НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Лабораторна робота № 7 з курсу ”Проектування мікропроцесорних систем керування технологічними процесами”

Програмування мікроконтролера i8051 з використанням програмної моделі EdSim51. Передача та прийом даних через послідовний порт мікроконтролера

Тернопіль 2019

Методичні вказівки до лабораторної роботи № 7 "Програмування мікроконтролера i8051 з використанням програмної моделі EdSim51. Передача та прийом даних через послідовний порт мікроконтролера" з курсу "Проектування мікропроцесорних систем керування технологічними процесами". Автор: Медвідь В.Р., Тернопіль: ТНТУ, 2019- 10 с.

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

Автор: Медвідь В.Р.

Лабораторна робота присвячена розгляду передачі інформації через послідовний порт у 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 №7

Програмування мікроконтролера i8051 з використанням програмної моделі EdSim51. Передача та прийом даних через послідовний порт мікроконтролера

1. Теоретичні відомості. Послідовний порт в мікроконтролері МК51

Послідовний порт ОМЕОМ МК51 забезпечує повний дуплексний режим обміну інформацією, представленої послідовним кодом (молодшими бітами вперед). До складу послідовного порту входять:

- регістри зсуву,
- передачі і прийому,
- буферний регістр SBUF прийомопередавача, що входить до складу РСФ.

Для керування роботою послідовного порту використовується побітно адресований регістр РСФ SCON, а також біт SMOD у регістрі PCON.

Послідовний порт може працювати в одному з чотирьох режимів.

Режим 0 встановлюється комбінацією бітів SM0 (TCON.7)=0 і SM1 (TCON.6)=0. У цьому режимі і прийом і передача інформації здійснюються через вивід Rx (P3.0).

Вивід Tx (P3.1) є виходом, по якому з мікроконтролера видаються супровідні імпульси (імпульси зсуву).

При виводі інформації (передачі) імпульси зсуву формуються мікроконтролером так, що їхні спади (переходи з «1» до «0») відповідають середині переданого символу і можуть бути використані для фіксації виведених даних.

При вводі інформації (прийомі) її читання з виводу Rx здійснюється у фазі S5P2 машинні цикли (рис. 1); імпульси зсуву встановлюються в одиницю у фазі S6P1, а в 0 - у фазі S3P1 (генератор мікроконтролера формує машинний цикл процесора з дванадцяти тактів резонатора (задаючого генератора) відповідно до рис.1.

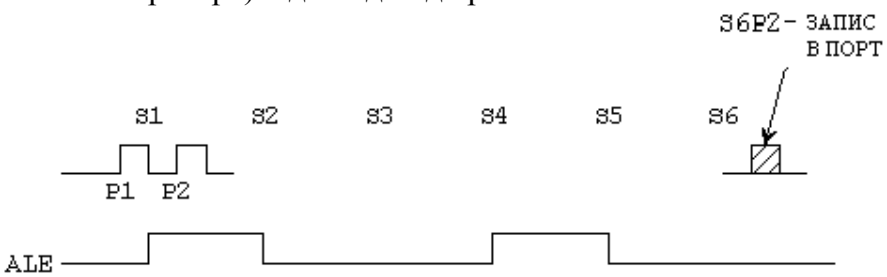

Рис.1

Машинний цикл містить 6 станів керуючого автомата S1...S6, кожен стан розбитий на дві фази P1, P2, що відповідає різним діям процесора).

Таким чином, при вводі інформації в процесор через послідовний порт у режимі 0 повинна бути забезпечена її синхронність відносно генерованих процесором імпульсів зсуву, причому зміна символу, що вводиться, може здійснюватися як по фронту (перехід з 1 у 0), так і по спаду імпульсів зсуву.

Швидкість обміну в режимі 0 фіксована і відповідає частоті проходження машинних циклів (якщо тактова частота процесора складає 12 МГц, то обмін інформацією відбувається зі швидкістю 1 біт/1 мкс) . Одиницею обміну є байт.

Передача з послідовного порту починається автоматично при виконанні мікроконтролером будь-якої команди, що реалізує запис інформації в регістр SBUF. Після того, як вміст SBUF буде передано послідовно на вивід Rx, автоматично встановлюється флажок TI (SCON.1) і формується запит на переривання.

