МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ
ННІ «Дніпровський інститут інфраструктури і транспорту»

Кафедра «Автоматика та телекомунікації»

В авторській редакції

ЦИФРОВІ СИСТЕМИ КЕРУВАННЯ

Навчально–методичні рекомендації до проведення практичних занять

Електронне видання

ДНІПРО
2025

УДК 004:656.05
П 84

Автор:
Володимир Профатилов

Електронне видання

Схвалено Групою забезпечення якості освітньої програми
G7.1.05 «Автоматика та автоматизація на транспорті»
Протокол № 1 від 28.08.2025

Схвалено Групою забезпечення якості освітньої програми
J7.1.02 «Системи керування рухом поїздів»
Протокол № 1 від 29.08.2025

П 84 Профатилов В. І. Цифрові системи керування: навчально – методичні рекомендації до проведення практичних занять / В. І. Профатилов; Укр. держ. ун–т науки і техноло-гій. – Електрон. вид. – Дніпро: УДУНТ, 2025. – 64 с.

Навчально–методичні рекомендації призначені для виконання практичних занять студентами, які здобувають освітній ступінь «бакалавр» по спеціальностям G7 «Автоматизація, комп'ютерно–інтегровані технології та робототехніка» (ОП Автоматика та автоматизація на транспорті) та J7 «Залізничний транспорт» (ОП Системи керування рухом поїздів) денної форми навчання.

Навчально–методичні рекомендації містять 9 практичних занять, у яких розглянуті робота з інтегрованим середовищем розробки програмного забезпечення для мікроконтролерів MPLAB X IDE компанії Microchip та розробка проектів цифрових систем керування в САПР «Proteus Design Suite», система команд мікроконтролерів PIC18, а також приклади програмування на PIC – асемблері та на мові Сі найбільш типових задач для мікроконтролерів. Навчально–методичні рекомендації містять основні теоретичні відомості, порядок виконання практичних занять та завдання на самостійну роботу.

Іл. 11. Табл. 3. Бібліогр. 8 назв.

ЗМІСТ

ВСТУП

Метою вивчення дисципліни є засвоєння принципів проектування та експлуатації цифрових систем керування на базі сучасних мікроконтролерів, а також отримання навиків розробки прикладного програмного забезпечення для вбудованих систем використовуючі мови асемблеру та Сі.

Студенти, що навчаються за освітньою програмою «Автоматика та автоматизація на транспорті», в результаті навчання по дисципліні отримають компетентності та будуть вміти використовувати для вирішення професійних завдань новітні технології у галузі автоматизації, робототехніки та комп'ютерно-інтегрованих технологій, зокрема, проектування багаторівневих систем керування та візуалізації інформації за допомогою засобів людино-машинного інтерфейсу на базі мікроконтролерів. Обґрунтовувати вибір технічної структури та вміти розробляти прикладне програмне забезпечення для мікропроцесорних систем керування на базі мікроконтролерів. Вміти вільно користуватись сучасними комп'ютерними та інформаційними технологіями для вирішення професійних завдань, програмувати та використовувати прикладні та спеціалізовані комп'ютерно-інтегровані середовища для вирішення задач автоматизації, робототехніки та зв'язку

Студенти, що навчаються за освітньою програмою «Системи керування рухом поїздів», в результаті навчання по дисципліні отримають компетентності та будуть мати навики розробляти та впроваджувати засоби автоматизації систем керування рухом поїздів, пристроїв залізничної автоматики та їх елементів, а також застосовувати сучасні програмні засоби для розробки проектно–конструкторської та технологічної документації зі створення, експлуатації, ремонту та обслуговування систем керування рухом поїздів, пристроїв залізничної автоматики та їх елементів.

Практичні заняття нададуть студентам можливість отримати практичні навики програмування на асемблері та мові Сі для мікроконтролерів сімейства PIC18, тестування цифрових систем керування за допомогою сучасних апаратних та програмних засобів, а також навчитися розробляти прикладне програмне забезпечення для систем автоматики та робототехніки. Практичні заняття передбачають роботу в спеціалізованій лабораторії та можливість користуватися сучасними спеціалізованими програмними й апаратними засобами проектування цифрових систем керування.

Вимоги до оформлення звіту з практичних занять:

– звіт можна оформляти у рукописному вигляді в зошиті або друкованому вигляді за допомогою комп'ютерної техніки на листах формату А4 при цьому необхідно дотримуватися вимог оформлення, що наведені у додатку А;

– титульний лист звіті повинен містити інформацію, яка однозначно ідентифікує його виконавця: назву дисципліни та вид занять, прізвище та ім'я студента, номер академічної групи, номер шифру або варіанту (якщо це є у вимогах до практичних занять);

– звіт повинен включати номер, тему та мету практичних занять;

– структура звіту повинна відповідати вимогам розділу практичних занять «Зміст звіту».

Практичне заняття № 1

ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ MPLAB X IDE

Мета заняття

Вивчити основні можливості інтегрованого середовища розробки MPLAB X IDE компанії Microchip, а також етапи розробки проекту на асемблері для PIC–мікроконтролерів.

1. Короткі теоретичні відомості

MPLAB X IDE – це безкоштовне інтегроване середовище розробки програмного забезпечення (ПЗ) для вбудованих систем на базі мікроконтролерів (MCU), цифрових сигнальних контролерів (DSC) і мікропроцесорів (MPU) компанії Microchip. MPLAB X IDE це набір програмних модулів, що дозволяють писати, редагувати та компілювати програмний код на мові асемблера або на мові Сі, тестувати ПЗ та програмувати мікроконтролери (МК). MPLAB X IDE розроблено на базі крос–платформи Netbeans, і може працювати під керуванням операційних систем Windows (Windows 7 і вище), Linux (Ubuntu 9.10 і вище) і Mac OS X 10.11 Intel. MPLAB X IDE підтримує роботу із внутрішньосхемними відладчиками–програматорами MPLAB ICD3 або ICD4, програматорами–відладчиками MPLAB Pickit 3, Pickit 4 і Snap, апаратним емулятором MPLAB REAL ICE, а також промисловим програматором MPLAB PM3. [1]

Інсталяцію інтегрованого середовища розробки MPLAB X IDE можна безкоштовно скачати з сайту компанії Microchip. [9]

Основні функції MPLAB X IDE [1]:

– створення й редагування вихідних текстів програмного забезпечення PIC–мікроконтролерів;

– об’єднання файлів в один проект і компіляція проекту в hex–файл;

– програмне або апаратне тестування програмного забезпечення PIC–мікроконтролерів;

– програмування PIC–мікроконтролерів.

MPLAB X IDE складається із програмних модулів, об’єднаних у єдине середовище розробки [1]:

– менеджер проектів (верхнє ліве вікно – закладки Projects, Files і Classes) – використовується для створення й роботи з файлами проекту, а також дозволяє поєднувати вихідні файли мовою асемблера або Сі, об’єктні файли, бібліотеки й файли сценаріїв в одному проекті;

– текстовий редактор – дозволяє набирати й редагувати вихідний текст програми на асемблері або мові Сі з підсвічуванням синтаксису, шаблони й файли сценарію компоновщика;

– компілятори для мови Сі XC8, XC16, XC32 із можливістю використання асемблера, компоновщика (лінкера) і менеджера бібліотек, що дозволяють одержати вихідний hex–файл проекту;

– програмний симулятор MPLAB X Simulator – дозволяє моделювати виконання програми в PIC мікроконтролері;

– програматор – використовується для прошивання FLASH–ПЗП кодом програми та енергонезалежної пам’яті даних EEPROM вихідними даними, а також для стирання, верифікації й читання програми із ПЗП мікроконтролера.

2. Етапи розробки програмного забезпечення на асемблері для PIC–мікроконтролера

1 етап: створення нового проекту

Створення нового проекту складається з шести кроків (деякі кроки можуть пропускатися автоматично) [1]:

1. Викликати майстер створення нового проекту: File / New Project або нажати швидку кнопку (New Project…):

– вибрати категорію проекту: Microchip Embedded;

– вибрати тип проекту: Application Project(s);

– натиснути кнопку Next.

2. Вибрати тип мікроконтролера (Select Device):

– вибрати сімейство (Family): Advanced 8–bit MCUs (PIC18);

– вибрати тип мікроконтролера (Device): PIC18F4520;

– вибрати інструменти для тестування (Tool): Simulator;

– натиснути кнопку Next.

3. Вибір з’єднувача для інструментів відладки (Select Header):

– пропустити даний крок;

– натиснути кнопку Next.

4. Вибір типу інтерфейсу для плати відладки або програмуваня (Select Plugin Board):

– пропустити даний крок;

– натиснути кнопку Next.

5. Вибрати тип компілятора (Select Compiler):

– вибрати асемблер: pic–as (v2.50);

– натиснути кнопку Next.

6. Вибір імені проекту та каталогу для зберігання файлів проекту (Select Project Name and Folder):

– ввести ім’я проекту в поле «Project Name»: pz1. Усі імена проекту, файлів, і папок повинні бути тільки англійською мовою, інакше програма може не компілюватися;

– вибрати папку для зберігання проекту «Project Location». Нажати кнопку «Browse...» і вибрати папку для розміщення файлів проекту. Якщо папка для зберігання файлів проекту не існує, то необхідно нажати кнопку «Create New Folder» і створити папку;

– встановити проект як основний: встановити пункт – Set as main project;

– встановити пункт – Use project location as the project folder (якщо цей пункт не встановити, то усередині папки проекту буде створений додатковий каталог «ім’я проекту.X», і в ньому будуть зберігатися всі необхідні файли проекту);

– для підтримки кирилиці в коментарях необхідно вибрати тип кодової сторінки: Encoding – window–1251;

– натиснути кнопку Finish.

2 етап: створення нового вихідного файлу

1. Викликати меню: File / New File або нажати швидку кнопку  (New File…).

2. Вибрати категорію файлу: Assembler;

– вибрати тип файлу: Assemblyfile.s;

– натиснути кнопку Next.

3. Ввести ім’я файлу в поле «File Name»: pz1. Усі імена файлів і папок повинні бути тільки англійською мовою, інакше програма може не компілюватися. За замовчуванням файл розміщається в основному каталозі проекту, але можна зберегти й в іншій папці. Для цього необхідно натиснути кнопку «Browse...» і вибрати папку для розміщення файлу проекту.

4. натиснути кнопку Finish.

5. Набрати в текстовому редакторі, що відкрився (праве верхнє вікно) текст програми на асемблері PIC18 (коментарі можна не набирати). При наборі ідентифікаторів (імена змінних, констант та міток) треба враховувати, що регістр символів повинен мати однакове значення у всій програмі (як у мові Сі). Команди можна набирати як великими, так і маленькими символами:

```
/* Цифрові системи керування
Тестова програма для практичного заняття №1 */
PROCESSOR 18F4520 // тип мікроконтролера
#include <xc.inc> // підключення опису МК PIC18
// визначення бітів конфігурацій мікроконтролера
// визначення бітів конфігурацій мікроконтролера
CONFIG WDT = OFF // вимикання сторожового таймера WDT
CONFIG OSC = XT ; вибір XT–генератора (4 МГц)
// оголошення глобальних змінних
GLOBAL X1,X2,SUM
// оголошення констант
K1 EQU 33h
K2 EQU 0x22
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
X1: DS 1 ;резервуємо в ОЗП 1 байт для змінної X1
X2: DS 1 ;розміщуємо змінну X2 у комірці пам'яті
SUM: DS 1 ;розміщуємо змінну SUM у комірці пам'яті

; початкова адреса програми після скидання МК
```

```
PSECT resetVec, abs, class=CODE, reloc=2
    ORG 0
resetVec:
; ініціалізація змінних
    MOVLW K1; WREG=33h
    MOVWF X1,c; помістити в x1=33H
    MOVLW K2; WREG=22h
    MOVWF X2,c; помістити в x2=22H
; обчислення суми SUM = X1+X2
    MOVF X1,W,c; WREG<=X
    ADDWF X2, W,c; WREG=WREG+X2
    MOVWF SUM,c; SUM<=WREG
;запис суми в порт C і його інверсія
    CLRF TRISC,c; налаштування порта PORTC на вивід
    MOVFF SUM, PORTC; записуємо SUM=>PORTC
    NOP; холоста команда (пауза 1 МЦ = 1мкс)
    COMF PORTC,c; інверсія порту PORTC
; зациклення програми
    GOTO $; нескінченний цикл
    END resetVec; кінець програми (точка входу в програму)
```

6. Зберегти файл із текстом програми: File / Save або натиснути швидку кнопку  (Save All)..

7. Для того щоб підключити файл із вихідним текстом до проекту необхідно натиснути праву клавішу миші у вікні менеджера проекту (виділивши папку «Source Files»), і вибрати контекстне меню «Add Existing Item...», вибрати файл із вихідним текстом і нажати кнопку «Select». Ім’я файлу з’явиться у вікні проекту в каталозі «Source Files».

Якщо необхідно вилучити файл із проекту, то необхідно натиснути праву кнопку миші на даному файлі й вибрати контекстне меню «Remove From Project».

3 етап: компіляція проекту

Для компіляції проектів, що написані на мові асемблера в середовища розробки MPLAB X IDE використовується MPLAB XC8 PIC Assembler (pic–as), що представляє собою автономний кросс–асемблер і компоновщик, який підтримує всі 8–розрядні PIC–мікроконтролери компанії Microchip.

1. Налаштування режимів компіляції:

– якщо необхідно змінити якісь параметри або властивості проекту, то необхідно натиснути праву клавішу миші у вікні менеджера проекту на імені проекту й вибрати контекстне меню «Properties», або викликавши меню: File / Project Properties (ім’я проекту), або натиснути в панелі навігатора (нижнє ліве вікно – закладка Dashboard) кнопку  (Project Properties), або натиснути ліву клавішу миші у вікні Dashboard на імені пакета Packs – PIC18Fxxxx_DFP(1.6.159);

– для налаштування опцій компілятора pic–as необхідно вибрати категорію «pic–as Global Options». Якщо редагування опції заблоковане, то необхідно нажати кнопку «Unlock». Після редагування опції необхідно зберегти зміни, для цього необхідно натиснути кнопку «Apply» і кнопку «OK»;

– вибрати категорію «pic–as Global Options» та ввести в полі «Additional options» додаткові параметри компіляції: –Wa,–a;

– натиснути кнопку «Apply»;

– – вибрати категорію «pic–as Assembler» та перевірити, щоб опція «Assembly files with preprocessor» була вибрана. Якщо при компіляції файлу на асемблері не запускати препроцесор, то деякі директиви асемблера не будуть працювати;

– натиснути кнопку «OK».

2. Виконати компіляцію проекту:

– вибрати меню: Production / Build Main Project або нажати клавішу F11 або нажати швидку кнопку  .

Якщо проект був розроблений на одному комп'ютері, а потім переноситися на інший комп'ютер, або в інший каталог, то він може видавати помилки при компіляції. Це пов'язано з тим, що на іншому комп'ютері можуть використовуватися інші налаштування або шляхи до файлів проекту. У такому випадку для першої компіляції краще використовувати меню Production / Clean and Build Main Project або нажати клавішу SHIFT+F11 або нажати швидку кнопку  . Даний спосіб компіляції видаляє всі старі файли проекту (крім вихідних файлів) і створює нові, але такий спосіб компіляції займає більше часу.

3. Для перегляду результатів компіляції необхідно в панелі завдань (нижнє праве вікно) відкрити закладку «Output»:

– якщо при компіляції проекту помилок не буде виявлено, то буде видане повідомлення «BUILD SUCCESSFUL», і буде створений вихідний файл із розширенням «hex». Машинний код, який зберігається в даному файлі, можна використовувати для налаштування програми, і завантаження в ПЗП мікроконтролера за допомогою програматора;

– якщо при компіляції проекту будуть виявлені помилки, то буде видане повідомлення «BUILD FAILED», а в рядку «ERROR» буде зазначений тип помилки й номер рядка, у якому ця помилка виявлена. Після виправлення помилок, необхідно повторити компіляцію.

4 етап: тестування програми

Програмний симулятор MPLAB X Simulator дозволяє тестувати ПЗ розроблене на асемблері або мові Сі для вбудованих систем на базі мікроконтролерів і цифрових сигнальних контролерів компанії Microchip [1].

Програмний симулятор дозволяє виконувати наступні операції:

– виконувати програму в автоматичному або покроковому режимах;

– установлювати точки зупинки виконання програми (Window / Debugging / Breakpoints);

– переглядати вміст пам’яті програм, енергонезалежної пам’яті даних EEPROM, пам’яті даних і регістрів спеціальних функцій (Window / Target Memory Views);

– контролювати значення змінних користувача (Window / Debugging / Watches);

– контролювати вміст апаратного стека (Window / Target Memory Views / Hardware Stack);

– контролювати час виконання програми і її окремих частин (Window / Debugging / Stopwatch);

– імітувати подачу на входи мікроконтролера зовнішніх логічних сигналів (Window / Simulator / Stimulus);

– виконувати скидання мікроконтролера (Debug / Reset).

Провести тестування програми:

1. При використанні симулятора MPLAB X Simulator необхідно встановити частоту машинних циклів (МЦ). Наприклад, при частоті внутрішнього ГТІ Fosc = 4 MHz (МГц), частота МЦ для МК PIC16 і PIC18 буде в 4 рази менше Fcyc = 1 MHz (1 мкс):

– натиснути в панелі навігатора (нижнє ліве вікно – закладка Dashboard) кнопку  (Project Properties) або ліву клавішу миші у вікні Dashboard на імені пакета Packs – PIC18Fxxxx_DFP(1.6.159);

– вибрати категорію «Simulator» і встановити частоту МЦ (Instruction Frequency) – 1 Mhz (1 МГц);

– натиснути кнопку «Apply» і кнопку «OK».

2. Запустити програмний симулятор на виконання програми:

– поставити точку зупинки програми (Breakpoint) на першій команді програми (Line 30 – для файлу «pz1.s», набраного з коментарями): нажати ліву кнопку миші на номері рядка першої команди програми. На місці номера рядка повинен з’явитися червоний квадрат. Це необхідно тому що при запуску програми вона запускається в автоматичному режимі;

– вибрати меню: Debug / Debug Main Project або нажати швидку кнопку  ;

3. Перед виконанням програми виконується скидання мікроконтролера, також скидання здійснюється, якщо потрібно повторити виконання програми: меню «Debug / Reset» або кнопка  .

4. Вивести на екран о вікно тестування для перевірки роботи програми:

– викликати меню: Window / Debugging / Watches.

