МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

«МІКРОПРОЦЕСОРНІ СИСТЕМИ»

для студентів спеціальності
141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА
освітньої програми «Електромеханічні системи автоматизації та електропривод»
усіх форм навчання

2023

Методичні вказівки до лабораторних робіт з дисципліни «Мікропроцесорні системи» для студентів спеціальності 141 – ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА освітньої програми «Електромеханічні системи автоматизації та електропривод» усіх форм навчання. /Укл: В.В. Осадчий, Е.М. Кулинич, О.С. Назарова. - Запоріжжя: НУ «Запорізька політехніка», 2023. – 62 с.

Укладачі:

В.В. Осадчий, к.т.н., доцент
Е.М. Кулинич, к.т.н., доцент
О.С. Назарова, к.т.н., доцент

Рецензент: А.В. Пирожок, к.т.н., доцент

Відповідальний за випуск: О.С. Назарова, к.т.н., доцент

Затверджено
на засіданні кафедри
Електропривода і автоматизації
промислових установок
протокол № 06 від 22.02.2023 р.

Рекомендовано
до видання НМК ЕТФ
протокол № 07 від 23.03.2023 р.

ЗМІСТ

ПЕРЕДМОВА

Методичні вказівки складаються з двох розділів і одного додатку і мають рекомендації до виконання лабораторних робіт з дисципліни «Мікропроцесорні системи» у відповідності до навчальних планів ОКР бакалаврів.

В першому розділі наведені короткі відомості про інструментальне програмне забезпечення для розробки та налагодження програм, а також методичні вказівки щодо їх написання. У другому розділі наведено перелік лабораторних робіт, їх короткий зміст та рекомендації з їх виконання.

У додатку подано перелік команд мікроконтролера Intel 8051, який включає назву, мнемокод та опис дій, що виконуються.

Для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітньої програми «Електромеханічні системи автоматизації та електропривод» усіх форм навчання.

1 РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

1.1 Засоби ProView для налагодження взаємодії з об'єктами керування.

Для прискорення процесу розробки керуючих програм існують спеціальні програмні засоби, що називаються інструментальним програмним забезпеченням. Одним з таких засобів є Proview від Franklin Software. Вказане інструментальне програмне забезпечення (ПЗ) дозволяє набирати, редагувати та зберігати вихідний текст програми, перетворювати його в машинний код, здійснювати перевірку працездатності програми, як окремих її частин, так і вцілому.

Інтегровані програмні емулятори апаратних засобів дозволяють імітувати роботу портів і таймерів мікроконтролера, обробляти зовнішні переривання й переривання від таймерів.

Інструментальне ПЗ Proview фірми Franklin Software Inc. містить декілька вікон, що призначені для налагодження взаємодії мікроконтролера з об'єктами керування. У вікні Main Registers (рисунок 1.1) відображається вміст лічильника команд PC, акумулятора ACC, слова стану процесу PSW, показчика стеку SP, показчика даних DPTR, допоміжного акумулятора B, біту позики/перенесення C, регістру масок переривань IE; регістру блокувань преривань EA, показчика банку регістрів RB, регістрів загального призначення R0-R7; портів P0-P3, регістру керування/статусу таймера TCON, регистрів таймерів THL0-THL2; регістру керування потужністю PCON.

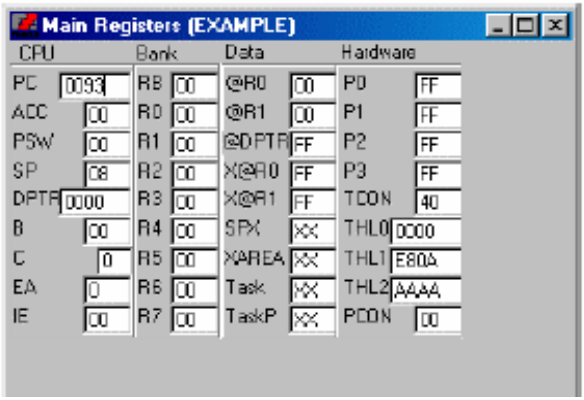


Рисунок 1.1 – Вікно регістрів

Крім того, через пункт Hardware меню View (рисунок 1.2) можна відкрити вікна паралельних портів контролера переривань, таймерів і послідовного порту.

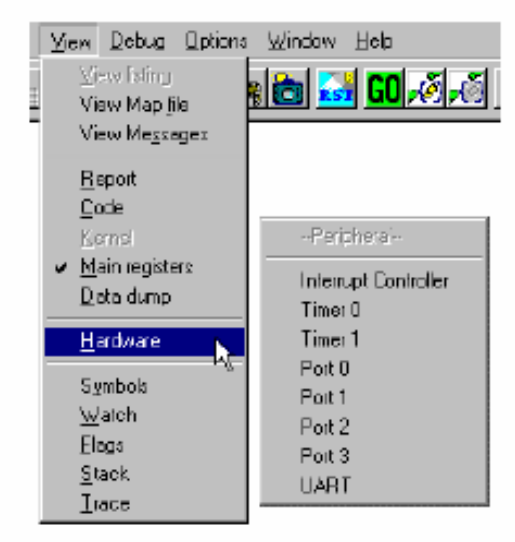


Рисунок 1.2 – Меню View

У інформаційному вікні контролера переривань «Interrupt Controller» (рисунок 1.3) вказується статус і пріоритет всіх джерел

переривань: «Enable» - наявність переривання, «Not Enable» - відсутність переривання.

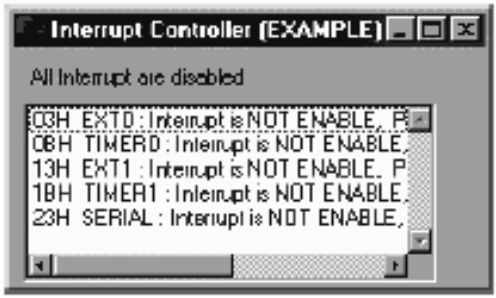
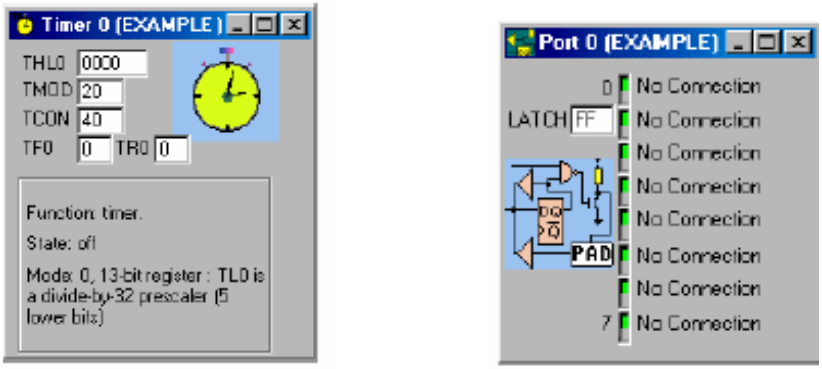


Рисунок 1.3 – Вікно контролера переривань

В рядку стану (рисунок 1.4) при виконанні програми у покроковому режимі або в режимі анімації показується поточний реальний час.

Рисунок 1.4 – Рядок стану



а) б)

а - вікно стану таймеру; б - вікно стану паралельного порту

Рисунок 1.5 – Вікна Proview таймера та паралельного порту

У вікнах таймерів (рисунок 1.5, а) відображається вміст 16-бітного таймера/лічильника THLx, регістру режиму роботи

таймера/лічильника TMOD, регістра керування/статусу таймера TCON, флаг переповнення таймера TFx, біт керування таймера TRx. В цьому вікні також вказується стан і режим роботи таймера.

У вікнах паралельних портів (рисунок 1.5, б) відображається вміст регістру-фіксатора LATCH і окремих ліній порту. Стан ліній порту може бути змінено за допомогою миші.

У вікні послідовного порту UART (рисунок 1.6) відображаються дані буфера передавача Buffer, режим роботи і швидкість. У буфер приймача Input може бути введена послідовність байтів. Інформація може бути представлена у символьному вигляді ASCII або у шістнадцятковій формі HEX. Буфер очищується за допомогою кнопки Reset Buffer.

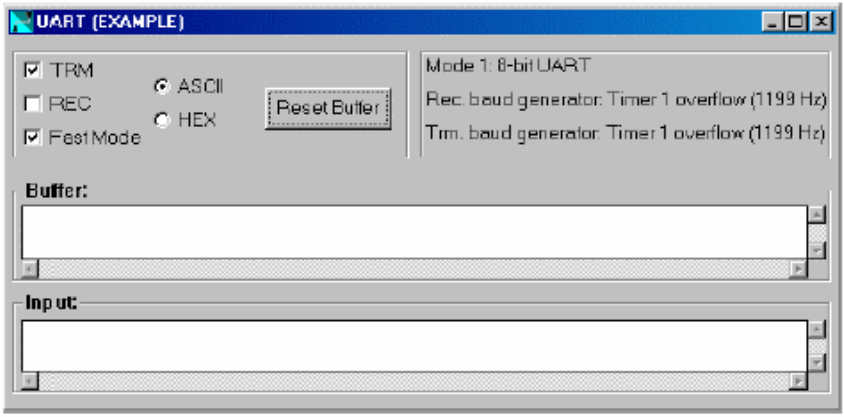


Рисунок 1.6 – Вікно послідовного порту

При відлагодженні програми (рисунок 1.7) часто виникає необхідність призупинити її виконання на певній команді.

Breakpoints – це маркери у вашій програмі, які примушують відлагоджувальник зупинити її виконання у «реальному масштабі часу». Ви можете встановлювати маркери на будь-яких командах програми (рисунок 1.8, 1.9).

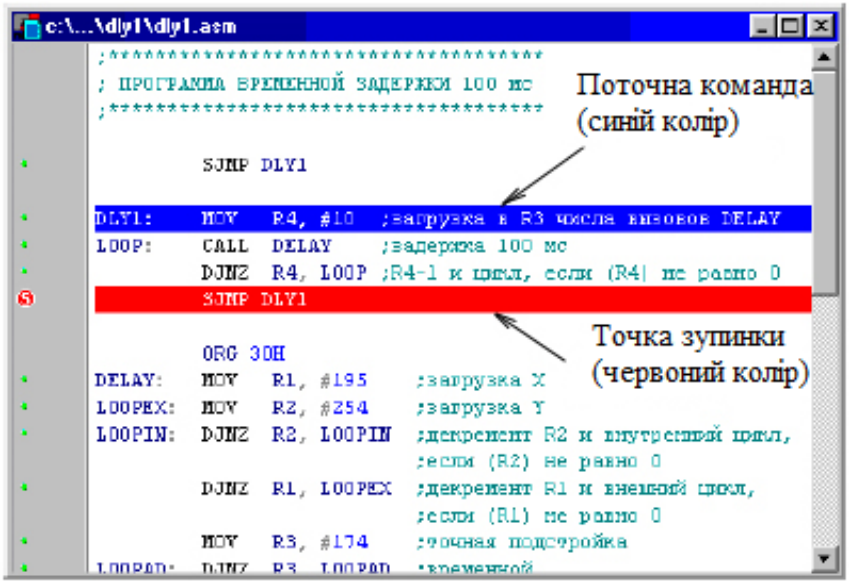
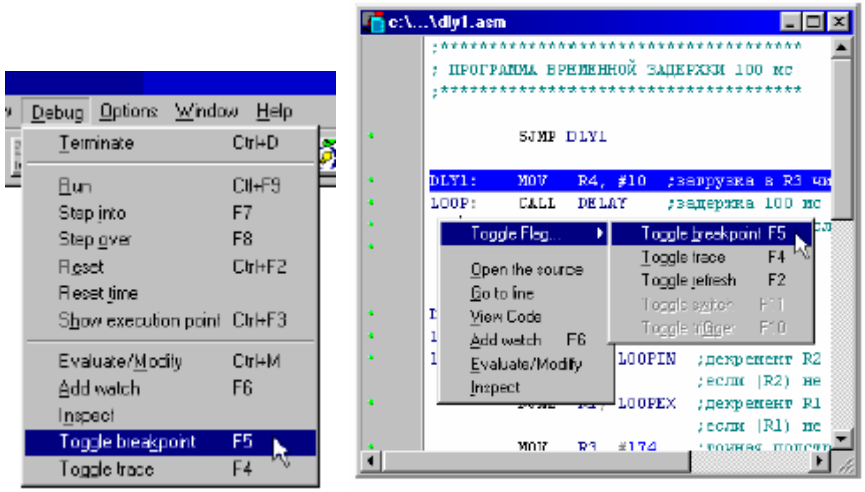


Рисунок 1.7 – Вікно відлагодження



а) б)

а) з головного меню; б) з контекстного меню

Рисунок 1.8 – Встановлення Breakpoint

1.2 Послідовність написання та відлагодження програми.