Прийом інформації до процесора через послідовний порт починається при одночасному виконанні умов REN (SCON.4)=1 і RI (SCON.0)=0 (минуле переривання обслуговуване чи скинуте). При виконанні цих умов послідовний порт виробляє на вході Tx вісім імпульсів зсуву, заповнюючи зсувний регістр інформацією з виводу Rx, після чого встановлює флажок RI (RI=1) і формує запит на переривання.

Режим 1 встановлюється комбінацією бітів SM0=0, SM1=1. У цьому й інших, що залишилися, режимах роботи послідовного порту здійснюється по старт-стоповому принципі:

- передача інформації з мікроконтролера виробляється з виводу Tx,
- прийом інформації у мікроконтролер - через вивід Rx.

Старт-стоповий принцип роботи порту полягає в тому, що передана інформація доповнюється двома символами: нульовим стартовим (на початку) і одиничним стоповим (наприкінці) як це показано на рис.2**.** Ці символи служать для розділення переданих байтів. При відсутності обміну на лініях порту встановлюється високий логічний рівень (одиниця).

Формат пакету даних включає в себе один стартовий біт, один біт паритету і два стопових біти.

Початок пакету даних завжди починається низьким рівнем стартового біту. Після нього йде 7 бітів даних символу кода ASCII. Біт парності містить «1» або «0» так, щоб загальна кількість одиниць у 8-бітній групі була непарною. Останім передають два стопових біти, які представлені високим рівнем напруги (еквівалентний ТТЛ-сигнал при передачі літери А показаний на рис.2).

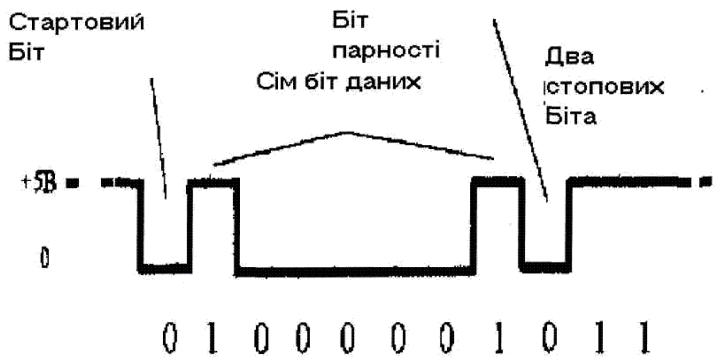

Рис.2 Подання коду літери А сигнальними рівнями ТТЛ

Таким чином, повне асинхронне слово, яке передається, складається з 11 біт (фактично, дані містять тільки 7 біт) і записується у вигляді 01000001011.

Передача починається (як і в режимі 0, при записі інформації в SBUF), з нульового стартового символу, що надходить із процесора на вивід Tx. Поява рівня логічного нуля на вході Rx сприймається портом як початок передачі байту і при виконанні умови (REN=1 і RI=0) ініціює початок прийому стартового символу.

Прийом кожного символу здійснюється трикратним опитуванням з мажоритарним принципом ухвалення рішення (прийнята одиниця, якщо хоча б два прийнятих символи дорівнюють одиниці).

Якщо стартовий символ розпізнається правильно, приймаються ще 9 символів: вісім інформаційних бітів і стоп-біт, записаний мікроконтролером у біт RB8 (SCON.2), після чого встановлюється RI=1 і формується запит на переривання.

Якщо ж стартовий символ не сприймається (тільки один раз із трьох прийняте нульове значення), порт повертається у вихідне положення чекання стартової посилки.

З метою запобігання помилкового прийому послідовним портом повідомлень, формат яких не відповідає показаному на рис.2, передбачено спеціальний режим захисту від помилок формату, встановлюваний програмно бітом SM2 (SCON.5).

Якщо встановити цей біт в одиницю (наприклад, командою SETB SM2), то послідовний порт буде аналізувати сигнал на позиції стопового посилання, і, якщо цей сигнал буде прийнятий як «0» (відсутність стоповго символу), не встановить біт RI і не сформує запит на переривання, ігноруючи прийняту інформацію.

Швидкість передачі в режимі 1 визначається частотою переповнення таймера/ лічильника T/C1, що може при цьому працювати як таймер або як лічильник зовнішніх подій у кожному з режимів 0,1 чи 2, що дозволяє змінювати швидкість передачі в широких межах. У загальному вигляді швидкість передачі може бути описана виразом:

fn=(2smod/32) * f ovt1;

де f ovt1- частота переповнення таймера/лічильника T/C1; SMOD - значення біту PCON.7.

