НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Лабораторна робота № 5 з курсу ”Проектування мікропроцесорних систем керування технологічними процесами”

Програмування мікроконтролера i8051 з використанням програмної моделі EdSim51. Керування двигуном постійного струму.

Тернопіль 2019

Методичні вказівки до лабораторної роботи № 5 "Програмування мікроконтролера i8051 з використанням програмної моделі EdSim51. Керування двигуном постійного струму" з курсу "Проектування мікропроцесорних систем керування технологічними процесами". Автор: Медвідь В.Р., Тернопіль: ТНТУ, 2019- 7 с.

Для студентів напряму підготовки: 151 "Автоматизація та комп'ютерно-інтегровані технології"

Автор: Медвідь В.Р.

Лабораторна робота присвячена розгляду керування двигунами постійного струму від широкорозповсюджених однокристальних мікро-ЕОМ сімейства MCS-51. Методичні вказівки орієнтовані на студентів, котрі навчаються за напрямом підготовки "Автоматизація та комп'ютерно-інтегровані технології" і можуть бути корисні особам, що вивчають споріднені дисципліни.

Лабораторна робота №5

Програмування мікроконтролера I8051 з використанням програмної моделі EdSim51. Керування двигуном постійного струму.

1. Теоретичні відомості

Керування двигуном постійного струму

Розглянемо роботу мікроконтролер з двигуном постійного струму.

Двигун обертається в напрямку за годинниковою стрілкою, напрямок і кількість обертів відображається на семисегментному дисплеї Disp0 – Disp3 (рис.1). Дисплей показує тільки до дев'яти оборотів, а потім перезавантажується.

Датчик двигуна підключений до P3.5, який є зовнішнім джерелом тактового сигналу для таймера 1, і яким таймер 1 вводиться в режим лічильника.

Збільшення таймера на одиницю щоразу приводить до руху двигуна.
Значення молодшого байту таймера 1 переміщується в A і це значення разом з покажчиком даних (DPH і DPL) використовуються для отримання коду семи сегментного індикатора з пам'яті програм.

Далі код надсилається через P1 для встановлення відповідного числа на дисплеї.

Обертання двигуну може бути змінене на рух проти годинникової стрілки, натискаючи кнопку SW0, під’єднану до лінії P2.0.

Напрямок обертання двигуна зберігається в F0 («1» для руху за годинниковою стрілкою, «0» - для руху проти годинникової стрілки).

Значення F0 виводиться на дисплей в десятковому коді.

Крапка дисплею під'єднана до виводу P1.7. Вона вказує на рух двигуна і його напрямок - якщо десяткова крапка горить, двигун обертається проти годинникової стрілки, Якщо не горить - двигун обертається за годинниковою стрілкою.

Значення в F0 порівнюється з значенням SW0. Якщо вони рівні, то напрямок обертання двигуна не повинен змінитися. Якщо вони не рівні, то це означає, що користувач натиснув SW0 і напрям руху двигуна має бути змінений.

Якщо натискання кнопки відбувається, то новий напрямок двигуна зберігається в F0.

2. Завдання

1. Дослідити програму обертання двигуна.
2. Виконати програму обертання двигуна постійного струму в прямому та реверсному Напрямках.

