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 MCS51 з використанням програмної моделі EdSim51. Керування двигуном постійного струму. Програмування таймера/лічильника

Тернопіль 2023

Методичні вказівки для виконання лабораторної роботи № 5 «Програмування мікроконтролера MCS51 з використанням програмної моделі EdSim51. Керування двигуном постійного струму. Програмування таймера/лічильника» з курсу «Проектування мікропроцесорних систем керування технологічними процесами»/ Укл.: Медвідь В.Р., Пісьціо В.П. - Тернопіль ТНТУ, 2023 - 12 с.

Розглянуто і затверджено на засіданні кафедри автоматизації технологічних процесів і виробництв (протокол № 1 від 30.08.2023 року)

Лабораторна робота № 5

Програмування мікроконтролера MCS51 з використанням програмної моделі EdSim51. Керування двигуном постійного струму. Програмування таймера/лічильника

1. Теоретичні відомості

1.1. Керування двигуном постійного струму в програмному симуляторі EdSim51

Двигун постійного струму обертається в напрямку за годинниковою стрілкою, напрямок і кількість обертів відображається на семисегментному дисплеї Disp 3 – Disp 0 (рис.1). Дисплей показує тільки до дев'яти обертів, а потім перезавантажується.

На двигун подається постійна напруга з виходів X,Y «bridge driver», входи якого A,B з'єднані відповідно з виходами P3.1, P3.0, а вхід керування CS – з виходом P0.7. Активний рівень на цьому вході для «bridge driver» - логічна «1».

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

Збільшення таймера на одиницю щоразу призводить до руху двигуна.

Значення молодшого байту таймера 1 пересилається в акумулятор A і це значення разом з покажчиком даних DPTR (DPH і DPL) використовується для отримання коду семисегментного індикатора з пам'яті програм.

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

Обертання двигуна може бути змінене на рух проти годинникової стрілки натисканням кнопки SW0 лінійки перемикачів «Switch bank», під'єднаної до лінії P2.0.

Напрямок обертання двигуна зберігається в F0 («1» для руху за годинниковою стрілкою, «0» - для руху проти годинникової стрілки). ***Флажок користувача F0*** займає біт D5 регістра стану PSW. Восьмирозрядний регістр PSW має наступний формат, в якому він також зображений в EdSim51:

D7	D6	D5	D4	D3	D2	D1	D0	
C	AC	F0	RS1	RS0	OV	-	P	PSW

Крапка дисплею, під'єднана до виводу P1.7, вказує на напрямок руху двигуна - якщо десяткова крапка ***світиться***, двигун обертається ***проти годинникової стр***ілки, якщо ***не світиться*** - двигун обертається ***за годинниковою стрілкою***.

Для обертання двигуна за годинниковою стрілкою, на входи «bridge driver» подаються логічні рівні A=«0», B=«1», якщо проти – відповідно A=«1», B=«0». Якщо A=B=«0» - двигун зупиняється.

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 натиснута кнопка SW0 і напрямок руху двигуна має бути змінений.

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

1.2. Програмування таймера/лічильника

Два програмовані 16-бітні таймери/лічильники (T/C0 і T/C1) можуть бути використані як таймери або лічильники зовнішніх подій.

При роботі в режимі *таймера* вміст T/C інкрементується у кожному машинному циклі, тобто через кожні 12 періодів кварцевого резонатора.

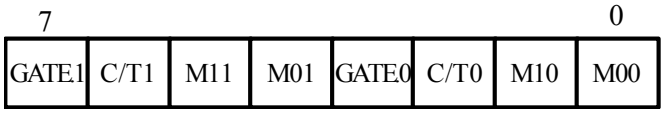
При роботі в режимі *лічильника* вміст T/C інкрементується під впливом переходу з «1» у «0» зовнішнього вхідного сигналу, що подається на відповідний вивід MCS51 (T0 або T1). Опитування значення зовнішнього вхідного сигналу виконується в момент часу S5P2 кожного машинного циклу. Вміст лічильника буде збільшений на 1 у випадку, якщо в попередньому циклі поступив сигнал високого рівня «1», а в наступному – сигнал низького рівня «0».