Переривання від таймера/лічильника T/C1 у цьому режимі повинні бути заблоковані.

Режим 2 встановлюється комбінацією біт SM0=1, SM1=0. Формат переданого слова в режимі 2 показаний на рис.3, де TB8=SCON.3 - програмно встановлюваний біт у РСФ SCON.

Дії послідовного порту в режимі 2 в основному аналогічні діям в режимі 1. Швидкість передачі в режимі 2 фіксована і складає значення:

fn=2smod/64f рез;

де f рез-тактова частота задаючого генератора.

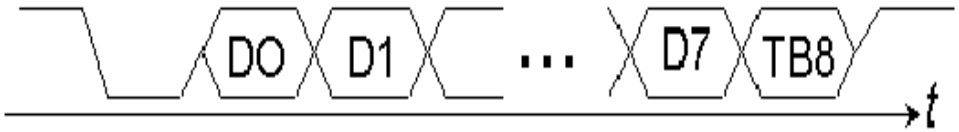

Рис.3

Передача додаткового символу TB8, прийнятого на протилежному боці портом у тригер RB8, надає додаткові можливості послідовного обміну.

Наприклад, при обміні по лінії з відчутним рівнем перешкод символ TB8 може бути використаний як символ парності для перевірки наявності помилок прийому. Крім того, цей символ використовують при побудові систем з кодово - адресним принципом опитування об'єктів, що підключаються до керуючого контролера одним ланцюгом, наприклад, при підключенні до послідовного порту мікроконтролера декількох мікропроцесорних систем.

У цьому випадку звертання до окремих об'єктів з боку керуючого контролера (комп'ютера) здійснюється в два етапи: спочатку передається адреса пристрою, а потім інформація, для нього призначена. Біт TB8 при цьому може бути ознакою характеру переданого посилання, - адреса чи інформація, - і використовуватися кожним з паралельно підключених до контролера пристроїв для ідентифікації приналежності до нього інформації, переданої в даному сеансі.

Режим 3 встановлюється комбінацією біт SM0=1, SM1=1. В режимі 3 робота послідовного порту аналогічна роботі в режимі 2, а швидкість передачі - аналогічна швидкості в режимі 1.

2. Передача даних через послідовний порт МК51

Дослідимо програму, що відправляє напис “ABC” через послідовний порт 8051 до зовнішнього UART зі швидкістю 4800 бод.

Для цього на полі інтерфейсу симулятора (рис. 5) “8-bit UART”, що знаходиться в нижній його частині під зображенням семисегментного індикатора, необхідно встановити швидкість передачі “4800” та наявність біту паритету “Even Parity” (зліва вгорі вікна “8-bit UART”).

Для створення заданої швидкості передачі даних таймер 1 повинен переповнюватися кожні 13 мкс з SMOD, рівним «1», (це рівнозначно тому, як ми можемо отримати 4800 бод при частоті системного тактового сигналу 12 МГц).

Дані передаються з бітом парності, тому для того, щоб отримати правильно дані, що передаються, зовнішній UART повинен бути встановлений на контроль за парністю.

При виконанні програми в автоматичному режимі в полі “Rx” вікна “8-bit UART” повинен з’явитися символ за символом напис “abc”.

2.1 Завдання

1. Дослідити **програму передачі даних** через послідовний порт.