Програма керування двигуном постійного струму

```
    MOV TMOD, #50H              ; встановити таймер 1 в режим підрахунку подій
    SETB TR1                    ; старт таймеру 1
    MOV DPL, #LOW(LEDcodes)     ; розмістити молодший байт початкової адреси
                                ; 7-сегментного коду в DPL
    MOV DPH, #HIGH(LEDcodes)    ; розмістити старший байт в DPH
    CLR P3.4                    ;
    CLR P3.3                    ; ввімкнути дисплей 0

again:
    CALL setDirection           ; встановити напрям руху двигуна
    MOV A, TL1                  ; завантажити молодший байт таймеру 1 в A
    CJNE A, #10, skip           ; якщо число обертів не 10, виконати
                                ; інструкцію skip
    CALL clearTimer             ; якщо число обертів дорівнює 10, скидання таймера 1

skip:
```

```
    MOVC A, @A+DPTR          ; отримати 7-сегментний код з кодової таблиці —
                             ; індекс в таблиці значень, що адресується A і DPTR
                             ; (Приклад: покажчик даних вказує на старті
                             ; наступне - якщо є два оберти, то A буде містити
                             ; число два, і тому другий код в таблиці буде
                             ; скопійований в A)
    MOV C, F0                ; перемістити значення напрямку обертання
                             ; двигуна в флаг перенесення C
    MOV ACC.7, C             ; а звідти в ACC.7 (це завантажить десяткову точку
                             ; дисплею 0 і вкаже напрямок обертання двигуна)
    MOV P1, A                ; і завантажить 7-сегментний код числа обертів і
                             ; напрямок обертання двигуна
                             ; Індикатор для відображення 0
    JMP again                ; перехід на мітку again

setDirection:
    PUSH ACC                 ; завантаження значення A в стек
    PUSH 20H                 ; зберегти адресу 20H (першого біті адреси
                             ;  розташування в оперативній пам'яті) в стек
    CLR A                    ; очистити A
    MOV 20H, #0              ; очистити адресу 20H
    MOV C, P2.0              ; завантажити значення вимикача SW0 в флаг C
    MOV ACC.0, C             ; і скопіювати його в ACC.0
    MOV C, F0                ; завантажити напрям руху двигуна в C
    MOV 0, C                 ; і перейти до місцезнаходження 20H (що має
                             ; адресу біта 0)
    CJNE A, 20H, changeDir   ; порівняти SW0 з F0
                             ; якщо вони не рівні, напрям руху двигуна
                             ; має бути змінений
    JMP finish               ; якщо вони однакові, напрямок двигуна
                             ; не має бути змінений
changeDir:
    CLR P3.0                 ;
    CLR P3.1                 ; зупинка двигуна
    CALL clearTimer          ; скинути таймер 1
    MOV C, P2.0              ; збереження значення SW0 в регістр ознак C
    MOV F0, C                ; і тоді копіювати в F0 - це новий напрямок двигуна
    MOV P3.0, C              ; завантаження значення SW0 для біту керування
                             ; двигуном 1
    CPL C                    ; Інвертування біту перенесення C
    MOV P3.1, C              ; і перемістити його в біт управління двигуном 0
                             ; він буде мати протилежне значення до величини
                             ; біту 1 для управління і двигун почне обертатися в
                             ; новому напрямку)
finish:
    POP 20H                  ; повернення дійсної адреси 20H з стеку
    POP ACC                  ; повернення значення A із стеку
    RET                      ; повернення з підпрограми

clearTimer:
    CLR A                    ; обнулення A
    CLR TR1                  ; зупинка таймера1
    MOV TL1, #0              ; скидання молодшого байту таймера 1 в нуль
```

```
        SETB TR1                    ; старт таймера 1
        RET                         ; повернення з підпрограми

LEDcodes:                           ; ця мітка вказує на початкову адресу кодової таблиці
                                    ; 7-сегментного індикатора, яка зберігається
                                    ; в пам'яті програми, використовуючи команду DB
                                    ; яка подана нижче
```

DB 11000000B, 11111001B, 10100100B, 10110000B, 10011001B, 10010010B, 10000010B, 11111000B, 10000000B, 10010000B.

Схема підключення двигуна постійного струму показана на рис. 1.

