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 MCS51 з використанням програмної моделі EdSim51. Команди звернення до пам'яті програм MCS51. Передача та прийом даних через послідовний порт мікроконтролера MCS51

Тернопіль 2023

Методичні вказівки для виконання лабораторної роботи № 7 «Програмування мікроконтролера MCS51 з використанням програмної моделі EdSim51. Передача та прийом даних через послідовний порт мікроконтролера MCS51» з курсу «Проектування мікропроцесорних систем керування технологічними процесами»/Укл.: Медвідь В.Р., Пісьціо В.П. - Тернопіль ТНТУ, 2023 - 16 с.

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

Лабораторна робота № 7

Програмування мікроконтролера MCS51 **з використанням програмної моделі EdSim51. Передача та прийом даних через послідовний порт мікроконтролера MCS51**

1. Теоретичні відомості

1.1. Послідовний порт в мікроконтролері MCS51

Послідовний порт мікроконтролера MCS51 забезпечує повний дуплексний (двонаправлений) режим обміну інформацією, представленої послідовним кодом (молодшими бітами вперед). До складу послідовного порту входять:

- регістри зсуву,
- регістри 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 боку мікроконтролера надходять 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 машинного циклу (рис. 1): імпульси зсуву встановлюються в «1» у фазі S6P1, і скидаються в «0» у фазі S3P1 (генератор мікроконтролера формує машинний цикл мікропроцесора з дванадцяти тактів резонатора (задаючого генератора) відповідно до рис.1.

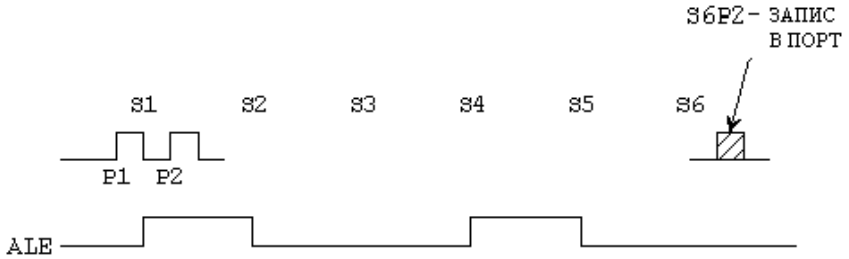


Рис.1

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

Таким чином, при вводі інформації в мікропроцесор 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» до «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 мікропроцесора 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, робота послідовного порту здійснюється по стартстоповому принципі:

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

Стартстоповий 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 містить 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 коду ASCII. Біт парності містить «1» або «0» так, щоб загальна кількість одиниць у 8-бітній групі була парною. Останіми передаються два стопових біти, які представлені високим рівнем напруги (еквівалентний ТТЛ-сигнал при передачі коду літери А показаний на рис.2).

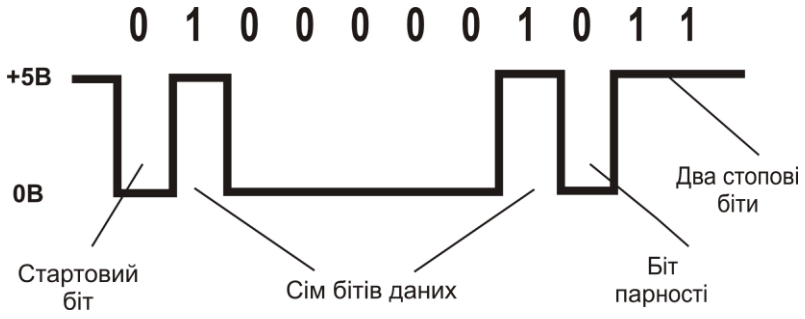


Рис.2. Пересилання через UART коду літери А рівнями ТТЛ

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

Передача починається (як і в режимі 0, при записі інформації в SBUF), з нульового стартового символу, який надходить 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) ініціюється початок прийому стартового символу.

Прийом кожного символу здійснюється трикратним опитуванням з мажоритарним принципом ухвалення рішення (сприймається за одиницю, 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 очікування стартового біту.

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:

$$f_n=(2^{SMOD}/32) * f_{0VT1},$$

де f_{0VT1}- 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 бітів 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 визначається як:

$$fn=2^{SMOD}/64f_{рез};$$

де $f_{рез}$ - тактова частота задаючого генератора.

