Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

Реалізація мікроконтролерного керування електронної апаратури

Методичні вказівки
до виконання лабораторних робіт
для студентів спеціальностей
«Радіотехніка», «Апаратура радіозв'язку, радіомовлення і телебачення»,
«Радіоелектронні пристрої, системи та комплекси»,
«Біотехнічна та медична апаратура та системи»,
«Виробництво електронних засобів»,
«Електронна побутова апаратура»

Затверджено Методичною радою НТУУ «КПІ»

Київ
НТУУ «КПІ»
2008

Реалізація мікроконтролерного керування електронної апаратури: Метод. вказівки до викон. лаборатор. робіт для студ. спец. «Радіотехніка», «Апаратура радіозв'язку, радіомовлення і телебачення», «Радіоелектронні пристрої, системи та комплекси», «Біотехнічні та медичні апарати та системи», «Виробництво електронних засобів», «Електронна побутова апаратура» / Уклад.: Г.О. Мірських, С.Б. Могильний, В.С. Мосійчук – К.: НТУУ «КПІ», 2008.-___ с.

Гриф надано Методичною радою НТУУ «КПІ»
(Протокол № ____ від _____)

Навчальне видання

Реалізація мікроконтролерного керування електронної апаратури

Методичні вказівки
до виконання лабораторних робіт
для студентів спеціальностей
«Радіотехніка», «Апаратура радіозв'язку, радіомовлення і телебачення»,
«Радіоелектронні пристрої, системи та комплекси»,
«Біотехнічні та медичні апарати та системи»,
«Виробництво електронних засобів»,
«Електронна побутова апаратура»

Укладачі: *Мірських Георгій Олександрович, канд. техн. наук*
Могильний Сергій Борисович, канд. техн. наук, доцент
Мосійчук Віталій Сергійович

Відповідальний редактор О.І. Рибін, д-р техн. наук, проф.

Рецензент _____

За редакцією укладачів
Надруковано з оригінал-макета замовника

Темплан 2008 р., поз. _____

Підп. до друку _____. Формат 60x84 $^{1}/_{16}$. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум.друк.арк. ___. Обл.-вид.арк ____. Зам.№____.
Наклад 100 пр.

НТУУ «КПІ» ВПП ВПК «Політехніка»
Свідоцтво ДК № 1665 від 28.01.2004 р.
03056, Київ, вул..Політехнічна, 14, корп. 15
Тел./факс (044) 241-68-78

ЗМІСТ

Вступ 4

Мета та основні завдання лабораторних робіт 4

Правила техніки безпеки при виконанні лабораторних робіт 5

Лабораторна робота № 1 6

Лабораторна робота № 2

Лабораторна робота № 3

Лабораторна робота № 4

Список рекомендованої літератури

Додатки

Додаток 1. Регістри, що використовуються при виконанні лабораторних робіт

ВСТУП

Методичні вказівки до виконання лабораторних робіт з дисциплін «Мікропроцесори в радіоелектронних засобах» та «Мікроконтролерне керування електронної апаратури» призначені для студентів радіотехнічного факультету НТУУ «КПІ» очної та заочної форм навчання.

Методичні вказівки містять

- конспективне викладення теоретичного матеріалу, який має бути засвоєним для успішного виконання робіт (наводиться безпосередньо при описі кожної лабораторної роботи);
- завдання і порядок виконання лабораторних робіт.
- правила оформлення отриманих результатів.

МЕТА ТА ОСНОВНІ ЗАВДАННЯ ЛАБОРАТОРНИХ РОБІТ

Мета лабораторних робіт – набуття студентами навичок розробки програм керування радіотехнічними пристроями, до складу яких входять мікроконтролери; навичок ладнання та тестування цих програм.

Порядок оформлення та зміст звітів, що мають складатися студентами за результатами виконання лабораторних робіт вказано при описі порядку виконання кожної лабораторної роботи.

Для успішного виконання лабораторних робіт необхідно засвоєння студентом відповідного теоретичного матеріалу. Основні теоретичні відомості, необхідні для виконання лабораторної роботи, наведені у відповідних розділах методичних вказівок та доповнені контрольними питаннями з посилками на джерела відповідної інформації, що вказуються при описі кожної лабораторної роботи (узагальнений перелік рекомендованої літератури

наведено у відповідному розділі в кінці методичних вказівок).

ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ПРИ ВИКОНАННІ ЛАБОРАТОРНИХ РОБІТ

В учбовій лабораторії при виконанні лабораторних робіт використовується стандартна комп'ютерна техніка, яка відповідає чинним вимогам з техніки безпеки та промислової санітарії. Але наявність високої (220 В) напруги на розподільних електрощитах обумовлює виконання з боку студентів певних правил, з метою запобігання нещасних випадків, надання шкоди здоров'ю (як власному так і оточуючих).

1. Забороняється студентам самостійно виконувати будь-які вмикання на розподільних електрощитах.

2. В лабораторії в процесі роботи мають постійно знаходитись не менше двох осіб.

3. Забороняється студентам виконувати будь-які ремонтні роботи елементів електропроводки (вилки, розетки).

4. Забороняється перебування в лабораторії у верхньому одязі, класти верхній одяг на стільці, столи.

5. Робоче місце в процесі виконання робіт має бути організоване - вільне від сторонніх предметів (сумок, папок, сторонніх книг).

ЛАБОРАТОРНА РОБОТА № 1

Створення проекту в системі MPLAB

1.1. Мета та основні завдання роботи

Ознайомитись із основними складовими частинами системи MPLAB для створення, побудови та тестування проекту на мікроконтролерах фірми Microchip, вивчити їх можливості та взаємодію.

1.2. Основні теоретичні відомості

1.2.1. Запуск MPLAB IDE

Для запуску MPLAB IDE необхідно зробити подвійне натискання (подвійний «клик») на піктограму програми або вибрати *Start>Programs>Microchip>MPLAB IDE vx.xx>MPLAB IDE*. На екрані відкриється вікно, зображене на рис. 1.1.

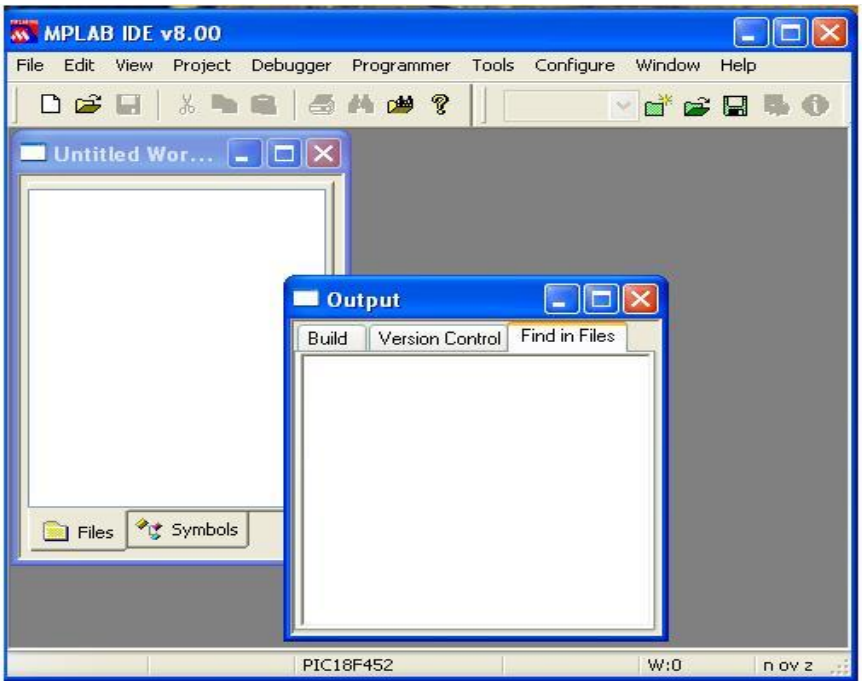


Рис.1.1. Стартове вікно MPLAB® IDE

Створення проекту полягає в виконанні наведених нижче кроків.

Вибір пристрою. Можливості MPLAB IDE залежать від вибраного пристрою. Вибір пристрою повинен бути виконаний до старту проекту.

Вибір методу створення проекту. Для створення проекту в рамках лабораторної роботи використовується майстер проекту MPLAB IDE (Project Wizard).

Вибір мовних інструментів. Мовні інструменти вибираються в процесі роботи з майстром проекту. В даній роботі використовуються вбудовані асемблер і лінкер. Для інших проектів можна вибрати один із компіляторів фірми Microchip, або інструменти сторонніх розробників. На даному кроці, за звичай, вибирається і назва проекту.

Введення (додавання) файлів у проект. В проект вводяться (додаються) два файли: тимчасовий файл (шаблон) і сценарій роботи лінкера (лінкування). Обидва файли знаходяться у відповідному підкаталозі каталогу MPLAB IDE. Це дозволяє легко почати проектування.

Створення програми. Необхідна програма створюється та додається в тимчасовий файл для передавання відповідних значень на порти входу/виходу.

Компіляція проекту. Компіляція проекту полягає в асемблеруванні і лінкуванні тимчасового файла для отримання програми в машинних кодах, яка потім викликається вибраним PIC MCU.

Перевірка програми симулятором. Програма перевіряється з використанням симулятора PIC MCU.

Майстер проекту спрощує проходження всіх названих кроків.

1.2.2. Вибір пристрою

Щоб показати вибір пунктів меню на панелі інструментів із верхнього рядка MPLAB IDE запишемо шлях: *MenuName>MenuItem*. Вибір підменю

Select Device в меню Configure: *Configure>Select Device*. Вибираємо *Configure>Select Device*. В діалоговому рядку Device вибираємо із списку мікроконтролер PIC16F84 (рис. 1.2).

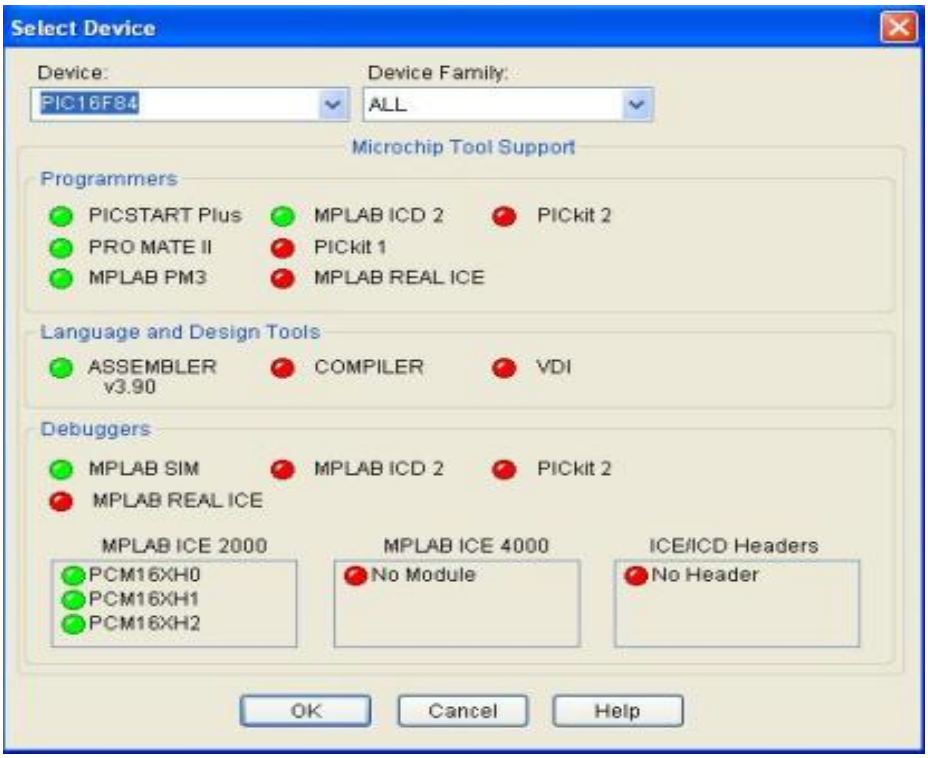


Рис.1.2. Вікно вибору пристрою

«Підсвічені» індикатори показують, які компоненти MPLAB IDE підтримують цей пристрій, а саме:

- зелений колір - повна підтримка;
- жовтий колір - часткова підтримка компонентів інструментів MPLAB IDE;
- червоний колір – відсутність підтримки даного пристрою. Підтримка також може бути некоректною для деяких інструментів, наприклад, пристрої dsPIC DSC не підтримуються MPLAB ICE 2000.

1.2.3. Вибір методу створення проекту

Наступний крок – вибір методу створення проекту – здійснюється з ви-

користанням майстра проекту - *Project>Project Wizard*. Проект – це форма організації файлів для компіляції. Використаємо для даного проекту простий файл асемблера і сценарій роботи лінкера. Із діалогового вікна привітання (Welcome), вибираємо *Next>* для продовження. Наступне діалогове вікно пропонує вибрати пристрій (що вже зроблено). Переконуємось, що відображується PIC16F84. Якщо це не так, то зі списку меню виберемо PIC16F84 і «кликнемо» *Next>*.

1.2.4. Вибір мовних інструментів

На другому кроці майстер проекту встановлює мовні інструменти, які будуть використані в цьому проекті. Вибираємо в діалоговому вікні Active Toolsuite "Microchip MPASM Toolsuite". "MPASM" і "MPLINK" повинні бути видимі в списку вікна (див. рис. 1.3). Виділіть кожного з них, щоб побачити їх розміщення. Якщо MPLAB IDE інстальований в запропоновану за замовчуванням папку, то шлях до асемблера MPASM записується як:

C:\Program Files\Microchip\MPASM Suite\mpasmwin.exe

Шлях до лінкера MPLINK:

C:\Program Files\Microchip\MPASM Suite\mplink.exe

Шлях до бібліотеки MPLIB:

C:\Program Files\Microchip\MPASM Suite\mplib.exe

Після закінчення «кликнемо» *Next>*. Якщо інструменти не відображуються коректно, отримуємо інформацію, наведену на рис. 1.4.

Відзначивши рядок інструмента у вікні Toolsuite Contents, вибираємо папку його розміщення за допомогою клавіші Browse (рис. 1.4) і добавляємо необхідний інструмент (рис. 1.5). Шляхи до папок за замовчуванням наведені вище.

На даному кроці слід ввести назву проекту і задати папку для його розміщення. Наприклад, проект матиме назву MyProject. Використовуючи

кнопку Browse, розміщуємо проект в папку з назвою своєї групи, наприклад, RT30_Projects. «Кликнемо» *Next*>.

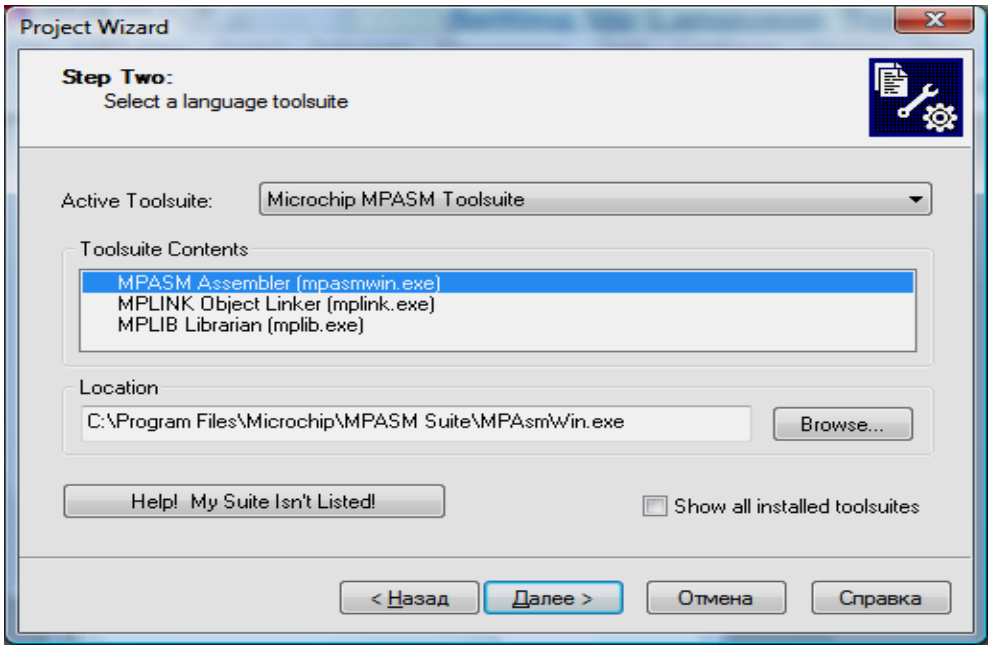


Рис.1.3. Майстер проекту. – Вибір мовних інструментів

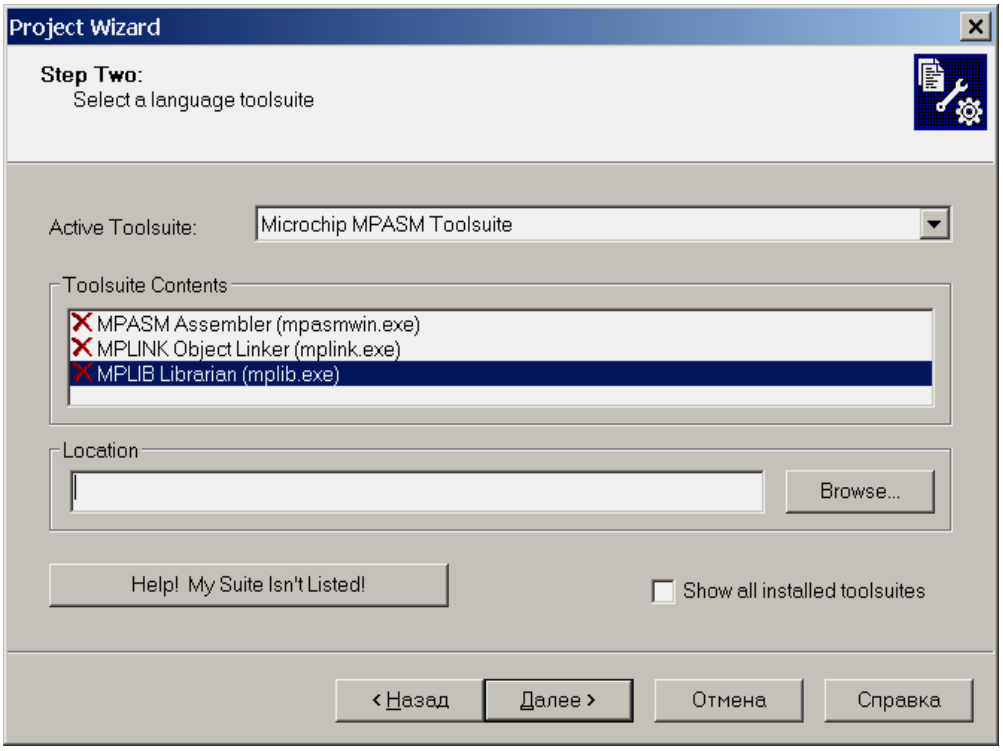


Рис.1.4. Вікно вибору інструментів при некоректному виборі

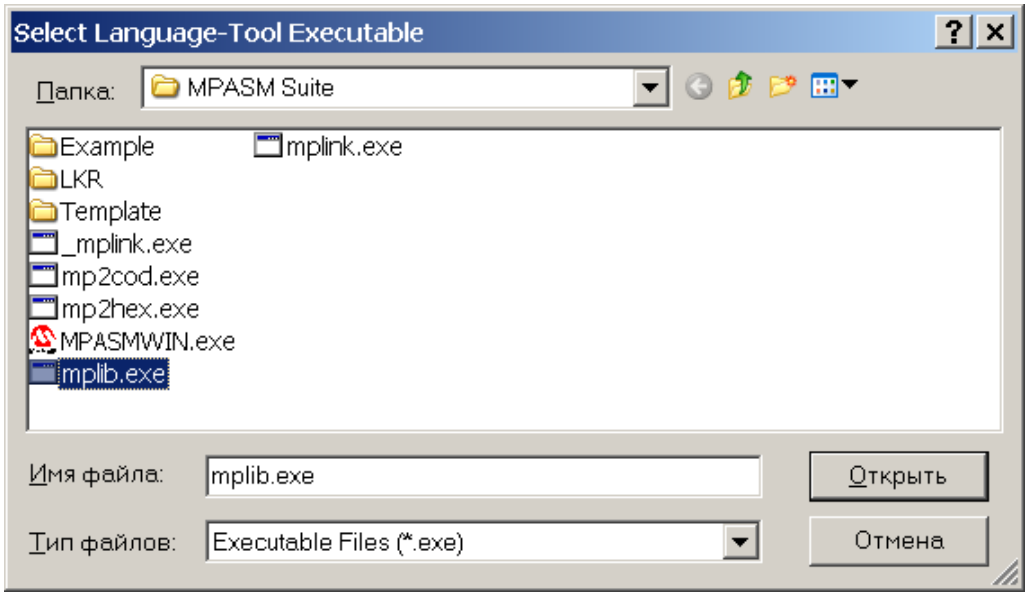


Рис.1.5. Вибір інструментів із папки

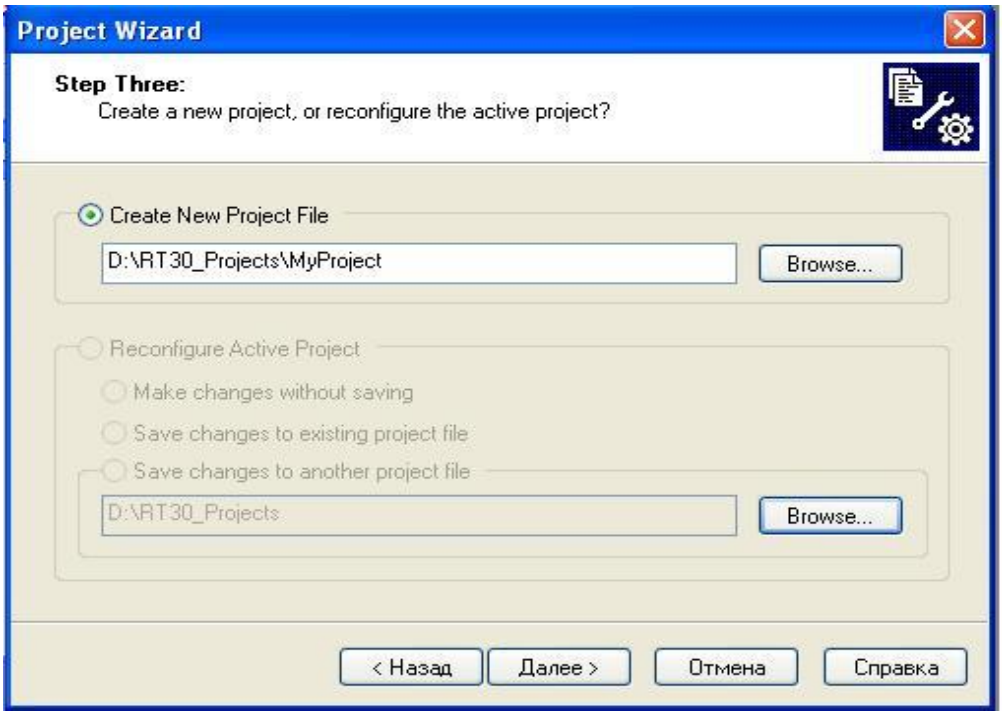


Рис.1.6. Майстер проекту. – Введення назви проекту

1.2.5. Введення (додавання) файлів у проект

На четвертому кроці виконується введення (додавання) вибраних файлів у проект. Файл першоджерела ще не вибраний і тому ми будемо вико-

ристовувати тимчасовий файл MPLAB IDE. Тимчасовий файл є файлом найпростішого шаблону, що дозволяє використати його для швидкого запуску проекту. До структури файлу входять розділи, присутні в будь-якому файлі першоджерела. Він вміщує інформацію, яка допоможе записати та розмістити код. Фірма Microchip пропонує один тимчасовий файл в якості шаблону для пристрою PIC MCU.

Вибираємо файл з назвою 16F84TMPO.ASM. Якщо MPLAB IDE інстальований в папку по замовчуванню, то повний шлях до тимчасової папки: *C:\Program Files\Microchip\MPASM Suite\Template\Object\16f84tmpo.asm*

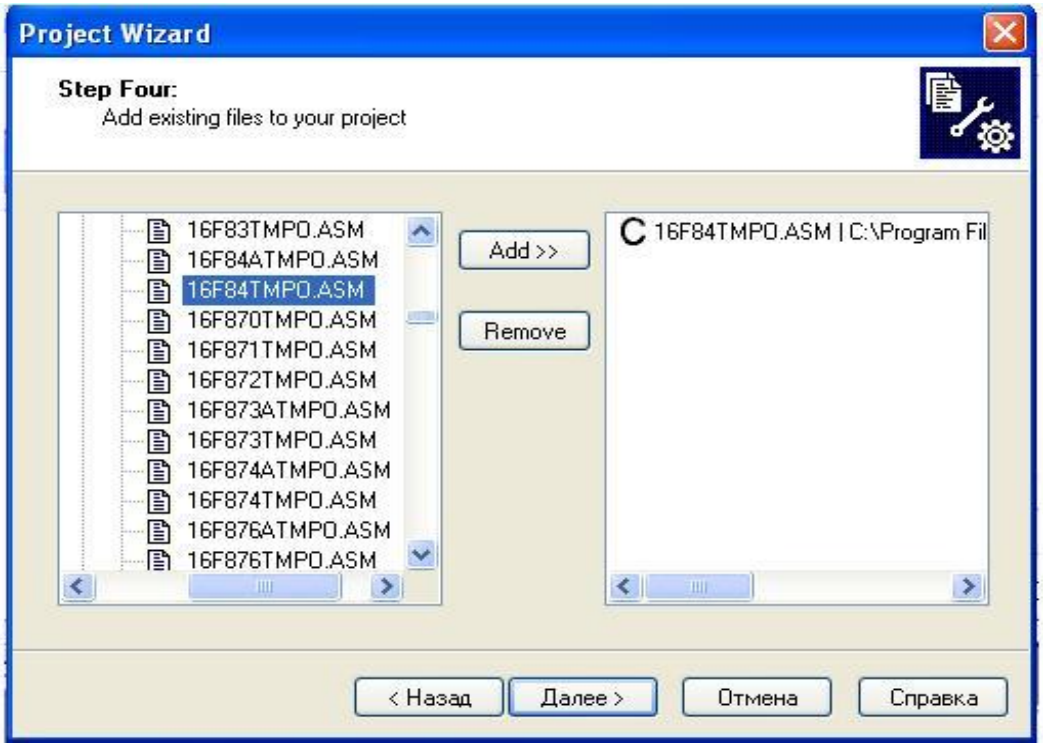


Рис.1.7. Майстер проекту. – Вибір тимчасового файлу

Натискуємо *Add>>*, щоб додати названий файл в праву панель і змінюємо біля назви вибраного файлу символ. Серед можливих варіантів (**A**>**U**>**S**>**C**) необхідно обрати останній, оскільки дана опція копіює файл шаблону до директорії проекту і додає в проект відносний шлях до скопійованого файлу. Будь-які наступні зміни в шаблоні тимчасового файлу бу-

дуть зберігатися у локальній копії, не модифікуючи оригінал. Крім того відносний шлях до файлу додає мобільності проекту, наприклад команда Save project as дозволяє зберегти проект в новому місці на локальному диску та/або перенести його на інший комп'ютер без додаткового настроювання.

Додамо в проект другий файл – файл сценарію роботи лінкера (файл лінкування). Для кожного пристрою існують свої файли лінкування. Ці файли задають відповідні регістри і конфігурацію пам'яті. Використаємо файл з назвою 16F84.lkr. Повний шлях до нього:

C:\Program Files\Microchip\MPASM Suite\LKR\16F84.lkr

Змінюємо символ **A** (заданий по замовчуванню) на **C** для копіювання цього файлу лінкування в проект.

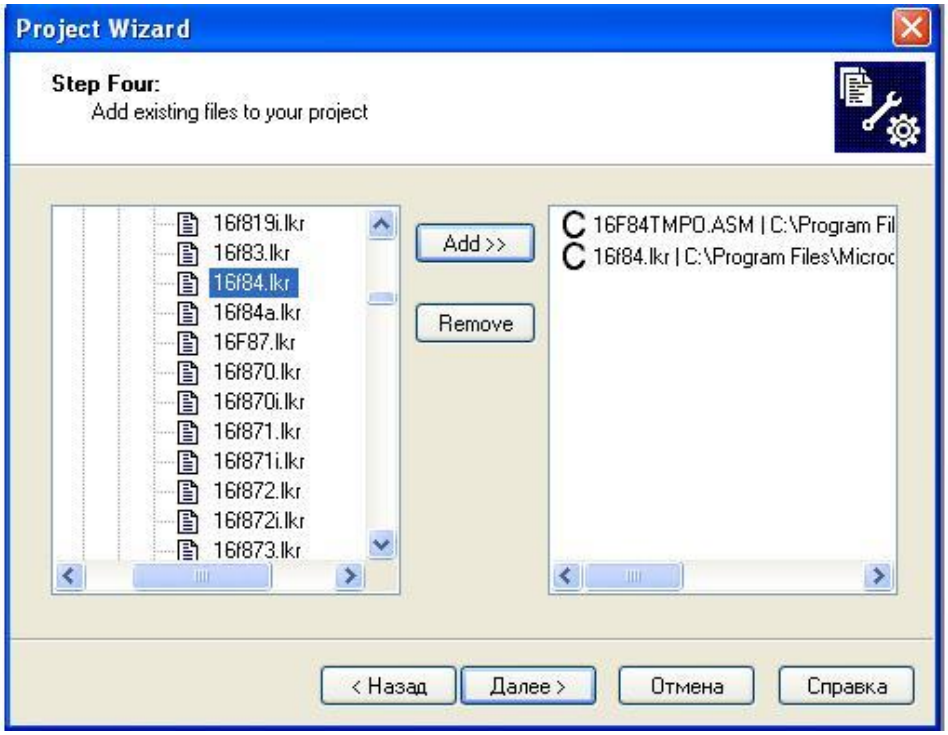


Рис. 1.8. Майстер проекту. – Вибір сценарію лінкування

Переконуємось, що діалогове вікно виглядає так, як на рис. 1.8 з двома символами **C** і натискуємо *Next>* для завершення роботи майстра проекту.

