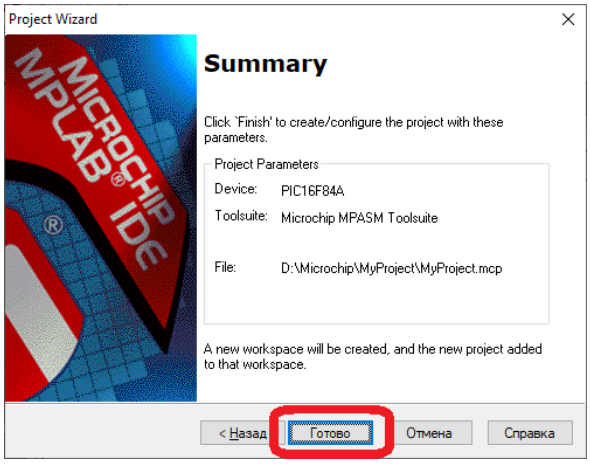


Рисунок 21 – Створення нового проєкту (крок 9)

Після цього у середовищі розробки буде відкрито два вікна: вікно з менеджером файлів проекту (розташовується у файлі із назвою MyProject.mcw) та вікно із вихідною інформацією (Output) (рис. 22).

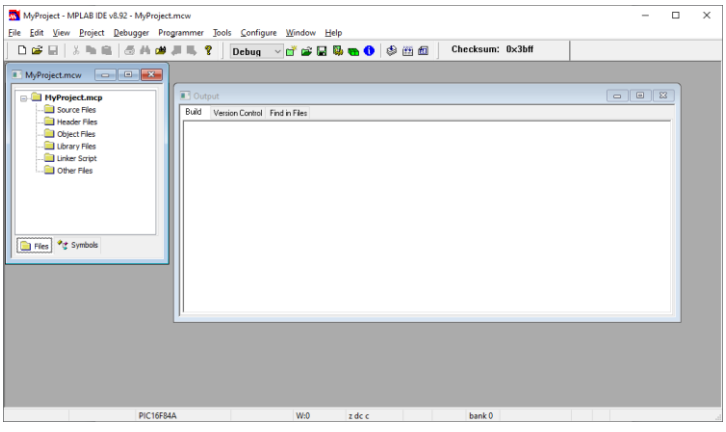

Рисунок 22 – Новий проєкт в IDE MPLAB

Якщо ці вікна не відображаються, то їх можна відкрити у любий час через меню View→Project та View→Output, відповідно (рис. 23).

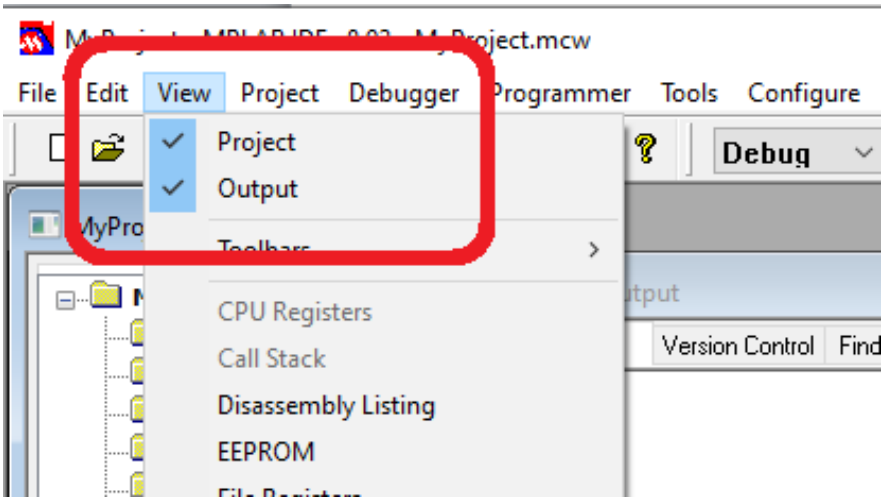


Рисунок 23 – Відображення вікна менеджера проєкту та вікна з вихідною інформацією

Додаємо до нашого проекту файл з вихідним кодом. Для цього обираємо пункт меню File→Add New File to Project, у діалоговому вікні, що відкриється, вигадуємо йому назву, наприклад MyProject.asm. Зверніть увагу, що розширення файлу потрібно вказувати явно – у даному випадку воно повинно бути «.asm», що означає, що ваш файл написаний на мові асемблеру (рис. 24).

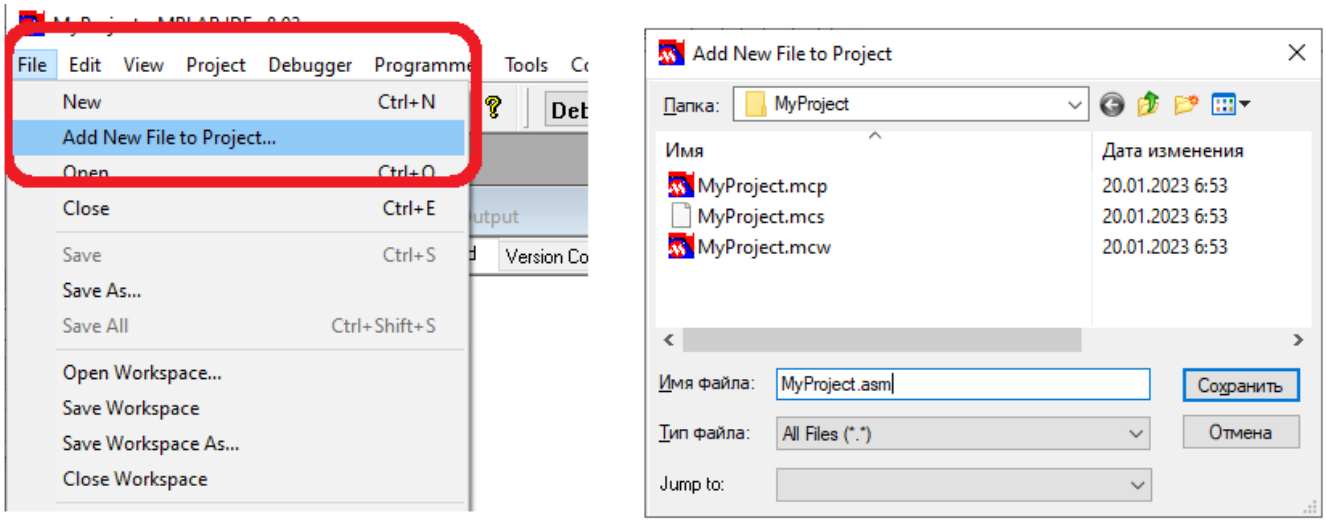


Рисунок 24 – Додавання файлу з вихідним кодом

Наступним кроком є додавання до проекту файлу, де описані назви регістрів. Його можна створити самому або взяти готовим. Цей файл має назву «p16f84.inc» і входить у стандартний комплект середовища MPLAB IDE. Його можна скопіювати із папки, де розташоване середовище MPLAB IDE

C:\Program Files (x86)\Microchip\MPASM Suite

або (для 32-розрядних версій Windows)

C:\Program Files\Microchip\MPASM Suite

Краще всього цей файл скопіювати у папку проекту. Після цього у вікні менеджеру проекту тиснемо правою кнопкою на папці із написом «Header Files» і у меню, що з'явиться, тиснемо на пункт «Add Files…». У діалоговому вікні, що з'явиться, обираємо файл «p16f84.inc». Після цього вікно менеджеру проекту буде виглядати наступним чином (рис. 25).

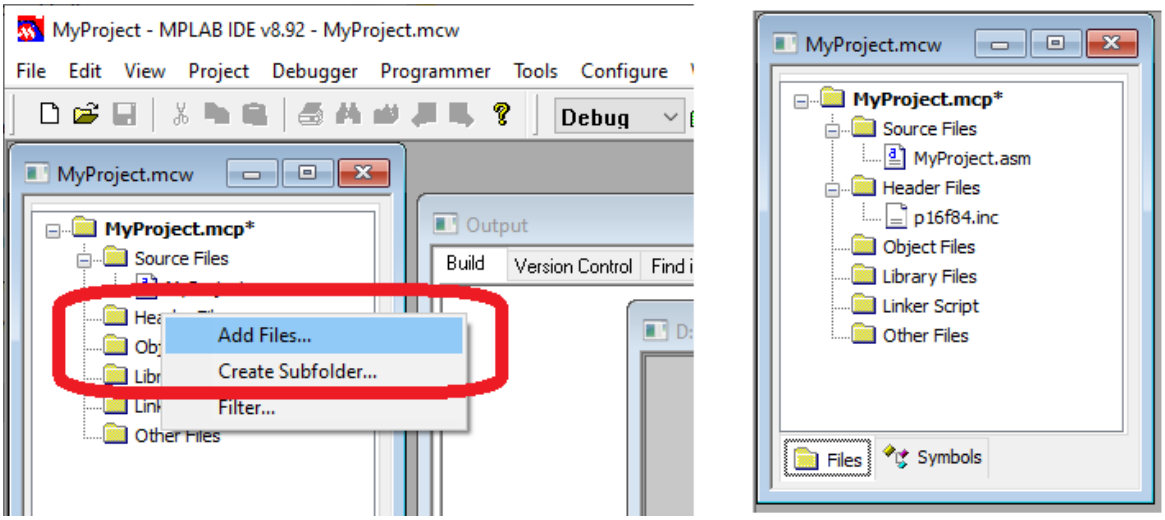


Рисунок 25 – Вікно проєкту з усіма необхідними файлами

Зберігаємо усі зміни шляхом вибору пункту «Save Workspace» у меню «File» (рис. 26).

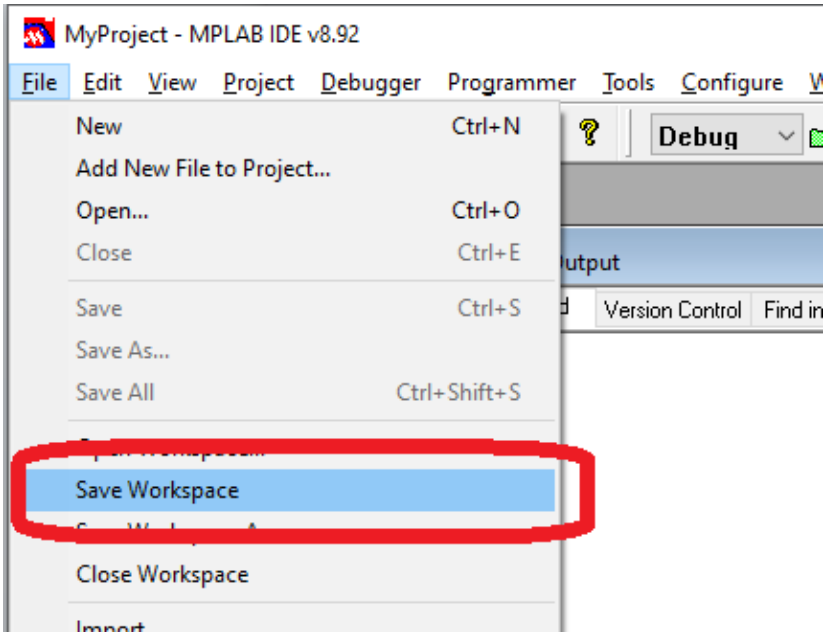


Рисунок 26 – Зберігання проєкту

На цьому підготовчі операції закінчилися і можна переходити безпосередньо до створення програмного забезпечення.

### 2.3 Написання програмного забезпечення

У вікні редактора файлу (у даному прикладі «MyProject.asm») набираємо програму, показану на наступному рисунку 27. Під час набору звертаємо увагу на наступні моменти.

1. Компілятор асемблера чутливий до регістру символів. Тому, наприклад, мітки з іменами «Loop1» та «LOOP1» будуть різними мітками.

2. Компілятор асемблера чутливий до позиції розташування тексту. Тут потрібно розуміти, що на першій позиції рядка програми повинні розташовуватися мітки (адреси куди потрібно переходити по командам goto, call тощо). Мнемонічне зображення асемблерних інструкцій рекомендується розташовувати з восьмої (або більше) позиції рядка.

3. Асемблерні інструкції можна писати малими та великими літерами. Наприклад, інструкції «incfsz» та «INCFSZ» повністю однакові.

4. Слід звертати увагу, що усі символи програми повинні бути набрані латинськими буквами. Використання кирилиці у програмному коді недопустимо (виняток – коментарі). Тому слід звертати особливу увагу на символи, що мають однакове зображення в українській та англомовній розкладках: «с», «о», «і» тощо.

5. Редактор коду вам постійно допомагає, виділяючи зрозумілі йому елементи кольором, шрифтом тощо. Звертайте на це увагу під час набору програм.

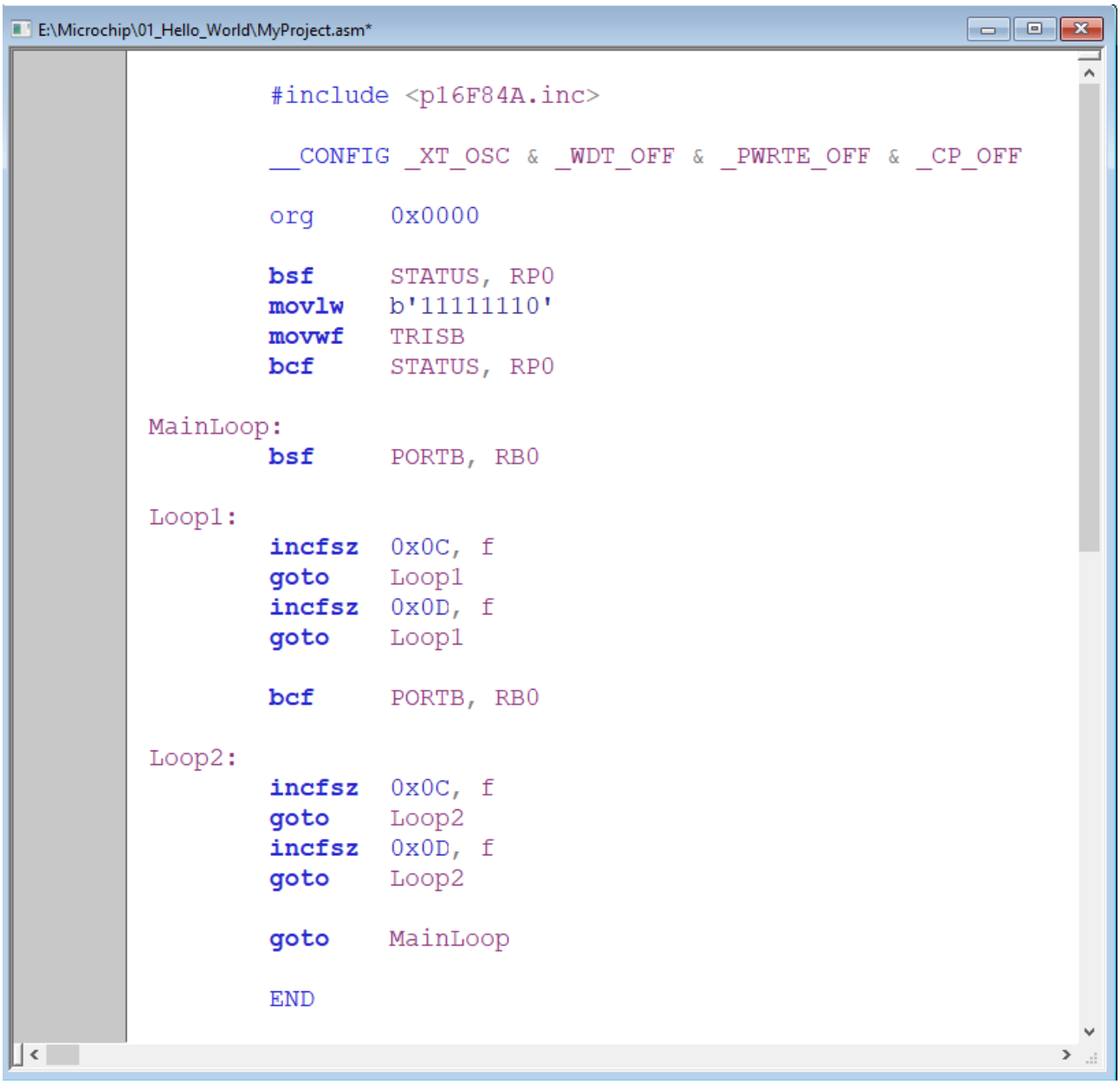


Рисунок 27 – Вихідний текст програми «Вітаю, світе!»

Після набору тексту програми його потрібно побудувати. Це можна зробити через меню Project→Build All, або натиснувши комбінацію клавіш Ctrl + F10. У процесі редагування та перевірки програми, якщо змінюється код лише в одному модулі, замість побудови проекту (яка у великих проектах може потребувати доволі велику кількість часу) можна його лише скомпілювати. Компілювання проекту можна виконати через меню Project→Make, або натиснувши клавішу F10 (рис. 28).

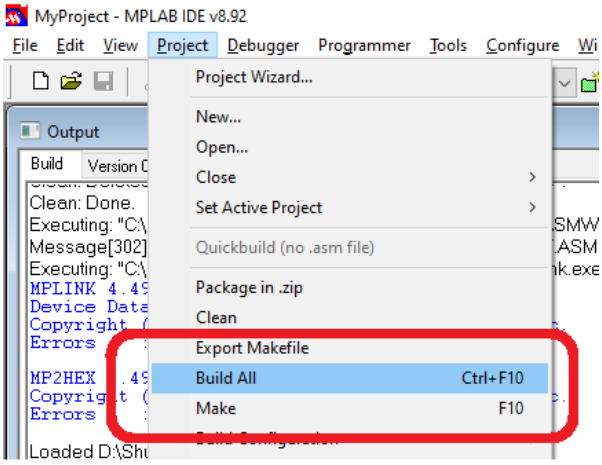


Рисунок 28 – Побудова проєкту

Результат компілювання проекту буде виведено у вікні Output. Якщо текст програми не містить помилок, то компілювання пройде успішно і в кінці з’явиться надпис «Build Succeeded» (рис. 29).

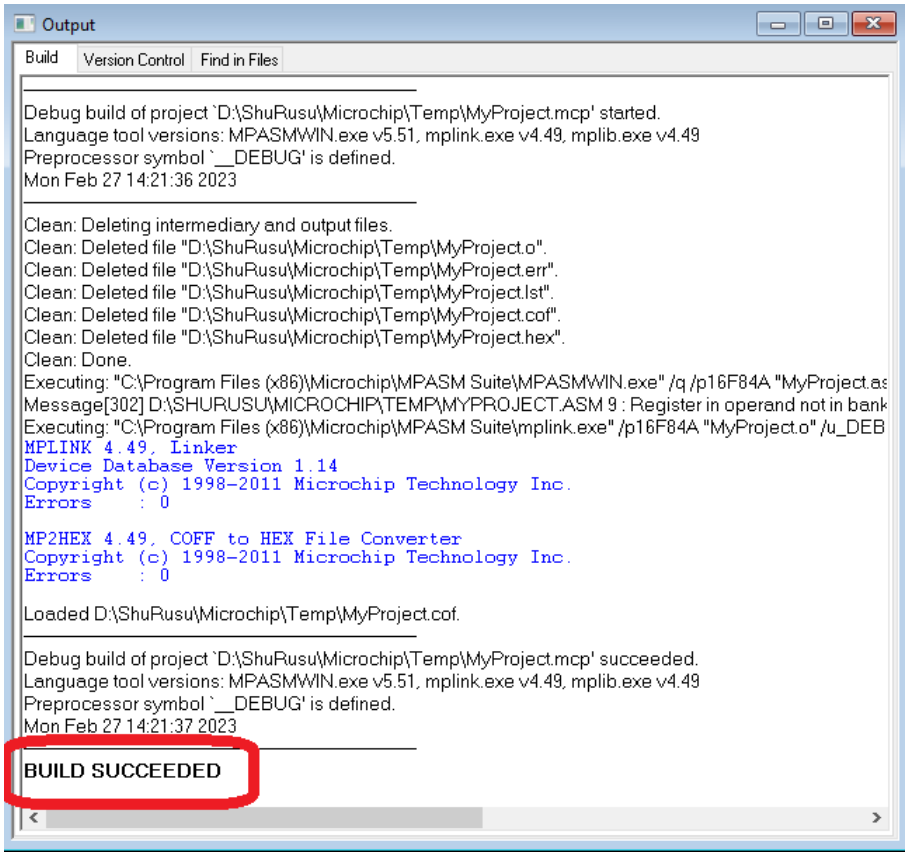


Рисунок 29 – Результат побудови проєкту без помилок

Якщо програма містить помилки, то з’явиться повідомлення «Build filed» і буде виведено перелік знайдених помилок. У наведеному прикладі компілятор повідомив, що символ «LOop1» не був попередньо визначений. У даному випадку він просто був неправильно набраний і його потрібно виправити на «Loop1». Перейти на рядок, що містить помилку, можливо двічі клацнувши на ньому у вікні Output (рис. 30).

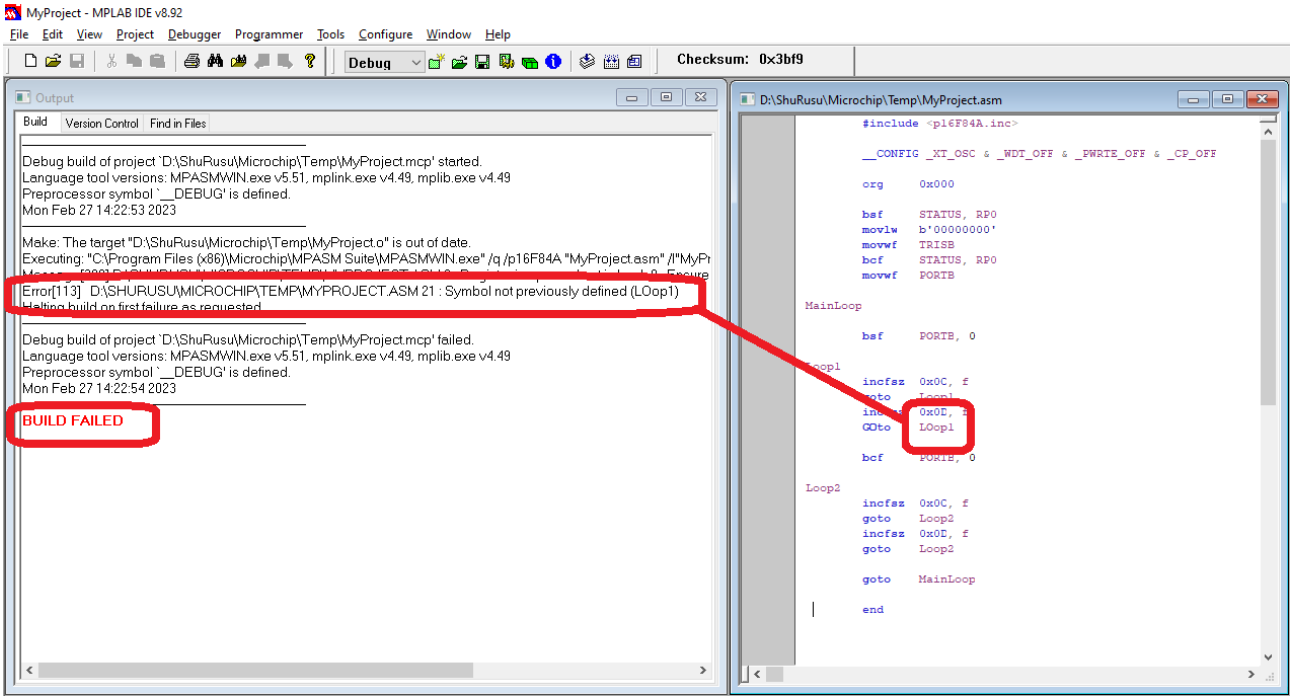


Рисунок 29 – Результат побудови проєкту з помилками

Результатом компіляції програми є текстовий файл «MyProject.hex», який містить машинні коди закодовані у шістнадцятковому форматі, що будуть завантажені у мікроконтролер. Цей файл на жаргоні називають «прошивкою мікроконтролера». Для даного проекту файл «MyProject.hex» виглядає наступним чином (рис. 30). Його можна переглянути та редагувати у будь якому текстовому редакторі, наприклад, «Блокнот»).