5. Налаштувати вікно спостереження «Watches»:

– натиснути праву клавішу миші у вікні «Watches» і викликати контекстне меню «New Watch»;

– вибрати у вікні «New Watch» кнопку «Global Symbols», потім вибрати ім’я змінної X1 (воно повинне з’явиться в рядку «Enter or Select Watch Expression») і натиснути кнопку «OK»;

– вивести також у вікно спостереження «Watches» змінні X2 і SUM;

– вибрати у вікні «New Watch» кнопку «SFR's» і вибрати регістри WREG, TRISC, PORTC і додати їх у вікно «Watch» натискаючи кнопку «OK».

6. Виконати програму в покроковому режимі: меню Debug / Step Into або натискаючи клавішу F7 або натиснути кнопку  .

7. Записати отримані результати з вікна «Watches» у звіт. Вікно «Watches» можна зберегти натиснувши кнопку «Export all watches to list file» (ліва панель кнопок у вікні «Watch»).

8. Для виходу з режиму тестування програми необхідно викликати меню Debug / Finish Debugger Session або натиснути кнопку  .

9. Якщо необхідно вручну налаштувати біти конфігурації мікроконтролера, то необхідно викликати меню: Window / Target Memory Views / Configuration Bits або Production / Set Configuration Bits. Наприклад, можна перевірити роботу директив CONFIG:

– поле (Field) OSC (адреса 300001) – установити опцію (Option) – XT;

– поле WDT (адреса 300003) – установити опцію – OFF;

– поле PBADEN (адреса 300005) – установити опцію – OFF;

– інші поля можна не налаштовувати, тому що їхнє значення в програмному симуляторі не буде впливати на тестування програми. Після налаштування дане вікно можна закрити.

3. Зміст звіту

1. Основні функції MPLAB X IDE.
2. Структура програмних модулів MPLAB X IDE.
3. Етапи розробки програмного забезпечення на асемблері для PIC–мікроконтролера.
4. Програма на асемблері з коментарями.
5. Результати виконання програми (значення змінних і регістрів, які використовувалися в програмі в 16, 10 і двійковому форматі).

4. Контрольні запитання

1. Призначення інтегрованого середовища розробки MPLAB X IDE.
2. Можливості інтегрованого середовища розробки MPLAB X IDE.
3. Мета компіляції проекту в MPLAB X IDE.
4. Можливості програмного симулятора MPLAB X Simulator.
5. Призначення вікна спостереження «Watches».
6. Основні кроки створення нового проекту в MPLAB X IDE.
7. Призначення менеджера проектів в MPLAB X IDE.
8. Призначення програматора в MPLAB X IDE.
9. Який тип файлу використовується для зберігання машинного коду в MPLAB X IDE.

Практичне заняття № 2

АСЕМБЛЕР ДЛЯ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ PIC18

Мета заняття

Вивчити основні можливості асемблера для програмування мікроконтролерів (МК) PIC18, а також створити шаблон файлу на асемблері для використання в подальших проектах на базі PIC–мікроконтролера.

1. Короткі теоретичні відомості

MPLAB XC8 PIC Assembler – це автономний крос–асемблер і компонувальник, що підтримує всі 8–бітні PIC–мікроконтролери компанії Microchip та дозволяє створювати й компілювати вихідний текст програми, написаний повністю на асемблері. PIC Assembler встановлюється одночасно з компілятором мови Сі MPLAB XC8, при цьому мова асемблера є загальною для обох інструментів. Асемблер працює з багатьма іншими інструментами компанії Microchip, включаючи інтегроване середовище розробки MPLAB X IDE і симулятор MPLAB X Simulator [2].

Асемблер для програмування PIC–мікроконтролерів включає в себе [2]:

1. Компілятор асемблера (pic–as), який використовується для генерації об'єктного коду, тобто файлу з розширенням «name.o», який потім може використовуватися компонувальником для формування вихідного HEX–файлу.

2. Компонувальник (hlink) дозволяє виконати складання проекту шляхом об'єднання об'єктних файлів, які генерують компілятори асемблера (PIC–AS) або мови Сі (XC8), а також бібліотек у вихідний HEX–файл машинного коду, який потім може бути завантажений безпосередньо в пам'ять програм МК. Даний метод дозволяє багаторазово використовувати протестовані модулі та бібліотеки, приховуючи при цьому механізм їх реалізації для захисту інтелектуальної власності.

3. Архіватор і менеджер бібліотек (xc8–ar) дозволяє об'єднувати кілька об'єктних файлів в один бібліотечний файл, у якому дані зберігаються в за архівованому виді й тому займають менше місця на диску.

PIC Assembler для програмування PIC–мікроконтролерів підтримує наступні типи файлів [2]:

– s, S, asm – файли з вихідним текстом програми на мові асемблера, які можуть бути створені у будь–якому текстовому редакторі;

– hex – файл із машинним кодом програми (код у шістнадцятковому форматі), який за допомогою програматора може бути завантажений в пам'ять програм МК;

– lst – файл абсолютного лістингу, у якому зберігається вихідний текст програми разом з машинними кодами команд, а також інформація про помилки й попередження, що були сформовані при компіляції проекту;

– inc – підключаємий файл, у якому зберігаються описи для певного типу PIC–мікроконтролера;

– mac – файл із набором готових макросів;

– a – бібліотечні файли;

– o – об’єктний файл програми, у якому формується переміщуваний код;

– map – карта розподілу пам’яті для даного проекту;

– elf – додатковий файл для тестування програми;

– sbs – файл імітації вхідних сигналів на мові SCL, що використовується для тестування програмного забезпечення в програмному симуляторі MPLAB X Simulator;

– c – файли з вихідним текстом програми на мові Сі, який може бути створений у будь–якому текстовому редакторі;

– h – файли з заголовками для програми на мові Сі.

Алфавіт PIC Assembler

Для створення ідентифікаторів (імен змінних, констант і міток) можна використовувати латинські букви (A–Z, a–z), цифри (0–9), знак підкреслення «_», знак питання «?» і знак долара «$». Асемблер чутливий до регістру символів, крім команд МК і директив асемблера (їх можна набирати як великими, так і маленькими буквами). Першим символом будь–якого ідентифікатора може бути тільки латинська буква або знак долара. Не можна використовувати в якості ідентифікаторів імена команд мікроконтролерів і директив асемблера.

Формати представлення чисел в асемблері.

Асемблер для мікроконтролерів PIC18 підтримує чотири системи числення й символи в системі ASCII:

– шістнадцяткова система числення:

1 спосіб – 25H (25h);

2 спосіб – 0x25;

– двійкова система числення: 10101010B (дозволяється використовувати тільки велику букву);

– десяткова система числення:

1 спосіб – 25;

2 спосіб – 25D (25d);

– вісімкова система числення: 765Q (765q).

– символи ASCII задаються двома способами:

– одиночні символи – 'F';

– рядок символів: – 'QWER' або "QWER".

Команда на мові асемблера PIC18 займає один рядок і може складатися із чотирьох полів:

[мітка:] команда [операнди] [; коментарі]

– у якості роздільника полів може використовуватися будь–яка кількість пробілів, табуляцій і перевід рядка;

– **1 поле – мітка:** – необов’язкове поле. Мітка повинна обов’язково починатися з першої колонки й закінчуватися двокрапкою;

– **2 поле – команда** – обов’язкове поле, у якому розміщується мнемонічне позначення (мнемоніка) команди PIC–мікроконтролера, директиви асемблера або

макрокоманди. Командне поле бажано починати із другої або більшої колонки, щоб не плутати з мітками;

– **3 поле – операнди** – необов'язкове поле. Операнди відділяються один від одного комами. Кількість операндів залежить від типу команди або директиви. У деяких командах асемблера, якщо операнд не зазначений, то при компіляції автоматично підставляється операнд за замовчуванням;

– **4 поле – коментарі** – необов'язкове поле, яке починається із символу крапка з комою «;». У коментарях можна використовувати будь–який текст, у тому, числі й українською мовою, так як все, що йде після символу «;» ігнорується компілятором. Також в PIC Assembler можна використовувати коментарі в стилі мови Сі: // – однорядковий коментар, /* */ – багаторядковий коментар.

Директиви асемблера PIC Assembler [2].

Директиви асемблера PIC Assembler безпосередньо не включаються у вихідний машинний код, але вони використовуються для керування процесом компіляції проекту:

– керують створенням розділів для компіляції коду й створення об'єктного файлу;

– управляють розподілом пам'яті й призначенням символічних імен змінним і константам;

– визначають формат і состав файлу лістингу;

– управляють роботою макросів.

1. Директива PROCESSOR – вказує тип мікроконтролера, для якого буде виконуватися компіляція вихідного тексту в машинний код. Приклад використання директиви:

PROCESSOR 18F4520; компілювати програму для МК PIC18F4520

2. Директива #include – вставляє текст із зазначеного файлу у вихідний текст програми. Приклад використання директиви:

#include <xc8.inc> // підключити файл із описами для PIC–мікроконтролера

3. Директива CONFIG – установка біт конфігурації для мікроконтролера. Приклад використання директиви:

CONFIG WDT = OFF // вимикання сторожового таймера WDT
CONFIG OSC = XT ; вибір XT–генератора (4 МГц)

4. Директива PSECT – об'являє або відновлює розділ програми. Усі дані й код програми повинні бути розміщені в одному з розділів за допомогою даної директиви. Структура директиви:

psect ім'я_розділу, [список прапорів]

Приклади використання директиви:

```
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
X1: DS 1 ; резервуємо в ОЗП 1 байт для змінної X1

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
```

```
ORG 0
resetVec:
    GOTO main

// розділ розміщення констант у пам’яті програм (ПЗП)
PSECT myconst, abs, class=CONST, delta=1
ORG 400h
    DB 1, 2, 3, 4, 5, 6

// розділ розміщення даних в енергонезалежній пам’яті даних (EEPROM)
PSECT mySetting, class=EEDATA, delta=1
    DB 'A', "String", 0
```

5. Директива END – кінець вихідного тексту програми на асемблері. Усе, що буде йти після даної директиви, буде проігноровано компілятором. Приклад використання директиви:

END resetVec; кінець програми (точка входу в програму resetVec)

6. Директива ORG – установка початкової адреси розділу програми. Команди або дані, які йдуть після директиви, розміщуються в пам’яті МК починаючи із зазначеної в директиві адреси. Директива буде працювати тільки в розділах програми із прапором abs. Приклад використання директиви:

ORG 10h; розмістити код починаючи з адреси 10h

7. Директива RADIX – задає систему числення за замовчуванням: hex, dec, oct. Приклад використання директиви:

RADIX = hex; задає 16–у систему числення за замовчуванням

8. Директиви для визначення констант:

– директива EQU – використовується для визначення констант. Пам’ять під константи не виділяється. Приклад використання директиви:

X1 EQU 10h; оголошення константи X1, яка має значення 10h

При компіляції вихідного тексту, скрізь де буде зустрічатися символ X1, він буде мінятися на значення 10h.

– директива SET – призначення директиви таке ж, як і директиви EQU, тільки значення, що задано директивою SET, можна перепризначувати у вихідному тексті програми. Приклад використання директиви:

Y1 SET 10H; задаємо константу Y1=10H
Y1 SET 20h; перепризначуємо константу Y1=20H

– директива препроцесора #define – використовується також як і директива EQU. Приклад використання директиви:

#define K1 22h; оголошення константи K1, яка має значення 22h

9. Директива DS – резервує необхідну кількість байт або біт у пам’яті даних (ОЗП) для зберігання неініціалізованих змінних. Приклад використання директиви:

X1: DS 1; резервуємо один байт (одну чарунку ОЗП) для зберігання змінної X1
Y1: ds 2; резервуємо дві чарунки ОЗП для зберігання змінної Y1

10. Директиви для розміщення даних у пам'яті програм (ПЗП) або енергонезалежної пам'яті даних типу EEPROM мікроконтролера:

– директива DB – розміщує в пам'яті програм (ПЗП) або EEPROM дані розміром у байт, при цьому для МК PIC18 два байти можна упакувати в одну чарунку пам'яті ПЗП;

– директива DW – розміщує в ПЗП дані розміром у слово.

Приклади використання директив:

```
// розділ розміщення констант в пам'яті програм мікроконтролера (ПЗП)
PSECT myconst, abs, class=CONST, delta=1
ORG 800H
    DB 10h, 20h
    DW 1000h, 2FCAh

// розділ розміщення даних в енергонезалежній пам'яті даних (EEPROM)
PSECT mySetting, class=EEDATA, delta=1
  DB 1, 2, 3, 4, 5, 6
```

11. Директива GLOBAL – об'являє мітки й ідентифікатори глобальними, і тоді їх можна використовувати в інших модулях, а також використовувати їх в симуляторі.

12. Директива EXTRN – задає глобальні ідентифікатори, які були вже оголошені в інших модулях.

Структура тексту програми

Текст програми повинен бути розбитий на розділи для того, щоб його було легко читати, супроводжувати, документувати й модифікувати. Стосовно до PIC–мікроконтролерів структура тексту програми може виглядати так:

блок визначень

– розділ заголовків (призначення програми, ім'я автора, дата, версія);

– розділ файлів, що підключаються (за допомогою директиви #include);

– розділ конфігурації (визначення бітів конфігурації та ідентифікації за допомогою директиви CONFIG);

– розділ визначення констант;

– розділи оголошення змінних у пам'яті даних (за допомогою директиви DS);

– розділи розміщення даних у пам'яті програм (ПЗП) або енергонезалежної пам'яті даних типу EEPROM (за допомогою директив DB і DW);

– розділ визначення макросів;

блок коду

– вектор скидання (безумовний перехід на код ініціалізації);

– оброблювачі переривань;

– ініціалізація програми;

– основний цикл програми;

– набір підпрограм;

END

2. Завдання на самостійну роботу

Використовуючи директиви асемблера PIC Assembler і типову структуру тексту програми підготувати шаблон файлу на асемблері, який виконує наступні дії:

– задати тип мікроконтролера: МК PIC18F4520;

– підключити файл із описами для PIC–мікроконтролера;

– задати біти конфігурації: вимкнути сторожовий таймер WDT та вибрати тип генератора мікроконтролера – середньочастотний XT–генератор;

– задати константу Z = 34 у десятковій системі числення з можливістю зміни в програмі її значення;

– розмістити в ПЗП масив даних (0, 1, 2, 3, 4, 5) починаючи з адреси 100H;

– розмістити в енергонезалежній пам'яті даних масив даних у десятковій формі числення (10, 11, 12, 13, 14, 15);

– зарезервувати під змінні X1 і Y1, по одній чарунці пам'яті в ОЗП (в банку швидкого доступу);

– установити початкову адресу програми 0h;

– помістити команду зациклення програми в нескінченному циклі;

– установити кінець вихідного тексту програми на асемблері.

3. Зміст звіту

1. Основні модулі асемблера для МК PIC18 і їх призначення.
2. Типи файлів, які підтримує асемблер PIC Assembler.
3. Структура команди на мові асемблера PIC Assembler.
4. Формати представлення чисел в асемблері PIC Assembler.
5. Основні директиви асемблера PIC Assembler.
6. Структура тексту програми для PIC–мікроконтролера.
7. Розроблений за завданням на самостійну роботу шаблон файлу на асемблері.

4. Контрольні запитання

1. Призначення компілятора асемблера (pic–as).
2. Призначення компонувальник (hlink).
3. Призначення архіватора і менеджера бібліотек (xc8–ar).
4. Призначення файлу проекту з розширенням «.lst».
5. Призначення директив асемблера PIC Assembler.
6. Які символи можна використовувати для створення ідентифікаторів в PIC Assembler.
7. Формати представлення чисел в PIC–асемблері.
8. Призначення директиви PSECT.
9. Які директиви в PIC–асемблера можна використовувати для визначення констант.
10. Призначення директиви GLOBAL.

Практичне заняття № 3

СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРА PIC18. ЛІТЕРАЛЬНІ ТА БІТ – ОРІЄНТОВАНІ КОМАНДИ

Мета заняття

Вивчити структуру системи команд мікроконтролера PIC18 і їх особливості, а також вивчити команди роботи з константами та з окремими бітами.

1. Короткі теоретичні відомості

Система команд мікроконтролера (МК) PIC18 включає в себе 77 команд, які розбиті на п’ять груп [3]:

– літеральні команди (команди роботи з константами);
– біт – орієнтовані команди (робота з окремими бітами);
– байт – орієнтовані команди;
– команди керування (розгалуження, цикли та підпрограми);
– табличні команди.

Більшість команд (73 команди) займають у пам’яті програм (ПЗП МК) одне 16–розрядне слово (одну чарунку пам’яті), і чотири команди займають два слова в пам’яті програм, тому що мають велику кількість операндів. Усі однослівні команди виконуються за 1 машинний цикл (МЦ), крім команд, що змінюють лічильник команд, такі команди виконуються за 2 МЦ. Усі двослівні команди виконуються за 2 МЦ.

Усі команди, що виконують арифметичні й логічні операції, змінюють біти регістру стану STATUS, який містить прапори фіксації ознак результату. Структура регістру стану:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	N	OV	Z	DC	C

– C – прапор переносу (для без знакових чисел);
– DC – прапор десяткового переносу;
– Z – прапор нуля;
– OV – прапор переповнення (для знакових чисел);
– N – прапор знаку: 0 – позитивний результат, 1 – від’ємний результат.