Заключне вікно майстра проекту показує підсумки вибору пристрою,

інструментів і назву файла нового проекту.

Після натискання кнопки Finish, з'являється вікно проекту на робочому столі MPLAB IDE, яке має вигляд, як на рис. 1.9. Якщо вікно проекту не відкрилось, то виберіть *View>Project*.

Порада: Файли можуть бути добавлені в проект збережений з використанням правої кнопки миші у вікні проекту. Якщо виникла помилка, то файли можна видалити з проекту в ручному режимі, попередньо вибравши їх і використавши контекстне меню правої кнопки миші.

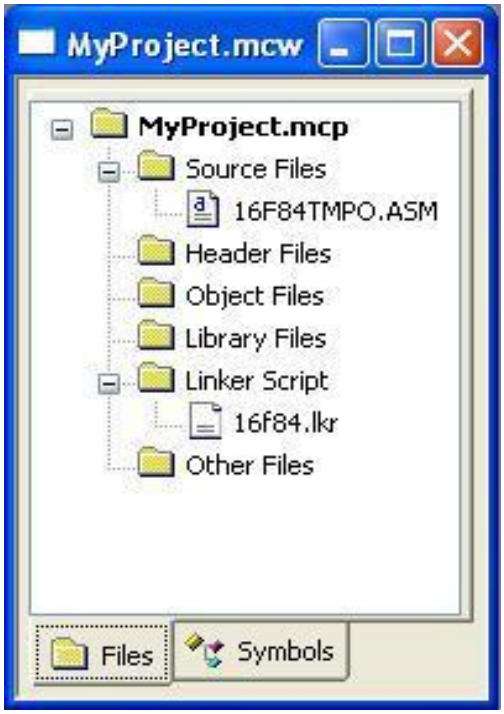


Рис.1.9. Вікно проекту

1.2.6. Тестова компіляція

Щоб переконатися в правильності виконання операцій введення в проект файлів шаблону слід здійснити тестову компіляцію. Для цього з використанням меню проекту виконуємо компіляцію добавлених файлів одним з наступних способів:

- *Project>Build All*
- «Кликнути» правою клавішею миші на назві проекту у вікні проекту і вибрати Build All
- «Кликнути» піктограму Build All на панелі інструментів проекту. Провести мишею через піктограми, щоб прочитати підказки.

Вихідне вікно показує результати процесу компіляції. В нашому випадку не повинно бути жодного повідомлення про помилки, або попередження. Якщо є повідомлення про помилки, то необхідно повернутися до попе-

реднього розділу, виправити помилки і повторити компіляцію. Повідомлення типу попередження можна ігнорувати, оскільки вони не призведуть до зупинки побудови проекту. Щоб вимкнути показ попереджень:

- виберіть *Project>Build Options>Project* і «кликніть» MPASM Assembler;
- виберіть "Output" із списку "Categories";
- виберіть "Errors only" із списку "Diagnostic level";
- натисніть OK.

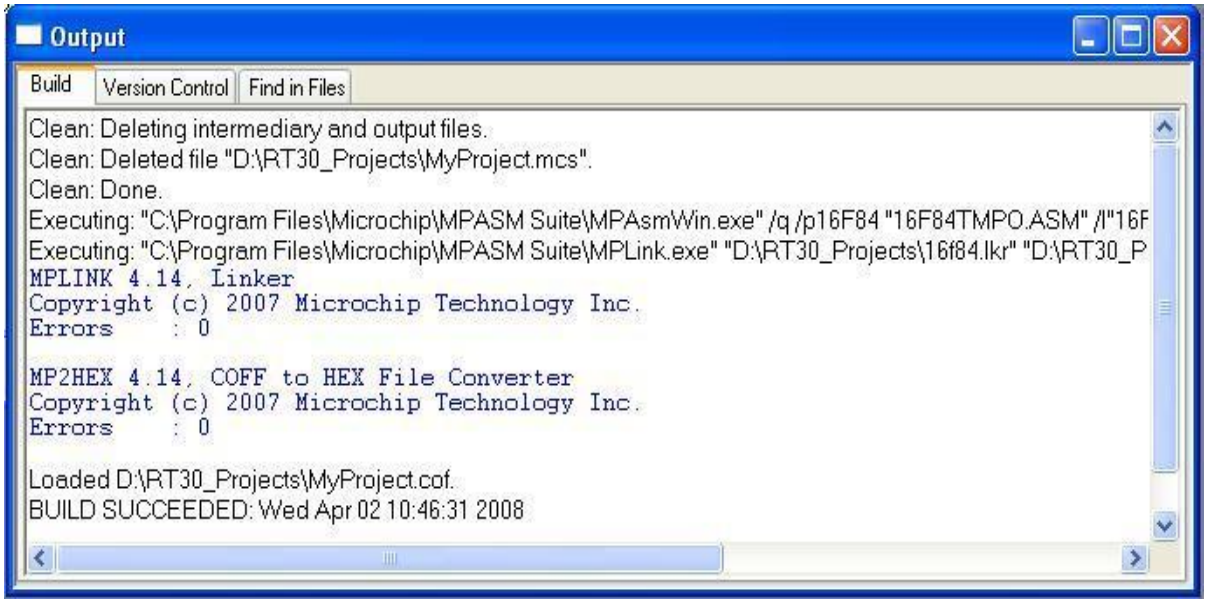


Рис.1.10. Вихідне вікно

1.2.7. Створення програми

Відкрийте тимчасовий файл подвійним клацанням мишкою на його назві у вікні проекту, або поставте на нього курсор миші і із контекстного меню правої клавіші виберіть Edit (рис. 1.11).

Файл на початку має коментарі і їх простір може бути використаний для стандартних коментарів програми. Зараз ми залишимо їх так, як вони є, але для реального проекту в цьому місці необхідно розмістити інформацію у відповідності з даними проекту.

Примітка: Можна показати номера рядків. Показ номерів рядків вмикається або вимикається через контекстне меню правої клавіші миші у вікні редактора. В меню вибрати «Властивості», а потім вибрати "Line Numbers" в розділі Editor діалогового вікна Editor Options.

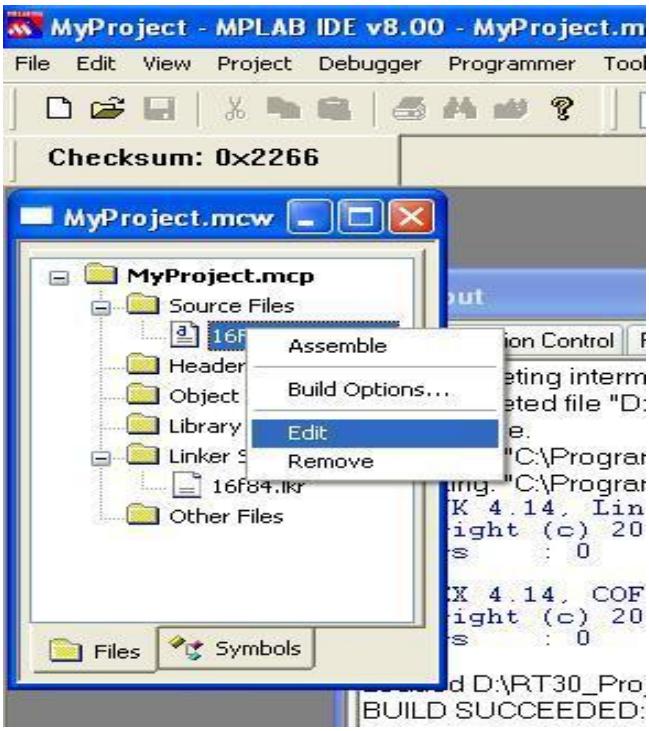


Рис. 1.11. Контекстне меню проекту

Програма першого розділу файлу використовується для складніших функцій, таких як настроювання переривань та біти конфігурації кінцевого пристрою. Пропустимо ці деталі, звернувши увагу на запис частини програми, яка розміщується у файлі після символу **Main**. Прокрутіть текст вниз до кінця файлу, де знаходиться символ **Main** (рис. 1.13).

Відкритий первинний файл можна автоматично редагувати. Наведений нижче фрагмент програми прикладу введіть в текст тимчасового файлу після символу **Main** (можна використати буфер обміну для копіювання):

```
    clrf    W
    movwf   PORTB        ;  обнулення PORTB
    bsf     STATUS,RP0   ;  вибір bank 1
    movwf   TRISB        ;  конфігурування всіх виводів PORTB як вихідних
    bcf     STATUS,RP0   ;  вибір bank 0
Init
    clrf     COUNT       ;  ініціалізація лічильника
```

```
IncCount
   incf      COUNT,f
   movf      COUNT,w   ; збільшити значення лічильника і
   movwf     PORTB     ; вивести через PORTB
   call      Delay     ; перейти на підпрограму Delay
   goto      IncCount  ; безкінцевий цикл
Delay
   movlw     0x40
   movwf     DVAR2     ; задати зовнішній цикл затримки
DelayOuter
   movlw     0xFF
   movwf     DVAR      ; задати внутрішній цикл затримки
DelayInner
   decfsz    DVAR,f
   goto      DelayInner
   decfsz    DVAR2,f
   goto      DelayOuter
   return
```

В даному фрагменту програми використані три змінні: **COUNT, DVAR, DVAR2,** які потрібно визначити в тимчасовому файлі. Використаємо блок **UDATA** (рис.1.14), що дозволить спростити код, а також відслідкувати значення змінних при виконанні симуляції. Встановимо початковий адрес блоку 0x20 (початок регістрів загального призначення). Кожна із змінних 8-ми бітова, тому слід зарезервувати лише по 1 байту для кожної з них.

```
UDATA   0x20
COUNT   RES  1
DVAR    RES  1
DVAR2   RES  1
```

D:\RT30_Projects\16F84TMP0.ASM

```
;*****************************************************************
;   This file is a basic code template for assembly code generation   *
;   on the PIC16F84. This file contains the basic code                 *
;   building blocks to build upon.                                     *
;                                                                      *
;   If interrupts are not used all code presented between the ORG      *
;   0x004 directive and the label main can be removed. In addition     *
;   the variable assignments for 'w_temp' and 'status_temp' can        *
;   be removed.                                                        *
;                                                                      *
;   Refer to the MPASM User's Guide for additional information on      *
;   features of the assembler (Document DS33014).                      *
;                                                                      *
;   Refer to the respective PIC data sheet for additional              *
;   information on the instruction set.                                *
;                                                                      *
;   Template file assembled with MPLAB V3.99.18 and MPASM V2.15.06.    *
;                                                                      *
;*****************************************************************
;                                                                      *
;    Filename:       xxx.asm                                           *
;    Date:                                                             *
;    File Version:                                                     *
```

Рис. 1.12. Тимчасовий файл

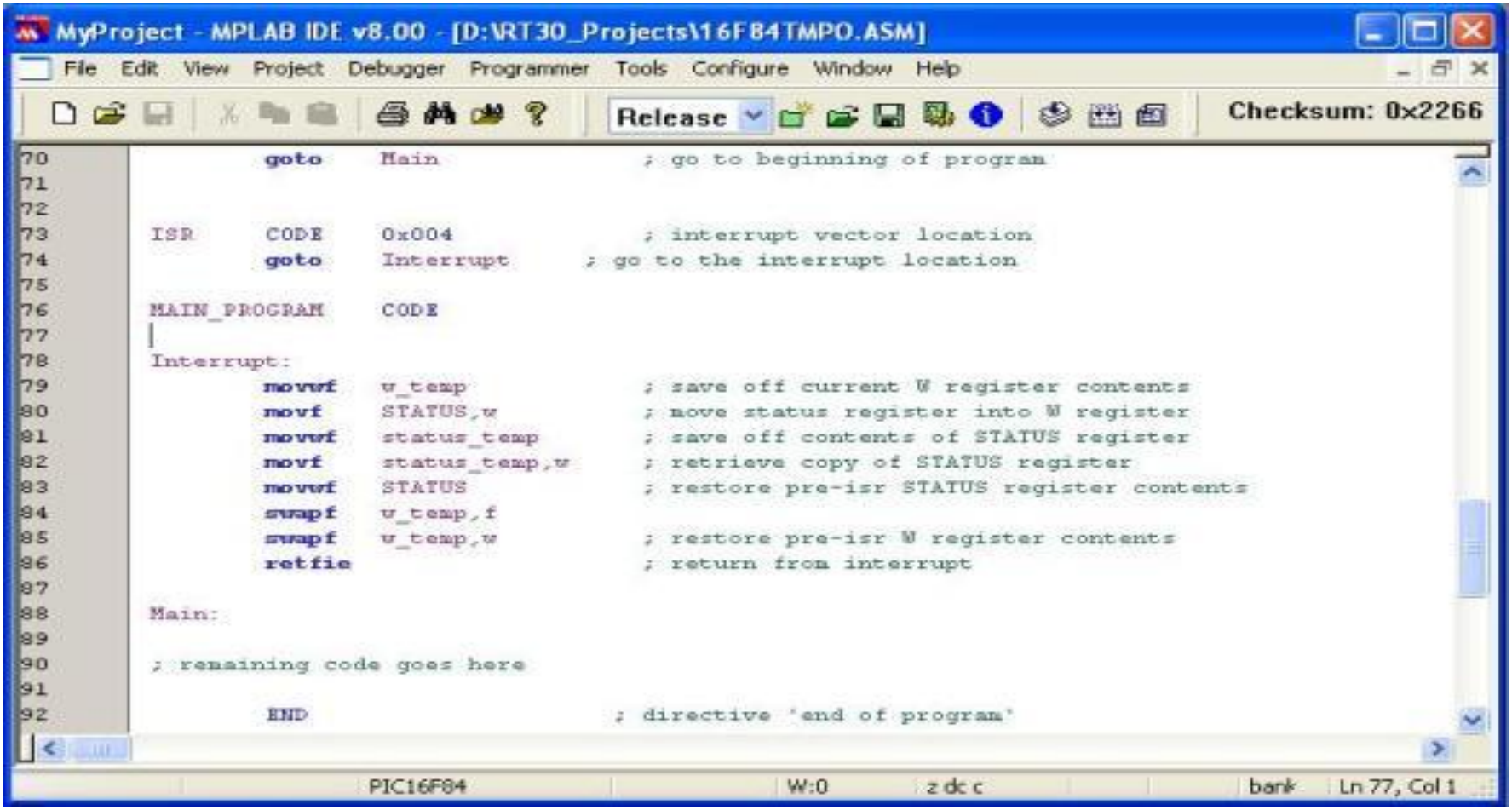


Рис. 1.13. Тимчасовий файл – Основний (**Main**)

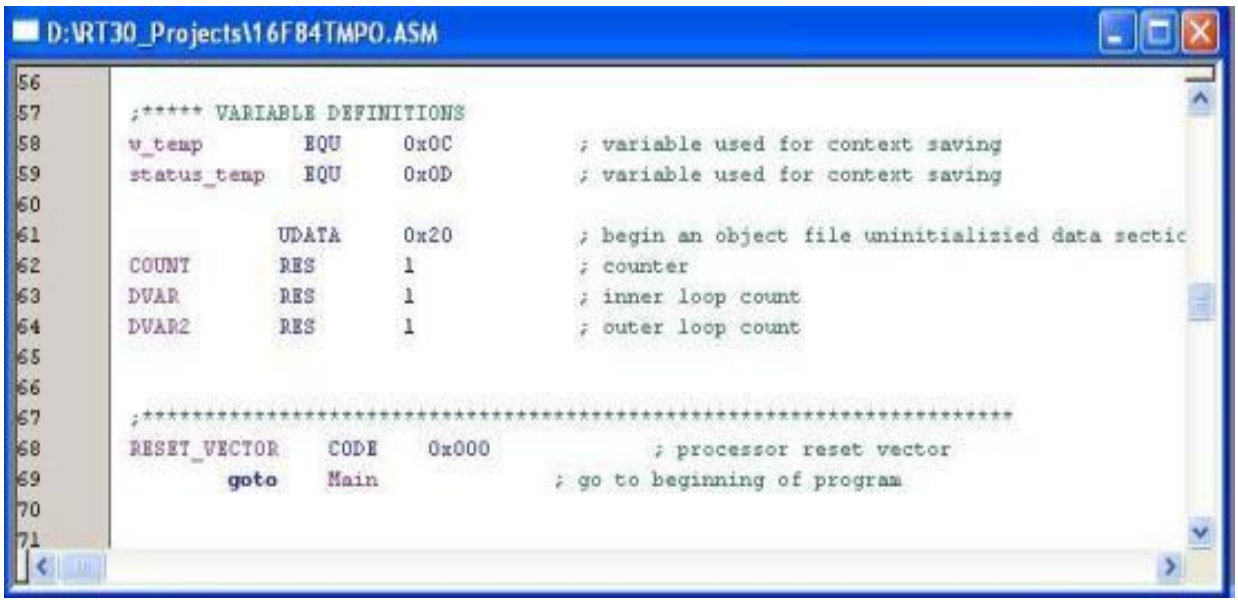


Рис.1.14. Тимчасовий Файл. Визначення змінних

1.2.8. Компіляція проекту

Виберіть *Project>Build All*. Якщо компіляція виконана без помилок, то вихідне вікно матиме вигляд, подібний наведеному на рис. 1.15.

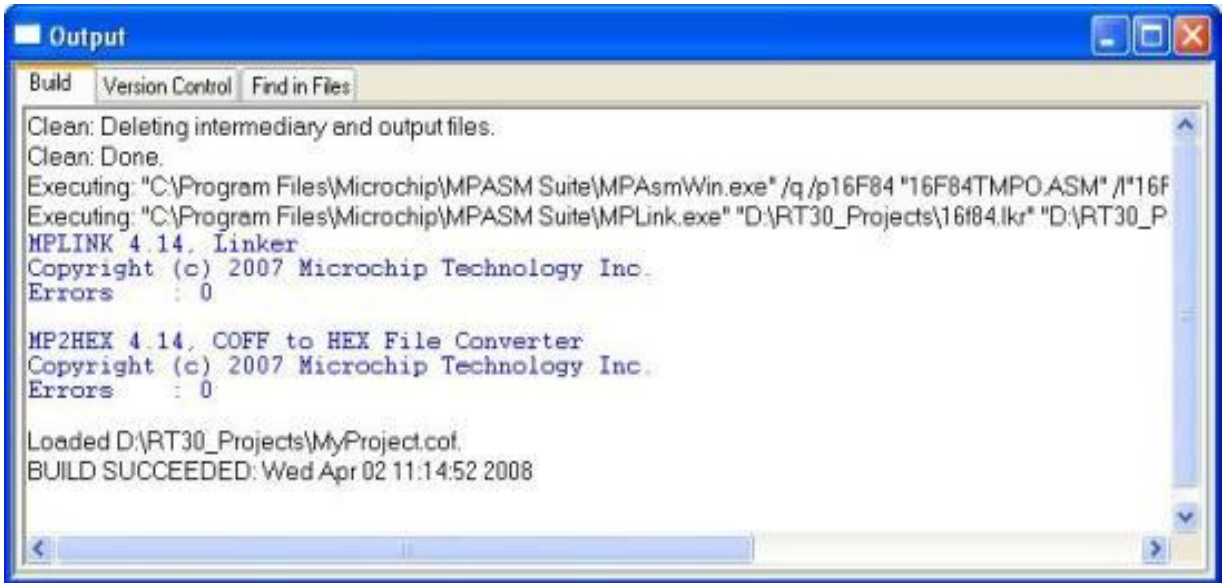


Рис. 1.15. Вихідне вікно (звіт про компіляцію)

Якщо компіляція виконана з помилками, то повторіть побудову проекту, попередньо виконавши наступне.

- Якщо в звіті вихідного вікна є помилки компіляції, то кликніть подвійно на помилці. MPLAB IDE відкриє відповідний рядок в програмі першоджерела із відміткою зеленого кольору в лівій частині вікна.
- Перевірте синтаксис і формат введеної програми у вікні редактора. Переконайтесь, що нові змінні, регістри спеціальних функцій, TRISB і PORTB, визначені у відповідних розділах.
- Перевірте, що для пристроїв PIC MCU використовуються коректні версії асемблера (MPASM асемблер) і лінкера. Виберіть *Project>Set Language Tool Locations*. Розверніть інструмент Microchip MPASM і його складові. «Кликніть» на MPASM асемблері (*mpasmwin.exe*) і перегляньте шлях до його розміщення. Якщо розміщення коректне, то «кликніть» *Cancel*. Якщо ні, то змініть його і потім натисніть *OK*.

Після успішної побудови буде завантажений вихідний файл, генерований мовним інструментом. Цей файл вміщує об'єктну програму, яка може бути записана в мікросхему PIC MCU, і інформацію для налагодження. Тому первинна програма може бути налагоджена, а змінні першоджерела переглянуті у вікні спостереження Watch, як символи.

1.2.9. Перевірка програми симулятором

Для завершення тестування програми або пристрою необхідно виконати команди PIC MCU. Інструментами для налагодження є програмні або апаратні інструменти, що використовуються для перевірки програми (в нашому випадку *84tmpo.asm*). Апаратні інструменти, такі як MPLAB ICE, MPLAB ICD 2 або MPLAB PICkit2 можуть виконати програму в реальних пристроях. Якщо апаратні інструменти недоступні, то для перевірки можна використати симулятор MPLAB SIM. Використаємо його в даній роботі.

Симулятор є програмним інструментом, який запускається на комп'ютері для симуляції команд PIC MCU. Він не працює в "реальному

часі", тому що програма симулятора залежить від швидкості комп'ютера, складності скомпільованої програми (коду), типу операційної системи, а також від кількості задач одночасно запущено на виконання. Незважаючи на це, симулятор точно визначає час, який був би витрачений на виконання програми при роботі пристрою в реальному часі.

Вибираємо симулятор, як інструмент виконання налагодження. Він доступний із меню *Debugger>Select Tool*. Після вибору MPLAB SIM стають доступними додаткові пункти меню налагодження (рис. 1.16).

Далі вибираємо *Debugger>Reset>Processor Reset* і зелена стрілка показує, звідки почнеться програма (див. рис. 1.17). На рис. 1.17 Показана ділянка тимчасового файлу. Перша команда в пам'яті викликає безумовний перехід на мітку **`Main`**, куди був вставлений наш фрагмент. Цією командою ми «перестрибуємо» в пам'яті через простір вектора переривання PIC16F84.

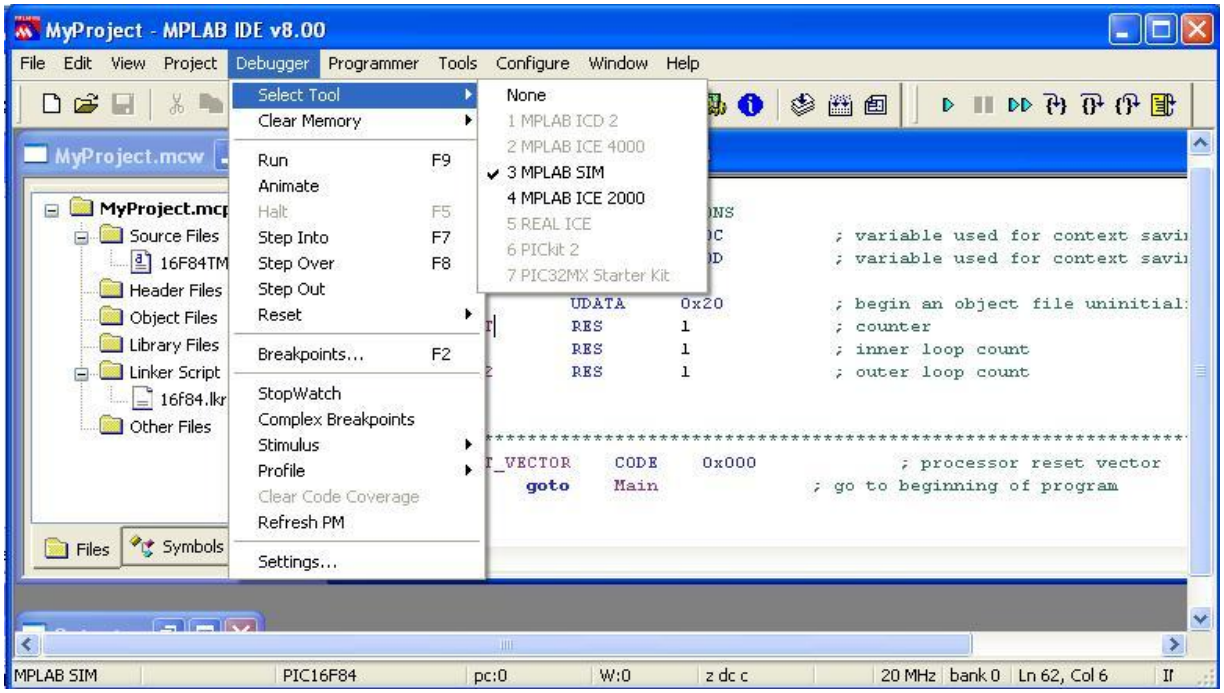


Рис.1.16. Вікно MPLAB® IDE із симулятором MPLAB SIM для ладнання

Для виконання програми «крок за кроком» виберіть *Debugger>Step*

Into. При цьому буде відзначатись рядок програми, що виконується, і відмітка буде переміщуватись до наступного рядка, що має бути виконаний. В табл. 1.1 показані функції піктограм панелі інструментів Debug Tool Bar.

Таблиця 1.1. Функції засобів ладнання MPLAB Debug Tool Bar.

Меню ладнання програми	Швидкий виклик	Функція
Run	F9	Запуск програми
Halt	F5	Зупинка програми
Animate		Анімація виконання програми
Step Into	F7	Покрокове виконання
Step Over	F8	Обхід підпрограми (кола затримки)
Step Out		Вихід з підпрограми (кола затримки)
Reset	F6	Скидання

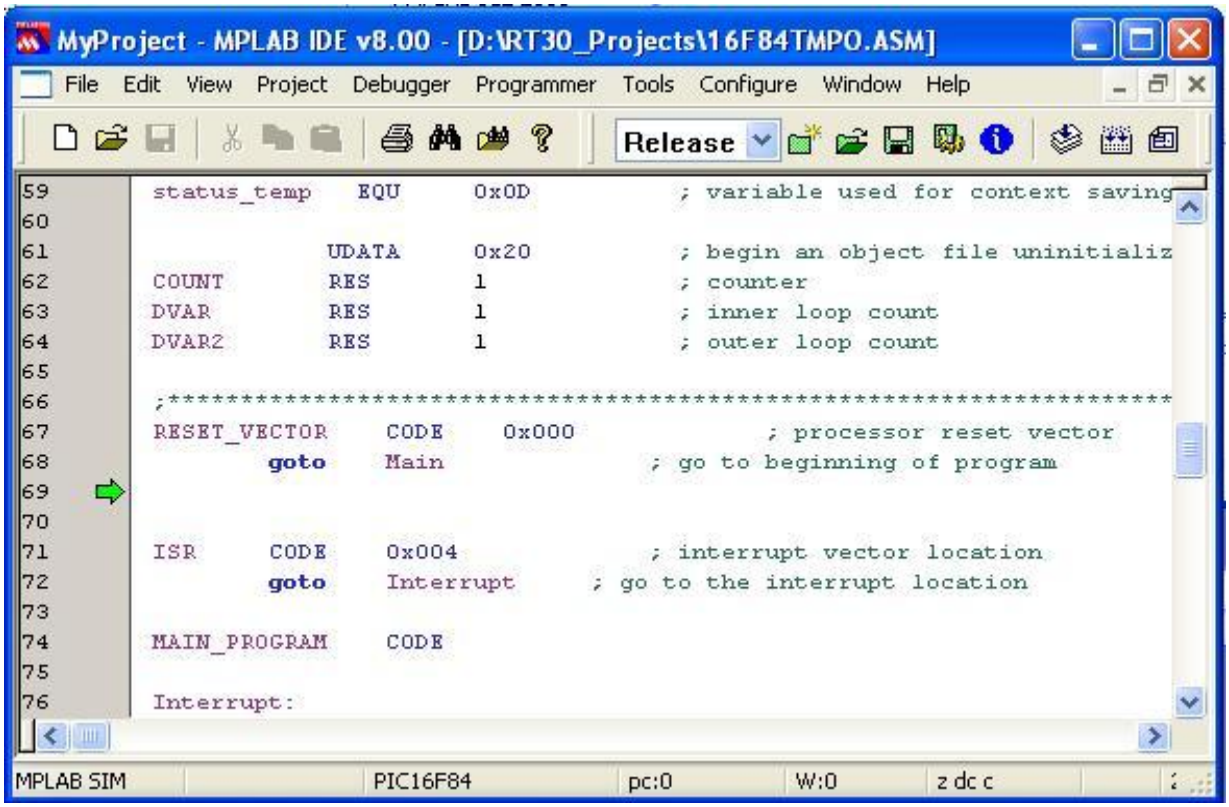


Рис. 1.17. Програма після виконання команди Processor Reset

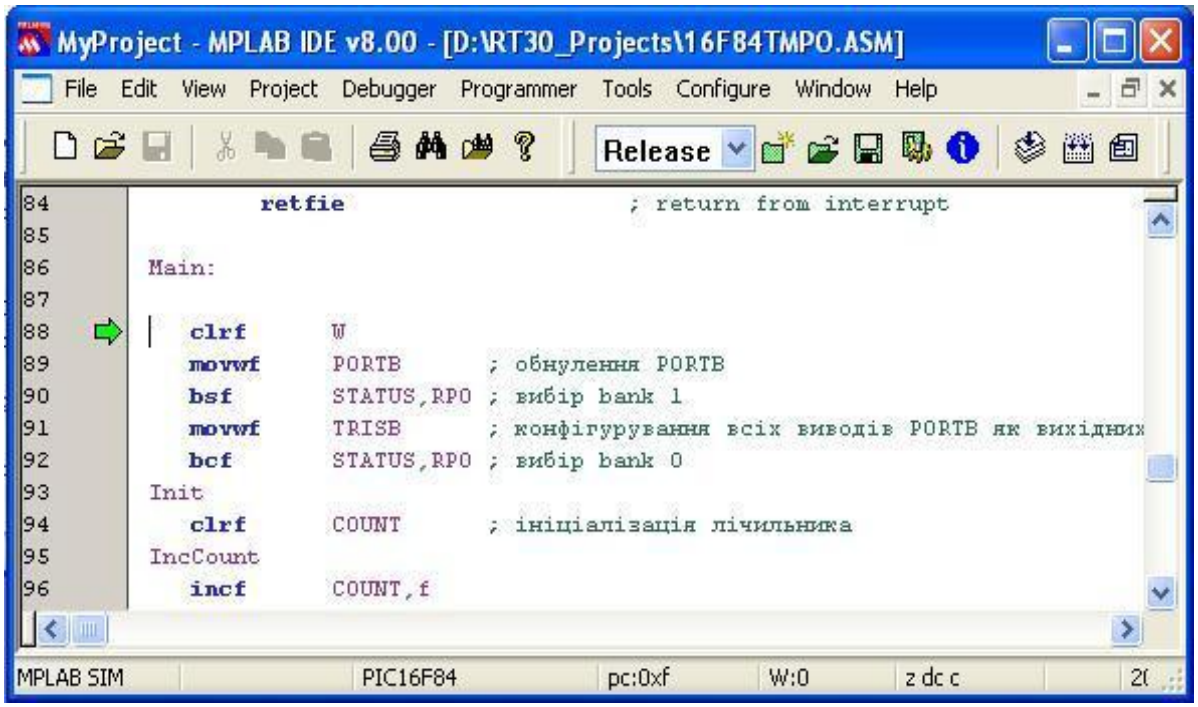


Рис. 1.18. Програма після виконання першого кроку (Step Into)

Натисніть піктограму Step Into, або виберіть *Debugger>Step Into* для виконання одного кроку коду з **`Main`** (рис. 1.18). Щоб переконатися, що програма працює так, як задано (тобто виводить збільшені значення через PORTB), контролюємо значення, які передаються в PORTB.