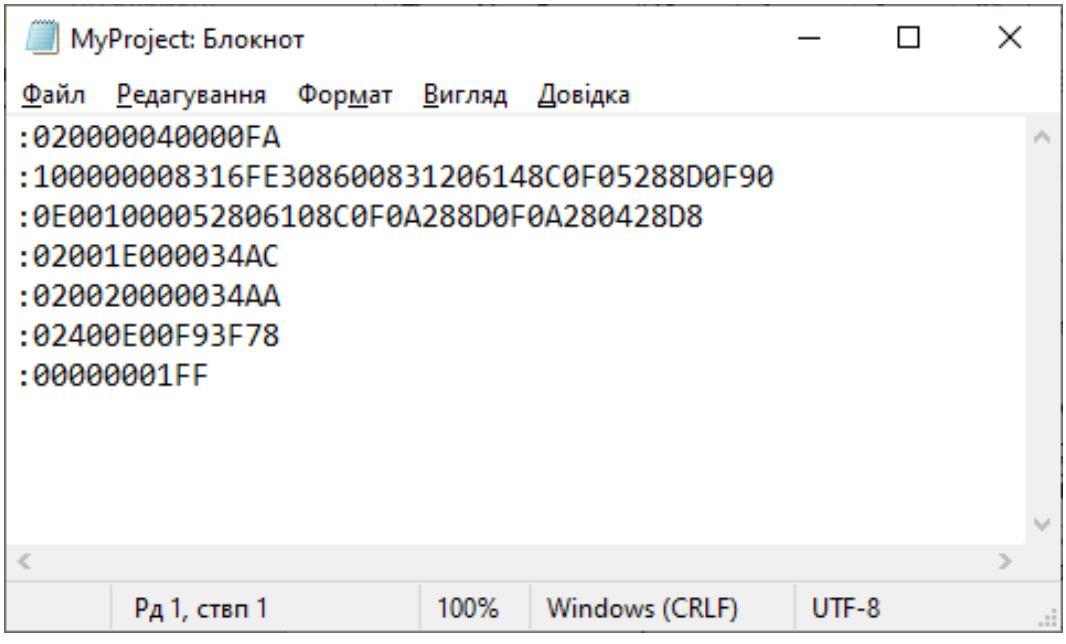


Рисунок 30 – Зміст файлу «MyProject.hex»

### 2.4 Перевірка роботи програмного забезпечення

Розроблене програмне забезпечення «Вітаю, світе!» змінює стан порту вводу-виводу RB0, з'єднаного із виводом 6 мікроконтролера, з частотою приблизно 2 Гц. Таким чином, на виводі 6 напруга буде періодично змінюватися від 0 В до напруги живлення мікроконтролера (5 В). Таким чином, якщо підключити до цього виводу, наприклад, світлодіод, то він буде засвічуватися та згасати із частотою зміни напруги на виводі 16.

Перевірити роботу програмного забезпечення можна любим із наступних способів:

– завантажити файл «MyProject.hex» у реальний мікроконтролер, що підключений до реальної схеми та перевірити його роботу (найкращій варіант);

– скористатися відомими симуляторами роботи мікроконтролерів, наприклад, Proteus або вбудованого симулятора MPLAB;

– скористатися макетами віртуальної лабораторії PIC_Lab, розробленої викладачами факультету кібербезпеки, програмної інженерії та комп'ютерних наук Міжнародного гуманітарного університету.

Останній варіант є найбільш прийнятними, особливо для здобувачів, що навчаються дистанційно. Тому далі розглянемо варіант перевірки програми у віртуальній лабораторії PIC_Lab.

### 2.5 Перевірка роботи програмного забезпечення за допомогою віртуальної лабораторії PIC_Lab

Зовнішній вигляд макету віртуальної лабораторії PIC_Lab для перевірки практичної роботи «Вітаю, світе!» показаний на рис 31. Якщо після запуску віртуальної лабораторії PIC_Lab відображається інший макет, то слід зайти в меню «Файл» та обрати макет «Вітаю, світе!» (рис. 32). За необхідності, віртуальну лабораторію PIC_Lab можна налаштувати, обравши мову інтерфейсу та номер варіанту електричної схеми у діалоговому вікні, що відкривається через меню «Налаштування» → «Налаштування системи» (рис. 33).

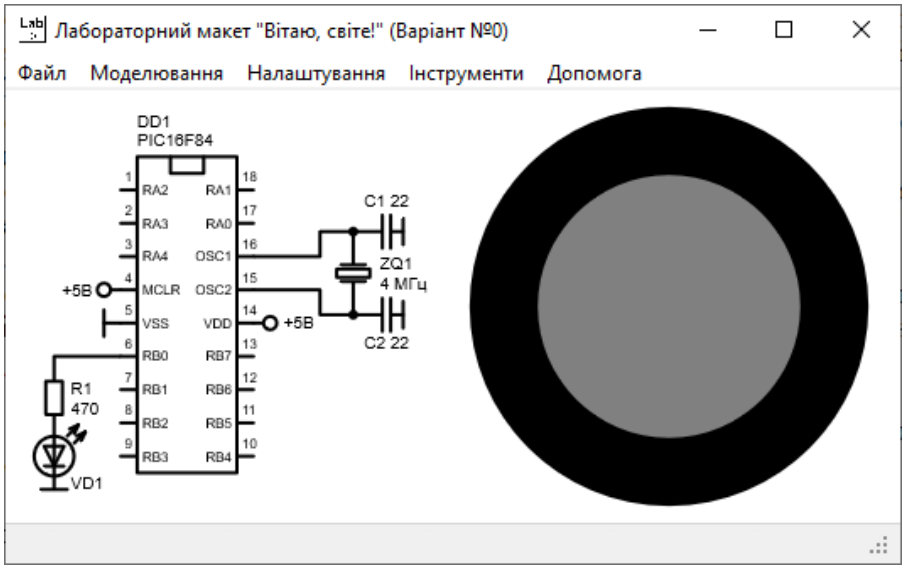


Рисунок 31 – Зовнішній вигляд макету для перевірки практичного завдання «Вітаю, світе!»

В лівій частині головного вікна макету «Вітаю, світе» розташована його електрична схема, а в правій – монітор для відображення результатів роботи максимально наближений до його вигляду у реальному житті. Електрична схема макету складається із мікроконтролера DD1, що вже підключений до джерела живлення, який на схемі не показаний – показані лише умовні точки підключення до його полюсів «⊥» та «+5В». Частота роботи мікроконтролера визначається кварцовим резонатором ZQ1 і дорівнює 4 МГц.

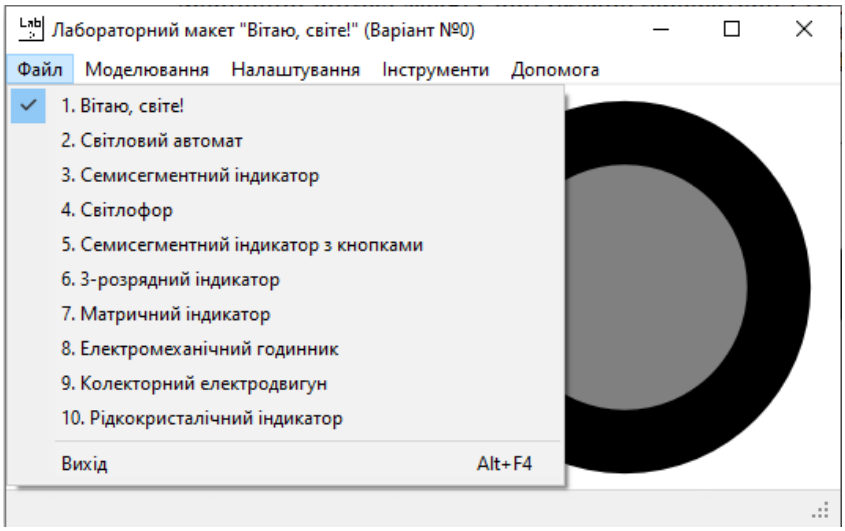


Рисунок 32 – Вибір потрібного макету через меню «Файл»

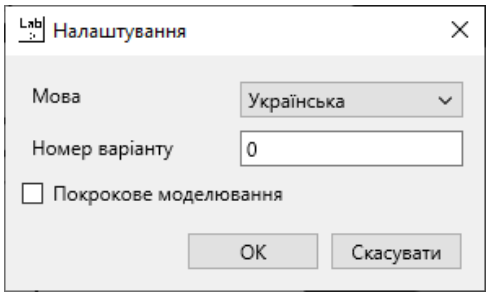


Рисунок 33 – Вікно налаштувань віртуальної лабораторії PIC_Lab

Єдиний світлодіодний індикатор VD1 підключений до виводу 6 мікроконтролера DD1 через струмообмежувальний резистор R1. За необхідності колів світіння індикатора можна змінити у діалоговому вікні «Налаштування макету», що відкривається через меню «Налаштування» → «Налаштування макету» (рис. 34). У цьому ж вікні можна обрати файл прошивки мікроконтролера (MyProject.hex), який був створений в IDE MPLAB. Файл прошивки мікроконтролера також можна змінити, розташувавши курсор миші на його умовному зображенні на електричній схемі (при цьому він буде підсвічений червоним кольором) і двічі натиснувши ліву кнопку миші.

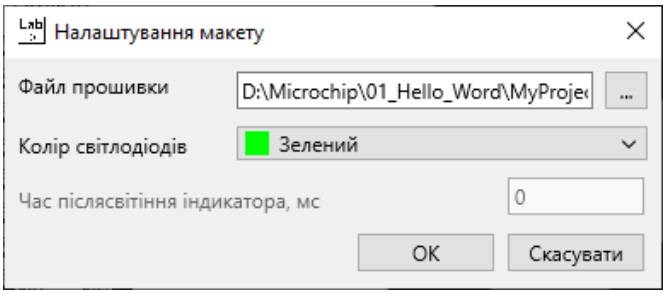


Рисунок 34 – Вікно налаштувань макету «Вітаю, світе!»

Для запуску мікроконтролера (початку симуляції) слід натиснути кнопку «F2» на клавіатурі. Для повної зупинки – кнопку «F3». Для тимчасової зупинки моделювання слід натиснути кнопку «F4». Усі зазначені дії з макетом (запуск, повна зупинка, тимчасова зупинка) також доступні через меню «Моделювання» (рис. 35).

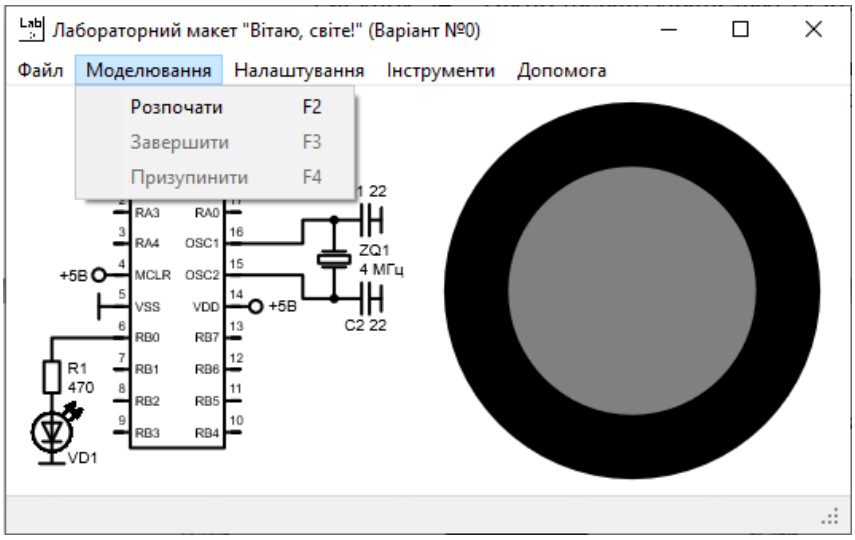


Рисунок 35 – Керування моделюванням в віртуальній лабораторії PIC_Lab

Якщо ви усе зробили правильно, то після натискання на кнопку «F2» ваш світлодіод почне мерехтіти (засвічуватися і гаснути) приблизно двічі за секунду (рис. 36). Отже вас можна привітати – ви створили першу програму для мікроконтролера, за допомогою якої можна мерехтіти світлодіодом.

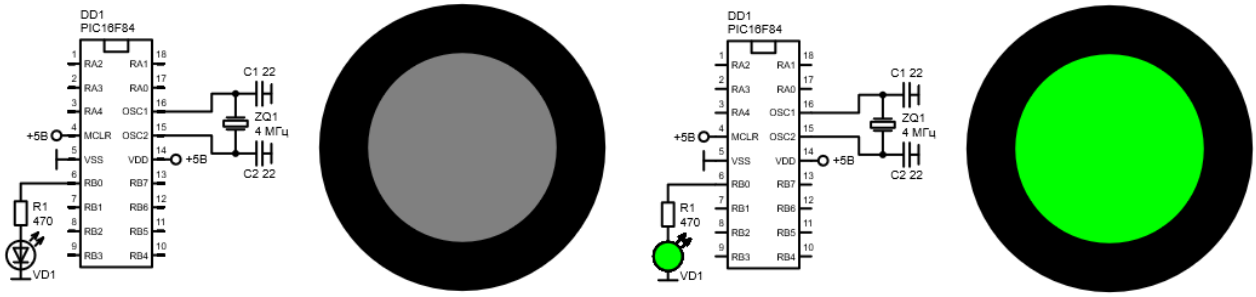


Рисунок 36 – Результат роботи програми «Вітаю, світе!»

### 3. Зміст звіту

Дана робота не містить індивідуальних завдань та завдань різнорівневої складності. Головною метою цієї роботи є опанування процесу написання та перевірки програмного коду, тому усі здобувачі, що напишуть програму і доведуть її працездатність, отримають оцінку 90 балів.

Результатом виконання цієї практичної роботи є:

– файл вихідного коду програми «xxx.asm»;

– файл програми у шістнадцятковому форматі «xxx.hex».

Ці файли завантажуються у систему дистанційного навчання і стають підтвердженням того, що ви виконали цю роботу.

# 2. АВТОМАТ СВІТЛОВИХ ЕФЕКТІВ

## 1 Мета роботи

Створити автомат світлових ефектів відповідно до індивідуального завдання.

## 2 Теоретична інформація

### 2.1 Опис апаратної частини

Апаратна частина автомату світлових ефектів складається із мікроконтролера DD1, який керує вісьмома індикаторними світлодіодами VD1 – VD8, розташованими по колу на індикаторному полі (екрані) (рис. 1).

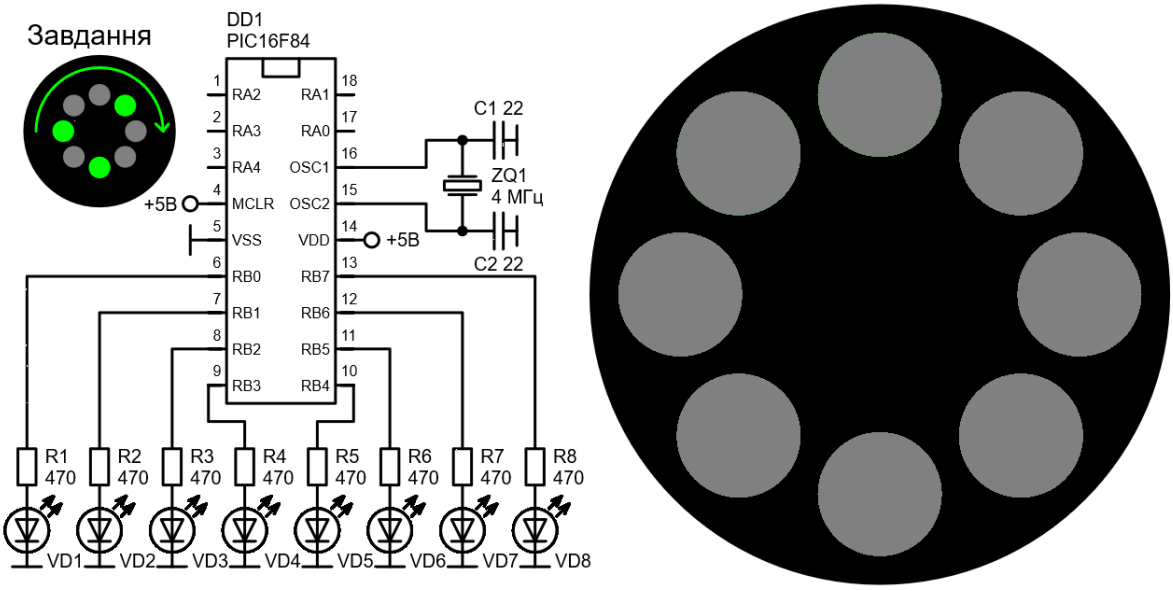


Рисунок 1 – Автомат світлових ефектів

Головним елементом апаратної частини є мікроконтролер DD1 (PIC16F84). Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цій схемі не використовується, тому він підключений до джерела живлення, і на ньому завжди присутній сигнал з рівнем логічної одиниці (+5 В). До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 4 МГц. Необхідний режим роботи кварцового резонатора та тактового генератора забезпечується за допомогою двох керамічних конденсаторів C1 та C2 з ємністю 22 пФ.

Вісім індикаторних світлодіодів VD1 – VD8 підключаються до порту B мікроконтролера (виводи 6 – 13) через струмообмежувальні резистори R1 – R8 з опором 470 Ом. Усі світлодіоди увімкнені за схемою, відповідно до якої, для засвічення світлодіода на виводі мікроконтролера необхідно згенерувати сигнал з рівнем логічної одиниці (+5 В).

У якості світлодіодів використані типові індикаторні світлодіоди з лінзами круглої форми діаметром 8…10 мм, наприклад GNL-10003xx компанії G-Nor (G-NOR OPTOELECTRONICS CO. LTD). Розташування світлодіодів на індикаторному полі наведено на рис. 2. За необхідності колір світлодіодів можна змінити у діалоговому вікні, яке можна відкрити через меню Налаштування → Налаштування макету.

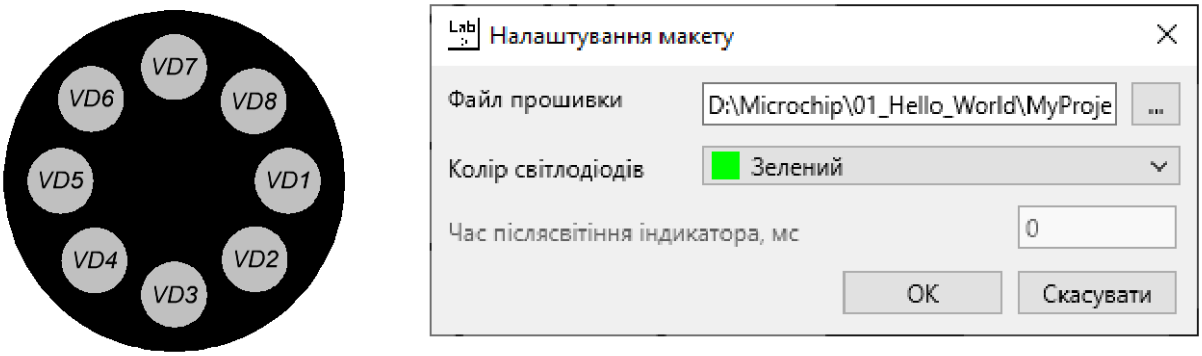


Рисунок 2 – Розташування світлодіодів на індикаторному полі та вікно налаштувань макету

### 2.2 Принцип створення світлових ефектів

Принцип створення світлових ефектів заснований на відомому принципі створення ілюзії руху шляхом швидкої зміни статичних зображень, який широко використовується в мультиплікації, кіно та системах передавання відеоінформації. Для створення ефекту руху динамічне зображення розділяється на декілька статичних зображень (фаз), які послідовно відтворюються на екрані.

У запропонованій схемі на індикаторному полі знаходиться вісім світлодіодів, які розташовані по колу. У цьому випадку, для створення ефекту руху, наприклад, «вогник, що біжить», потрібно розбити ефект на вісім фаз (за кількістю світлодіодів) і послідовно їх відтворити шляхом встановлення необхідних бітів у регістрі PORTB (рис. 3). Після створення такої програми та достатньо швидкої зміни фаз у глядач побачить вогник, що рухається по колу за годинниковою стрілкою. Щоб змінити напрям обертання (проти годинникової стрілки) потрібно змінити порядок зображень (фаз). У цьому випадку, після фази 1 потрібно відтворити фазу 8, потім фазу 7, потім фазу 6 і, врешті-решт, дійти у зворотному напрямку до фази 1.

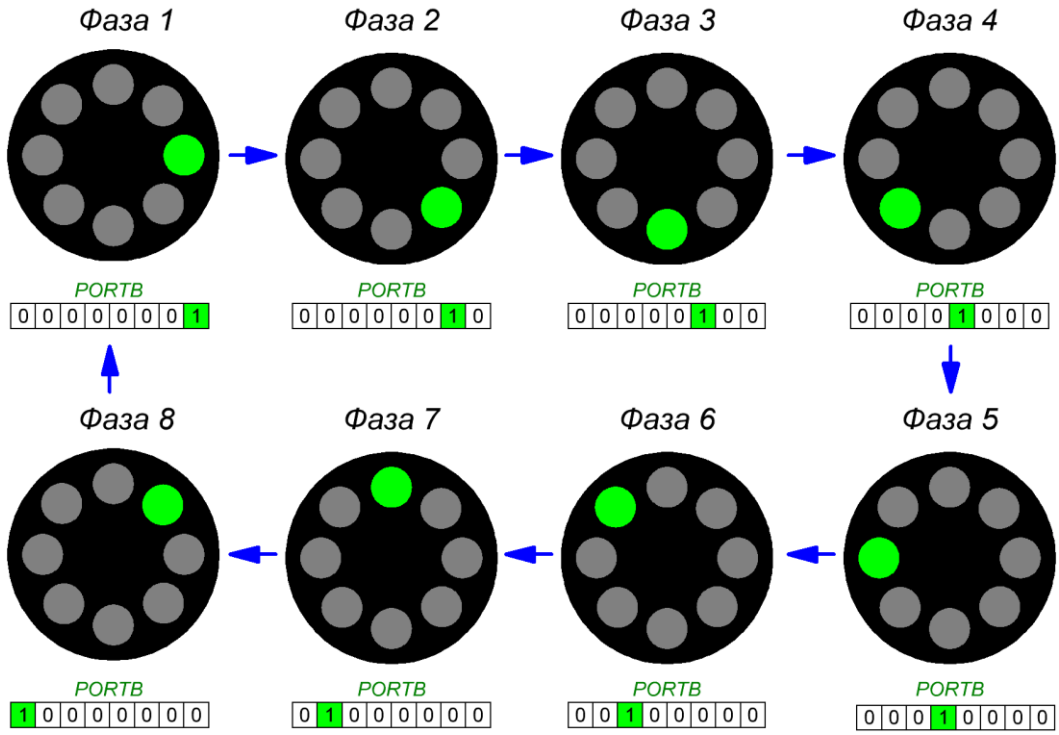


Рисунок 3 – Принцип створення світлового ефекту «Вогник, що біжить»

Кількість світлових ефектів, які можна відтворити за допомогою запропонованої схеми, обмежується лише уявою розробника. За аналогічним принципом працює достатньо велика кількість практичних пристроїв, зокрема світломузикальні пристрої для концертних залів, анімовані театральні костюми, рекламні вивіски та ялинкові гірлянди.

Рисунок 4 – Приклади використання автомату світлових ефектів: анімований сценічний танцювальний костюм (ліворуч) та ялинкові гірлянди (праворуч)

### 3 Завдання на практичну роботу

Створити автомат світлових ефектів типу «Вогні, що біжать». Конкретний вид ефекту, який потрібно реалізувати, залежить, від номеру варіанту. Номер варіанту визначає викладач.

При створенні ефекту слід звертати уваги на наступні моменти:

– взаємне розташування світлодіодів, що світяться та не світяться;

– напрямок обертання ефекту: за годинниковою стрілкою або проти неї.

Щоб побачити індивідуальне завдання необхідно встановити номер варіанту лабораторного макету. Для цього слід відкрити діалогове вікно через меню Налаштування → Налаштування системи (рис. 5).