Ознайомитися з теоретичними відомостями та прикладами з тематики запропонованого завдання.

Скласти блок-схему виконання завдання:

- виділити із умов завдання вихідні дані;
- визначити послідовність виконання дій для вирішення задачі;
- визначити регістри, в яких будуть зберігатися необхідні дані, вигадати для них коротке та інформативне ім'я, а також чіткий описуючий коментар;
- за необхідності визначити перелік дій, які будуть (якщо буде необхідність) повторюватися в ході виконання завдання і вирішити, що саме для цього використати: цикл (декілька циклів) або підпрограму.

Згідно складеної блок-схеми написати текст програми з коментарями до кожної команди (групи команд), які реалізують виконання однієї з умовних частин усієї програми. Бажано уникати коментарів, що дублюють дію окремої команди, тобто дані, які відомі або можуть бути взяті з опису ситеми команд. Коментар повинен роз'яснювати з якою метою використовується команда (група команд) у конкретній програмі.

Наприклад:

```
не бажано (погано) INC R1      ;R1+1→R1
бажано (добре )    INC R1      ;збільшення вказівника
                               ;елементів масиву
```

Процес створення програми бажано розбити на умовні частини:

- складання основного переліку дій для виконання завдання;
- введення вихідних даних;
- опис дій, які виконуються циклічно;
- підпрограми, які будуть викликатися під час виконання основної програми.

Перевірити наявність міток у тексті програми і команд переходу за мітками, а також команд виклику підпрограм і виходу з підпрограм.

Наприклад:

```
org 0h
     mov r0, #30h  ;адреса початкового елемента першого
                   ;масиву
     mov r7, #8h   ;кількість елементів першого масиву
     call count_null; виклик п/п підрахунку
                   ;кількості елементів масиву, що
                   ;дорівнюють нулю
     mov b, a      ;збереження результату виконання
                   підпрограми
     mov r0, #40h  ;адреса початкового елемента другого
                   ;масиву
     mov r7, #8h   ; кількість елементів другого масиву
     call count_null ; виклик п/п підрахунку кількості
                   ;елементів масиву, що дорівнюють нулю
     cjne a, b, m_ne; порівняння результатів підрахунків
     mov 20h, #0h  ;запис у комірку 20h числа 0h, якщо
                   ;кількість елементів, що дорівнюють
                   ;нулю, в обох масивах однакова
      jmp m_end    ;
m_ne:
     jc m1 ;
     mov 20h, #2h  ;запис у комірку 20h числа 2h, якщо
                   ;у другому масиві кількість ;елемен-
                   ;тів, що дорівнюють нулю, більша, ніж
                   ;у першому
     jmp m_end

m1:
     mov 20h, #1h  ;запис у комірку 20h числа 1h, якщо
                   ;у першому масиві кількість елемен-
                   ;тів, що дорівнюють нулю, більша ніж
                   ;у другому

loop: jmp loop
     count_null:   ;п/п підрахунку кількості елементів
                   ;масиву, що дорівнюють нулю
     mov r6, #0h   ;встановлення початкового стану
                   ;лічильника
cnt_m1:            ;початок циклу
     mov a, @r0    ;
     jnz cnt_m2    ;перехід, якщо елемент масиву не
```

```
                ;дорівнює нулю
    inc r6      ;збільшення вмісту лічильника на 1
cnt_m2:         ;
    inc r0      ;збільшення "вказівника" елементів
                ;масиву на 1
    djnz r7, cnt_m1    ; перевірка завершення циклу
    mov a, r6   ;запис результату підрахунку у
                ;акумулятор
    ret
END
```

Перевірити правильність програми, виконавши такі дії:

Набрати у вікні вводу програми одну з умовних частин всієї програми;

Зберегти файл з цим текстом програми
(File / Save as / [ім'я файлу латинськими літерами]. asm);

Запустити проєкт для відлагодження (Project / Build all).

За наявності у тексті програми синтаксичних помилок, у вікні «Message» з'явиться виділене червоним кольором повідомлення «ERROR», яке вказує на рядок, що містить помилку.

Запустити проєкт для перевірки працездатності цієї частини програми (Debug / Start). При запуску програми доступні такі функції емулятора (таблиця 1.1).

Вікно «Main Registers» дозволяє перевіряти стан портів та регістрів та порівнювати його з бажаним.

У вікні «Code» відображуються адреси, коди, мнемоніки програми.

Перевірити стан і режим роботи таймерів можна додатково відкривши відповідні вікна (View / Hardware / Timer 0 та/або Timer 1, Timer 2).

За необхідності не програмного керування окремими лініями паралельних портів їх стан може бути змінений за допомогою миші у вікнах паралельних портів (View / Hardware / Port 0 та/або Port 1, Potr 2, Port 3).

Таблиця 1.1 - Функції емулятора

Граф. зображення	Назва	Функції
	«Animate»	Анімоване виконання програми – при запуску програми у відповідності до порядку виконання команд вони будуть виділені рядком синього кольору
	«Step into»	Покрокове виконання програми – при натисканні на цю кнопку буде виконана тільки одна команда
 	«Run» / «Stop»	Запуск/зупинка програми – після запуску програми ця ж кнопка використовується для зупинки виконання програми, при повторному натисканні програма буде запущена з того місця, на якому була зупинена.
	«Reset»	Повернення у початкове положення – при наступному запуску виконанння програми почнеться з першої команди.

Після того як відлагоджена одна частина програми, слід поступово доповнювати її іншими частинами до повного тексту програми і повної працездатності.

2 ЛАБОРАТОРНА РОБОТА №1

Послідовне пересилання інформації. Регістр зсуву.

Мета: ознайомитися з способом послідовного пересилання інформації, навчитися заносити інформацію в регістр зсуву шляхом змінювання стану його вхідних сигналів.

Короткі теоретичні відомості

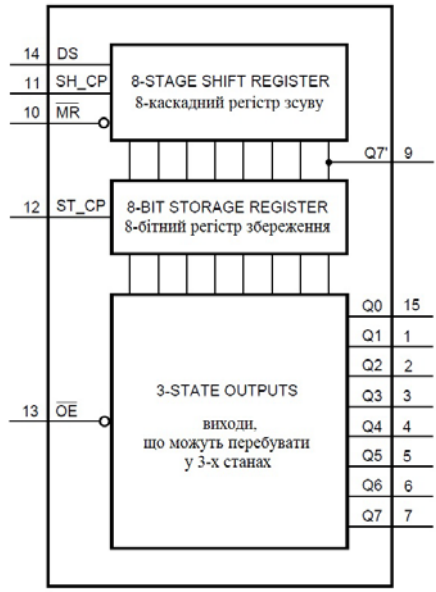
В мікропроцесорних системах передача інформації може здійснюватися паралельно або послідовно.

При паралельній передачі інформація одночасно йде декількома (зазвичай 8, 16, 32, 64) паралельними лініями зв'язку. Характерними особливостями паралельної передачі інформації є: висока швидкість передачі даних; коротка довжина шин; високі витрати на реалізацію з'єднань з огляду на велику кількість ліній зв'язку.

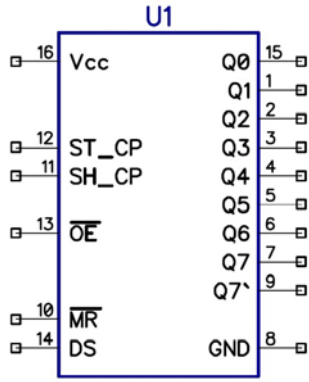
При послідовному пересиланні інформація біт за бітом йде однією лінією зв'язку. Характерними особливостями послідовної передачі інформації є: більш низька швидкість у порівнянні з паралельної передачею такої самої тактової частоти; менші витрати на реалізацію з'єднань з огляду на малу кількість ліній зв'язку; більша їх протяжність, яка може коливатися в межах від декількох десятків метрів до декількох кілометрів; використання паралельно-послідовних та послідовно-паралельних перетворювачів даних, оскільки інформація в передавачеві та приймачеві, як правило, обробляється в паралельній формі.

Одним з різновидів послідовно-паралельних перетворювачів є мікросхема (МС) 74HC595 (рисунок 2.1), яка містить в собі 8-каскадний послідовний регістр зсуву та регістр зберігання, виходи якого можуть перебувати у 3-х станах («0», «1». «від'єднаний»).

Зсувний регістр і регістр зберігання мають окремі входи для тактових імпульсів (SH_CP, ST_CP). Дані зміщуються при додатному переході (від «0» до «1») входу SH_CP. Вміст регістра зсуву передається в регістр зберігання при додатному переході входу ST_CP. Якщо обидві тактові входи з'єднані між собою, регістр зсуву завжди буде на один тактовий імпульс попереду регістра зберігання.



а)



б)

а - структурна схема; б - позначення мікросхеми 74HC595 на принципових схемах.

Рисунок 2.1 – Мікросхема 74HC595

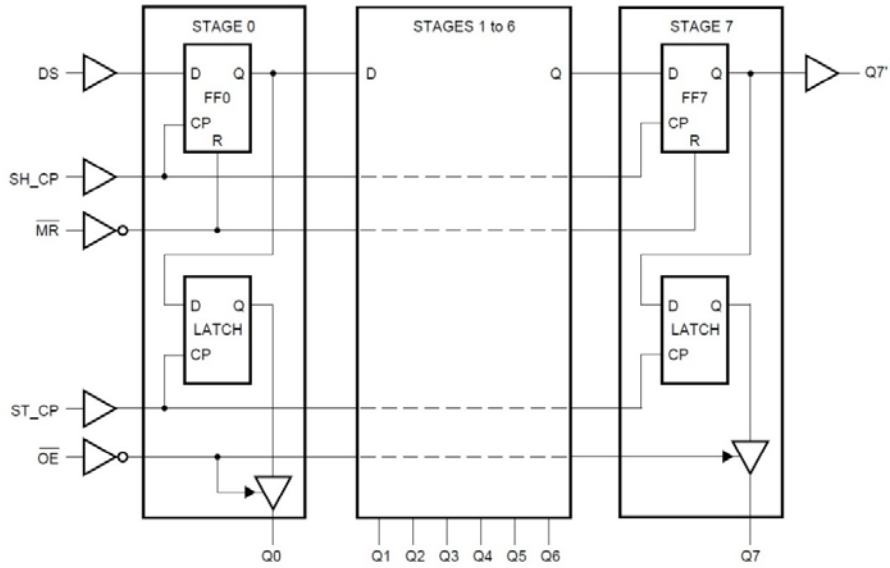


Рисунок 2.2 – Логіка функціонування мікросхеми 74HC595

Регістр зсуву має послідовний вхід (DS) і послідовний стандартний вихід (Q7') для каскадного з'єднання. Також є вхід асинхронного скидання (активний рівень - «низький») для всіх 8-ми каскадів регістра зсуву.