Вибираємо *View>Watch,* щоб відкрити пусте вікно Watch. В ньому зверху є два рядки з «випадаючими» списками (рис. 1.19). Вікно зліва (позначене "Add SFR") може бути використане для додавання регістру спеціальних функцій, PORTB, в область спостереження. Вибираємо PORTB із списку і натискуємо Add SFR, щоб добавити його у вікно. «Випадаючий» список праворуч дозволяє вибрати символи, які можуть бути добавлені з програми. Використаємо цей список, щоб добавити змінну COUNT у вікно Watch. Вибираємо COUNT із списку, а потім натискуємо Add Symbol, щоб добавити його у вікно (рис.1.20).

Примітка: Змінні, визначені за допомогою директиви EQU не можуть добавлятися у вікно Watch.

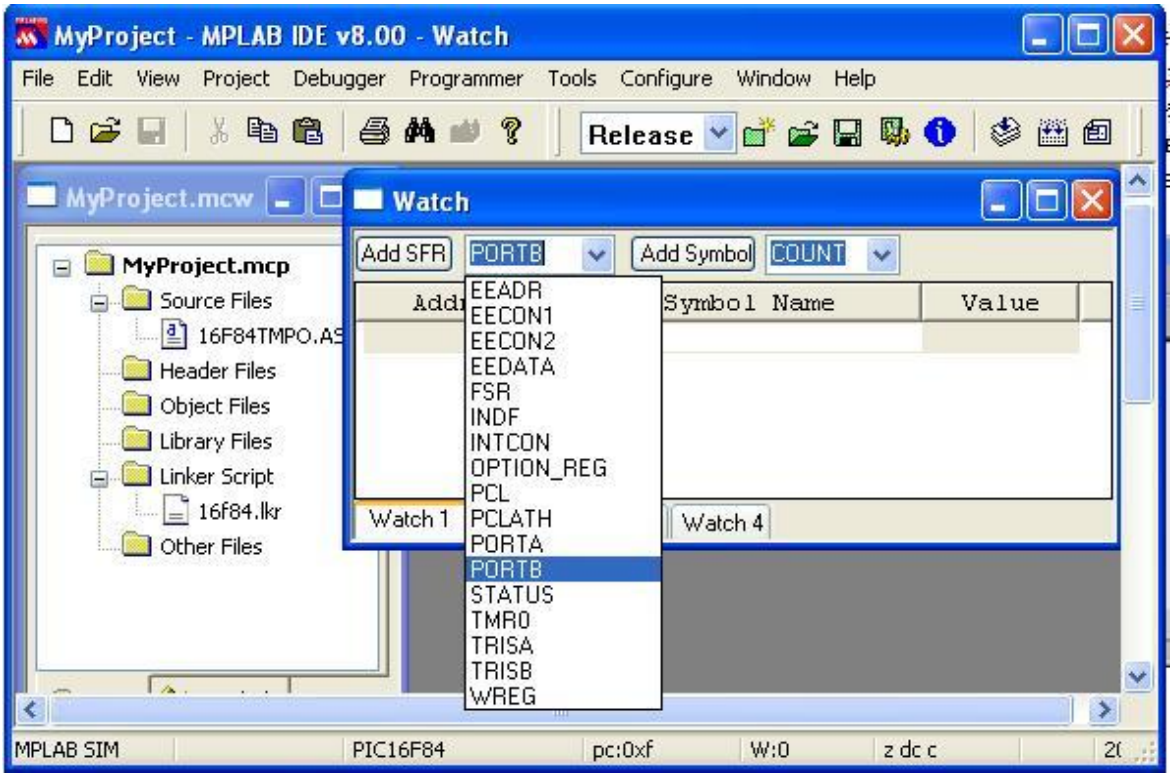


Рис. 1.19. Вибір PORTB

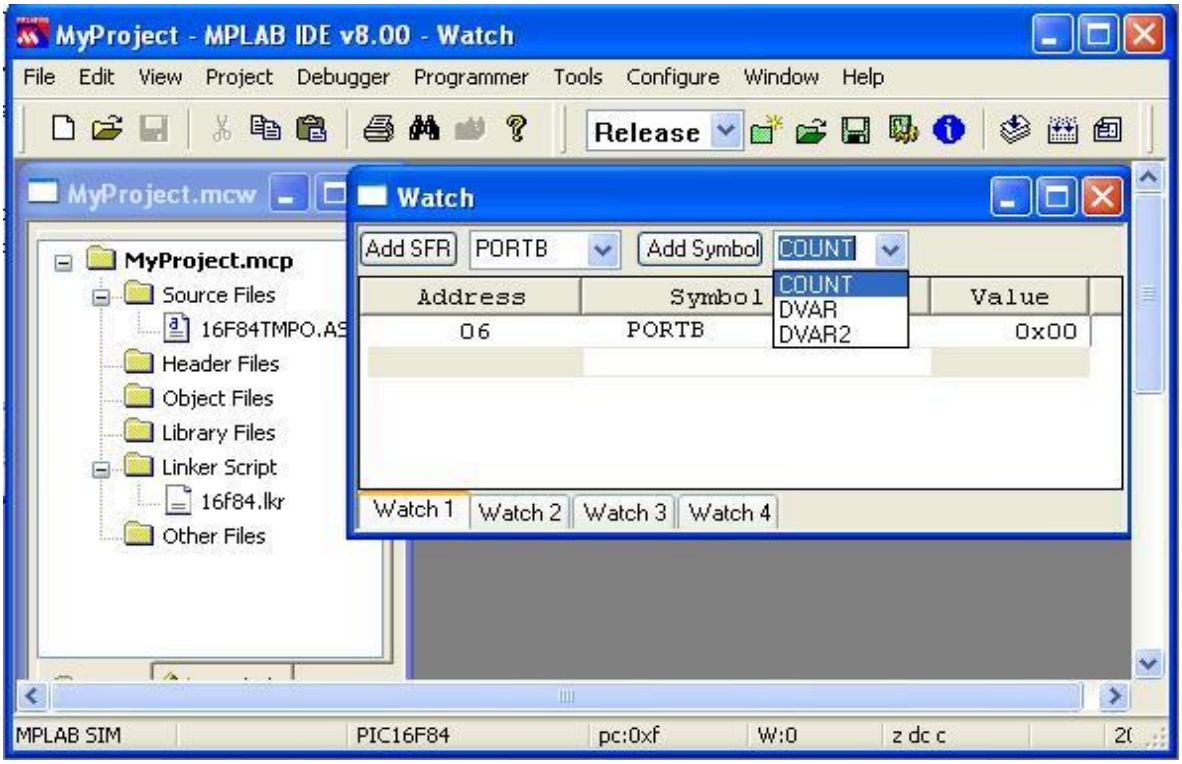


Рис. 1.20. Вибір змінної "COUNT"

Вікно Watch тепер буде показувати адреси, назви і значення двох регістрів (рис.1.21). В цій точці програми значення обох регістрів рівні нулю.

Примітка: Терміни можна добавити у вікно Watch простим перетягуванням символів із SFR, файлового регістру або з вікна редактора. Можна також «кликнути» мишкою прямо у вікні нижче імені символу і надрукувати необхідний термін

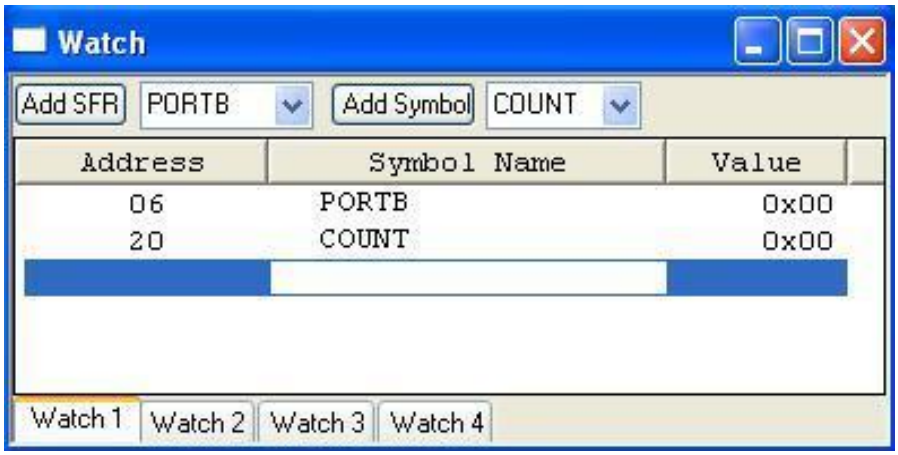


Рис. 1.21. Вікно Watch після виконання команди RESET

Можете продовжити «покрокове» виконання програми, але зручніше встановити точку контролю перед тим, як перше значення буде виведене в PORTB. Для встановлення точки контролю ставимо курсор на рядок, який відповідає рядку програми:

```
movwf PORTB  ;переміщення змінної COUNT на PORTB (display ;COUNT on
             ; PORTB)
```

і натискуємо праву клавішу миші (рис. 1.22).

Вибираємо Set Breakpoint із контекстного меню. З'являється червоне "B" в рядку коду (рис. 1.23).

Примітка: Точку контролю (зупинки) також можемо добавити подвійним кликом на рядку.

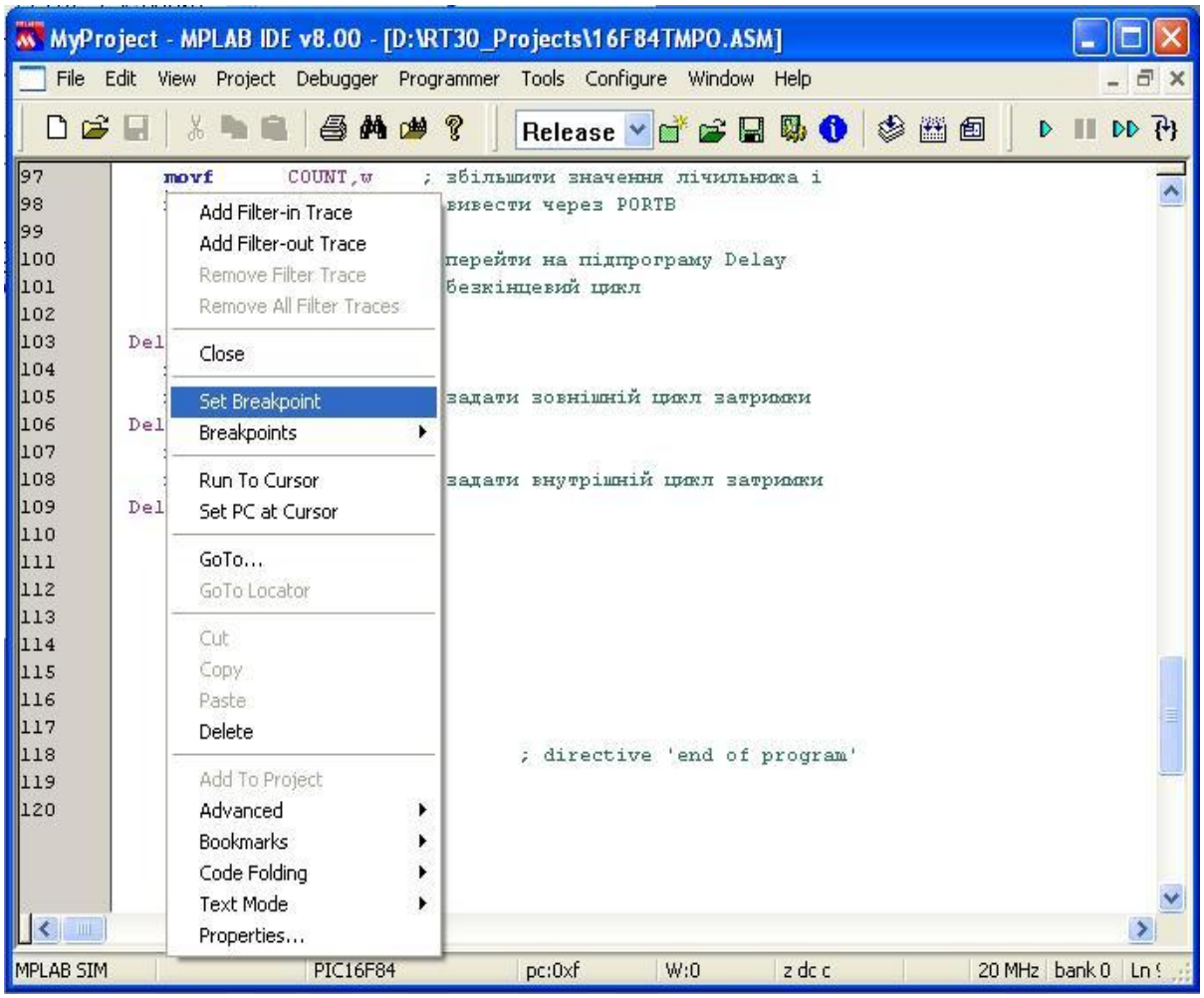


Рис.1.22. Настроювання контекстного меню

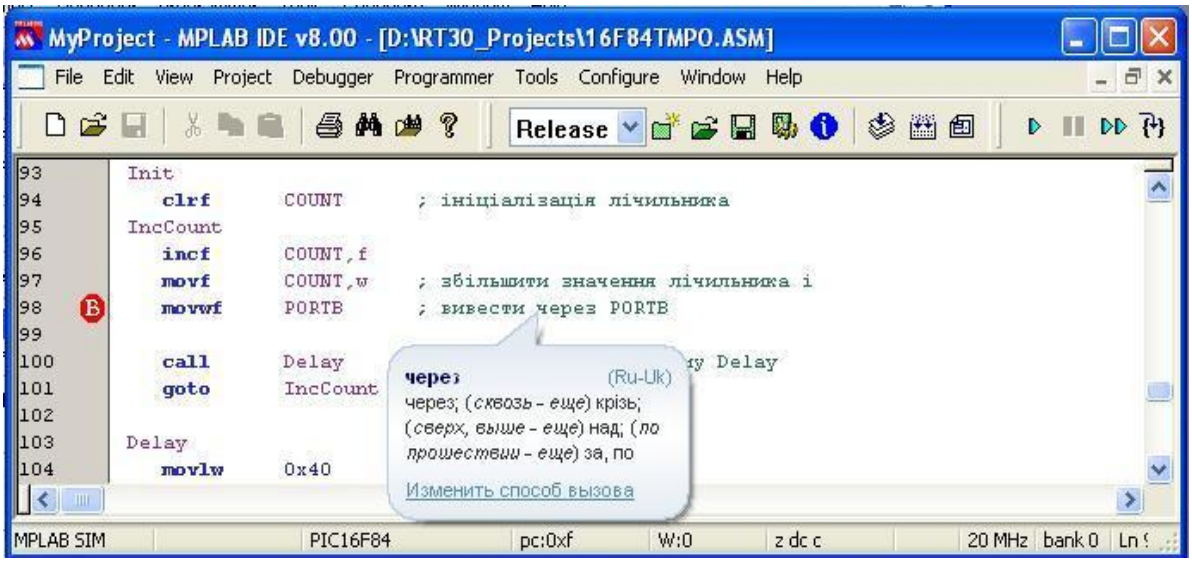


Рис. 1.23. Вікно редактора. Задавання точки контролю

Вибираємо *Debugger>Run* для запуску програми. На панелі статусу (нижній рядок вікна) на короткий час з'явиться текстове повідомлення "Running..." перш ніж програма зупиниться на першій контрольній точці.

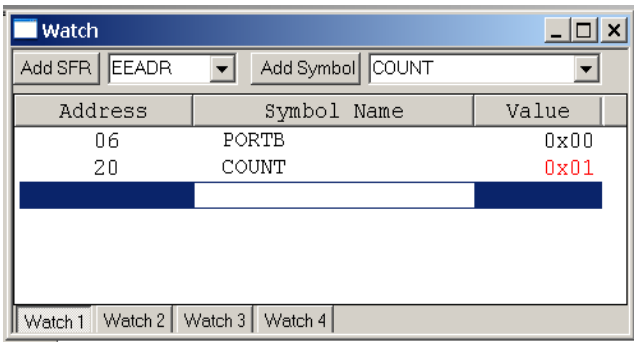


Рис. 1.24. Перша точка контролю

Вікно Watch покаже (рис.1.24), що змінна COUNT збільшилась на одиницю, але оскільки точка контролю вибрана в рядку перед виводом в PORTB, то значення PORTB все ще рівне нулю. Натискуємо піктограму Run для виконання програми до нової зупинки в контрольній точці. Вікно Watch тепер показує, що обидва значення збільшились на одиницю (рис.1.25).

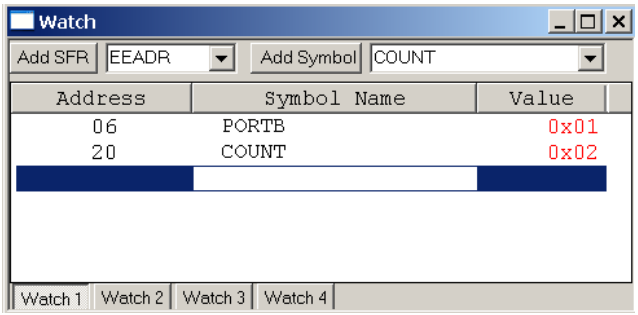


Рис. 1.25. Наступна точка контролю

Якщо ви спробуєте виконати команди крок за кроком в колі затримки, то вам доведеться пройти тисячі кроків, перш ніж досягнете кінця. Для виходу із кола затримки скористайтесь *Debugger>Step Out*.

Для розрахунку часу затримки скористайтесь довідниковими даними для мікроконтролера, де вказані тривалості кожної з команд. Зручно використати MPLAB IDE StopWatch для вимірювання часу затримки. Відстежимо, наприклад, час появи кожного нового значення COUNT. Якщо встановити точку контролю так, як було показано вище, на команді виводу COUNT в PORTB, то можна визвати наступну контрольну зупинку в цьому ж місці і таким чином виміряти час затримки для заданої тактової частоти мікроконтролера.

Частоту тактового сигналу мікроконтролера можна задати, відкривши в меню параметри «налагоджувача» *Debugger>Settings* (рис. 1.26).

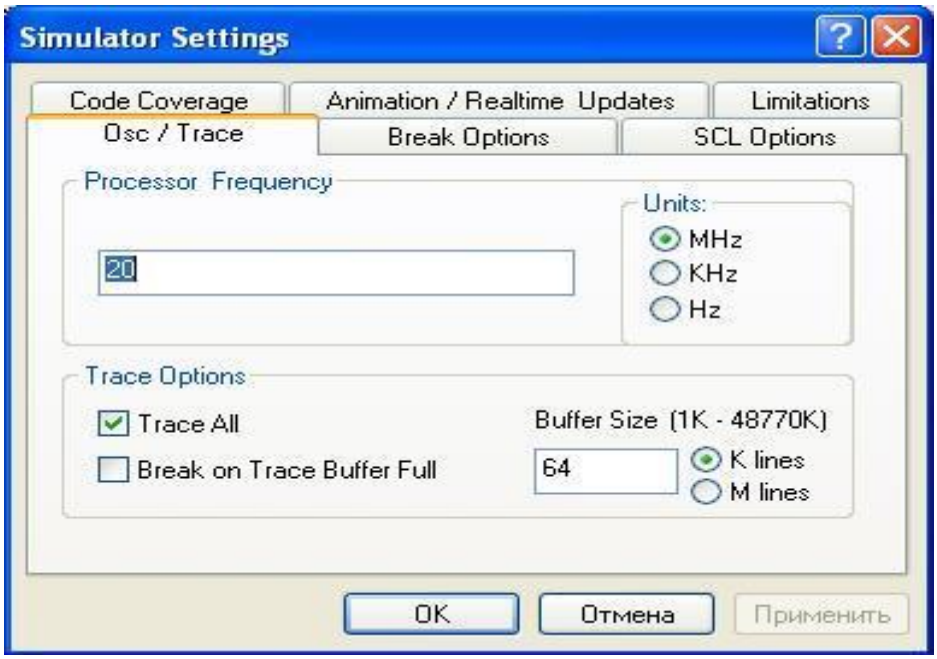


Рис. 1.26. Встановлення тактової частоти контролера

Використайте *Debugger>StopWatch* для відображення діалогового вікна StopWatch. Переконайтесь, що точка зупинки встановлена в рядку команди **`movwf COUNT`**, а потім натисніть *Debugger>Reset>Processor Reset* і далі *Debugger>Run* для зупинки програми на команді **`movwf COUNT`**. Для заданої за замовчуванням частоти контролера 20 MHz маємо тривалість машинного циклу (МЦ) рівну 1/20 *4=0,2 мкс. StopWatch покаже час 9,8 мс в момент першої зупинки в контрольній точці (рис. 1.27).

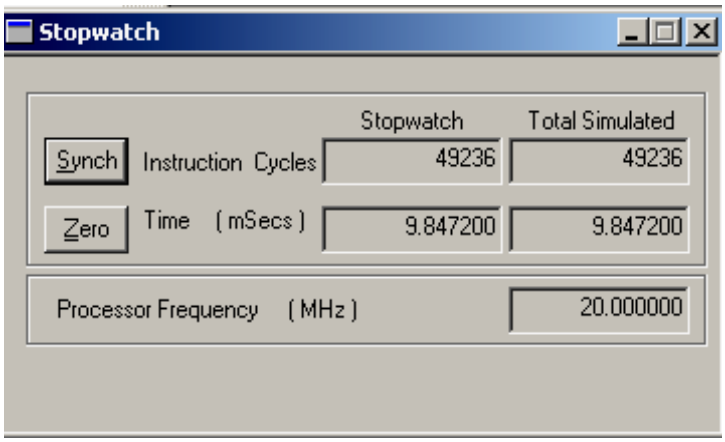


Рис. 1.27. Перша зупинка в контрольній точці

Запустивши Run знову і виконавши ще одне коло затримки, фіксуємо, що StopWatch показує вже приблизно 19,7 мс (рис. 1.28). Для зміни часу затримки необхідно змінити значення змінних в колі затримки.

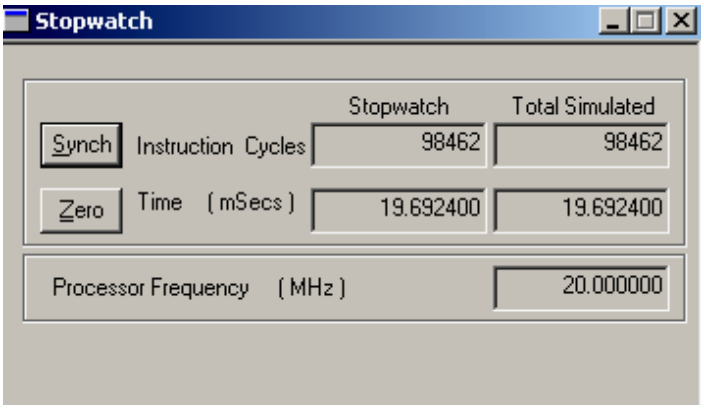


Рис.1.28. Зупинка після затримки

Порівнюючи значення вікон рис. 1.27 і рис. 1.28 можемо розрахувати реалізоване значення часу затримки в колі затримки.

1.3. Завдання і порядок виконання роботи

В лабораторній роботі слід створити, побудувати та виконати тестування простого проекту. Завдання виконується в такій послідовності:

- Вибрати тип мікроконтролера - PIC16F84.
- Запустити майстер проекту для створення проекту та:

  - вибрати вбудовані в MPLAB IDE мовні інструменти - асемблер MPASM і лінкер MPLINK;

  - добавити файли в проект: тимчасовий файл для вибраного пристрою і сценарій лінкування.
- Записати приведений в описі роботи фрагмент програми, яка виконує запис змінних значень в порт входу/виходу.
- Вивчити алгоритм роботи записаної програми.
- Побудувати проект.
- Тестувати програму за допомогою симулятора.
- Отримати у викладача індивідуальне завдання: тактову частоту (f_{CLK}) мікроконтролера та значення часу затримки ($T_{ЗАТР}$), яке необхідно реалізувати в програмі з точністю до 1 машинного циклу (МЦ).

Приклади варіантів завдань наведені в табл. 1.2.

Таблиця 1.2. **Приклади варіантів завдань**

№	f_{CLK}, МГц	$T_{ЗАТР}$, мс	№	f_{CLK}, МГц	$T_{ЗАТР}$, мс
1	2	40	7	8	32
2	3.66	20	8	10	0.8
3	4	10	9	11.66	12
4	4.33	16	10	12.33	0.5
5	5	9	11	16	0.125
6	6	8	12	20	3

- Розрахувати значення змінних в колі затримки для реалізації необхідної величини затримки.
- Ввести необхідні зміни в програму та перевірити правильність розрахунків за допомогою симулятора (відхилення від розрахункового значення не повинно перевищувати один машинний цикл).
- Оформити в текстовому редакторі протокол, в якому привести розрахунок затримки і зображення вікон симулятора з підтвердженням результатів розрахунку.

Контрольні питання

1. Поясність алгоритм роботи програми формування часової затримки.
2. Поясніть використання шаблонів MPLAB при створені проекту.
3. Поясніть як виконується команда **`decfsz`**. Які команди подібні до приведеної?
4. Що означають директиви
   #include, __CONFIG, EQU, UDATA, CODE.
5. Охарактеризуйте можливості мікроконтролера PIC16F84.
6. Поясніть призначення регістра результату виконання арифметичних операцій STATUS.

7. Якими способами можна сформувати часову затримку на мікроконтролері?

Література до лабораторної роботи № 1

1. Руководство пользователя компилятора MPASM [стр. 5-10, 14-39] *http://www.microchip.ru:/files/d-sheets-rus/mpasm.pdf*

2. Руководство пользователя MPLAB IDE [стр. 13-50] *http://www.microchip.ru:/files/d-sheets-rus/mplab_ide.pdf*

3. Справочник по всем микроконтроллерам PIC16 [разделы 4, 6, 29]. *http://www.microchip.ru:/files/d-sheets-rus/PIC16_Manual.pdf*

ЛАБОРАТОРНА РОБОТА № 2

Формування сигналів заданої тривалості та періоду на виходах мікроконтролера

2.1. Мета роботи

Вивчити використання таймера та системи переривань для формування на виводах портів сигналів заданої тривалості та періоду. Ознайомитись з можливостями MPLAB для трасування пам'яті при симуляції виконання програми. Дослідити зміну значень регістрів (портів) в часі за допомогою логічного аналізатора.

2.2. Основні теоретичні відомості

2.2.1. Створення проекту для виводу заданих сигналів

Для прикладу розглянемо програму формування на виводі 0 порту B послідовності імпульсів високого рівня тривалістю 50 мс. Період імпульсів приблизно 250 мс. Тактова частота мікроконтролера 20 МГц. Використаємо систему переривання таймера TMR0 по переповненню для формування затримки 1 мс.

Створимо проект MyProject_2 для мікроконтролера PIC16F84 на основі шаблону, аналогічно лабораторній роботі №1.