2. Літеральні команди

Структура літеральних команд наведена на рис. 3.1, де OPCODE – це код операції даної команди, k – це константа, що входить до складу команди. Літеральні команди використовують безпосередню адресацію операнда. [3]

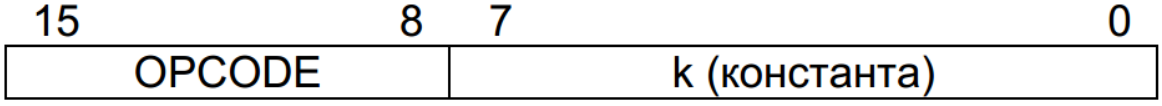

Рис. 3.1. Структура літеральних команд МК PIC18

У групу команд роботи з константами входить 10 команд [3]:

– команди завантаження констант:

1. MOVLW k – завантаження константи в регістр WREG: k → W, де k = 0...255

2. MOVLB k – завантаження константи в регістр вибору банку BSR: k → BSR, де k = 0...255, але в регістр BSR запишуться тільки 4 молодших біта

3. LFSR f, k – завантаження константи в регістр покажчик непрямої адресації FSR: k → FSRn, де f = 0...2 (номер регістру), k = 0...4095 (адреса чарунки ОЗП)

– арифметичні операції з константами:

4. ADDLW k – додавання: WREG + k → WREG, де k = 0...255

5. SUBLW k – вирахування: k – W → W, де k = 0...255

6. MULLW k – множення: WREG × k → PRODH:PRODL, де k = 0...255

– логічні операції з константами:

7. ANDLW k – логічне множення: W AND k → W, де k = 0...255

8. IORLW k – логічне додавання: W OR k → W, де k = 0...255

9. XORLW k – додавання по модулю два: W XOR k → W, де k = 0...255

– повернення з підпрограми із завантаженням константи в регістр WREG:

10. RETLW k – завантаження k → W, TOS → PC, де k = 0...255

3. Біт–орієнтовані команди

Структура біт–орієнтованих команд наведена на рис. 3.2, де b – номер біту в 8–розрядному регістрі (0...7), a=0 – використовується 8–розрядна адреса в банку швидкого доступу, a=1 – використовується 12–розрядна адреса з урахуванням значення регістру BSR, f – 8–розрядна адреса регістру в ОЗП. Біт–орієнтовані команди використовують пряму адресацію операндів. У полі <a> замість <0> краще вказувати константу <a> або < c>, а замість <1> необхідно вказувати константу <b> (вони визначені у файлі – <xc.inc>). Якщо операнд <a> у команді не вказати тоді за замовчуванням він буде рівним нулю. [3]

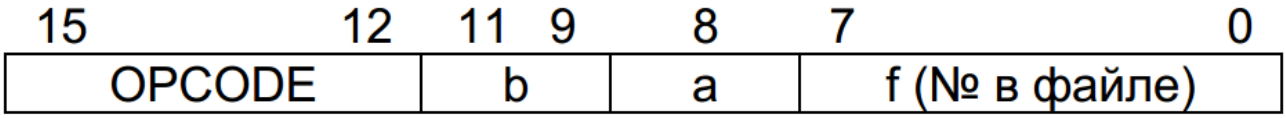

Рис. 3.2. Структура біт–орієнтованих команд МК PIC18

У групу біт–орієнтованих команд входить п’ять команд: [3]

– команди керування станом біта:

1. BCF f, b, a – скидання біту «b» у регістрі «f» в стан лог.0

2. BSF f, b, a – установка біту «b» у регістрі «f» в стан лог.1

3. BTG f, b, a – інвертування біту «b» у регістрі «f»

– команди контролю стану біта:

1. BTFSC f, b, a – перевірка біту «b» у регістрі «f», і пропуск наступної команди, якщо біт дорівнює нулю

2. BTFSS f, b, a – перевірка біту «b» у регістрі «f», і пропуск наступної команди, якщо біт дорівнює одиниці

4. Завдання на самостійну роботу

1. Використовуючи вивчені команди МК PIC18 і шаблон файлу на асемблері із ПЗ2, скласти програму для МК PIC18F4520, яка виконує наступні дії:

– визначити константи X1 = 22, X2 = 18, X3 = 90; X4 = 5 (числа задані в десятковій формі);

– завантажити в регістр FSR0 адресу чарунки ОЗП 100H;

– установити регістр вибору банку BSR на 2 банк ОЗП;

– завантажити в регістр WREG константу X1;

– скласти регістр WREG з константою X2;

– відняти з константи X3 вміст регістру WREG (результат записати у звіт);

– помножити регістр WREG на константу X4 (результат записати у звіт);

– обнулити за допомогою логічної маски 1 і 4 біти регістру WREG (результат записати у звіт);

– установити за допомогою логічної маски біти 0 і 7 регістру WREG (результат записати у звіт);

– інвертувати за допомогою логічної маски біти 2 і 5 регістру WREG (результат записати у звіт);

– установити в стан лог.1 третій біт чарунки пам’яті ОЗП з адресою 200H (результат записати у звіт);

– інвертувати четвертий біт чарунки ОЗП з адресою 0H у банку швидкого доступу (результат записати у звіт);

– зациклити програму.

5. Зміст звіту

1. Структура регістру стану STATUS і призначення прапорів.
2. Список літеральних команд МК PIC18 та їх призначення.
3. Список біт–орієнтованих команд МК PIC18 та їх призначення.
4. Програма на асемблері для МК PIC18F4520, розроблена за індивідуальним завданням.
5. Результати роботи програми (у десятковій, двійковій і шістнадцятковій системах числення).

6. Контрольні запитання

1. Призначення прапору переносу.
2. Призначення прапору переповнення.
3. Призначення поля «a» в біт–орієнтованих командах МК PIC18.
4. Призначення поля «b» в біт–орієнтованих командах МК PIC18.
5. Структура системи команд мікроконтролера PIC18.

Практичне заняття № 4

СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРА PIC18. БАЙТ–ОРІЄНТОВАНІ КОМАНДИ

Мета заняття

Вивчити групу байт–орієнтованих команд мікроконтролера PIC18, а також розібрати приклади їх використання.

1. Короткі теоретичні відомості

Структура байт–орієнтованих команд наведена на рис. 4.1, де: [3]
– OPCODE – це код операції даної команди;
– d=0 – результат операції розміщується в регістрі WREG;
– d=1 – результат операції розміщується в регістрі «f»;
– a=0 – 8–розрядна адреса в банку швидкого доступу;
– a=1 – 12–розрядна адреса з урахуванням значення регістру BSR;
– f – 8–розрядна адреса регістру в ОЗП.

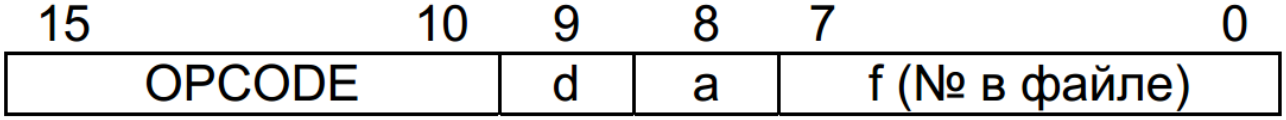

Рис. 4.1. Структура байт–орієнтованих команд МК PIC18

Байт–орієнтовані команди використовують пряму адресацію операндів. Поле <a> в команді вказує спосіб звертання до чарунки пам'яті ОЗП. У полі <a> замість <0> краще вказувати константу <a> або < c>, а замість <1> необхідно вказувати константу <b> (вони визначені у файлі – <xc.inc>). Якщо операнд <a> у команді не вказати тоді за замовчуванням він буде рівним нулю.

Система команд МК PIC18 є симетричної, що дозволяє зберігати результат операції на місці, як першого, так і другого операнда. Для того щоб вказати місце збереження результату операції використовується поле <d>. У полі <d> замість <0> краще вказувати константу <w> , а замість <1> необхідно вказувати константу <f> (вони визначені у файлі – <xc.inc>). Якщо операнд <d> у команді не вказати тоді за замовчуванням він буде дорівнювати одиниці.

У групу команд роботи з байтами входить 31 команда [3]:

– команди переміщення даних:

1. MOVWF f, a – переміщення даних з регістру WREG з у регістр «f»: WREG $\rightarrow$ f

2. MOVF f, d, a – переміщення даних з регістру «f» у регістр WREG (якщо d=0) або в регістр «f» (якщо d=1)

3. MOVFF fs, fd – переміщення даних з регістру «fs» у регістр «fd». Команда має довжину два слова, тобто займає дві чарунки пам'яті ПЗП

4. SWAPF f, d, a – поміняти місцями старший і молодший напівбайти регістру «f», при цьому результат запишеться в регістр WREG (якщо d=0) або залишається в регістрі «f» (якщо d=1)

– арифметичні операції з даними:

5. ADDWF f, d, a – додавання: WREG + f → W (d=0) або «f» (якщо d=1)

6. ADDWFC f, d, a – додавання з урахуванням прапора переносу C: WREG + f+ C → W (d=0) або «f» (якщо d=1)

7. SUBWF f, d, a – віднімання: f – W → W (d=0) або «f» (якщо d=1)

8. SUBWFB f, d, a – віднімання з урахуванням прапора переносу C: f – W – C → W (d=0) або «f» (якщо d=1)

9. SUBFWB f, d, a – віднімання з урахуванням прапора переносу C: W – f – C → W (d=0) або «f» (якщо d=1)

10. MULWF f, a – множення: WREG × f → PRODH:PRODL

11. INCF f, d, a – інкремент регістру f: f + 1 → W (d=0) або «f» (якщо d=1)

12. INCFSZ f, d, a – інкремент регістру f та пропуск наступної команди, якщо f=0 (замість наступної команди виконується команда NOP): f + 1 → W (d=0) або «f» (якщо d=1)

13. INFSNZ f, d, a – інкремент регістру f та пропуск наступної команди, якщо f≠0: f + 1 → W (d=0) або «f» (якщо d=1)

14. DECF f, d, a – декремент регістру f: f – 1 → W (d=0) або «f» (якщо d=1)

15. DECFSZ f, d, a – декремент регистру f та пропуск наступної команди, якщо f=0: f – 1 → W (d=0) або «f» (якщо d=1)

16. DCFSNZ f, d, a – декремент регистра f та пропуск наступної команди, якщо f≠0: f – 1 → W (d=0) або «f» (якщо d=1)

17. NEGF f, a – зміна знака числа (негативні числа представляються в додатковому коді): f → –f

– логічні операції з даними:

18. ANDWF f, d, a – логічне множення: W AND f → W (d=0) або «f» (якщо d=1)

19. IORWF f, d, a – логічне додавання: W OR f → W (d=0) або «f» (якщо d=1)

20. XORWF f, d, a – додавання по модулю два: W XOR f → W (d=0) або «f» (якщо d=1)

21. CLRF f, a – очищення регістру f: 00h → f

22. SETF f, a – установка всіх біт регістру f: FFh (255) → f

23. COMF f, d, a – інвертування всіх біт регістру f, при цьому результат запишеться в регістр WREG (якщо d=0) або залишається в регістрі «f» (якщо d=1)

24. RLCF f, d, a – циклічний зсув вмісту регістру f вліво на один біт через прапор переносу, при цьому результат запишеться в регістр WREG (якщо d=0) або залишається в регістрі «f» (якщо d=1)

25. RLNCF f, d, a – циклічний зсув вмісту регістру f вліво на один біт, при цьому результат запишеться в регістр WREG (якщо d=0) або залишається в регістрі «f» (якщо d=1)

26. RRCF f, d, a – циклічний зсув вмісту регістру f вправо на один біт через прапор переносу, при цьому результат запишеться в регістр WREG (якщо d=0) або залишається в регістрі «f» (якщо d=1)

27. RRNCF f, d, a – циклічний зсув вмісту регістру f вправо на один біт, при цьому результат запишеться в регістр WREG (якщо d=0) або залишається в регістрі «f» (якщо d=1)

Команди зсуву можна використовувати для перестановки біт у байті, для швидкого множення або ділення на числа кратні два, а також для перетворення паралельного коду в послідовний.

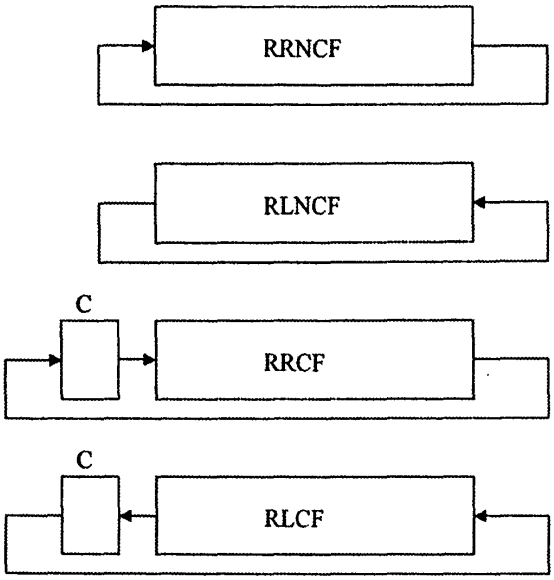

Рис. 4.2. Команди циклічного зсуву МК PIC18

– операції порівняння:

28. CPFSEQ f, a – порівняння значень регістрів f і WREG, і пропуск наступної команди, якщо f = WREG (замість наступної команди виконується команда NOP)

29. CPFSGT f, a – порівняння значень регістрів f і WREG, і пропуск наступної команди, якщо f > WREG

30.CPFSLT f, a – порівняння значень регістрів f і WREG, і пропуск наступної команди, якщо f < WREG

31. TSTFSZ f, a – перевірка (тест) регістру f, і пропуск наступної команди, якщо f = 0

Команд порівняння використовуються разом з командами керування для реалізації умовних розгалужень.

2. Приклади використання байт–орієнтованих команд

Приклад 1. Скласти дві 8–розрядні змінні X1+X2 і результат записати в змінну X3.

```
; розміщення змінних в банку швидкого доступу (Access bank)
GLOBAL X1, X2, X3
PSECT udata_acs
X1: DS 1; розміщення змінної X1 у чарунці ОЗП
X2: DS 1; розміщення змінної X2 у чарунці ОЗП
X3: DS 1; розміщення змінної X3 у чарунці ОЗП
```

```
; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
    MOVF X1, w, a; переміщення X1 до W
    ADDWF X2, w, a; додавання W + X2 → W
    MOVWF X3, a; переміщення W до X3
```

Приклад 2. Скласти два 16–розрядних числа X1+X2 і результат записати замість числа X2.

```
; розміщення змінних в банку швидкого доступу (Access bank)
GLOBAL X1L, X1H, X2L, X2H
PSECT udata_acs
X1L: DS 1; розміщення молодшого байту змінної X1 у чарунці ОЗП
X1H: DS 1; розміщення старшого байту змінної X1 у чарунці ОЗП
X2L: DS 1; розміщення молодшого байту змінної X2 у чарунці ОЗП
X2H: DS 1; розміщення старшого байту змінної X2 у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
    MOVF X1L, w, c; переміщення X1L до W
    ADDWF X2L, f, c; додавання молодших байт W + X2L → X2L
    MOVF X1H, w, c; переміщення X1H до W
    ADDWFC X2H, f, c; додавання старших байт з урахуванням біту переносу W + X2H → X2H
```

Приклад 3. Порівняти два 8–розрядні числа X1 і X2, і якщо вони дорівнюють (X1=X2), то необхідно очистити змінну X3, а якщо не дорівнюють, то встановити всі біти в змінній X3.

```
; розміщення змінних в банку швидкого доступу (Access bank)
GLOBAL X1, X2, X3
PSECT udata_acs
X1: DS 1; розміщення змінної X1 у чарунці ОЗП
X2: DS 1; розміщення змінної X2 у чарунці ОЗП
X3: DS 1; розміщення змінної X3 у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
    MOVF X1, w, a; переміщення X1 до W
    CPFSEQ X2, a; порівняння W(X1) і X2
```

```
    GOTO M1; перехід на мітку M1, якщо W(X1) не дорівнює X2
    CLRF X3, a; очищення змінної X3
    GOTO M2; обхід наступної команди
M1: SETF X3, a; установка всіх біт у змінній X3
M2: ....; продовження програми
```

Приклад 4. Перетворення паралельного коду в послідовний код. Передати вміст змінної X1 через вивід МК RB0 у послідовному коді починаючи з молодшого біта.

```
; розміщення змінних в банку швидкого доступу (Access bank)
GLOBAL X1, CL
PSECT udata_acs
X1: DS 1; розміщення змінної X1 у чарунці ОЗП
CL: DS 1; лічильник циклу

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
    BCF TRISB, 0, a; налаштувати RB0 на вивід даних
    MOVLW 8; початкове значення лічильника циклів
    MOVWF CL, a
M3: RRCF X1, f, a; зсув X1 вправо через прапор переносу CF
    BTFSS STATUS, 0, a; перевірка прапору переносу CF=1
    GOTO M1; перехід на мітку M1, якщо прапор переносу CF=0
    BSF PORTB, 0, a; установка RB0=1
    GOTO M2
M1: BCF PORTB, 0, a; скидання RB0=0
M2: DECFSZ CL, F, a; CL–1 та вихід з циклу, якщо CL=0
    GOTO M3; повернення на початок циклу
```

3. Завдання на самостійну роботу

1. Використовуючи вивчені команди МК PIC18 і шаблон файлу на асемблері із ПЗ2, скласти програму для МК PIC18F4520, яка виконує наступні дії:

– задати глобальні змінні розміром у байт у банку швидкого доступу: X1, X2, X3;

– задати глобальні змінні розміром у слово в банку швидкого доступу: Y1, Y2;

– задати початкові значення змінних: Y1=2222h, Y2=5555h;

– задати початкові значення змінних: X1=190, X2=20, X3=0 (десяткова форма);

– обчислити X3 = X1 – X2 (результат записати у звіт);

– змінити знак змінної X3 (результат записати у звіт);

– обчислити Y2–Y1 і результат записати замість числа Y2 (результат записати у звіт);

– порівняти X1 і X3, і якщо X1>X3, то очистити змінну X2, а якщо ні, то встановити всі біти в змінній X2 (результат записати у звіт);

– передати вміст змінної X3 через вивід МК RC1 у послідовному коді починаючи зі старшого біта;

– зациклити програму.

4. Зміст звіту

1. Список байт–орієнтованих команд МК PIC18 та їх призначення.

2. Програма на асемблері для МК PIC18F4520, розроблена за індивідуальним завданням.

3. Результати роботи програми (у десятковій, двійковій і шістнадцятковій системах числення).

4. Часова діаграма сигналу на виході виводу RC1 побудована за допомогою логічного аналізатора.

Для спостереження за сигналом на виводі МК RC1 необхідно викликати меню: Window / Simulator / Logic Analyzer. У вікні логічного аналізатора (Logic Analyzer) нажати кнопку «Settings» і перейти на закладку «Add/ Delete Pins». У лівому вікні «Available pins» вибрати вивід RC1, і перенести його в праве вікно «Selected Pins» нажавши кнопку «→», а потім кнопку «OK».

5. Контрольні запитання

1. Призначення поля «d» в байт–орієнтованих командах МК PIC18.

2. В якому регістрі зберігається результат множення командою MULWF.

3. Які дії з окремими бітами можна зробити за допомогою команди логічного множення ANDWF?

4. Які дії з окремими бітами можна зробити за допомогою команди логічного множення IORWF?

5. Які дії з окремими бітами можна зробити за допомогою команди логічного множення XORWF?

6. Призначення команд циклічного зсуву.

7. Призначення команди NEGF.

8. Яку арифметичну операцію виконує команда INCF.

9. Яку арифметичну операцію виконує команда DECF.

10. Призначення команд додавання та віднімання з урахуванням прапора переносу CF.

11. Яку умову перевіряє команда мікроконтролера PIC18 CPFSEQ.

12. Які команди мікроконтролера PIC18 можна використовувати для перетворення паралельного коду в послідовний код.

Практичне заняття № 5

СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРА PIC18. КОМАНДИ УПРАВЛІННЯ

Мета заняття

Вивчити групу команд управління мікроконтролера PIC18, а також приклади реалізації умовних розгалужень і циклів.