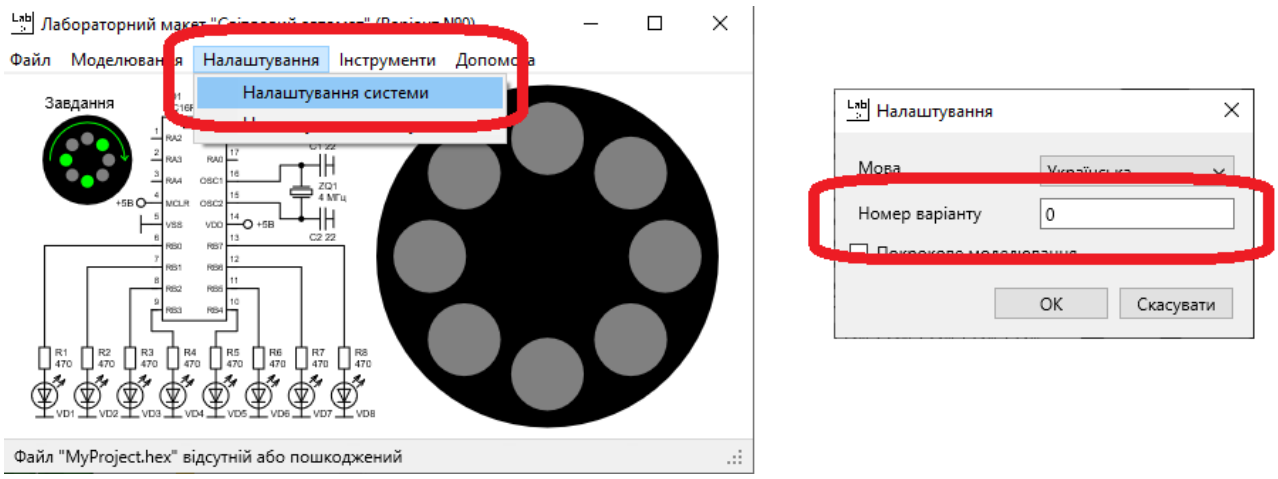


Рисунок 5 – Встановлення номеру варіанту лабораторного макету

Після цьому у лівому верхньому куті лабораторного макету вам буде показано завдання – світловий ефект, який вам потрібно буде реалізувати (рис. 6).

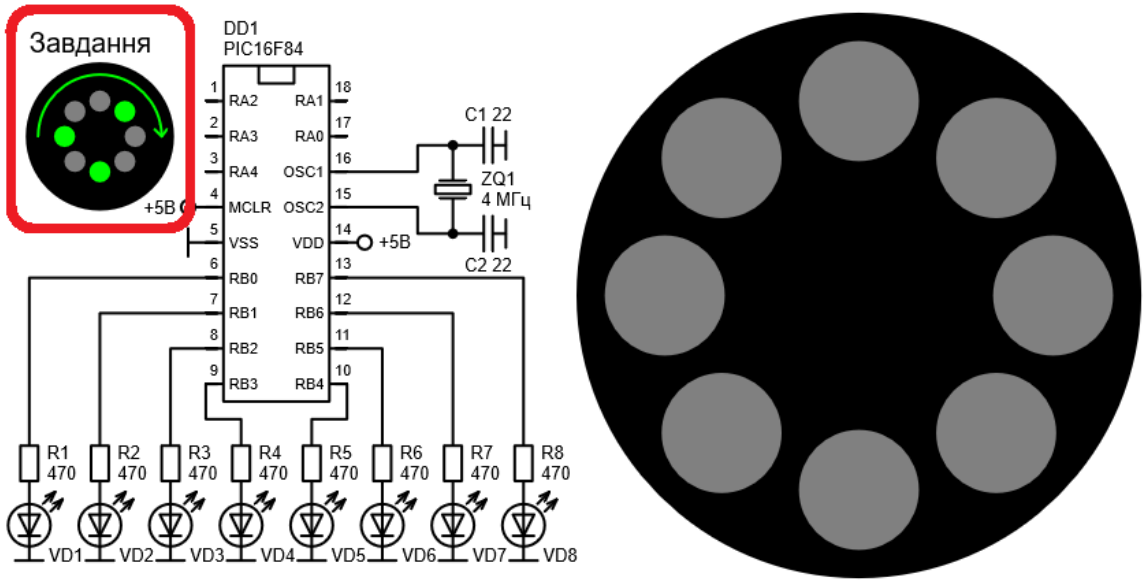


Рисунок 6 – Індивідуальне завдання, яке потрібно реалізувати

Для отримання оцінки «Задовільно» (60 – 73 бали) необхідно реалізувати світловий ефект, що відповідає номеру варіанту. На рівень оцінки впливає якість написання коду, у тому числі форматування та наявність коментарів.

Для отримання оцінки «Добре» (74 – 89 балів) необхідно реалізувати два світлових ефекти: один – що відповідає номеру варіанту, а другий – любий інший, наприклад, періодична зміна напряму обертання. Світлові ефекти повинні перемикатися автоматично, наприклад, 5 кіл одного ефекту, 5 кіл другого. На рівень оцінки впливає якість написання коду, у тому числі форматування та наявність коментарів.

Для отримання оцінки «Відмінно» (90 – 100 балів) потрібно реалізувати три світлових ефекти: один – що відповідає номеру варіанту, два інших потрібно вигадати самостійно. Світлові ефекти повинні перемикатися автоматично, наприклад 5 кіл один ефект, 5 кіл другий, а потім третій (він не обов'язково повинен бути циклічним). На рівень оцінки впливає рівень креативності, якість написання коду, форматування та наявність коментарів.

## 4 Рекомендації по розробці програмного забезпечення

### 4.1 Створення проєкту

Для написання програмного забезпечення в середовищі MPLAB слід створити новий проєкт. Це можна зробити або у відповідності до рекомендацій, наведених в методичних вказівках до практичної роботи «Програма «Вітаю, Світе!»», або шляхом копіювання файлів попередніх проєктів, наприклад, вже створеного проєкту «Програма «Вітаю, Світе!» в окрему папку (рис. 7). Рекомендовано використання другого варіанту, оскільки він займає менше часу.

### 4.2 Загальний алгоритм програмного забезпечення

Загальний алгоритм програмного забезпечення проєкту «Автомат світлових ефектів» аналогічний алгоритму програмного забезпечення проєкту «Програма «Вітаю, Світе!». У найпростішому варіанті (для отримання оцінки «Задовільно») алгоритм програмного забезпечення буде складатися з 18 кроків.

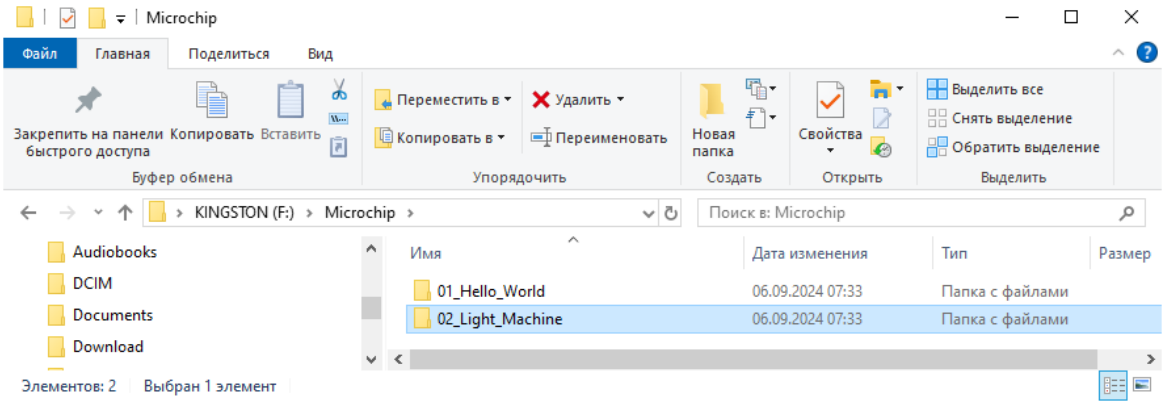


Рисунок 7 – Збереження файлів проєктів практичних робіт навчального курсу

1. Налаштувати усі канали порту B для роботи в режимі виведення.
2. Відтворити зображення, що відповідає фазі 1.
3. Зачекати 0,2…0,5 с.
4. Відтворити зображення, що відповідає фазі 2.
5. Зачекати 0,2…0,5 с.
6. Відтворити зображення, що відповідає фазі 3.
7. Зачекати 0,2…0,5 с.
8. Відтворити зображення, що відповідає фазі 4.
9. Зачекати 0,2…0,5 с.
10. Відтворити зображення, що відповідає фазі 5.
11. Зачекати 0,2…0,5 с.
12. Відтворити зображення, що відповідає фазі 6.
13. Зачекати 0,2…0,5 с.
14. Відтворити зображення, що відповідає фазі 7.
15. Зачекати 0,2…0,5 с.
16. Відтворити зображення, що відповідає фазі 8.
17. Зачекати 0,2…0,5 с.
18. Перейти до кроку 2.

Алгоритми програмного забезпечення для отримання більш високих оцінок («Добре», «Відмінно») будуть мати більше кроків (реальна кількість кроків визначається самостійно).

### 4.2 Налаштування порту B

Налаштування порту B відбувається шляхом запису відповідної константи в регістр TRISB. Для даної схеми необхідно, щоб усі канали порту B працювали в режимі виведення, що забезпечується встановленням усіх бітів в регістрі TRISB у значення «0». При зверненні до регістру TRISB, що має адресу 0x86, слід пам’ятати, що він знаходиться на першій сторінці пам’яті (Bank 1), тому перед зверненням до нього слід спочатку встановити біт RP0 у регістрі STATUS у значення «1» (рис. 8).

### 4.3 Керування портом B

Після налаштування усіх каналі порту B для роботи у режимі виведення, встановлення будь-якого біту у значення «1» призведе до засвічення світлодіода, підключеного до виводу мікроконтролера, пов’язаного з цим каналом порту, а встановлення біту у значення «0» – до його згасання. Для зміни зображення необхідно одночасно змінювати стан кількох бітів. Найпростіше це зробити за допомогою пари асемблерних команд movlw + movwf – так само, як і при налаштуванні регістру TRISB. У цьому випадку ви одночасно встановите в необхідне значення усі біти регістру PORTB (рис. 9).

```
bsf     STATUS, RP0
movlw   b'00000000'
movwf   TRISB
bcf     STATUS, RP0
```

Рисунок 8 – Рекомендований код ініціалізації порту B

```
movlw   b'00000001'
movwf   PORTB
```

Рисунок 9 – Рекомендований код керування портом B

### 4.4 Формування затримки

Затримку 0,2…0,5 с на цьому етапі опанування мистецтвом програмування мікроконтролерів рекомендується виконувати так само, як і в попередній практичній роботі «Програма, «Вітаю, Світе!» – шляхом виконання певної кількості «порожніх» операції. У даному випадку рекомендується 256 раз збільшити значення комірок пам'яті з адресами 0x0C та 0x0D. При вставленні коду затримки після кожної фази слід правильно встановити назви міток, щоб уникнути неправильної роботи програми (рис. 10).

```
        ; код відтворення зображення 5
        ; . . .

        ; затримка після зображення 5
Loop5:
        incfsz  0x0C, f
        goto    Loop5
        incfsz  0x0D, f
        goto    Loop5

        ; код відтворення зображення 6
        ; . . .

        ; затримка після зображення 6
Loop6:
        incfsz  0x0C, f
        goto    Loop6
        incfsz  0x0D, f
        goto    Loop6

        ; код відтворення зображення 7
        ; . . .
```

Рисунок 10 – Рекомендований код формування затримки 0,2…0,5 с

## 5. Зміст звіту

Результатом виконання цієї практичної роботи є:
– файл вихідного коду програми «xxx.asm»;
– файл програми у шістнадцятковому форматі «xxx.hex».
Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

# 3. СВІТЛОФОР

## 1 Мета роботи

Створити застосунок для регулювання дорожнього руху – світлофор.

## 2 Опис апаратної частини

Апаратна частина застосунку складається із мікроконтролера DD1, який шістьма потужними лампами HL1 – HL6, які утворюють дві секції світлофора (рис. 1).

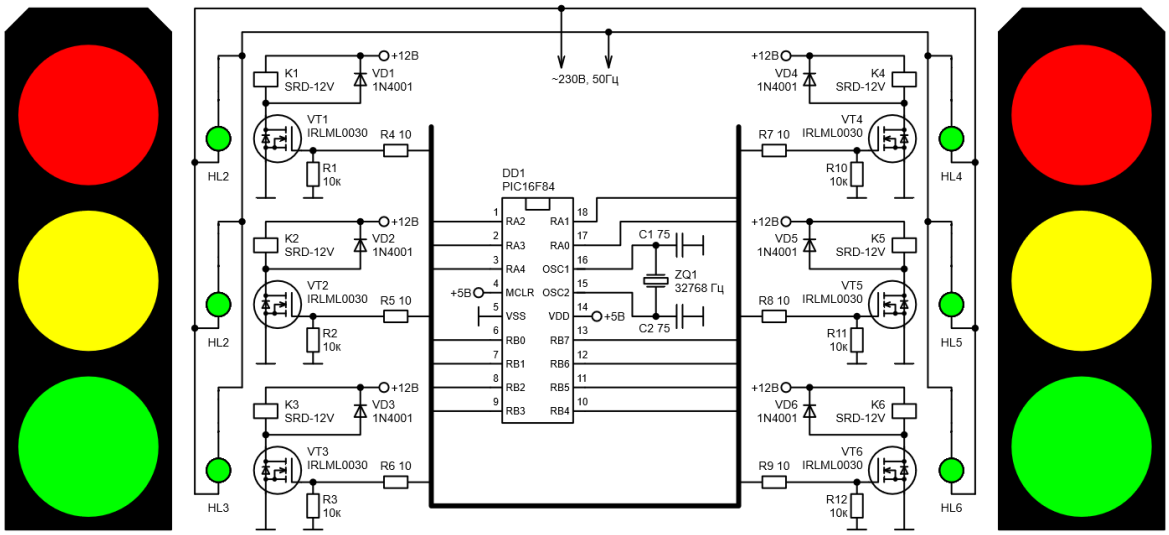


Рисунок 1 – Електрична схема апаратної частини світлофора

Головним елементом апаратної частини застосунку є мікроконтролер DD1 (PIC16F84). Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цій схемі не використовується, тому він підключений до джерела живлення, і на ньому завжди присутній сигнал з рівнем логічної одиниці (+5 В). До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 32768 Гц (годинниковий кварц). Необхідний режим роботи кварцового резонатора та тактового генератора забезпечується за допомогою двох керамічних конденсаторів C1 та C2 з ємністю 75 пФ.

Потужні лампи HL1 – HL6, розраховані на живлення від промислової мережі змінного струму із напругою 230 В та частотою 50 Гц, підключаються до мережі за допомогою реле K1 – K6 (SRD-12V), які, у свою чергу, підключаються до мікроконтролера DD1 за допомогою підсилювачів, реалізованих на n-канальних польових транзисторах із ізольованим затвором (Metal-Oxide-Semiconductor Field-Effect Transistor, MOSFET) VT1 – VT6 (IRLML0030). Діоди VD1 – VD6 (1N4001) призначені для запобігання електричного пробою транзисторів електрорушійною силою, що виникає на виводах реле під час їх вимкнення. Резистори R1 – R3, R10 – R13 з опором 10 кОм призначені для утримання транзисторів VT1 – VT6 у непровідному стані під час перезавантаження мікроконтролера, коли усі канали усіх портів введення-виведення мікроконтролера DD1 знаходяться у режимі введення. Резистори R4 – R9 з опором 10 Ом обмежують струми в колах затворів транзисторів VT1 – VT6 під час їх перемикання. Робоча напруга реле K1 – K6 дорівнює 12 В, тому колекторні кола транзисторів підключені до джерела живлення із напругою +12 В.

### 3 Завдання на практичну роботу

Створити програму, таким чином, щоб лампи світлофора світилися, відповідно до алгоритму, показаному на рисунку 2. Програму необхідно створити для свого варіанту лабораторного макету.

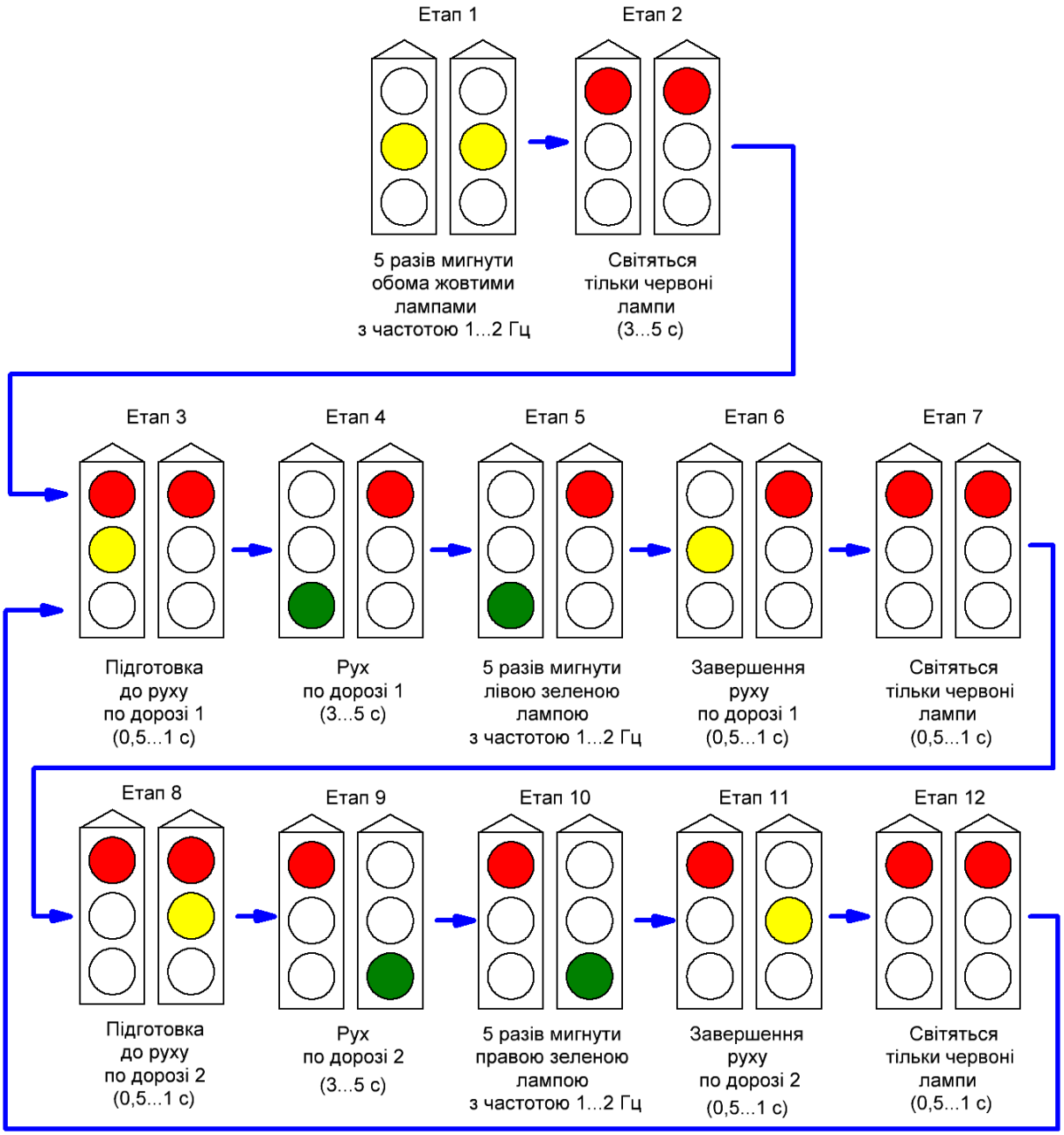


Рисунок 2 – Завдання на практичну роботу

Встановити необхідний номер варіанту лабораторного макету можна через меню Налаштування → Налаштування системи. При цьому відкриється діалогове вікно, у якому можна буде встановити необхідний номер варіанту (рис. 3).

Для отримання оцінок «Задовільно» (60 – 73 бали) та «Добре» (74 – 89 балів) необхідно виконати практичне завдання в повному обсязі.

Для отримання оцінки «Відмінно» (90 – 100 балів) потрібно реалізувати Більш складний алгоритм роботи світлофора (на власний розсуд).

На рівень оцінки впливає якість написання коду, у тому числі використання підпрограм, форматування та наявність коментарів.

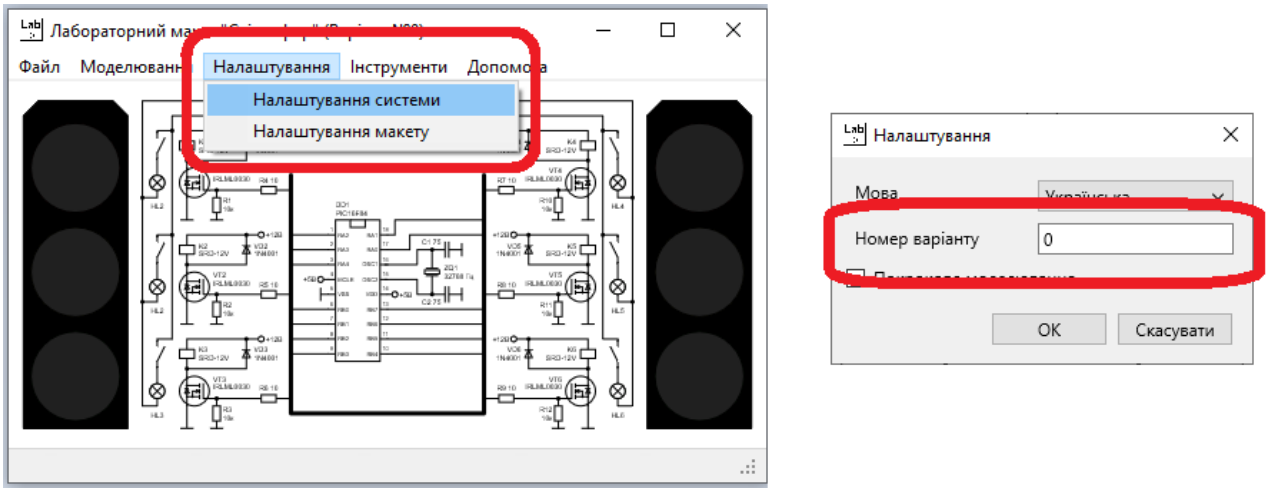


Рисунок 3 – Встановлення номеру варіанту лабораторного макету

## 4 Рекомендації по розробці програмного забезпечення

Алгоритм та вихідний код програмного забезпечення потрібно розробити самотужки, використовуючи усі знання та вміння, які були отримані під час попередніх лекційних та практичних занять.

### 4.1 Особливості апаратної частини

При створенні програмного забезпечення слід звертати увагу на наступні особливості схеми, які відрізняються від попередніх практичних занять.

1. Частота роботи тактового генератора мікроконтролера стабілізована низькочастотним кварцовим резонатором із частотою 32768 Гц, тому у слові конфігурації мікроконтролера біти, що відповідають за режим роботи тактового генератора, потрібно встановити в значення, що відповідає режиму LP (Low Power).

Для цього в слові конфігурації мікроконтролера (після директиви __CONFIG) потрібно вказати за допомогою константи _LP_OSC, що використовується низькочастотний кварцовий резонатор (рис. 4) (у попередніх практичних роботах використовувався високочастотний кварцовий резонатор з частотою 4 МГц, для якого у слові конфігурації потрібно було використовувати константи _XT_OSC або _HS_OSC).

```
__CONFIG _LP_OSC & _WDT_OFF & _PWRTE_OFF & _CP_OFF
```

Рисунок 4 – Слово конфігурації для макету «Світлофор»

2. Оскільки тактова частота дорівнює 32768 Гц, то тривалість одного командного циклу (час виконання більшості асемблерних команд) буде дорівнювати:

$$T_C = \frac{4}{32768} \approx 122 \text{ мкс}.$$

Це у 122 разів повільніше ніж при роботі на частоті 4 МГц (коли тривалість одного командного циклу становила 1 мкс). Цю різницю слід враховувати під час створення коду, що формує затримки у часі. Наприклад, код, що формував затримку у попередніх практичних роботах (рис. 5), на частоті 4 МГц виконувався приблизно за 0,2 с. У цьому макеті – на частоті 32768 Гц він буде виконуватись приблизно за 24 с – це дуже великий проміжок часу, який не відповідає технічному завданню.

```
Loop1:
    incfsz  0x0C, f
    goto    Loop1
    incfsz  0x0D, f
    goto    Loop1
```

Рисунок 5 – Код формування затримки у попередніх практичних роботах

Для зменшення часу затримки треба зменшити кількість ітерації зовнішнього циклу. Це можна зробити шляхом зменшення кількості ітерацій зовнішнього циклу. Наприклад, якщо прибрати зовнішній цикл і залишити лише внутрішній, то він на частоті 32768 Гц буде виконуватись приблизно за 0,094 с (рис. 6). Збільшити тривалість затримки можна шляхом попереднього завантаження у комірку зовнішнього циклу (у прикладі рис. 6 – комірка з адресою 0x0D) якогось числа. Наприклад, попереднє завантаження в комірку оперативної пам'яті з адресою 0x0D числа 2 призведе до подвоєння часу виконання коду (0,18 с), числа 3 – до потроєння (0,28 с), а числа 255 – до збільшення часу виконання коду у 255 разів (23,9 с).

```
Loop1:
    decfsz  0x0C, f
    goto    Loop1
```

0,094 с

```
    movlw   d'2'
    movwf   0x0D
Loop1:
    decfsz  0x0C, f
    goto    Loop1
    decfsz  0x0D, f
    goto    Loop1
```

0,18 с

```
    movlw   d'5'
    movwf   0x0D
Loop1:
    decfsz  0x0C, f
    goto    Loop1
    decfsz  0x0D, f
    goto    Loop1
```

0,47 с

```
    movlw   d'255'
    movwf   0x0D
Loop1:
    decfsz  0x0C, f
    goto    Loop1
    decfsz  0x0D, f
    goto    Loop1
```

23,9 с

Рисунок 6 – Час виконання різних фрагментів коду на частоті 32768 Гц

3. Транзистори VT1 – VT6 увімкнені за схемою із загальним витоком, тому для того, щоб їх відкрити і, відповідно, засвітити лампу на виході каналу введення-виведення потрібно сформувати сигнал із рівнем, що відповідає рівню логічної одиниці.