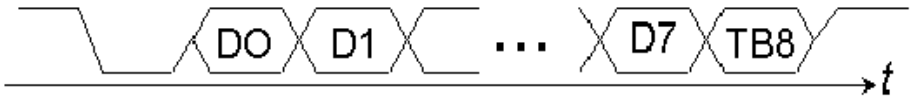


Рис. 3

Передача додаткового символу TB8, який приймається 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.

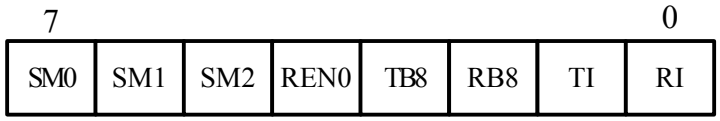
У цьому випадку звертання до окремих об'єктів з боку керуючого контролера (комп'ютера) здійснюється в два етапи: спочатку передається адреса пристрою, а потім призначена для нього інформація. Біт TB8 при цьому може бути ознакою характеру переданого посилання, - адреса чи інформація, - і використовуватися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

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

1.2. Регістр керування/статусу UART

Керування режимом роботи UART здійснюється через спеціальний регістр SCON. Він містить не тільки керуючі біти, що визначають режим роботи послідовного порту, але й дев'ятий біт прийнятих або переданих даних (RB8 і TB8) і біти переривання прийомопередавача (RI і TI). Формат регістра, що керує роботою UART, зображено нижче.

Призначення бітів регістра керування/статусу наведено у таблиці 1.



Формат регістра керування/статусу UART (SCON)

Таблиця 1. Регістр керування/статусу UART

Символ	Ім'я та призначення
SM0, SM1	Біти вибору режиму роботи UART. Встановлюються та скидаються програмно. Записаний у біти код вказує номер режиму
SM2	Біт керування режимом UART. Встановлюється програмно для заборони приймання повідомлення, у якому дев'ятий біт має значення «0»

Символ	Ім'я та призначення
REN	Біт дозволу приймання. Встановлюється та скидається програмно для дозволу або заборони приймання послідовних даних
TB8	Стан восьмого біту передавача. Встановлюється та скидається програмно для задавання дев'ятого переданого біту в 9-бітному режимі UART
RB8	Стан восьмого біту приймача. Встановлюється/скидається апаратно для фіксації дев'ятого прийнятого біту в 9-бітному режимі
TI	Ознака переривання передавача. Встановлюється апаратно у момент закінчення передавання байту. Скидається програмно після обслуговування переривання
RI	Ознака переривання приймача. Встановлюється апаратно у момент закінчення приймання байту. Скидається програмно після обслуговування переривання

Програма шляхом завантаження в старші біти SCON 2-бітного коду визначає режим роботи UART. В усіх *чотирьох* режимах роботи передавання з UART ініціюється будь-якою командою, що записує дані в буферний регістр SBUF. Приймання даних UART у режимі 0 здійснюється за умови, що RI = 0 і REN = 1. У режимах 1, 2, 3 приймання даних починається з приходом старт-біту, якщо REN = 1.

У біті TB8 програмно встановлюється значення дев'ятого біту даних, який буде переданий у режимі 2 або 3. У біті RB8 фіксується в режимах 2 і 3 дев'ятий прийнятий біт даних. У режимі 1, якщо SM2 = 0, у біт RB8 заноситься стоп-біт. У режимі 0 біт RB8 не використовується.

Ознака переривання передавача TI встановлюється апаратно в кінці періоду передавання восьмого біту даних у режимі 0 і на початку періоду передавання стоп-біту в режимах 1, 2 і 3. Відповідна підпрограма обслуговування переривання повинна скинути біт TI.

Ознака переривання приймача RI встановлюється апаратно в кінці періоду приймання восьмого біту даних у режимі 0 і в середині періоду приймання стоп-біту в режимах 1, 2 і 3. Підпрограма обслуговування переривання повинна скинути біт RI.

1.2. Швидкість послідовного обміну

Швидкість послідовного обміну даними UART у різних режимах визначається різними способами.

У режимі 0 швидкість обміну залежить лише від резонансної частоти кварцового резонатора (fрез) і дорінює $f_0 = f_{рез}/12$. За один машинний цикл послідовний порт передає один біт інформації.

В режимах 1, 2 і 3 швидкість обміну даними залежить від значення керуючого біту SMOD у регістрі спеціальних функцій (табл. 2). Формат регістра подано нижче.