2. Виконати програму на програмному симуляторі:

```
CLR SM0          ;
```

```
        SETB SM1            ; встановити режим 8-біт UART
        MOV A, PCON         ;
        SETB ACC.7          ;
        MOV PCON, A         ; встановити SMOD в PCON подвійну швидкість передачі даних
        MOV TMOD, #20H      ; поставив часовий інтервал таймера 1 в 8-бітний режим
                            ; автоперезавантаження
        MOV TH1, #243       ; завантажити в старший байт таймера 1 число 13 (таймер
                            ; переповнюється кожні 13 мкс)
        MOV TL1, #243       ; помістити таке ж значення в молодший байт, тому,коли таймер
                            ; запускається в перший раз, він буде переповнюватися
                            ; через 13 мкс
        SETB TR1            ; запустити таймер 1
        MOV 30H, #'a'       ;
        MOV 31H, #'b'       ;
        MOV 32H, #'c'       ; поміщати дані, які будуть відправлені, в оперативну пам'ять,
                            ; початкова адреса якої 30H
        MOV 33H, #0         ; записати “0”- припинити передачу даних (коли акумулятор
                            ; містить 0, немає більше даних для відправки)
        MOV R0, #30H        ; поміщати дані в пам'ять з початковою адресою з R0

again:
        MOV A, @R0          ; завантажити вміст РПД, на який вказує R0, до акумулятора
        JZ finish           ; якщо акумулятор містить 0, більше немає даних для пересилання,
                            ; перехід на кінець програми
        MOV C, P            ; в іншому випадку перемістити біт парності в флаг ознак C
        MOV ACC.7, C        ; і перемістити флагперенесення C до акумулятора
        MOV SBUF, A         ; перемістити дані в буфердля відправлення до послідовного порту
        INC R0              ; інкремент R0, щоб адресувати на наступний байт даних, які
                            ; повинні бути відправлені
        JNB TI, $           ; чекати TI, щоб встановити, чи вказує послідовний порт про
                            ; завершення передачі байту
        CLR TI              ; очистка TI
        JMP again           ; відправити наступний байт

finish:
        JMP $               ; нічого не робити
```

Потрібно звернути увагу на те, що відбувається, якщо зовнішній UART не встановлено на контроль парності (тобто, запустити програму з UART без контролю по парності, а потім запустити його знову з встановленим контролером непарності). За допомогою таблиці ASCII пояснити фактичні дані, які відображаються. Схема підключення інтерфейсу зовнішнього UART до МК51 показана на рис. 4.

3. Прийом даних через послідовний порт МК51

Після того, як програма ініціалізації завершена, він чекає даних, що надходить на RXD лінії (дані від зовнішнього UART).

Для цього на полі інтерфейсу симулятора (рис. 5) “8-bit UART” необхідно встановити швидкість передачі “19200” та наявність біту паритету “Even Parity” (зліва вгорі вікна “8-bit UART”).

Будь-який текст, набраний у вікні “Tx” поля “8-bit UART”, пересилається один раз натисканням кнопки “Tx Send”. Після чого він має з’явитися символ за символом у вікні поля оперативної пам’яті симулятора “Data Memory”.

Оскільки, кожен символ передається повністю, він зникає з вікна Tx.

Дані додаються з поверненням ; символ (0D HEX). Зовнішня швидкість передачі UART за замовчуванням 19200.

Для генерування цієї швидкості передачі даних, TH1 повинен бути встановлений у значення -3.

Якщо системний годинник має частоту 12 МГц, може виникати помилка передачі даних при спробі генерувати таку високу швидкість. Тому, для швидкості 19200 бод системний годинник повинен бути встановлений на значення 11.059 МГц.

Програма написана з використанням зайнятості-очікування. Вона постійно перевіряє флаг RI. Цей флаг встановлюється в i8051 апаратно при прийомі даних..