4. Схема підключення транзисторів VT1 – VT6 до портів введення-виведення мікроконтролера DD1 залежить від номеру варіанту. При цьому транзистори VT1 – VT6 можуть підключатися як до порту PORTA, так і до порту PORTB. Який транзистор і, відповідно, яка лампа підключена до якого каналу введення-виведення доведеться з'ясовувати експериментально.

Для цього рекомендується спочатку написати тестовий код, за допомогою якого визначити до яких каналів підключені певні лампи. Для цього спочатку потрібно налаштувати усі канали усіх портів мікроконтролера для роботи у режимі виведення, а потім шляхом послідовного встановлення бітів у регістрах PORTA та PORTB експериментально визначити яка лампа підключена до якого каналу порту (рис. 7). Результати експерименту рекомендується записати у таблицю 1.

```
org     0x0000

bsf     STATUS, RP0
movlw   b'00000000'
movwf   TRISA
movwf   TRISB
bcf     STATUS, RP0
```

```
MainLoop:
    movlw   b'00000001'
    movwf   PORTA
    goto    MainLoop

    END
```

Рисунок 7 – Тестовий код, який визначає яка лампа підключена до каналу RA0

Таблиця 1 – Результати визначення схеми світлофора

| Секція | Лампа | Порт | Канал |
|---|---|---|---|
| Ліва | Червона | | |
| | Жовта | | |
| | Зелена | | |
| Права | Червона | | |
| | Жовта | | |
| | Зелена | | |

### 4.2 Рекомендації по створенню програмного забезпечення

1. Створити новий проєкт або скопіювати проєкт з попередніх практичних робіт.
2. Встановити у слові конфігурації, що використовується низькочастотний кварцовий резонатор.
3. Написати тестовий код ініціалізації, в якому налаштувати усі канали усіх портів для роботи в режимі виведення.
4. Написати тестовий основний цикл, в якому послідовно встановлювати на виході усіх каналів сигнал з рівнем логічної одиниці. При цьому слід контролювати до якого каналу підключена та чи інша лампа. Результати занести в табл. 1.
5. Модифікувати код ініціалізації, в якому залишити в режимі виведення лише ті канали, які задіяні в вашому варіанті апаратної частини світлофора.
6. Написати підпрограму, що буде формувати базову затримку, наприклад, 0,5 с.
7. Написати підпрограму, що буде формувати довільну затримку, кратну базовій (0,5, 1, 1,5 2 с тощо)
8. Реалізувати алгоритм перемикання ламп світлофора, відповідно до технічного завдання.
9. Перевірити працездатність програмного коду.
10. Завантажити необхідні файли проєкту в системи дистанційної освіти для перевірки викладачем.

## 5. Зміст звіту

Результатом виконання цієї практичної роботи є:
– файл вихідного коду програми «xxx.asm»;
– файл програми у шістнадцятковому форматі «xxx.hex».
Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

# 4. ЕЛЕКТРОМЕХАНІЧНИЙ ГОДИННИК

## 1 Мета роботи

Створити застосунок для вимірювання часу – електромеханічний годинник на основі крокового двигуна Лаве.

## 2 Опис апаратної частини

Апаратна частина застосунку складається із мікроконтролера DD1, що керує кроковим двигуном Лаве, зв'язаним шестернями із механізмом зі стрілками (рис. 1).

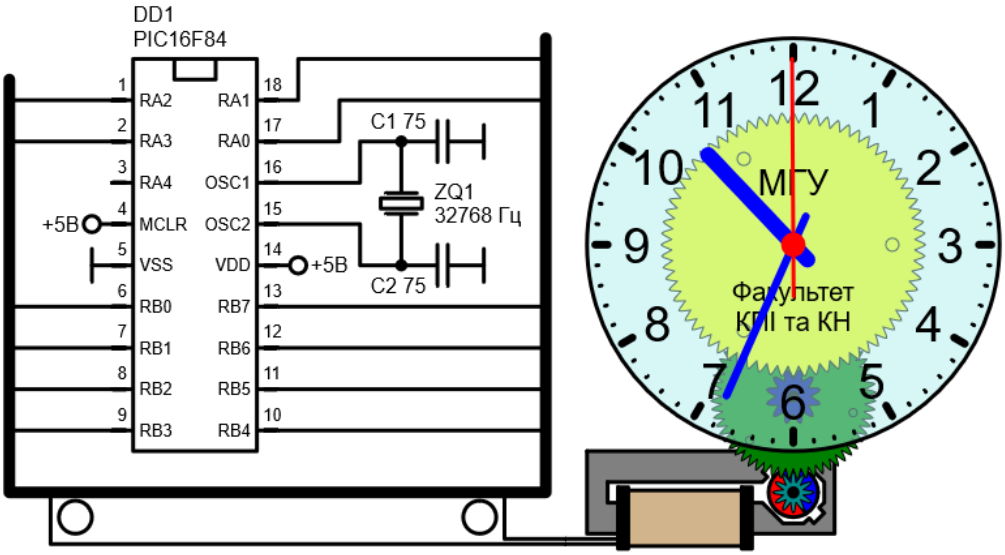


Рисунок 1 – Електрична схема апаратної частини світлофора

Головним елементом апаратної частини застосунку є мікроконтролер DD1 (PIC16F84). Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цій схемі не використовується, тому він підключений до джерела живлення, і на ньому завжди присутній сигнал з рівнем логічної одиниці (+5 В). До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 32768 Гц (годинниковий кварц). Необхідний режим роботи кварцового резонатора та тактового генератора забезпечується за допомогою двох керамічних конденсаторів C1 та C2 з ємністю 75 пФ.

Кроковий двигун Лаве підключається безпосередньо до виводів мікроконтролера, однак схема його підключення (до яких портів ведення-виведення електрично підключені його виводи) залежить від номеру варіанту. Для спрощення визначення електричної схеми біля виводів двигуна розташовані два логічні пробники, які показують логічний рівень на цьому виводі (рис. 2). Якщо вивід двигуна підключений до каналу порту введення-виведення, що налаштований для роботи в режимі введення, цифра всередині логічного пробника буде відсутня. Якщо канал порту налаштований в режимі виведення, то всередині пробника буде знаходитись цифра 0 (рівень логічного нуля) або 1 (рівень логічної одиниці). При встановлені на виводі мікроконтролера сигналу з рівнем логічної одиниці позначення пробнику буде додатково забарвлене кольором, який можна змінити в налаштуваннях макету.

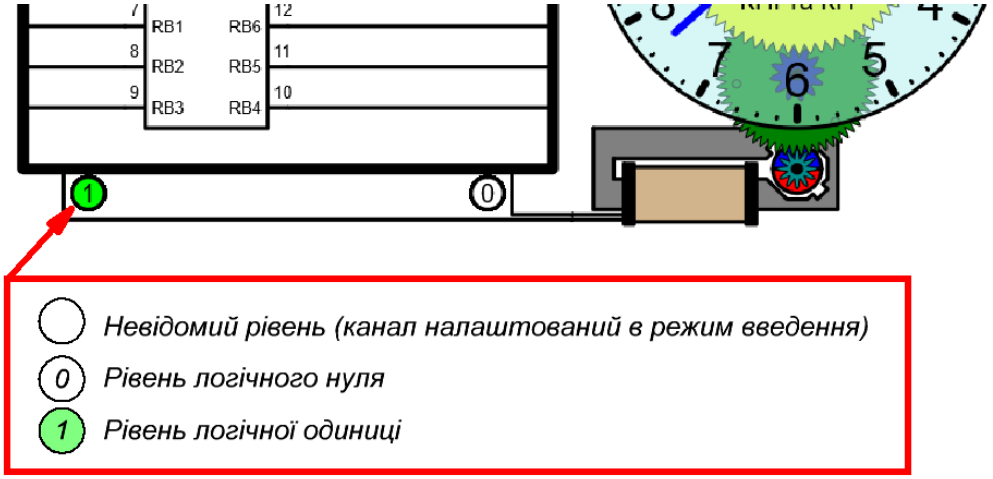


Рисунок 2 – Визначення рівня сигналу за допомогою логічного пробника

### 3 Завдання на практичну роботу

Створити програмне забезпечення для мікроконтролера, таким чином, щоб годинник почав вимірювати реальний час. Програмне забезпечення необхідно створити для свого варіанту лабораторного макету.

Встановити необхідний номер варіанту лабораторного макету можна через меню Налаштування → Налаштування системи. При цьому відкриється діалогове вікно, у якому можна буде встановити необхідний номер варіанту (рис. 3).

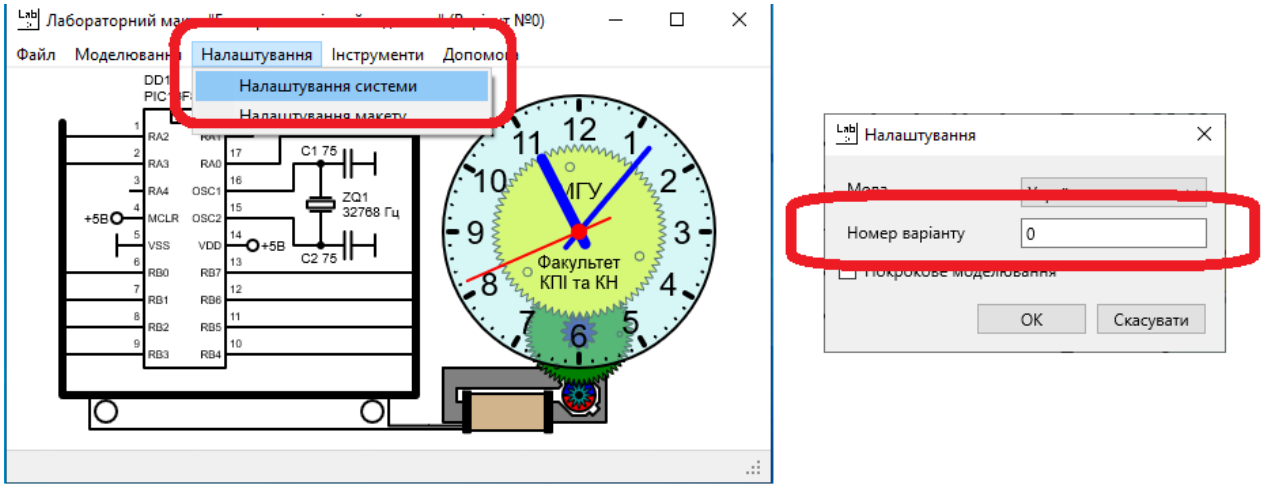


Рисунок 3 – Встановлення номеру варіанту лабораторного макету

Для отримання оцінки «Задовільно» (60 – 73 бали) можна виконати практичне завдання без використання таймерів та переривань.

Для отримання оцінок «Добре» (74 – 89 балів) та «Відмінно» (90 – 100 балів) потрібно побудувати програмне забезпечення з використанням таймерів та переривань.

На рівень оцінки впливає якість написання коду, у тому числі використання підпрограм, форматування та наявність коментарів.

### 4 Рекомендації по розробці програмного забезпечення

Алгоритм та вихідний код програмного забезпечення потрібно розробити самотужки, використовуючи усі знання та вміння, які були отримані під час попередніх лекційних та практичних занять.

### 4.1 Принцип роботи крокового двигуна Лаве

Кроковий двигун (Stepper Motor) – електричний двигун, в якому імпульсне живлення електричним струмом призводить до того, що його ротор не обертається неперервно, а виконує щоразу обертальний рух на заданий кут. Завдяки цьому, кут повороту ротора залежить від числа поданих імпульсів струму, а кутова швидкість ротора точно рівна частоті імпульсів помноженій на кут повороту ротора за один цикл роботи двигуна.

Крокові електродвигуни застосовуються в приводах машин і механізмів, що працюють у старт-стопному режимі, або в приводах безперервного руху, де позиціонування у просторі задається шляхом формування певної послідовності електричних імпульсів, наприклад, в верстатах з ЧПУ або 3D-принтерах. На відміну від сервоприводів крокові приводи дозволяють отримувати точне позиціонування без використання зворотного зв'язку від датчиків кутового положення.

Кроковий електродвигун Лаве – різновид крокового електродвигуна, що набув широкого застосування в електромеханічних годинниках (рис. 4). Цей тип двигуна винайшов і запатентував французький інженер Марюс Лаве у 1936 році.


Рисунок 4 – Двигун Лаве в електромеханічному годиннику

Двигун Лаве складається із магнітопроводу, виготовленому з трансформаторного заліза, на якому розміщена одна єдина обмотка. Оскільки вихідна потужність мікросхем електромеханічних годинників обмежена, то для забезпечення необхідного для обертання ротора магнітного потоку обмотка декілька тисяч обвитків тонкого емаль-проводу. Ротором крокового двигуна Лаве є двополюсний магніт циліндричної форми, розташований всередині статора.

При відсутності напруги на обмотці робота магнітне поле магніту ротора замикається через магнітопровід статора. При цьому орієнтація магніту самостійно встановлюється такою, щоб енергія магнітного поля була мінімальною. Через наявність розрізів в магнітопроводі магніт ротора самостійно орієнтується таким чином, щоб площина розділення полюсів магніту проходила через середину прорізів статора. При подачі напруги на обмотку статора в ньому утворюється власне магнітне поле, яке взаємодіє з магнітним полем магніту ротора, встановлюючи його в положення з мінімальною енергією магнітного поля – коли полюси магніту ротора розташовуються супротив полюсів магнітного полу статора. При знятті напруги з обмотки статора магніт ротора знову встановлюється в положення з найменшою енергією – коли площина розділення полюсів магніту проходила через середину прорізів статора.

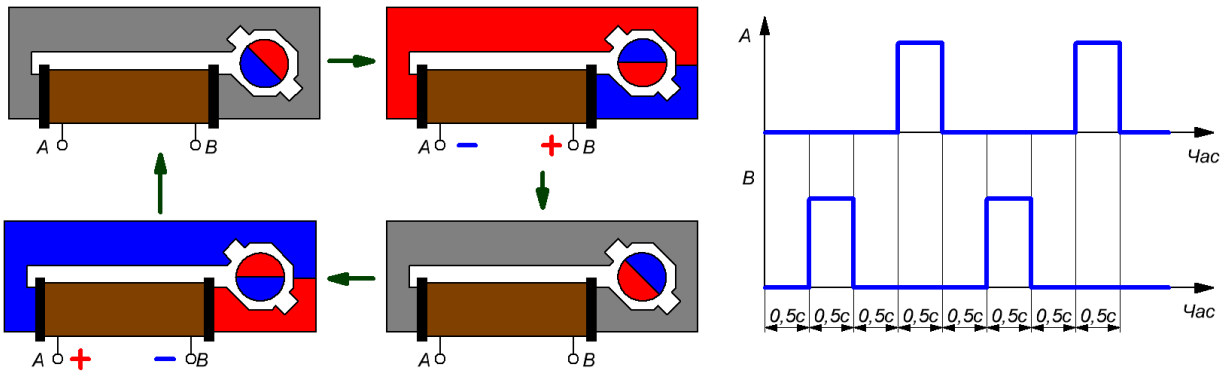


Рисунок 5 – Принцип роботи двигуна Лаве

Таким чином, цикл керування двигуном Лаве складається з чотирьох кроків, кожен з яких триває 0,5 секунд:

– подати на обмотку статора напругу позитивної полярності – для цього на виводі В потрібно сформувати сигнал з рівнем логічної одиниці, а та виході А – сигнал з рівнем логічного нуля – це призведе до обертання ротора на кут $135^0$;

– знати напругу з обмотки статора шляхом встановлення на виводах А та В сигналів з рівнем логічного нуля – це призведе до самостійного обертання ротора на кут $45^0$;

– подати на обмотку статора напругу негативної полярності – для цього на виводі А потрібно сформувати сигнал з рівнем логічної одиниці, а та виході В – сигнал з рівнем логічного нуля – це призведе до обертання ротора ще на кут $135^0$;

– знати напругу з обмотки статора шляхом встановлення на виводах А та В сигналів з рівнем логічного нуля – це призведе до самостійного обертання ротора ще на кут $45^0$.

Після цього слід розпочати наступний цикл.

Таким чином, ротор двигуна Лаве зробить повний оберт ($360^0$) за 2 секунди. Коефіцієнт передачі шестерень механізму годинника обраний таким чином, щоб секундна стрілка при обертанні робота на $180^0$ повернулася на $6^0$. Таким чином, секундна стрілка годинника зробить повний оберт за 60 секунд (1 хвилину).

### 4.2 Рекомендації по створенню програмного забезпечення

При створенні програмного забезпечення слід звертати увагу на наступні особливості схеми, які відрізняються від попередніх практичних занять.

1. Частота роботи тактового генератора мікроконтролера стабілізована низькочастотним кварцовим резонатором із частотою 32768 Гц, тому у слові конфігурації мікроконтролера біти, що відповідають за режим роботи тактового генератора, потрібно встановити в значення, що відповідає режиму LP (Low Power).

2. При використанні таймеру слід налаштувати його таким чином, щоб переривання від нього відбувалися кожні 0,5 секунд (частота – 2 Гц). Особливістю тактової частоти 32768 Гц є те, що для отримання частоти 1 Гц це число потрібно 15 разів поділити на 2, що можна достатньо просто зробити за допомогою двійкових лічильників та Т-тригерах:

$$2^{15} = 32768$$

Через це кварцові резонатори з резонансною частотою 32768 Гц отримали неофіційну назву «годинникові кварци», тому що в усіх електронних годинниках, зокрема і годинниках BIOS персональних комп'ютерів використовуються резонатори саме з такою частотою.

Для отримання частоти спрацьовування переривань 2 Гц цю частоту необхідно поділи на $2^{14} = 16384$. При цьому, відповідно до технічної документації на мікроконтролер PIC16F84, слід враховувати наступне.


Рисунок 6 – Годинниковий кварц на материнській платі комп’ютера

Тактування вузлів таймеру відбувається тактовою частотою, що вже поділена на 4.

Переповнення регістру TMR0 відбувається через $2^8 = 256$ тактів.

Таким чином, для того, щоб переповнення регістру TMR0 відбувалося кожні 0,5 секунд необхідно ще додатково поділити тактову частоту на

$$16384 / (256 \cdot 4) = 16$$

Це і буде коефіцієнт передачі передподільника частоти, який потрібно буде встановити за допомогою бітів PS0 – PS2 в регістрі INTCON (не забудьте також підключити до таймеру передподільник частоти за допомогою біту PSA та обрати необхідне джерело тактового сигналу за допомогою біту T0CS).

### 4.3 Алгоритм формування сигналів

Переривання спрацьовує кожні 0,5 Гц, однак у кожному перериванні потрібно формувати різні сигнали. Поточний крок алгоритму потрібно зберігати в одній комірок оперативної пам’яті загального призначення, наприклад, з адресою 0x0C. Число в цій комірці краще зменшувати у кожному перериванні від 4 до 1, а коли воно стане рівним нулю, то записувати туди знову 4. Це простіше за все реалізувати в кінці обробника переривання за допомогою коду, наведеного на рисунку 7.

```
; зміна номеру кроку
decfsz  0x0C, f
retfie

; перезавантаження алгоритму
movlw   d'4'
movwf   0x0C
retfie
```

Рисунок 7 – Рекомендований код для зміни номеру кроку

Значення комірки 0x0C спочатку зменшується на одиницю за допомогою команди **decfsz**, результат зменшення зберігається в оперативній пам’яті. Якщо в результаті зменшення значення комірки 0x0C більше нуля, то відбувається вихід з переривання. Якщо ж після зменшення значення в комірці 0x0C дорівнює нулю, то вихід з переривання не відбувається (замість команди **retfie** виконується команда **nop**). Після цього в комірку 0x0C завантажується число 4 і відбувається вихід з переривання.

Визначити поточний крок алгоритму можна шляхом порівняння змісту комірки 0x0C з константою. Простіше за все це зробити за допомогою команди побітового виключного

АБО комірки пам’яті 0x0C з певною константою (команда **xorwf**) (результат при цьому зберігається в акумуляторі, щоб не псувати зміст оперативної пам’яті). У випадку коли зміст комірки 0x0C буде дорівнювати константі, результат виключного АБО буде дорівнювати нулю, що призведе до підняття прапору **Z** (Zero) в регістрі STATUS. Після цього потрібно зробити перевірку стану прапору Z за допомогою команд **btfsc**/**btfss** (ці команди перевіряють стан біту в регістрі, і якщо він очищений/встановлений, то наступна команда пропускається і замість неї виконується команда **nop**), і якщо він піднятий, то встановити на виводах необхідні сигнали (рис. 8).

```
; крок 1
movlw   d'4'
xorwf   0x0C, w
btfsc   STATUS, Z
bsf/bcf PORTx, RBx
```

Рисунок 8 – Рекомендований код формування сигналів

На кожному кроці потрібно змінювати рівень сигналу лише на одному з виводів мікроконтролера, тому для цього можна скористатися командами **bcf/bsf**. Рекомендована послідовність формування сигналів наведена в табл. 1 (порти та канали потрібно визначити самостійно, відповідно до номеру варіанту).

Таблиця 1 – Рекомендований порядок формування сигналів

| Крок | Значення в комірці 0x0C | Команда | Порт | Канал | Дія (див. рис. 5) |
|---|---|---|---|---|---|
| 1 | d’4’ | bsf | | | Встановлення на виводі B сигналу з рівнем логічної одиниці |
| 2 | d’3’ | bcf | | | Повернення виводу B в початковий стан |
| 3 | d’2’ | bsf | | | Встановлення на виводі A сигналу з рівнем логічної одиниці |
| 4 | d’1’ | bcf | | | Повернення виводу A в початковий стан |

### 4.4 Основний цикл

В цьому застосунку в основному циклі не виконується жодних операцій, тому його можна залишити порожнім (рис. 9).

```
    ; основний цикл
MainLoop:
    goto    MainLoop
```

Рисунок 9 – Рекомендований код основного циклу

### 4.5 Ініціалізація

Під час ініціалізації мікроконтролера потрібно

– налаштувати необхідні канали портів введення-виведення для роботи в режимі виведення;

– встановити початкове значення сигналів на виходах портів (сигнали з рівнем логічного нуля);

– налаштувати таймер (джерело тактового сигналу, передподільник);

– налаштувати переривання (дозволити переривання від таймеру);

– встановити початкове значення в комірку з адресою 0x0C (d’4’);

– увімкнути переривання (встановити біт GIE в регістрі INTCON).

Дозвіл на виникнення переривань (біт GIE в регістрі INTCON) зазвичай встановлюється останнім за допомогою окремої команди **bsf** – перед самим початком основного циклу – це дозволяє гарантувати, що переривання виникне вже після того, як і апаратні, і програмні вузли будуть готові до роботи (рис. 10).

```
Init:
    bsf     STATUS, RP0

    ; налаштування порту
    movlw   b'xxxxxxxx'
    movwf   TRISx

    ; налаштування таймеру
    movlw   b'xxxxxxxx'
    movwf   OPTION_REG

    ; налаштування переривань
    movlw   b'0xxxxxxx'
    movwf   INTCON
```

```
    bcf     STATUS, RP0

    ; ініціалізація порту
    bcf     PORTx, Rxx
    bcf     PORTx, Rxx

    ; ініціалізація алгоритму
    movlw   d'4'
    movwf   0x0C

    ; вмикання переривань
    bsf     INTCON, GIE
```

Рисунок 9 – Рекомендований код ініціалізації мікроконтролера

### 4.6 Рекомендації по створенню програмного забезпечення

1. Створити новий проєкт або скопіювати проєкт з попередніх практичних робіт.
2. Встановити у слові конфігурації, що використовується низькочастотний кварцовий резонатор.
3. Написати тестовий код ініціалізації, в якому налаштувати усі канали усіх портів для роботи в режимі виведення.
4. Написати тестовий основний цикл, в якому послідовно встановлювати на виході усіх каналів сигнал з рівнем логічної одиниці. При цьому слід контролювати до яких каналів підключений двигун годинника. Результати занести в табл. 1.
5. Модифікувати код ініціалізації, в якому залишити в режимі виведення лише ті канали, які задіяні в вашому варіанті апаратної частини.
6. Налаштувати таймер таким чином, щоб його переповнення відбувалося кожні 0,5 с.
6. Налаштувати переривання від таймера.
7. Реалізувати обробник переривань.
8. Реалізувати основний цикл.
9. Перевірити працездатність програмного коду.
10. Завантажити необхідні файли проєкту в системи дистанційної освіти для перевірки викладачем.