Регістр зберігання має 8 паралельних виходів з драйверами шини, що можуть перебувати у 3-х станах. Дані, які знаходяться в регістрі зберігання з'являються на виході завжди, якщо рівень входу дозволу виходу (OE) «низький».

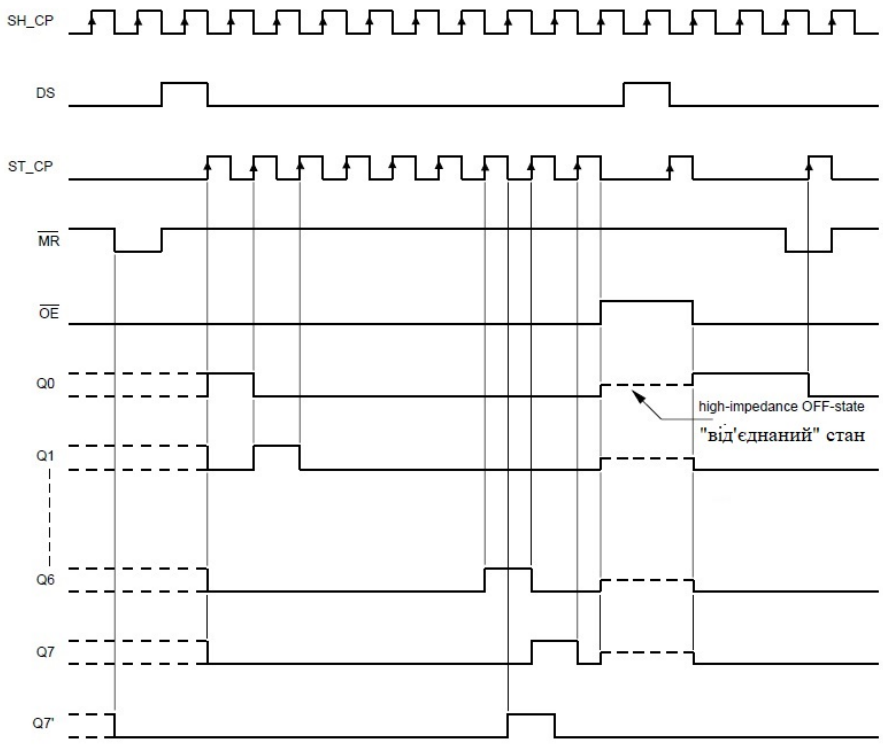


Рисунок 2.3 – Графіки вхідних і вихідних сигналів мікросхеми 74HC595

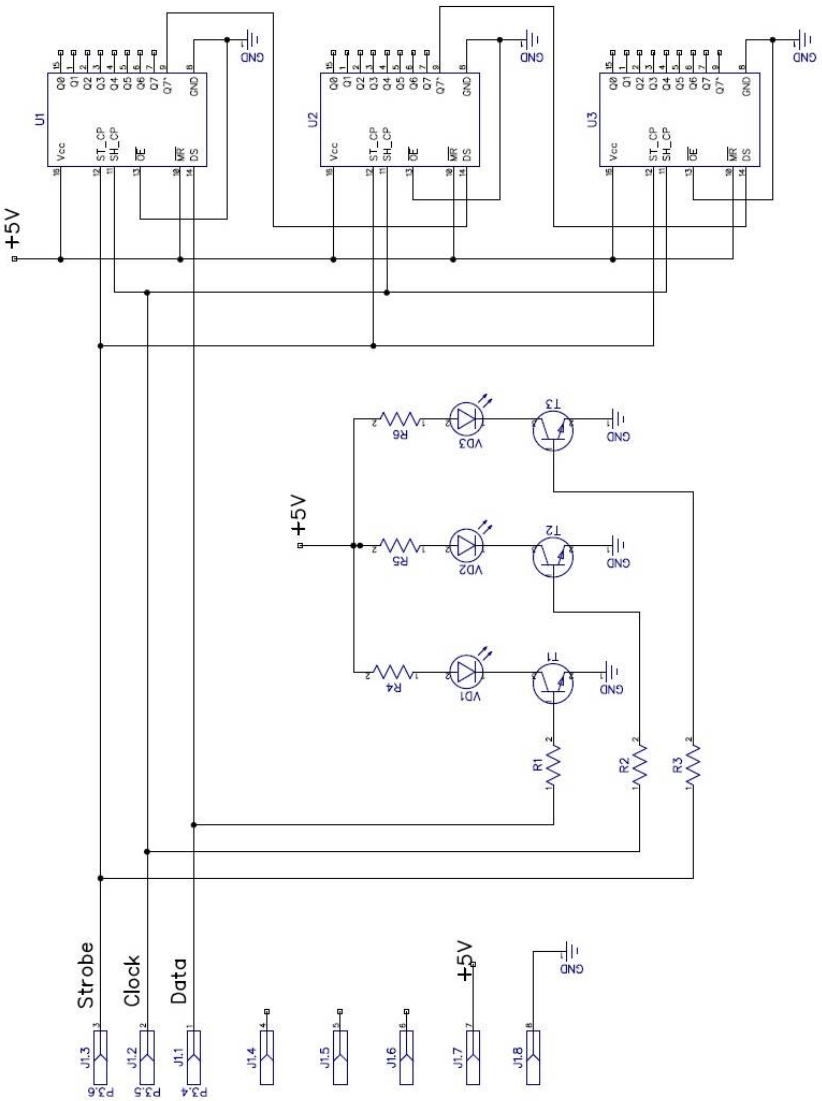


Рисунок 2.4 – Регістри зсуву (фрагмент електричної принципової схеми модуля індикації)

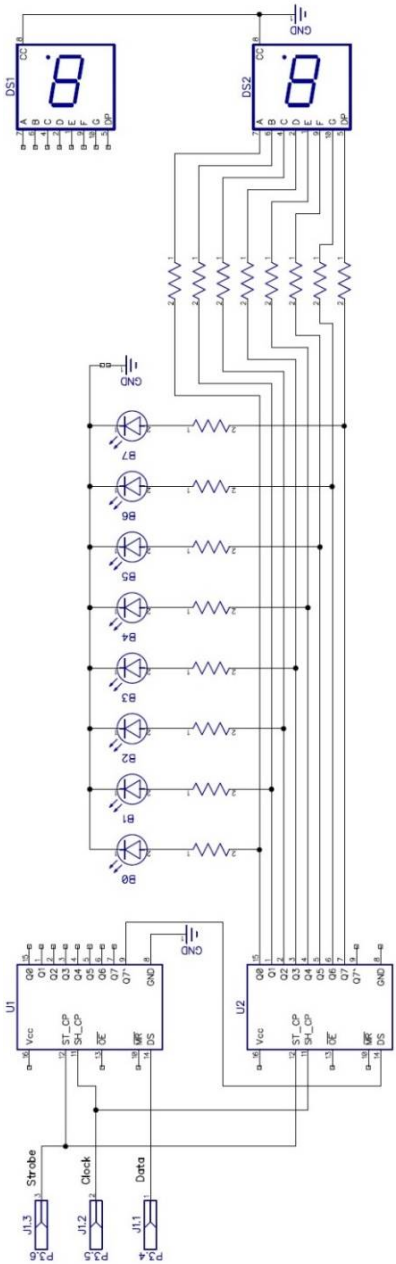


Рисунок 2.5 – Під’єднання семи-сегментних індикаторів до регістрів зсуву (фрагмент електричної принципової схеми модуля індикації)

а) б)

а - умовне зображення; б - зовнішній вигляд.

Рисунок 2.6 – Семисегментний індикатор

На часовій діаграмі (рисунок 1.7), показано як, при змінюванні стану входів SH_CP, DS та ST_CP мікросхеми, можна встановлювати на її виходах Q0 ... Q7 значення 01001111, яке відповідає символу «3» на семи-сегментному індикаторі.

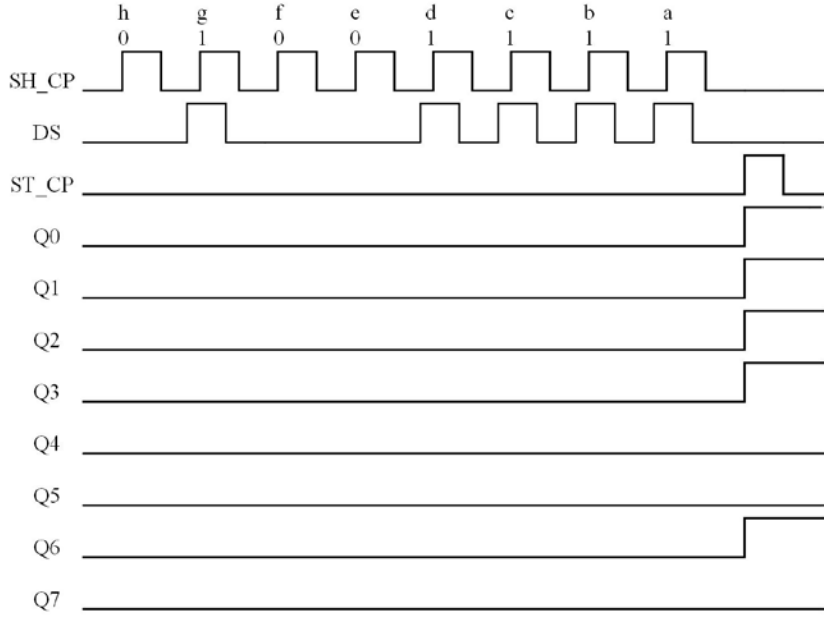


Рисунок 2.7 – Часова діаграма вхідних і вихідних сигналів регістру зсуву

Виконання лабораторної роботи

Перед початком виконання лабораторної роботи переконатися, що пульт, модуль індикації та з'єднувальні кабелі не мають механічних пошкоджень.

Спочатку з'єднати пульт з модулем індикації (рисунок 2.8), потім, за допомогою USB-кабелю – з комп'ютером.