Формат регістра спеціальних функцій PCON

Таблиця 2. Біти регістра спеціальних функцій

Символ	Ім'я та призначення
SMOD	Подвоєна швидкість обміну. Якщо біт встановлений в «1», то швидкість обміну вдвічі більша, ніж при SMOD = «0»
–	Не використовуються
GF1, GF0	Ознаки, що визначаються користувачем
PD	Біт зниженої потужності. При встановленні біту в «1» мікроконтролер переходить у режим зниженої споживаної потужності
IDL	Біт холостого ходу. Якщо біт встановлений в «1», то мікроконтролер переходить у режим холостого ходу

У режимах 1 і 3 у формуванні швидкості обміну, крім керуючого біту SMOD, приймає участь таймер 1 (табл. 3). При цьому швидкість обміну залежить від частоти переповнення таймера (OVT1) і визначається у такий спосіб:

$$f_{1,3} = (2^{SMOD}/32)\cdot f_{OVT1}.$$

Таблиця 3. Налаштування таймера 1 для керування частотою роботи UART

Швидкість обміну	Режим роботи	Частота резонатора, МГц	SMOD	Число, що записується у регістр TH1
1 МГц	0	12	X	X
375 кГц	2	12	1	X
62.5 кГц	1, 3	12	1	0FFH
19.2 кГц	1, 3	11.059	1	0FDH
9.6 кГц	1, 3	11.059	0	0FDH
4.8 кГц	1, 3	11.059	0	0FAH
2.4 кГц	1, 3	11.059	0	0F4H
1.2 кГц	1, 3	11.059	0	0E8H
137.5 Гц	1, 3	11.059	0	1DH
110 Гц	1, 3	6	0	72H

Переривання від таймера 1 у цьому випадку повинно бути заблоковано. Сам таймер/лічильник може працювати і як таймер, і як лічильник подій у будь-якому із трьох режимів.

Однак, найзручніше використовувати режим таймера з ***автоперезавантаженням.*** При цьому швидкість обміну визначається виразом

$$f_{1,3} = (2^{SMOD}/32)\cdot(f_{рез}/12)/(256 - (TH1)).$$

У табл. 3 наведено опис способів налаштування таймера/лічильника для отримання типових швидкостей обміну даними через UART.

2. Передача даних через послідовний порт MCS51

Потрібно дослідити програму, що пересилає напис “ABC” через послідовний порт MCS 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 нс з SMOD, рівним «1», (це рівнозначно швидкості 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 по парності.

При виконанні програми ***в автоматичному режимі*** (натиснути ***«Run»***) 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 1 на програмному симуляторі відповідно до вибраного варіанту індивідуального завдання:

Програма 1

```
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       ; поставити часовий інтервал таймера 1 в 8-бітний режим
                             ; автоперезавантаження
        MOV TH1, #243        ; завантажити в старший байт таймера 1 число - 13 (таймер
                             ; переповнюється кожні 13 нс)
        MOV TL1, #243        ; записати таке ж значення в молодший байт, тому, коли таймер
                             ; запускається в перший раз, він буде переповнюватися
                             ; через 13 нс
        SETB TR1             ; запустити таймер 1
        MOV 30H, #'a'        ;
        MOV 31H, #'b'        ;
        MOV 32H, #'c'        ; переслати 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 передачі)
        MOV R0, #30H         ; переслати 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 флажок ознак C
        MOV ACC.7, C         ; і переслати флажок перенесення C до акумулятора
        MOV SBUF, A          ; переслати дані в буфердля відправлення до послідовного порту
        INC R0               ; інкремент R0, щоб адресувати наступний байт даних, які
                             ; повинні бути відправлені
        JNB TI, $            ; чекати TI, щоб встановити, чи вказує послідовний порт на
                             ; завершення передачі байту
        CLR TI               ; скидання в нуль TI
        JMP again            ; відправити наступний байт

finish:
        JMP $                ; нічого не робити
```

Звернути увагу на те, що відбувається, якщо зовнішній UART не встановлено на контроль парності (тобто, запустити програму з UART без контролю парності, а потім запустити її знову з встановленим контролером непарності).

За допомогою таблиці ASCII пояснити фактичні дані, що відображаються.

Схема підключення інтерфейсу UART до MCS51 в схемі симулятора показана на рис. 4.