## 5. Зміст звіту

Результатом виконання цієї практичної роботи є:
– файл вихідного коду програми «xxx.asm»;
– файл програми у шістнадцятковому форматі «xxx.hex».
Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

# СЕМИСЕГМЕНТНИЙ ІНДИКАТОР

## 1 Мета роботи

Створити застосунок, який буде автоматично формувати на однорозрядному семисегментному світлодіодному індикаторі цифри від «0» до «9».

## 2 Опис апаратної частини

Апаратна частина складається із мікроконтролера DD1, який керує однорозрядним семисегментним світлодіодним індикатором HL1, що містить вісім світлодіодів, з'єднаних за схемою із спільним катодом (рис. 1).

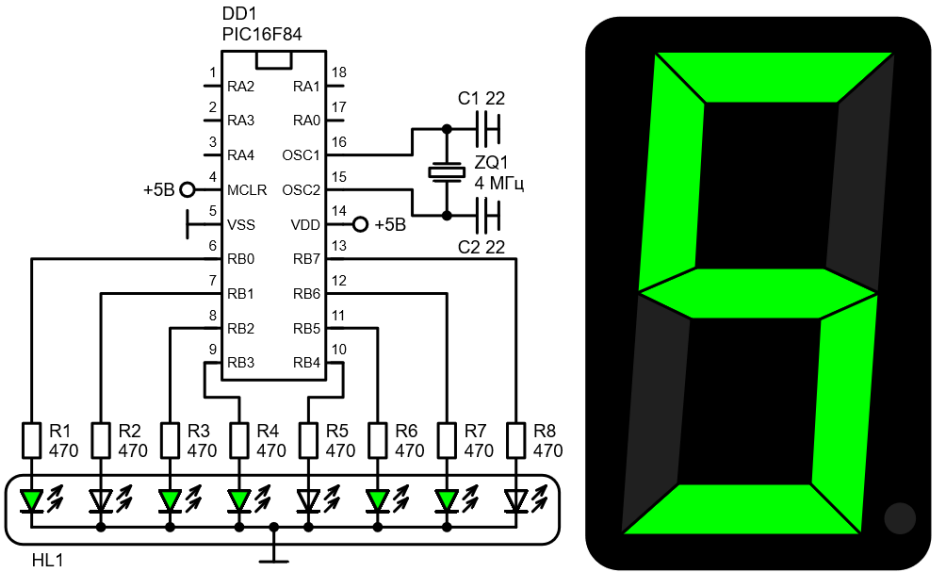


Рисунок 1 – Світловий автомат

Головним елементом автомату є мікроконтролер DD1. Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цьому макеті не використовується, тому він підключений до джерела живлення і на ньому завжди присутній сигнал з рівнем, що дорівнює рівню логічної одиниці. До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 4 МГц. Необхідний режим роботи тактового генератора забезпечується за допомогою двох блокувальний конденсаторів C1 та C2. Вісім світлодіодів індикатора HL1 підключені до виводів RB0 – RB7 через струмообмежувальні резистори R1 – R8.

## 3 Завдання на практичну роботу

Створити застосунок, який буде формувати на семисегментному індикаторі цифри від «0» до «9» (рис. 2). Цифри повинні змінюватися автоматично, у послідовності: «0», «1», «2», «3», «4», «5», «6», «7», «8», «9» з інтервалом 0,5…1 с. Особливістю цієї роботи є те, що підключення сегментів індикатора HL1 (A – G, DP) до виводів RB0 – RB7 мікроконтролера DD1 залежить від номеру варіанта і заздалегідь невідоме. Таким чином, кожен здобувач повинен написати власну програму для мікроконтролера, яка буде правильно працювати тільки на макеті, номер якого буде відповідати номеру варіанта здобувача.

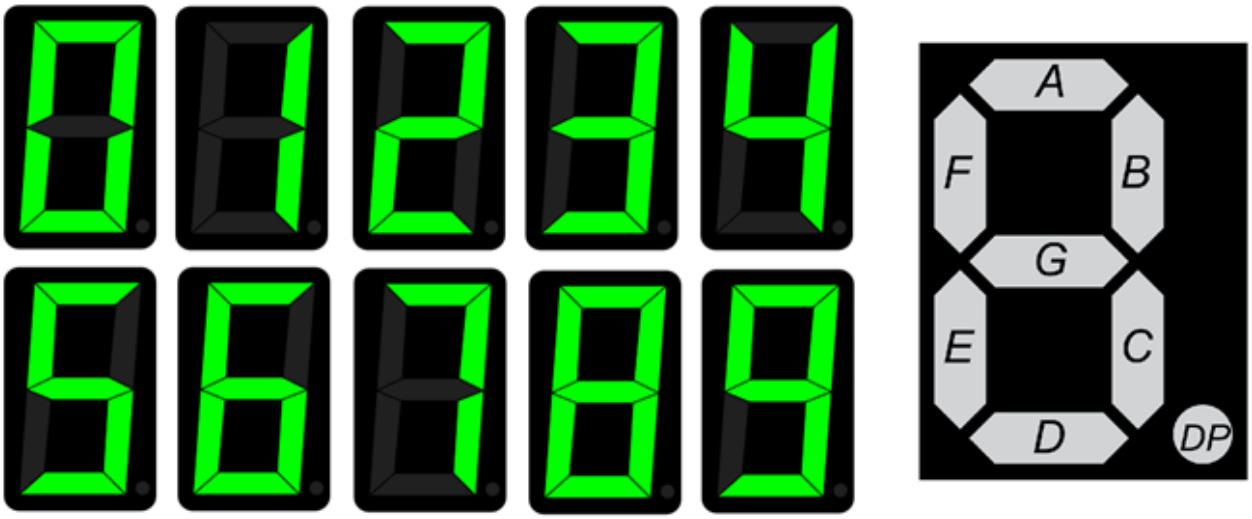


Рисунок 2 – Завдання на практичну роботу

Встановити необхідний номер варіанту лабораторного макету можна через меню Налаштування → Налаштування системи. При цьому відкриється діалогове вікно, у якому можна буде встановити необхідний номер варіанту (рис. 3).

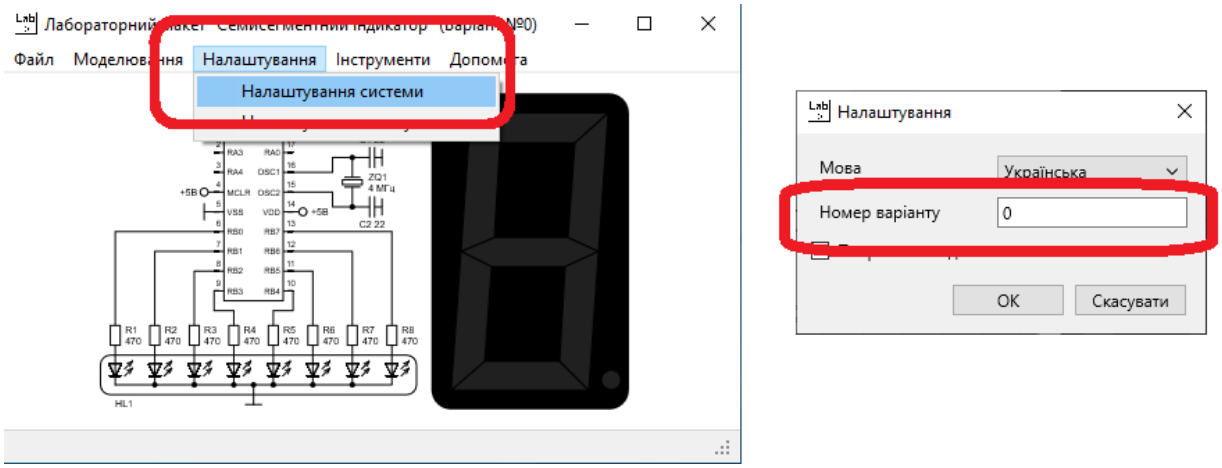


Рисунок 3 – Встановлення номеру варіанту лабораторного макету

Для отримання оцінки «Задовільно» (60 – 73 бали) необхідно виконати практичне завдання в повному обсязі.

Для отримання оцінок «Добре» (74 – 89 балів) та «Відмінно» (90 – 100 балів) потрібно реалізувати більшу кількість символів, наприклад, цифри «0» – «9» та літери «A» – «F», або забезпечити два напрямки лічення, наприклад, спочатку від «0» до «9», а потім від «9» до «0», або створити інший додатковий алгоритм, наприклад, вивести за допомогою цифр та інших символів рік свого народження.

На рівень оцінки впливає якість написання коду, у тому числі форматування та наявність коментарів, тому при неналежному форматуванні або при відсутності коментарів оцінка може бути знижена незалежно від обсягу виконаної роботи.

## 4 Рекомендації по розробці програмного забезпечення

Рекомендований алгоритм програмного забезпечення для виконання поставленої практичної задачі складається із трьох основних частин (рис. 4): блоку ініціалізації, основного циклу та обробника переривань.

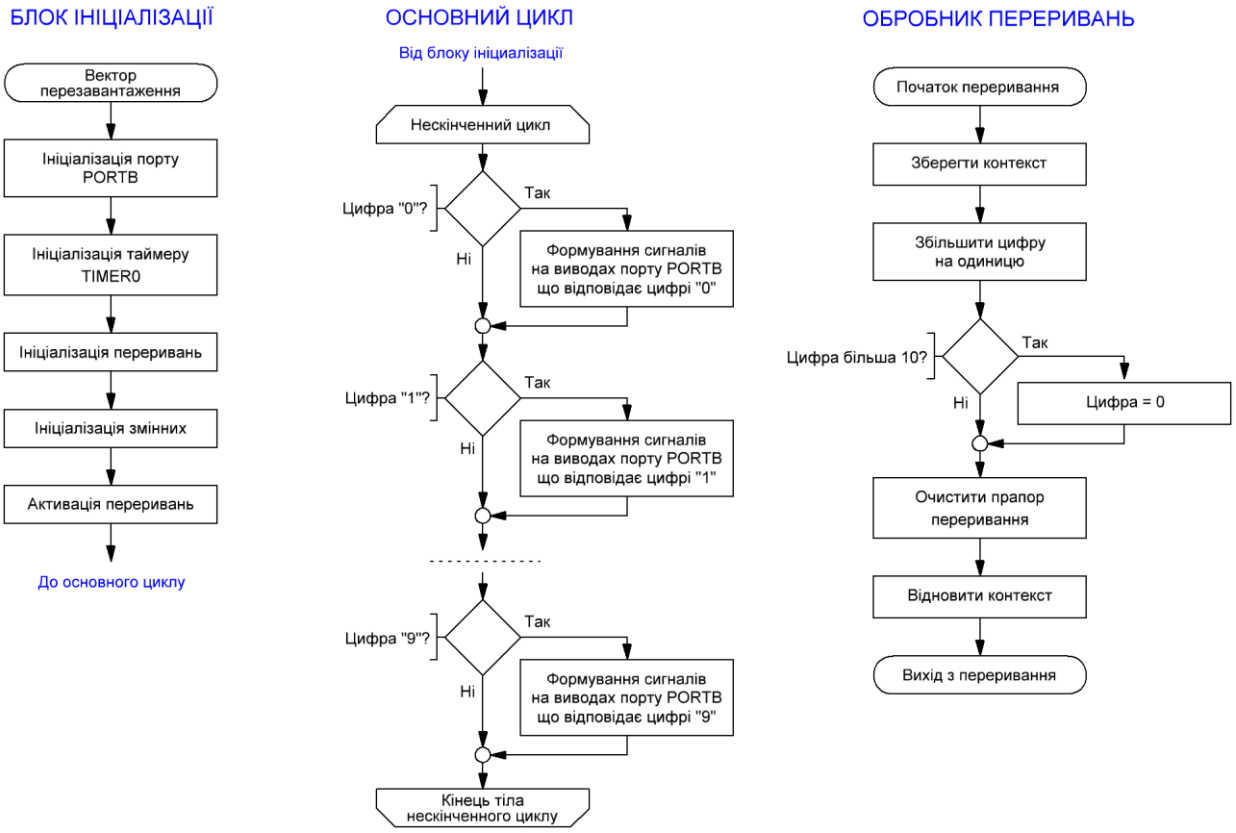


Рисунок 4 – Алгоритм програми

У блоці ініціалізації потрібно налаштувати наступні апаратні периферійні модулі:
– порт вводу-виводу PORTB (регістри TRISB, PORTB);
– таймер Timer0 (регістри OPTION_REG, TMR0);
– механізм переривань (Interrupts) (регістр INTCON).

Крім того, слід створити щонайменше одну змінну (наприклад, Numeric), у якій буде зберігатися поточне значення цифри, що відображається, та ініціалізувати її, наприклад, встановивши її значення рівним 0. Для визначення змінної краще використовувати конструкцію **cblock** (Лістинг 1).

```
#include <p16F84A.inc>

__CONFIG _XT_OSC & _WDT_OFF & _PWRTE_OFF &

cblock  0x0C
    Numeric
endc
```

Лістинг 1 – Рекомендований код для визначення імен змінних

Ініціалізацію модулів мікроконтролера можна виконувати у довільному порядку. Останньою командою блоку ініціалізації повинно бути активізація переривань.

Рекомендований код блоку переривань наведений у лістингу (Лістинг 2).

```
; ----------------------------------------
; ІНІЦІАЛІЗАЦІЯ
; ----------------------------------------

; налаштування каналів (вхід/вихід)
; порту PORTB
bsf     STATUS, RP0
movlw   b'xxxxxxxx'
movwf   TRISB
bcf     STATUS, RP0

; встановлення вихідних сигналів
; на виводах порту PORTB
movlw   b'xxxxxxxx'
movwf   PORTB

; налаштування таймеру
bsf     STATUS, RP0
movlw   b'xxxxxxxx'
movwf   OPTION_REG
bcf     STATUS, RP0

; налаштування переривань
movlw   b'xxxxxxxx'
movwf   INTCON

; ініціалізація змінних
movlw   d'0'
movwf   Numeric

; активація переривань
bsf     INTCON, GIE
```

Лістинг 2 – Рекомендований код блоку ініціалізації

В основному циклі виконується послідовна перевірка значень змінної Numeric. І якщо її значення дорівнюють цифрам 0…9 то на виводах порту PORTB формується відповідна комбінація сигналів (Лістинг 3). Зверніть увагу, що який канал порту PORTB підключений до якого сегменту індикатора (A – G, DP), доведеться з'ясовувати експериментально. Для цього можна на перших порах написати лише код ініціалізації, я якому послідовно засвічувати по одному світлодіоду, і з'ясовувати, який сегмент буде світитися. Результати експерименту краще занести до таблиці 1.

Таблиця 1 – Таблиця для визначення електричної схеми (кольором виділені комірки, що заповнюються самостійно)

| Тестова константа, що записується у регістр PORTB | Активний канал порту | Сегмент, який світиться (A – G або DP) |
|---|---|---|
| b'00000001' | RB0 | A |
| b'00000010' | RB1 | B |
| b'00000100' | RB2 | C |
| b'00001000' | RB3 | D |
| b'00010000' | RB4 | E |
| b'00100000' | RB5 | F |
| b'01000000' | RB6 | G |
| b'10000000' | RB7 | DP |

```
        ; -------------------------
        ; ОСНОВНИЙ ЦИКЛ
        ; -------------------------
MainLoop:
        ; перевірка, чи дорівнює
        ; число 0
        movfw   Numeric
        xorlw   d'0'
        btfss   STATUS, Z
        goto    Check_01

        ; вивід на порт B сигналів,
        ; що відповідають цифрі 0
        movlw   b'xxxxxxxx'
        movwf   PORTB

Check_01:
        ; перевірка, чи дорівнює
        ; число 1
        movfw   Numeric
        xorlw   d'1'
        btfss   STATUS, Z
        goto    Check_02

        ; вивід на порт B сигналів,
        ; що відповідають цифрі 1
        movlw   b'xxxxxxxx'
        movwf   PORTB

Check_02:
        ; перевірка цифр 2...8

        ; ---------------------

Check_09:
        ; перевірка, чи дорівнює
        ; число 9
        movfw   Numeric
        xorlw   d'9'
        btfss   STATUS, Z
        goto    EndCheck

        ; вивід на порт B сигналів,
        ; що відповідають цифрі 1
        movlw   b'xxxxxxxx'
        movwf   PORTB

EndCheck:
        goto    MainLoop
```

Лістинг 3 – Рекомендований код основного циклу

Після цього, слід визначити, яку комбінацію символів потрібно сформувати на виводах мікроконтролера, щоб світилися потрібні сегменти (табл. 2).

Таблиця 2 – Константи, що будуть формувати сигнали керування індикатором (кольором виділені комірки, що заповнюються самостійно)

| Цифра | Активні сегменти | Сегмент, який світиться (A – G або DP) |
|---|---|---|
| 0 | A, B, C, D, E, F | b'00111111' |
| 1 | A, B | b'00000011' |
| 2 | A, B, D, E, G | b'01011011' |
| 3 | A, B, C, D, G | b'01001111' |
| 4 | B, C, F, G | b'01100110' |
| 5 | A, C, D, F, G | b'01101101' |
| 6 | A, C, D, E, F, G | b'01111101' |
| 7 | A, B, C | b'00000111' |
| 8 | A, B, C, D, E, F, G | b'01111111' |
| 9 | A, B, C, D, F, G | b'01101111' |

Обробник переривань містить код, що реалізує зміну цифри через певний проміжок часу. При зміні цифри слід слідкувати, щоб її значення знаходились у межах 0…9. Рекомендований код, приведений на лістингу (Лістинг 4)

Зверніть увагу, що у обробнику переривань змінюється значення акумулятора і, відповідно, регістра STATUS, тому для того, щоб код в основному циклі правильно працював, одразу після входу в переривання потрібно зберегти контекст (зміст акумулятора і регістру STATUS), а потім – перед виходом із переривання, його відновити. Зробити це краще за все, відповідно рекомендаціям, наведеним у технічній документації на мікроконтролер (Лістинг 5).

```
; збільшення значення на 1
incf    Numeric, f

; перевірка на переповнення
movfw   Numeric
sublw   MAX_NUMERIC, w
btfss   STATUS, C
clrf    Numeric
```

Лістинг 4 – Рекомендований код зміни цифри

**EXAMPLE 6-1: SAVING STATUS AND W REGISTERS IN RAM**

```
PUSH    MOVWF   W_TEMP              ; Copy W to TEMP register,
        SWAPF   STATUS,     W       ; Swap status to be saved into W
        MOVWF   STATUS_TEMP         ; Save status to STATUS_TEMP register
ISR     :                           :
        :                           ; Interrupt Service Routine
        :                           ; should configure Bank as required
        :                           ;
POP     SWAPF   STATUS_TEMP,W       ; Swap nibbles in STATUS_TEMP register
                                    ; and place result into W
        MOVWF   STATUS              ; Move W into STATUS register
                                    ; (sets bank to original state)
        SWAPF   W_TEMP,     F       ; Swap nibbles in W_TEMP and place result in W_TEMP
        SWAPF   W_TEMP,     W       ; Swap nibbles in W_TEMP and place result into W
```

Лістинг 5 – Рекомендований код збереження та відновлення контексту

### 5. Зміст звіту

Результатом виконання цієї практичної роботи є:
– файл вихідного коду програми «xxx.asm»;
– файл програми у шістнадцятковому форматі «xxx.hex».

Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

# 5. КОНТРОЛЬ КНОПОК

## 1 Мета роботи

Створити застосунок, який буде формувати на однорозрядному семисегментному світлодіодному індикаторі цифри від «0» до «9», які повинні змінюватися користувачем при натисканні на кнопки SW1 та SW2.

## 2 Опис апаратної частини

Апаратна частина складається із мікроконтролера DD1, який керує однорозрядним семисегментним світлодіодним індикатором HL1, що містить вісім світлодіодів, з'єднаних за схемою із спільним анодом (рис. 1).

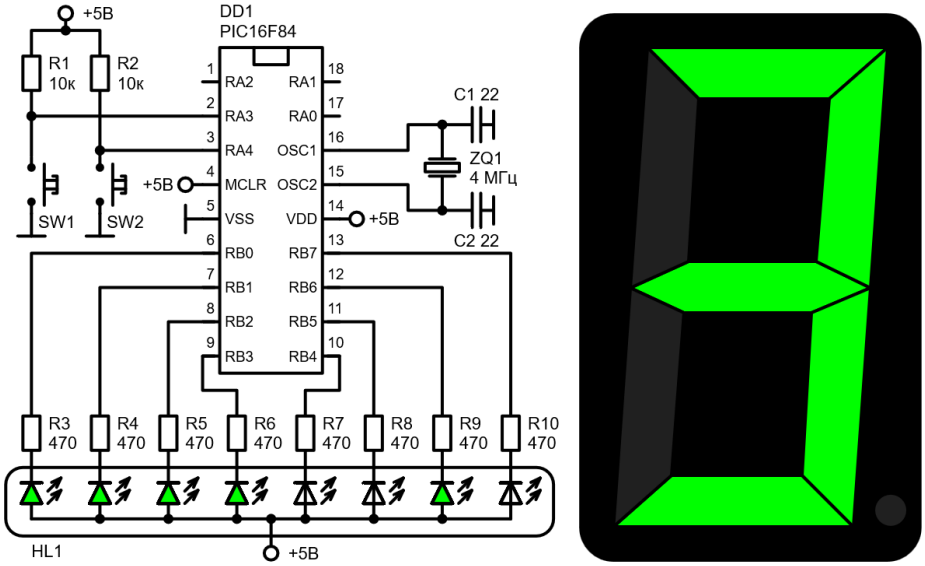


Рисунок 1 – Електрична схема макету

Головним елементом автомату є мікроконтролер DD1. Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цьому макеті не використовується, тому він підключений до джерела живлення і на ньому завжди присутній сигнал з рівнем, що дорівнює рівню логічної одиниці. До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 4 МГц. Необхідний режим роботи тактового генератора забезпечується за допомогою двох блокувальний конденсаторів C1 та C2. Вісім світлодіодів індикатора HL1 підключені до виводів RB0 – RB7 через струмообмежувальні резистори R3 – R10.

Для зміни цифри призначені дві тактові кнопки SW1 та SW2 із нормально розімкненими контактами, які підключені до виводів RA3 та RA4. Для забезпечення визначеного рівня сигналу на виводах RA3 та RA4 при ненатиснених кнопках (коли вони не проводять електричний струм) призначені два підтягувальні резистори R1 та R2. Відповідно до обраної схеми підключення кнопок до мікроконтролера, при ненатиснених кнопках SW1 та SW2 на виводах RA3 та RA4 присутній сигнал із напругою +5 В, який інтерпретується мікроконтролером як сигнал із рівнем логічної одиниці. При натисканні на кнопки SW1 та SW2 вони електрично з'єднують відповідні виводи мікроконтролера із спільним проводом, що призводить до встановлення на них напруги 0 В, яка інтерпретується мікроконтролером як сигнал із рівнем логічного нуля.

### 3 Завдання на практичну роботу

Створити застосунок, який буде формувати на семисегментному індикаторі цифри від «0» до «9» (рис. 2). Цифри повинні змінюватися у послідовності: «0», «1», «2», «3», «4», «5», «6», «7», «8», «9» при натисканні на кнопки SW1 та SW2, наприклад, при натисканні на кнопку SW1 цифра збільшується на одиницю, а при натисканні на кнопку SW2 – зменшується на одиницю. Якщо користувач натисне на кнопку «Зменшення» коли цифра дорівнює 0, то вона повинна стати рівною 9, аналогічно, якщо користувач натисне на кнопку «Збільшення», коли цифра дорівнює 9, то вона повинна стати рівною 0.

Особливістю цієї роботи є те, що підключення сегментів індикатора HL1 (A – G, DP) до виводів RB0 – RB7 мікроконтролера DD1 залежить від номеру варіанта і заздалегідь невідоме. Таким чином, кожен здобувач повинен написати власну програму для мікроконтролера, яка буде правильно працювати тільки на макеті, номер якого буде відповідати номеру варіанта здобувача.