Нове значення лічильника буде сформоване в момент S3P1 у циклі, що йде за тим, у якому був виявлений перехід сигналу з рівня «»1 у «0». На розпізнавання переходу потрібно

два машинні цикли, тому максимальна частота підрахунку вхідних сигналів дорівнює 1/24 частоти резонатора. На тривалість періоду вхідних сигналів обмежень зверху немає.

Для гарантованого зчитування вхідного сигналу, що підраховується, він повинен утримувати значення «1» як мінімум протягом одного машинного циклу MCS51.

Формат регістра режиму TMOD для таймера/лічильника T1 подано нижче.



Таблиця 1. Регістр режиму роботи таймера/лічильника TMOD

Ознака	Ім'я і призначення
GATEx	Керування блокуванням. Якщо біт установлено, то таймер/лічильник "x" дозволений доти, поки на вході "INTx" високий рівень і біт керування "TRx" встановлений. Якщо біт скинутий, то T/Cx дозволяється як тільки біт керування "TRx" встановлюється
C/Tx	Біт вибору режиму таймера або лічильника подій. Якщо біт скинутий, то блок працює в режимі таймера від внутрішнього джерела сигналів синхронізації. Якщо біт встановлено, то блок працює в режимі лічильника від зовнішніх сигналів на вході "Tx"
M1.x, M0.x	Режим роботи, що вибирається згідно з таблицею 29

Таблиця 2. Режим роботи таймера залежно від бітів настроювання M1.x та M0.x

M1	M0	Режим роботи
0	0	Таймер сумісний з MCS-48. TL працює як 5-бітний попередній подільник, TH – у режимі, сумісному з таймером MCS-48
0	1	16-бітний таймер/лічильник. TL і TH увімкнені послідовно
1	0	8-бітний таймер/лічильник, що перезавантажується. TH зберігає значення, що повинно бути перезавантажене в TLx у момент переповнення
1	1	Таймер/лічильник 1 зупиняється. У таймері/лічильнику 0 TL0 працює як 8-бітний таймер/лічильник, і його режим визначається керуючими бітами таймера 0. TH0 працює тільки як 8-бітний таймер, і його режим визначається керуючими бітами таймера 1

Старший та молодший байти лічільника мають назву THx та TLx. Керування лічильниками здійснюється загальним *регістром режиму* TMOD та *регістром конфігурації* TCON, стан лічильників відображається у регістрі TCON.

В програмі для завдання використовується програмування таймера/лічильника 1 в режим лічильника зовнішніх подій. Вміст регістру TMOD при цьому має значення 50H або 01010000b (вміст C/T1=1, M01=1).

2. Завдання

1. Дослідити команди програми обертання двигуна.
2. Виконати програму обертання двигуна постійного струму в прямому та реверсному напрямках відповідно до заданого варіанту в ***автоматичному режимі роботи*** натисканням клавіші «***Ran***» на «***Панелі коду Асемблера***» симулятора. Щоб зупинити виконання програми, потрібно натиснути клавішу «***Pause***», яка з'явиться на місці клавіші «***Ran***» під час виконання програми.