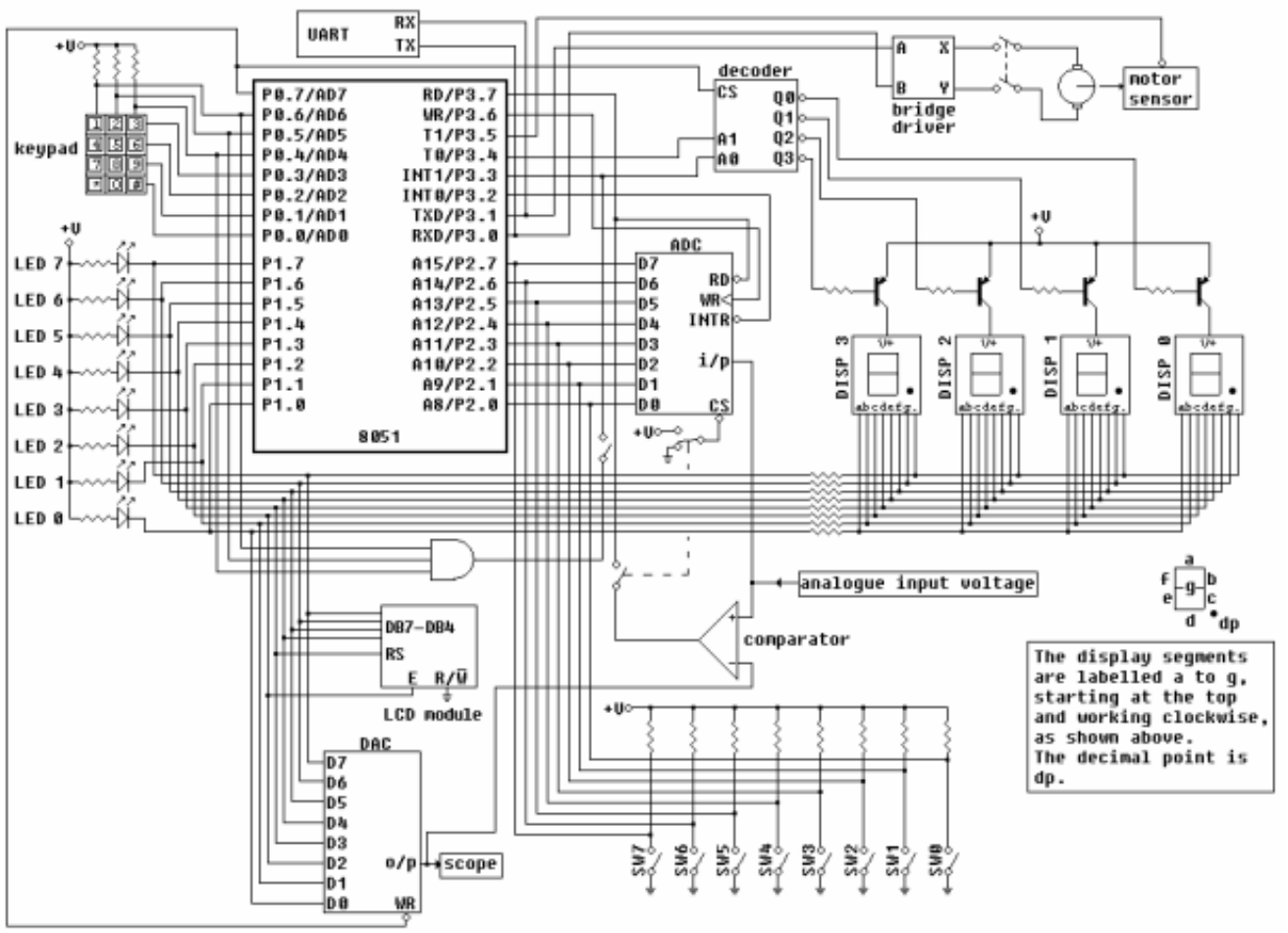

Рис. 1 Схема підключення семисегментного дисплею і двигуна постійного струму

3. Послідовність виконання роботи

3.1. Вивчити команди відповідно до завдання. Вивчення кожної команди проводити наступним чином:

3.1.1. Відкрити інтерфейс емулятора, двічі клацнувши клавішею миші на архівованому файлі «EdSim51.jar". Відкриється інтерфейс програмного емулятора, зображений на рис. 2.