Інтерфейс симулятора EdSim51 з виконаною Програмою 1 передачі даних показаний на рис. 5.

3. Прийом даних через послідовний порт MCS51

Після того, як програма ініціалізації завершена, мікропроцесор очікує дані, 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”). Після чого виконати програму **в автоматичному режимі** (натиснути ***«Run»***).

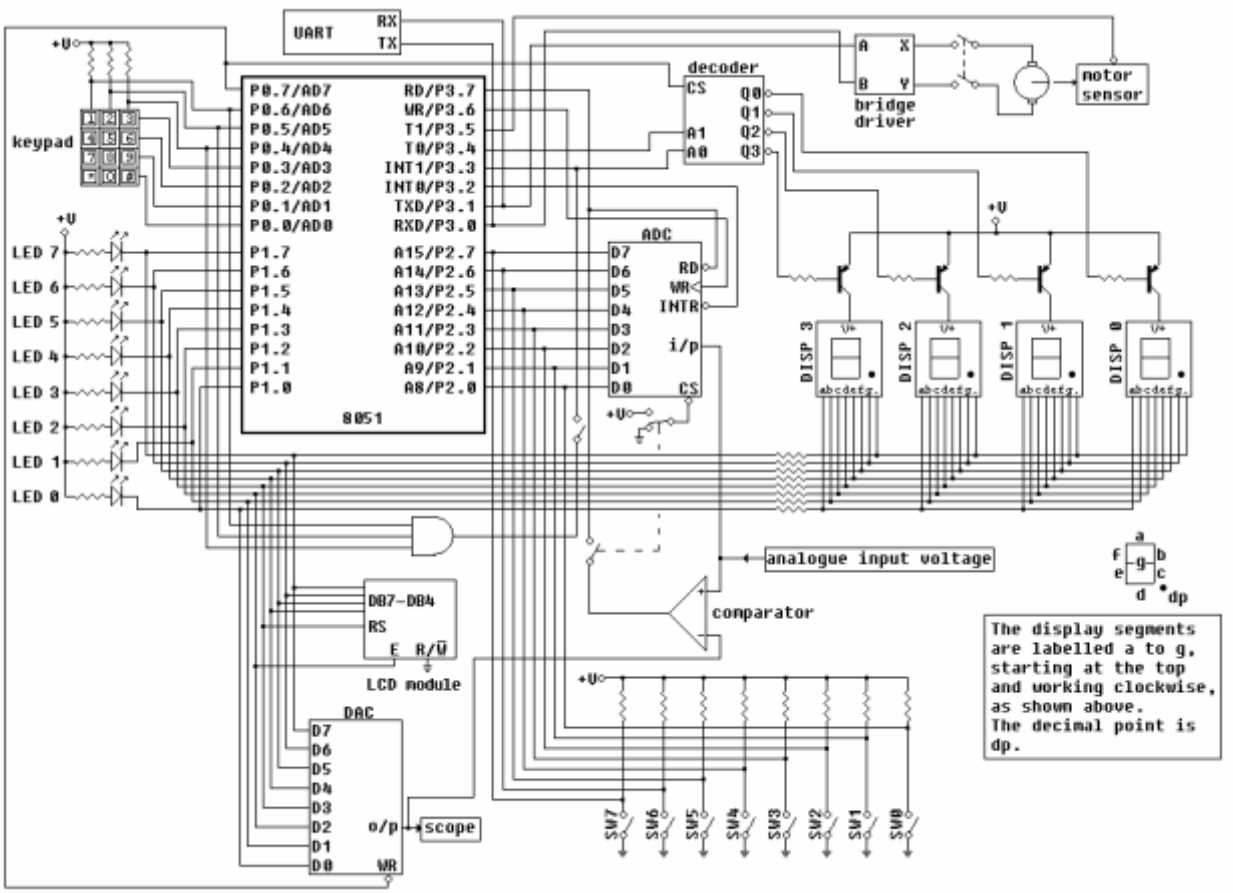


Рис. 4. Схема підключення інтерфейсу UART до MCS51 в схемі симулятора EdSim51

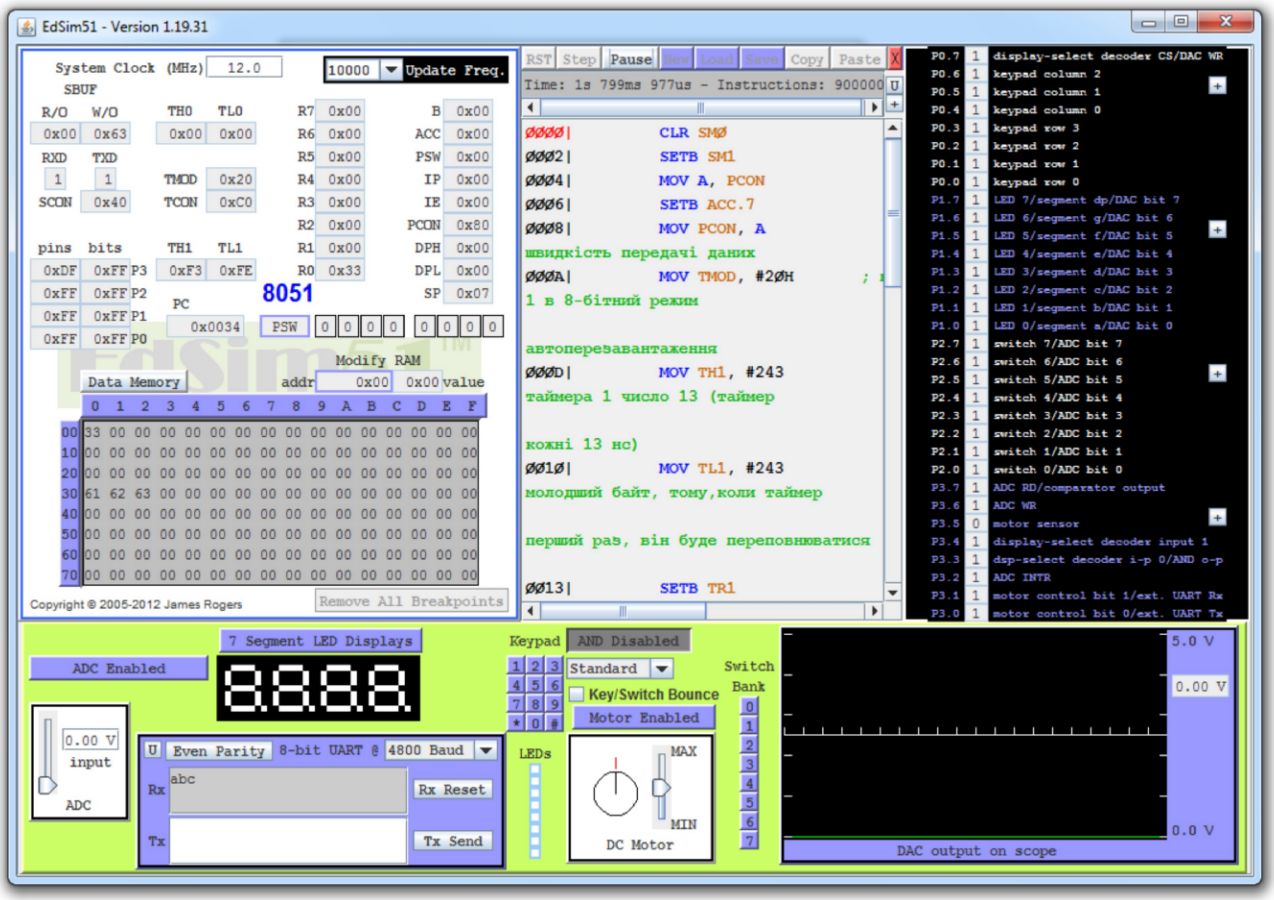


Рис. 5. Інтерфейс симулятора EdSim51 з виконаною програмою передачі даних

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

Після того, як кожен символ передається повністю, він зникає з вікна Tx.

В програмі дані, що завантажуються в акумулятор, порівнюються з символом завершення прийому даних (0D HEX).

Зовнішня швидкість передачі UART за замовчуванням 19200.

Для генерування цієї швидкості передачі даних TH1 повинен бути встановлений до -3.

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

Програма, написана з використанням зайнятості-очікування, постійно перевіряє флажок RI. В MCS 8051 цей флажок встановлюється один раз. Якщо байт було отримано, програма скидає в нуль флажок R1, а потім пересилає байт з SBUF в акумулятор A.

Значення прийнятого байту первіряється. Якщо це символ завершення (0D HEX), то програма переходить на її закінчення, в іншому випадку програма пересилає байт в пам'ять даних і повертається в очікуванні наступного байту.