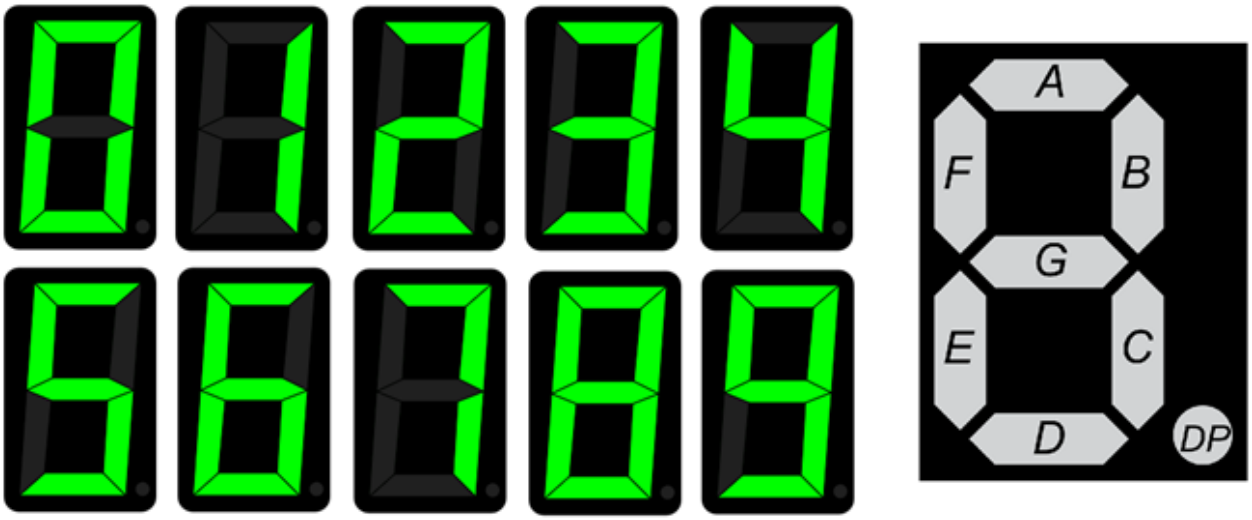


Рисунок 2 – Завдання на практичну роботу

Встановити необхідний номер варіанту лабораторного макету можна через меню Налаштування → Налаштування системи. При цьому відкриється діалогове вікно, у якому можна буде встановити необхідний номер варіанту (рис. 3).

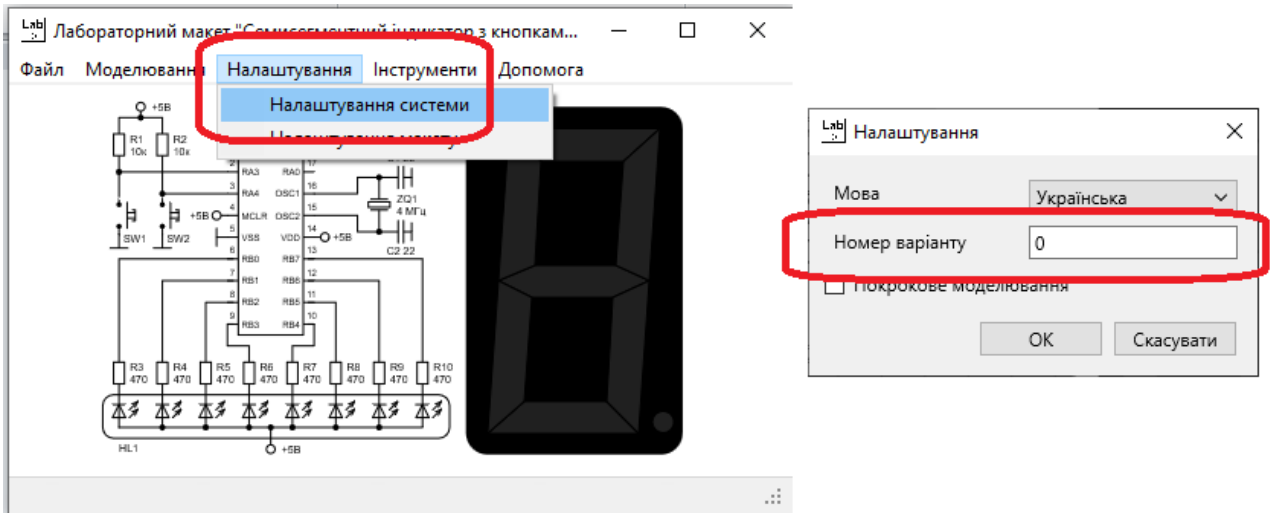


Рисунок 3 – Встановлення номеру варіанту лабораторного макету

Для отримання оцінки «Задовільно» (60 – 73 бали) необхідно виконати практичне завдання в повному обсязі.

Для отримання оцінок «Добре» (74 – 89 балів) та «Відмінно» (90 – 100 балів) потрібно реалізувати більш складний алгоритм поведінки. Наприклад, при короткому натисканні на кнопку цифра спочатку збільшується (зменшується) на одиницю, а при тривалому натисканні (якщо кнопка зажата користувачем, який її не відпускає, цифра почне швидко та автоматично збільшуватися (зменшуватися).

На рівень оцінки впливає якість написання коду, у тому числі форматування та наявність коментарів, тому при неналежному форматуванні або при відсутності коментарів оцінка може бути знижена незалежно від обсягу виконаної роботи.

## 4 Рекомендації по розробці програмного забезпечення

Рекомендований алгоритм програмного забезпечення для виконання поставленої практичної задачі складається із трьох основних частин (рис. 4): блоку ініціалізації, основного циклу та обробника переривань.

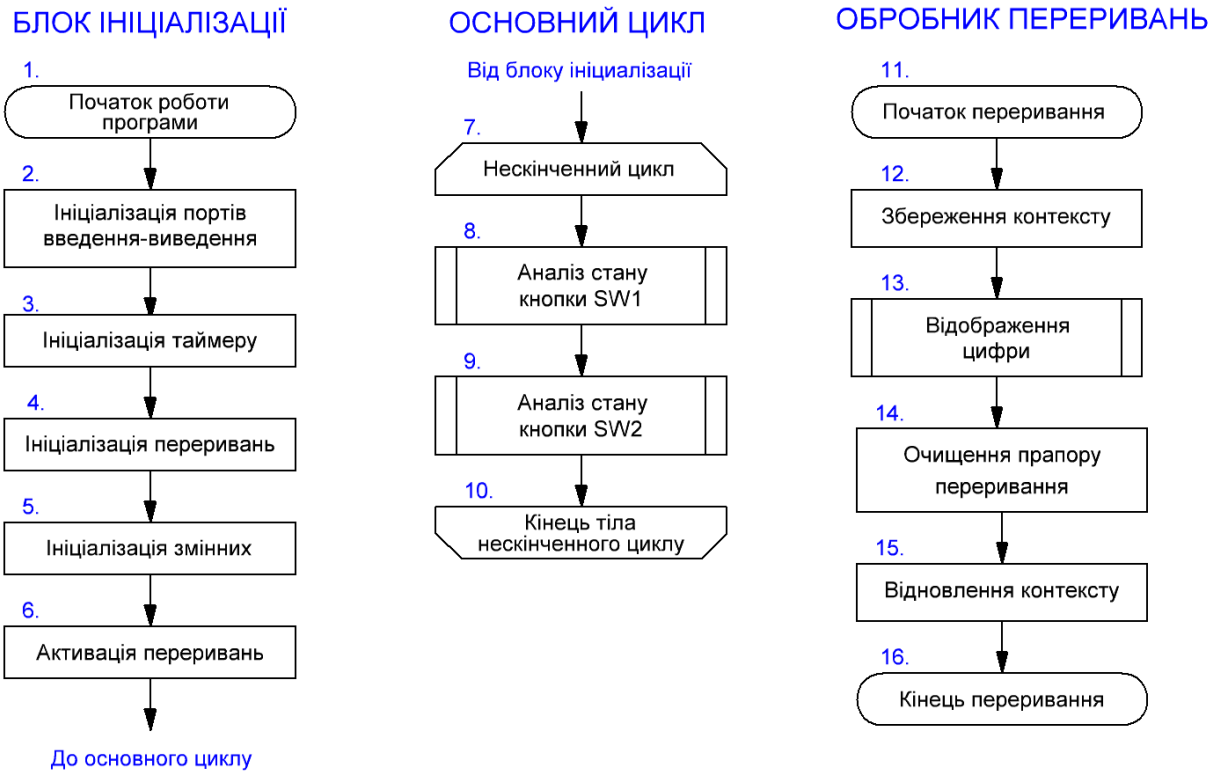


Рисунок 4 – Алгоритм програми

У блоці ініціалізації потрібно налаштувати наступні апаратні периферійні модулі:

– порти введення-виведення PORTA та PORTB (регістри TRISA, TRISB, PORTA та PORTB) (блок 2);

– таймер Timer0 (регістри OPTION_REG, TMR0) (блок 3);

– механізм переривань (Interrupts) (регістр INTCON) (блок 4).

В основному циклі виконується перевірка станів кнопок SW1 та SW2 (блоки 8, 9) шляхом контролю стану каналів RA3 та RA4 порту введення-виведення PORTA. У перериванні виконується відображення поточного значення цифри шляхом формування відповідних сигналів на виводах RB0 – RB7 порту введення-виведення PORTB.

У даному лабораторному макеті використовується два порти: PORTA та PORTB. Усі канали порту PORTB працюють на вихід, тому на етапі ініціалізації регістр TRISB достатньо просто очистити. Зверніть увагу, що у схемі використовується світлодіодний індикатор зі

спільним анодом, тому для запобігання небажаного «блимання» сегментів індикатора, після налаштування порту слід встановити на всіх каналах рівень, що відповідає рівню логічної одиниці (Лістинг 1).

```
Init:
    ;налаштування PORTB
    bsf     STATUS, RP0
    clrf    TRISB
    bcf     STATUS, RP0

    ; ініціалізація PORTB
    movlw   b'11111111'
    movwf   PORTB

    ;налаштування PORTA
    bsf     STATUS, RP0
    movlw   b'00011000'
    movwf   TRISA
    bcf     STATUS, RP0

    ; ініціалізація PORTA
    movlw   b'00011000'
    movwf   PORTA
```

Лістинг 1 – Рекомендований код ініціалізації портів введення-виведення

У порту PORTA канали RA3 та RA4 працюють у режимі входу, а канали RA0 – RA2 не використовуються. Для зменшення енергоспоживання мікроконтролера та підвищення стабільності роботи програми незадіяні канали порту рекомендується налаштувати у режим виходу. При цьому на них може бути будь-який рівень сигналу (у прикладі – сигнал, що має рівень логічного нуля).

Зверніть увагу, що канали RA5 – RA7 у мікроконтролері PIC16F84A фізично не реалізовані, тому у біти 5 – 7 у регістрах TRISA та PORTA можна записувати будь-яке значення, при цьому вони завжди зчитуються як 0.

Рекомендований алгоритм контролю стану кнопок (блоки 8 та 9 алгоритму, зображеного на рис. 4) наведено на рис. 5.

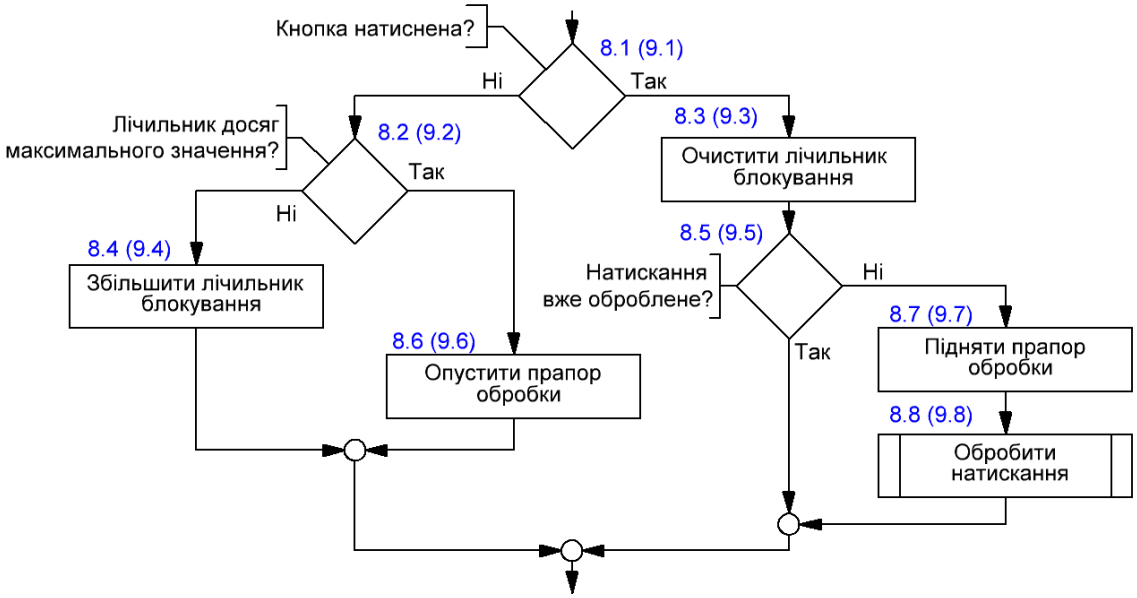


Рисунок 5 – Рекомендований алгоритм обробки стану кнопок (блоки 8 та 9 рис. 4)

Якщо програма виявила факт натискання на кнопку (наявність очищеного біту у відповідному каналі регістру PORTA) (блок 8.1), то спершу очищується лічильник блокування (у прикладі – лічильники KeyUpLockCounter та KeyDownLockCounter – по одному для кожної кнопки KeyUp та KeyDown) (блок 8.3) (Лістинг 2), а потім перевіряється стан прапору обробки (у прикладі – прапори KeyUpHandleFlag та KeyDownHandleFlag) (блок 8.5). Якщо прапор опущений (ще не було обробки), тоді цей прапор піднімається (блок 8.7) і виконується код обробника натискання (блок 8.8). Якщо ж прапор піднятий (вже

була обробка цього натискання), тоді нічого не відбувається. Таким чином, для того, щоб обробити натискання двох кнопок, потрібно три змінні у оперативній пам'яті: одна змінна (Flags), що місить прапори обробки KeyUpHandleFlag та KeyDownHandleFlag та дві змінні лічильників блокування (KeyUpLockCounter та KeyDownLockCounter). На етапі ініціалізації ці змінні повинні бути очищені (Лістинг 3).

```
    ; основний цикл
MainLoop:
    ; обробник натискання
    ; на кнопку "Вперед"
    btfss   PORTA, 3
    goto    KeyUpPress

    ; кнопка "Вперед" не натиснена

    ; збільшення лічильника блокування
    incfsz  KeyUpLockCounter, w
    incf    KeyUpLockCounter, f

    ; перевірка значення лічильника
    ; і очищення прапора обробки
    movfw   KeyUpLockCounter
    xorlw   MaxCounterValue
    btfsc   STATUS, Z
    bcf     KeyUpHandleFlag

    goto    CheckKeyDown

    ; кнопка "Вперед" натиснена

KeyUpPress:
    ; очищення лічильника блокування
    clrf    KeyUpLockCounter

    ; перевірка прапора обробки
    btfsc   KeyUpHandleFlag
    goto    CheckKeyDown

    ; встановлення прапора обробки
    bsf     KeyUpHandleFlag

    ; обробка натиснення
    ; на кнопку "Вперед"
    ----------------------------------
```

```
CheckKeyDown:
    ; обробник натискання
    ; на кнопку "Назад"
    btfss   PORTA, 4
    goto    KeyDownPress

    ; кнопка "Назад" не натиснена

    ; збільшення лічильника блокування
    incfsz  KeyDownLockCounter, w
    incf    KeyDownLockCounter, f

    ; перевірка значення лічильника
    ; і очищення прапора обробки
    movfw   KeyDownLockCounter
    xorlw   MaxCounterValue
    btfsc   STATUS, Z
    bcf     KeyDownHandleFlag

    goto    MainLoop

    ; кнопка "Назад" натиснена

KeyDownPress:
    ; очищення лічильника блокування
    clrf    KeyDownLockCounter

    ; перевірка прапора обробки
    btfsc   KeyDownHandleFlag
    goto    ShowSymbol

    ; встановлення прапора обробки
    bsf     KeyDownHandleFlag

    ; обробка натиснення
    ; на кнопку "Назад"
    ----------------------------------
```

Лістинг 2 – Рекомендований код контролю кнопок

```
; змінні і константи
cblock  0x0C
    Flags
    KeyUpLockCounter
    KeyDownLockCounter
endc

#define KeyUpHandleFlag     Flags, 0
#define KeyDownHandleFlag   Flags, 1
#define MaxCounterValue     0xFF
```

```
;ініціалізація змінних

; прапори
clrf    Flags

; лічильники блокування
clrf    KeyUpLockCounter
clrf    KeyDownLockCounter
```

Лістинг 3 – Змінні, константи та блок ініціалізації, пов'язані із кнопками

Рекомендований алгоритм обробників натискання на кнопки (блоки 8.8 та 9.8 алгоритму, наведеного на рис. 5) наведено на рис. 6. В обробниках натискання спочатку значення цифри збільшується (зменшується) на одиницю (блоки 8.8.1 або 9.8.1), а потім перевіряється знаходження нового стану в межах допустимих значень (блоки 8.8.2 та 9.8.2). І якщо це значення виходить за межі, тоді цифра встановлюється рівною 0 або 9 (блоки 8.8.3 та 9.8.3).

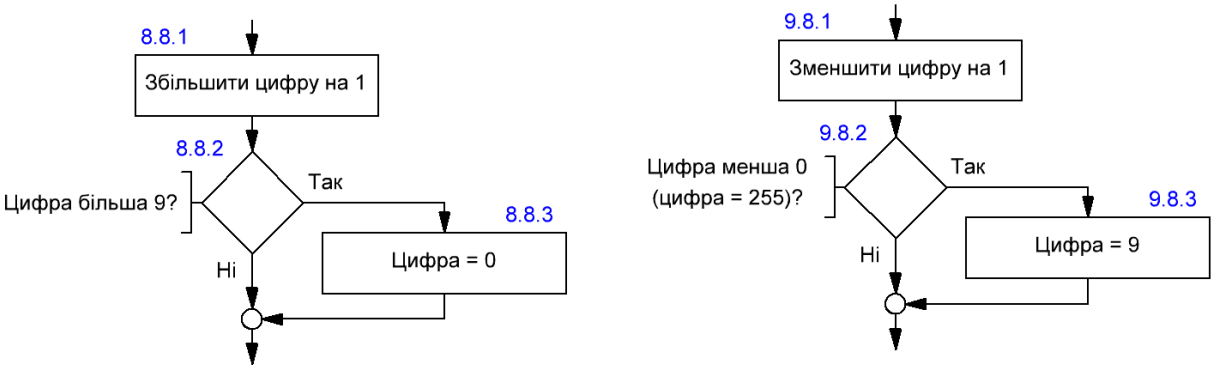


Рисунок 6 – Рекомендований алгоритм обробників натискання на кнопку (блоки 8.8 та 9.8 алгоритму рис. 5)

Формування сигналів на виводах індикатора відбувається у перериванні. Для відображення символів можна скористатися алгоритмом, що використовувався у практичній роботі 4 (рис. 7, лістинг 4).

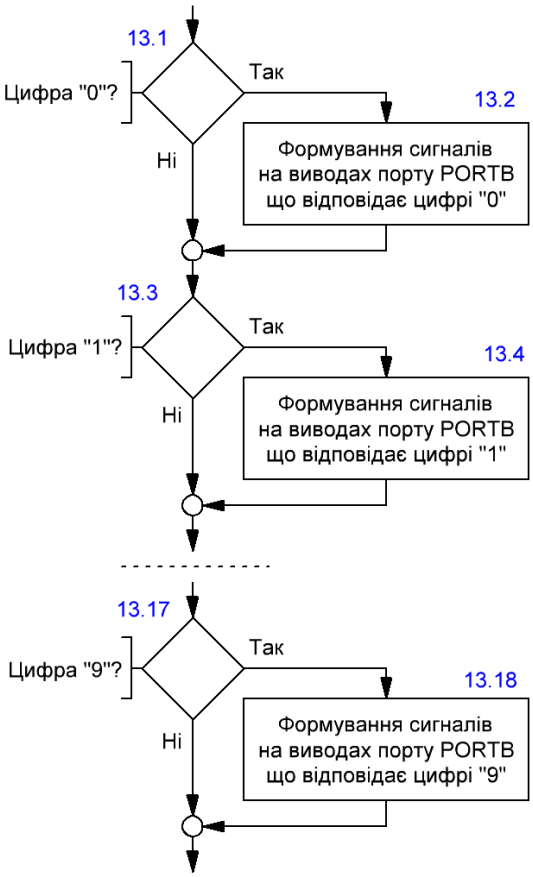


Рисунок 7 – Рекомендований алгоритм формування сигналів індикації (блок 13 алгоритму, наведеного на рис. 4)

```
        ; перевірка, чи дорівнює
        ; число 0
        movfw   Numeric
        xorlw   d'0'
        btfss   STATUS, Z
        goto    Check_01

        ; вивід на порт B сигналів,
        ; що відповідають цифрі 0
        movlw   b'xxxxxxxx'
        movwf   PORTB

Check_01:
        ; перевірка, чи дорівнює
        ; число 1
        movfw   Numeric
        xorlw   d'1'
        btfss   STATUS, Z
        goto    Check_02

        ; вивід на порт B сигналів,
        ; що відповідають цифрі 1
        movlw   b'xxxxxxxx'
        movwf   PORTB

Check_02:
        ; перевірка цифр 2...8
        ; ---------------------
Check_09:
        ; перевірка, чи дорівнює
        ; число 9
        movfw   Numeric
        xorlw   d'9'
        btfss   STATUS, Z
        goto    EndCheck

        ; вивід на порт B сигналів,
        ; що відповідають цифрі 1
        movlw   b'xxxxxxxx'
        movwf   PORTB

EndCheck:
```

Лістинг 4 – Рекомендований відображення значення цифри

Зверніть увагу, що який канал порту PORTB підключений до якого сегменту індикатора (A – G, DP), доведеться з'ясовувати експериментально. Для цього можна на перших порах написати лише код ініціалізації, я якому послідовно засвічувати по одному світлодіоду, і з'ясовувати, який сегмент буде світитися. Результати експерименту краще занести до таблиці 1.

Таблиця 1 – Таблиця для визначення електричної схеми (кольором виділені комірки, що заповнюються самостійно)

| Тестова константа, що записується у регістр PORTB | Активний канал порту | Сегмент, який світиться (A – G або DP) |
|---|---|---|
| b'00000001' | RB0 | A |
| b'00000010' | RB1 | B |
| b'00000100' | RB2 | C |
| b'00001000' | RB3 | D |
| b'00010000' | RB4 | E |
| b'00100000' | RB5 | F |
| b'01000000' | RB6 | G |
| b'10000000' | RB7 | DP |

Лістинг 6 – Рекомендований код основного циклу

Після цього, слід визначити, яку комбінацію символів потрібно сформувати на виводах мікроконтролера, щоб світилися потрібні сегменти (табл. 2).

Не слід забувати, що у обробнику переривань змінюється значення акумулятора і, відповідно, регістра STATUS, тому для того, щоб код в основному циклі правильно працював, одразу після входу в переривання потрібно зберегти контекст (зміст акумулятора і регістру STATUS), а потім – перед виходом із переривання, його відновити. Зробити це краще за все, відповідно рекомендаціям, наведеним у технічній документації на мікроконтролер (Лістинг 5).

Таблиця 2 – Константи, що будуть формувати сигнали керування індикатором (кольором виділені комірки, що заповнюються самостійно)

| Цифра | Активні сегменти | Сегмент, який світиться (A – G або DP) |
|---|---|---|
| 0 | A, B, C, D, E, F | b'00111111' |
| 1 | A, B | b'00000011' |
| 2 | A, B, D, E, G | b'01011011' |
| 3 | A, B, C, D, G | b'01001111' |
| 4 | B, C, F, G | b'01100110' |
| 5 | A, C, D, F, G | b'01101101' |
| 6 | A, C, D, E, F, G | b'01111101' |
| 7 | A, B, C | b'00000111' |
| 8 | A, B, C, D, E, F, G | b'01111111' |
| 9 | A, B, C, D, F, G | b'01101111' |

**EXAMPLE 6-1: SAVING STATUS AND W REGISTERS IN RAM**

```
PUSH    MOVWF   W_TEMP              ; Copy W to TEMP register,
        SWAPF   STATUS,     W       ; Swap status to be saved into W
        MOVWF   STATUS_TEMP         ; Save status to STATUS_TEMP register
ISR     :                           :
        :                           ; Interrupt Service Routine
        :                           ; should configure Bank as required
        :                           ;
POP     SWAPF   STATUS_TEMP,W       ; Swap nibbles in STATUS_TEMP register
                                    ; and place result into W
        MOVWF   STATUS              ; Move W into STATUS register
                                    ; (sets bank to original state)
        SWAPF   W_TEMP,     F       ; Swap nibbles in W_TEMP and place result in W_TEMP
        SWAPF   W_TEMP,     W       ; Swap nibbles in W_TEMP and place result into W
```

Лістинг 7 – Рекомендований код збереження та відновлення контексту

**5. Зміст звіту**

Результатом виконання цієї практичної роботи є:
– файл вихідного коду програми «xxx.asm»;
– файл програми у шістнадцятковому форматі «xxx.hex».

Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

# 6. СЕКУНДОМІР

## 1 Мета роботи

Створити пристрій для вимірювання часу від 0 до 999 секунд з точністю до однієї секунди.

## 2 Апаратна частина для тестування програмного забезпечення

### 2.1 Електрична схема

Апаратна частина для тестування програмного забезпечення складається із мікроконтролера DD1, який керує трирозрядним семисегментним світлодіодним індикатором HL1, що містить три секції (розряди), кожна з яких містить вісім світлодіодів, з'єднаних за схемою із спільним анодом (рис. 1).

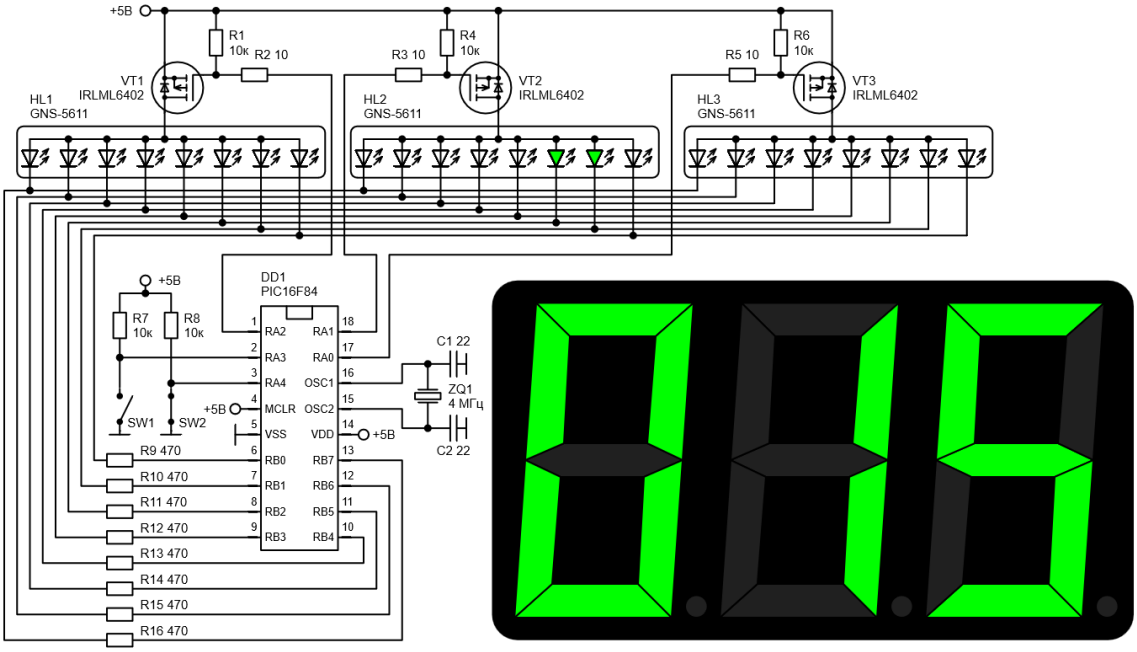


Рисунок 1 – Схема для тестування програмного забезпечення

Головним елементом апаратної частини є мікроконтролер DD1. Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цій схемі не використовується, тому він підключений до джерела живлення, і на ньому завжди присутній сигнал з рівнем логічної одиниці. До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 4 МГц. Необхідний режим роботи кварцового резонатора та тактового генератора забезпечується за допомогою двох керамічних конденсаторів C1 та C2 ємністю 22 пФ.

Індикатор HL1 містить три розряди, кожен з яких містить по вісім світлодіодних сегментів, сім з яких утворюють цифру «8», а восьмий – десяткову точку, розташовану в нижньому правому куті індикаторної області розряду. Аноди усіх світлодіодів в межах розряду об'єднані між собою. Кожен розряд має окремий вивід анодів, що дозволяє контролювати світінням кожного окремого розряду та використовувати для керування індикатором HL1 принцип динамічної індикації.

Струм кожного з восьми сегментів обмежується струмообмежувальними резисторами R9 – R18 з опором 470 Ом, які підключені до порту PORTB мікроконтролера DD1. Оскільки

для світіння світлодіодного індикатора необхідно, щоб його катод по відношенню до аноду мав менший потенціал, то для засвічення сегментів (світлодіодів) на виводах порту PORTB потрібно формувати сигнали і з рівнем логічного нуля.

Оскільки загальний струм усіх сегментів може перевищувати максимально допустимий струм каналу порту введення-виведення мікроконтролера, то вони підключені до нього через підсилювачі, реалізовані на транзисторах VT1 – VT2. Підсилювачі сигналів мікроконтролера побудовані на основі польових транзисторів з ізольованим затвором з каналом p-типу (p-channel Metal-Oxide-Semiconductor Field-Effect Transistor, P-MOSFET) IRLML6402 виробництва компанії International Rectifier. Особливістю цих транзисторів є знижена порогова напруга між затвором та витоком (Threshold Voltage), що дозволяє використовувати для керування ними сигнали з логічними рівнями.

Підсилювачі сигналів на транзисторах VT1 – VT2 побудовані за схемами із загальними витоками, причому витоки усіх транзисторів з'єднані із шиною живлення +5 В. Для переводу каналів транзисторів типу P-MOSFET у провідний стан необхідно щоб потенціал їх затвору був менший за потенціал витоку, тому для того, щоб транзистори VT1 – VT2 почали пропускати струм між витоком та стоком, необхідно сформувати на виводах RA0 – RA2 мікроконтролера DD1 сигнали із рівнем логічного нуля. Наявність на виводі мікроконтролера сигналу із рівнем логічного нуля призведе до утворення негативної різниці потенціалів між затвором та витоком, що, у свою чергу, призведе до зменшення опору каналу між стоком та витоком транзисторів до величини приблизно 65 мОм. При формуванні на виводі мікроконтролера сигналу із рівнем логічної одиниці потенціал затвора стане рівним потенціалу витоку, провідний канал між стоком та витоком зникне, що призведе до розімкнення кола протікання струмів через усі сегменти розряду.

Резистори R1, R4 та R6 з опором 10 кОм призначені для утримання транзисторів VT1 – VT2 в непровідному стані, коли порти RA0 – RA2 мікроконтролера знаходяться у режимі введення, наприклад, після перезавантаження мікроконтролера. Резистори R2, R3 та R5 з опором 10 Ом обмежують струм у колі затворів транзисторів VT1 – VT2, які виникають під час перезаряду їх вхідної ємності (ємності між затвором та витоком).

Для керування схемою призначені два перемикача SW1 та SW2, які підключені до виводів RA3 та RA4. Для забезпечення визначеного рівня сигналу на виводах RA3 та RA4 при розімкнених перемикачах – коли вони не проводять електричний струм – призначені два підтягувальні резистори R7 та R8 з опором 10 кОм. Відповідно до обраної схеми підключення кнопок до мікроконтролера, при розімкнених перемикачах SW1 та SW2 на виводах RA3 та RA4 присутній сигнал із напругою +5 В, який інтерпретується мікроконтролером як сигнал із рівнем логічної одиниці. При замиканні перемикачів SW1 та SW2 вони електрично з'єднують відповідні виводи мікроконтролера із спільним проводом, що призводить до встановлення на них напруги 0 В, яка інтерпретується мікроконтролером як сигнал із рівнем логічного нуля.

### 2.2 Принцип динамічної індикації

При динамічній індикації розряди багаторозрядного індикатора відображаються послідовно. Який розряд буде активним у даний момент визначається сигналами на виходах порту PORTA. У даній схемі, якщо сигнал на виводі RA0 буде мати рівень логічного нуля, то відкриється транзистор VT3 і на аноди світлодіодів третього розряду індикатора HL1 буде подана напруга +5 В, отже у даний момент буде відображатися молодший (нульовий) розряд індикатора, а усі інші розряди світитися не будуть (Рис. 2). Для відображення розряду 1 потрібно подати сформувати низький рівень сигналу на виводі RA1, що призведе до відкриття транзистора VT2, а для засвічення старшого потрібно сформувати низький рівень сигналу на виводі RA2, що призведе до відкриття транзистора VT1.

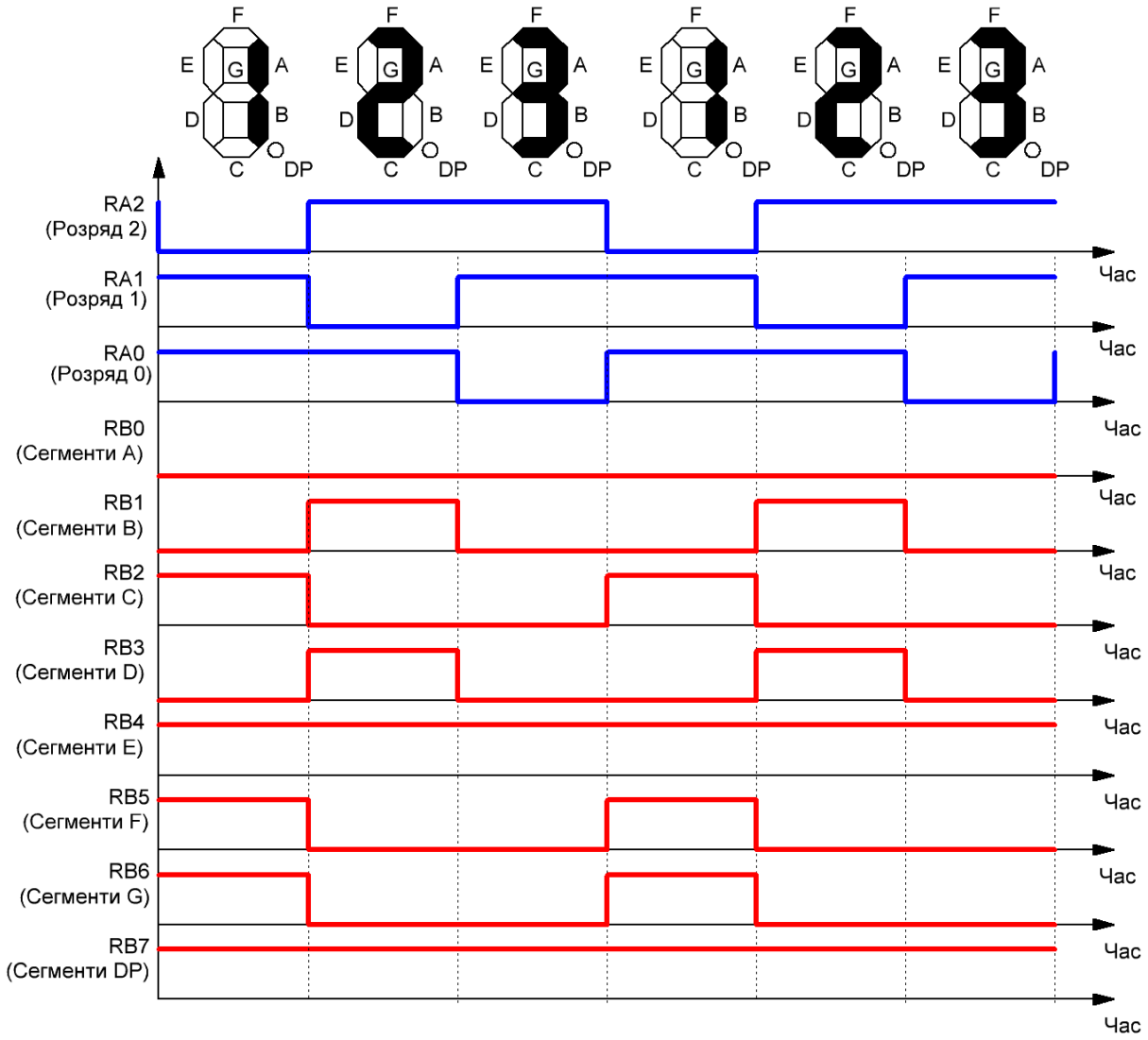


Рисунок 2 – Принцип динамічної індикації

Слід зауважити, що на виводах RA0 – RA2 порту PORTA у кожен момент часу низький рівень сигналу може бути присутній лише на одному із виводів. Одночасне формування низького рівня сигналу на кількох виводах RA0 – RA2 призведе до одночасного засвічення кількох сегментів, що, по-перше, спотворить інформацію, що відображається на індикаторі, а по-друге – може призвести до виходу з ладу мікроконтролера, оскільки через канали порту PORTB одночасно буде протікати струм кількох світлодіодів, який може перевищити максимально допустиме значення.

Частота оновлення розрядів індикатора, зазвичай, приблизно дорівнює 100 Гц. При такій частоті оновлення людське око вже не спроможне розрізнити блимання світлодіодів, тому людині буде здаватися, що усі розряди індикатора світяться одночасно.

### 3 Завдання на практичну роботу

Запрограмувати мікроконтролер PIC16F84, таким чином, щоб при замиканні одного із перемикачів він послідовно виводив на табло цифри від «000» до «999», збільшуючи цифри автоматично 1 раз за секунду. При замиканні другого перемикача повинно відбуватися встановлення значень рівним «000». Таким чином, схему, яка пропонується для тестування програмного забезпечення, можна розглядати як найпростіший секундомір.

Особливістю цієї роботи є те, що підключення сегментів індикатора HL1 (A – G, DP) до виводів RB0 – RB7 мікроконтролера DD1 залежить від номеру варіанта і заздалегідь невідоме. Таким чином, кожен здобувач повинен написати власну програму для мікроконтролера, яка буде правильно працювати тільки на схемі, номер якої буде відповідати номеру варіанта здобувача.

Встановити необхідний номер варіанту схеми можна через меню Налаштування → Налаштування системи. При цьому відкриється діалогове вікно, у якому можна буде встановити необхідний номер варіанту (рис. 3).

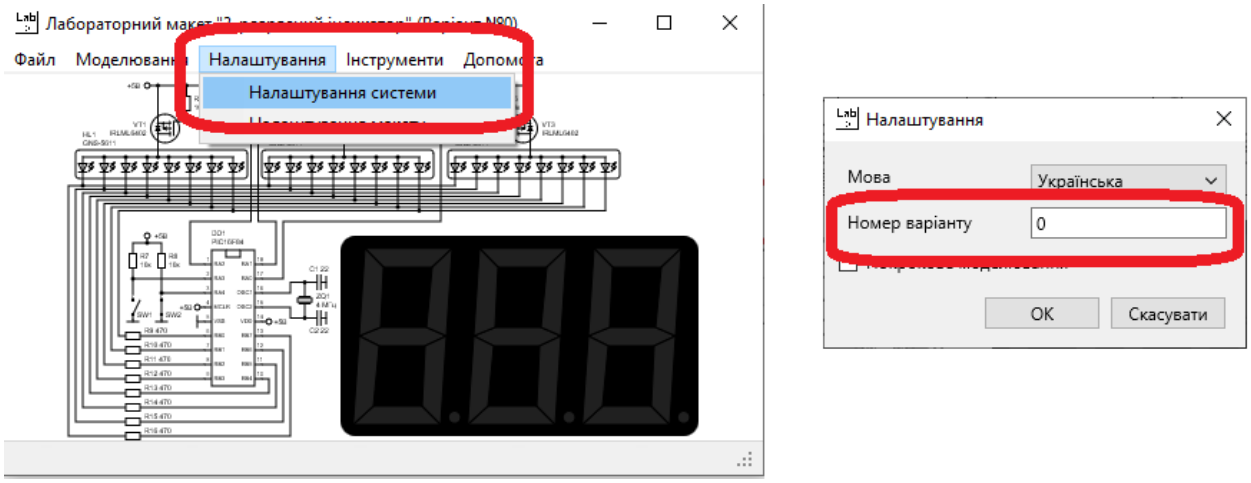


Рисунок 3 – Встановлення номеру варіанту лабораторного макету

Для отримання оцінок «Задовільно» (60 – 73 бали) та «Добре» (74 – 89 балів) та відмінно (90 – 100 балів) необхідно виконати практичне завдання в повному обсязі. На рівень оцінки впливає якість написання коду, у тому числі форматування та наявність коментарів, тому при неналежному форматуванні або при відсутності коментарів оцінка може бути знижена незалежно від обсягу виконаної роботи.

## 4 Рекомендації по розробці програмного забезпечення

### 4.1 Загальний алгоритм програми

Рекомендований алгоритм програмного забезпечення для виконання поставленої практичної задачі складається із трьох основних частин (рис. 4): блоку ініціалізації, основного циклу та обробника переривань.

У блоці ініціалізації потрібно налаштувати наступні апаратні периферійні модулі:

– порти введення-виведення PORTA та PORTB (регістри TRISA, TRISB, PORTA та PORTB) (блок 2);

– таймер Timer0 (регістри OPTION_REG, TMR0) (блок 3);

– механізм переривань (Interrupts) (регістр INTCON) (блок 4).

В основному циклі виконується перевірка станів перемикачів SW1 та SW2 (блоки 8, 9) шляхом контролю стану каналів RA3 та RA4 порту PORTA. У перериванні виконується відображення поточного значення цифри шляхом формування відповідних сигналів на виводах RA0 – RA3 порту PORTA та RB0 – RB7 порту PORTB.

У даній схемі використовується два порти: PORTA та PORTB. Усі канали порту PORTB працюють у режимі виведення, тому на етапі ініціалізації регістр TRISB достатньо просто очистити. Зверніть увагу, що у схемі використовуються світлодіодні індикатори зі спільним анодом, тому для запобігання небажаного «блимання» сегментів індикатора, після налаштування порту PORTB слід встановити на всіх каналах рівень, що відповідає рівню логічної одиниці (лістинг 1).

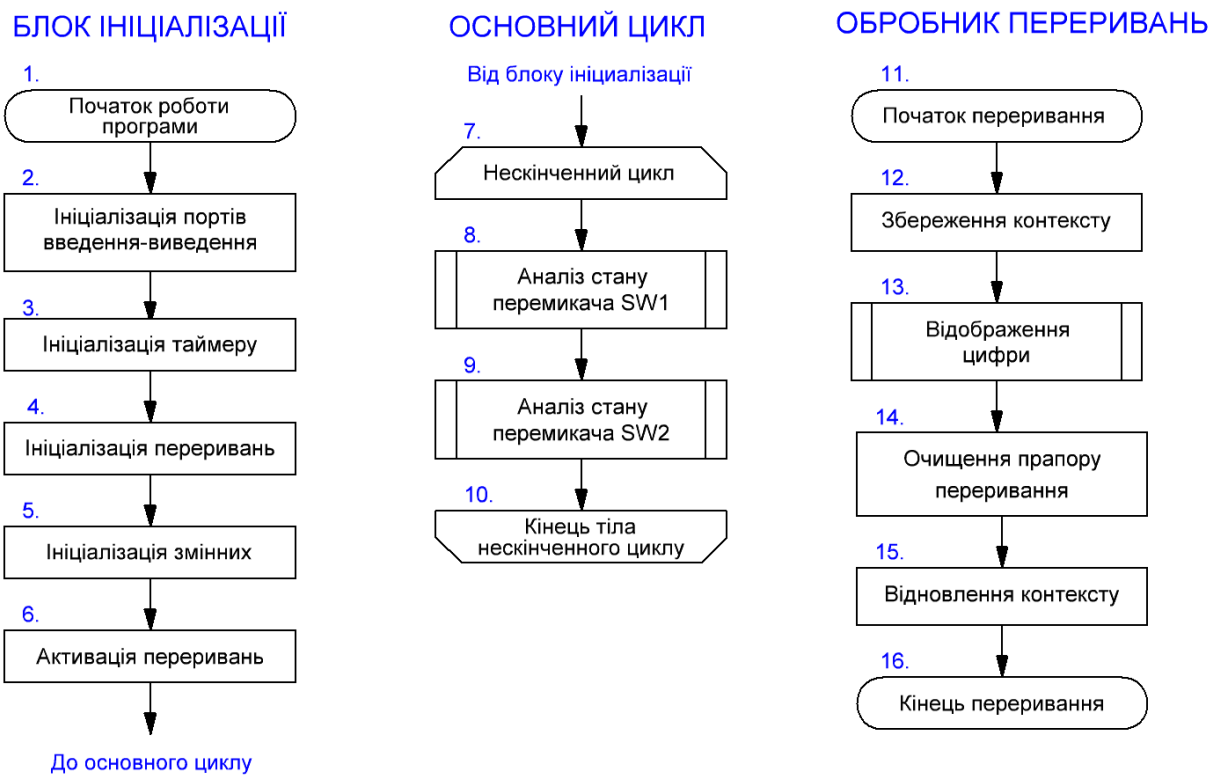


Рисунок 4 – Рекомендований алгоритм програми

```
Initialization:
    ; налаштування PORTB
    bsf     STATUS, RP0
    clrf    TRISB
    bcf     STATUS, RP0

    ; ініціалізація PORTB
    movlw   b'11111111'
    movwf   PORTB

    ; налаштування PORTA
    bsf     STATUS, RP0
    movlw   b'00011000'
    movwf   TRISA
    bcf     STATUS, RP0

    ; ініціалізація PORTA
    movlw   b'00011111'
    movwf   PORTA

    ;налаштування таймеру
    bsf     STATUS, RP0
    movlw   b'xxxxxxxx'
    movwf   OPTION_REG
    bcf     STATUS, RP0

    ;налаштування переривань
    movlw   b'xxxxxxxx'
    movwf   INTCON
```

```
    ;ініціалізація змінних
    ; масив чисел
    movlw   b'xxxxxxxx'
    movwf   Numerics + 0

    movlw   b'xxxxxxxx'
    movwf   Numerics + 1

    movlw   b'xxxxxxxx'
    movwf   Numerics + 2

    movlw   b'xxxxxxxx'
    movwf   Numerics + 3

    movlw   b'xxxxxxxx'
    movwf   Numerics + 4

    movlw   b'xxxxxxxx'
    movwf   Numerics + 5

    movlw   b'xxxxxxxx'
    movwf   Numerics + 6

    movlw   b'xxxxxxxx'
    movwf   Numerics + 7

    movlw   b'xxxxxxxx'
    movwf   Numerics + 8

    movlw   b'xxxxxxxx'
    movwf   Numerics + 9
```

```
    ;ініціалізація числа
    clrf    Digit0
    clrf    Digit1
    clrf    Digit2

    clrf    ShowDigitNumber

    ;активація переривань
    bsf     INTCON, GIE
```

Лістинг 1 – Рекомендований код ініціалізації мікроконтролера

Канали RA3 та RA4 порту PORTA працюють у режимі введення, а канали RA0 – RA2 – у режимі виведення. Так само як і для порту PORTB, для усунення небажаного «блимання», на етапі ініціалізації на всіх каналах порту PORTA, що працюють у режимі виведення, потрібно сформувати високий рівень вихідного сигналу.

Зверніть увагу, що канали RA5 – RA7 у мікроконтролері PIC16F84A фізично не реалізовані, тому у біти 5 – 7 у регістрах TRISA та PORTA можна записувати будь-яке значення, при цьому вони завжди зчитуються як 0.

На останньому етапі ініціалізації (блок 5) потрібно встановити початкове значення усіх змінних, які використовуються в програмі.

Зверніть увагу, що який канал порту PORTB підключений до якого сегменту індикатора (A – G, DP), доведеться з'ясовувати експериментально. Для цього можна на перших порах написати тимчасовий експериментальний код, в якому потрібно спочатку сформувати сигнал з рівнем логічного нуля на одному з виводів RA0 – RA2 (це призведе до засвічення одного з сегментів індикатора), а потім послідовно засвічувати по одному світлодіоду, і з'ясовувати, який сегмент буде світитися. Результати експерименту краще занести до таблиці 1.

Після цього, слід визначити, яку комбінацію символів потрібно сформувати на виводах мікроконтролера, щоб світилися потрібні сегменти (табл. 2).

Після визначення призначення кожного з виводів мікроконтролера можна починати писати основний код (тимчасовий експериментальний код при цьому можна видалити).