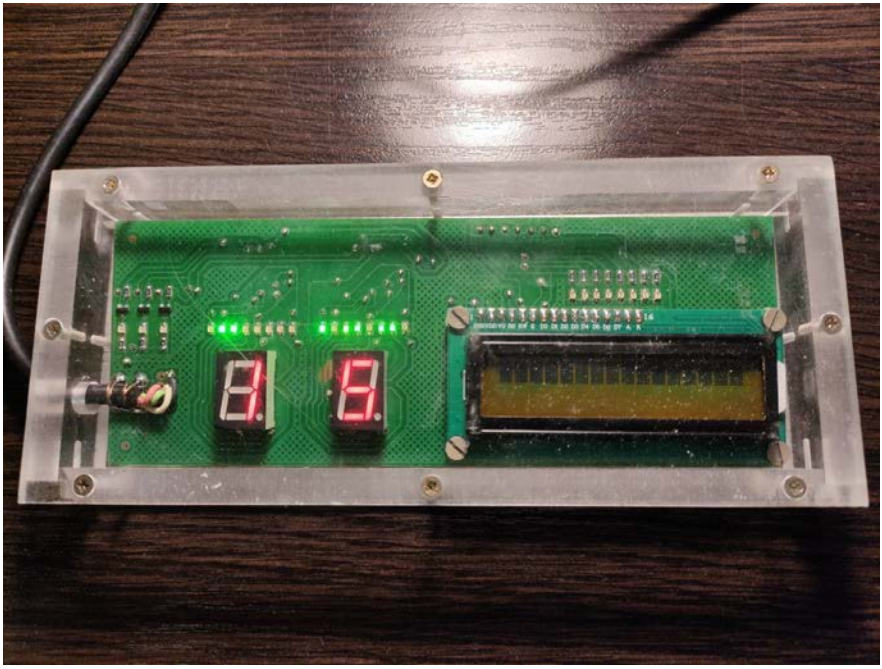


Рисунок 2.8 - Зовнішній вигляд модуля індикації

У середовищі Franclin Software, використовуючи текст програми, що наведений далі, згенерувати HEX-файл.

Положення перемикачів SB1, SB2, SB3 (порт 2) відповідно визначає; логічні рівні сигналів DS, SH_SP, ST_SP.

```
D_S     equ     p3.4    ; дані
SH_SP   equ     p3.5    ; зсув
ST_SP   equ     p3.6    ; фіксація
```

```
        org 0h
                mov     p2, #0FFh
        m1:
                mov c, p2.0
                cpl c
                mov D_S, c
                mov c, p2.1
                cpl c
                mov SH_SP, c
                mov c, p2.2
                cpl c
                mov ST_SP, c
                call wait
                jmp m1
        wait:
                mov r5,#2h
                mov r4,#0ffh
                mov r3,#0ffh
        w1:
                djnz r3,w1
                djnz r4,w1
                djnz r5,w1
                ret
END
```

За допомогою програми WSD (Windows Serial Downloader) завантажити отриманий HEX-файл і запустити програму. Змінюючи положення перемикачів SA1, SA2 та SA3, що відповідають сигналам SH_CP, DS та ST_CP, відтворити послідовність показаних на часовій діаграмі вхідних сигналів регістру зсуву. Переконатися, що на семисегментному індикаторі отримано символ «3».

З метою закріплення навичок, вибрати, на власний розсуд, два символи і вивести їх на індикатори.

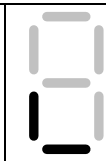
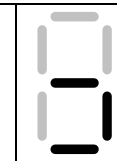
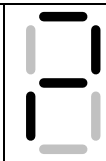
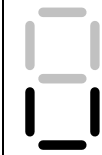
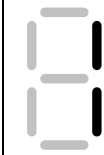
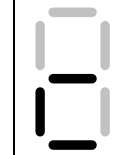
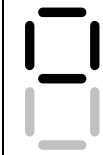
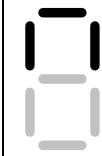
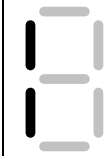
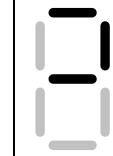
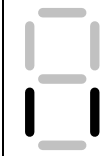
Завдання.

Згідно варіанту вивести на індикатори символи, наведені у таблиці 2.1.

Таблиця 2.1. – Дані для виконання індивідуального завдання

№	Символи		№	Символи		№	Символи	
	P0	P1		P0	P1		P0	P1
1.			2.			3.		
4.			5.			6.		
7.			8.			9.		
10.			11.			12.		
13.			14.			15.		

Продовження табл. 2.1.

16.			17.			18.		
19.			20.			21.		
22.			23.			24.		

Зміст звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист; мету лабораторної роботи; текст програми, часову діаграму, що відповідає завданню; фото результату; висновки.

Контрольні запитання.

1. Як відбувається паралельне передавання інформації?
2. Переваги і недоліки паралельного передавання інформації.
3. Як відбувається послідовне передавання інформації?
4. Переваги і недоліки послідовного передавання інформації.
5. Які пристрої застосовуються для перетворення інформації, що передається послідовно, до паралельного виду?
6. Основні структурні елементи мікросхеми 74HC595.
7. Призначення входів DS, SH_CP, ST_CP, OE.
8. Алгоритм підпрограми **wait** та її призначення в основній програмі.
9. По скільки імпульсів треба подати на входи DS, SH_CP, ST_CP щоб гарантовано отримати на семи-сегментних індикаторах «десять».

3 ЛАБОРАТОРНА РОБОТА №2

Програмна реалізація послідовної передачі інформації.

Мета: ознайомитися з принципом програмної реалізації послідовного способу передачі інформації, навчитися записувати інформацію в регістр зсуву шляхом програмної зміни стану його вхідних сигналів.

Короткі теоретичні відомості

Команда RRC A.

Функція: Зсув бітів акумулятора вправо через прапор переносу.

Опис: Вісім бітів акумулятора та прапор переносу разом змішуються на один біт праворуч. Біт 0 переміщується у прапор переносу; початкове значення прапора переносу переміщується в бітове положення 7. Інші прапори не змінюються.

Приклад: Акумулятор містить значення 0C5H (11000101B), прапор переносу – «нуль». Наступна інструкція,

RRC A

залишає в акумуляторі значення 62 (01100010B) разом із прапором переносу, що містить «один».

Дія: RRC

(An) ← (An + 1) n = 0,1,…6

(A7) ← (C)

(C) ← (A0)

Команда DJNZ.

Function: Decrement and Jump if Not Zero (табл. 3.1)

Функція: Зменшення на 1 та перехід, якщо результат не нульовий.

Опис: DJNZ зменшує перший операнд і переходить до адреси, вказаної другим операндом, якщо отримане значення не дорівнює нулю. Початкове значення 00H зменшується до 0FFH. Жоден прапор не змінюється.

Операндом, який зменшується може бути регістр або прямо адресований байт.

Примітка: Коли ця інструкція використовується для зміни значення вихідного порту, поточне значення буде зчитуватися з

комірки-фіксатора вихідних даних, а не вхідних контактів мікроконтролера.

Приклад: Комірки внутрішньої пам’яті 40H, 50H та 60H містять значення 01H, 70H та 15H відповідно. Наступна

послідовність інструкцій,

DJNZ 40H,LABEL_1

DJNZ 50H,LABEL_2

DJNZ 60H,LABEL_3

здійснює перехід до інструкції з міткою LABEL_2 зі значеннями 00H, 6FH та 15H у трьох комірках оперативної пам'яті. Перший перехід не було виконано, оскільки результат був нульовим.

Ця інструкція забезпечує простий спосіб виконання циклу програми певну кількість разів або реалізації

короткої часової затримки (від 2 до 512 машинних циклів) за допомогою однієї інструкції. Наступна послідовність інструкцій,

MOV R2, # 8

TOGGLE: CPL P1.7

DJNZ R2,TOGGLE

перемикає P1.7 вісім разів, викликаючи появу чотирьох вихідних імпульсів (біт 7 вихідного порту 1). Стале значення триває три машинні цикли; два для DJNZ і один для зміни стану вхідного біту.

Таблиця 3.1 ¬ Мнемоніка команди DJNZ.

Команда	**DJNZ Rn,rel**	**DJNZ direct,rel**
Дія	(PC) ← (PC) + 2 (Rn) ← (Rn) - 1 IF (Rn) > 0 or (Rn) < 0 THEN (PC) ← (PC) + rel	(PC) ← (PC) + 2 (direct) ← (direct) - 1 IF (direct) > 0 or (direct) < 0 THEN (PC) ← (PC) + rel

Команда SETB.

Функція: Встановити біт (табл. 3.2).

Опис: SETB встановлює вказаний біт в одиницю. SETB може працювати з прапором переносу або з будь-яким бітом, що прямо адресується. Ніякі інші прапори не змінюються.

Приклад: прапор переносу скинуто в «0». Вихідний порт 1 містить значення 34H (00110100B). Наступні

інструкції,

SETB C

SETB P1.0

встановлюють прапор переносу в «1» і змінюють вихідні дані в порту 1 на 35H (00110101B).

Таблиця 3.2 ¬ Мнемоніка команди SETB.

Команда	**SETB C**	**SETB bit**
Дія	(C) ← 1	(bit) ← 1

Команда CLR.

Функція: Очистити біт (табл. 3.3).

Опис: CLR очищає вказаний біт (скидає в «0»). Інші прапори не змінюються. CLR може працювати з прапором переносу або з будь-яким бітом, що прямо адресується. Ніякі інші прапори не змінюються.

Приклад: Порт 1 містить значення 5DH (01011101B). Наступна інструкція

CLR P1.2

змінює значення порту 1 на 59H (01011001B).

Таблиця 3.3 ¬ Мнемоніка команди CLR.

Команда	**CLR C**	**CLR bit**
Дія	(C) ← 0	(bit) ← 0

Виконання лабораторної роботи

Перед початком виконання лабораторної роботи переконатися, що пульт, модуль індикації та з'єднувальні кабелі не мають механічних пошкоджень.

Спочатку з'єднати пульт з модулем індикації (рисунок 2.8), потім, за допомогою USB-кабелю – з комп'ютером.

У середовищі Franclin Software, використовуючи текст програми, наведеної нижче, згенерувати HEX-файл.

```
D_C   equ  P3.4
SH_CP equ  P3.5
```

```
ST_CP equ  P3.6
org 0
clr D_C
clr SH_CP
clr ST_CP
mov A, #01001111b
call shift
call wait
setb ST_CP
call wait
clr ST_CP

m1: jmp m1

shift:
  mov r0, #8
s1: rlc a
  mov D_C, c
  call wait
  setb SH_CP
  call wait
  clr SH_CP
  call wait
  djnz r0, s1
  ret
wait:
  mov r5, #08h
w1:
  djnz r3, w1
  djnz r4, w1
  djnz r5, w1
  ret
 end
```

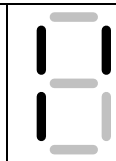
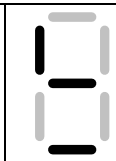
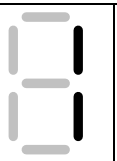
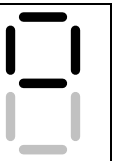
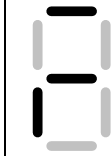
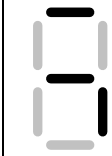
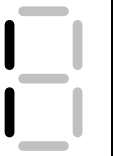
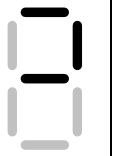
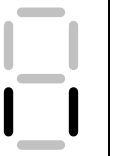
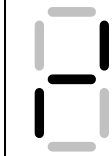
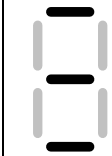
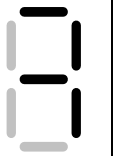
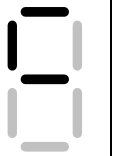
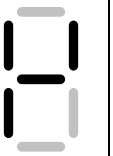
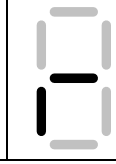
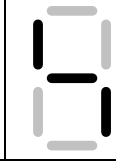
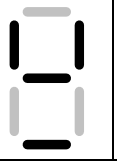
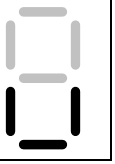
За допомогою програми WSD (Windows Serial Downloader) завантажити отриманий HEX-файл і запустити програму. Переконатися, що на семисегментному індикаторі отримано символ «3».

Завдання 1

Згідно з варіантом вивести на індикатори символи, що наведені у таблиці 2.4.