1. Короткі теоретичні відомості

Група команд управління мікроконтролера PIC18 включає в себе 22 команди, які можуть використовуватися для організації безумовних і умовних розгалужень, організації підпрограм, керування стеком, а також для керування режимами роботи мікроконтролера. [3]

Команди безумовного переходу:

1. GOTO n – безумовний перехід у межах усього адресного простору МК PIC18 обсягом 2 Мбайта, де n = 0...1048575: n (PC<20:1>, PC<0> = 0. Команда займає два слова в ПЗП й виконується за два машинні цикли (МЦ);

2. BRA n – безумовний відносний перехід у межах (PC+2)+2*n → PC, де n – це зсув у діапазоні –1024...+1023, відносно поточного стану лічильника команд PC. Для переходу вперед використовуються позитивні числа, а для переходу назад – від'ємні числа. Команда займає одне слово в ПЗП, а виконується за два МЦ.

Команди умовного переходу:

3. BC n – перехід, якщо прапор переносу C=1;
4. BNC n – перехід, якщо прапор переносу C=0;
5. BOV n – перехід, якщо прапор переповнення OV=1;
6. BNOV n – перехід, якщо прапор переповнення OV=0;
7. BN n – перехід, якщо прапор знака N=1;
8. BNN n – перехід, якщо прапор знака N=0;
9. BZ n – перехід, якщо прапор нуля Z=1;
10. BNZ n – перехід, якщо прапор нуля Z=0;

Усі команди умовного переходу виконують відносний перехід у межах (PC+2) + 2*n → PC, де n – це зсув у діапазоні –128...+127, відносно поточного стану лічильника команд PC. Для переходу вперед використовуються позитивні числа, а для переходу назад – від'ємні числа. Команда займає одне слово в ПЗП. Якщо умова дійсна, то команда переходу виконується за два МЦ, а якщо умова неправильна, то виконується наступна команда.

Команди організації підпрограм і роботи зі стеком:

11. CALL n, s – виклик підпрограми в межах усього адресного простору МК PIC18 обсягом 2 Мбайта, де n = 0...1048575. Команда виконує наступні дії:

– збереження адреси повернення з підпрограми в стеку (PC+4 → TOS);

– якщо параметр s = 1, то вміст регістрів WREG, STATUS і BSR зберігаються в тіньових регістрах;

– якщо параметр s = 0 (за замовчуванням), то вміст регістрів WREG, STATUS і BSR не зберігаються в тіньових регістрах;

– завантаження константи n у лічильник команд PC (n → PC<20:1>, PC<0> = 0.).

Команда займає два слова в ПЗП й виконується за два МЦ.

12. RCALL n – виклик підпрограми в межах (PC+2) + 2*n → PC, де n – це зсув у діапазоні –1024...+1023, відносно поточного стану лічильника команд PC. Команда займає одне слово в ПЗП, а виконується за два МЦ.

13. RETURN s – повернення з підпрограми. Команда виконує наступні дії:

– завантаження адреси зі стека в лічильник команд PC (TOS → PC);

– якщо параметр s = 1, то значення тіньових регістрів завантажується в регістри WREG, STATUS і BSR;

– якщо параметр s = 0 (за замовчуванням), то значення регістрів WREG, STATUS і BSR не відновлюється.

14. RETFIE s – повернення з підпрограми обробки переривання. Команда виконує наступні дії:

– завантаження адреси зі стека в лічильник команд PC (TOS → PC);

– дозвіл переривань – установка глобального біта переривань (GIE = 1);

– якщо параметр s = 1, то значення тіньових регістрів завантажується в регістри WREG, STATUS і BSR;

– якщо параметр s = 0 (за замовчуванням), той значення регістрів WREG, STATUS і BSR не відновляється.

15. PUSH – запис на вершину стека повернення (PC+2 → TOS), при цьому попереднє значення вершини стека, переміщається на один рівень нижче.

16. POP – читання з вершини стека повернення при якому вміст стека переміщається на один рівень нагору. Попереднє значення вершини стека зникає.

Команди PUSH та POP використовуються для програмного керування стеком.

Команди керування мікроконтролером:

17. CLRWDT – скидання сторожового таймера (переддільник WDT скидається в початковий стан, а таймер WDT скидається в нуль).

18. RESET – програмне скидання МК, аналогічний апаратному скиданню на вході MCLR.

19. SLEEP – перевід МК у сплячий режим (режим зниженого енергоспоживання). Команда виконує наступні дії:

– зупинка тактового генератора МК;

– скидання сторожового таймера WDT;

– скидання біта PD=0, установка біта TO = 1.

20. DAW – десяткова корекція вмісту регістру WREG у двійково–десятковий формат (BCD).

21. NOP – холоста операція (нічого не робить але виконується за 1 МЦ). Команда має два різні коди операції, що виключає зависання МК у випадку влучення адреси в нереалізовані фізично чарунки ПЗП.

2. Реалізація розгалужень на асемблері PIC18

Перший спосіб. Для реалізації розгалужень на мові асемблер для МК PIC18 можна використовувати команди порівняння разом з командами переходів.

Приклад 1. У змінних T1 і T2 зберігаються значення двох вимірювань температури об'єкта. Необхідно визначити яке з них має більше значення, і помістити його в змінну Tmax.

```
; розміщення змінних в банку швидкого доступу (Access bank)
GLOBAL T1, T2, Tmax
PSECT udata_acs
T1: DS 1; розміщення змінної T1 у чарунці ОЗП
T2: DS 1; розміщення змінної T2 у чарунці ОЗП
Tmax: DS 1; розміщення змінної Tmax у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
    MOVFF T1, Tmax; переміщуємо змінну T1 до Tmax
    MOVF T2, w, a; переміщуємо змінну T2 до регістра WREG
; порівняння Tmax (T1) та W (T2) та пропуск наступної команди, якщо T1 > T2
    CPFSGT Tmax, a
    MOVWF Tmax, a; переміщуємо змінну T2 (WREG) до Tmax
    – – – – – – – – – – – – ; продовження програми
```

Другий спосіб. Для реалізації розгалужень також можна використовувати команди умовних переходів, які аналізують стан прапорів умови в регістрі STATUS.

Приклад 2. Виконати віднімання двох змінних W = X1 – X2, і якщо результат від'ємний, то інкрементувати значення змінної X3, а якщо ні, то декрементувати значення змінної X3.

```
; розміщення змінних в банку швидкого доступу (Access bank)
GLOBAL X1, X2, X3
PSECT udata_acs
X1: DS 1; розміщення змінної X1 у чарунці ОЗП
X2: DS 1; розміщення змінної X2 у чарунці ОЗП
X3: DS 1; розміщення змінної X3 у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
```

```
    MOVF X2, w, a; переміщуємо змінну X2 до регістру WREG
    SUBWF X1, w, a; віднімання X1 – WREG → WREG
    BN M1; перехід на мітку M1, якщо результат від'ємний
    DECF X3, f, a; декрементування змінної X3–1 → X3
    BRA M2; обхід наступної команди
M1: INCF X3, f, a; інкрементування змінної X3+1 → X3
M2: – – – – – – – – – – – –; продовження програми
```

3. Реалізація циклів на асемблері PIC18

Для реалізації циклів на асемблері МК PIC18 можна використовувати два способи.

Перший спосіб.

```
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
Count: DS 1; розміщення змінної Count у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:

AGAIN:
    тіло циклу
    DECFSZ Count, f, a; декремент Count = Count – 1 і пропуск наступної команди, якщо Count = 0
    BRA AGAIN; повернення на початок циклу
    – – – – – – – – – – – – ; продовження програми
```

Другий спосіб.

```
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
Count: DS 1; розміщення змінної Count у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:

AGAIN:
    тіло циклу
    DECF Count, F; декремент Count = Count – 1
    BNZ AGAIN; повернення в початок циклу, якщо Count ≠ 0
    – – – – – – – – – – – – ; продовження програми
```

Приклад 3. Додати до змінної X1 константу K 10 раз.

```
K EQU 3; константа К=3
PSECT udata_acs
X1: DS 1; розміщення змінної X1 у чарунці ОЗП
Count: DS 1; розміщення лічильника циклів у чарунці ОЗП

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
    MOVLW 0x0A
    MOVWF COUNT, a; завантаження в лічильник кількість циклів
    CLRF X1, a; обнуління змінної X1
    MOVLW K; завантаження константи в WREG
AGAIN:
    ADDWF X1, f, a; додавання X1 + К = X1
    DECF COUNT, f, a; декремент лічильника циклу COUNT – 1 → COUNT
    BNZ AGAIN; повернення на початок циклу, якщо COUNT ≠ 0
    – – – – – – – – – – – – ; продовження програми
```

4. Завдання на самостійну роботу

1. Використовуючи вивчені команди МК PIC18 і шаблон файлу на асемблері із ПЗ2, скласти програму для МК PIC18F4520, яка виконує наступні дії:

– задати глобальні змінні розміром у байт у банку швидкого доступу: X1, X2, X3;

– задати глобальні змінні для зберігання трьох значень температури у банку швидкого доступу: T1, T2, Tmin;

– задати початкові значення змінних X1 та X2;

– задати початкові значення температури T1 та T2;

– обнулити значення змінних X3 та Tmin;

– визначити яка зі змінних T1 і T2 має менше значення та помістити її значення в змінну Tmin (результат записати у звіт);

– виконати додавання: W = X1 + X2 (результат записати у звіт);

– якщо при додаванні відбудеться переповнення регістру WREG, то необхідно додати до змінної X3 константу *K*1=4 п’ять разів (результат записати у звіт). Для організації циклу використати перший спосіб реалізації циклів на асемблері PIC18;

– якщо при додаванні не буде переповнення регістру WREG, то необхідно відняти зі змінної X3 константу *K*2=5 вісім разів (результат записати у звіт). Для організації циклу використати другий спосіб реалізації циклів на асемблері PIC18;

– зациклити програму.

5. Зміст звіту

1. Список групи команд управління МК PIC18 та їх призначення.
2. Програма на асемблері для МК PIC18F4520, розроблена за індивідуальним завданням.
3. Результати роботи програми (у десятковій, двійковій і шістнадцятковій системах числення).

6. Контрольні запитання

1. Чим відрізняються команди безумовного переходу GOTO та BRA?
2. Призначення команд умовного переходу.
3. Чим відрізняються команди організації підпрограм CALL та RCALL?
4. Призначення поля «s» в команді повернення з підпрограми RETURN.
5. Призначення команди RETFIE.
6. Призначення команди CLRWDT.
7. Призначення команди SLEEP.
8. Які існують способи реалізації розгалужень на мові асемблер для МК PIC18.
9. Які існують способи реалізації циклів на асемблері МК PIC18.
10. Принцип роботи стека в мікроконтролері PIC18.

Практичне заняття № 6

СИСТЕМА КОМАНД МІКРОКОНТРОЛЕРА PIC18. ТАБЛИЧНІ КОМАНДИ

Мета заняття

Вивчити групу команд мікроконтролера PIC18 для роботи з таблицями, а також приклади використання непрямої адресації.

1. Короткі теоретичні відомості

Група табличних команд мікроконтролера PIC18 призначена для виконання операцій читання та запису даних в пам'ять програм (ПЗП) мікроконтролера. Відповідно до Гарвардської архітектури – це окремий вид команд, який дозволяє здійснювати обмін даними між ПЗП та пам'яттю даних (ОЗП) мікроконтролера. Група табличних команд включає в себе вісім команд: [3]

Команди читання з ПЗП:

1. TBLRD* – табличне читання;
2. TBLRD*+ – табличне читання з пост–інкрементом;
3. TBLRD*– – табличне читання з пост–декрементом;
4. TBLRD+* – табличне читання з перед–інкрементом.

Команди запису в ПЗП:

1. TBLWT* – табличний запис;
2. TBLWT*+ – табличний запис з пост–інкрементом;
3. TBLWT*– – табличний запис з пост–декрементом;
4. TBLWT+* – табличний запис з пред–інкрементом.

Для роботи із ПЗП табличні команди використовують два регістри спеціальних функцій (РСФ):

– TABLAT – 8–розрядний регістр даних, використовується для передачі даних між чарунками ПЗП та ОЗП МК;

– TBLPTR (TBLPTRU, TBLPTRH, TBLPTRL) – 22–розрядний покажчик адреси, використовується для адресації чарунок пам'яті в області ПЗП, і складається із трьох 8–розрядних РСФ. Молодші 21 біт використовуються для адресації пам'яті програм (2 Мбайта), а старший 22–ой біт використовується для звертання до області ідентифікації та конфігурації МК.

Операція читання ПЗП МК приведена на рис. 6.1. Спочатку необхідно помістити в покажчик TBLPTR адресу чарунки ПЗП з якої необхідно прочитати данні, а потім після виконання команди табличного читання, прочитаний байт з'явиться в регістрі TABLAT. При цьому вміст регістру адреси TBLPTR може змінюватися залежно від типу використаної команди читання. Табличне читання оперує байтами, що дозволяє прочитати будь–яку чарунку ПЗП (парну та непарну). За одну операцію читання можна прочитати один байт даних.

Операція запису в ПЗП МК приведена на рис. 6.2. При запису спочатку байт даних записується в регістр даних TABLAT, а в регістрі адреси TBLPTR розміщується адреса чарунки ПЗП. Після виконання команди табличного запису, дані з

регістру TABLAT, записуються в задану чарунку ПЗП. Операція запису FLASH–пам'яті може виконуватися тільки блоками по вісім байт. Стирання пам'яті програм також може здійснюватися тільки блокам по 64 байта. При виконанні команд запису в ПЗП необхідно враховувати, що виконання поточної програми припиняється на час запису, тобто наступна команда почне виконуватися тільки по закінченню операції запису.

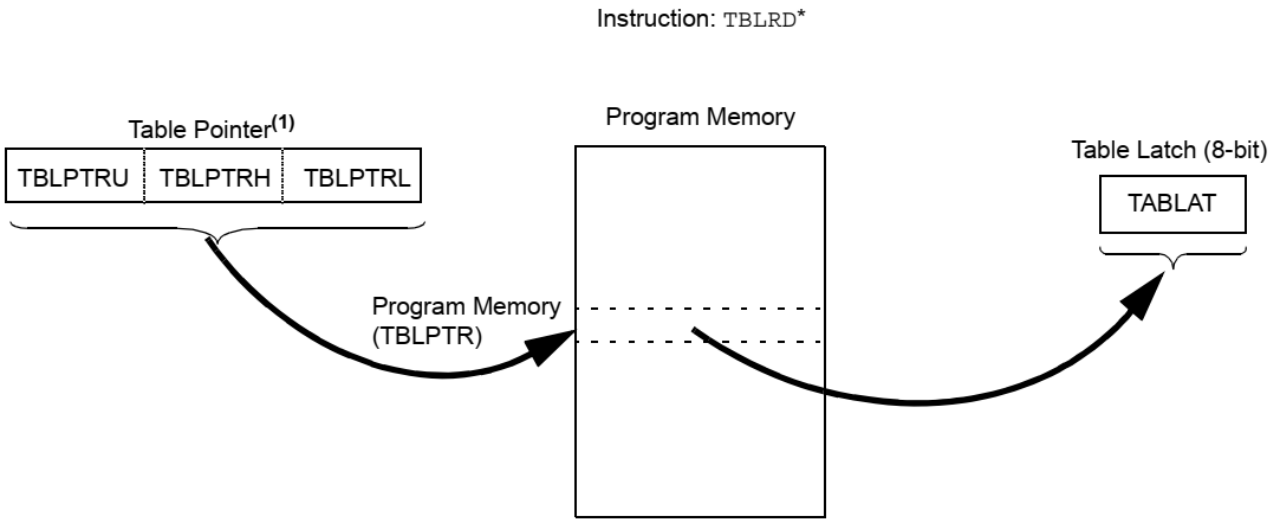

Рис. 6.1. Операція табличного читання з ПЗП МК

Операції табличного запису й читання можна заблокувати шляхом скидання відповідних біт конфігурації.

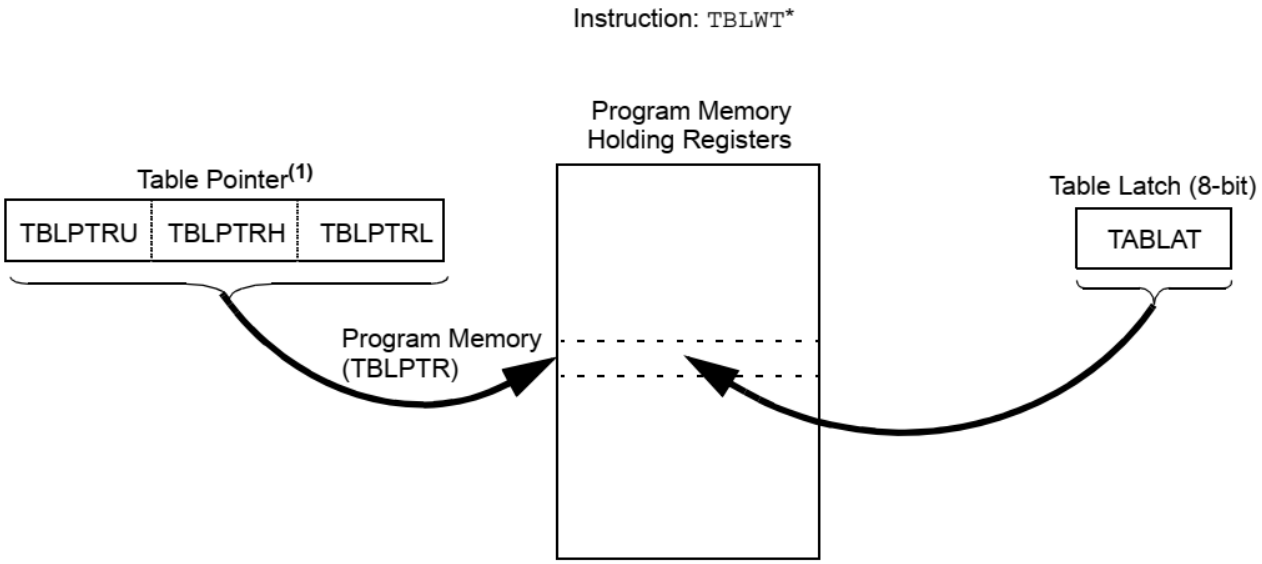

Рис. 6.2. Операція табличного запису в ПЗП МК

2. Непряма адресація на асемблері PIC18

Непряма адресація – це спосіб адресації, при якому в команді вказується ім'я регістра–покажчика, у якому в свою чергу зберігається адреса розміщення даних. В МК PIC18 реалізовано три 12–розрядних регістри непрямої адресації: FSR0 (FSR0H:FSR0L), FSR1 (FSR1H:FSR1L) та FSR2 (FSR2H:FSR2L). Кожному регіст-

ру непрямої адресації відповідає п’ять додаткових регістрів, які визначають дії з регістром FSR при виконанні непрямої адресації (де *n* – номер регістра непрямої адресації):

– INDFn – регістр FSR при непрямій адресації не змінюється;

– POSTINCn – інкремент регістру FSR після непрямої адресації;

– POSTDECn – декремент регістру FSR після непрямої адресації;

– PREINCn – інкремент регістру FSR перед непрямою адресацією;

– PLUSWn – для непрямої адресації використовується сума значень регістрів FSR і WREG, що дозволяє використовувати індексну непряму адресацію.