Таблиця 1 – Таблиця для визначення електричної схеми (кольором виділені комірки, що заповнюються самостійно)

| Тестова константа, що записується у регістр PORTB | Активний канал порту | Сегмент, який світиться (A – G або DP) |
|---|---|---|
| b'00000001' | RB0 | A |
| b'00000010' | RB1 | B |
| b'00000100' | RB2 | C |
| b'00001000' | RB3 | D |
| b'00010000' | RB4 | E |
| b'00100000' | RB5 | F |
| b'01000000' | RB6 | G |
| b'10000000' | RB7 | DP |

Таблиця 2 – Константи, що будуть формувати сигнали керування індикатором (кольором виділені комірки, що заповнюються самостійно)

| Цифра | Активні сегменти | Сегмент, який світиться (A – G або DP) |
|---|---|---|
| 0 | A, B, C, D, E, F | b'00111111' |
| 1 | A, B | b'00000011' |
| 2 | A, B, D, E, G | b'01011011' |
| 3 | A, B, C, D, G | b'01001111' |
| 4 | B, C, F, G | b'01100110' |
| 5 | A, C, D, F, G | b'01101101' |
| 6 | A, C, D, E, F, G | b'01111101' |
| 7 | A, B, C | b'00000111' |
| 8 | A, B, C, D, E, F, G | b'01111111' |
| 9 | A, B, C, D, F, G | b'01101111' |

### 4.2 Відображення цифр

Відповідно до технічного завдання, на індикаторі необхідно відображати цифри від «000» до «999». Оскільки мікроконтролер PIC16F84A э 8-розрядним, то одного байту для зберігання таких цифр буде замало (999 >> 255). Оскільки модуля апаратного множення/ділення у мікроконтролерах PIC16F84A немає, то для зберігання інформації про цифру, що відображається, краще використати три змінні (у прикладі – змінні Digit0 – Digit2), які будуть зберігати значення, відповідних розрядів. У цьому випадку, збільшення цифр відбувається порозрядно із перевіркою на перевищення максимального значення (рис. 5). Зміна цифри повинна відбуватися при низькому рівні сигналу на одному з каналів порту PORTA – RA3 або RA4, залежно від того, який перемикач ви виберете для збільшення цифри – SW1 чи SW2. Рекомендований код для збільшення значень цифри (блок 8 на рис. 4) наведено на лістингу 2.

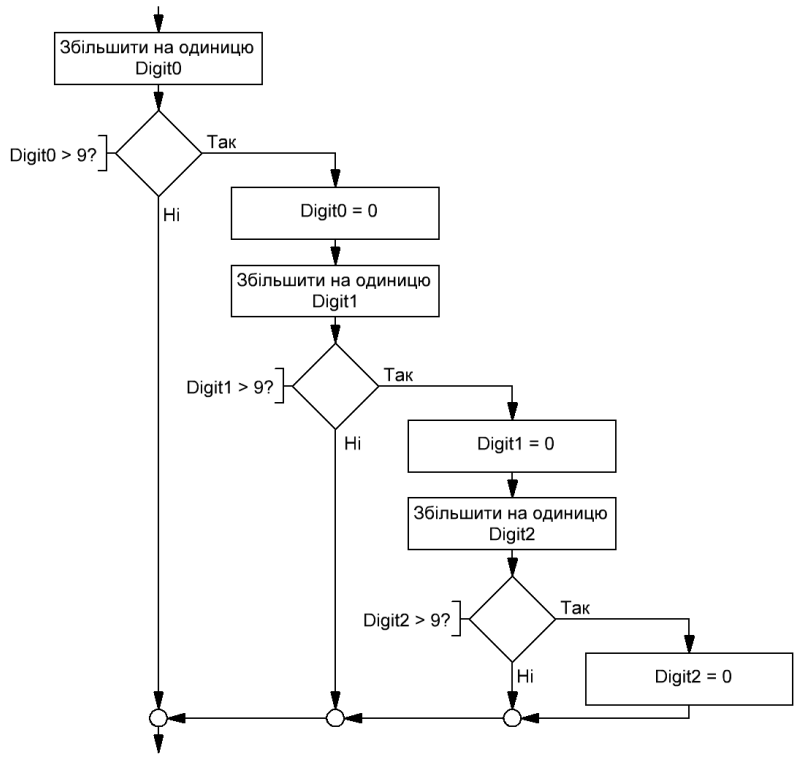


Рисунок 5 – Рекомендований алгоритм збільшення цифри

```
    ; перевірка перемикача збільшення цифри
CheckChangeDigits:
    btfsc   PORTA, d'x'
    goto    EndIncDigit

    ; збільшення розряду 0
    incf    Digit0, f

    ; перевірка розряду 0
    movfw   Digit0
    sublw   d'9'
    btfsc   STATUS, C
    goto    EndIncDigit

    ; збільшення розряду 1
    clrf    Digit0
    incf    Digit1, f

    ; перевірка розряду 1
    movfw   Digit1
    sublw   d'9'
    btfsc   STATUS, C
    goto    EndIncDigit

    ; збільшення розряду 2
    clrf    Digit1
    incf    Digit2, f

    ; перевірка розряду 2
    movfw   Digit2
    sublw   d'9'
    btfss   STATUS, C
    clrf    Digit2

EndIncDigit:
    ; подальший код
```

Лістинг 2 – Рекомендований код збільшення цифри

Очищення цифри (встановлення значення «000») повинно відбуватися при низькому рівні сигналу на іншому каналі порту PORTA – RA3 або RA4, залежно від того, яку кнопку ви для цього виберете – SW1 чи SW2. Рекомендований код для очищення цифри (блок 9 на рис. 4) наведено на лістингу 3.

```
; перевірка кнопки очищення
btfsc   PORTA, d'x'
goto    CheckClearDigits

clrf    Digit0
clrf    Digit1
clrf    Digit2

CheckClearDigits:
    ; подальший код
```

Лістинг 3 – Рекомендований код очищення цифри

Для відображення цифр рекомендується використовувати технологію непрямої адресації з використання регістру FSR у якості покажчика. Для цього в оперативній пам'яті мікроконтролера потрібно створити масив із 10 елементів, кожен елемент якого буде містити комбінацію сигналів, що відповідає певній цифрі (табл. 2). Таким чином, для реалізації наведеного технічного завдання необхідно використовувати наступні змінні (лістинг 4):

– Digit0 – Digit2 – змінні, призначені для зберігання поточного значення цифри (порозрядно);

– Numerics – масив із 10 елементів, в якому будуть зберігатися комбінації сигналів для порту PORTB, що відповідають цифрам «0» – «9» (табл. 2).

– ShowDigitNumber – змінна, в якій зберігається інформація про номер розряду, який зараз відображається на індикаторі.

Крім того, потрібні ще змінні для збереження контексту в перериванні (W_Temp та STATUS_Temp).

```
; змінні
cblock  0x0C
    W_Temp
    STATUS_Temp

    Digit0
    Digit1
    Digit2

    ShowDigitNumber

    Numerics: 10
endc
```

Лістинг 4 – Змінні, що використовуються в програмі

У реальних схемах код оновлення сегментів індикатора, зазвичай, виконується в обробнику переривання, що генерує один із таймерів мікроконтролера з частотою близько 100 Гц. Цей варіант також можна реалізувати і в даній схемі. Для генерації переривань із частотою 100 Гц потрібно налаштувати таймер і активізувати перериванні від нього.

Рекомендований код відображення інформації приведений на лістингу 5. Як видно з коду, сегменти індикатора відображаються по черзі. Якщо змінна ShowDigitNumber дорівнює 0, то відображається нульовий розряд індикатора, значення якого зберігається у змінній Digit0. Якщо значення змінної ShowDigitNumber дорівнює 1, тоді відображається перший розряд індикатора, значення якого зберігається у змінній Digit1. Якщо ж значення змінної ShowDigitNumber дорівнює 2, тоді відображається другий розряд індикатора, значення якого зберігається у змінній Digit2.

```
; обробник переривання
org     0x004

; збереження контексту
movwf   W_Temp
swapf   STATUS, w
movwf   STATUS_Temp

; вимикання порту A
movlw   b'11111111'
movwf   PORTA

; відображення розряду 0
movfw   ShowDigitNumber
xorlw   d'0'
btfss   STATUS, Z
goto    CheckDigit1

ShowDigit0:
movfw   Digit0
addlw   Numerics
movwf   FSR

movfw   INDF
movwf   PORTB

movlw   b'11111110'
movwf   PORTA

goto  ChangeShowDigitNumber
```

```
; відображення розряду 1
CheckDigit1:
    movfw   ShowDigitNumber
    xorlw   d'1'
    btfss   STATUS, Z
    goto    ShowDigit2

ShowDigit1:
    movfw   Digit1
    addlw   Numerics
    movwf   FSR

    movfw   INDF
    movwf   PORTB

    movlw   b'11111101'
    movwf   PORTA

    goto  ChangeShowDigitNumber

    ; відображення розряду 2
ShowDigit2:
    movfw   Digit2
    addlw   Numerics
    movwf   FSR

    movfw   INDF
    movwf   PORTB

    movlw   b'11111011'
    movwf   PORTA
```

```
ChangeShowDigitNumber:
    incf    ShowDigitNumber, f

    ; перевірка на максимальне значення
    movfw   ShowDigitNumber
    sublw   d'2'
    btfss   STATUS, C
    clrf    ShowDigitNumber

EndInterrupt:
    ; очищення прапору переривання
    bcf     INTCON, T0IF

    ; відновлення контексту
    swapf   STATUS_Temp, w
    movwf   STATUS
    swapf   W_Temp, f
    swapf   W_Temp, w

    ; повернення із переривання
    retfie
```

Лістинг 4 – Рекомендований код обробника переривань

Після відображення поточного розряду значення змінної ShowDigitNumber збільшується на одиницю, що призведе по відображення наступного розряду при наступному перериванні. Останнім кроком алгоритму є перевірка значення ShowDigitNumber. Якщо воно більше 2, тоді їй присвоюється значення 0. Такий алгоритм забезпечує циклічне послідовне відображення усіх трьох розрядів індикатора з частотою виникнення переривань.

Виведення комбінації сигналів на порт PORTB відбувається наступним чином. Значення змінних Digit0 – Digit2 знаходиться в межах 0 – 9. Таким чином, якщо додати до значення цих змінних адресу початку масиву сигналів (константа Numerics), то можна отримати адресу комірки пам'яті, в якій зберігається комбінація сигналів, що відповідає цій цифрі. Ця адреса спочатку зберігається в регістр FSR, а потім використовується технологія непрямої адресації (звернення за допомогою константи INDF до неіснуючої нульової комірки оперативної пам'яті). Отримане значення з комірок пам’яті Numerics[0] – Numerics[9] спочатку завантажується в акумулятор, а потім – у порт PORTB.

Зверніть увагу, що перед зміною сигналів на виводах порту PORTB відбувається гасіння усіх сегментів індикатора, шляхом встановлення високих рівнів сигналів на виводах RA0 – RA2 порту PORTA – це запобігає небажаному блиманню сегментів індикатора під час перехідних процесів. Після формування потрібних сигналів на порту PORTB, встановлюється низький рівень сигналу на одному з каналів порту PORTA (RA0, RA1 або RA2), залежно від того, який сегмент індикатора зараз відображається на індикаторі.

Не слід забувати, що у обробнику переривань змінюється значення акумулятора і регістра STATUS, тому для того, щоб код в основному циклі правильно працював, одразу після входу в переривання потрібно зберегти контекст (зміст акумулятора і регістру STATUS), а потім – перед виходом із переривання, його відновити.

## 5. Зміст звіту

Результатом виконання цієї практичної роботи є:

– файл вихідного коду програми «xxx.asm»;

– файл програми у шістнадцятковому форматі «xxx.hex».

Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

# 7. РЯДОК, ЩО БІЖИТЬ

## 1 Мета роботи

Створити застосунок, який на рядку, що біжить, покаже прізвище здобувача.

## 2 Опис апаратної частини

Апаратна частина складається із мікроконтролера DD1, який керує матричним світлодіодним індикатором HL1, що містить 40 світлодіодів, з'єднаних за матричною схемою (рис. 1).

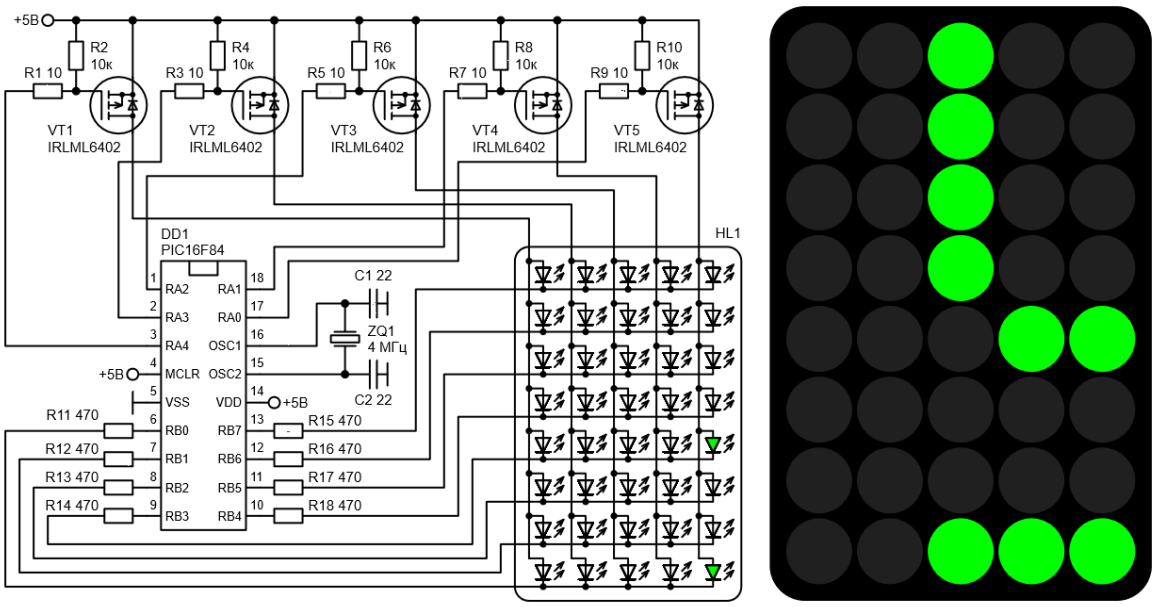


Рисунок 1 – Світловий автомат

Головним елементом апаратної частини є мікроконтролер DD1. Електрична енергія, необхідна для роботи мікроконтролера DD1, подається на пари виводів VSS-VDD. Напруга живлення мікроконтролера дорівнює 5 В. Вивід MCLR, призначений для апаратного перезавантаження мікроконтролера, у цій схемі не використовується, тому він підключений до джерела живлення, і на ньому завжди присутній сигнал з рівнем логічної одиниці. До виводів OSC1 та OSC2 підключений кварцовий резонатор ZQ1 із резонансною частотою 4 МГц. Необхідний режим роботи кварцового резонатора та тактового генератора забезпечується за допомогою двох керамічних конденсаторів C1 та C2 ємністю 22 пФ.

Індикатор HL1 містить п'ять вертикальних ліній, кожна з яких містить по вісім світлодіодних точкових індикаторів. Аноди усіх світлодіодів в межах вертикальної лінії об'єднані між собою. Кожна лінія має окремий вивід анодів, що дозволяє контролювати світінням кожної окремої лінії та використовувати для керування індикатором HL1 принцип динамічної індикації.

Струм світлодіодів кожного горизонтального рядку обмежується струмообмежувальними резисторами R11 – R18 з опором 470 Ом, які підключені до порту PORTB мікроконтролера DD1. Оскільки для світіння світлодіодного індикатора необхідно, щоб його катод по відношенню до аноду мав менший потенціал, то для засвічення світлодіодів на виводах порту PORTB потрібно формувати сигнали і з рівнем логічного нуля. Оскільки загальний струм усіх світлодіодів вертикальної лінії може перевищувати максимально допустимий струм каналу порту введення-виведення мікроконтролера, то вертикальні лінії підключені до нього через підсилювачі, реалізовані на транзисторах VT1 –

VT5. Підсилювачі сигналів мікроконтролера побудовані на основі польових транзисторів з ізольованим затвором з каналом p-типу (p-channel Metal-Oxide-Semiconductor Field-Effect Transistor, P-MOSFET) IRLML6402 виробництва компанії International Rectifier. Особливістю цих транзисторів є знижена порогова напруга між затвором та витоком (Threshold Voltage), що дозволяє використовувати для керування ними сигнали з логічними рівнями.

Підсилювачі сигналів на транзисторах VT1 – VT5 побудовані за схемами із загальними витоками, причому витоки усіх транзисторів з'єднані із шиною живлення +5 В. Для переводу каналів транзисторів типу P-MOSFET у провідний стан необхідно щоб потенціал їх затвору був менший за потенціал витоку, тому для того, щоб транзистори VT1 – VT5 почали пропускати струм між витоком та стоком, необхідно сформувати на виводах RA0 – RA2 мікроконтролера DD1 сигнали із рівнем логічного нуля. Наявність на виводі мікроконтролера сигналу із рівнем логічного нуля призведе до утворення негативної різниці потенціалів між затвором та витоком, що, у свою чергу, призведе до зменшення опору каналу між стоком та витоком транзисторів до величини приблизно 65 мОм. При формуванні на виводі мікроконтролера сигналу із рівнем логічної одиниці потенціал затвора стане рівним потенціалу витоку, провідний канал між стоком та витоком зникне, що призведе до розімкнення кола протікання струмів через усі сегменти розряду.

Резистори R2, R4, R6, R8 та R10 з опором 10 кОм призначені для утримання транзисторів VT1 – VT5 в непровідному стані, коли порти RA0 – RA2 мікроконтролера знаходяться у режимі введення, наприклад, після перезавантаження мікроконтролера. Резистори R1, R3, R5, R7 та R9 з опором 10 Ом обмежують струм у колі затворів транзисторів VT1 – VT2, які виникають під час перезаряду їх вхідної ємності (ємності між затвором та витоком).

### 3 Завдання на практичну роботу

Створити застосунок, який буде формувати на матричному світлодіодному індикаторі прізвище здобувача, який виконує цю роботу.

Для отримання оцінок «Задовільно» (60 – 73 бали) та «Добре» (74 – 89 балів) та відмінно (90 – 100 балів) необхідно виконати практичне завдання в повному обсязі. На рівень оцінки впливає якість написання коду, у тому числі форматування та наявність коментарів, тому при неналежному форматуванні або при відсутності коментарів оцінка може бути знижена незалежно від обсягу виконаної роботи.

### 4 Рекомендації по розробці програмного забезпечення

Спершу рекомендується намалювати текст, що буде відображатися на папері. Перші 5 стовпчиків при цьому слід повторити у самому кінці (Рисунок 2).

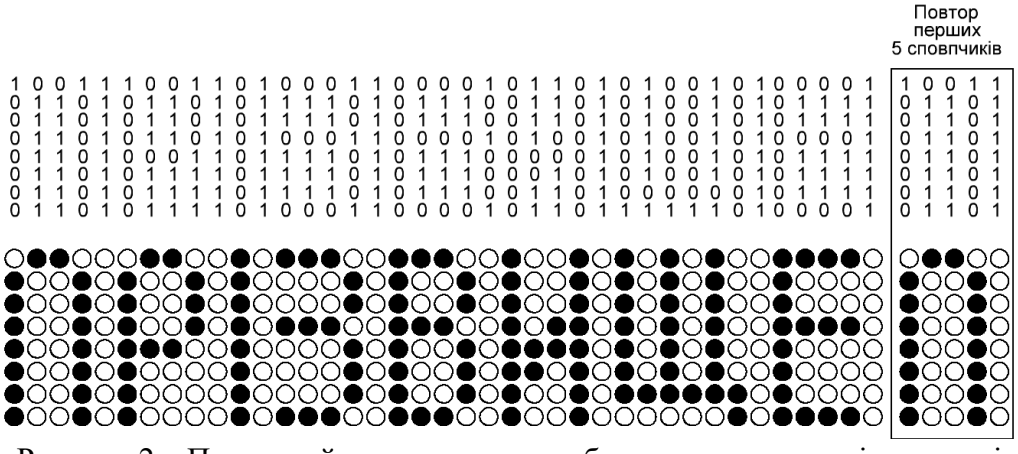


Рисунок 2 – Паперовий макет тексту, що буде виводитися на індикаторі

Після цього слід закодувати символи за допомогою констант, що завантажуються в акумулятор разом із поверненням із підпрограми (команда **retlw**) (лістинг 1). При цьому слід звернути увагу, що для того, щоб світлодіод у рядку світився, необхідно у відповідний розряд записати нуль. Молодший (нульовий) розряд при цьому відповідає верхньому рядку індикатора.

```
Start:
    retlw   b'00000001'     retlw   b'10111101'
    retlw   b'11101110'     retlw   b'11111111'
    retlw   b'11101110'     retlw   b'01110000'
    retlw   b'11110001'     retlw   b'01101111'
    retlw   b'11111111'     retlw   b'01101111'
    retlw   b'01110000'     retlw   b'10000000'
    retlw   b'01101111'     retlw   b'11111111'
    retlw   b'01101111'     ; повтор перших 5 стовпчиків
    retlw   b'10000000'     retlw   b'00000001'
    retlw   b'11111111'     retlw   b'11101110'
    retlw   b'10000001'     retlw   b'11101110'
    retlw   b'01111110'     retlw   b'11110001'
    retlw   b'01111110'     retlw   b'11111111'
```

Лістинг 1 – Рекомендований код для створення рядка, що біжить

Для відображення цифр слід рекомендується використовувати метод обчислювальних переходів, що обчислюється, який реалізовується шляхом додавання значення змінної ActiveLine до лічильника програм (лістинг 2).

```
GetCode:
    movlw   High(Start)
    movwf   PCLATH

    movlw   Low(Start)
    addwf   ActiveLine, w

    btfsc   STATUS, C
    incf    PCLATH, f

    movwf   PCL
```

Лістинг 2 – Рекомендований код визначення коду поточного рядка, який зберігається у змінній ActiveLine

Для послідовного відображення стовпчиків необхідна змінна Shift, яка буде зберігати поточне значення зміщення, відносно початку тексту (лістинг 3).

```
cblock  0x0C
    Shift
    ActiveLine
    PauseCounterLow
    PauseCounterHigh
endc
```

Лістинг 3 – Змінні, що використовуються у програмі

Спочатку ця змінна копіюється у змінну ActiveLine, після цього викликається підпрограма GetCode, яка повертає в акумулятор змінну, у якій закодовано, які сегменти поточного рядку потрібно засвітити. У даному випадку ці сегменти відповідають нульовому (крайньому лівому) стовпчику матричного індикатора. Отриманий код з акумулятора копіюється у порт PORTB, після цього на певний час (визначається підпрограмою Pause) засвічується нульовий рядок матричного індикатора шляхом запису 0 у відповідний розряд порту PORTA (лістинг 4).

```
    ; Основний цикл
MainLoop:
    ; стовпчик 0
    movfw   Shift
    movwf   ActiveLine

    call    GetCode
    movwf   PORTB

    movlw   b'11101111'
    movwf   PORTA

    call    Pause

    movlw   b'11111111'
    movwf   PORTA

    ; стовпчик 1
    incf    ActiveLine, f

    call    GetCode
    movwf   PORTB

    movlw   b'11110111'
    movwf   PORTA

    call    Pause

    movlw   b'11111111'
    movwf   PORTA

    ; стовпчик 2
    incf    ActiveLine, f

    call    GetCode
    movwf   PORTB

    movlw   b'11111011'
    movwf   PORTA

    call    Pause

    movlw   b'11111111'
    movwf   PORTA

    ; стовпчик 3
    incf    ActiveLine, f

    call    GetCode
    movwf   PORTB

    movlw   b'11111101'
    movwf   PORTA

    call    Pause

    movlw   b'11111111'
    movwf   PORTA

    ; стовпчик 4
    incf    ActiveLine, f

    call    GetCode
    movwf   PORTB

    movlw   b'11111110'
    movwf   PORTA

    call    Pause

    movlw   b'11111111'
    movwf   PORTA

    ; зміщення
    incf    Shift, f

    movfw   Shift
    sublw   MAX_SHIFT
    btfss   STATUS, C
    clrf    Shift

    goto    MainLoop
```

Лістинг 4 – Рекомендований код основного циклу

Такі ж самі операції повторюються і для наступних рядків.

**5 Зміст звіту**

Результатом виконання цієї практичної роботи є:
– файл вихідного коду програми «xxx.asm»;
– файл програми у шістнадцятковому форматі «xxx.hex».
Ці файли завантажуються у систему дистанційного навчання, перевіряються і оцінюються викладачем.

Навчальне видання

**Русу Олександр Петрович**

**ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ ТА РОБОТОТЕХНІКА В ЗАКЛАДАХ ОСВІТИ**

Методичні рекомендації до практичних робіт