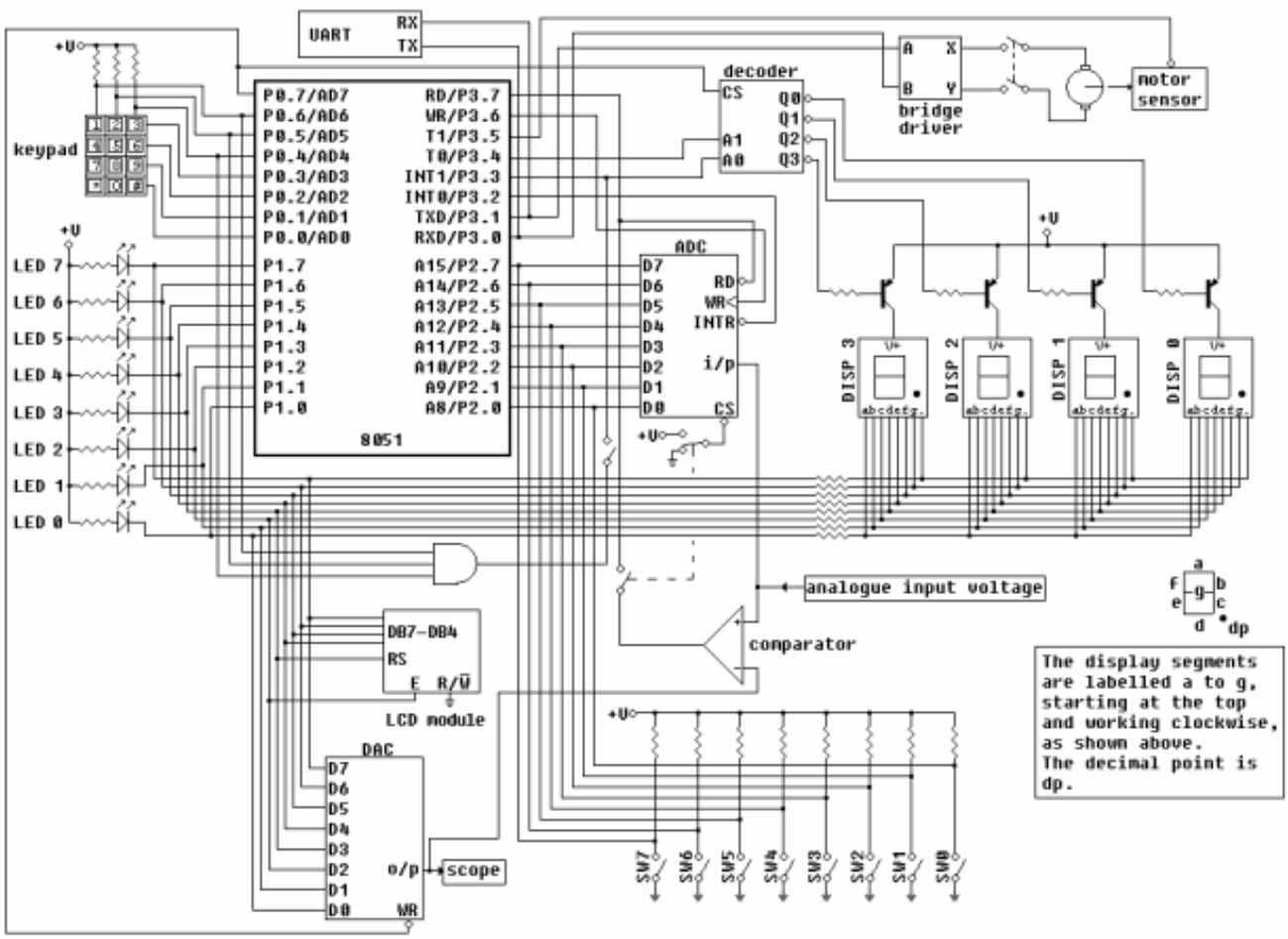

Рис. 4 Схема підключення зовнішнього інтерфейсу UART до МК51

Якщо байт було отримано, програма очищає його, а потім переміщує байт з SBUF A.

Значення прийнятого байту первіряється.

Якщо це символ завершення (0D HEX), то програма переходить до закінчення, в іншому випадку програма переміщує байт в пам'ять даних і повертається в очікуванні наступного байту.

1. Дослідити **програму прийому даних** через послідовний порт.
2. Виконати програму на програмному симуляторі:

```
CLR SM0          ;
SETB SM1         ; встановити режим 8-біт UART
SETB REN         ; дозволити прийом через порт
MOV A, PCON      ;
SETB ACC.7       ;
```

```
    MOV PCON, A          ; встановити SMOD в PCON подвійну швидкість
                         ; передачі даних
    MOV TMOD, #20H       ; поставити часовий інтервал таймера 1 в 8-бітний режим
                         ; автоперезавантаження
    MOV TH1, #0FDH       ; завантажити -3 в старший байт таймера 1 (таймер
                         ; переповнюється кожні 3 мкс)
    MOV TL1, #0FDH       ; помістити таке ж значення в молодший байт, тому коли
                         ; таймер запускається в перший раз він, буде переповнюватися
                         ; кожні 3 мкс
    SETB TR1             ; старт таймеру 1
    MOV R1, #30H         ; завантажити початкову адресу даних в R1

again:
    JNB RI, $            ; чекати прийому байту
    CLR RI               ; очистити флагRI
    MOV A, SBUF          ; завантажити отриманий байт в A
    CJNE A, #0DH, skip   ; порівняти його з 0DH, якщо не рівні, перейти на
                         ; процедуру skip
    JMP finish           ; якщо це символ завершення, перехід до кінця програми

skip:
    MOV @R1, A           ; перехід від A за адресою, на яку вказує R1
    INC R1               ; інкремент R1, щоб вказати на наступну адресу, де будуть
                         ; зберігатися дані
    JMP again            ; перейти назад в очікуванні наступного байту

finish:
    JMP $                ; не робити нічого
```

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 симулятора, зображений на рис. 5.