Приклад 1. В ПЗП, починаючи з адреси 300H, розташовані дані: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Перенести дані із ПЗП в чарунки ОЗП (масив X1 який розміщений в банку швидкого доступу).

```
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
CL: DS 1; лічильник циклу
X1: DS 10; резервуємо 10 чарунок в ОЗП для масиву даних X1

// розділ розміщення даних у пам’яті програм (ПЗП)
PSECT myconst, abs, class=CONST, delta=1
ORG 300h
    DB 1, 2, 3, 4, 5, 6, 7, 8, 9, 0x0A

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
; завантажуємо початкову адресу даних в ПЗП в TBLPTR=300H
    MOVLW 0H
    MOVWF TBLPTRL,a
    MOVLW 3H
    MOVWF TBLPTRH,a
    CLRF TBLPTRU,a
    LFSR 0, X1; завантаження в FSR0 початкової адреси даних в ОЗП
    MOVLW 10
    MOVWF CL, a; організація циклу на 10
M1:
    TBLRD*+ ; читання даних із ПЗП з пост–інкрементом адреси в TBLPTR
; завантаження даних в ОЗП з пост–інкрементом адреси в FSR0
    MOVFF TABLAT, POSTINC0
    DECFSZ CL, f, a; CL=CL–1, і пропуск наступної команди, якщо CL=0
    BRA M1; перехід на початок циклу
    —— продовження програми
```

Приклад 2. В ПЗП, починаючи з адреси 800H, розташовані дані: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Виконати підрахунок контрольної суми даної області ПЗП, а результат записати в змінну CHECKSUM.

```
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
CL: DS 1; лічильник циклу
CHECKSUM: DS 1; резервуємо 1 чарунку в ОЗП для змінної CHECKSUM

// розділ розміщення даних у пам’яті програм (ПЗП)
PSECT myconst, abs, class=CONST, delta=1
ORG 800h
    DB 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
; завантажуємо початкову адресу даних в ПЗП в TBLPTR=800H
    MOVLW 0H
    MOVWF TBLPTRL, a
    MOVLW 8H
    MOVWF TBLPTRH, a
    CLRF TBLPTRU
    CLRF CHECKSUM, a
    MOVLW 10
    MOVWF CL, a; організація циклу на 10
    MOVLW 0
M1:
    TBLRD*+ ; читання даних із ПЗП з пост–інкрементом адреси в TBLPTR
    ADDWF TABLAT, w, a; накопичення суми в регістрі WREG
    DECFSZ CL, f, a; CL=CL–1, і пропуск наступної команди, якщо CL=0
    BRA M1; перехід на початок циклу
    MOVWF CHECKSUM, a; збереження контрольної суми
    —— продовження програми
```

Приклад 3. В ОЗП в банку швидкого доступу розміщено два 4–х байтних числа Y1 і Y2. Виконати додавання багатобайтних чисел: Y2=Y2+Y1.

```
// об'явлення констант
#define CF 0 // прапор переносу CF в регістрі STATUS

; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
CL: DS 1; лічильник циклу
```

```
Y1: DS 4; резервуємо 4 чарунки в ОЗП для числа Y1
Y2: DS 4; резервуємо 4 чарунки в ОЗП для числа Y2

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
resetVec:
; завантаження початкових значень змінних Y1 і Y2
    MOVLW 11H
    MOVWF Y1
    MOVWF Y1+1
    MOVWF Y1+2
    MOVWF Y1+3
    MOVLW 33H
    MOVWF Y2
    MOVWF Y2+1
    MOVWF Y2+2
    MOVWF Y2+3
    LFSR 0, Y1; завантаження в FSR0 початкової адреси Y1
    LFSR 1, Y2; завантаження в FSR1 початкової адреси Y2
    MOVLW 4
    MOVWF CL; організація циклу на 4
    BCF STATUS, CF; скидання прапора переносу CF
M1:
    MOVF POSTINC0, w, a; завантаження і–го байта Y1 в WREG
    ADDWFC POSTINC1, f, a; додавання і–х байт: Y2[i] = Y2[i] + Y1[i]
    DECFSZ CL, f, a; CL=CL–1, і пропуск наступної команди, якщо CL=0
    BRA M1; перехід на початок циклу
    —— продовження програми
```

Приклад 4. В енергонезалежній пам’яті даних EEPROM розміщено текстовий рядок. Прочитати дані з EEPROM і перемістити їх в ОЗП (змінна NAME).

```
; розміщення змінних в банку швидкого доступу (Access bank)
PSECT udata_acs
CL: DS 1; лічильник циклу
NAME: DS 10; резервуємо 10 чарунок в ОЗП для масиву даних NAME

// розділ розміщення текстового рядка в EEPROM
PSECT mySetting, class=EEDATA, delta=1
    DB "PROFATYLOV"

; початкова адреса програми після скидання МК PIC18
PSECT resetVec, abs, class=CODE, reloc=2
ORG 0
```

```
resetVec:
    CLRF EEADR, a; початкова адреса в EEPROM
    LFSR 0, NAME; початкова адреса в ОЗП
    BCF EEPGD; звертання до EEPROM
    BCF CFGS; звертання до FLASH/EEPROM
    MOVLW 10
    MOVWF CL, a; організація циклу на кількість букв в текстовому рядку
M1:
    BSF RD; читання одного символу з EEPROM
    NOP
    MOVFF EEDATA, POSTINC0; перенос символу в ОЗП
    INCF EEADR, f, a; обчислення адреси наступного символу рядка
    DECFSZ CL, f, a; перевірка закінчення циклу
    BRA M1; перехід на початок циклу
    —— продовження програми
```

3. Завдання на самостійну роботу

1. Використовуючи вивчені команди МК PIC18 і шаблон файлу на асемблері із ПЗ2, скласти програму для МК PIC18F4520, яка виконує наступні дії:

– розмістити в ПЗП починаючи з адреси 400H масив з 10 констант: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10;

– розмістити в EEPROM починаючи з адреси 0H текстовий рядок: своє прізвище латинськими буквами;

– задати в ОЗП дві змінні Y1 і Y2 розміром в 4 байта в банку швидкого доступу;

– задати вихідні значення змінних Y1 і Y2 (значення змінних вибрати самостійно);

– перемістити масив констант із ПЗП в масив X1, який розміщується в ОЗП в банку швидкого доступу;

– обчислити контрольну суму масиву X1 і помістити її в змінну CHECKSUM (результат записати у звіт);

– обчислити Y2 = Y2 – Y1 (результат записати у звіт);

– прочитати текстовий рядок з EEPROM і помістити її в змінну NAME, яка розміщується в ОЗП в банку швидкого доступу;

– зациклити програму в нескінченому циклі.

4. Зміст звіту

1. Список групи команд обробки таблиць МК PIC18 та їх призначення.

2. Програма на асемблері для МК PIC18F4520, розроблена за індивідуальним завданням.

3. Результати роботи програми (у десятковій, двійковій і шістнадцятковій системах числення).

5. Контрольні запитання

1. Призначення регістра TABLAT в табличних командах МК PIC18.
2. Призначення регістра TABPTR в табличних командах МК PIC18.
3. Порядок реалізації операції табличного читання з чарунки ПЗП МК PIC18.
4. Порядок реалізації операції табличного запису в ПЗП МК PIC18.
5. Призначення регістрів FSRn в мікроконтролерах PIC18.
6. Способи реалізації непрямої адресації в мікроконтролерах PIC18.
7. Призначення енергонезалежної пам’яті EEPROM.
8. Призначення регістра EEADR в мікроконтролерах PIC18.
9. Призначення регістра EEDATA в мікроконтролерах PIC18.
10. Яка архітектура пам’яті використовується в мікроконтролерах PIC18.

Практичне заняття № 7

РОЗРОБКА ПРОЕКТУ ЦИФРОВОЇ СИСТЕМИ КЕРУВАННЯ НА БАЗІ МІКРОКОНТРОЛЕРА PIC18 В САПР «PROTEUS DESIGN SUITE»

Мета заняття

Вивчити основні можливості пакета проектування електронних пристроїв «Proteus Design Suite», етапи розробки проекту для PIC–мікроконтролерів PIC18, а також приклади тестування цифрової системи керування на базі мікроконтролера PIC18.

1. Короткі теоретичні відомості

Proteus Design Suite – це система автоматизованого проектування (САПР), яка представляє собою пакет програм, що поєднує в собі дві основні програми:

– ISIS – модуль для розробки та моделювання аналогових та цифрових електронних схем в режимі реального часу, у тому числі й на базі мікроконтролерів (МК): мікропроцесор 8086, MCS–51, PIC–мікроконтролери (Microchip), AVR–мікроконтролери (Atmel) та ін.;

– ARES – модуль для розробки та автоматичного трасування друкованих плат.

Інсталяцію демо-версії САПР Proteus Design Suite можна безкоштовно скачати з сайту компанії Labcenter Electronics. [10]

Головною перевагою САПР Proteus є:

– наявність великої кількості бібліотек компонентів, у тому числі й периферійних пристроїв: світлодіодні та рідино–кристалічні індикатори, температурні датчики, годинник реального часу (RTC), інтерактивних елементів вводу–виводу: кнопок, перемикачів, віртуальних портів і віртуальних вимірювальних приладів, інтерактивних графіків, які не завжди присутні в інших подібних програмах;

– якісна графічна візуалізація роботи пристрою, що моделюється.

Основні функції Proteus: [5]

– створення та редагування принципових схем;

– створення та редагування вихідних текстів програмного забезпечення мікроконтролерів;

– моделювання роботи розробленого пристрою;

– розробка та редагування друкованої плати.

2. Етапи розробки проекту пристрою на базі PIC–мікроконтролера

Перед створенням проекту необхідно створити папку для зберігання файлів проекту (наприклад, PZ7). Бажано, усі імена для папки та шляхи до неї називати тільки з використанням англійських букв.

1 етап. Створення нового проекту

1. Запустити САПР Proteus і у вікні «Start» вибрати меню «New Project» або в меню «File» вибрати пункт «New Project».

2. У полі «Name» вести ім’я проекту: pz7.pdsprj, а в полі «Path» вибрати створену заздалегідь папку для зберігання проекту (папку можна вибрати натиснувши кнопку «Browse»). Натиснути кнопку «Next».

3. Вибрати шаблон для розробки принципової схеми (Create a schematics from the selected template): за замовчуванням DEFAULT. Натиснути кнопку «Next».

4. Вибрати шаблон для розробки друкованої плати: Do not create a PCB layout (не створювати друковану плату). Натиснути кнопку «Next».

5. Вибрати тип мікропроцесора або мікроконтролера: Create Firmware Project

– в полі «Family» (Сімейство) вибрати – PIC18;

– в полі «Controller» (Тип мікроконтролера) вибрати – PIC18F4520;

– в полі «Compiler» (Компілятор) – MPLAB XC8.

Натиснути кнопку «Next».

6. Перевірити остаточно всі налаштування проекту й нажати кнопку «Finish». Після цього на екрані з’явиться дві закладки:

– Schematic Capture – принципова схема, на якій уже буде присутній обраний мікроконтролер;

– Source code – вікно із програмою для мікроконтролера, у якому будуть вбудовані секції для введення своєї програми.

2 етап. Створення принципової схеми

1. Перейти на закладку Schematic Capture.

2. Створити базу даних компонентів необхідних для проектування принципової схеми логічного пристрою:

– перейти в режим «Component Mode» – друга з верху іконка у полі інструментів або у вікні «Devices» натиснути синю кнопку «P»;

– з’явиться вікно «Pick Devices», у якому у вікні «Category» будуть згруповані всі доступні компоненти. Після вибору категорії, у правому вікні «Showing local results» з’являється список компонентів. Його можна уточнити по підкатегоріям «Sub–category» і по виробниках компонентів «Manufacturer». Наприклад, для категорії мікропроцесори «Microprocessor ICs», у підкатегорії можна вибрати необхідне сімейство «PIC18 Family» і у вікні «Showing local results» залишаться тільки мікроконтролери сімейства PIC18;

– після вибору компонента у вікні «Showing local results», з’являється його графічне позначення у вікні «Preview». Для того щоб додати компонент в базу даних необхідно два рази клацнути на ньому лівою кнопкою миші. Після вибору усіх необхідних компонентів необхідно натиснути кнопку «OK» для закриття вікна «Pick Devices»;

– для вставки в схему шин живлення (POWER) та «землі» (GROUND) необхідно перейти в режим «Terminals Mode» (восьма з верху іконка у полі інструментів), вставка даних компонентів аналогічна попереднім компонентам.

3. Редагування параметрів компонентів:

– необхідно навести курсор на компонент на схемі й двічі натиснути ліву кнопку миші;

– з’явиться вікно «Edit Component», у якому можна редагувати необхідні параметри: позиційне позначення (Part Reference), опір для резисторів (Resistance),

ємність для конденсаторів (Capacitance) та інші доступні параметри. Перелік параметрів залежить від типу компонента.

4. З'єднання компонентів:

– навести курсор на кінець виводу першого компонента, при цьому курсор прийме вид олівця, а на виводі з'явиться червоний квадрат;

– натиснути один раз ліву кнопку миші, і прокласти лінію з'єднання до виводу другого компонента. Якщо необхідно виконати поворот провідника необхідно щоразу натискати ліву кнопку миші;

– коли на виводі другого компонента з'явиться червоний квадрат, натиснути ліву кнопку миші кнопку ще раз, з'єднання компонентів буде завершено;

– для видалення з'єднання, необхідно виділити провідник (він стане червоним) і натиснути клавішу «Delete»;

– якщо провідників занадто багато, то краще для з'єднання елементів схеми використовувати шину. Для цього необхідно включити режим «Buses Mode» (шоста з верху іконка у полі інструментів). Початок шини наноситься шляхом натискання лівої клавіші миші, для повороту шини необхідно натиснути ще раз ліву клавішу миші, для закінчення малювання шини необхідно двічі натиснути ліву кнопку миші;

– для з'єднання провідників підключених до шини, необхідно привласнити їм однакові імена. Для цього необхідно включити режим «Wire Label Mode» (четверта з верху іконка у полі інструментів)) або вибрати провідник, натиснути праву клавішу й вибрати контекстне меню «Place Wire Label». Відкриється вікно «Edit Wire Label», у поле «String» необхідно ввести ім'я провідника та натиснути кнопку «OK».

3 етап. Створення програмного забезпечення мікроконтролера

1. Перейти на закладку Source code.

2. Набрати у вікні, що відкрилося, текст програми на мові Сі (XC8) для мікроконтролера PIC18. Структура програми аналогічна системі проектування MPLAB X IDE. Програма на мові XC8 має схожу структуру з програмами на мові асемблера: [4]

– розділ заголовків (призначення програми, ім'я автора, дата, версія);

– розділ конфігурації (визначення бітів конфігурації та ідентифікації мікроконтролера за допомогою директиви #pragma config);

– розділ файлів, що підключаються (за допомогою директиви #include);

– розділ визначення глобальних констант та оголошення змінних;

– секція коду (void main (void)) – у цій секції розміщається код програми, при цьому вставляється зациклення програми (while (1) { }), але цю команду можна поміняти відповідно до алгоритму роботи розробляє мого пристрою.

3. Зберегти файл із розширенням «c». Файл за замовчуванням зберігається з іменем «main.c». Щоб перейменувати файл, необхідно у вікні «Projects» натиснути праву кнопку миші на імені файлу й вибрати контекстне меню «Rename». Після цього можна ввести нове ім'я.

4. Компіляція програми:

– вибрати меню: Build / Build Project або нажати клавішу «Ctrl+F7»;

– результати компіляції виводяться у вікні «VSM Studio Output»;

– якщо при компіляції проекту помилок не буде виявлено, то у вікні «VSM Studio Output» буде видане повідомлення «Compiled successfully», тобто можна приступати до тестування програми та моделюванню роботи спроектованого пристрою;

– якщо при компіляції проекту будуть виявлені помилки, то буде видане повідомлення «Error code», а в рядку «ERROR» буде зазначений тип помилки й номер рядка, у якому ця помилка виявлена. Після виправлення помилок, необхідно повторити компіляцію.

4 етап. Моделювання роботи пристрою

Для запуску моделювання роботи проектованого пристрою необхідно зайти в меню «Debug» або скористатися чотирма швидкими кнопками, які розташовані ліворуч у нижньому рядку програми.

У режимі моделювання симулятор дозволяє виконувати наступні операції:

– моделювання роботи пристрою в автоматичному режимі з підтримкою анімації: меню «Debug / Run Simulation», швидка кнопка «  » або клавіша F12;

– покрокове моделювання роботи пристрою: меню «Debug / Start VSM Debugging», швидка кнопка «  » або комбінація клавіш CTRL+F12. Використовується в основному для покрокового виконання програми мікроконтролера;

– для тимчасової зупинки роботи пристрою необхідно включити режим «PAUSE»: меню «Debug / Pause VSM Debugging», швидка кнопка «  » або клавіша «Pause»;

– для виходу з режиму моделювання роботи пристрою необхідно його виключити: меню «Debug / Stop VSM Debugging», швидка кнопка «  » або клавіші SHIFT + «Pause».

3. Налаштування біт конфігурації мікроконтролера PIC18

Біти конфігурації використовуються для вмикання та вимикання окремих вузлів МК, для вибору режиму роботи вузлів МК, а також для захисту змісту пам’яті програм мікроконтролера (ПЗП) від несанкціонованого читання, тобто для захисту від піратського копіювання (біти таємності).

Для налаштування біт конфігурації мікроконтролера PIC18F4520 на мові XC8 використовується директива #pragma config, яка має наступну структуру: [4]

#pragma config ім’я_параметра = значення_параметра

В одній директиві #pragma config можна поєднувати кілька параметрів, розділяючи їх комами. Повний список біт конфігурації і їх значень зберігається у файлі описів конкретної модифікації мікроконтролера.