Таблиця 2.4 Дані для виконання індивідуального завдання

№	Символи		№	Символи		№	Символи	
	P0	P1		P0	P1		P0	P1
1.			11.			21.		
2.			12.			22.		
3.			13.			23.		
4.			14.			24.		
5.			15.			25.		
6.			16.			26.		

7.			17.			27.		
8.			18.			28.		
9.			19.			29.		
10.			20.			30.		

Завдання 2

В залежності від положення перемикача вивести задані символи на семисегментні індикатори (номер перемикача і символи взяти з таблиці 2.5).

Таблиця 2.5 Дані для виконання індивідуального завдання

№ варіанту	X	SAx ON*	SAx OFF*
1	7	10	18
2	5	11	19
3	3	12	20
4	1	13	21
5	6	14	22
6	4	15	23
7	2	16	24
8	0	17	25
9	3	18	26
19	4	19	27

* - числа у даних стовпчиках вказують розміщення заданих символів в таблиці 2.5.

Зміст звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист; мету лабораторної роботи; текст програми, фото результату; висновки.

Контрольні запитання.

1. Команда RRC A. Дія та місце знаходження результату.
2. Команда DJNZ. Дія та місце знаходження результату.
3. Команда SETB. Дія та місце знаходження результату.
4. Команда CLR. Дія та місце знаходження результату.
5. Команди JB та JNB. Функція, опис та використання для організації розгалужених програмних структур.
6. Підпрограма “shift”. Алгоритм та призначення в основній програмі.
7. Команда RRC A. Призначення у підпрограмі “shift”.
8. Команда DJNZ. Призначення у підпрограмі “shift”.
9. Команда CLR. Призначення у підпрограмі “shift”.
10. Команда SETB. Призначення у підпрограмі “shift”.
11. Програмна реалізація часової затримки. Розрахунок тривалості.
12. Скільки разів необхідно викликати підпрограму “shift” для того, щоб вивести на семи-сегментні індикатори «сімнадцять»?

4 ЛАБОРАТОРНА РОБОТА №3

Індикація чисел у шістнадцятковому форматі.

Мета: ознайомитися з принципом програмного перетворення вмісту комірок пам'яті для подальшої індикації на семисегментних індикаторах, навчитися корегувати програму для виконання нею додаткових обчислень.

Короткі теоретичні відомості

Команда CPL A.

Функція. Інверсія акумулятора.

Опис. CPL A інвертує кожен біт Акумулятора (доповнення до 1). Інші прапори не змінюються.

Приклад. Акумулятор містить 5CH (01011100B). Наступна інструкція, CPL A залишає в акумуляторі 0A3H (10100011B).

Дія. CPL ← ¬ (A)

Команда ANL.

Функція. логічне-«І» для змінних розмірністю один байт.

Опис. ANL виконує побітну логічну операцію «І» між вказаними змінними і зберігає результат в змінній призначення. Жоден прапор не змінюються.

Приклад. Якщо акумулятор вміщує 0C3H (1100001lB), а регістр 0 вміщує 55H (01010101B), то наступна інструкція, ANL A, R0 залишає 41H (01000001B) в акумуляторі.

Команда може використовуватися для встановлення в «нуль» окремих бітів.

Наступна інструкція, ANL P1, # 01110011B очищає біти 7, 3 і 2 вихідного порту 1.

Команда SWAP A.

Функція: обмін напівбайтів в межах акумулятора.

Опис. SWAP A міняє старший і молодший напівбайти акумулятора (біти 3 - 0 та біти 7 - 4). Операцію також можна розглядати як 4-бітну інструкцію зсуву. Жоден прапор не змінюється.

Приклад. Акумулятор містить значення 0C5H (11000101B).

Інструкція, SWAP A залишає в Акумуляторі значення 5CH (01011100B).

Операція: SWAP (A3-A0) ↔ (A7-A4)

Команда MOVC A,@A+ <base-reg>.

Функція. Копіювання байту з пам’яті програм.

Опис. Інструкції MOVC завантажують акумулятор байтом коду програми або константою з програмної пам'яті. Адреса отриманого байту - це сума беззнакового 8-бітного вмісту акумулятора та вмісту 16-бітного базового регістру, який може бути або вказівником даних (DPTR), або програмним лічильником (PC).

Жоден прапор не змінюється.

Приклад. Значення між 0 і 3 знаходиться в акумуляторі. Наступні інструкції (команди) помістять в акумулятор одне з чотирьох значень, визначених директивою DB (define byte – визначити байт).

```
REL_PC:
INC A
MOVC A, @A+PC
RET
DB 66H
DB 77H
DB 88H
DB 99H
```

Якщо підпрограма викликається з Акумулятором, рівним 01H, вона повертається з числом 77H в Акумуляторі. Команда INC перед інструкцією MOVC потрібна, щоб "обійти" інструкцію RET над таблицею.

Виконання лабораторної роботи

Перед початком виконання лабораторної роботи переконатися, що пульт, модуль індикації та з’єднувальні кабелі не мають механічних пошкоджень.

Спочатку з’єднати пульт з модулем індикації (рисунок 1.8), потім, за допомогою USB-кабелю – з комп’ютером.

У середовищі Franclin Software, використовуючи текст програми, наведеної нижче, згенерувати HEX-файл.

```
Sh1     equ   20h
Sh2     equ   21h
Sh3     equ   22h

D_S     equ   p3.4
SH_SP   equ   p3.5
ST_SP   equ   p3.6


org 0h

m1:
        mov a, p2
mov p0, a
cpl a
mov b,a

        mov a,b
        anl a,#0fh
        call DECODE
        mov Sh2, A

        mov a,b
        swap a
        anl a,#0fh
        call DECODE
        mov Sh1, A

        mov Sh3,p2
        xrl Sh3,#0FFh

        mov a,Sh3
        call Shift

        mov a,Sh2
        call Shift
```

```
            mov a,Sh1
            call Shift
            call wait
            setb ST_SP
            call wait
            clr ST_SP
            jmp m1
Shift:
            mov r0, #8
S1:
rlc a
            mov D_S,c
            call wait
            setb SH_SP
            call wait
            clr SH_SP
            call wait
            djnz r0,S1
            ret
wait:
  mov r4,#2h
w1:
            ;djnz r2,w1
            djnz r3,w1
            djnz r4,w1
            ret
DECODE:     MOV DPTR, #TAB7SEG
            MOVC A,@A+DPTR
            RET
TAB7SEG:                    ;
DB 00111111b ; '0'
DB 00000110b ; '1'
DB 01011011b ; '2'
DB 01001111b ; '3'
DB 01100110b ; '4'
DB 01101101b ; '5'
DB 01111101b ; '6'
DB 00000111b ; '7'
```

```
DB 01111111b ; '8'
DB 01101111b ; '9'
DB 01110111b ; 'A'
DB 01111100b ; 'b'
DB 00111001b ; 'C'
DB 01011110b ; 'd'
DB 01111001b ; 'E'
DB 01110001b ; 'F'
```

За допомогою програми WSD (Windows Serial Downloader) завантажити отриманий HEX-файл і запустити програму. Переконатися, що двійкове число, встановлене за допомогою перемикачів SA1 – SA8, виводиться на VD1 – VD8 у двійковому вигляді та на семисегментні індикатори у шістнадцятковому . вигляді.

Завдання

Згідно з варіантом вивести на індикатори число у шістнадцятковому форматі, що обчислене за наведеним у таблиці 4.1 виразом.

Таблиця 4.1 Дані для виконання індивідуального завдання

№	Вираз	Дані для перевірки
1	$(P2)*02H+8AH$	0A2H
2	$(P2)/02H-3FH$	0FCH;
3	$(P2)\forall 23H-0D2H$	56H;
4	$(P2)\forall 23H+7AH$	0E3H
5	$(P2)*03H+0A2H$	0C5H
6	$(P2)/02H-0CFH$	0CDH
7	$(P2)\forall 8EH-027H$	4BH
8	$(P2)\forall 81H+37H$	87H;
9	$(P2)*05H+1CH$	2DH
10	$(P2)/02H-7DH$	73H
11	$(P2)\forall 5AH-4BH$	1CH
12	$(P2)\forall 7EH+3CH$	0ADH

Зміст звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист; мету лабораторної роботи; текст програми, фото результату; обчислення, що підтверджують правильність роботи програми; висновки.

Контрольні запитання.

1. Команда **ANL**. Функція, опис та призначення у програмі.
2. Команда **DJNZ**. Функція, опис та призначення у програмі.
3. Команда **SWAP A**. Функція, опис та призначення у програмі.
4. Підпрограма **DECODE**. Місце знаходження вихідної інформації та результату. Призначення в основній програмі.
5. Команда **MOVC A,@A+ <base-reg>**. Функція, опис та призначення у підпрограмі **DECODE**.
6. Який результат буде в акумуляторі після виконання команди **SWAP A**, якщо до цього в ньому знаходилося число 78H?
7. Який результат буде в акумуляторі після виконання команди **CPL A**, якщо до цього в ньому знаходилося число 87H?
8. Чому для чисел 78H та 87H команди **SWAP A** та **CPL A** дають однаковий результат? Для яких ще чисел це можливо?

5 ЛАБОРАТОРНА РОБОТА №4

Обробка аналогових сигналів.

Мета: ознайомитися зі структурою та принципом роботи аналогово-цифрового перетворювача мікроконтролера ADuC841, навчитися приводити вхідний аналоговий сигнал до заданого числового проміжку.

Короткі теоретичні відомості

Блок аналогово-цифрового перетворення являє собою, 8-канальний, 12-бітний однополярний АЦП (рис. 5.1). Основними елементами блоку є: багатоканальний мультиплексор, пристрій вибірки та зберігання, внутрішнє джерело опорної напруги, засоби калібрування і власне АЦП.

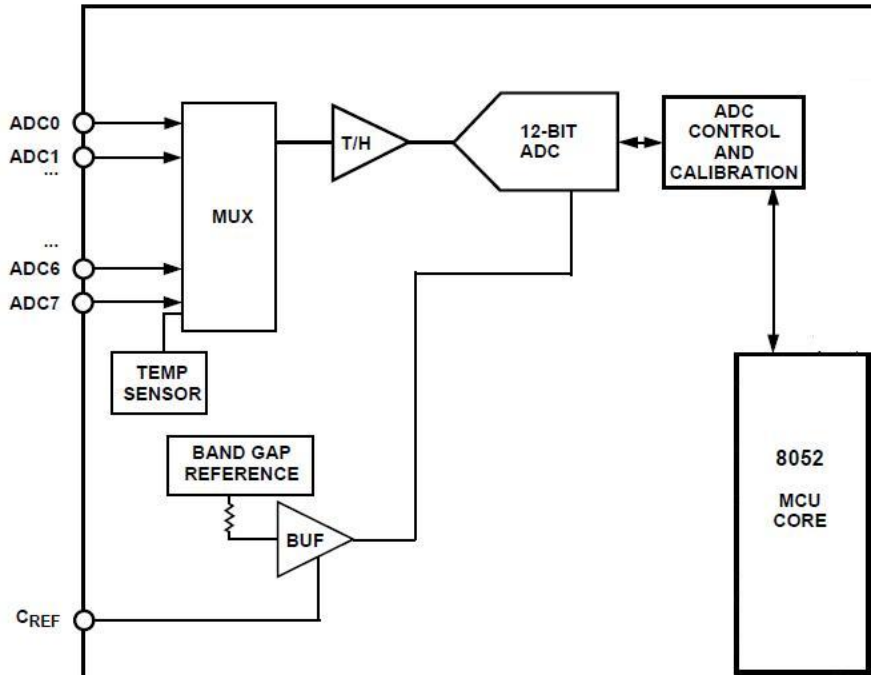


Рисунок 5.1 - Структура блоку аналогово-цифрового перетворення

Компоненти блоку АЦП конфігуруються за допомогою трьох регістрів спеціальних функцій.