3.1. Завдання

1. Дослідити **програму прийому даних** через послідовний порт.
2. Виконати Програму 2 на програмному симуляторі відповідно до вибраного варіанту індивідуального завдання.

Перед виконанням програми встановити частоту резонатора симулятора «***System Clock***» рівною 11.059 МГц, а частоту «***Updata Freg***» рівною 100 Гц (рис. 6).

Програма 2

```
    adr EQU 30H          ; змінній adr присвоїти початкову адресу РПД 30H
    n EQU 0AH            ; n визначить кількість символів, які будуть записані в РПД
    MOV R1,#adr          ; початкову адресу пам’яті записати в R1
    MOV R2,#n            ; записати константу n в R2
start:
    MOV @R1,#00H         ; очистити вміст початкової адреси РПД
    INC R1               ; інкремент адреси пам’яті
    DJNZ R2,start        ; декремент вмісту R2 і цикл, якщо R2 не нуль

    CLR SM0              ;
    SETB SM1             ; встановити режим 8-біт UART
    SETB REN             ; дозволити прийом через порт
    MOV A, PCON          ;
    SETB ACC.7           ; встановити старший біт акумулятора в «1»
    MOV PCON, A          ; встановити SMOD в «1» в PCON на подвійну швидкість
                         ; передачі даних
    MOV TMOD, #20H       ; встановити часовий інтервал таймера 1 в 8-бітний режим
                         ; автоперезавантаження
    MOV TH1, #0FDH       ; завантажити -3 в старший байт таймера 1 (таймер
                         ; переповнюється кожні 3 нс)
    MOV TL1, #0FDH       ; записати таке ж значення в молодший байт, тому, коли
                         ; таймер запускається в перший раз, він буде переповнюватися
                         ; кожні 3 нс
    SETB TR1             ; старт таймера 1
    MOV R1,#adr          ; завантажити початкову адресу даних в R1

again:
```

```
        JNB RI, $               ; чекати прийому байту
        CLR RI                  ; скинути в нуль флажок RI
        MOV A, SBUF             ; завантажити отриманий байт в А
        ANL A,#0FH              ; виділити молодшу тетраду акумулятора
        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 А 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
```

Інтерфейс симулятора EdSim51 з виконаною Програмою 2 прийому даних показаний на рис. 6. В UART пересилається текст «12022020» по одному символу в кожну наступну комірку резидентної пам'яті даних (РПД) симулятора «Data Memory», починаючи з адреси 30H.

Щоб переслати нові символи, необхідно зупинити виконання програми (натиснути «***Pause***»), далі скинути вміст регістрів (натиснути ***«RST»***) і знову запустити програму ***(«Run»***).

Після чого в полі Tx UART набрати потрібні символи та однократно натиснути «***Tx Send***». Символи один за одним, починаючи з першого, з'являться в пам'яті даних.