Добавимо у відповідному місці шаблону визначення змінної COUNT, зарезервувавши в області регістрів загального призначення 1 байт:

```
;***** VARIABLE DEFINITIONS (визначення змінних)
  w_temp       EQU  0x0C ; змінні, що використовуються для зберігання
  status_temp  EQU 0x0D ; змінні, що використовуються для зберігання
  UDATA    0x20
  count     RES    1
```

Для використання системи переривань добавимо в розділ для оброблення переривань звертання до підпрограми **ServiceInterrupts**:

```
Interrupt:
   movwf  w_temp
   movf   STATUS,w
   movwf  status_temp
   call  ServiceInterrupts ; визивання підпрогр. обробл. переривань
   movf   status_temp,w
   movwf  STATUS
   swapf  w_temp,f
   swapf  w_temp,w
   retfie
```

Після мітки **Main** додамо основний текст програми:

```
Main:
   call   InitPort      ; підпрограма ініціалізації порту B
   call   InitTimer     ; підпрограма ініціалізації таймера затримки
Loop:
   Goto Loop            ; нескінчений цикл
InitPort:               ; підпрограма ініціалізації порту B
   bsf  STATUS,RP0      ; банк 1
   clrf TRISB           ; всі виводи порту B на вихід
   bcf  STATUS,RP0      ; банк 0
   clrf   PORTB         ; обнуління порту B
   return
InitTimer:              ; Ініціалізації TMR0 для формування затримки
   bsf    STATUS,RP0    ; банк 1
   movlw  B'10000100'   ; попередній подільник приєднати до TMR0
   movwf OPTION_REG     ; задати коефіцієнт ділення 32
```

```
   bcf     STATUS,RP0    ; банк 0
   movlw   B'10100000'  ; дозвіл переривання TMR0
   movwf INTCON
   movlw .100     ; попереднє встановлення TMR0
   movwf TMR0     ; формування затримки 1 мс для частоти 20 МГц
   retfie
ServiceInterrupts:      ; підпрограма оброблення переривань
   btfsc INTCON,T0IF   ; перевірка прапора переривань TMR0
   goto ServiceTMR0     ; перехід на оброблення переривання від TMR0
   movlw B'10100000'   ; відновлення дозволу переривання TMR0
   movwf INTCON
   retfie        ; повернення без оброблення переривання
ServiceTMR0:     ; повторний запуск (перезапуск) таймера
   movlw .100   ; відновлення початкового значення TMR0
   movwf TMR0
   bcf INTCON,T0IF ; скидання прапора переривання TMR0
   call IncCounter ; виклик підпрограми формування тривалості
   retfie
IncCounter:      ;формування тривалості і періоду імпульсів
   incf count,w ;збереження збільшеного знач. лічильника в регістрі WREG
   xorlw  .50     ; перевірка тривалості імпульсу
   btfsc  STATUS,Z ;
   bcf    PORTB,0   ;  0, якщо тривалість імпульсу досягнута
   incf   count,w   ; збереження збільшеного знач. лічильника в регістрі W
   xorlw  .200      ; перевірка величини періоду
   btfss  STATUS,Z ;
   goto Inc         ; перехід, якщо не кінець періоду
   bsf  PORTB,0     ; 1, якщо початок нового періоду
```

```
   movlw   0xFF        ; обнуління лічильника на наступному кроці
   movwf   count       ;
Inc:
   Incf    count,f     ; збільшення значення лічильника
   return
   end                 ; кінець програми (end of program)
```

При наборі програми в текстовому редакторі коментарі вводити не обов'язково.

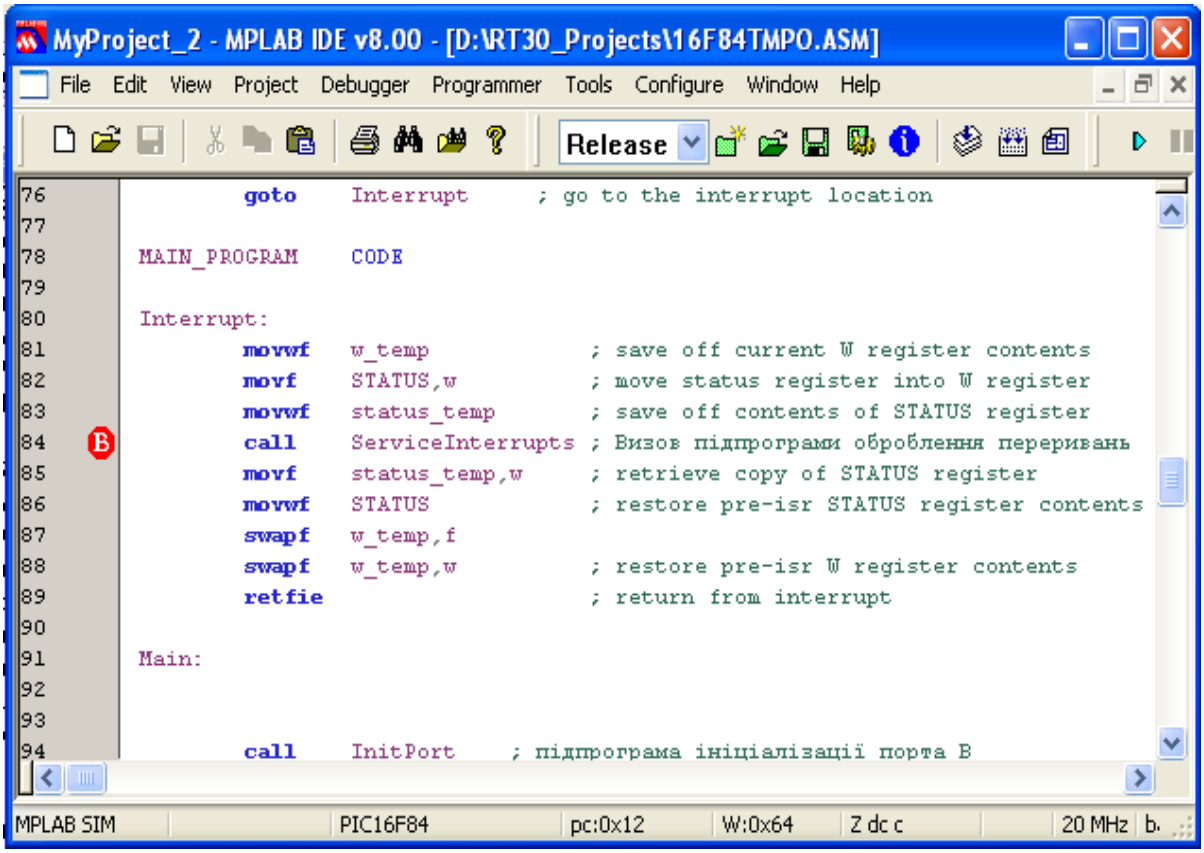


Рис. 2.1. Встановлення точки контролю для визначення часу затримки

Уважно розгляньте коментарі, щоб зрозуміти роботу програми. В ній за допомогою таймера TMR0 формується затримка тривалістю 1 мс. Для цього використовується система переривання по переповненню даного таймера. А час до переповнення розраховується через частоту мікроконтролера і встановлюється записом початкового значення в регістр таймера та задаю-

чи коефіцієнт попереднього ділення. Необхідне значення тривалості імпульсу і його період повторення формуються із необхідного числа запусків TMR0, кожен з яких має тривалість 1 мс.

Після налагодження програми перевірте реалізований час затримки таймера, використовуючи контрольну точку (рис. 2.1) і вікна *View>Watch* та *Debugger>StopWatch* (див. лабораторну роботу №1).

2.2.2. Аналіз вихідних сигналів за допомогою логічного аналізатора

Вікно логічного аналізатора (Logic Analyzer) працює подібно до реального логічного аналізатора, показуючи цифрові рівні всіх вибраних виводів за заданий період. Дані відображаються у відповідності до буферу трасування. Якщо відповідні дані відсутні в буфері трасування, то вони не можуть відображатися і у вікні аналізатора.

Вікно трасування пам'яті (Trace Memory Window)

Розглянемо роботу вікна трасування пам'яті, щоб зрозуміти, як здійснюється захват даних в буфер трасування.

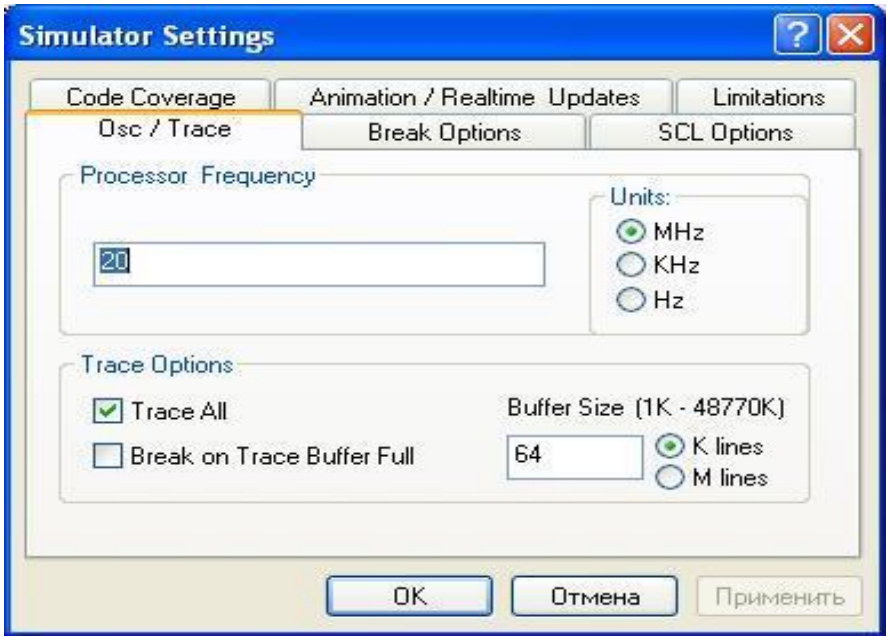


Рис. 2.2. Встановлення опції трасування програми

Вікно трасування пам’яті (*View>Simulator Trace*) допомагає виконати моніторинг багатьох операцій мікроконтролера. В кожному циклі фіксуються дані всіх портів. Для кожного порту об’єм буфера становить 64K.

Для виконання трасування всіх рядків програми, які виконуються, поставте «галочку» Trace All (рис. 2.2) (*Debugger>Settings*)

Щоб окремі вибрані рядки програми добавити (In) в буфер трасування, або видалити (Out) з нього, використайте опції Filter-In Trace або Filter-Out із контекстного меню (рис. 2.3 – «клік» правою клавішею миші на тексті програми), попередньо виділивши необхідний рядок. В прикладі (рис. 2.3) в буфер трасування добавлені рядки виводу значень 0 і 1 в PORTB, про що сигналізують маленькі квадрати на лівому сірому полі діалогового вікна.

Рекомендація: Обмежте число рядків програми для захвату у буфер трасування лише тими, які необхідні для роботи логічного аналізатора.

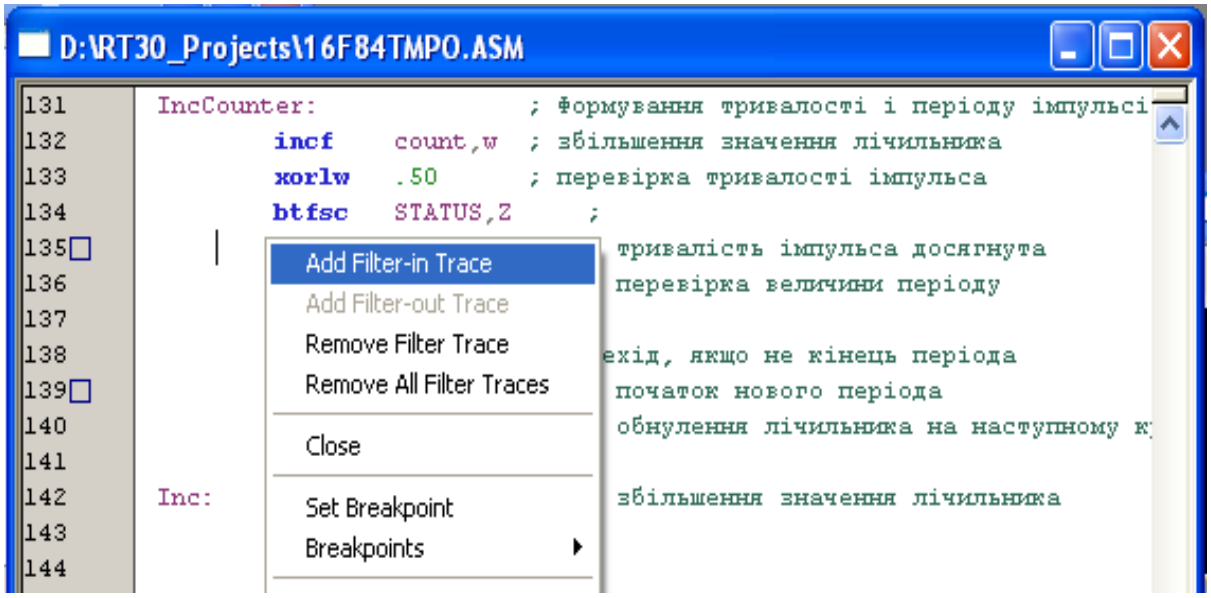


Рис. 2.3. Контекстне меню вибору окремих рядків програми трасування

У вікні трасування пам’яті може бути відображено до 32767 команд. Програму першоджерела можна відкрити лише для читання в нижній час-

тині вікна, якщо у вікні трасування пам’яті вибрати із контекстного меню параметр “Show Source” (рис. 2.4).

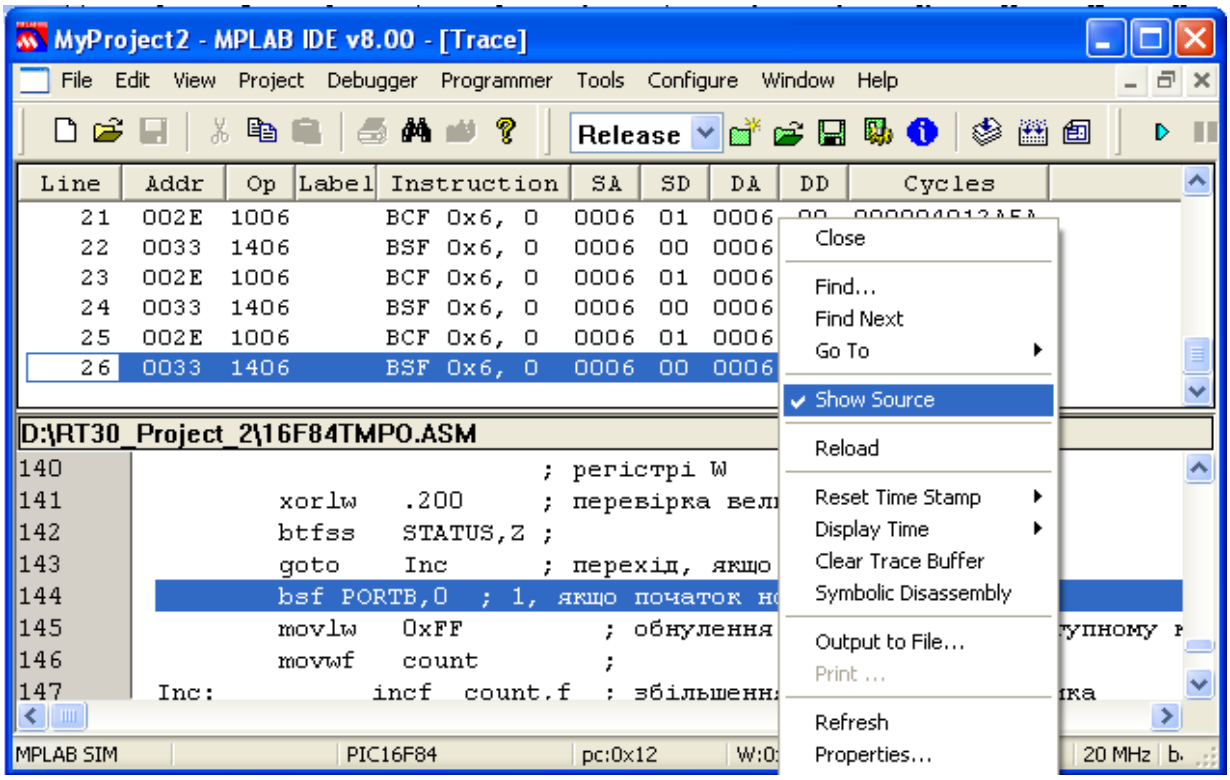


Рис. 2.4. Вікно трасування пам’яті

Відображення вікна трасування пам’яті

Вікно трасування пам’яті може залишатись весь час відкритим, переміщуватись, або бути схованим. Вікно має такі інформаційні поля для кожного циклу виконання:

- (**Line**) Номер рядка – Позиція в циклах відносно події або точки зупинки.
- (**Addr**) Адрес – Адрес команди в пам’яті програми.
- (**Op**) Код операції – Машинний код команди.
- (**Label**) Мітка – Мітка у відповідності із адресом пам’яті програми.
- (**Instruction**) Команда – «Дизасемблерована» команда.
- (**SA**) Адрес джерела даних – Адрес або символ джерела даних.

- (**SD**) Значення джерела даних – Значення даних джерела.
- (**DA**) Адрес місця призначення даних – Адрес або символ місця призначення даних.
- (**DD**) Значення вихідних даних – Значення вихідних даних.
- (**Cycles/Time**) Фіксація часу – Визначення (захват) часу в циклах або секундах.

Цикл на момент захвату нумерується, як цикл 0. Всі інші цикли будуть пронумеровані відносно нього. Цикли, які відбулися до моменту захвату матимуть від'ємне значення, а цикли після точки захвату – додатній номер.

Дисплей логічного аналізатора

Дисплей логічного аналізатора призначений для стеження за сигналами на портах або шинах мікроконтролера. Якщо буфер трасування пустий, то з'явиться вікно із словами "No items to display" (рис. 2.5). В іншому випадку буде показаний вміст буфера трасування. Виберіть пункт Channels, щоб відкрити діалогове вікно конфігурування каналів (рис. 2.6). Виберіть вивід і/або сигнали шини, які будуть використані у вікні логічного аналізатора**.** Сигнали на виводі показують зміни окремих виводів. Сигнали шини показують зміни (в HEX коді) для багатьох виводів. Кожен сигнал має свій власний канал на дисплеї логічного аналізатора.

Керування вікном (рис. 2.5).

- (**Trigger Position**) Позиція фіксації – Три перемикачі дозволяють вибрати, коли відбувається фіксація: при старті захвату, в середині захвату, або в кінці захвату.
- (**Trigger PC**) = Фіксація PC (program counter - програмного лічильника) – Є два методи фіксації захвату даних:
  - **Trigger PC** = *значення* – Фіксація на значенні, вказаному у текстовому вікні.
  - **Trigger Now** – Фіксація в момент натискання даної клавіші. Значен-

ня PC розміщується в текстовому вікні.

- **Trigger Clear** – «Обнуляються» значення фіксації і дані буфера для нового захвату. Дані буфера очищаються, якщо дисплей закривається.

- (**Time Base**) Базовий час - Базовий час задається у циклах, секундах, мілісекундах і наносекундах.

- (**Mode**) Режим – Відображається вибраний режим захвату, тобто одиночний або комплексний.

- (**Channels**) Канали – Натисніть кнопку для добавлення або видалення сигналів/каналів із вікна дисплея.

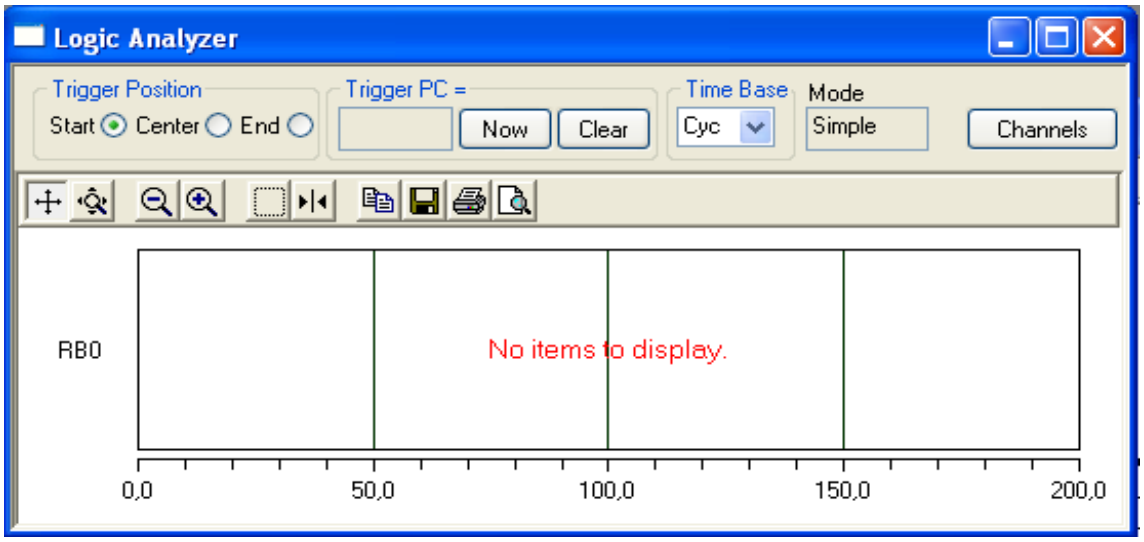


Рис. 2.5. Вікно логічного аналізатора з відмітками проміжку часу в МЦ

Додавання сигналів до каналу:

- Виберіть тип сигналу з «випадаючого» списку вікна: вивід, шина або одне і друге разом. Шини можуть бути створені натисканням Configure Bus(es).

- Для вибору сигналу натисніть на нього в стовпцю "Available Signals". Для вибору більш, ніж одного сигналу одночасно, вказану операцію слід виконати утримуючи клавіші Shift або Ctrl.

- «Кликніть» Add=>, щоб добавити сигнали в стовпець "Selected Signals".

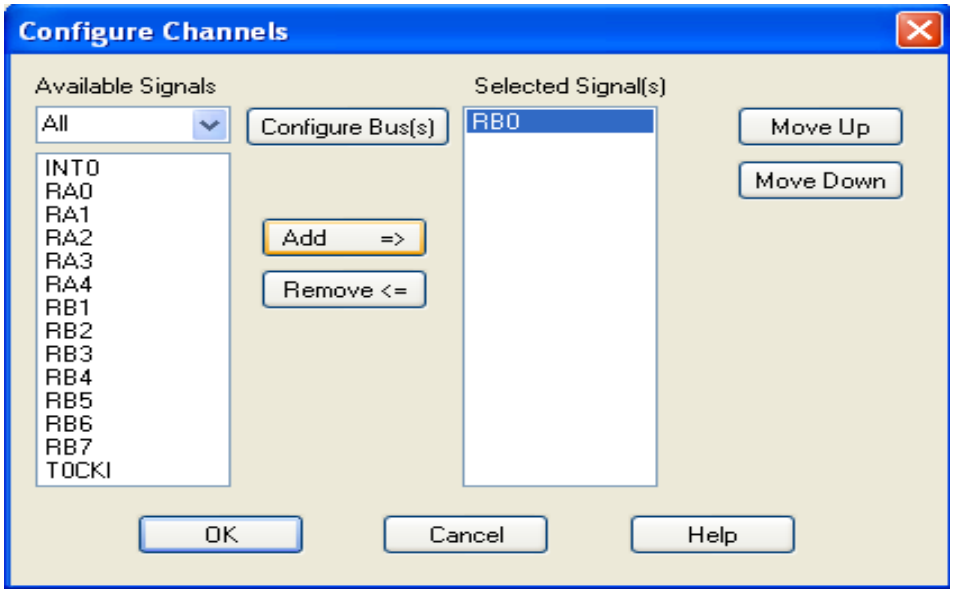


Рис. 2.6. Вибір каналів для відображення у вікні аналізатора

Видалення сигналів з каналу:

- «Кликніть» на сигналі в стовпцю "Selected Signals" для його вибору. Для вибору більш, ніж одного сигналу одночасно, вказану операцію слід виконати утримуючи клавіші Shift або Ctrl.
- «Кликніть» Remove<= щоб видалити сигнал із стовпця "Selected Signals" .

Зміна порядку чередування каналів:

- «Кликніть» мишкою на сигналі/каналі в стовпцю "Selected Signals".
- Скористайтесь кнопками Move Up або Move Down для переміщення вибраного в листі.

Дисплей може відображати максимум 64 канали. Дані показуються дискретним рівнем 0/1, відповідно до стану кожного біта.

Відображення сигналу/каналу:

- Перевірте чи добавленні відповідні рядки програми в буфер трасування (рис. 2.3);
- Запустіть виконання програми командою Run, для накопичення даних в буфері;

- Зупиніть виконання програми командою Halt. На дисплеї логічного аналізатору мають відображатись дані накопичені в буфері.

Якщо програма не виконується в режимі Run повноцінно, наприклад, зупиняється, то дані не зможуть накопичуватись в буфері і відображатися на дисплеї. На рис. 2.7 показано вікно попередження, яке виникає при зупинці симуляції в режимі Run. Дане попередження інформує, що в настройках симуляції ввімкнений сторожовий таймер. Оскільки дана програма не передбачає використання сторожового таймеру, його необхідно вимкнути в меню симулятора *Configure>Configuration Bits*…(рис. 2.8).

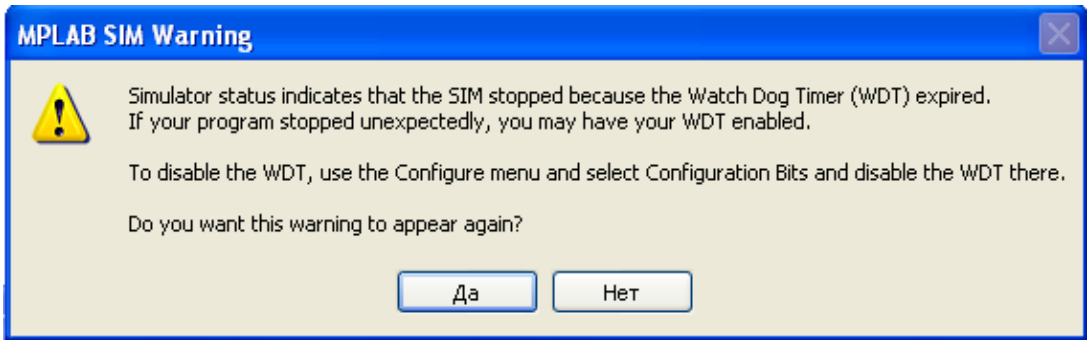


Рис. 2.7. Попередження симулятора про роботу таймеру WDT

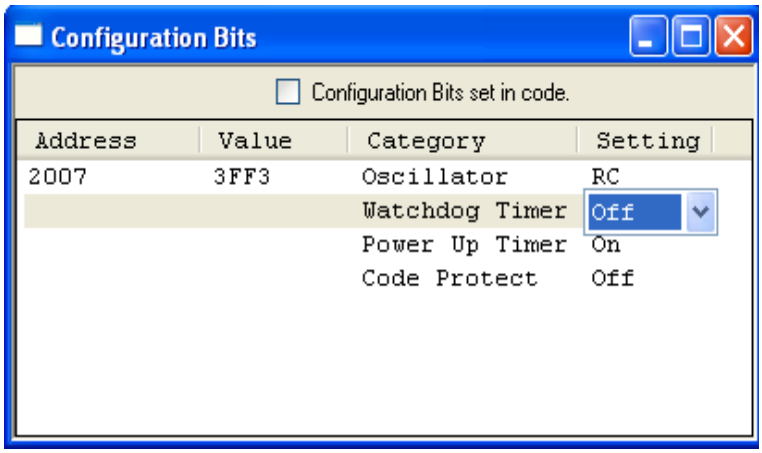


Рис. 2.8. Меню визначення бітів конфігурації

Меню логічного аналізатора («клик» правої клавіші миші у вікні аналізатора):

- (**Close**) – Закрити діалог.

- (**Zoom To Fit**) – Розтягнути або стиснути до часового періоду, який вміщується на екрані.
- (**Go To**) – Ввести час, до якого «перекрутити» центр дисплею.
- (**Refresh**) – Добавити на дисплей інформацію нинішнього трасування.
- (**Properties**) – Задати параметри вікна.

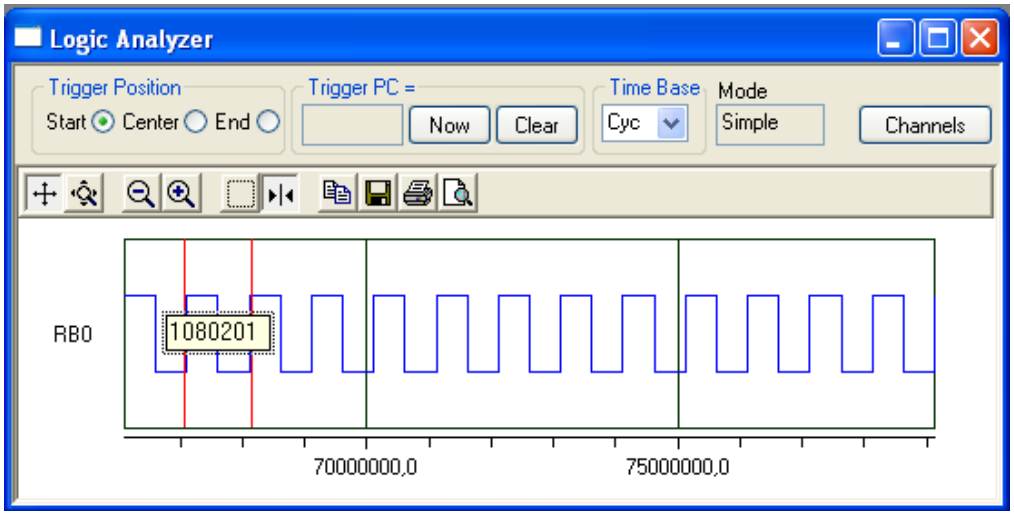


Рис. 2.9. Вікно логічного аналізатора з відображенням

2.3. Завдання і порядок виконання роботи

В лабораторній роботі треба створити, побудувати та виконати тестування на логічному аналізаторі проекту. Завдання виконується в такій послідовності:

- Вибрати тип мікроконтролера - PIC16F84.
- Запустити майстер проекту і створити проект.
- Записати приведений в описі роботи простий код, в якому виконується виведення в порт входу/виходу імпульсу сигналу заданої тривалості і періоду.
- Вивчити за допомогою коментарів алгоритм роботи записаного коду.

- Скомпілювати проект.
- Тестувати код за допомогою симулятора.

Отримати у викладача індивідуальне завдання: тактову частоту мікроконтролера та значення часу для формування на виходах PORTB трьох різних сигналів.

Завдання. Сформуйте на виході PORTB сигнал з заданими тривалістю імпульсу і періодом з точністю до 1 МЦ з формуванням переривання по переповненню таймера.

Приклади варіантів завдань наведені в табл. 2.1.