Список біт конфігурації і їх значень для МК PIC18F4520:

– OSC (вибір режим роботи тактового генератора), може приймати наступні значення: XT, HS, LP, RC, RCIO, EC, ECIO, HSPLL. Приклад налаштування тактового генератора:

#pragma config OSC = HS // високочастотний кварцовий генератор

– IESO (дозвіл перемикання джерела тактового сигналу), може приймати наступні значення: ON, OFF. Приклад налаштування:
#pragma config IESO = OFF // функція перемикання системного ГТІ заборонена

– PWRT (дозвіл роботи таймера вмикання живлення), може приймати наступні значення: ON, OFF. Приклад налаштування:
#pragma config PWRT = ON // робота таймера вмикання живлення дозволена

– BOREN (дозвіл скидання при зниженні напруги живлення), може приймати наступні значення: ON, OFF;

– BORV (установка рівня напруги для модуля BOR), може приймати наступні значення: 45 (4,5 В), 42 (4,2 В), 27 (2,7 В), 20 (2,0 В). Приклад налаштування:
#pragma config BOREN = ON, BORV = 27 // модуль BOR включений, скидання при зниженні напруги живлення нижче 2,7 В

– WDT (дозвіл роботи сторожового таймера), може приймати наступні значення: ON, OFF;

– WDTPS (установка коефіцієнта пост–дільника сторожового таймера), може приймати наступні значення: 1, 2, 4, 8, 16, 32, 64, 128. Приклад налаштування:
#pragma config WDT = ON, WDTPS = 2 // дозвіл роботи сторожового таймера, з коефіцієнтом пост–дільника 1:2

– PBADEN (дозвіл роботи виводів порту PORTB як аналогових входів), може приймати наступні значення: ON, OFF;

– Cpx (біти захисту ПЗП від несанкціонованого читання), може приймати наступні значення: ON, OFF. Приклад налаштування:
#pragma config CP0=OFF, CP1=OFF, CP2=OFF, CP3=OFF, CPB=OFF, CPD=OFF
// захист ПЗП вимкнений.

4. Завдання на самостійну роботу

1. По заданому варіанту (табл. 7.1) розробити принципову схему навчального цифрового пристрою.

Опис роботи цифрового пристрою керування двигуном постійного струму на базі МК PIC18F4520:

– при натисканні кнопки SB1 необхідно включити, за допомогою електромагнітного реле *K*1, двигун постійного струму *M*1;

– при натисканні кнопки SB2 необхідно виключити обертання двигуна постійного струму *M*1;

– при включенні реле *K*1 включити зелений світлодіод *VD*2, а при вимиканні реле *K*1 повинен бути включений червоний світлодіод *VD*1;

– частота роботи МК PIC18F4520 – 4 МГц.

При розробці принципової схеми необхідно підключити до МК наступні вузли:

– типову схему скидання;

– схему синхронізації для кварцового генератора;

– типові схеми підключення двійкових датчиків (кнопки та контакту реле);

– типові схеми підключення обмотки реле та світлодіодів.

Приклад принципової схеми цифрового пристрою керування двигуном постійного струму приведений на рис. 7.1. Для живлення реле *К*1 і двигуна *М*1 необхідно використовувати додаткове джерело живлення з напругу +12В.

Таблиця 7.1 – Варіанти завдань на самостійну роботу

№ варі-анту	Кнопка SB1	Кнопка SB2	Обмотка реле К1	Імена виводів, до яких підк-лючені світлодіоди	Контакт реле К1.2
1	RB0	RB1	RC0	VD1 (RA1) – вмик. лог.1 VD2 (RB2) – вмик. лог.0	RD0
2	RB2	RB3	RC1	VD1 (RA0) – вмик. лог.0 VD2 (RB0) – вмик. лог.1	RD1
3	RB4	RB5	RC2	VD1 (RA2) – вмик.. лог.1 VD2 (RB3) – вмик. лог.0	RD2
4	RB6	RB7	RC3	VD1 (RB4) – вмик. лог.0 VD2 (RA2) – вмик. лог.1	RD3
5	RB2	RB0	RC4	VD1 (RB5) – вмик.. лог.1 VD2 (RA1) – вмик. лог.0	RD4
6	RB6	RB4	RC5	VD1 (RB7) – вмик. лог.0 VD2 (RA2) – вмик. лог.1	RD5
7	RB3	RB1	RC6	VD1 (RB6) – вмик. лог.1 VD2 (RA0) – вмик. лог.0	RD6
8	RB7	RB5	RC7	VD1 (RA1) – вмик. лог.0 VD2 (RA0) – вмик. лог.1	RD7
9	RD1	RC2	RD0	VD1 (RA1) – вмик. лог.1 VD2 (RA2) – вмик. лог.0	RC0
10	RD2	RC3	RD1	VD1 (RA2) – вмик. лог.0 VD2 (RA3) – вмик.лог.1	RC1
11	RC4	RD4	RD2	VD1 (RA0) – вмик. лог.1 VD2 (RA3) – вмик. лог.0	RC2
12	RC5	RD5	RD3	VD1 (RB1) – вмик. лог.0 VD2 (RA2) – вмик. лог.1	RC3
13	RC6	RD6	RD4	VD1 (RB0) – вмик. лог.1 VD2 (RA1) – вмик. лог.0	RC4
14	RC7	RD7	RD5	VD1 (RC1) – вмик. лог.0 VD2 (RA2) – вмик. лог.1	RC5
15	RA0	RB0	RD6	VD1 (RC2) – вмик. лог.1 VD2 (RA1) – вмик. лог.0	RC6
16	RA1	RB3	RD7	VD1 (RC3) – вмик. лог.0 VD2 (RA3) – вмик. лог.1	RC7

2. Розробити алгоритм роботи пристрою й скласти програму на мові Сі (XC8).

Алгоритм і програма повинні бути зациклені в безкінечному циклі, тобто поки є живлення, мікроконтролер повинен контролювати датчики та управляти виконавчими механізмами.

При розробці програми на мові Сі необхідно: [7]

– розділ конфігурації налаштувати біти конфігурації: вибрати режим роботи ГТІ, виключити сторожовий таймер WDT, порти PORTB встановити як цифрові виводи;

– в розділі ініціалізації програми:

– налаштувати режим роботи виводів порту PORTA як цифрові виводи;

– встановити необхідні режими роботи портів вводу–виводу які будуть використовуватись в проекті;

– встановити початкові значення виконавчих пристроїв: реле та світлодіодів.

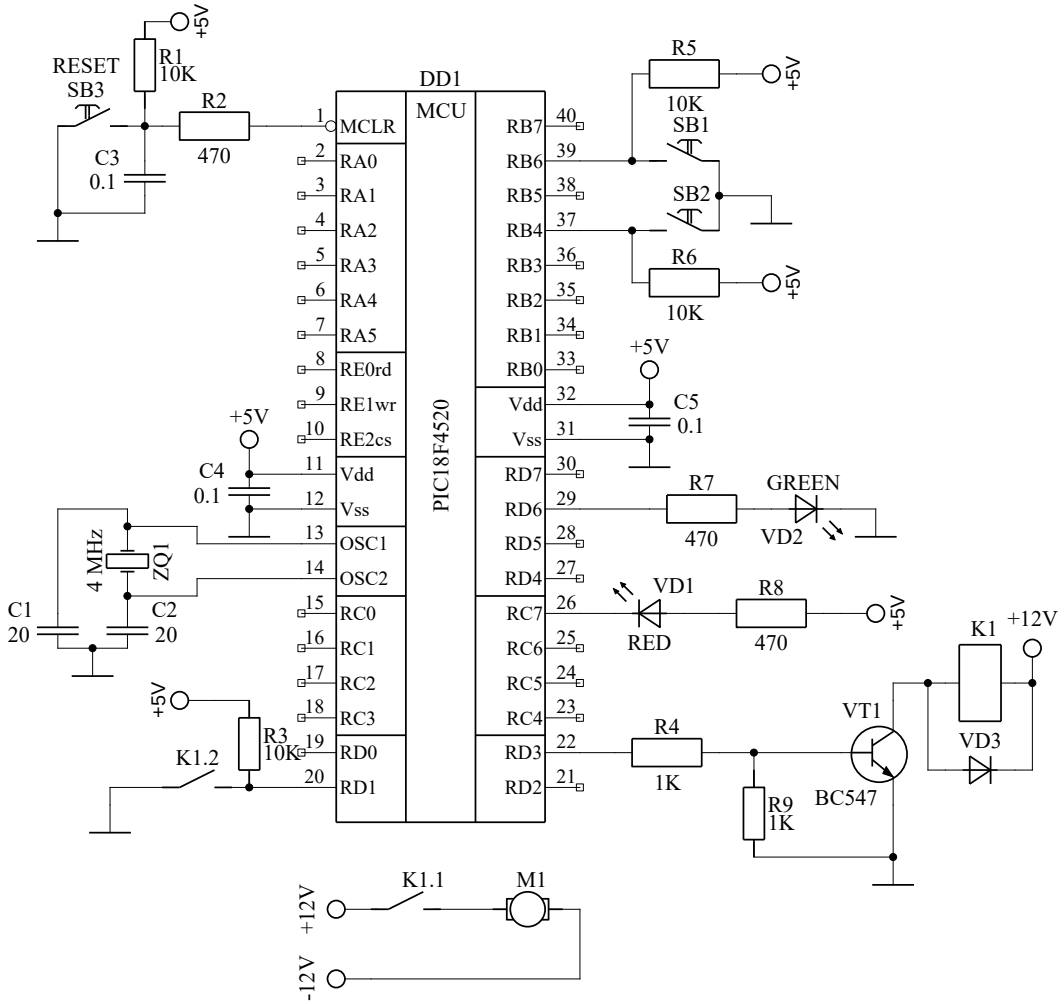

Рис. 7.1. Приклад принципової схеми цифрового пристрою керування двигуном постійного струму

При розробці програми на мові Сі необхідно:

– розділ конфігурації налаштувати біти конфігурації: вибрати режим роботи ГТІ, виключити сторожовий таймер WDT, порти PORTB встановити як цифрові виводи;

– в розділі ініціалізації програми:

– налаштувати режим роботи виводів порту PORTA як цифрові виводи;

– встановити необхідні режими роботи портів вводу–виводу які будуть використовуватись в проекті;

– встановити початкові значення виконавчих пристроїв: реле та світлодіодів.

Приклад налаштування біт конфігурації для МК PIC18F4520

```
#pragma config OSC = XT // вибір XT–генератора (4 МГц)
#pragma config IESO = OFF // функція перемикання системного ГТІ заборонена
#pragma config BOREN = OFF // система BOR вимкнена
#pragma config PWRT = ON // дозвіл роботи таймера вмикання живлення
#pragma config WDT = OFF // вимикання сторожового таймера
#pragma config PBADEN = OFF // після скидання виводи порту PORTB конфігуруються як цифрові виводи
```

3. Розробити проект пристрою на базі МК PIC18F4520 в САПР Proteus

3.1. Створити проект.

3.2. Намалювати принципову схему у вікні «Schematic Capture».

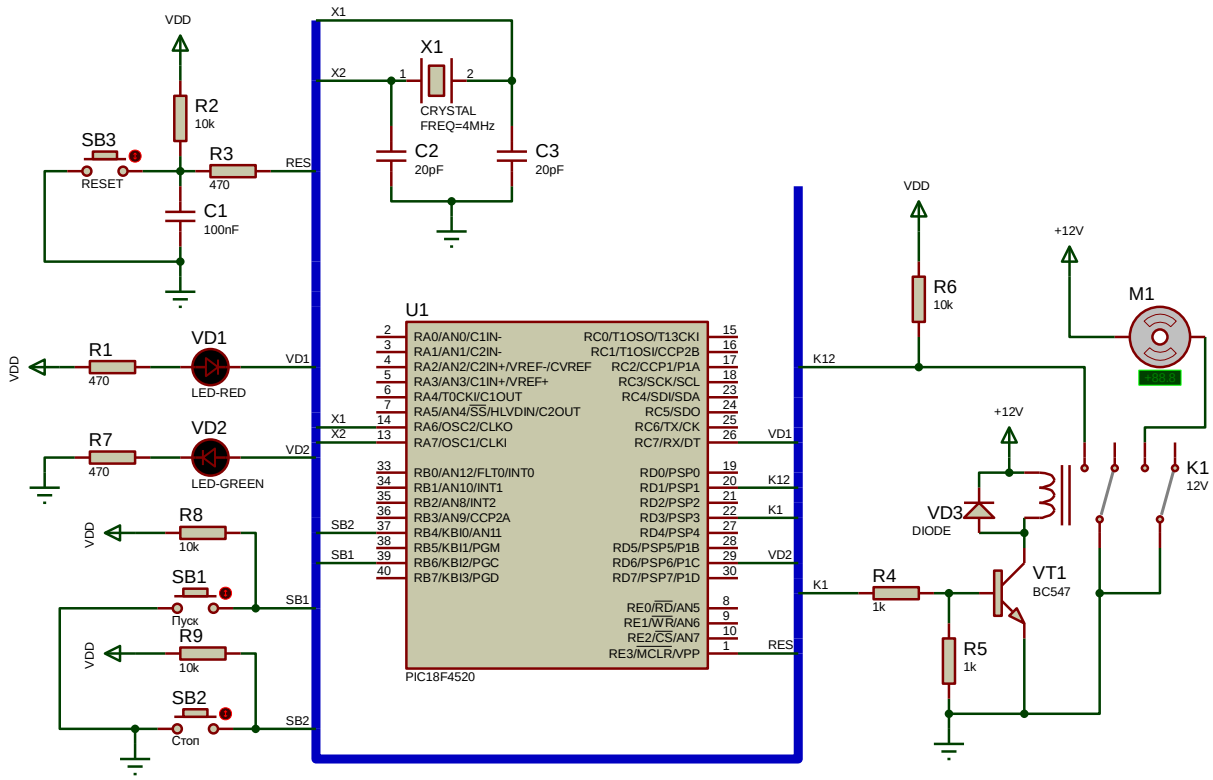

Рис. 7.2. Приклад розробки принципової схеми цифрового пристрою керування двигуном постійного струму в САПР Proteus

При створенні принципової схеми пристрою необхідно створити базу даних компонентів (Категорія –> Підкатегорія –> Ім’я компонента):

– конденсатор: Capacitors –> Generic –> –>CAP або REALCAP;

– діод: Diodes –> Generic –> DIODE;

– двигун: Electromechanical –> Motor–DC;

– кварцовий резонатор: Miscellaneous –> CRYSTAL;
– світлодіоди: Optoelectronics –> Leds –> LED–GREEN і LED–RED;
– резистор: Resistors –> Generic –> RES;
– кнопка: Switches & Relays –> Switches –> BUTTON;
– реле із двома контактами та обмоткою на 12В: Switches & Relays –> Relays (Generic) –> RELAY2P
– транзистор: Transistors –> Bipolar –> BC547 або Generic –> NPN.

3.3. Набрати програму для мікроконтролера у вікні «Source code» (коментарі можна не набирати).

Приклад програми на мові Сі

```
/*
 * Цифрові системи керування
 * Практичне заняття №7 (pz7.c)
 * Автор: Профатилов Володимир
*/
// Налаштування біт конфігурації МК PIC18F4520
#pragma config OSC = XT // вибір XT–генератора (4 МГц)
#pragma config IESO = OFF // функція перемикання системного ГТІ заборонена
#pragma config BOREN = OFF // система BOR вимкнена
#pragma config PWRT = ON // дозвіл роботи таймера вмикання живлення
#pragma config WDT = OFF // вимикання сторожового таймера
#pragma config PBADEN = OFF // після скидання виводи порту PORTB
                            // конфігуруються як цифрові виводи

#include <xc.h>

void main (void)
{
// Налаштування портів вводу–виводу
    ADCON1=0x0F; // налаштувати усі виводи портів A та B як цифрові
    CMCON=0x07; // налаштувати виводи компараторів PORTA як цифрові
    TRISDbits.RD3=0; // налаштувати RD3 на вивід
    TRISDbits.RD6=0; // налаштувати RD6 на вивід
    TRISCbits.RC7=0; // налаштувати RC7 на вивід
// Встановлення початкового значення пристроїв
    PORTDbits.RD3=0; // вимкнути реле K1
    PORTCbits.RC7=1; // вимкнути світлодіод VD1
    PORTDbits.RD6=0; // вимкнути світлодіод VD2
// Робочий цикл управління двигуном
    while (1)
    {
        if (PORTDbits.RD1==1) // контакт K1.2 розімкнутий
        {   PORTDbits.RD6=0; // VD2 (світлодіод GREEN) – вимкнений
```

```
            PORTCbits.RC7=0; // VD1 (світлодіод RED) – включений
        }
        else // контакт К1.2 замкнутий
        {
            PORTCbits.RC7=1; // VD1 (світлодіод RED) – вимкнений
            PORTDbits.RD6=1; // VD2 (світлодіод GREEN) – включений
        }
        if (PORTBbits.RB6==0) //контроль натискання кнопки SB1
        {
            PORTDbits.RD3=1; //включити реле К1
        }
        if (PORTBbits.RB4==0) //контроль натискання кнопки SB2
        {
            PORTDbits.RD3=0; //вимкнути реле К1
        }
    }
}
```

4. Провести моделювання роботи розробленого пристрою

Результати моделювання роботи цифрового пристрою керування двигуном М1 продемонструвати викладачеві.

5. Зміст звіту

1. Основні функції САПР Proteus.
2. Етапи розробки проекту пристрою на базі PIC–мікроконтролера в САПР Proteus.
3. Завдання на самостійну роботу.
4. Принципова схема цифрового пристрою.
5. Алгоритм і програма на мові Сі з коментарями.

6. Контрольні запитання

1. Призначення та переваги САПР «Proteus Design Suite».
2. Основні етапи розробки проекту цифрового пристрою на базі PIC–мікроконтролера в САПР «Proteus Design Suite».
3. Призначення біт конфігурації в мікроконтролерах PIC18.
4. Призначення директиви «#pragma config» в компіляторі XC8 для мови Сі.
5. Призначення портів вводу-виводу в мікроконтролерах PIC18.
6. Призначення регістрів TRISn в мікроконтролерах PIC18.
7. Призначення регістрів PORTn та LATn в мікроконтролерах PIC18.
8. Призначення діоду VD3 на принциповій схемі пристрою (рис. 7.1).
9. Призначення елементу ZQ1 на принциповій схемі пристрою (рис. 7.1).

Практичне заняття № 8

ПРОГРАМУВАННЯ ШІМ У МІКРОКОНТРОЛЕРАХ PIC18

Мета заняття

Вивчити приклади програмування ШІМ на мові Сі для мікроконтролера PIC18, а також навчитися регулювати швидкість обертання двигуна постійного струму за допомогою ШІМ.

1. Навчальний приклад

Схема цифрового пристрою керування двигуном постійного струму наведена на рис. 8.1. Пристрій реалізовано на МК PIC18F4520 із кварцовим ГТІ, який працює на частоті 4 МГц. Алгоритм роботи цифрового пристрою:

– при натисканні (замиканні) кнопки SB1 необхідно ввімкнути двигун зі швидкістю обертання 50% від максимальної швидкості обертання двигуна;

– при натисканні (замиканні) кнопки SB2 необхідно виключити двигун;

– при натисканні кнопки SB3 необхідно збільшити швидкість обертання двигуна на заданий крок;

– при натисканні кнопки SB4 необхідно зменшити швидкість обертання двигуна на заданий крок.

Керування швидкістю обертання двигуна необхідно реалізувати за допомогою широтно–імпульсної модуляції (ШІМ).