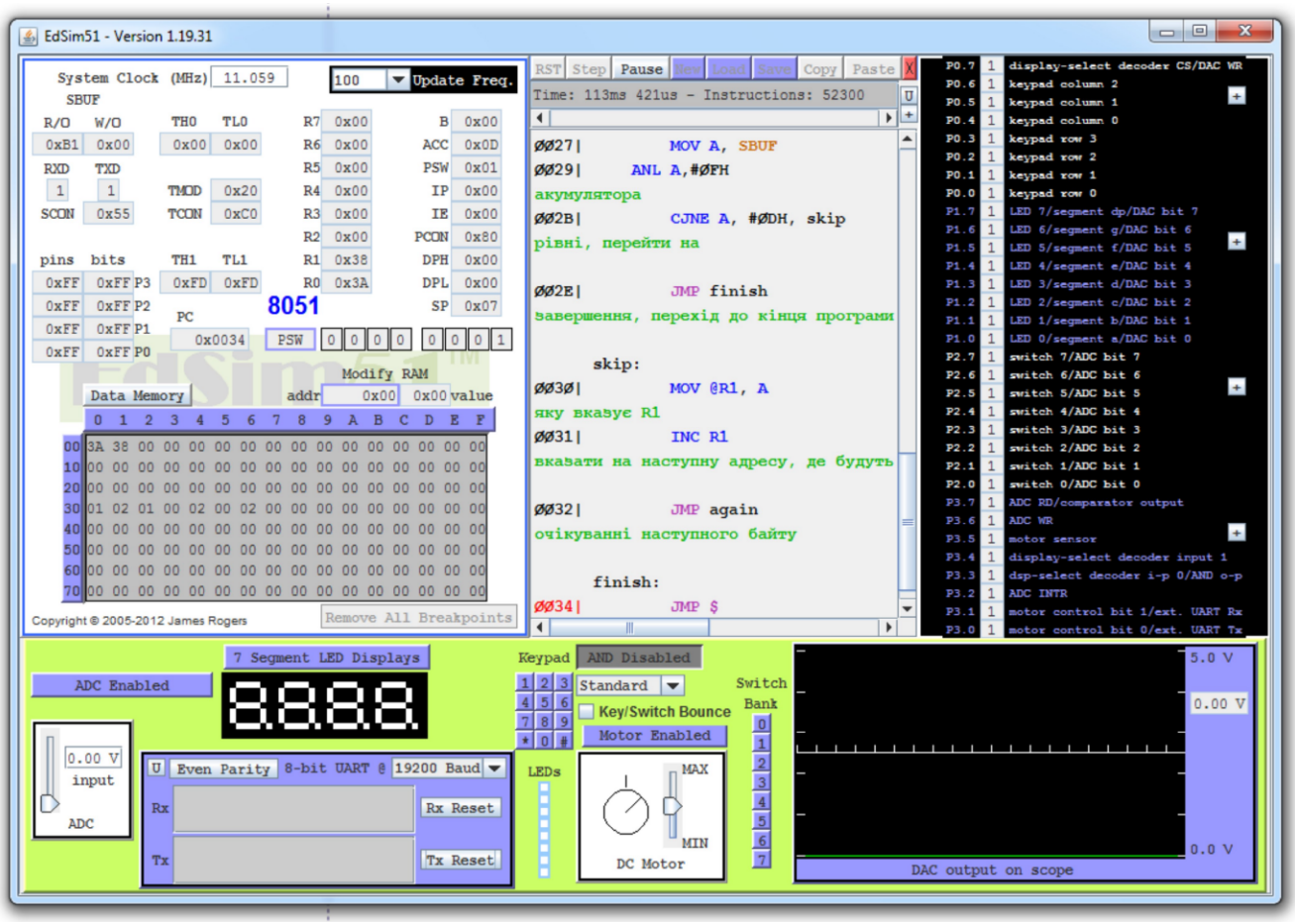


Рис. 6. Інтерфейс симулятора EdSim51 з виконаною програмою прийому даних

Варіанти індивідуальних завдань

№	Зміст індивідуального завдання
1	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 40H, регістр для її зберігання – R0. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 35H, регістр для її зберігання – R0.
2	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 3AH, регістр для її зберігання – R1. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 5DH, регістр для її зберігання – R1.
3	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 50H, регістр для її зберігання – R1. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 3AH, регістр для її зберігання – R0.
4	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 5FH, регістр для її зберігання – R0. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 49H, регістр для її зберігання – R1.
5	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 5DH, регістр для її зберігання – R1. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 65H, регістр для її зберігання – R0.

№	Зміст індивідуального завдання
6	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 3FH, регістр для її зберігання – R0. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 4EH, регістр для її зберігання – R1.
7	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 4CH, регістр для її зберігання – R0. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 5CH, регістр для її зберігання – R0.
8	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 3FH, регістр для її зберігання – R1. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 60H, регістр для її зберігання – R1.
9	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 40H, регістр для її зберігання – R0. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 55H, регістр для її зберігання – R0.
10	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 70H, регістр для її зберігання – R1. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 6EH, регістр для її зберігання – R0.
11	1. Написати і виконати на симуляторі програму для передачі даних через порт UART

№	Зміст індивідуального завдання
	за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 4DH, регістр для її зберігання – R0. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 3DH, регістр для її зберігання – R1.
12	1. Написати і виконати на симуляторі програму для передачі даних через порт UART за прикладом Програми 1 для наступних даних: - переслати своє ім'я через порт із швидкістю 4800 бод при наявності біту паритету та без нього; - початкова адреса в РПД для зберігання символів 3EH, регістр для її зберігання – R1. 2. Написати і виконати на симуляторі програму для прийому даних через порт UART за прикладом Програми 2 для наступних даних: - переслати символи, що відповідають даті народження в форматі день, місяць, рік повністю, наприклад «12022020»; - початкова адреса в РПД для зберігання символів 5EH, регістр для її зберігання – R0.