***Таблиця 2.1*. Приклади варіантів завдань**

№	f_{CLK}, *МГц*	Тривалість імпульсу, мс	Тривалість періоду , *мс*	№	f_{CLK}, *МГц*	Тривалість імпульсу, мс	Тривалість періоду , *мс*
1	2	50	100	7	8	80	320
2	3.66	100	200	8	10	0.5	2
3	4	125	250	9	11.66	100	120
4	4.33	60	160	10	12.33	150	200
5	5	70	130	11	16	300	500
6	6	90	180	12	20	150	300

- Розрахувати значення змінних, необхідних для реалізації заданих тривалості і періоду сигналів.
- Ввести необхідні зміни в програми та перевірити правильність розрахунків за допомогою симулятора (трасування пам’яті та вивід на дисплей логічного аналізатора).
- Оформити в текстовому редакторі протокол, в якому привести розрахунок затримки і зображення вікон логічного аналізатора з підтвердженням результатів розрахунку.

Контрольні питання

1. Проаналізуйте режими роботи таймера TMR0.
2. Ініціалізація портів мікроконтролера. Ініціалізація таймеру TMR0.
3. Обробка переривання при переповненні таймеру TMR0.
4. Навіщо при зверненні до регістрів TRISB, OPTION_REG необхідно програмно змінювати банк пам'яті даних?
5. Проаналізуйте можливості логічного аналізатора симулятора.
6. Охарактеризуйте вікно трасування пам'яті.

Література до лабораторної роботи № 2

1. Справочник по всем микроконтроллерам PIC16 [разделы 6, 8, 9, 11].
http://www.microchip.ru:/files/d-sheets-rus/PIC16_Manual.pdf
2. Руководство пользователя MPLAB IDE [стр. 55-62]
http://www.microchip.ru:/files/d-sheets-rus/mplab_ide.pdf

ЛАБОРАТОРНА РОБОТА № 3

Введення зовнішніх сигналів мікроконтролера і перевірка роботи програми за допомогою модуля одноразових впливів (стимулу)

3.1. Мета роботи

Вивчити систему введення зовнішніх сигналів мікроконтролера. Ознайомитись з можливостями використання симулятора MPLAB для перевірки програми введення зовнішніх сигналів. Виконати моделювання системи приймання даних від зовнішнього джерела з одночасним використанням трьох типів сигналів: інформаційних, тактування та дозволу.

3.2. Основні теоретичні відомості

3.2.1. Створення проекту для вводу даних

Для прикладу розглянемо програму, в якій при виявленні синхроімпульсу на вході зовнішнього переривання здійснюється введення даних, які поступають на вхід RB1. Використаємо систему переривання для зовнішнього переривання. Для перевірки роботи програми скористаємось можливостями модуля «стимул» (stimulus). З його допомогою сформуємо на виводі INT (RB0) послідовність синхроімпульсів високого рівня тривалістю 20 машинних циклів (МЦ) і періодом 50 МЦ, а також симулюємо послідовність зовнішніх даних на виводі RB1. Прийнята частота мікроконтролера 20 МГц.

Створимо проект MyProject_3 для мікроконтролера PIC16F84 на основі шаблону, аналогічно лабораторній роботі №1.

Добавимо у відповідному місці шаблону визначення регістра RB_Temp для збереження введених даних:

```
;***** VARIABLE DEFINITIONS
w_temp EQU 0x0C ; змінні для збереження (variable used for context saving)
status_temp  EQU 0x0D ;  variable used for context saving
UDATA     0x20
RB_Temp       RES  1    ; регістр для збереження введених даних
```

Для використання системи переривань добавимо в розділ для оброблення переривань звертання до підпрограми **ServiceInterrupts:**

```
Interrupt:
   movwf  w_temp
   movf   STATUS,w
   movwf  status_temp
   call   ServiceInterrupts ; виклик підпрограми обробл. переривань
   movf   status_temp,w
   movwf  STATUS
   swapf  w_temp,f
   swapf  w_temp,w
   retfie
```

Після мітки **Main** додамо основний текст програми:

```
Main:
   call InitPort          ; підпрограма ініціалізації PORTB
   call InitINT           ; підпрограма ініціалізації переривання
Loop:
   goto Loop              ; нескінченний цикл
InitPort:                 ; підпрограма ініціалізації PORTB
   bsf    STATUS,RP0      ; банк 1
   movlw  B'00000011'     ;
   movwf  TRISB           ; виводи портів RB0 і RB1 на вхід
   bcf    STATUS,RP0      ; банк 0
```

```
   clrf    PORTB            ; обнуління PORTB
   return
InitINT:            ; ініціалізації системи зовнішнього переривання
   movlw   B'10010000'  ; дозвіл переривання INT
   movwf INTCON
   retfie
ServiceInterrupts:     ; підпрограма оброблення переривань
   btfsc   INTCON,INTF  ; перевірка прапора переривань INT
   goto ServiceINT      ; перехід на оброблення переривання від INT
   movlw B'10010000'    ; відновлення дозволу переривання INT
   movwf INTCON
   retfie               ; повернення без оброблення переривання
ServiceINT:             ; зчитування і збереження даних
   bcf  INTCON,INTF     ; скидання прапора переривання INT
   movf   PORTB,w       ; зчитування порту B
   andlw 0x02           ; маскування розрядів
   movwf RB_Temp        ; збереження вхідних даних
   retfie
   end                  ; кінець програми
```

Уважно розгляньте коментарі, щоб зрозуміти роботу програми.

Під час виконання симуляції наведеної вище програми необхідно вводити різні зовнішні дані. Модуль стимулу (далі стимул) системи симуляції дозволяє змінити рівень, або сформувати імпульс на виводах входу/виходу порту. Можна також змінити значення регістрів спеціальних функцій (SFR), або інші дані в пам’яті.

Стимул може бути корисним, коли перевіряються певні умови; наприклад, коли програма, що виконується, досягне конкретного адреса команди під час симуляції.

Стимул має два базові режими роботи:

- *Синхронний* – Наперед задана серія сигналів змінюється на виводах входу/виходу, SFR та ін. із визначеною циклічністю.
- *Асинхронний* – Одноразова зміна на виводах входу/виходу.

Щоб задати параметри зовнішніх впливів (тобто визначити *що*, *коли* і *як* буде подаватись на мікроконтролер зовні) під час виконання програми, потрібно скористатись діалоговим вікном стимулу для створення у робочій книзі синхронного і/або асинхронного стимулу.

Досвідчені користувачі можуть створити і приєднати файл SCL, щоб задати необхідний режим стимулу.

Примітка: Діалог Stimulus під час симуляції повинен залишатися відкритим, щоб стимуляція була активною

3.2.2. Діалогове вікно модулю стимулу

Для синхронного стимулу використовуємо 5 перших закладок вікна (рис. 3.1), щоб створити стимул з використанням мови Simulator Control Language (SCL). Якщо ж бажаємо використати лише асинхронний стимул, то можна перейти безпосередньо до закладки (Asynch).

Діалогове вікно дозволяє ввести інформацію стимулу, яка зберігається у файлі, названим робочою книгою (workbook). Для відкриття нової книги виберіть *Debugger>Stimulus>New Workbook.*

Щоб працювати з раніше створеною робочою книгою, виберіть *Debugger>Stimulus>OpenWorkbook.*

Діалогове вікно має закладки:

- Активність виводу/регістру (**Pin/Register Actions**)
- Розширені настройки виводу/регістра (**Advanced Pin/Register**)
- Стимул синхронізації (**Clock Stimulus**)
- Зміна значення регістра (**Register Injection**)

- Трасування регістра (**Register Trace**)
- Асинхронна стимуляція (**Asynch**)

Всі закладки мають одні й ті ж самі значення одиниць вимірювань по замовчуванню.

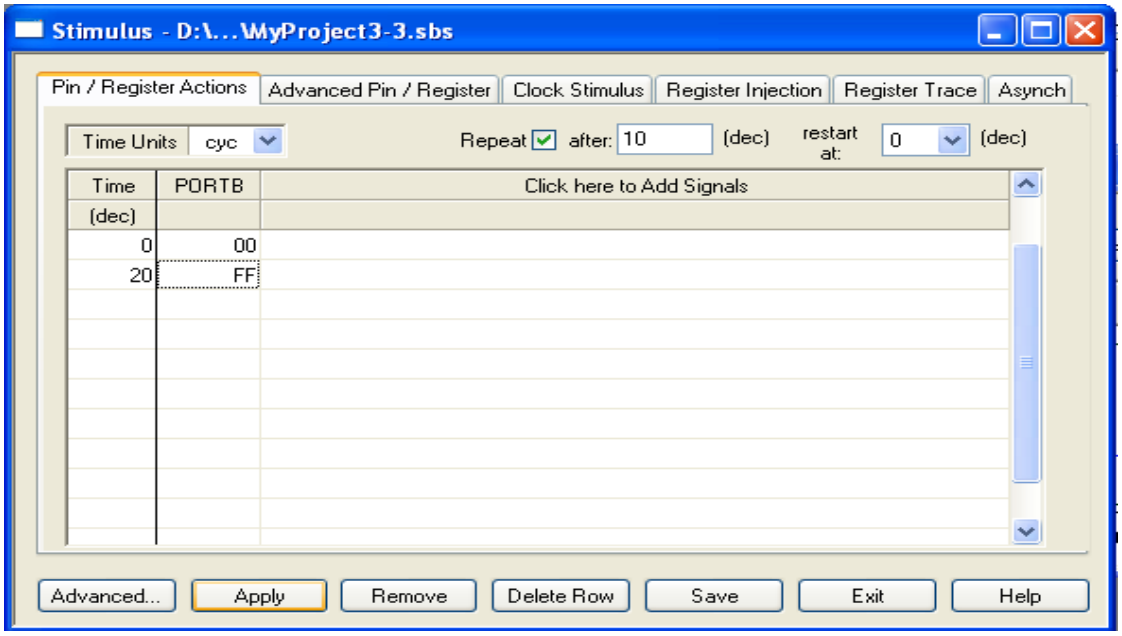


Рис. 3.1. Діалогове вікно стимулу. – Закладка Pin/Register Actions

Рис. 3.2. Діалогове вікно стимулу. – Закладка Pin/Register Actions

Для вводу даних у закладці, поданій на рис. 3.1:

1. Виберіть одиницю виміру часу, яку будете використовувати при фіксації всіх стимулів, із «випадаючого» списку рядка "Time Units".

2. «Кликніть» текст, де написано "Click Here to Add Signals" щоб відкрити діалогове вікно «Add/Remove Signals Dialog». В цьому вікні виберіть виводи, регістри або інші сигнали, до яких хочете застосувати стимул (рис.3.2). Результати вибору стануть заголовками стовпчиків.

3. Вибираючи кожен рядок, введіть час події ("Time") і значення для кожного стовпчика вивід/регістр. Час реалізації події, що визначена рядком, є загальним часом (тобто часом з моменту початку симуляції).

4. Для видалення рядка «виберіть» його, і «кликніть» «Delete Row».

5. Встановіть відмітку "Repeat after X (decimal)" для повторення стимулу у закладці після того, як відбулась остання дія стимулу. Задайте інтервал затримки, необхідний для повторення дії стимулу. Наприклад, на рис.3.1 задано тривалість затримки 10 МЦ. Правильність стимулу можна проконтролювати логічним аналізатором (рис. 3.3).

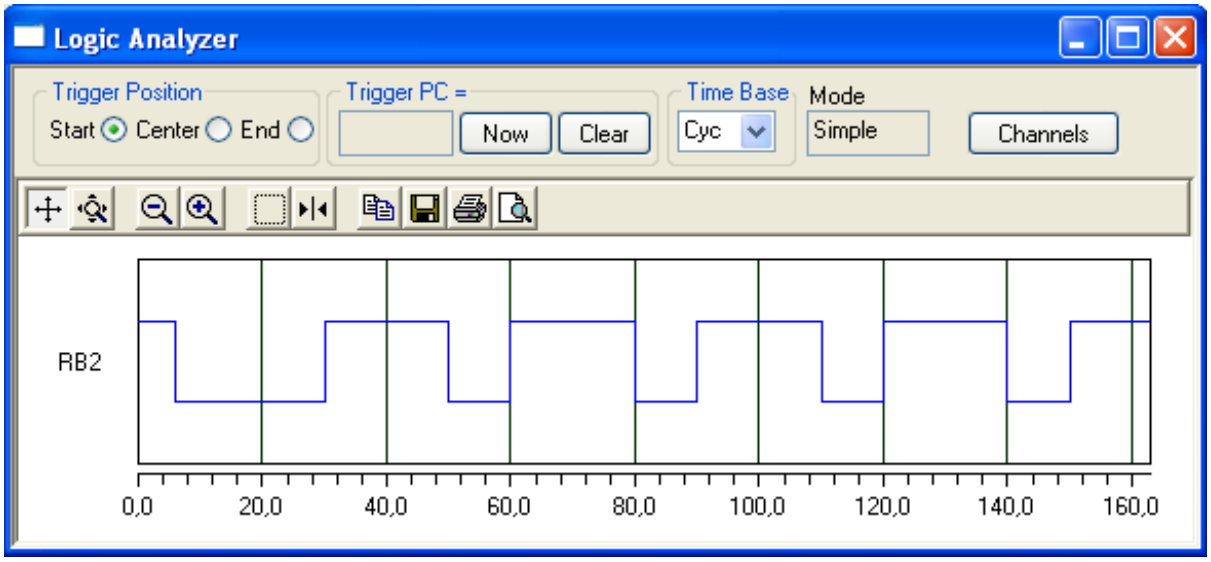


Рис.3.3. Вікно логічного аналізатора. - Робота стимулу базових дій

Одиниці часу.

Значення одиниць часу вибираються для всіх часових величин в стовп-

чику часу. Можна вибрати:

- cyc – машинні цикли (МЦ)
- h:m:s – години : хвилини : секунди
- ms - мілісекунди
- us - мікросекунди
- ns - наносекунди

При виборі h:m:s, вхідні значення в стовпчику часу мають формат:

- 1 - відповідає 1 секунді
- 1:00 - відповідає 1 хвилині
- 60 - відповідає 1:00
- 999 - відповідає 16:39
- 9999999 - відповідає 2777:46:39
- 1:5 - відповідає 1:05
- 60:5 - відповідає 1:00:05

Значення часу може бути в діапазоні від 0 до (2^{31} - 1) секунд. Все, що за межами даного діапазону, буде замінено на значення по замовчуванню (0 для першої закладки і затримки після події; 1 для умови чекання).

Наприклад, якщо задамо 100 МЦ, але одиниці часу вибрані як h:m:s, то число 100 буде інтерпретоване, як ціле число 100 і перетворене в 1:40.

3.2.3. Розширені настройки вивід/регістр (Advanced Pin/Register)

На закладці Advanced Pin/Register (рис. 3.4) можуть бути задані розширені (комплексні) синхронні дії виводу/регістру, які дозволяють ще додатково задати умови, за яких відбувається симуляція певної події. Тим самим є можливість виконувати більш складну симуляцію. Для простих базових дій використовуйте закладку Pin/Register Actions (див. рис. 3.1).

Визначення умов

Спочатку необхідно визначити умови, а потім задати реакцію при їх

виконанні. Умови визначаються для одного або кількох стимулів в кожному рядку вікна у відповідності з табл. 3.1.

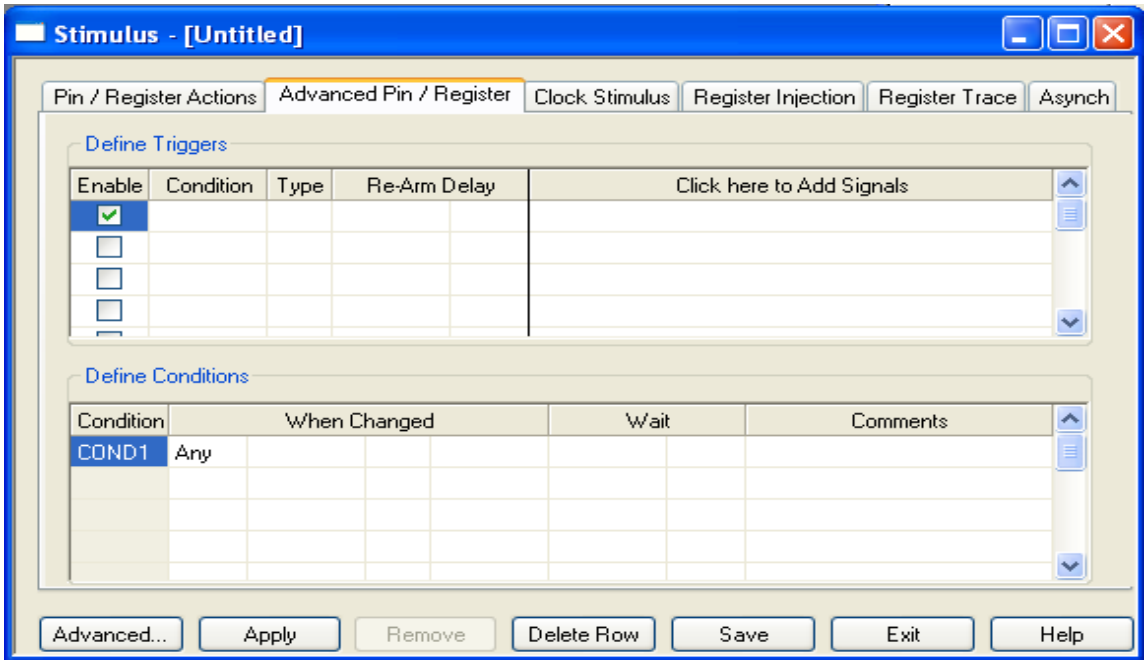


Рис. 3.4. Діалогове вікно для задавання умов, за яких відбувається подія

Таблиця 3.1. **Визначення умов для стимулу**

Функція	Визначення
Condition	Назва умови, яка створюється. Умова автоматично генерується при вводі даних в будь-якому іншому стовпчику. Ця назва – мітка буде використана для ідентифікації умови у стовпчику Condition розділу Define Triggers цієї закладки.
When Changed	Визначається вибір умови. **Стовпчик 1:** Вибір типу виводу/регістра ("SFR", "Bitfield", "Pin" або "All" для вибору всіх). Це буде фільтром значення в стовпчику 2. **Стовпчик 2:** Вибір виводу/регістра, на якому перевіряється виконання умови. **Стовпчик 3:** Вибір умови, дорівнює (=), не дорівнює (!=), менше або дорівнює (<=), більше або дорівнює (>=), менше, ніж (<), більше, ніж (>). **Стовпчик 4:** Введення значення для умови.

Продовження табл. 3.1.

Wait	Вказує час затримки після виконання умови до виконання стимулу. **Стовпчик 1:** Значення часу очікування (затримки). **Стовпчик 2:** Одиниця значення часу очікування.
Comments	Опис умови користувачем.

Наприклад, задамо умову: COND1, що у випадку, якщо значення регістра PORTB дорівнює FF, реалізує визначений в "Define Triggers" стимул з затримкою 10 мс (рис. 3.5).

Примітка. Якщо на PORTB ніколи не буде подане значення FF, або він ніколи не зміниться на FF, то умова ніколи не виконається і стимул не буде реалізований.

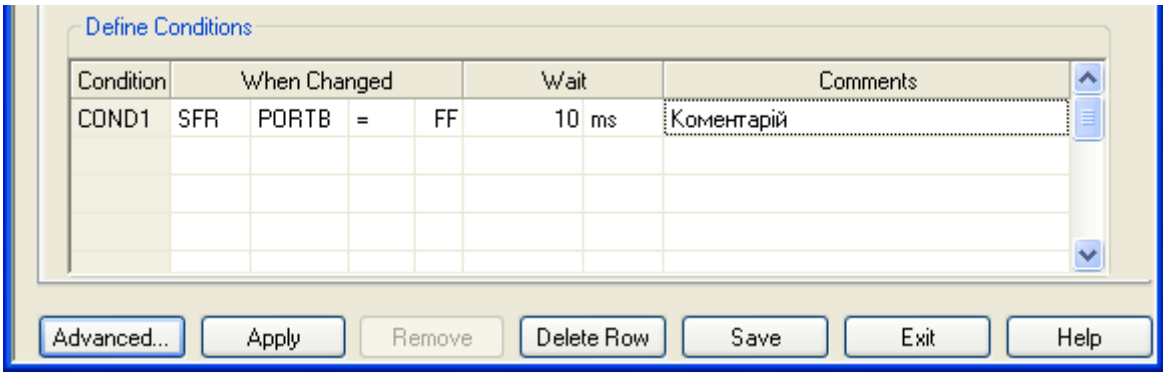


Рис. 3.5. Визначення умови прикладення стимулу

Визначення подій

Визначаємо події стимулу для кожного рядка даної секції закладки (рис. 3.6.) у відповідності з інформацією таблиці 3.2. Події відповідають заданим раніше умовам.

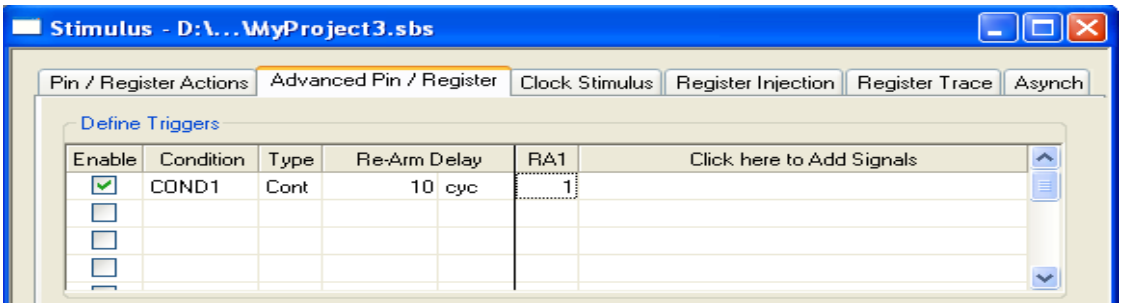


Рис.3.6. Секція визначення події для кожної із умов

Таблиця 3.2. **Визначення подій стимулу**

Функція	Визначення
Enable	Альтернативний дозвіл або заборона на подію даного рядка.
Condition	Посилання на мітку заданої умови нижньої секції даної закладки. Доступна умова вибирається зі списку.
Type	Визначаємо (вибираємо) чи умова події використовується одноразово (1x), чи повторюється (Cont).
Re-Arm Delay	Якщо Type = Cont, то вводимо значення затримки, після якої умова події знову буде перевірятися. Значення затримки вводиться в першому стовпчику, а одиниця вимірювання – у другому.
Click Here to Add Signals	«Кликнемо» текст з написом "Click here to add/remove signals" щоб відкрити діалог Add/Remove. В цьому діалогу вибираємо виводи, регістри або інші сигнали по відношенню до яких буде застосований стимул. Вибрані налаштування будуть наведені у заголовках стовпчиків.

Для умови, наведеної у попередньому прикладі, COND1, задана така послідовність реалізації стимулу:

1. Встановлюється на виводі RB0 високий рівень при першому виконанні умови.

2. Очікується 10 МЦ і знову перевіряється умова. Якщо вона виконується, то знову на виводі RB0 високий рівень.

3. Повторюємо крок 2 до завершення роботи програми.

Різниця між часом очікування в "Define Conditions" і часом затримки після події (Re-Arm Delay) в "Define Trigger". В "Define Conditions" час очікування - це час від моменту виконання умови до моменту, коли почне виконуватись стимул. В "Define Trigger", затримка після події – це час до моменту, коли буде проведена повторна перевірка умови для події і на протязі якого стимул все ще виконується.

3.2.4. Стимул синхронізації (Clock Stimulus)

Вікно вводу сигналів стимулу синхронізації наведено на рис. 3.7. Значення імпульсу (високий або низький рівні), прикладеного до виводу, використовується, відповідно таблиці 3.3.

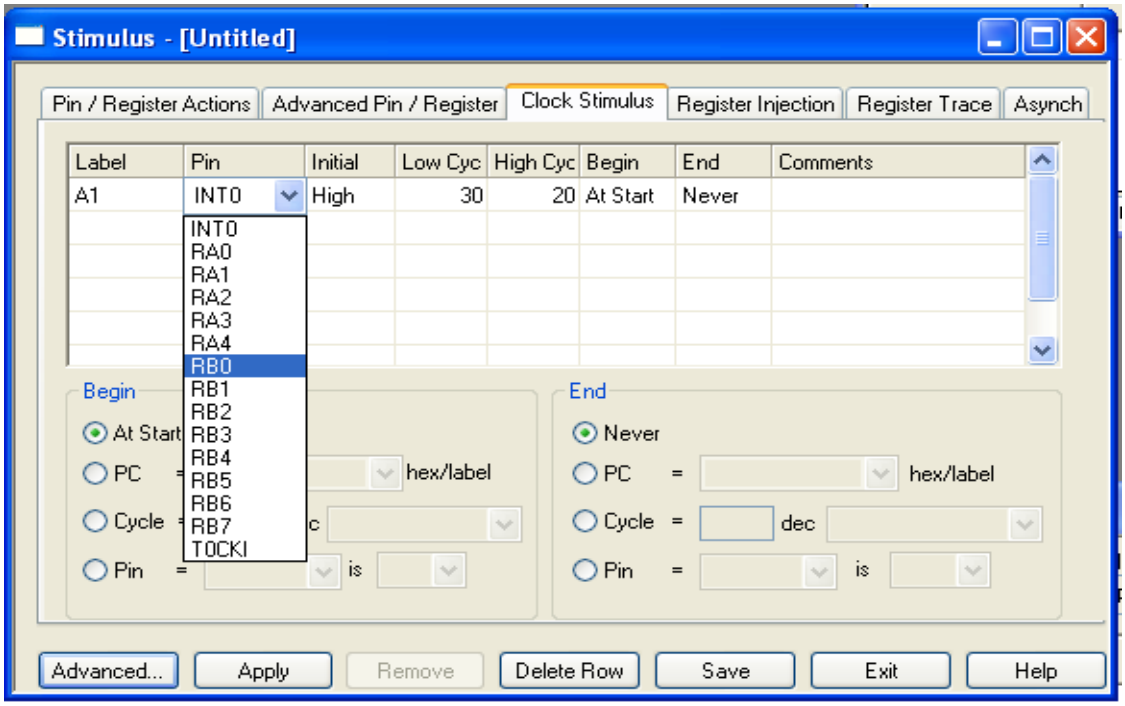


Рис. 3.7. Вікно стимулу синхронізації

На рис. 3.8 показано сигнал на вході RB0. Кожен імпульс призводить до обробки переривання INT0, відповідно до наведеної програми.

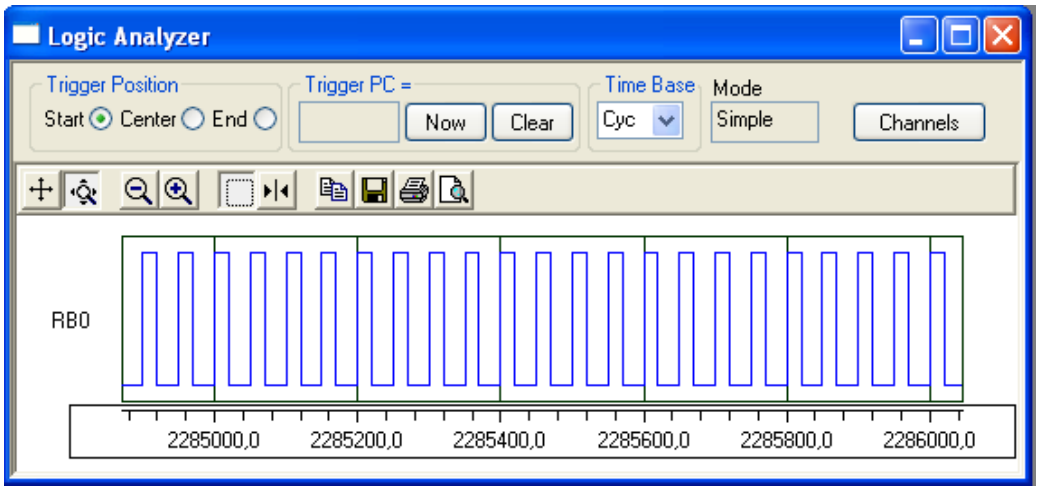


Рис. 3.8. Демонстрація роботи стимулу

***Таблиця 3.3.* Визначення для сигналів стимулу синхронізації**

Функція	Визначення	
Label	Ім’я стимулу синхронізації, яке задається користувачем (опція)	
Pin	Вибір виводу, до якого буде прикладений стимул синхронізації	
Initial	Введення моменту ініціалізації синхронізації низьким або високим рівнем	
Low Cycles	Введення значення для числа циклів в імпульсі синхронізації низького рівня.	
High Cycles	Введення значення для числа циклів в імпульсі синхронізації високого рівня	
Begin	«Кликнути» Begin для активації розділу:	
	Always (default)	Почати стимуляцію негайно після запуску програми
	PC=	Почати стимуляцію, коли програмний лічильник досягне введеного значення
	Cycle=	Почати стимуляцію, коли лічильник циклів досягне введеного значення. Цей вираз завжди показує абсолютний час, тобто від моменту початку симуляції. Цикл завершення (End Cycle) може показувати відносне значення до початку симуляції або заданого тут Begin Cycle (див. нижче)
	Pin=	Почати стимуляцію, коли вибраний вивід прийме вибране значення (0 або 1)
End	«Кликнути» End для активації розділу:	
	Never (default)	Стимул використовується доки програма не зупиниться
	PC=	Завершення стимулу, якщо програмний лічильник досяг введеного значення
	Cycle=	Завершення стимулу, якщо лічильник циклів досяг введеного значення *absolute time* (cyc) – відносно початку симуляції. *from clock start* (cyc+) – відносно початку циклу (див. вище)
	Pin=	Завершення стимулу, коли вибраний вивід має вибране значення (0 або 1)

3.2.5. Примусова зміна значення регістру (Register Injection)

В регістр Register Injection може бути записане («вкинуте») значення (вміст), використовуючи закладку, наведену на рис. 3.9. Інформація для визначення параметрів наведена в табл. 3.4.