Приклад 1. Програма керування двигуном постійного струму

```
ORG 00H                 ; встановити початкову адресу завантаження програми
                        ; 00H в пам'ять програм
```

```
    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                    ; увімкнути молодший розряд дисплею DISP 0

again:
    CALL setDirection           ; встановити напрямок 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 2, і тому
                                ; другий код в таблиці буде скопійований в A)
    MOV C, F0                   ; перемістити значення напрямку обертання
                                ; двигуна в флажок перенесення C,
    MOV ACC.7, C                ; а звідти в ACC.7 (завантажиться десяткова точка
                                ; дисплею DISP 0. яка вкаже напрямок обертання
                                ; двигуна)
    MOV P1, A                   ; і завантажить 7-сегментний код числа обертів та
                                ; напрямок обертання двигуна.
                                ; Індикатор для відображення – DISP 0
    JMP again                   ; перехід на мітку again

setDirection:
    PUSH ACC                    ; завантаження значення A в стек
    PUSH 20H                    ; зберегти адресу 20H (першого байту адреси
                                ;  розташування в оперативній пам'яті) в стек
    CLR A                       ; очистити вміст A
    MOV 20H, #0                 ; очистити вміст за адресою 20H
    MOV C, P2.0                 ; завантажити значення кнопки SW0 в флажок C
    MOV ACC.0, C                ; і скопіювати його в ACC.0
    MOV C, F0                   ; завантажити напрямок руху двигуна в C
    MOV 0, C                    ; і перейти за адресою 20H (що має адресу біта 0)
    CJNE A, 20H, changeDir      ; порівняти SW0 з F0 якщо вони не рівні, напрямок
                                ; руху двигуна має бути змінений
    JMP finish                  ; якщо вони однакові, напрямок руху двигуна
                                ; не має бути змінений
changeDir:
    CLR P3.0                    ;
    CLR P3.1                    ; зупинка двигуна
    CALL clearTimer             ; скинути таймер 1 (зміна руху перезапускається при
                                ; зміні напрямку обертання двигуна)
    MOV C, P2.0                 ; зберегти значення SW0 в регістр ознак C і тоді
    MOV F0, C                   ; скопіювати в F0 - це новий напрямок руху двигуна
    MOV P3.0, C                 ; завантажити значення SW0 для біту керування
                                ; двигуном 1
```

```
        CPL C                   ; Інвертувати біт 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 із стеку
        POP ACC                 ; повернення значення A із стеку
        RET                     ; повернення з підпрограми

clearTimer:
        CLR A                   ; скидання в нуль 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. Кодова
                                ; таблиця подана далі:
        DB 11000000B, 11111001B, 10100100B, 10110000B, 10011001B, 10010010B, 10000010B,
11111000B, 10000000B, 10010000B
```

Схема електрична принципова підключення 7-сегментного дисплею та двигуна постійного струму, роботу якої симулює EdSim51, показана на рис. 1.