Середнє поле емулятора, що називається **"Панель коду Асемблера"**, в верхній частині містить кнопки **"*Reset*", "*Assm*", "*Run*", "*Load*", "*Save*", "*Copy*", "*Paste*"**.

Панель коду використовується для:

- набору команд програми з клавіатури. Для цього курсор встановлюється в верхній частині панелі і вводиться програма по одній команді в рядку (при потребі, з міткою та коментарем)(див. рис. 2);

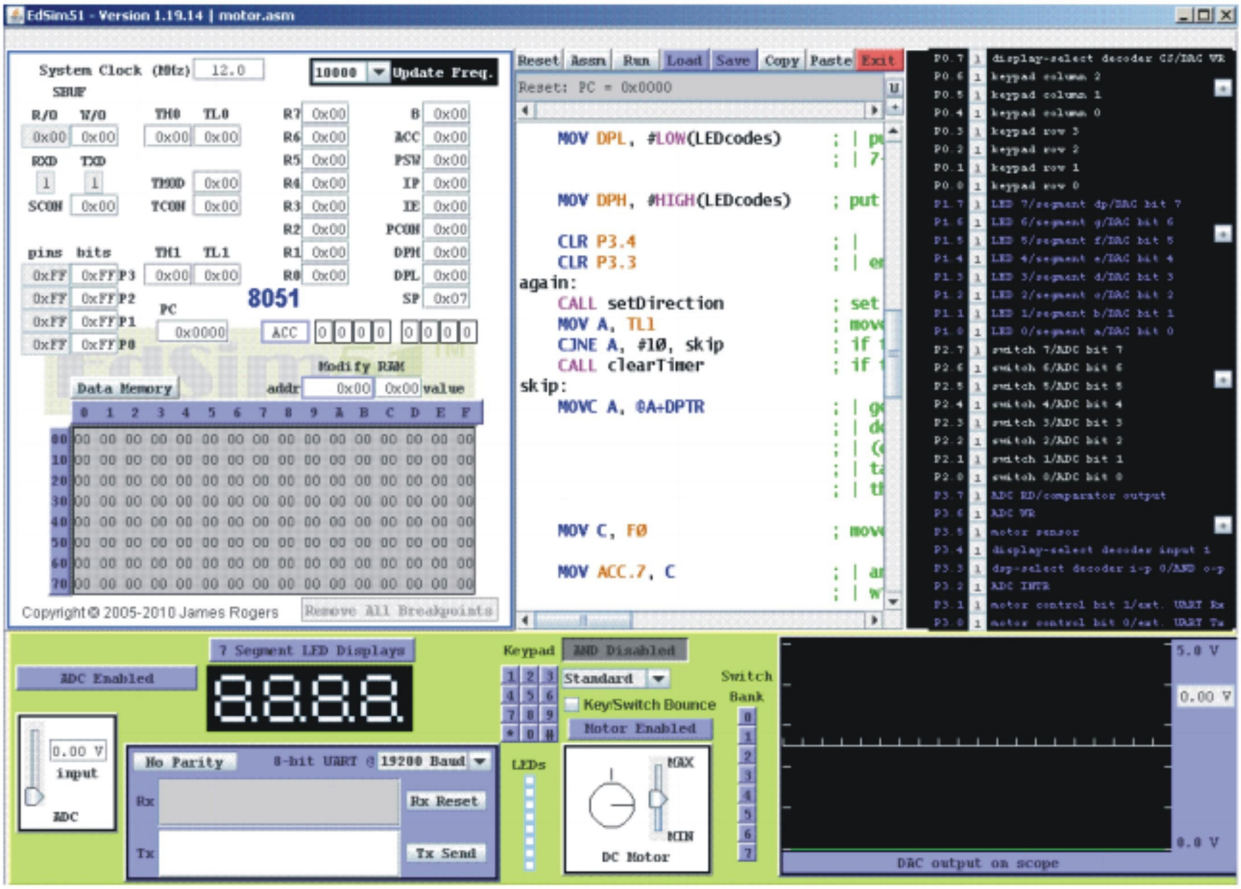

Рис. 2 Інтерфейс програмного емулятора

- **завантаження** вже існуючої програми. Для цього необхідно на панелі вгорі натиснути кнопку **"*Load*"** і вказати шлях до потрібного файлу;
- **запису** набраного файлу. Для цього потрібно натиснути кнопку **"*Save*"** і вказати шлях для збереження файлу.

3.1.2. Перед виконанням програми необхідно натиснути кнопку **"*Assm*"** панелі для асемблювання програми. Після цього, якщо команда записана невірно, в рядку під верхнім рядом кнопок панелі (на рис.1 виділений сірим кольором) з'явиться повідомлення про помилку, а **колір рядка зміниться на червоний**. Червоним кольором буде виділена також невірно написана команда.

Якщо помилки відсутні, зліва від команд набраної програми з'являться адреси, і сама програма буде готова до виконання. Після асемблювання кнопка **"*Assm*"** зміниться на кнопку **"*Step*"**. Таким чином, є можливим виконувати програму по командно **в кроковому режимі**, натискаючи кнопку **"*Step*"**після виконання кожної команди, або **в автоматичному режимі**, коли виконується вся програма, натиснувши один раз кнопку **"*Run*"**. В останньому випадку програму слід закінчувати командою ***"Stop".***

При написанні програми можна користуватися для копіювання її фрагментів та вставки в будь-якому місці "Панелі коду Асемблера" кнопками ***"Copy"*** *та* ***"Paste"***.