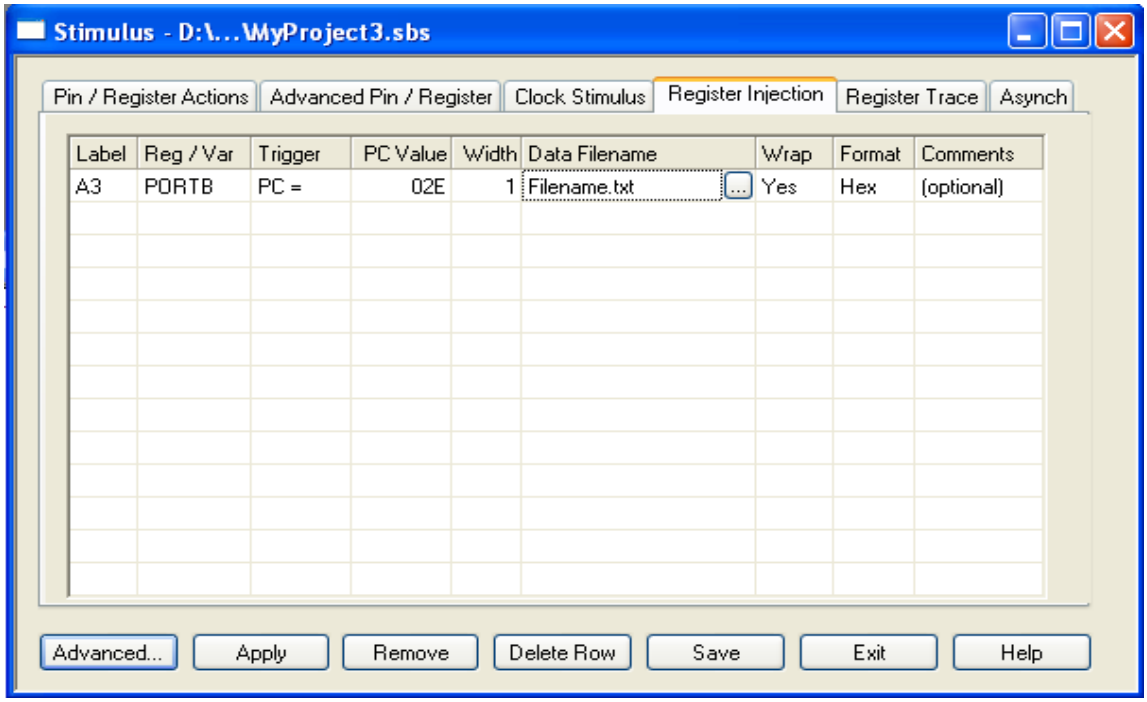


Рис. 3.9. Вікно визначення примусової зміни вмісту регістру

Таблиця 3.4. **Визначення для зміни вмісту регістра**

Функція	Визначення
Label	Ім’я події, за якої відбувається зміна вмісту регістру, задається користувачем
Destination	Вибір із списку регістра, призначеного для запису даних (зміни вмісту). Список регістрів включає SFR (вверху списку) і будь-якої GPR користувача (внизу списку)
Trigger	Вибір «значення» події, відповідно до якої відбуватиметься зміна вмісту регістру: «за вимогою» або коли PC рівний заданому значенню Якщо периферія приєднана, то можна вибрати лише подію «за вимогою». Якщо периферія відсутня, то слід використовувати PC. Наприклад, якщо встановлений біт ADON і вибраний регістр ADRESL, то почнеться зміна вмісту (трасування)

Продовження табл. 3.4

Функція	Визначення
PC Value	Введіть значення, при якому відбуваєтиметься зміна вмісту регістра (це може бути абсолютний адрес або мітка програми)
Width	Введіть число байт, відведених для зміни вмісту
Data Filename	Вибір файла даних для запису
Rewind	**Yes** – Відразу після запису всіх даних з вибраного файлу процес запису повторюється **No** - Відразу після запису всіх даних з файлу останнє значення буде використовуватись для здійснення зміни вмісту регістру
Format	Визначає формат файлу для зміни вмісту регістра Файл постійних даних: **Hex** - ASCII Hex- десятковий **Raw** – Raw - бінарний **SCL** – формат SCL **Dec** - ASCII десятковий Файл даних, основаних на повідомленні: **Pkt** - Hex або Raw пакетний формат (використовується з регістром приймача UART) **Зауваження:** коли імпортується STI-файл стимулу, довжина кожної лінії не повинна перевищувати 260 символів
Comments	Добавлення описової інформації про зміну вмісту регістру

3.2.6. Трасування регістру (Register Trace)

Вміст вибраного регістру для певних умов може бути збережений у файлі (бути трасованим) при виконанні програми. Вікно параметрів показано на рис. 3.10, з якого видно, як задати для трасування змінну RB_Temp, а результати трасування записати у файл Filename.txt. Більш детальна інформація для визначення параметрів трасування регістру наведена в табл. 3.5.

Таблиця 3.5. **Визначення для трасування регістра**

Функція	Визначення
Label	Задане ім’я трасування регістру (опційно)

Продовження табл. 3.5

Функція	Визначення
Source	Вибір зі списку регістра джерела для трасування. Наведені у списку регістри: SFR (вгорі списку) і будь-який регістр загального призначення GPR (внизу списку)
Trigger	Вибір «значення» події, відповідно до якої відбуватиметься трасування: «за вимогою» або коли PC рівний заданому значенню Якщо периферія приєднана, то можна вибрати лише подію «за вимогою». Якщо периферія відсутня, то слід використовувати PC. Наприклад, якщо встановлений біт ADON і вибраний регістр ADRESL, то почнеться трасування.
PC Value	Введіть значення, при якому відбуватиметься трасування (це може бути абсолютний адрес або мітка коду програми)
Width	Введіть число байт, відведених для трасування
Trace Filename	Вибір розміщення файлу трасування
Format	Виберіть формат файлу для трасування: **Hex** - ASCII Hex - десятковий **Raw** – Raw - бінарний **SCL** – формат SCL **Dec** - ASCII десятковий
Comments	Внесення описової інформації про трасування регістру.

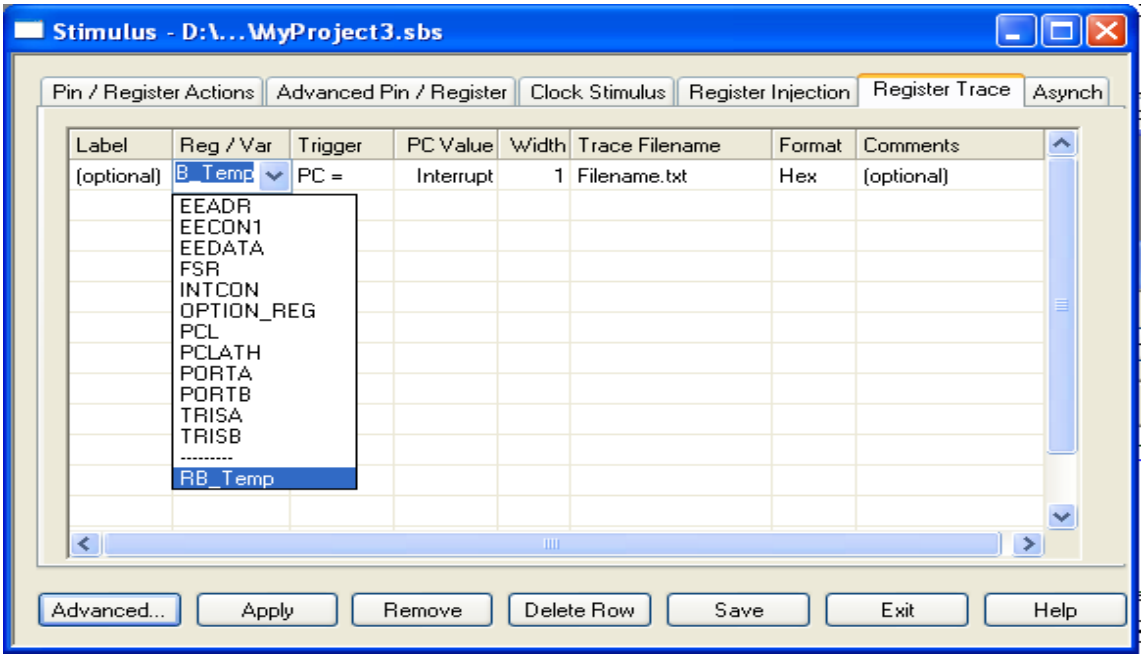


Рис. 3.10. Вікно параметрів трасування регістру

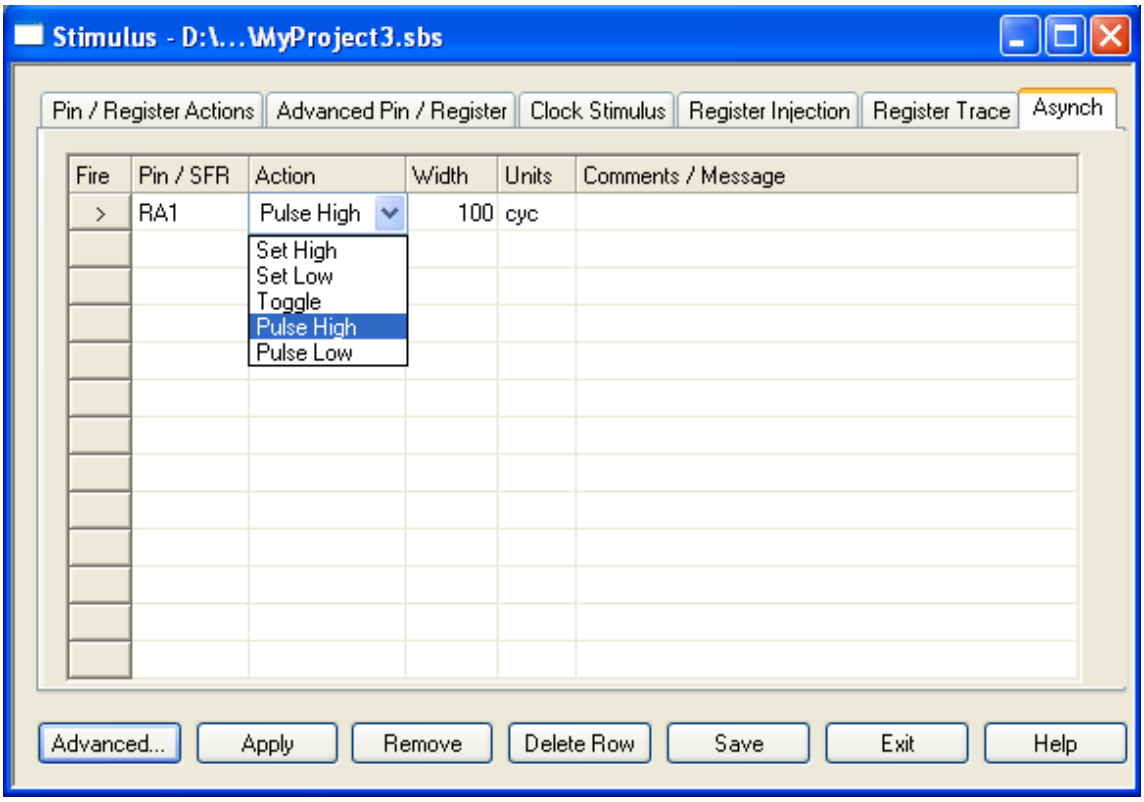


Рис.3.11. Вікно керування асинхронними стимулами

3.2.7. Керування асинхронним стимулом (Asynch)

Закладка Asynch (рис. 3.11) задає параметри використання стимулу в MPLAB SIM. Asynch також дозволяє вибрати стимули, які ви виконаєте, і зберегти цю інформацію у робочій книзі.

Примітка: Діалог стимулу повинен бути відкритий для стимуляції, щоб залишатись активним на протязі симуляції

Введіть рядок за рядком будь-які асинхронні стимули. Для видалення рядка «виберіть» його та «кликніть» Delete Row.

Можливі два типи асинхронних стимулів: регулярний стимул, для більшості регістрів SFR, і стимул, оснований на повідомленнях, для регістрів SFR модуля USART/UART.

Таблиця 3.6. **Визначення параметрів керування стимулом в MPLAB SIM**

Функція	Визначення
Fire	«Кликніть» кнопку у цьому стовпчику відповідно до рядка, для якого бажаєте виконати стимул події. Звичайно, слід встановити вверх меню всіх інших рядків, перш ніж використаєте "Fire"
Pin/SFR	Вибір або зміна виводу мікроконтролера (Pin) або регістру (SFR) до якого прикладено стимул. Доступні вікна наведені у списку «випадаючого» меню. Перелік імен залежить від вибраного в MPLAB® IDE пристрою
Action	Вибір або зміна дії стимулу **Регулярний стимул:** Виберіть "Pulse", "High", "Low" або "Toggle". **Стимул, оснований на повідомленні:** Виберіть "File Message" або "Direct Message". "File Message" означає використання файлу з пакетним повідомленням
Width	Якщо вибрано "Pulse", то можна вибрати або змінити ширину імпульсу - введіть нове значення в наступній комірці рядка
Units	Якщо вибрано "Pulse", то можна вибрати або змінити одиницю вимірювання ширини імпульсу. Введіть значення для одиниці вимірювання ширини імпульсу
Comments / Message	**Регулярний стимул:** Задайте або змініть коментарій про стимул **Стимул, оснований на повідомленнях:** Задайте або змініть повідомлення стимулу (повідомлення не повинно мати даних, наприклад, час затримки)

3.2.8. Збереження лаштувань модуля стимулу (General SCL File)

Після завершення роботи з діалоговим вікном можна створити файл робочої книги **(**General SCL File), попередньо «кликнувши» кнопку Advanced... (рис. 3.11).

Примітка: Стовпчик Time залишається зафіксованим, коли ви прокручуєте стовпчики сигналів

Значення одиниць виміру за замовчуванням

Для всіх таблиць файлу SCL, використовуються такі правила:

- Одиниці часу завжди в десятковій системі числення.
- Значення PC завжди в HEX коді.
- Сигнали на виводах завжди 0 або 1.
- Значення регістрів завжди в HEX коді.
- Значення бітових полів завжди у двійковому коді.

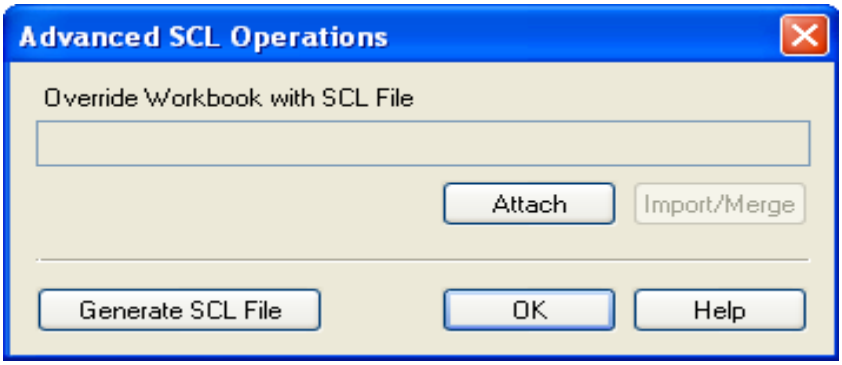


Рис. 3.12 Вікно розширення для синхронної стимуляції

Розширені операції SCL

Для використання створеного файлу синхронного стимулу скористайтесь діалоговим вікном стимулу, «кликнувши» Advanced... для його відкриття (рис. 3.12).

***Таблиця 3.7.* Функції діалогового вікна кнопки Advanced**

Функція	Визначення
Override Workbook with SCL File	Ім’я та вибране місце розміщення файлу SCL
Attach/Detach	Реалізує приєднання (Attach) або від’єднання (Detach) залежно від наявності вказаного файлу в полі Override Workbook with SCL File
Import/Merge	Виконання імпорту SCL файлу, або об’єднання кількох SCL файлів
Generate SCL File	Генерування SCL файлу на основі параметрів будь-якої з перших п’яти закладок вікна (синхронний стимул)
OK	Підтвердження дії, вибраної в даному діалоговому вікні
Help	Відкривання допомоги модуля стимуляції

Знайдіть необхідну папку і файл та «кликніть» Attach для його приєднання. Щоб видалити файл SCL «кликніть» Detach. Якщо необхідно об'єднати два файли SCL в один - «кликніть» Import/Merge.

3.3. Завдання і порядок виконання роботи

В лабораторній роботі необхідно створити та побудувати проект, а також виконати його тестування за допомогою стимулятора.

Завдання виконується в такій послідовності:

- Вибрати тип мікроконтролера - PIC16F84.
- Запустити майстер проекту і створити проект.
- Записати наведену в описі лабораторної роботи програму, в якій виконується введення в порт входу/виходу зовнішніх даних після фіксації синхроімпульсу на вході зовнішнього переривання.
- Вивчити алгоритм роботи записаної програми.
- Побудувати проект.
- Тестувати програму за допомогою стимулятора.
- Отримати у викладача індивідуальне завдання: тактову частоту мікроконтролера, частоту зовнішнього синхросигналу, тривалість строба дозволу, значення інформаційного байта для вводу.

Завдання. Налаштуйте в модулі стимулу задану швидкість передачі *n*-розрядних даних на вхід мікроконтролера. Забезпечте розпізнавання коду та збереження його в пам'яті даних (варіанти завдання наведені в табл.3.8).

- Ввести необхідні зміни в програму та перевірити правильність алгоритму за допомогою стимулу, задавши зовнішні сигнали і перевіривши результат за допомогою трасування регістра зберігання введених даних.
- Оформити в текстовому редакторі протокол, в якому навести необхідні розрахунки і зображення діалогових вікон з підтвердженням результатів розрахунку.

Таблиця 3.8. **Приклади варіантів завдань**

№	f_{CLK}, *МГц*	Частота зовнішнього синхросигналу, *Гц*	Інформаційний байт	тривалість стробу дозволу
1	2	2400	0xAA	0.01 сек
2	3.66	4800	0x55	
3	4	9600	0xC6	
4	4.33	19200	0x3C	
5	5	38400	0x81	
6	6	57600	0xE7	
7	8	115200	0xD4	
8	10	230400	0x96	
9	11.66	924600	0x33	
10	12.33	1200	0x4B	
11	16	300	0xF1	
12	20	110	0x4E	

Контрольні питання

1. Правила виконання обробки зовнішніх переривань.
2. Які можливості для розробника надає модуль стимуляції?
3. Чим відрізняються синхронний і асинхронний стимули?
4. Трасування регістру.
5. Яким чином можна розпізнати код, що передається на вхід порту мікроконтролера послідовно?
6. Охарактеризувати команди циклічного зсуву через перенос.

Література до лабораторної роботи № 3

1. Справочник по всем микроконтроллерам PIC16 [разделы 8, 9].
http://www.microchip.ru:/files/d-sheets-rus/PIC16_Manual.pdf
2. Руководство пользователя MPLAB IDE. [стр. 63-68, 70-73]
http://www.microchip.ru:/files/d-sheets-rus/mplab_ide.pdf

ЛАБОРАТОРНА РОБОТА №4

Реалізація схеми динамічної індикації на мікроконтролері та її дослідження за допомогою інтерактивного симулятора PROTEUS

4.1. Мета роботи

Вивчити використання інтерактивного симулятора PROTEUS для побудови, налагодження і дослідження схеми радіотехнічного пристрою. Вивчити організацію динамічної індикації з використанням мікроконтролера і семисегментних індикаторів, реалізувати запропонований алгоритм та дослідити його роботу за допомогою інтерактивного симулятора.

4.2. Основні теоретичні відомості

4.2.1. Створення проекту за допомогою PROTEUS

Програма симуляції запускається викликом ISIS Professional – складової частини PROTEUS. При появі вікна програми пропонується переглянути приклади проектів.

Масштаб зображення проекту можна змінити за допомогою колеса миші, або за допомогою інструментів ZOOM (див. рис. 4.1). На жаль, програма не має елементу прокручування зображення і тому для переміщення зображення слід вказати новий центр зображення інструментом "хрест". Для перегляду всього робочого листа необхідно натиснути кнопку правіше від ZOOM "-". Для детального перегляду певної області схеми її необхідно виділити за допомогою інструмента виділення (ще правіше).

В лівій верхній області вікна програми (рис. 4.1) видно мінімакет сторінки, а нижче нього - панель DEVICES (див. рис. 4.2*а*).

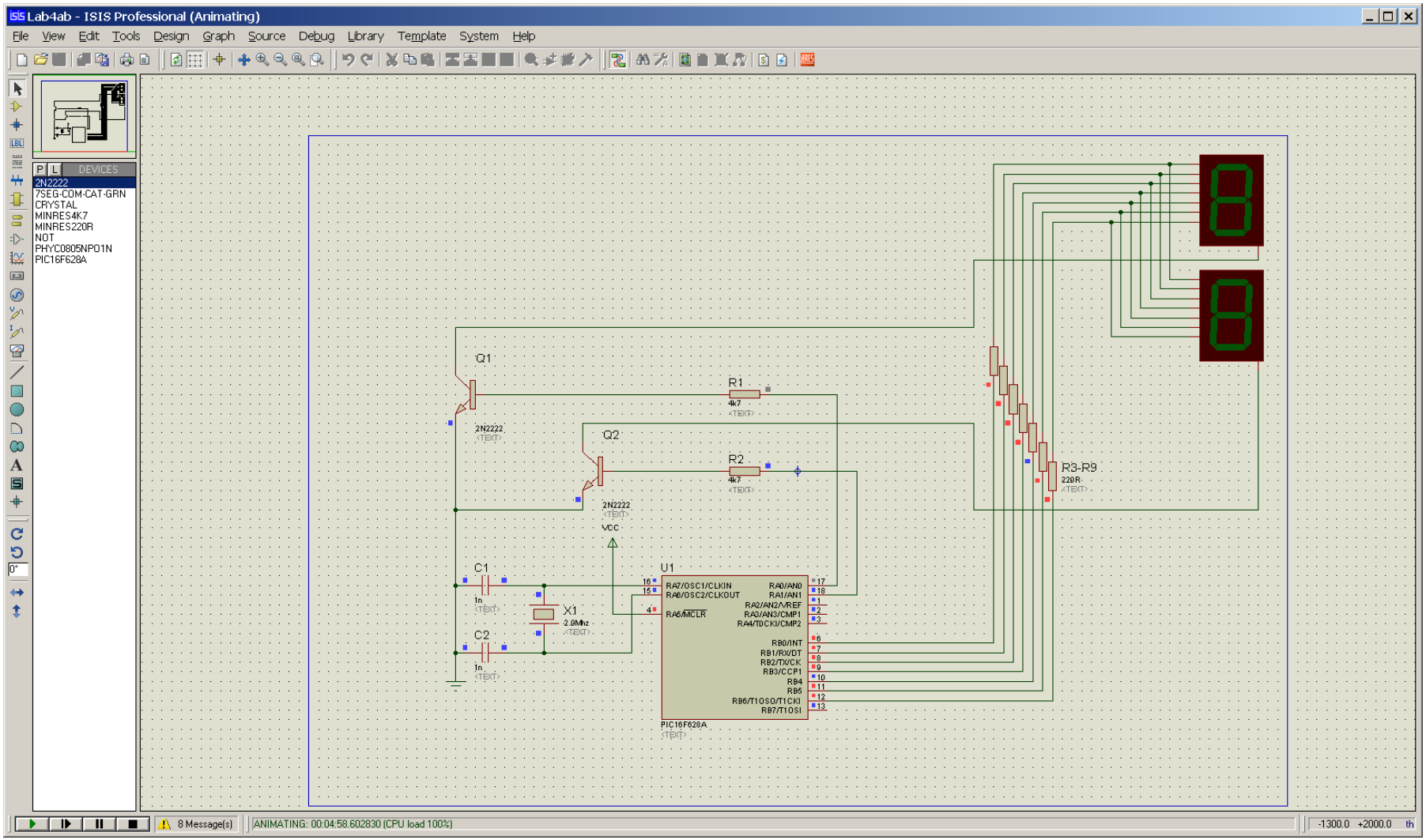
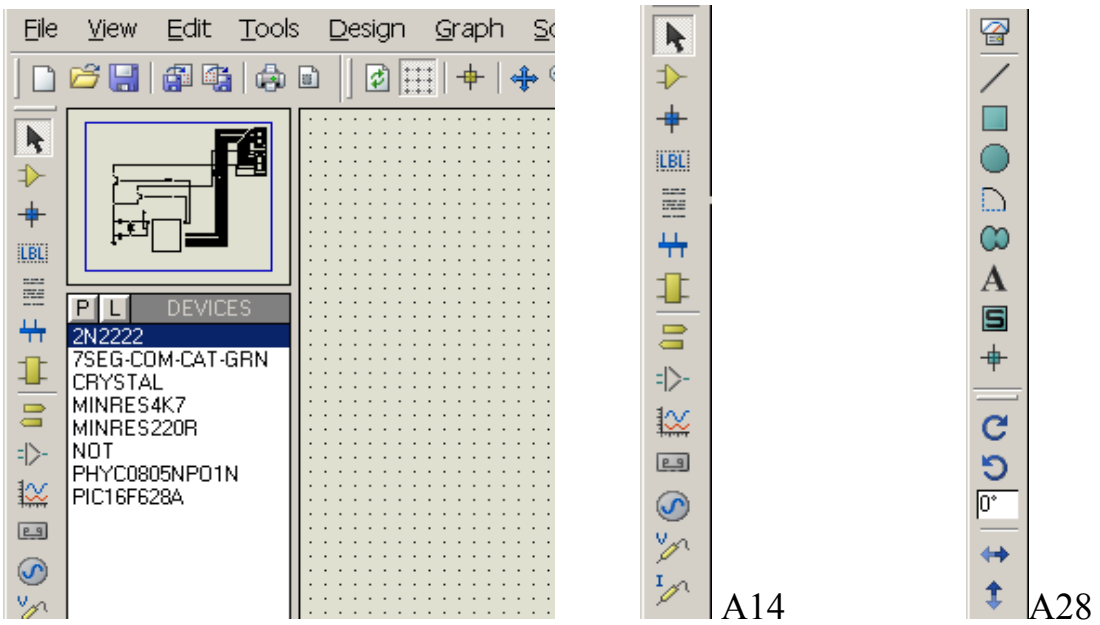


Рис. 4.1. Вікно програми PROTEUS із створеним проектом

У вікні панелі DEVICES відображуються всі елементи (компоненти проекту), які використовуються в схемі. Кнопка з літерою "P" відкриває форму пошуку в бібліотеках PROTEUS (рис. 4.3) компонента, який добавляємо в схему. Кнопка з літерою "L" запускає менеджер бібліотек і з його допомогою можна приєднати нові бібліотеки компонентів.



а *б* (A1..A14) *в* (A15..A28)

Рис. 4.2. Список елементів схеми:

а - доданих з бібліотеки (кнопка P); *б*, *в* - інструменти лівої панелі.

Ліва панель інструментів багатофункціональна (рис. 4.2 *б*, *в*) і функції основних її інструментів такі:

A1 – стрілка-показник для виділення областей, побудови з’єднань, ін.;

A2 – відкрити панель DEVICES - компоненти проекту і пошук нових;

A3 – проставити вручну місце з’єднання провідників;

A4 – назвати провідник (зручно через об’єднання провідників, які мають однакову назву);

A5 – добавити текст в довільне місце схеми;

A6 – провести шину - на схемі широка темно-синя лінія;

A7 – створення підсхеми (блоку, який має власну схему і елементи з’єднання);

A8 – Terminals – кінцеві елементи схеми: живлення (VCC), земля (GND), виводи...;

A9 – режим Device Pin - додати вивід до компонента;

A10 – графічне відображення, збереження і аналіз результатів симуляції;

A11 – запис процесу симуляції в файл і відтворення даних;

A12 – генератори різних форм напруги та струмів;

A13 – визначення місця на провіднику для вимірювання напруги;

A14 – визначення місця на провіднику для вимірювання струму;

A15 – Virtual Instruments - віртуальні вимірювальні пристрої:

A16 – проведення провідників на схемі;

і т.д.

A24 – поворот за часовою стрілкою на 90 градусів виділеного компонента або блока;

A25 – поворот проти часової стрілки на 90 градусів виділеного компонента або блока;

A26 – поле для визначення довільного кута повороту;

A27 – дзеркальне відображення елемента в горизонтальній площині;

A28 – дзеркальне відображення елемента у вертикальній площині.

Відкриємо форму – меню пошуку та вибору компонентів "Pick Devices", натиснувши на клавіатурі кнопку з літерою "**P**". В полі "Keywords" (ключові слова) (див. рис. 4.3) можна задати для пошуку ключове слово, наприклад, початок назви елемента, а потім вибрати елемент. В полі "Description" (опис) побачимо опис елемента.