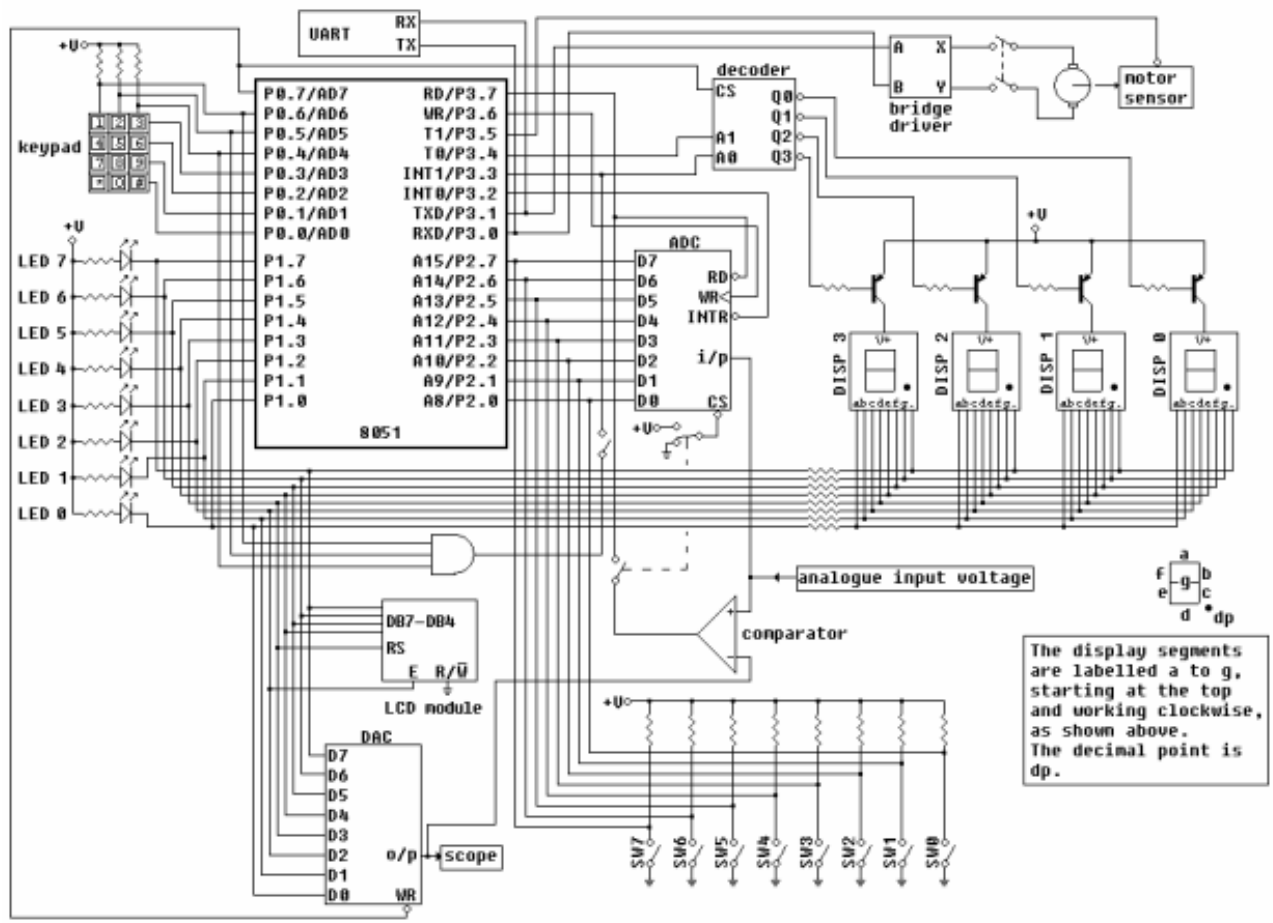


Рис. 1. Схема підключення семисегментного дисплею і двигуна постійного струму

Варіанти індивідуальних завдань

№	Зміст індивідуального завдання
1	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 10H; - задати напрямок обертання двигуна кнопкою SW1, під'єднаної до лінії P2.1 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 1 індикатора, а напрямок обертання – на його точку.
2	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 14H; - задати напрямок обертання двигуна кнопкою SW2, під'єднаної до лінії P2.2 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 2 індикатора, а напрямок обертання – на його точку.
3	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 18H; - задати напрямок обертання двигуна кнопкою SW3, під'єднаної до лінії P2.3 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 3 індикатора, а напрямок обертання – на його точку.
4	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 1CH; - задати напрямок обертання двигуна кнопкою SW4, під'єднаної до лінії P2.4 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 3 індикатора, а напрямок обертання – на його точку.
5	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 08H; - задати напрямок обертання двигуна кнопкою SW1, під'єднаної до лінії P2.1 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 0 індикатора, а напрямок обертання – на його точку.
6	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 15H; - задати напрямок обертання двигуна кнопкою SW2, під'єднаної до лінії P2.2 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 3 індикатора, а напрямок обертання – на його точку.
7	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 11H; - задати напрямок обертання двигуна кнопкою SW3, під'єднаної до лінії P2.3 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 1 індикатора, а напрямок обертання – на його точку.

№	Зміст індивідуального завдання
8	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 16H; - задати напрямок обертання двигуна кнопкою SW5, під'єднаної до лінії P2.5 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 2 індикатора, а напрямок обертання – на його точку.
9	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 1AH; - задати напрямок обертання двигуна кнопкою SW4, під'єднаної до лінії P2.4 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 3 індикатора, а напрямок обертання – на його точку.
10	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 08H; - задати напрямок обертання двигуна кнопкою SW1, під'єднаної до лінії P2.1 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 0 індикатора, а напрямок обертання – на його точку.
11	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 22H; - задати напрямок обертання двигуна кнопкою SW0, під'єднаної до лінії P2.0 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 3 індикатора, а напрямок обертання – на його точку.
12	Написати і виконати на симуляторі програму обертання двигуна за Прикладом 1, для чого попередньо: - встановити початкову адресу завантаження програми в пам'ять програм, рівною 1EH; - задати напрямок обертання двигуна кнопкою SW1, під'єднаної до лінії P2.1 (кнопка натиснута – рух проти годинникової стрілки); - виводити швидкість обертання двигуна на розряд DISP 4 індикатора, а напрямок обертання – на його точку.