4. Послідовність виконання роботи

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

4.1.1. Відкрити інтерфейс емулятора, двічі клацнувши клавішею миші на архівованому файлі «EdSim51.jar". Відкриється інтерфейс програмного симулятора, зображений на рис. 7.

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. 7);
- **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.7 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*".**

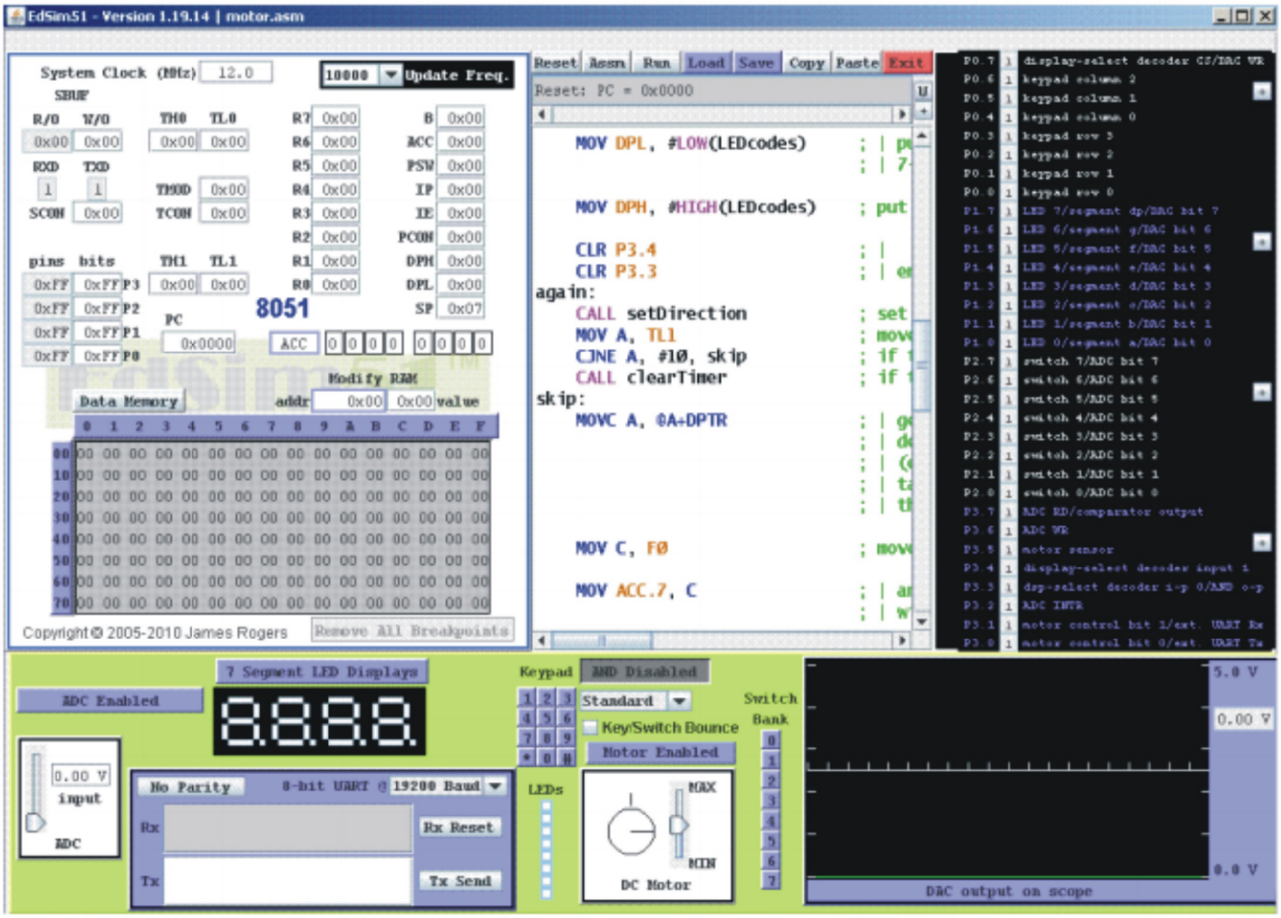


Рис. 7. Інтерфейс програмного симулятора

***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, пояснити, як здійснюється прийом-передача даних від мікроконтролера через послідовний 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.

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.