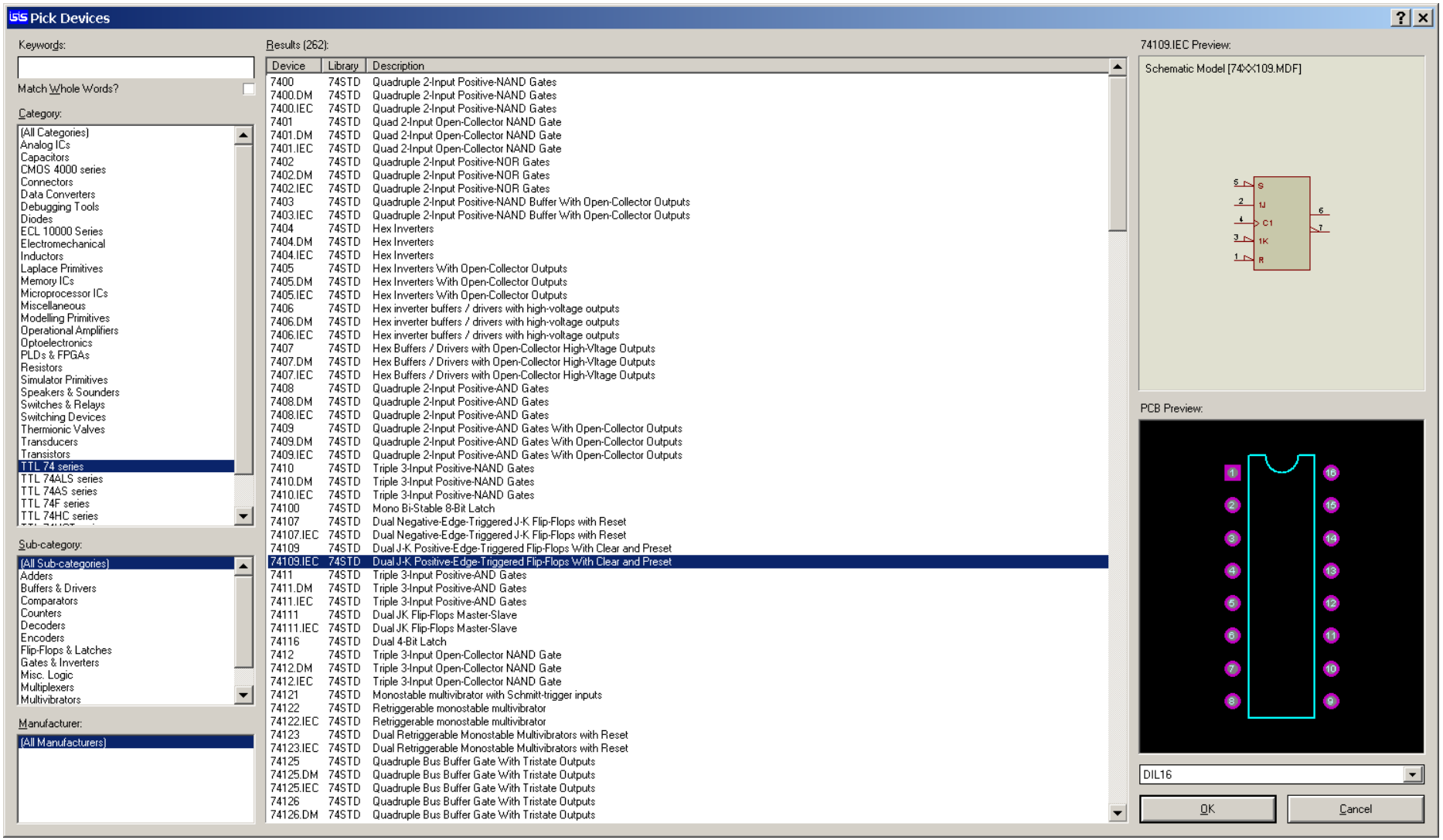
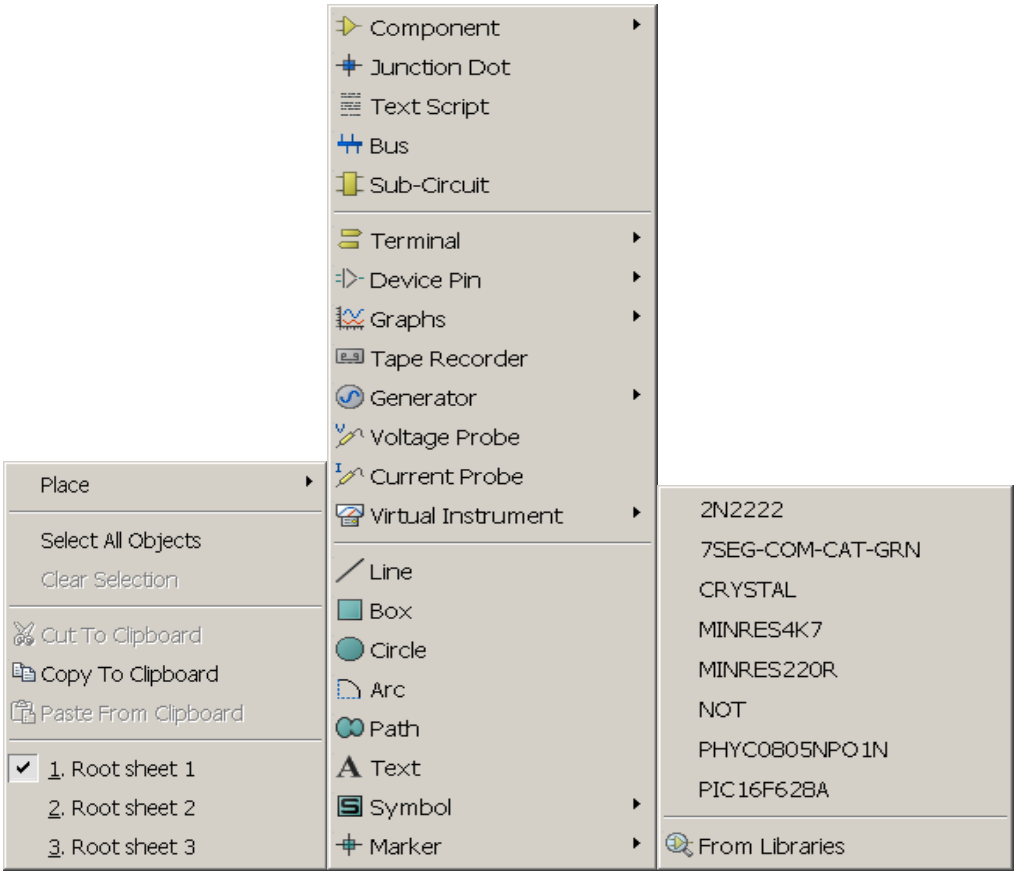


Рис. 4.3. Вікно бібліотеки вибору елементів схеми

В правій частині форми приводиться назва моделі компонента, його зображення на схемі, а внизу форми – "FootPrint" – вигляд елемента на друкованій платі. Ще нижче – назва корпуса компонента DIL16 (рис. 4.3) – прямокутний пластмасовий корпус з 16 виводами.

Щоб розмістити компонент на схемі необхідно «кликнути» "OK", а потім навести курсор миші в потрібне місце на листку схеми і натиснути ліву клавішу миші. Можна спочатку додати всі компоненти схеми у вікно компонентів, а потім скористатися контекстним меню правої кнопки миші (див. рис. 4.4). Наприклад, додати мікроконтролер PIC16F628 у вікно компонентів із бібліотеки і перейти в контекстному меню по шляху *Place* (рис. 4.4*а*) > *Component* (рис. 4.4*б*) > *PIC16F628* (рис. 4.4*в*).



а *б* *в*

Рис. 4.4. Контекстне меню додавання елементів схеми

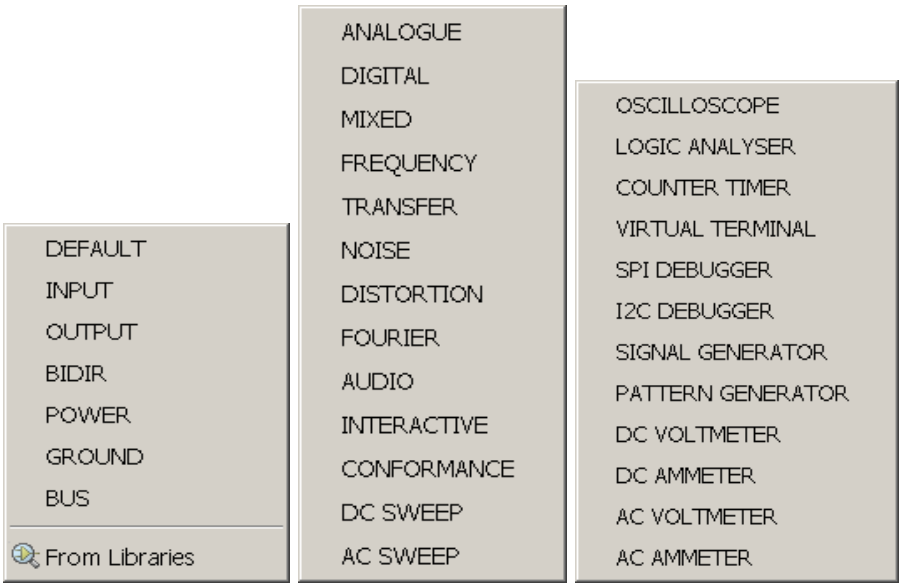
Можна також скористатися копіюванням через буфер обміну елементів,

які вже присутні на схемі, але при цьому доведеться вручну вводити порядковий номер нового елемента.

Виділяють будь-який компонент або інший елемент схеми за допомогою правої кнопки миші. При цьому його колір змінюється на червоний і одночасно виділяться всі провідники, які приєднані до нього – виділяються червоним до першого з'єднання з чим-небудь. Відмінити виділення всіх елементів можна «кликом» правої кнопки миші на пустому місці поля.

УВАГА! Подвійний «клик» правою кнопкою миші на елементі видаляє його з поля.

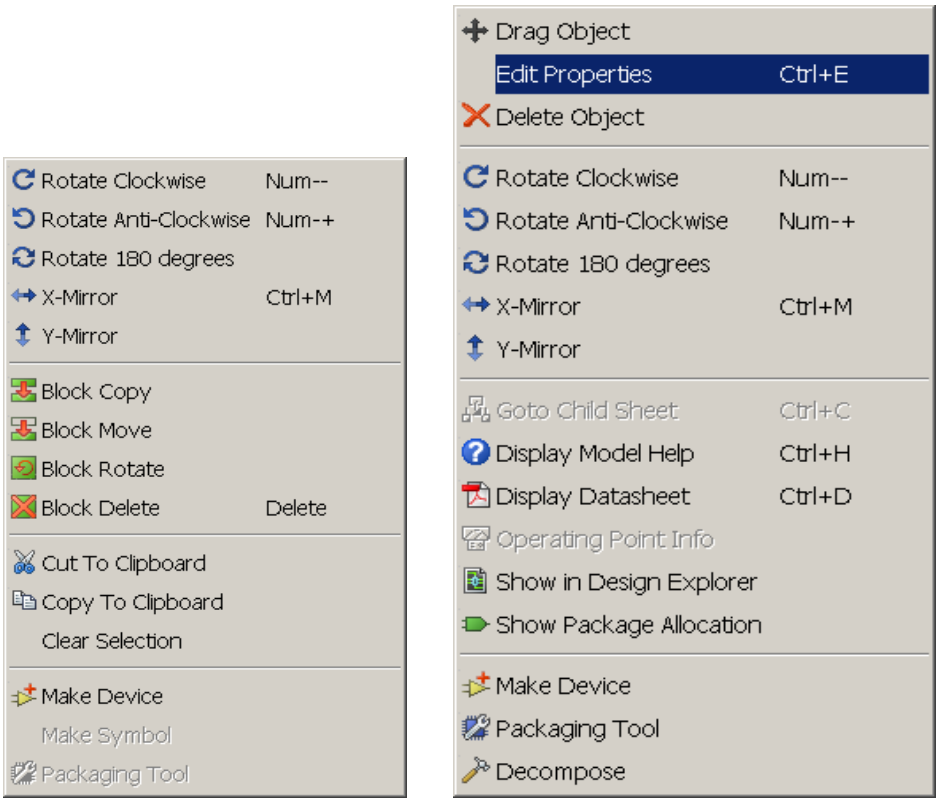
Додаткові поля контекстного меню наведені на рис. 4.5, а для редагування параметрів елементів використовують контекстні меню елементів (див. рис. 4.6).



а *б* *в*

Рис.4.5. Додаткові вікна контекстного меню поля проекту:
а - Terminal, *б* - Graphs, *в* – Generator

На рис. 4.7 наведена схема динамічної індикації, симуляцію якої необхідно виконати в даній лабораторній роботі, а список її елементів заданий в розділі 4.3.



а *б*

Рис. 4.6. Контекстні меню елементів для зміни їх положення та редагування параметрів

4.2.2. Налаштовування процесу симуляції схеми

Виділимо мікроконтролер (МК) за допомогою правої кнопки миші. МК і приєднані до нього провідники червоніють. З меню Edit Component (рис. 4.6б) лівою кнопкою миші відкриємо панель редагування властивостей компонента. «Кликнемо» кнопку "Hidden Pins" (заховані виводи), щоб відкрити додаткове меню, в якому показано як по замовчуванню називаються вузли (провідники) схеми до яких приєднано живлення мікроконтролера – **VCC** і його "земля" – **GND**. Можливо потрібно змінити ці назви, якщо живлення подане від різних джерел.

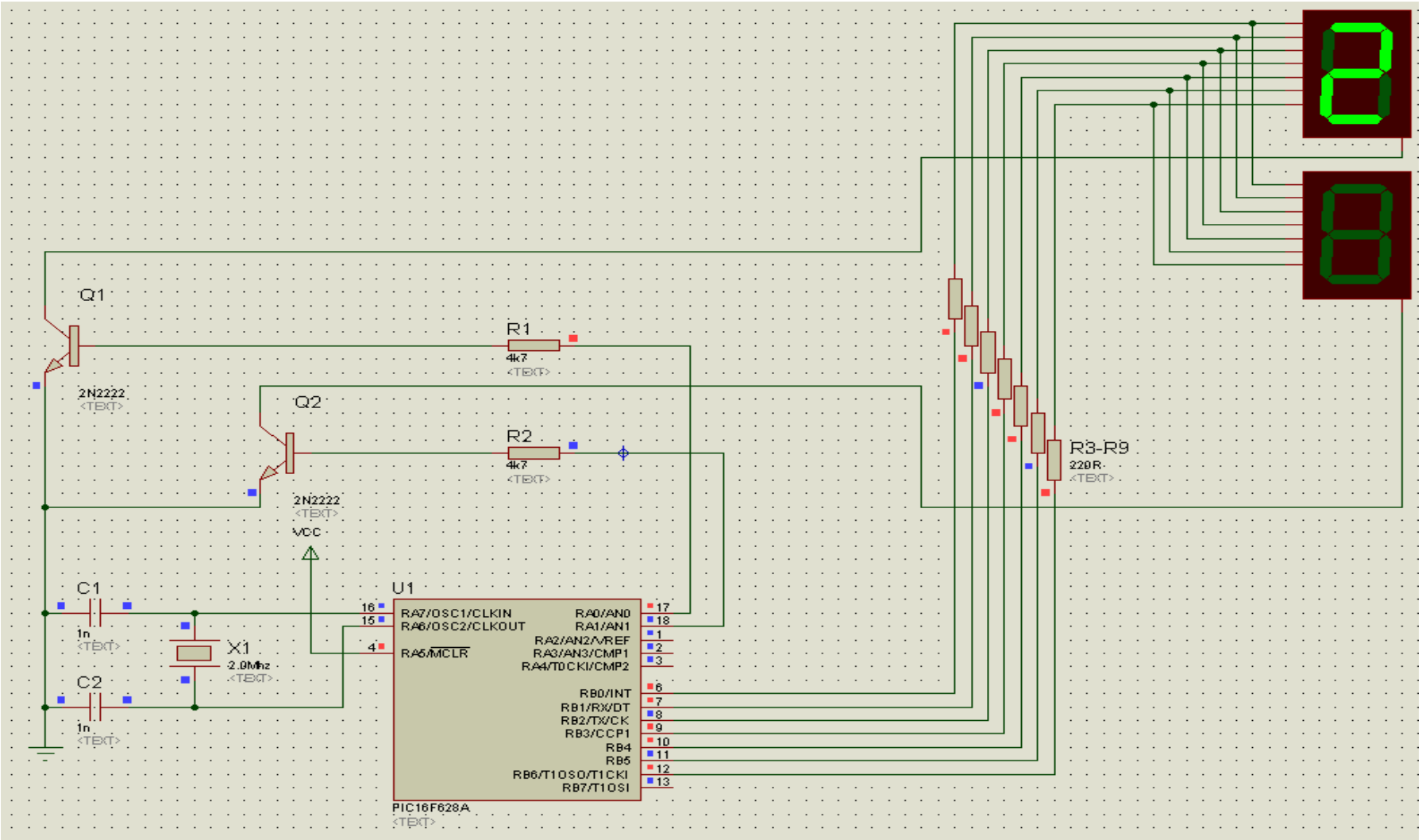


Рис. 4.7. Схема динамічної індикації

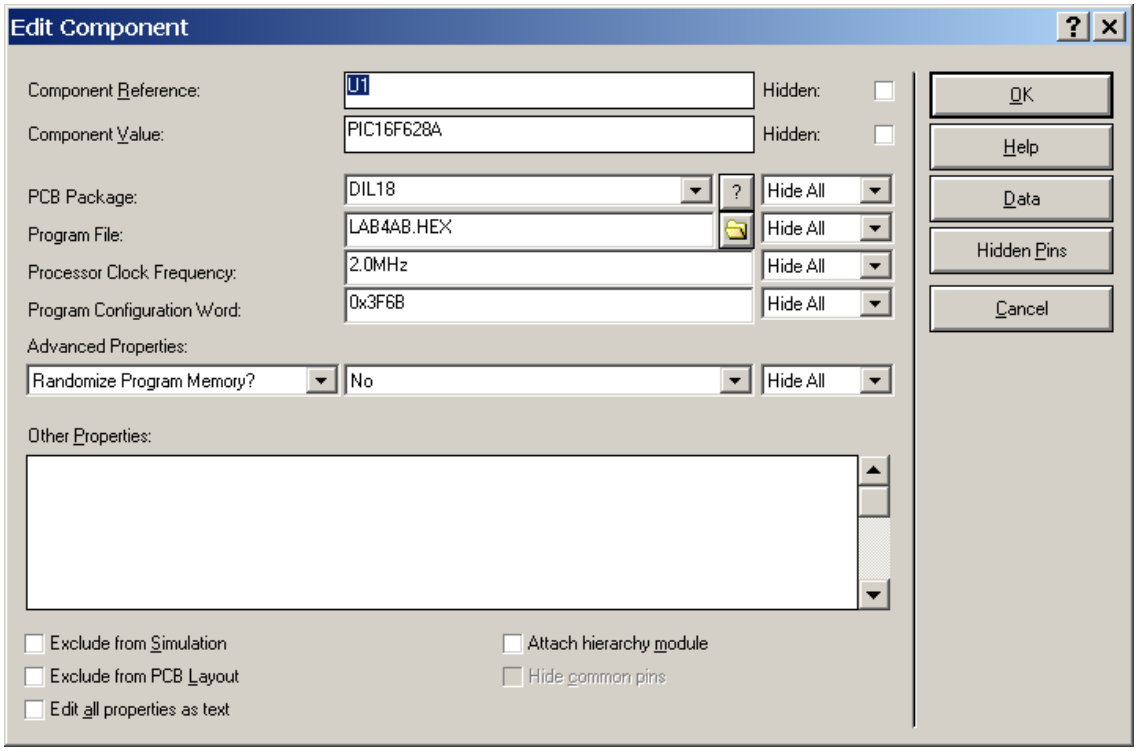


Рис. 4.8. Вікно параметрів мікроконтролера

(тактова частота, HEX-файл програми)

«Кликнемо» "ОК", щоб закрити панель захованих виводів і ознайомимось із змістом панелі редагування компонента. Для мікроконтролера головне програма, за якою він працюватиме. В полі "Program File" слід вказати:

.cof – файл, якщо для симуляції використовується первинний текст програми на мові С;

.asm - файл, якщо симуляції використовується первинний текст програми на мові Асемблер;

.hex – файл прошивки, якщо немає первинного тексту на жодній з мов.

Якщо є первинний текст .asm, то потрібно далі вказати назву .hex файла, а також через меню *Source> Add/Remove Source File...* – додати назву файла з текстом програми на асемблері і вибрати необхідний компілятор (наприклад, для PIC це MPASM) (рис. 4.9).

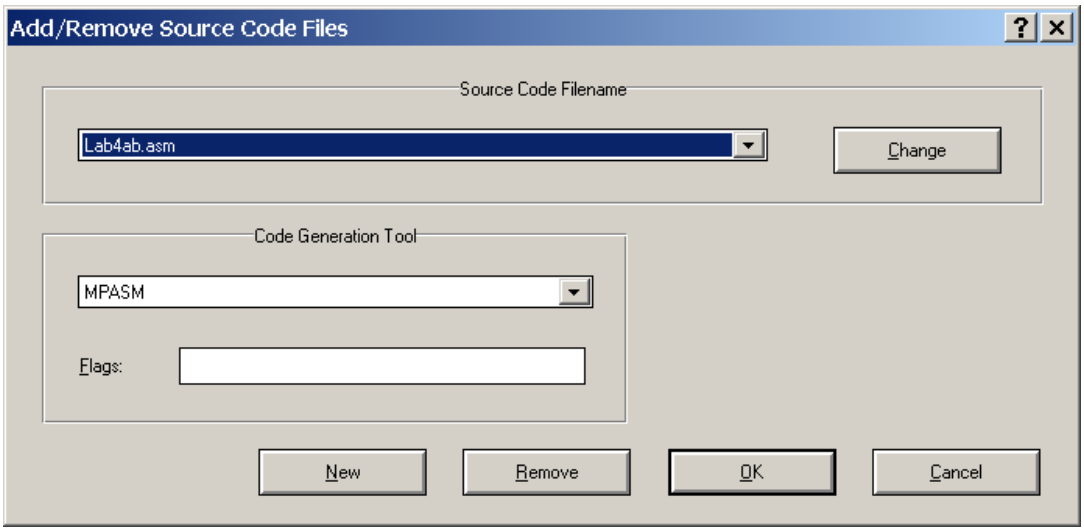


Рис. 4.9. Вікно добавлення програми для мікроконтролера та вибору інструмента для отримання HEX-файла

Тепер при запуску симуляції буде проведене компіляція файла Lab4ab.asm і при відсутності помилок мікроконтролер почне працювати за вже обновленим файлом LAB4AB.HEX.

Для керування симуляцією використовуються кнопки надані внизу ліворуч робочого листа (див. рис. 4.10):

"Play" – запуск симуляції або продовження призупиненої симуляції;

"Step" – виконати мінімальний крок програми мікроконтролера (як правило, це одна команда на асемблері). Цією кнопкою також можна почати симуляцію;

"Pause" – пауза симуляції. Можна продовжити симуляцію кнопками "Play" або "Step";

"Stop" – зупинка симуляції. Після цього симуляція почнеться з початку кнопками "Play" або "Step".

Рис. 4.10

Для того, щоб побачити всі параметри компонента поставте «галочку» напроти Edit all properties as text (рис. 4.8). При цьому відкриється вікно рис. 4.11. Наприклад, CLOCK (рис. 4.11) однозначно визначає тактову час-

тоту мікроконтролера при симуляції, а наведені на рис. 4.7 кварц і конденсатори для симуляції не потрібні – вони вводяться в схему на плату лише для того, щоб врахувати їх при розведенні друкованої плати. WDT_PERIOD - показує час, через який спрацює сторожовий таймер.

З рис. 4.11 видно, що мікроконтролер буде працювати за програмою, заданою файлом LAB4AB.HEX, який, в свою чергу, створений компілятором MPASM.

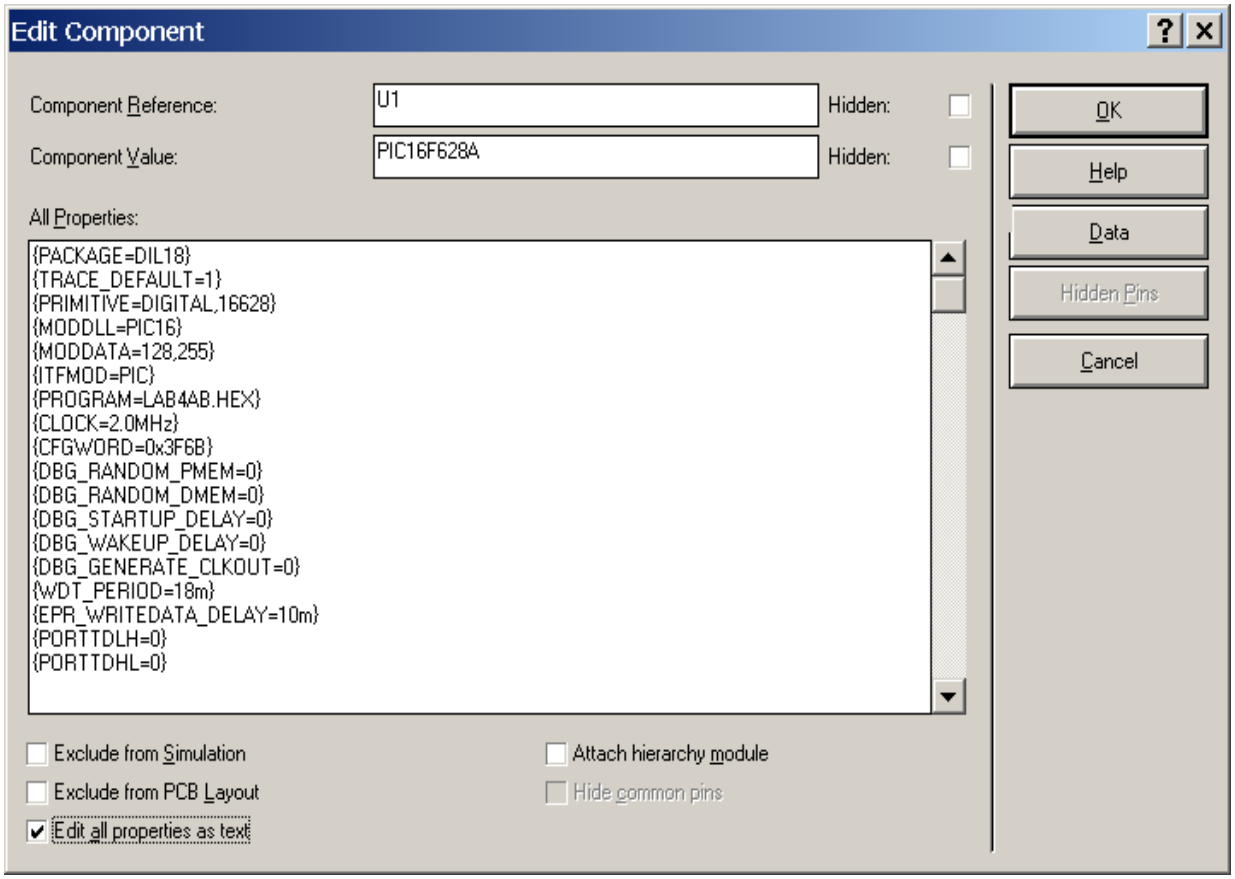


Рис. 4.11. Вікно редагування параметрів у текстовому режимі

Крім того, керувати процесом симуляції програми на асемблері можна за допомогою кнопок F10, F11, F12 та їх комбінацій з кнопками Alt і Ctrl або через меню DEBUG (рис. 4.12).

Виділена на рис. 4.12 команда дозволяє встановити час симуляції схеми (від початку до припинення процесу симуляції). Якщо вибрати цю команду, то виведеться поле, в якому можна задати вказаний час (при цьому для

одиниць виміру прийнято: n – наносекунди, u – мікросекунди, m – мілісекунди, секунди можно не позначати).

Процес симуляції схеми буде здійснюватись заданий час, якщо до його закінчення в програмі мікроконтролера не зустрінеться точка зупинки BreakPoint (*BP*) – симуляція зупиниться на ній.

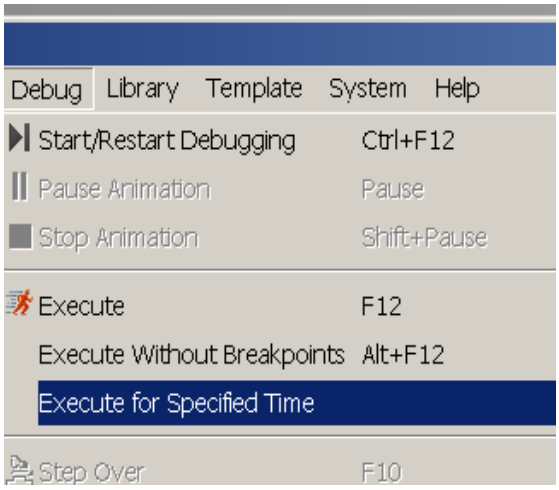


Рис.4.12. Вибір часу симуляції

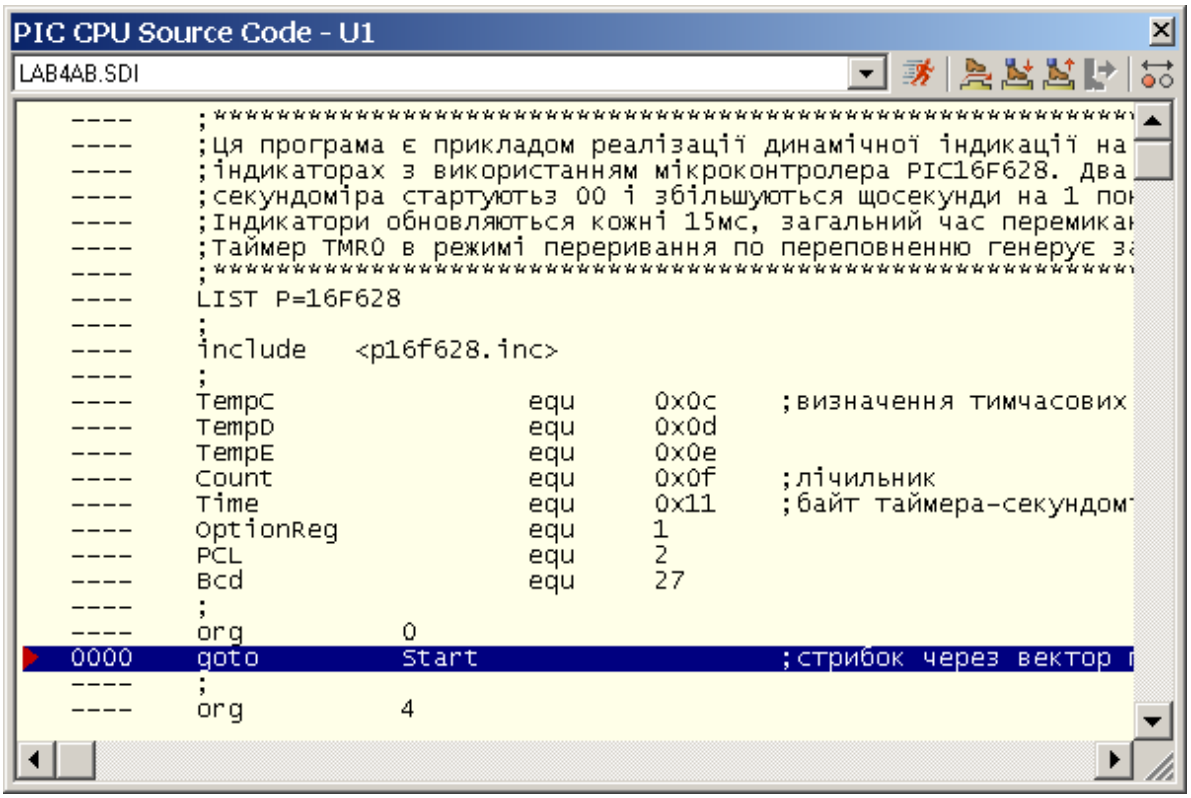


Рис.4.13. Вікно налагодження програми

Натиснення кнопки F12 (еквівалентно «клику» кнопки "Play") – запускає або продовжує симуляцію до *BP*. Alt+F12 - запускає або продовжує симуляцію, яка не зупиняється на *BP*. F10 –симуляція виконується без де-

талізації визначених підпрограм – вони будуть виконані за 1 крок. F11 – симуляція виконується з деталізації визначених підпрограм. Ctrl+F11 – перериває виконання визначених підпрограм. Ctrl+F10 – симуляція здійснюється до виділеного «кликом» миші рядка в програмі.