Одиночний або безперервний режими перетворення можуть бути ініційовані програмно (шляхом встановлення відповідного біта РСФ) або апаратно (подачею сигналу на відповідний зовнішній вивід контролера). Також для формування імпульсів, що ініціюють аналогово-цифрове перетворення, може використовуватися Таймер 2.

АЦП може бути налаштований в режим DMA (прямий доступ до пам'яті) при якому блок АЦП постійно перетворює і поміщає результати перетворення в область зовнішнього ОЗУ без будь-якої взаємодії з ядром МК.

Передатна характеристика АЦП.

Діапазон вхідного аналогового сигналу для АЦП знаходиться в межах від 0 В до VREF (значення опорної напруги). Результуюче кодування - пряме бінарне з 1 LSB = FS / 4096 або 2.5 В / 4096 = 0,61 мВ при VREF = 2.5 В (LSB - молодший значущий розряд, FS - повний діапазон, VREF - значення напруги опорного джерела). Ідеальна передавальна характеристика АЦП для діапазону від 0 В до VREF показана на рисунку 5.2.

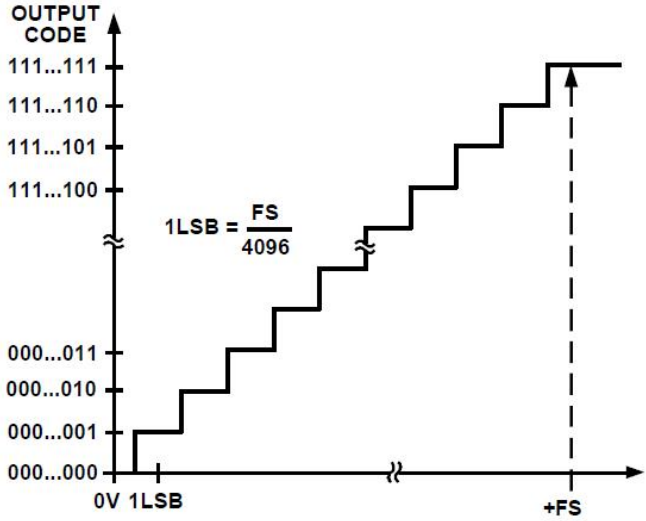


Рисунок 5.2 - Типовий процес перетворення

Після налаштування за допомогою регістрів спеціальних функцій ADCCON 1-3 АЦП перетворює аналоговий вхідний сигнал і записує 12-бітове результуюче слово в РСФ ADCDATAH / L. Старші 4 біта ADCDATAH записуються відповідно до стану бітів вибору каналів для ідентифікації результату обраного каналу.

Формат результуючого слова (рис. 5.3).

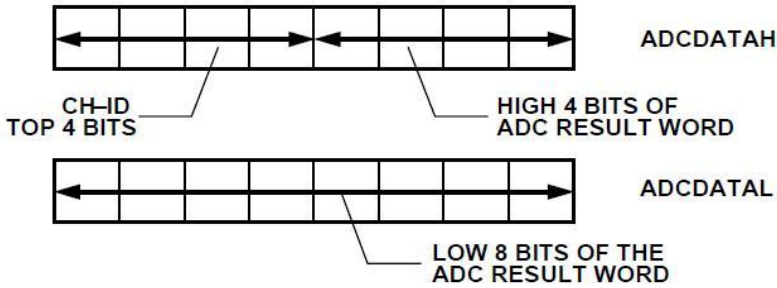


Рисунок 5.3 - Формат результуючого слова

Керуючі регістри.

Регістр ADCCON1 керує перетворенням і часом вимірювання, апаратними режимами перетворення і режимами відключення живлення.

Адреса регістра - 0EFH; початкове значення, що записується при включенні живлення - 40H; бітова адресація - відсутня.

Таблиця 5.1 - Режими регістру ADCCON1.

Біт	Ім’я	Опис
7	MD1	Біт режиму. Визначає активний робочий режим АЦП. Встановлюється користувачем для активації АЦП. Скидається користувачем для деактивації АЦП.
6	EXT_REF	Встановлюється користувачем для вибору зовнішнього джерела опорної напруги (ДОН). Скидається користувачем для використання внутрішнього ДОН.

Продовження таблиці 5.1

<table>
<tr><td>5</td><td>CK1</td><td rowspan="2">Біти поділу тактових імпульсів АЦП. Визначають коефіцієнт ділення для зовнішнього кварцового резонатора, який використовується для генерації тактових імпульсів АЦП.
CK1 CK0 MCLK Divider
0 0 32
0 1 4 (Do not use with a CD setting of 0)
1 0 8
1 1 2</td></tr>
<tr><td>4</td><td>CK0</td></tr>
<tr><td>3</td><td>AQ1</td><td rowspan="2">Біти вибору тривалості вимірювання. Визначають час, передбачений для входу підсилювача «вибірки і зберігання» щоб виміряти вхідний сигнал. Рекомендований час вимірювання становить три або більше тактів АЦП. Кількість тактів зазначено далі:
AQ1 AQ0 No. ADC Clks
0 0 1
0 1 2
1 0 3
1 1 4</td></tr>
<tr><td>2</td><td>AQ0</td></tr>
<tr><td>1</td><td>T2C</td><td>Біт ініціювання перетворення таймером 2. Встановлюється користувачем для того, щоб біт переповнення таймера 2 використовувався в якості входу тригера запуску перетворення АЦП.</td></tr>
<tr><td>0</td><td>EXC</td><td>Біт дозволу зовнішнього сигналу запуску. Встановлюється користувачем для того, щоб використовувати зовнішній вхід P3.5 з активним низьким рівнем (CONVST) в якості входу запуску перетворення. Мінімальна ширина імпульсу > 100 нс.</td></tr>
</table>

Регістр ADCCON2 керує вибором каналу АЦП і режимом перетворення.

Таблиця 5.2 - Режими регістру ADCCON2.

<table>
<tr><th>Біт</th><th>Ім’я</th><th>Опис</th></tr>
<tr><td>7</td><td>ADCI</td><td>Біт переривання АЦП. Встановлюється апаратним забезпеченням після закінчення одиничного циклу перетворення АЦП або після закінчення перетворення блоку DMA (ПДП). Скидається апаратно при переході до процедури обслуговування переривання АЦП. В іншому випадку, ADCI біт скидається програмою користувача.</td></tr>
<tr><td>6</td><td>DMA</td><td>Біт дозволу режиму DMA. Встановлюється користувачем для того, щоб дозволити роботу АЦП в заздалегідь налаштованому режимі прямого доступу до пам'яті.</td></tr>
<tr><td>5</td><td>CCONV</td><td>Біт безперервного перетворення. Встановлюється користувачем для ініціювання роботи АЦП в безперервному режимі перетворення. Наступне перетворення починається автоматично, як тільки закінчено попереднє перетворення.</td></tr>
<tr><td>4</td><td>SCONV</td><td>Біт одиночного перетворення. Встановлюється для ініціювання одиночного циклу перетворення. Біт SCONV автоматично скидається в 0 після завершення одиночного циклу перетворення.</td></tr>
<tr><td>3</td><td>CS3</td><td rowspan="3">Біти вибору каналу. Дозволяють користувачеві запрограмувати вибір каналу АЦП під керуванням програми. Коли перетворення ініціюється, перетвореним каналом є той, на який вказують ці біти вибору каналу.
<table>
<tr><th>CS3</th><th>CS2</th><th>CS1</th><th>CS0</th><th>CH#</th><th></th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td></td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td><td></td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>3</td><td></td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td></td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td></td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td></td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>7</td><td></td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Temp Monitor</td><td>Requires minimum of 1 µs to acquire.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>DAC0</td><td>Only use with internal DAC output buffer on.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>DAC1</td><td>Only use with internal DAC output buffer on.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>AGND</td><td></td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>V_{REF}</td><td></td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>DMA STOP</td><td>Place in XRAM location to finish DMA sequence; refer to the ADC DMA Mode section.</td></tr>
</table>
All other combinations reserved.</td></tr>
<tr><td>2</td><td>CS2</td></tr>
<tr><td>1</td><td>CS1</td></tr>
<tr><td>0</td><td>CS0</td><td></td></tr>
</table>

Адреса регістра - 0D8H; початкове значення, що записується при включенні живлення - 00H; бітова адресація - можлива.

Регістр ADCCON3 керує роботою різноманітних режимів калібрування, а також вказує на стан зайнятості АЦП. Адреса регістра - 0F5H; початкове значення, що записується при включенні живлення - 00H; бітова адресація - відсутня.

Виконання лабораторної роботи

Перед початком виконання лабораторної роботи переконатися, що пульт, модуль індикації та з'єднувальні кабелі не мають механічних пошкоджень.

З початку з'єднати пульт з модулем індикації, потім, за допомогою USB-кабелю – з комп'ютером.

У середовищі Franclin Software, використовуючи текст програми, наведеної нижче, згенерувати HEX-файл.

```
ADCCON1     equ     0EFH
ADCCON2     equ     0D8H
ADCDATAL    equ     0D9H
ADCDATAH    equ     0DAH


TL2         equ     0CCh
TH2         equ     0CDh
RCAP2L      equ     0CAh
RCAP2H      equ     0CBh
TR2         equ     0CAh    ; bit address


Sh1         equ     30h
Sh2         equ     31h
Sh3         equ     32h
ADC_8bit    equ     33h
ADC_norm    equ     34h


D_S         equ     p3.4
SH_SP       equ     p3.5
ST_SP       equ     p3.6
```

```
org 0h
jmp main

org 33h
jmp adc


main:

                setb    IE.6    ;EADC
                mov     ADCCON1, #8Ah

                setb    EA

                mov TH2,        #80h
                mov TL2,        #00h

                mov RCAP2H, #80h
                mov RCAP2L, #00h

                setb    TR2

loop:
                mov a, P2
                cpl a
                anl a, #07h
                mov     ADCCON2, a

                mov a, ADC_8bit
                cpl a
                mov P0, a

                mov a, ADC_8bit
                mov b, #3
                div ab
                mov ADC_norm, a
```

```
                mov a, ADC_norm
                mov b,#10
                div ab
                call decode
                mov Sh1, a

                mov a, ADC_norm
                mov b,#10
                div ab
                mov a,b
                call decode
                mov Sh2, a

                mov a, P2
                cpl a
                mov Sh3, a

                call out2ind
                jmp loop


out2ind:
                mov a,Sh3
                call Shift

                mov a,Sh2
                call Shift

                mov a,Sh1
                call Shift

                call wait
                setb ST_SP
                call wait
                clr ST_SP
```

```
                ret
adc:
                push acc
                push b
                push psw


                mov a, ADCDATAL
                swap a
                anl     a, #0fh
                mov b, a

                mov a, ADCDATAH
                anl a, #0fh
                swap a
                orl a, b
                mov ADC_8bit, a

                pop psw
                pop b
                pop acc

                reti


Shift:
                mov r0, #8
S1:
                rlc a
                mov D_S,c
                call wait
                setb SH_SP
                call wait
                clr SH_SP
                call wait
                djnz r0,S1
                ret
```

```
wait:
                mov r2,#2h
w1:
                djnz r3,w1
                djnz r2,w1
                ret

  DECODE:
                MOV DPTR,#TAB7SEG
                MOVC A,@A+DPTR
                RET

TAB7SEG:                        ;
DB 00111111b ; '0'
DB 00000110b ; '1'
DB 01011011b ; '2'
DB 01001111b ; '3'
DB 01100110b ; '4'
DB 01101101b ; '5'
DB 01111101b ; '6'
DB 00000111b ; '7'
DB 01111111b ; '8'
DB 01101111b ; '9'
DB 01110111b ; 'A'
DB 01111100b ; 'b'
DB 00111001b ; 'C'
DB 01011110b ; 'd'
DB 01111001b ; 'E'
DB 01110001b ; 'F'

          END
```

За допомогою програми WSD (Windows Serial Downloader) завантажити отриманий HEX-файл і запустити програму. Переконатися, що положення одного з резисторів R1-R8, що вибраний за допомогою перемикачів SA1 – SA3, виводиться на VD1 – VD8 у двійковому вигляді та на семисегментні індикатори у десятковому вигляді (діапазон від 0 до 85).