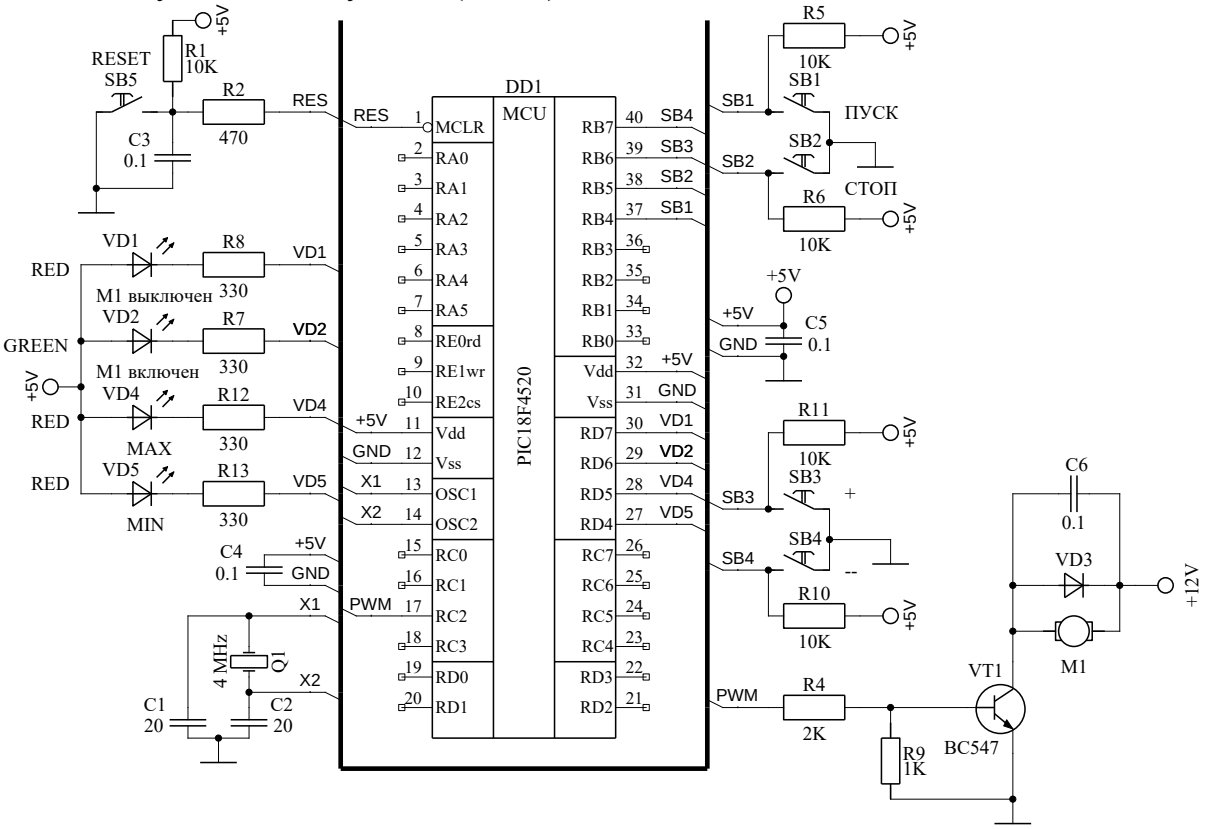

Рис. 8.1. Принципова схема цифрового пристрою керування двигуном постійного струму за допомогою ШІМ

У цифровому пристрої необхідно реалізувати наступну індикацію:

– світлодіод VD1 (червоний) – горить, коли двигун вимкнений;

– світлодіод VD2 (зелений) – горить, коли двигун включений;

– світлодіод VD4 (червоний) – горить, коли двигун обертається з максимальною швидкістю;

– світлодіод VD5 (червоний) – горить, коли двигун обертається з мінімальною швидкістю.

Для керування швидкістю обертання двигуна необхідно реалізувати ШІМ із частотою 1 кГц (період ШІМ 1 мс). Період ШІМ визначається по наступній формулі: [6]

$$_{\text{ШІМ}} = 4 \times \ _{\text{ГТІ}} \times \text{К}_{\text{TMR2}} \times (\text{PR2} + 1)$$

де K_{TMR2} – коефіцієнт переддільника таймера TMR2 (вихідний дільник TMR2 на роботу ШІМ не впливає).

Для реалізації режиму ШІМ разом з модулем CCP1 (модуль захват / порівняння / ШІМ) необхідно використовувати таймер TMR2. Налаштуємо таймер TMR2 для реалізації періоду ШІМ 1 мс (1 кГц):

– регістр періоду PR2 = 249D (період рахування таймера 250 мкс);

– коефіцієнт переддільника таймера TMR2 – К = 1:4;

– керуюче слово для регістру керування таймером TMR2 (T2CON) = 0000.0001B.

Для регулювання швидкості обертання двигуна виберемо наступні установки, які будуть завантажуватися в регістр CCPR1L:

– крок зміни швидкості обертання (STEP) – 5;

– максимальне значення швидкості обертання (MAX) – 245 (якщо значення CCPR1L установити в 250, то ШІМ перетвориться в лог.1);

– мінімальне значення швидкості обертання (MIN) – 5 (якщо значення CCPR1L установити в 0, то ШІМ перетвориться в лог.0);

– швидкість обертання двигуна 50%, при шпаруватості імпульсів ШІМ – 50% (Q50) – 125 (половина від максимального значення тривалості імпульсу ШІМ).

Навчальна програма для практичного заняття № 8

```
/*
 * Цифрові системи керування
 * Програмування ШІМ для регулювання швидкості обертання двигуна
 * Практичне заняття №8 (pz8.c)
 * Автор: Профатилов Володимир
*/
#include <xc.h>

// Налаштування біт конфігурації МК PIC18F4520
#pragma config OSC = XT // вибір XT–генератора (4 МГц)
#pragma config IESO = OFF // функція перемикання системного ГТІ заборонена
#pragma config BOREN = OFF // система BOR вимкнена
#pragma config PWRT = ON // дозвіл роботи таймера вмикання живлення
```

```
#pragma config WDT = OFF // вимикання сторожового таймера
#pragma config PBADEN = OFF // після скидання виводи порту PORTB конфігу-
руються як цифрові виводи

// Оголошення констант
#define STEP 5     // крок зміни швидкості обертання
#define MAX 245 // максимальне значення швидкості обертання
#define MIN 5      // мінімальне значення швидкості обертання
#define Q50 125   // швидкість обертання двигуна 50% (шпаруватість ШІМ – 50%)

void main(void)
{
// налаштування портів вводу–виводу
  ADCON1=0x0F; // налаштувати усі виводи портів A та B як цифрові
  CMCON=7; // налаштувати виводи компараторів PORTA як цифрові
  TRISD=0b00001111; // налаштувати порти RD4–RD7 на вивід
  TRISCbits.RC2=0; // налаштувати порт RC2 на вивід
// Встановлення початкового значення пристроїв
// вимкнути усі світлодіоди та двигун M1
  PORTD=0xFF;
  PORTCbits.RC2=0;
// налаштування таймера TMR2
// K=1:4, T=250 мкс (Fшім = 1 кГц (1 мс) – 0b00000001)
  PR2=249;
  T2CON=0b00000001;
  TMR2=0;
// налаштування CCP1 в режимі ШІМ
  CCPR1L=Q50; // швидкість обертання двигуна 50%
  CCP1CON=0; // модуль CCP1 вимкнений
// основний цикл керування двигуном
  while (1)
  {
Main_Loop:
      PORTDbits.RD6=1; // вимикання світлодіода VD2 (GREEN)
      PORTDbits.RD7=0; // вмикання світлодіода VD1 (RED)
      while (PORTBbits.RB4 == 1) { } // контроль натискання кнопки SB1 (пуск)
// вмикання двигуна M1
      CCP1CON= 0b00001100; // вмикання режиму ШІМ
      T2CONbits.TMR2ON=1; // вмикання таймера TMR2
      PORTDbits.RD7=1; // VD1 – вимикання світлодіода VD1 (RED)
      PORTDbits.RD6=0; // VD2 – вмикання світлодіода VD2 (GREEN)
// контроль натискання кнопки SB2 (стоп) для вимикання двигуна M1
M1:
      if (PORTBbits.RB5==0) // перевірка стану кнопки SB2
      {
```

```
            T2CONbits.TMR2ON=0; // вимикання таймера TMR2
            TMR2=0;
            CCP1CON=0; // вимикання модуля CCP1
            PORTCbits.RC2=0; // вимикання двигуна M1
            goto Main_Loop; // зациклення основної програми
        }
// контроль натискання кнопки SB3 (+) збільшення швидкості обертання двигуна
        if (PORTBbits.RB6==0) // перевірка стану кнопки SB3
        {
        if (CCPR1L<MAX) // перевірка максимальної швидкості
        { CCPR1L=CCPR1L+STEP;} // збільшення швидкості обертання
        if (CCPR1L>=MAX) // установка світлодіодів швидкості
        { PORTDbits.RD5=0;} // вмикання VD4 (швидкість максимальна)
        else
        {
        PORTDbits.RD4=1; // вимикання світлодіода VD5 (швидкість мінімальна)
        PORTDbits.RD5=1; // вимикання світлодіода VD4 (швидкість максимальна)
        }
        while (PORTBbits.RB6 == 0) { } // перевірка відпускання кнопки SB3 (+)
        }
// контроль натискання кнопки SB4 (–) зменшення швидкості обертання двигуна
        if (PORTBbits.RB7==0) // перевірка стану кнопки SB4
        {
        if (CCPR1L>MIN) // перевірка мінімальної швидкості
        { CCPR1L=CCPR1L–STEP;} // зменшення швидкості обертання
        if (CCPR1L<=MIN) // установка світлодіодів швидкості
        { PORTDbits.RD4=0;} // вмикання світлодіод VD5 (швидкість мінімальна)
        else
        {
        PORTDbits.RD4=1; // вимикання світлодіода VD5 (швидкість мінімальна)
        PORTDbits.RD5=1; // вимикання світлодіода VD4 (швидкість максимальна)
        }
        while (PORTBbits.RB7 == 0) { } // перевірка відпускання кнопки SB4 (–)
        }
        goto M1; // зациклення основної програми
  } // end while
} // end main
```

2. Завдання на самостійну роботу

2.1. Розробити по заданому варіанту (табл. 8.1) цифровий пристрій для керування двигуном постійного струму за допомогою ШІМ

1. Розробити принципову схему цифрового пристрою на базі МК PIC18F4520 для керування двигуном постійного струму за допомогою ШІМ.

2. Провести розрахунки режиму ШІМ:

– визначити значення регістру періоду PR2;

– скласти керуюче слово для регістру T2CON;

– визначити константи для регулювання швидкості обертання двигуна.

3. Скласти програму для МК PIC18F4520 використовуючи навчальний приклад програми для пристрою, принципова схема якого наведена на рис. 8.1. Програму обов’язково зациклити в нескінченному циклі.

Таблиця 8.1 – Варіанти індивідуальних завдань

№ варіанта	Кнопка SB1 (пуск)	Кнопка SB2 (стоп)	Кнопка SB3 (+)	Кнопка SB4 (–)	Частота ШІМ, Гц (при частоті ГТІ, МГц)
1	RB0	RB1	RA1	RA0	4000 (4)
2	RB2	RB3	RA2	RA3	1000 (8)
3	RB4	RB5	RC0	RC1	2000 (4)
4	RB6	RB7	RC3	RC5	2000 (8)
5	RB2	RB0	RC4	RC6	250 (4)
6	RB6	RB4	RD0	RD2	2500 (8)
7	RB3	RB1	RD1	RD3	625 (4)
8	RB7	RB5	RD2	RD4	1250 (8)
9	RD1	RC7	RA0	RA2	500 (4)
10	RD2	RC3	RA1	RA3	500 (8)
11	RC4	RD4	RB0	RB1	2500 (4)
12	RC5	RD5	RB2	RB3	625 (8)
13	RC6	RD6	RB4	RB5	1250 (4)
14	RC7	RD7	RD0	RD1	4000 (8)
15	RA0	RB0	RD2	RD4	500 (4)
16	RA1	RB3	RD3	RD5	1000 (8)

2.2. Провести моделювання роботи пристрою з використанням САПР «Proteus Design Suite»

1. Створити новий проект: «pz8.pdsprj» (на базі попереднього проекту із ПЗ7). Для цього необхідно скопіювати всі файли проекту в нову папку проекту «pz8», відкрити проект у новій папці й зберегти його з новим іменем: File –> Save Project As.

2. Намалювати принципову схему пристрою (приклад на рис. 8.2) та набрати програму на мові Сі.

3. Налаштувати частоту роботи мікроконтролера у відповідності зі своїм варіантом завдання.

4. Налаштувати параметри двигуна *M*1 для швидкої реакції на зміни керуючого сигналу:

– індуктивність обмотки (Coil Inductance) – 1000 мГн;

– обертаючий момент (Torque) Load/Max – 1 %;

– ефективна маса (Effective Mass) – 0.00001.

5. Виконати моделювання роботи цифрового пристрою:

– підключити до схеми осцилограф для спостереження за імпульсами ШІМ (вибрати режим «Instruments», а в ньому вимірювальний прилад Oscilloscope): канал «А» підключити до сигналу «PWM», а канал «В» до кола колектора транзистора VT1;

– перевірити роботу пристрою керування двигуном змінюючи швидкість обертання у всьому діапазоні регулювання. Для спостереження за зміною регістру CCPR1L необхідно вивести вікно Debug / Watch Window. Нажати в цьому вікні праву клавішу та вибрати меню «Add Items (by Name)», вибрати регістр CCPR1L і нажати кнопку «Done»;

– продемонструвати результати викладачеві.

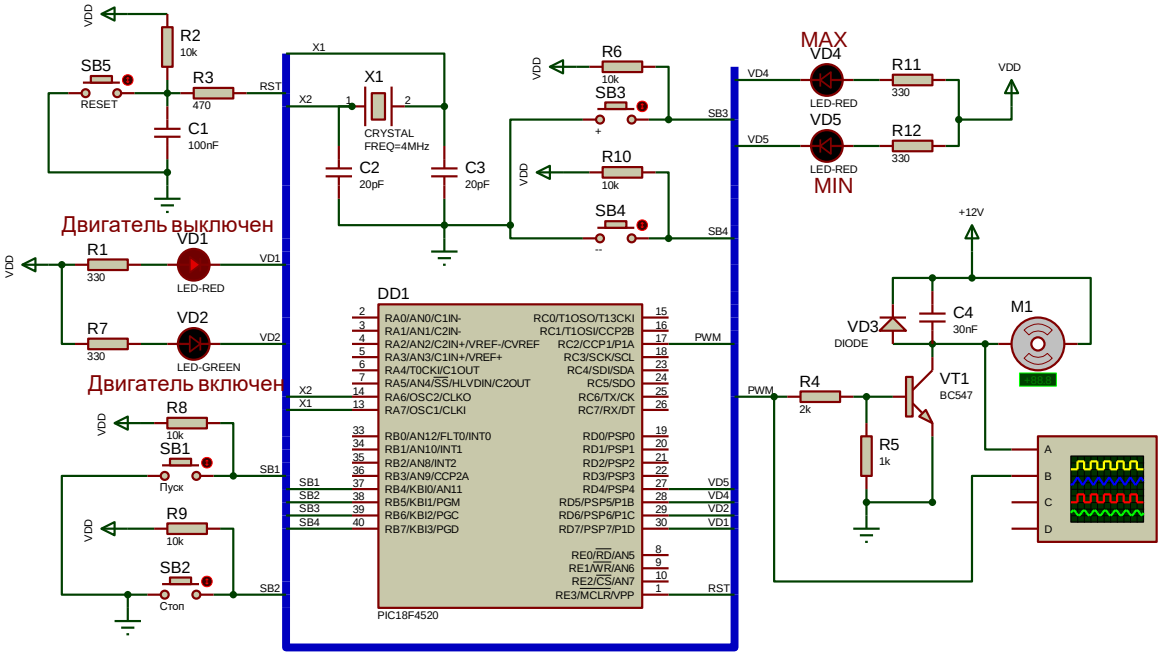

Рис. 8.2. Приклад принципової схеми пристрою керування двигуном постійного струму за допомогою ШІМ в САПР Proteus

3. Зміст звіту

1. Завдання до самостійної роботи.
2. Принципова схема розробленого цифрового пристрою.
3. Розрахунки режиму ШІМ для керування двигуном.
4. Програма на мові Сі для МК PIC18F4520, розроблена по заданому варіанту.

5. Контрольні запитання

1. Призначення модуля CCP в мікроконтролерах PIC18.
2. Принцип керування двигуном постійного струму за допомогою ШІМ.
3. Порядок розрахунку періоду ШІМ в мікроконтролерах PIC18.
4. Призначення регістрів CCPR1L та CCPR1H в режимі ШІМ.
5. Яким чином задається тривалість імпульсу в режимі ШІМ в МК PIC18.

Практичне заняття № 9

ПРОГРАМУВАННЯ ТАЙМЕРІВ НА МІКРОКОНТРОЛЕРАХ PIC18

Мета заняття

Навчитись програмувати таймери на мікроконтролера PIC18 для реалізації часових функцій та використовувати систему переривання для фіксації подій, а також навчитися імітувати подачу зовнішніх сигналів на входи МК.

1. Навчальний приклад

Схема цифрового пристрою керування приведена на рис. 9.1. Пристрій реалізовано на МК PIC18F4520 із кварцовим ГТІ, який працює на частоті 4 МГц. Алгоритм роботи пристрою:

– при натисканні кнопки SB1 необхідно включити миготіння світлодіода VD1 з періодом 1 секунда;

– при натисканні кнопки SB2 необхідно вимкнути миготіння світлодіода VD1;

– часовий інтервал реалізувати за допомогою таймера TMR2 і системи переривання мікроконтролера PIC18.