Якщо вказати файл з первинним текстом програми в меню Debug, то він з'явиться у вікні (рис. 4.13). Щоб встановити точку зупинки *BP* слід виділити необхідний рядок програми, «кликнувши» його мишею (рядок підсвічується темно-синім кольором), і натиснути кнопку F9. Програма зупиниться в режимі паузи, коли дійде до цієї точки. При цьому на екрані з'являться відповідні вікна з інформацією для налагодження. Якщо натиснути F9 ще раз, *BP* дезактивується (у вікні з'явиться відповідна інформація у вигляді червоного кола) і програма не зупинятиметься на ній. Наступний натиск кнопки F9 прибере *BP* з виділеного рядка. Крім того, в меню DEBUG є команда – «запустити симуляцію, ігноруючи будь-які *BP*».

Правою кнопкою миші у вікні первинного коду відкривається контекстне меню, через яке можна:

- побачити написану на мові С програму паралельно з відповідним їй кодом асемблера;
- ввімкнути відображення кодів операцій записаних на асемблері команд;
- ввімкнути нумерацію рядків програми;
- ввімкнути або вимкнути всі *BP*;
- виконати пошук в первинному тексті програми;
- перейти на необхідний рядок програми;
- поміняти шрифт у вікні.

Для спостереженням за вмістом регістрів мікроконтролера виберемо вікно слідкування Watch Window в меню Debug. Після налагодження воно не зникає з екрану ні при симуляції, ні при знаходженні в стані паузи. В цьо-

му вікні ми можемо розмістити регістри мікроконтролера і не лише відслідковувати їх вміст в ході виконання програми, але і задавати деякі умови і дії при виконанні цих умов, наприклад, зупинити симуляцію. Правою кнопкою миші у вікні рис. 4.14 викликаємо контекстне меню і додаємо регістри для спостереження: по їх назві або по адресі (рис.4.15). Якщо перед назвою регістру в Watch Window стоїть квадратик з плюсом, то можна розгорнути регістр на групи функціонально зв'язаних бітів.

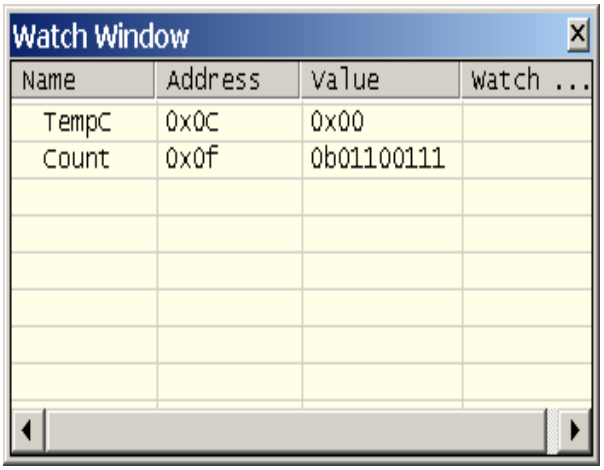


Рис. 4.14. Вікно регістрів SFR

При симуляції пристрою на екрані будемо спостерігати картинку анімації. Режим анімації можна встановити через меню *System>Set Annimation Options...*, коли відкривається вікно (рис. 4.16).

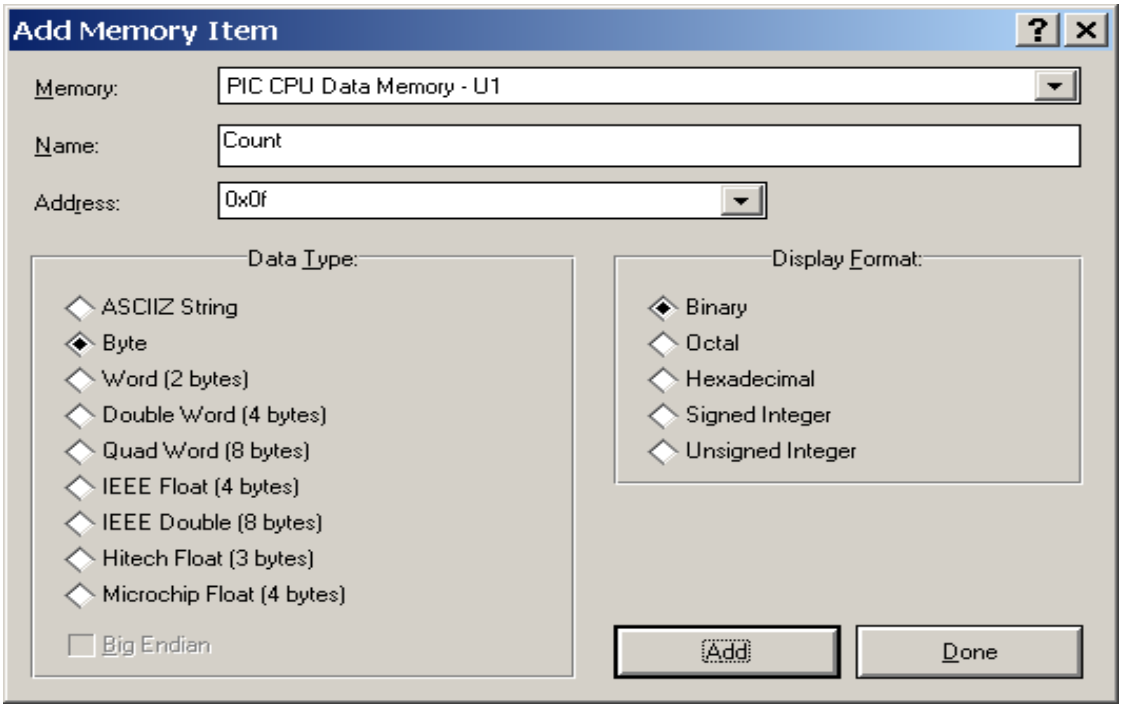


Рис. 4.15. Вікно вибору регістрів для спостереження за їх значенням

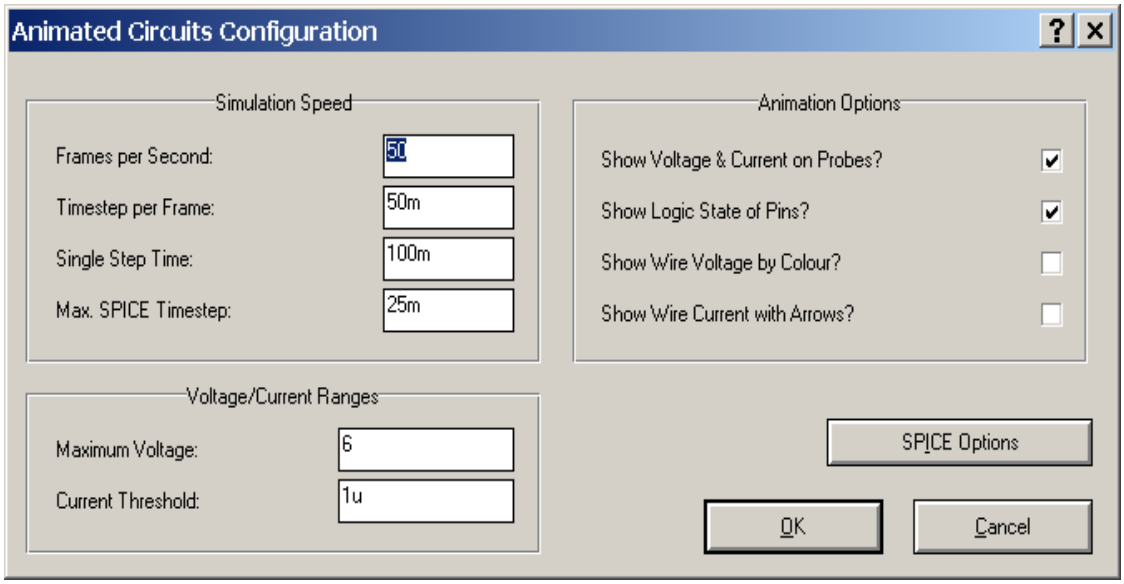


Рис.4.16.

Звичайний режим:

- екран оновлюється 20 разів на секунду;
- відображається напруга і струм на щупах A13 и A14;
- підсвічуються логічні рівні на виводах;
- показана кольором напруга на провідниках;
- показаний стрілками напрямок струму в провідниках.

При симуляції схеми динамічної індикації в рамках лабораторної роботи бажано збільшити число оновлень екрану до максимального значення – 30 разів на секунду (див. рис. 4.16). Максимальна напруга не обмежує напругу в схемі, яка симулюється. Вона вказується лише для того, щоб PROTEUS зміг вирішити, яким кольором показувати логічні рівні. Крок оцифрування струму 1 мкА. Для підвищення точності симуляції вказаний параметр оцифрування можна зменшити, що призведе до поліпшення інших параметрів, але при цьому збільшується необхідна обчислювальна потужність комп’ютера.

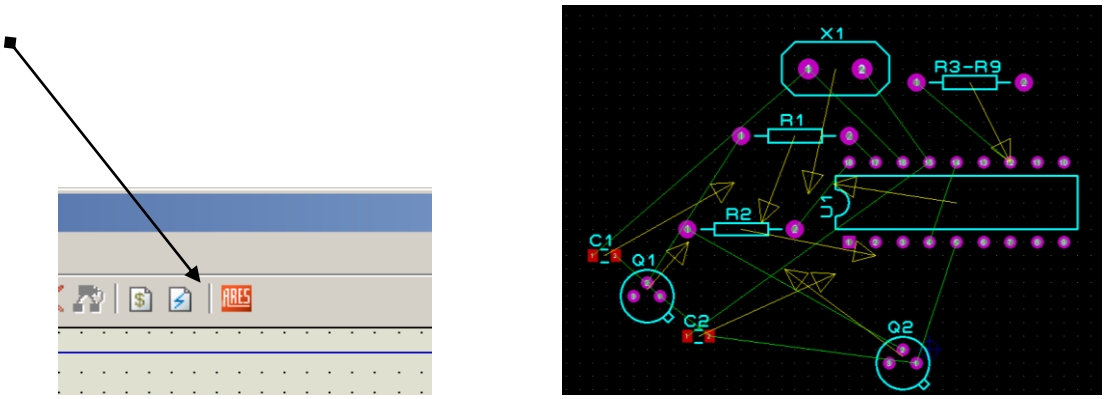
При анімації логічні рівні на виводах вказуються кольоровими квадратиками:

червоний – "гарячий" – логічна "1";

синій – "холодний" – логічний "0";

сірий – високоімпедансний вхід (Z-стан; для визначеності PROTEUS вважає, що на виводі є половина напруги живлення мікроконтролера).

Після виконання симуляції і налагодження роботи схеми, як правило, переходять до розведення друкованої плати, яке виконується за допомогою програми ARES Professional. Оскільки вказана програма входить в систему PROTEUS, то перехід із "перекиданням" даних проекту здійснюється натисканням лише однієї кнопки (рис.4.17).



а *б*

Рис. 4.17. Кнопка переходу в програму ARES з програми ISIS (*а*) та приклад розміщення елементів для розведення монтажної плати (б)

4.3. Завдання і порядок виконання роботи

В лабораторній роботі треба побудувати схему, додати програму (текст програми наведено нижче) для мікроконтролера і виконати симуляцію двохрозрядного таймера-секундоміра з динамічною індикацією. Знайти значення параметрів для отримання максимальної точності, пояснити роботу програми і особливості симуляції.

Завдання виконується в такій послідовності:

- Запустити програму PROTEUS, щоб відкрилось пусте вікно з полем

для розміщення схеми

- Вибрати елементи, які будуть використовуватись, із бібліотеки:
  - тип мікроконтролера - PIC16F628A
  - транзистори Q1 і Q2 NPN-типу (наприклад, 2N222)
  - малопотужні обмежувальні опори в базі транзисторів – 4.7 кОм
  - транзистори на виходах порту B – по 220 Ом
  - конденсатори схеми тактування по 1 нФ – керамічні
  - кварц частотою 2 МГц
  - два 7- сегментні індикатори із загальним катодом.
- Розмістити елементи на робочому полі і виконати необхідні з'єднання, користуючись при необхідності *Junction Dot* та схемою рис.4.6.

УВАГА!!! В симуляторі не передбачена автоматизація збереження змін і тому якнайчастіше робіть збереження схеми, яку будуєте

- Записати наведену нижче програму (без коментарів - коментарі на кирилиці в PROTEUS не завжди читаються) та приєднати її до проекту за допомогою меню *Source*.

Лістинг програми реалізації динамічної індикації
на базі 7-сегментних індикаторів

```
;*****************************************************************
; Ця програма є прикладом реалізації динамічної індикації на двох 7-сегментних
; індикаторах з використанням мікроконтролера PIC16F628.
; Два розряди таймера-секундоміра «стартують» зі значення 00 і збільшуються
; щосекунди на 1 поки не досягнуть 99.
; Індикатори оновляються кожні 15мс, загальний час перемикання - 30мс.
; Таймер TMR0 генерує затримку 15мс в режимі переривання по переповненню.
;*****************************************************************
    LIST P=16F628
    include   <p16f628.inc>
```

```
   TempC  equ  0x0c          ; визначення тимчасових регістрів
   TempD  equ  0x0d
   TempE  equ  0x0e
   Count  equ  0x0f          ; лічильник
   Time   equ  0x11          ; байт таймера-секундоміра
   OptionReg  equ 1
   PCL        equ 2
   Org  0
   Goto  Start               ; «стрибок» через вектор переривання
   Org  4
   goto ServiceInterrupts ; визів підпрограми обробл. переривання
Start
   call   InitPorts          ; ініціалізації портів
   call   InitTimers         ; ініціалізація таймера
loop
   goto   loop               ; безкінцевий цикл
InitPorts
   bsf    STATUS,RP0         ; вибір Bank1
   clrf   TRISA              ; задаємо RA0-5 вихідними
   clrf   TRISB              ; задаємо RB0-7 вихідними
   bcf    STATUS, RP0        ; вибір Bank0
   clrf   PORTA              ; передаємо на вихід PORTA 0
   clrf   PORTB              ; передаємо на вихід PORTB 0
   bsf   PORTA, 1            ; дозволяємо індикацію старшого розряду
   return
;тактова частота мікроконтролера 2.0Мгц. Використовуємо попередній подільник, на
;32 і тому TMR0 збільшується на 1 кожні 62.5мкс. Якщо TMR0 попередньо встановити
;в 16, то отримаємо затримку (256-16)*4/(2*32)=15мс до його переповнення,
```

;тобто програма через кожні 15 мс виходить на переривання.

```
InitTimers
   Clrf    Time              ; обнулення таймера-секундоміра
   Bsf     STATUS, RP0       ; вибір Bank1
   Movlw   B'10000100'       ; приєднати попередній подільник до TMR0
   Movwf   OptionReg         ; і задати коефіцієнт подільника 32
   Bcf     STATUS, RP0       ; вибір Bank0
   Movlw   B'00100000'       ; дозвіл переривання TMR0
   Movwf   INTCON
   Movlw   .16               ; попереднє завантаження TMR0
   Movwf   TMR0              ; старт TMR0
   retfie
ServiceInterrupts            ; підпрограма оброблення переривання
   btfsc INTCON, T0IF        ; переривання визване переповненням TMR0?
   Goto  ServiceTMR0         ; якщо так, то продовження підпрограми
   Movlw B'00100000'         ; обнуління  дозволів, крім заданого
   Movwf INTCON
   retfie
ServiceTMR0
   Movlw   .16               ; ініціалізація TMR0
   Movwf   TMR0
   bcf     INTCON, T0IF      ; скидання прапора переривання
   call    IncTimer    ; визов підпрограми збільшення таймера-секундоміра
   call    UpdateDisplay     ; обновлення індикаторів
   retfie
```

;показання індикаторів збільшуються кожну 66*15мс = 1 с.

```
IncTimer
   Incf    Count, W          ; збільшення значення лічильника Count
```

```
    Xorlw   .66             ; вже 66?
    btfsc   STATUS, Z       ; якщо ні, то пропуск наступної команди
    Goto    DoIncTime       ; інакше збільшуємо значення Count
    incf    Count, F
    return
DoIncTime
    Clrf    Count           ; обнуління Count
    Incf    Time, W         ; маємо молодший розряд
    Andlw   0x0F            ; маскування старшої тетради
    Xorlw   0x0a            ; молодша тетрада дорівнює 10?
    btfsc   STATUS, Z       ; якщо не дорівнює, то пропуск наступної команди
    Goto    IncMsd          ; збільшення наступного розряду
    Incf    Time, F         ; інакше збільшення таймера-секундоміра
    return
IncMsd
    Swapf   Time, W         ; переносимо старшу тетраду на місце молодшої
    Andlw   0x0F            ; маскуємо старшу тетраду
    Addlw   1               ; збільшуємо значення робочого регістру
    Movwf   Time            ; перезаписуємо отримане значення
    Swapf   Time, F         ;
    Xorlw   0x0a            ; тетрада рівна 10?
    btfsc   STATUS, Z       ; якщо не рівна, то пропуск наступної команди
    Clrf    Time
    return
UpdateDisplay
    Movf    PORTA, W        ; запис сигналів дозволу індикаторів в W
    clrf    PORTA           ; заборона всіх індикаторів
    andlw   0x0f
```

```
    movwf   TempC       ; збереження значення сигналів дозволу в TempC
    rrf     TempC, F    ; визначення наступного дозволеного розряду
    btfsc   STATUS, C   ; сигнал переповнення c=1?
    bsf     TempC, 1    ; якщо так, то дозволяємо старший розряд
    btfsc   TempC, 0    ; інакше перевіряємо молодший розряд
    Goto    UpdateMsd   ; якщо він 1, то виводимо значення індикатора Msd
UpdateLsd
    Movf    Time, W     ; значення молодшого індикатора в W
    Andlw   0x0f        ; маскування "зайвого" старшого індикатора
    Goto    DisplayOut  ; дозвіл відображення
UpdateMsd
    Call    ChkZero     ; перевірка, чи значимий розряд старшого індикатора?
    btfss   STATUS, Z   ; якщо він рівний нулю - пропуск наступної команди
    Swapf   Time,   W   ; значення старшого індикатора в W
    Andlw   0x0f        ; маскування "зайвого"
DisplayOut
    call    LedTable    ; формування зображення індикаторів
    movwf   PORTB       ; виведення значень в PORTB
    movf    TempC, W    ; перезапис сигналу дозволу індикатора в W
    movwf   PORTA       ; дозвіл індикатора
    return
; Таблиця перетворення двійково-десяткового коду в код семисегментного індикатора
LedTable
    addwf   PCL, F          ; додаємо до молодшого байта PC
    retlw   B'00111111'     ; перетворення для 0
    retlw   B'00000110'     ; перетворення для 1
    retlw   B'01011011'     ; перетворення для 2
    retlw   B'01001111'     ; перетворення для 3
```

```
   retlw     B'01100110'   ; перетворення для 4
   retlw     B'01101101'   ; перетворення для 5
   retlw     B'01111101'   ; перетворення для 6
   retlw     B'00000111'   ; перетворення для 7
   retlw     B'01111111'   ; перетворення для 8
   retlw     B'01101111'   ; перетворення для 9
   retlw     B'00000000'   ; гасіння індикатора
ChkZero                    ; перевірка старшого розряду на рівність 0
   movf      Time, W       ; запис лічильника-секундоміра в W
   andlw     0xf0
   btfss     STATUS, Z     ; якщо старший розряд рівний 0,то пропуск
   return                  ; якщо старший розряд не 0, то повернення
   retlw     .10           ; повернення з гасінням старшого розряду
      end
```

- Вивчити за допомогою коментарів алгоритм роботи записаного коду, відкривши програму на асемблері за допомогою блокнота, або іншого текстового редактора.
- Задати параметри елементів схеми, викликаючи контекстне меню на елементах.
- Тестувати програму за допомогою симулятора схеми.
- Отримати у викладача індивідуальне завдання для дослідження схеми. Розрахувати значення змінних, необхідних для реалізації заданих параметрів сигналів.

В даній лабораторній роботі студенту надається можливість на власний розсуд сформувати схему з компонентів, що доступні в бібліотеці PROTEUS, з подальшим виконанням всіх вказаних етапів. Робота закінчується пред’явленням складеної та налагодженої програми та розведеної друкованої плати (в електронному вигляді)

- Оформити протокол (аналогічно лабораторній роботі № 3)

Контрольні питання

1. Проаналізувати можливості інтерактивного симулятора PROTEUS.
2. Роз'яснити суть динамічної індикації.
3. Приведіть схему підключення двох семисегментних індикаторів до мікроконтролера.
4. Поясніть по лістингу алгоритм роботи мікроконтролера з динамічною індикацією.

Література до лабораторної роботи № 4

1. Product Guide Proteus design suite.
http://downloads.labcenter.co.uk/proteus7brochure.pdf

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. http://www.microchip.ru/lit/ Документація на продукцію Microchip
2. Катцен С. PIC-микроконтроллеры. Все что вам необходимо знать. М.: Додэка, 2008.- 656 с.
3. Яценков В.С. Микроконтроллеры MicroCHIP с аппаратной поддержкой USB. М.: Радио и связь, 2008.- 400 с.
4. Уилмсхерст Т. Разработка встроенных систем с помощью PIC: принципы и практические примеры. М.: МК-Пресс, 2008.- 544 с.
5. Заец Н.И. Радиолюбительские конструкции на PIC микроконтроллерах. М.: МК-Пресс, 2008.- 336 с.
6. Болл С.Р. Аналоговые интерфейсы микроконтроллеров.- М.: Додэка, 2007.-362 с.

7. Кохц Д. Измерение, управление и регулирование с помощью PIC-микроконтроллеров.- М.: МК-Пресс, 2006.- 304 с.
8. Яценков В.С. Микроконтроллеры MicroCHIP rfPIC со встроенным маломощным радиопередатчиком.- М.: Радио и связь, 2006. - 344 с.
9. Кёниг А, Кёниг М. Полное руководство по PIC-микроконтроллерам.- М.: МК-Пресс, 2006.- 256 с.
10. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров.- М.: МК-Пресс, 2006.- 400 с.
11. Яценков В.С. Микроконтроллеры MicroCHIP: Практическое руководство. Схемы, примеры программ, описание.- 2-е изд.. перераб.- М.: «Горячая линия – Телеком», 2005.- 280 с.
12. Ульрих В.А. Микроконтроллеры PIC16X7XX.- 3-е изд.- М.: «Солон-Пресс», 2005.- 320 с.
13. Предко М. Справочник по PIC-микроконтроллерам.- М.: «ДМК Пресс», 2004. - 512 с.
14. Таверньe К. PIC-микроконтроллеры. Практика применения: Справочник.- М.: «ДМК Пресс», 2004. - 272 с.
15. Интегрированная среда разработки MPLabIDE для микроконтроллеров PICmicro компании Microchip Technology Incorporated / Тех. документации DS51025D компании Microchip Technology Incorporated, USA. *http://www.microchip.ru:/files/d-sheets-rus/mplab_ide.pdf*
16. Руководство пользователя MPASM *http://www.microchip.ru:/files/d-sheets-rus/mpasm.pdf*
17. Справочник по всем микроконтроллерам PIC16 *http://www.microchip.ru:/files/d-sheets-rus/PIC16_Manual.pdf*
18. Product Guide «Proteus design suite» *http://downloads.labcenter.co.uk/proteus7brochure.pdf*

Додаток 1

Регістри, що використовуються при виконанні лабораторних робіт

Регістр INTCON

Регістр INTCON доступний як для запису так і для читання. В ньому зберігаються біти дозволів і прапорців переривань.

INTCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	RW-0	RW-0	R/W-x
GIE	**PEIE**	**TOIE**	**INTE**^{W}	**RBIE**	**TOIF**	**INTF**^{W}	**RBIF**

біт 7: **GIE:** Глобальний дозвіл переривань.

1 = дозволені всі не замасковані переривання

0 = всі переривання заборонені

біт 6: **PEIE:** Дозвіл переривань від периферійних модулів

1 = дозволені всі не замасковані переривання периферійних модулів

0 = переривання периферійних модулів заборонені

біт 5: **T0IE:** Дозвіл переривань по переповненню TMR0

1 = переривання дозволено

0 = переривання заборонено

біт 4: **INTE:** Дозвіл зовнішнього переривання INT

1 = переривання дозволено

0 = переривання заборонено

біт 3: **RBIE:** Дозвіл переривань по зміні сигналу на входах RB7:RB4 PORTB

1 = переривання дозволено

0 = переривання заборонено

біт 2: **T0IF:** Прапорець переривання по переповненню TMR0

1 = відбулося переповнення TMR0 (скидається програмно)

0 = переповнення TMR0 не було

біт 1: **INTF:** Прапорець зовнішнього переривання INT

1 = виконалась умова зовнішнього переривання на виході RB0/INT (скидається програмно)

0 = зовнішнього переривання не було

біт 0: **RBIF:** Прапорець переривання по зміні рівня сигналу на входах RB7:RB4 PORTB

1 = зафіксовано зміну рівня сигналу на одному з входів RB7:RB4 (скидається програмно)

0 = не було зміни рівня сигналу ні на одному з входів RB7:RB4

Регістр OPTION_REG

Регістр OPTION_REG доступний як для запису так і для читання.

- Попереднім подільником таймерів TMR0/WDT;
- Активним фронтом зовнішнього переривання RB0/INT;
- Резисторами на входах PORTB.

OPTION_REG

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	RW-1	RW-1	R/W-1
RBPU	**NTEDG**	**TOCS**	**TOSE**	**PSA**	**PS2**	**PS1**	**PS0**

біт 7: **RBPU**: Ввімкнення резисторів на входах PORTB

1 = підтягуючи резистори відімкнені

0 = підтягуючи резистори ввімкненні

біт 6: **INTEDG:** Вибір активного фронту сигналу на вході зовнішнього переривання INT

1 = переривання по передньому фронту сигналу

0 = переривання по задньому фронту сигналу

біт 5: **T0CS:** Вибір тактового сигналу для TMR0

1 = зовнішній тактовий сигнал з виходу RA4.T0CKI

0 = внутрішній тактовий сигнал CLKOUT

біт 4: **T0SE:** Вибір фронту приросту TMR0 при зовнішньому тактовому сигналі

1 = по задньому фронту сигналу (з високого до низькому рівня) на виході RA4.T0CKI

0 = по передньому фронту сигналу (з низького до високого рівня) на виході RA4.T0CKI

біт 3: **PSA:** Вибір ввімкнення попереднього подільника

1 = попередній подільник ввімкнений перед WDT

0 = попередній подільник ввімкнений перед TMR0

біти 2-0: **PS2 : PS0**: Встановлення коефіцієнта ділення попереднього подільника

Значення	000	001	010	011	100	101	110	111
Для TMR0	1:2	1:4	1:8	1:16	1:32	1:64	1:128	1:256
Для WDT	1:1	1:2	1:4	1:8	1:16	1:32	1:64	1:128

Регістр STATUS

В регістрі STATUS зберігаються прапорці стану АЛУ, прапорці причини зброду мікроконтролера і біти керування банками пам’яті даних.

STATUS

R/W-0	R/W-0	R/W-0	R-1	R-1	RW-x	RW-x	R/W-x
IRP	**RP1**	**RP0**	**-TO**	**-PD**	**Z**	**DC**	**C**

біт 7: **IRP**: Біт вибору банку при непрямій адресації

1 = банк 2, 3(100h-1FFh)

0 = банк 0.1 (000h - 0FFh)

біти 6-5: **RP1:RP0**: Біти вибору банку при прямій адресації

11 = банк3 (180h - 1FFh)

10 = банк2 (100h - 17Fh)

01 = банк1 (080h - 0FFh)

00 = банк0 (000h - 07Fh)

біт 4: **-TO**: Прапорець переповнення сторожового таймера

1 = після POR або виконання команд CLRWOT, SLEEP

0 = після переповнення WDT

біт 3: **-PD**: Прапорець ввімкнення живлення

1 = після POR або виконання команди CLRWDT

0 = після виконання команди SLEEP

біт 2: **Z**: Прапорець нульового результату

1 = нульовий результат виконання арифметичної або логічної операції

0 = не нульовий результат виконання арифметичної або логічної операції

біт 1: **DC**: Прапорець десяткового переносу/позики (для команд ADDWF, ADDWL, SUBWF, SUBWL),

позика має інверсне значення

1 = був перенос з молодшої тетради байта

0 = не було переносу з молодшої тетради байта

біт 0: **C**: Прапорець переносу/позики (для команд ADDWF, ADDWL. SUBWF, SUBWL),

позика має інверсне значення

1 = був перенос з молодшого байту

0 = не було переносу з молодшого байту