Завдання 1

Змінити початкову програму таким чином, щоб положення вибраного резистора виводилося на індикатори у шістнадцятковому вигляді (діапазон від 0 до FF).

Завдання 2

Згідно з варіантом (табл. 5.3) вивести на індикатори нормалізоване значення у десятковому форматі, що відповідає положенню резистора.

Таблиця 5.3 Дані для виконання індивідуального завдання.

№	Резистор	Значення при «0»	Значення при «10»
1	R5	20	60
2	R3	90	10
3	R1	30	80
4	R2	75	25
5	R4	15	90
6	R6	80	35
7	R8	60	20
8	R7	10	90
9	R3	80	30
10	R1	25	75
11	R2	90	15
12	R4	35	80

Зміст звіту з лабораторної роботи.

Звіт з лабораторної роботи повинен містити титульний лист; мету лабораторної роботи; текст програми, чотири фото результатів (початкове, кінцеве та два проміжних положення резистора); графік залежності значення, що виводиться на індикатор, від положення резистора (чотири точки, що відповідають чотирьом фото.

Контрольні запитання.

1. Скільки зовнішніх каналів АЦП має мікроконтролер ADuC841?
2. Яка розрядність АЦП мікроконтролера ADuC841?
3. Основні елементи блоку АЦП мікроконтролера ADuC841.

4. Кількість регістрів спеціальних функцій, за допомогою яких конфігуруються компоненти блоку АЦП.
5. Яким чином можуть бути ініційовані одиночний або безперервний режими перетворення?
6. Що представляє собою режим DMA?
7. Значення напруги опорного джерела, що знаходиться безпосередньо в мікроконтролері?
8. В яких регістрах спеціальних функцій знаходиться результат перетворення аналогового сигналу?
9. Призначення регістру ADCCON1.
10. Призначення регістру ADCCON2.
11. Призначення регістру ADCCON3.
12. Алгоритм перетворення шістнадцяткового числа в BCD-формат.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мікропроцесорна техніка: навч. посібник / В.В. Ткачов, Г.Грулер, Н. Нойбергер та ін. – Д.: Національний гірничий університет, 2012. – 188 с.

2. Мікропроцесорна техніка: Електронний підручник / В.Я. Жуйков, Т.О. Терещенко, Ю.С. Ямненко, А.В. Заграничний ; відп. ред. О.В. Борисов. – К. : НТУУ «КПІ», 2016. – 440 с.

3. Мікропроцесорна техніка: Навчальний посібник з дисципліни для всіх форм навчання та студентів іноземців напряму підготовки 6.050701 “Електротехніка та електротехнології”/ Укл. В.В.Кирик. – К.: ІВЦ «Видавництво «Політехніка», 2014. – 183с.

4. Електроніка і мікропроцесорна техніка / В.І. Сенько, В.П. Лисенко, О.М. Юрченко, В.Є. Лукін, А.А. Руденський – К. : «Агроосвіта», 2015. – 676 с.

5. Сташин, В.В. Проектирование цифровых устройств на одно кристальных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Молногонцева – М.: Энергоатомиздат, 1990. – 224 с.

6. Хіхловська, І.В. Обчислювальна техніка та мікропроцесори. Підручник. – [2-ге вид.]. / І.В. Хіхловська, О.С. Антонов – Одеса: Одеська національна академія зв’язку ім. О.С. Попова, 2011. – 440 с.

7. Nazarova, O. Software and Hardware Complex for The Study of Electropneumatic Mechatronic Systems / O. Nazarova, V. Osadchyy, S. Shulzhenko, M. Olieinikov // 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), Kremenchuk, Ukraine, 2022, pp. 1-6, doi: 10.1109/MEES58014.2022.10005698.

8. Назарова, О.С. Використання програмно-апаратного комплексу електропневматичних мехатронних систем при роботі зі здобувачами вищої освіти / О.С. Назарова, В.В. Осадчий, М.О. Олєйніков, С.С. Шульженко // Мехатронні системи : інновації та інжиніринг : тези доповідей VI Міжнародної наук.-практ. конф., 24 листопада 2022 р. - Київ : КНУТД, 2022. – С. 35-36.

9. Новацький А. О. Мікропроцесорні та мікроконтролерні системи : підручник. У 2 ч. Ч. 1. Мікропроцесорні системи [Електронний ресурс] / А. О. Новацький. – Електронні текстові дані (1 файл: 16,7 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2020. – 361с.

10. Ельперін, І.В. Автоматизація виробничих процесів [Текст]: підручник / І.В. Ельперін, О.М. Пупена, В.М. Сідлецький, С.М. Швед. – Вид. 2-ге, виправлене. – К.: Вид. Ліра-К, 2015. – 378 с.

11. Osadchyy, V. Adjustable Vibration Exciter Based on Unbalanced Mo-tors / V. Osadchyy, O. Nazarova, T. Hutsol, S. Glowacki, K. Mudryk, A. Bry´s, A. Rud, W. Tulej, M. Sojak // Sensors, 2023. – Vol. 23. – P. 2170. https://doi.org/10.3390/s23042170

12. Невлюдов І. Ш. Виробничі процеси та обладнання об’єктів автоматизації : Підручник для студентів вищих навчальних закладів. – Кривий Ріг: Криворізький коледж НАУ, 2017. – 444 с.

13. Гончаренко Б. М., Осадчий С. І., Віхрова Л. Г. Автоматизація виробничих процесів: навч. посіб. – Кіровоград: Лисенко В.Ф., 2016. – 352 с.

14. Островерхов М.Я. Електротехнічні системи на основі електромагнітних виконавчих пристроїв для керування параметрами технологічних процесів: монографія. – К.: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2017. – 348 с.

Додаток А

Перелік команд мікроконтролера Intel 8051

Таблиця А.1 - Група команд пересилання даних

Назва команди	Мнемокод	Т	Б	Ц	Операція
Пересилання в акумулятор з регістра (n=0..7)	MOV A,Rn	1	1	1	(A)←(Rn)
Пересилання в акумулятор вмісту комірки ВПД	MOV A,dir	3	2	1	(A)←(dir)
Пересилання в акумулятор байта з РПД (i=0,1)	MOV A,@Ri	1	1	1	(A)←((Ri))
Завантаження в акумулятор константи	MOV A,#d	2	2	1	(A)←#d
Пересилання в регістр з акумулятора	MOV Rn,A	1	1	1	(Rn)←(A)
Пересилання в регістр вмісту комірки ВПД	MOV Rn,dir	3	2	2	(Rn)←(dir)
Завантаження в регістр константи	MOV Rn,#d	2	2	1	(Rn)←#d
Пересилання у комірку ВПД вмісту акумулятора	MOV dir,A	3	2	1	(dir)←(A)
Пересилання у комірку ВПД вмісту регістра	MOV dir,Rn	3	2	2	(dir)←(Rn)
Пересилання у комірку ВПД з комірки ВПД	MOV dir,dir	9	3	2	(dir)←(dir)
Пересилка байта з РПД у комірку ВПД	MOV dir,@Ri	3	2	2	(dir)←((Ri))
Пересилання константи у комірку ВПД	MOV dir,#d	7	3	2	(dir)←#d
Пересилання в РПД вмісту акумулятора	MOV @Ri,A	1	1	1	((Ri))←(A)
Пересилання в РПД вмісту комірки ВПД	MOV @Ri, dir	3	2	2	((Ri))←(dir)

Продовження таблиці А.1

Пересилання в РПД константи	MOV @Ri,#d	2	2	1	((Ri))←#d
Завантаження вказівника даних	MOV DPTR,#d16	13	3	2	(DPTR)←#d16
Пересилання байта коду, пов'язаного з DPTR в акумулятор	MOVC A,@A+DPTR	1	1	2	(A)←((A)+(DPTR))
Пересилання байта коду пов'язаного з PC в акумулятор	MOVC A,@A+PC	1	1	2	(PC)←(PC)+1 (A)← ((A)+(PC))
Пересилання байта із ЗПД в акумулятор	MOVX A,@Ri	1	1	2	(A)←((Ri))
Пересилання байта із ЗПД в акумулятор	MOVX A,@DPTR	1	1	2	(A)←((DPTR))
Пересилання з акумулятора в комірку ВПД	MOVX@Ri,A	1	1	2	((Ri))←(A)
Пересилання з акумулятора комірку ЗПД	MOVX @DPTR,A	1	1	2	((DPTR))←(A)
Завантаження комірки ВПД у стек	PUSH dir	3	2	2	(SP)←(SP)+1 ((SP))←(dir)
Вивантаження зі стека в комірку ВПД	POP dir	3	2	2	(dir)←(SP) (SP)←(SP)-1
Обмін акумулятора з регістром	XCH A,Rn	1	1	1	(A)↔(Rn)
Обмін акумулятора з коміркою ВПД	XCH A, dir	3	2	1	(A)↔(dir)
Обмін акумулятора з непрямоадресованою коміркою ВПД	XCH A,@Ri	1	1	1	(A)↔((Ri))
Обмін молодшими тетрадами між непрямоадресованою коміркою ВПД і акумулятором	XCHD A,@Ri	1	1	1	$(A_{0..3})\leftrightarrow((Ri)_{0..3})$

Таблиця А.2 - Команди арифметичних операцій

Назва команди	Мнемокод	Т	Б	Ц	Операція
Додавання акумулятора і регістра (n=0..7)	ADD A,Rn	1	1	1	(A) ← (A)+(Rn)
Додавання акумулятора і комірки ВПД	ADD A, dir	3	2	1	(A) ← (A)+(dir)
Додавання акумулятора і непрямо адресованої комірки ВПД	ADD A,@Ri	1	1	1	(A) ← (A)+((Ri))
Додавання акумулятора і константи	ADD A,#d	2	2	1	(A) ← (A)+#d
Додавання акумулятора і регістра з урахуванням переносу	ADDC A,Rn	1	1	1	(A) ← (A)+(Rn)+(C)
Додавання акумулятора і коміркою ВПД з урахуванням переносу	ADDC A, dir	3	2	1	(A) ← (A)+(dir)+(C)
Додавання акумулятора і непрямо адресованої комірки ВПД з урахуванням переносу	ADDC A,@Ri	1	1	1	(A) ← (A)+((Ri))+(C)
Додавання акумулятора і константи з урахуванням переносу	ADDC A,#d	2	2	1	(A) ← (A)+#d+(C)
Десяткова корекція акумулятора	DA A	1	1	1	Якщо ($A_{0..3}$)>9 або ((AC)=1), то ($A_{0..3}$) ← ($A_{0..3}$)+6, і якщо ($A_{4..7}$)>9 або ((C)=1), то ($A_{4..7}$) ← ($A_{4..7}$)+6