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. 5);
- **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.

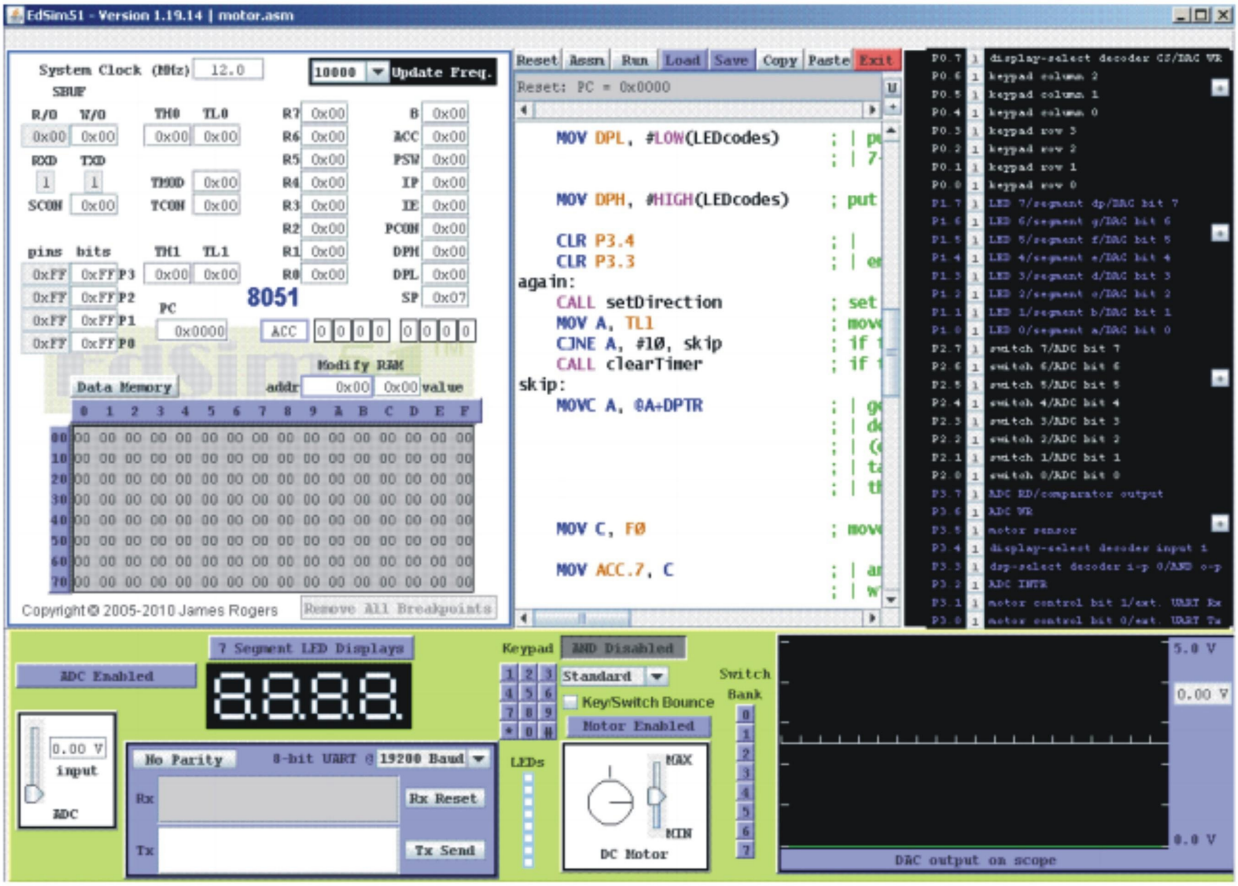

Рис. 5 Інтерфейс програмного стимулятора

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 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 здійснюється прийом-передача даних від мікроконтролера через послідовни й порт.
2. Який формат даних при обміні даними через послідовний порт?
3. Які лінії мікроконтролера використовуються для прийому та передачі даних через послідовний порт?
4. 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.