3. **Додати у звіт копію екрану** з виконаною програмою до Завдання на програмному симуляторі відповідно до вказаного варіанту.

4. Послідовність виконання роботи

4.1. Вивчити команди відповідно до завдання. Завантаження команд в симулятор проводити наступним чином:

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

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

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);

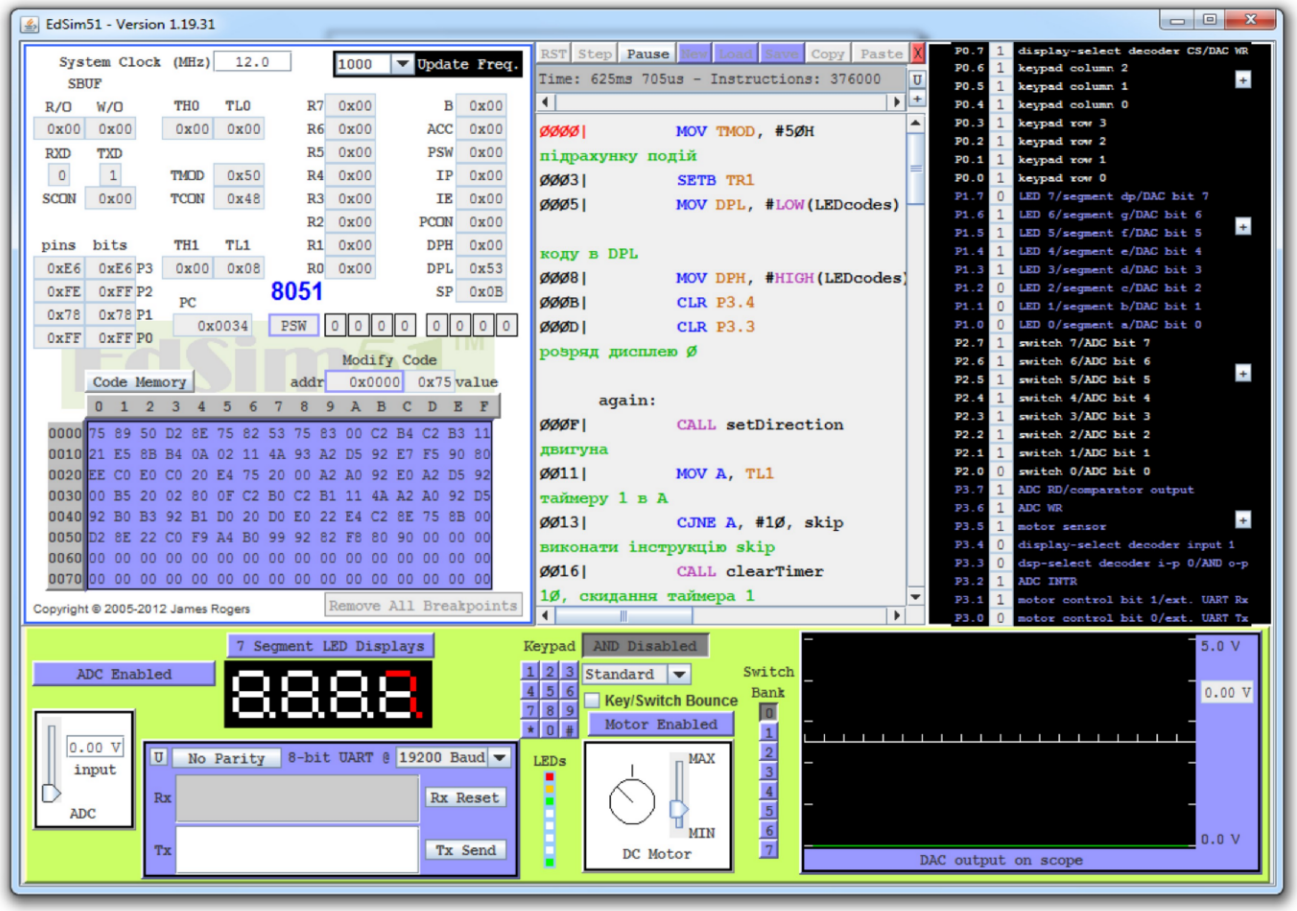


Рис. 2 Інтерфейс програмного симулятора

- **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.