Щоб зупинити виконання прорами і скинути в початковий стан регістри мікроконтролера емулятора необхідно натиснути кнопку **"*Reset*".**

3.1.3. Записати в звіт зміни в вікнах регістрів мікроконтролера, які використовуються при виконанні програм, за прикладом, наведеним в табл. 1 (для 5..6 команд з програми).

Таблиця 1. Результати виконання команд

№	Команда	Код	Виконувана операція	Вміст використовуваних регістрів і комірок пам'яті до і після виконання		Пояснення
				До	**Після**	
1	MOV A,R0	E8	Пересилання байту даних з регістру R0 в акумулятор A	A/00 PC/00 PSW/00	A/F2 PC/01 PSW/01	
2	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…

***Примітка**

1. Якщо Ви хочете виконати якусь з команд над вмістом регістру чи комірки пам'яті, наприклад, команду пересилання з регістру в регістр, необхідно в регістр, з якого буде здійснене пересилання, командою MOV попередньо записати якесь значення операнду (адресу чи константу).

2. Програма, що виконується, буде записана в пам'ять програм, вміст якої можна побачити, натиснувши на кнопку ***"Data memory"*** в нижній частині **"Панелі пам'яті даних та програмної пам'яті"**, що знаходиться зліва від **"Панелі коду Асемблера"**. Після натискання кнопка ***"Data memory"*** зміниться на кнопку ***"Code memory"***, тобто буде висвічуватися в полі пам'яті вміст пам'яті програм.

4. Контрольні запитання

1. Використовуючи електричну принципову схему, пояснити, як під'єднаний двигун постійного струму до мікроконтролера?
2. Яке призначення лінії P3.5 мікроконтролера?
3. Пояснити алгоритм роботи програми.

5. Література.

1. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин и др. - М.: Энергоатомиздат, 1990. – 224 с.
2. Микропроцессоры / Под ред. Преснухина Л.Н., т. 1, 2, 3. - М.: Высшая школа, 1986.
3. Бойко Н.П., Стеклов В.К. Системы автоматического управления на базе микро-ЭВМ. - К.: Техника, 1989. - 182 с.