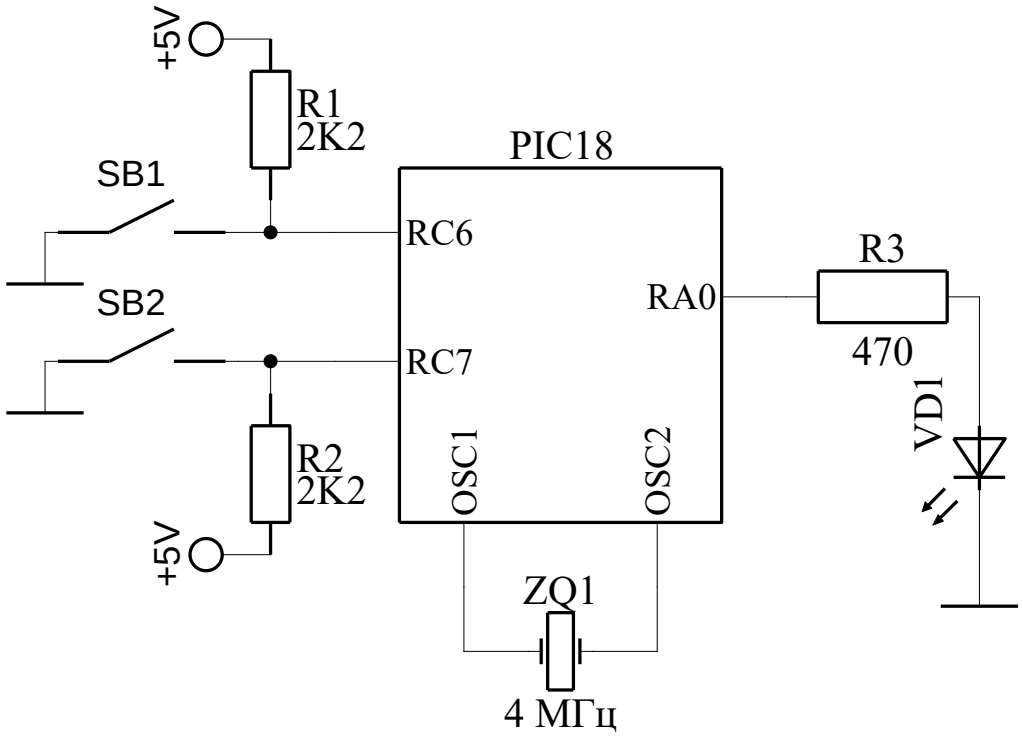

Рис. 9.1. Спрощена схема навчального цифрового пристрою керування

Навчальна програма для практичного заняття № 9

```
/* Цифрові системи керування
Навчальна програма для практичного заняття №9
Автор: Володимир Профатилов
*/
```

```
PROCESSOR 18F4520 // тип мікроконтролера
#include <xc.inc> // підключення опису МК PIC18

// визначення бітів конфігурацій мікроконтролера
CONFIG OSC = XT // вибір XT–генератора (4 МГц)
CONFIG IESO = OFF // функція перемикання системного ГТІ заборонена
CONFIG BOREN = OFF // система BOR вимкнена
CONFIG PWRT = ON // дозвіл роботи таймера вмикання живлення
CONFIG WDT = OFF // вимикання сторожового таймера
CONFIG PBADEN = OFF // після скидання виводи порту PORTB конфігуруються як цифрові виводи
;————————————————————————————————————————
// оголошення глобальних змінних
GLOBAL COUNT
; розміщення змінних в банку швидкого доступу (ACCESS BANK)
psect udata_acs
COUNT: DS 1 ; лічильник переповнень таймера TMR2
;————————————————————————————————————————
// програмний код
PSECT resetVec, abs, class=CODE, reloc=2
    ORG 0 ; початкова адреса програми після скидання МК
resetVec:
    GOTO INIT; перехід на ініціалізацію МК
;————————————————————————————————
; підпрограма обробки переривань
PSECT intVec, abs, class=CODE, reloc=2
    ORG 8h ; вектор підпрограми обробки переривань
    BTFSS TMR2IF; аналіз прапора переповнення таймера TMR2
    BRA TR_EXIT; вихід із підпрограми обробки переривання
; обробка переривання від TMR2
INT_TMR2:
    INCF COUNT, f, a; підрахунок інтервалів dT = 10мс
    MOVLW 100
    CPFSEQ COUNT,a; перевірка часового інтервалу T = 1 с.
    BRA TR_EXIT; перехід, якщо 1 секунда не пройшла
    CLRF COUNT,a; скидання лічильника переповнень таймера TMR2
    BTG PORTA, 0,a; інвертування стану світлодіода VD1 (миготіння)
TR_EXIT:
    BCF TMR2IF; скидання прапора переповнення таймера TMR2
    RETFIE 1; повернення в основну програму з відновленням контексту
;————————————————————————————————
PSECT mainprog, abs, class=CODE, reloc=2
    ORG 30h
; ініціалізація програми
```

```
INIT:
; налаштування портів та таймера TMR2
; порт A
    MOVLW 0x0F
    MOVWF ADCON1,a; налаштувати усі виводи портів A та B як цифрові
    MOVLW 07H
    MOVWF CMCON,a; налаштувати виводи компараторів PORTA як цифрові
    CLRF LATA,a; вимкнути світлодіод VD1
    MOVLW 11111110B
    MOVWF TRISA,a; налаштування виводу порта RA0 на вивід
; таймер TMR2: переддільник 1:4, постдільник 1:9 (переривання dT = 10 мс)
    MOVLW 01001001B
    MOVWF T2CON, a
    CLRF TMR2, a; ініціалізація таймера TMR2
    MOVLW 249; переповнення кожні 10 мс
    MOVWF PR2,a; ініціалізація регістру періоду таймера TMR2
; налаштування системи переривань
    BCF RCON,7,a; вимикання пріоритетної системи переривань
    CLRF PIR1,a; обнуління прапорів переривань периферійних пристроїв
    CLRF PIR2,a; обнуління прапорів переривань периферійних пристроїв
    CLRF PIE1,a; вимикання біт дозволу переривань від периферійних пристроїв
    CLRF PIE2,a; вимикання біт дозволу переривань від периферійних пристроїв
    BSF TMR2IE; дозвіл переривання від таймера TMR2
; дозвіл роботи системи переривань і периферійних переривань
    MOVLW 11000000B
    MOVWF INTCON, a
;_____________________________________________
; основна програма
MAIN:
; ініціалізація змінних
    CLRF COUNT,a; скидання лічильника переповнень таймера TMR2
; контроль замикання кнопки SB1
MN1:
    BTFSC PORTC, 6, a
    BRA MN1; очікування замикання кнопки SB1
    BSF TMR2ON; вмикання таймера TMR2
; контроль замикання кнопки SB2
MN2:
    BTFSC PORTC, 7, a
    BRA MN2; очікування замикання кнопки SB2
    BCF TMR2ON; вимикання таймера TMR2
    BCF PORTA, 0,a; вимикання світлодіода VD1
    CLRF TMR2,a; обнуління таймера TMR2
    BRA MAIN; зациклення основної програми
    END resetVec; кінець програми (точка входу в програму resetVec)
```

2. Завдання на самостійну роботу

2.1. По заданому варіанту (табл. 9.1) скласти програму для МК PIC18F4520

При замиканні першої кнопки SB1 необхідно включити миготіння світлодіодів із заданим періодом часу. При замиканні другої кнопки SB2 вимкнути миготіння світлодіодів. Часову затримку реалізувати за допомогою таймера та системи переривань мікроконтролера. Програму обов'язково зациклити в нескінченному циклі.

Таблиця 9.1 – Варіанти завдань на самостійну роботу

№ варіанта	Кнопка SB1	Кнопка SB2	Період миготіння, с (при частоті ГТІ, МГц)	Імена виводів, до яких підключені світлодіоди	Тип таймера
1	RB0	RB1	0,5 (4)	Одночасне миготіння світлодіодів VD1 (RA1) і VD2 (RB2)	TMR0
2	RB2	RB3	1 (4)	Почергове миготіння світлодіодів VD1 (RA0) і VD2 (RB0)	TMR1
3	RB4	RB5	2 (4)	Миготіння світлодіода VD1 – RA2	TMR2
4	RB6	RB7	2 (4)	Одночасне миготіння світлодіодів VD1 (RB4) і VD2 (RA2)	TMR0
5	RB2	RB0	0,5 (4)	Миготіння світлодіода VD1 – RB5	TMR1
6	RB6	RB4	1 (16)	Почергове миготіння світлодіодів VD1 (RB7) і VD2 (RA2)	TMR2
7	RB3	RB1	0,5 (8)	Миготіння світлодіода VD1 – RB6	TMR0
8	RB7	RB5	2 (4)	Почергове миготіння світлодіодів VD1 (RA1) і VD2 (RA0)	TMR1
9	RD1	RC2	2 (16)	Одночасне миготіння світлодіодів VD1 (RA1) і VD2 (RA2)	TMR2
10	RD2	RC3	2 (8)	Почергове вмикання світлодіодів VD1 (RA2) і VD2 (RA3)	TMR0
11	RC4	RD4	1 (8)	Миготіння світлодіода VD1 – RA0	TMR1
12	RC5	RD5	0,5 (8)	Одночасне миготіння світлодіодів VD1 (RB1) і VD2 (RA2)	TMR2
13	RC6	RD6	1 (4)	Миготіння світлодіода VD1 – RB0	TMR0
14	RC7	RD7	2 (8)	Одночасне миготіння світлодіодів VD1 (RC1) і VD2 (RA2)	TMR1
15	RA0	RB0	0,5 (4)	Почергове вмикання світлодіодів VD1 (RC2) і VD2 (RA1)	TMR2
16	RA1	RB3	1 (4)	Миготіння світлодіода VD1 – RC3	TMR0

2.2. Провести тестування роботи програми з використанням програмного симулятора MPLAB X SIMULATOR

1. Встановити частоту машинних циклів (МЦ) МК (частота МЦ для МК PIC18 буде в 4 рази менше частоти ГТІ МК) [8]:

– натиснути в панелі навігатора (нижнє ліве вікно – закладка Dashboard) кнопку  (Project Properties) або ліву клавішу миші у вікні Dashboard на імені пакета Packs – PIC18Fxxxx_DFP(1.1.19);

– вибрати категорію «Simulator» і встановити частоту МЦ (Instruction Frequency) у відповідності зі своїм варіантом;

– нажати кнопку «Apply» і кнопку «OK».

2. У покроковому режимі перевірити правильність ініціалізації модулів мікроконтролера та змінних програми. Для цього вивести у вікно спостереження «Watches» усі оголошені в програмі змінні й використовувані регістри спеціальних функцій.

3. У покроковому режимі перевірити роботу перемикачів відповідно до алгоритму роботи програми. Для імітації зовнішніх сигналів, що надходять від зовнішніх пристроїв або датчиків, використовується моделювання логічних станів портів уведення – виводу настроєних на введення інформації:

– викликати меню: Window / Simulator / Stimulus;

– перейти на закладку «Asynchronous» для моделювання асинхронних сигналів (тобто сигналів вступників у довільні моменти часу);

– натиснути кнопку «Add a row» ()і додати у вікно необхідна кількість рядків, яка повинна збігаюся з кількістю сигналів, що моделюються;

– вибрати в колонку «Pin» ім’я виводу PIC – мікроконтролера, на який буде подаватися зовнішній сигнал;

– вибрати в колонку «Action» тип імітації вхідного сигналу: Toggle (перемикання (інвертування) сигналу на вході);

– у поле коментарі (Comments) можна підписати імена кнопок і їх призначення. Наприклад, «Кнопка SB1 (старт)» і «Кнопка SB2 (стоп)»;

– натиснути кнопку «Save to stimulus workbook» ()і зберегти настроювання у файлі: pz9.sbs.

Для подачі сигналу на вхід мікроконтролера необхідно нажати кнопку в колонку «Fire».

3. Перевірити період миготіння світлодіодів в автоматичному режимі:

– для спостереження за станом світлодіодів і кнопок можна використовувати вікно «I/O Pins»: Window / Simulator / IOpin. Вибрати в колонку «Pin» ім’я виводів PIC – мікроконтролера, за яким необхідно спостерігати;

– вивести на екран вікно для перевірки часових параметрів програми: Window / Debugging / Stopwatch;

– поставити крапку зупинки на команду миготіння світлодіодів і запустити програму в автоматичному режимі: меню Debug / Continue або натискаючи клавішу F5 або кнопку  .

3. Зміст звіту

1. Завдання до самостійної роботи.
2. Принципова схема розробленого цифрового пристрою.
3. Розрахунки часових затримок для таймера.
4. Програма на асемблері для МК PIC18F4520, розроблена по заданому варіанту.

4. Контрольні запитання

1. Призначення таймерів в мікроконтролерах.
2. Призначення системи переривань в мікроконтролерах.
3. Призначення вікна Stopwatch в MPLAB X IDE.
4. Призначення регістра керування перериваннями INTCON.
5. Які керуючі біти має кожне джерело переривань в МК PIC18.
6. Порядок обробки переривань у мікроконтролері PIC18.
7. Структура модуля таймера TMR0 МК PIC18.
8. Структура модуля таймера TMR1 МК PIC18.
9. Структура модуля таймера TMR2 МК PIC18.
10. Порядок розрахунку часових затримок для реалізації за допомогою таймерів МК PIC18.
11. Реалізація тривалих часових затримок за допомогою таймерів МК PIC18.

СПИСОК ЛІТЕРАТУРИ ТА ІНФОРМАЦІЙНІ РЕСУРСИ

1. MPLAB X IDE User’s Guide [Електронний ресурс] / DS–50002027F. – Microchip Technology Inc., 2022. – 387 p. – Режим доступу: https://www.microchip.com/en–us/tools–resources/develop/mplab–x–ide

2. MPLAB XC8 PIC Assembler User's Guide [Електронний ресурс] / DS50002974D. – Microchip Technology Inc., 2023. – 226 p. – Режим доступу: https://www.microchip.com/en–us/tools–resources/develop/mplab–xc–compilers/xc8

3. PIC18F4520 Data Sheet [Електронний ресурс] / DS39631E. – Microchip Technology Inc., 2008. – 412 p.

4. MPLAB XC8 C Compiler User’s Guide for PIC MCU [Електронний ресурс] / DS50002737G. – Microchip Technology Inc., 2022. – 382 p. – Режим доступу: https://www.microchip.com/en–us/tools–resources/develop/mplab–xc–compilers/xc8.

5. Новацький А. О. Мікропроцесорні та мікроконтролерні системи: підручник. Ч. 2. Проектування мікропроцесорних систем [Електронний ресурс] / А. О. Новацький. – Київ: КПІ ім. Ігоря Сікорського, Вид–во «Політехніка», 2021. – 462 с.

6. Смірнов, В.В. Програмування мікроконтролерних систем [Текст]: навчальний посібник / В.В. Смірнов, Н.В. Смірнова, Ю.М. Пархоменко; Центральноукраїн. нац. техн. ун–т. – Кропивницький: ЦНТУ, 2021. – 262 с.

7. Subero, Armstrong. Programming PIC Microcontrollers with XC8 series [Text] / Armstrong Subero. – Apress, 2018. – 434 pages.

8. Rafiquzzaman, M. Microcontroller theory and applications with the PIC18F [Text] / M. Rafiquzzaman. – John Wiley & Sons, 2018. – 508 p.

9. https://www.microchip.com – сайт компанії виробника мікроконтролерів Microchip, містить документацію на PIC–мікроконтролери та AVR–мікроконтролери, приклади використання мікроконтролерів, а також безплатну версію середовища програмування IDE MPLAB X.

10. https://www.labcenter.com – сайт компанії розробника системи автоматизованого проектування Proteus Design Suite.

Додаток А

ПРАВИЛА ОФОРМЛЕННЯ ЗВІТУ З ПРАКТИЧНИХ ЗАНЯТЬ

1. Формат листів:

– А4 (без рамок), орієнтування – книжне;

– поля: ліве і праве – по 2,25 см, верхнє – 1,8 см, нижнє – 3,0 см.

2. Шрифт тексту:

– тип – Times New Roman (TNR);

– розмір – 14;

– в набраному тексті не повинно бути шрифтів іншого типу.

3. Параметри абзацу для основного тексту:

– міжрядковий інтервал – 1,0 (одинарний);

– абзацний відступ на 0,77 см від початку рядка, однаковий по всьому основному тексту;

– вирівнювання абзаців – по ширині;

– абзаци основного тексту не відділяються один від одного.

4. Параметри абзацу для заголовків розділу:

– шрифт TNR, розмір 14, напівжирний;

– вирівнювання абзаців – по центру;

– відступу першого рядка не має;

– від попереднього тексту підзаголовок відділяється інтервалом в 12 пунктів, а наступного – 6 пунктів.

5. Зразок оформлення рисунків:

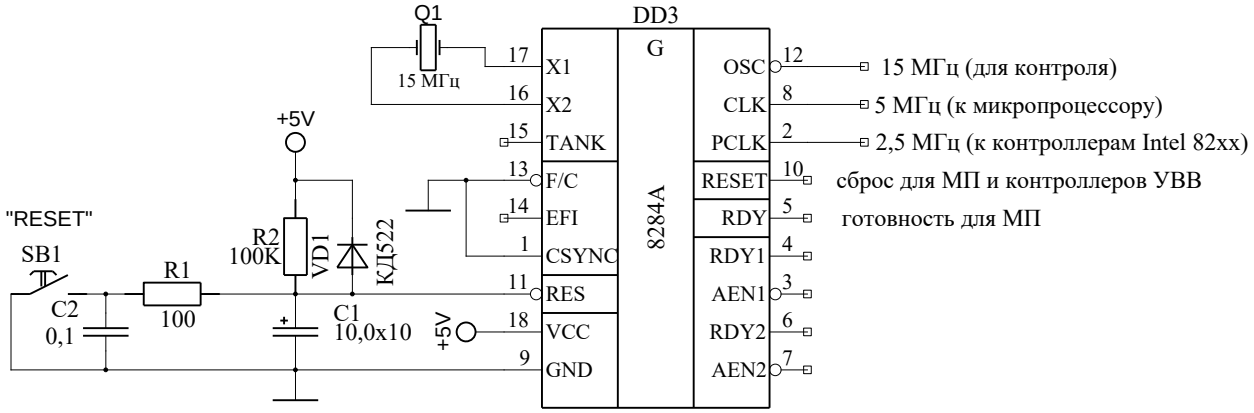

Рис. 1. УГО ИМС i8284A

Параметри абзацу для рисунків та підписом під рисунком:

– вирівнювання абзаців – по центру;

– відступу першого рядка не має;

– від попереднього тексту рисунок та підпис відділяється інтервалом в 6 пунктів, а наступного – 12 пунктів.

Навчально–методичне видання

Профатилов Володимир Іванович

ЦИФРОВІ СИСТЕМИ КЕРУВАННЯ

Навчально–методичні рекомендації до практичних занять

Електронне видання

Експертний висновок склала канд. техн. наук, доц. Тетяна Сердюк

Зареєстровано НМВ УДУНТ (№ 1.834 від 17.09.2025)

В авторській редакції
Комп’ютерна верстка Профатилов В.І.

Формат 60х84 $^{1/16}$. Ум. друк. арк. 3,72. Обл.–вид. арк. 3,76
Зам. № 96

Видавець: Український державний університет науки і технологій
вул. Лазаряна, 2, м. Дніпро, 49010.
Свідоцтво суб’єкта видавничої справи ДК № 7709 від 14.12.2022

Адреса видавця та дільниці оперативної поліграфії:
вул. Лазаряна, 2, м. Дніпро, 49010