4.1.2. 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. 2 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 покомандно **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 директивою ***“End”.***

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

Щоб зупинити виконання програми і скинути в початковий стан регістри мікроконтролера симулятора, необхідно натиснути кнопку **“*Reset*”.**

***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.

5. Контрольні запитання

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

Додаток 1

Приклад програми для виконання

```
    ORG 10H                     ; встановити початкову адресу завантаження програми
                                ; 10H в пам'ять програм
    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                    ;
    SETB P3.3                   ; увімкнути розряд дисплею DISP 1

again:
    CALL setDirection           ; встановити напрямок 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 2, і тому другий код в таблиці буде
                                ; скопійований в A)
    MOV C, F0                   ; перемістити значення напрямку обертання
                                ; двигуна в флажок перенесення C,
    MOV ACC.7, C                ; а звідти в ACC.7 (завантажиться десяткова точка
                                ; дисплею DISP 1, яка вкаже напрямок обертання
                                ; двигуна)
    MOV P1, A                   ; і завантажить 7-сегментний код числа обертів та
                                ; напрямок обертання двигуна.
                                ; Індикатор для відображення – DISP 1
    JMP again                   ; перехід на мітку again

setDirection:
    PUSH ACC                    ; завантаження значення A в стек
    PUSH 20H                    ; зберегти адресу 20H (першого байту адреси
```

```
                                ;  розташування в оперативній пам'яті) в стек
    CLR A                       ; очистити A
    MOV 20H, #0                 ; очистити адресу 20H
    MOV C, P2.1                 ; завантажити значення кнопки SW1 в флажок C
    MOV ACC.0, C                ; і скопіювати його в ACC.0
    MOV C, F0                   ; завантажити напрямок руху двигуна в C
    MOV 0, C                    ; і перейти за адресою 20H (що має адресу біта 0)
    CJNE A, 20H, changeDir      ; порівняти SW1 з F0
                                ; якщо вони не рівні, напрямок руху двигуна
                                ; має бути змінений
    JMP finish                  ; якщо вони однакові, напрямок руху двигуна
                                ; не має бути змінений
changeDir:
    CLR P3.0                    ;
    CLR P3.1                    ; зупинка двигуна
    CALL clearTimer             ; скинути таймер 1 (зміна руху перезапускається при
                                ; зміні напрямку обертання двигуна)
    MOV C, P2.1                 ; зберегти значення SW1 в регістр ознак C і тоді
    MOV F0, C                   ; скопіювати в F0 - це новий напрямок руху двигуна
    MOV P3.0, C                 ; завантажити значення SW1 для біту керування
                                ; двигуном 1
    CPL C                       ; Інвертувати біт 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 із стеку
    POP ACC                     ; повернення значення A із стеку
    RET                         ; повернення з підпрограми

clearTimer:
    CLR A                       ; скидання в нуль 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, що подана
                                ; нижче
    DB 11000000B, 11111001B, 10100100B, 10110000B, 10011001B, 10010010B, 10000010B,
11111000B, 10000000B, 10010000B
```

Рекомендована література

1. Проектування мікропроцесорних систем керування : навчальний посібник, перевидання / Медвідь В.Р., Пісьціо В.П., Козбур І.Р. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. – 360 с.

2. Handbook of Microcontrollers/Predko Michael. NYc. McGraw-Hill. 1998. 861 p.

3. Бойко В. І., Гуржій А. М., Жуйков В. Я. та ін.Схемотехніка электронних схем: У 3 кн. Кн.3 Мікропроцесори та мікроконтролери: підручник. 2-ге вид., допов. і переробл. К.: Вища шк., 2004. 399 с.

4 Мілих В. І., Шавьолкін О. О. Електротехніка, електроніка та мікропроцесорна техніка: підручник; за ред. В. І. Мілих. 2-е вид. К.: Каравела, 2008. 688 с.