Продовження таблиці А.2

Віднімання від акумулятора регістра і позики	SUBB A,Rn	1	1	1	(A) ← (A)-(C)-(Rn)
Віднімання від акумулятора комірки ВПД і позики	SUBB A, dir	3	2	1	(A) ← (A)-(C)-(dir)
Віднімання від акумулятора непрямо адресованої комірки ВПД і позики	SUBB A,@Ri	1	1	1	(A) ← (A)-(C)-((Ri))
Віднімання від акумулятора константи і позики	SUBB A,#d	2	2	1	(A) ← (A)-(C)-#d
Інкремент акумулятора	INC A	1	1	1	(A) ← (A)+1
Інкремент регістра	INC Rn	1	1	1	(Rn) ← (Rn)+1
Інкремент комірки ВПД	INC dir	3	2	1	(dir) ← (dir)+1
Інкремент непрямо-адресованої комірки ВПД	INC @Ri	1	1	1	(Ri) ← (Ri)+1
Інкремент покажчика даних	INC DPTR	1	1	2	(DPTR) ← (DPTR)+1
Декремент акумулятора	DEC A	1	1	1	(A) ← (A)-1
Декремент регістра	DEC Rn	1	1	1	(Rn) ← (Rn)-1
Декремент комірки ВПД	DEC dir	3	2	1	(dir) ← (dir)-1
Декремент непрямо-адресованої комірки ВПД	DEC @Ri	1	1	1	(Ri) ← (Ri)-1

Продовження таблиці А.2

Множення акумулятора на регістр В	MUL AB	1	1	4	(B)(A) ← (A)*(B)
Ділення акумулятора на регістр В	DIV AB	1	1	4	(A).(B) ← (A)/(B)

Таблиця А.3 - Команди логічних операцій

Назва команди	Мнемокод	Т	Б	Ц	Операція
Логічне І акумулятора і регістра	ANL A,Rn	1	1	1	(A)←(A) ∧ (Rn)
Логічне І акумулятора і комірки ВПД	ANL A, dir	3	2	1	(A)←(A) ∧ (dir)
Логічне І акумулятора і непрямо адресованої комірки ВПД	ANL A,@Ri	1	1	1	(A)←(A) ∧ ((Ri))
Логічне І акумулятора і константи	ANL A,#d	2	2	1	(A)←(A) ∧ #d
Логічне І комірки ВПД і акумулятора	ANL dir,A	3	2	1	(dir)←(dir) ∧ (A)
Логічне І комірки ВПД і константи	ANL dir,#d	7	3	2	(dir)←(dir) ∧ #d
Логічне АБО акумулятора і регістра	ORL A,Rn	1	1	1	(A)←(A) ∨ (Rn)
Логічне АБО акумулятора и комірки ВПД	ORL A, dir	3	2	1	(A)←(A) ∨ (dir)
Логічне АБО акумулятора і непрямоадресованої комірки ВПД	ORL A,@Ri	1	1	1	(A)←(A) ∨ ((Ri))

Продовження таблиці А.3

Логічне АБО акумулятора і константи	ORL A,#d	2	2	1	(A)←(A) $\vee$ #d
Логічне АБО комірки ВПД і акумулятора	ORL dir,A	3	2	1	(dir)←(dir) $\vee$ (A)
Логічне АБО комірки ВПД і константи	ORL dir,#d	7	3	2	(dir)←(dir) $\vee$ #d
Що виключає АБО акумулятора і регістра	XRL A,Rn	1	1	1	(A)←(A) $\forall$ (Rn)
Що виключає АБО акумулятора і комірки ВПД	XRL A, dir	3	2	1	(A)←(A) $\forall$ (dir)
Що виключає АБО акумулятора і непрямоадресованої комірки ВПД	XRL A,@Ri	1	1	1	(A)←(A) $\forall$ ((Ri))
Що виключає АБО акумулятора і константи	XRL A,#d	2	2	1	(A)←(A) $\forall$ #d
Що виключає АБО комірки ВПД і акумулятора	XRL dir,A	3	2	1	(dir)←(dir) $\forall$ (A)
Що виключає АБО комірки ВПД і константи	XRL dir,#d	7	3	2	(dir)←(dir) $\forall$ #d
Очищення акумулятора	CLR A	1	1	1	(A)←0
Інверсія акумулятора	CPL A	1	1	1	$(A) \leftarrow (\overline{A})$
Зрушення акумулятора вліво	RL A	1	1	1	$(A_{n+1}) \leftarrow (A_n)$, n=0..6, $(A_0) \leftarrow (A_7)$

Продовження таблиці А.3

Зрушення акумулятора вліво через прапор переносу	RLC A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$
Зрушення акумулятора вправо	RR A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_7)\leftarrow(A_0)$
Зрушення акумулятора вправо через прапор переносу	RRC A	1	1	1	$(A_{n+1})\leftarrow(A_n)$, n=0..6, $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$
Обмін місцями тетрад в Акумуляторі	SWAP A	1	1	1	$(A_{0\text{-}3})\leftarrow(A_{4\text{-}7})$

Таблиця А.4 - Команди роботи з бітами

Назва команди	Мнемокод	Т	Б	Ц	Операція
Очищення переносу	CLR C	1	1	1	$(C)\leftarrow 0$
Очищення біта	CLR bit	4	2	1	$(bit)\leftarrow 0$
Установка переносу	SETB C	1	1	1	$(C)\leftarrow 1$
Установка біта	SETB bit	4	2	1	$(bit)\leftarrow 1$
Інверсія переносу	CPL C	1	1	1	$(C) \leftarrow (\overline{C})$
Інверсія біта	CPL bit	4	2	1	$(bit) \leftarrow (\overline{bit})$
Логічне І біта і прапору переносу	ANL C,bit	4	2	2	$(C)\leftarrow(C) \wedge (bit)$
Логічне І інверсії біта і переносу	ANL C,/bit	4	2	2	$(C) \leftarrow (C) \wedge (\overline{bit})$

Продовження таблиці А.4

Логічне АБО біта і прапора переносу	ORL C,bit	4	2	2	(C)←(C) ∨ (bit)
Логічне АБО інверсії біта і прапора переносу	ORL C,/bit	4	2	2	(C) ← (C) ∨ ($\overline{\text{bit}}$)
Пересилання біта в прапор переносу	MOV C,bit	4	2	1	(C)←(bit)
Пересилання прапору переносу в біт	MOV bit,C	4	2	2	(bit)←(C)

Таблиця А.5 - Команди передачі керування

Назва команди	Мнемокод	Т	Б	Ц	Операція
Довгий перехід	LJMP	12	3	2	(PC)← dir 16
Абсолютний перехід всередині сторінки у 2 Кбайта	AJMP dir11	6	2	2	(PC)←(PC)+2 ($PC_{0..10}$)← dir 11
Короткий відносний перехід всередині сторінки у 256 байт	SJMP rel	5	2	2	(PC)←(PC)+2 (PC)← (PC)+rel
Непрямий відносний перехід	JMP @A+DPTR	1	1	2	(PC)←(A)+(DPTR)
Перехід, якщо акумулятор дорівнює нулю	JZ rel	5	2	2	(PC)←(PC)+2, якщо (A)=0, то (PC)← (PC)+rel

Продовження таблиці А.5

Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	5	2	2	(PC)←(PC)+2, якщо (A)≠0, то (PC)← (PC)+rel
Перехід, якщо прапор переносу дорівнює одиниці	JC rel	5	2	2	(PC)←(PC)+2, якщо (C)=1, то (PC)← (PC)+rel
Перехід, якщо прапор переносу дорівнює нулю	JNC rel	5	2	2	(PC)←(PC)+2, якщо (C)=0, то (PC)← (PC)+rel
Перехід, якщо біт дорівнює одиниці	JB bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=1, то (PC)← (PC)+rel
Перехід, якщо біт дорівнює нулю	JNB bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=0, то (PC)← (PC)+rel
Перехід, якщо біт установлено, з наступним скиданням біта	JBC bit,rel	11	3	2	(PC)←(PC)+3, якщо (bit)=1, то (bit)←0 і (PC)← (PC)+rel
Декремент регістра і перехід, якщо він не дорівнює нулю	DJNZ Rn,rel	5	2	2	(PC)←(PC)+2, (Rn)←(Rn)-1, якщо (Rn)≠0, то (PC)← (PC)+rel
Декремент комірки ВПД і перехід, якщо її вміст не дорівнює нулю	DJNZ dir,rel	8	3	2	(PC)←(PC)+2, (dir)←(dir)-1, якщо (dir)≠0, то (PC)← (PC)+rel

Продовження таблиці А.5

Порівняння акумулятора з коміркою ВПД і перехід, якщо вони не дорівнюють одне одному	CJNE A, dir,rel	8	3	2	(PC)←(PC)+3, якщо (A)≠(dir),то (PC)← (PC)+rel, якщо (A)<(dir), то (C)←1, інакше (C)←0
Порівняння акумулятора з константою і перехід, якщо вони не дорівнюють одне одному	CJNE A,#d,rel	10	3	2	(PC)←(PC)+3, якщо (A)≠#d, то (PC)← (PC)+rel, якщо (A)<#d, то (C)←1, інакше (C)←0
Порівняння регістра з константою і перехід, якщо вони не дорівнюють одне одному	CJNE Rn,#d,rel	10	3	2	(PC)←(PC)+3, якщо (Rn)≠#d, то (PC)← (PC)+rel, якщо (Rn)<#d, то (C)←1, інакше (C)←0
Порівняння непрямоадресованої комірки ВПД з константою і перехід, якщо вони не дорівнюють одне одному	CJNE @Ri,#d,rel	10	3	2	(PC)←(PC)+3, якщо ((Ri))≠#d, то (PC)← (PC)+rel, якщо ((Ri))<#d, то (C)←1, інакше (C)←0
Довгий виклик підпрограми	LCALL dir16	12	3	2	(PC)← (PC)+3, (SP)←(SP)+1, ((SP))← ($PC_{0..7}$), (SP)← (SP)+1, ((SP))← ($PC_{8..15}$), (PC)← dir 16

Продовження таблиці А.5

Абсолютний виклик підпрограми в межах сторінки в 2 Кбайта	ACALL dir11	6	2	2	(PC)←(PC)+2, (SP)← (SP)+1, ((SP))← ($PC_{0..7}$), (SP)← (SP)+1, ((SP))← ($PC_{8..15}$), (PC0..10)← dir 11
Повернення з підпрограми	RET	1	1	2	(PC8..15)← ((SP)), (SP)←(SP)-1, (PC0..7)← ((SP)), (SP)←(SP)-1
Повернення з підпрограми оброблення переривання	RETI	1	1	2	($PC_{8..15}$)←((SP)), (SP)←(SP)-1, ($PC_{0..7}$)← ((SP)), (SP)←(SP)-1
Порожня команда	NOP	1	1	1	(PC)← (PC)+1

Т – тип команди;
Б – формат у байтах;
Ц – кількість машинних циклів, необхідних для оброблення команди.

Додаток Б

Зразок оформлення титульної сторінки

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

Кафедра ЕПА

Лабораторна робота № 1
з дисципліни
«Мікропроцесорні системи»

Виконав:
студ. гр. Е-333 Іваненко І.І.

Перевірив:
доцент Петренко П.П.

2023