НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет

імені Івана Пулюя

Кафедра автоматизації технологічних

процесів та виробництв

Методичні вказівки

по роботі з програмним симулятором «AVR simulator IDE»

з курсу

“Мікропроцесорні та програмні засоби автоматизації”

Тернопіль 2020

Методичні вказівки по роботі з програмним симулятором «AVR simulator IDE» з курсу “Мікропроцесорні та програмні засоби автоматизації”.

Методичні вказівки розглянуті і схвалені кафедрою «Автоматизація технологічних процесів та виробництв», протокол № 8 від 18.02.2020 р.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Методичні вказівки по роботі з програмним симулятором «AVR simulator IDE»

AVR Simulator IDE - це графічне середовище розробки на базі мікроконтролерів Atmel для Windows з інтегрованим симулятором (емулятором), базовим AVR-компілятором, асемблером, дизасемблером та налагоджувачем.

AVR Simulator IDE підтримує велику кількість 8-розрядних мікроконтролерів від Atmel, архітектурних продуктів AVR і повного сімейства 90S (вибрані моделі ATmega, ATtiny, AT90S).

Основні можливості програми AVR Simulator:

● Основний інтерфейс моделювання, який показує внутрішню архітектуру мікроконтролера;
● Редактор пам'яті програм FLASH, редактор пам'яті даних EEPROM, редактор простору SRAM;
● Інтерфейс з'єднань мікроконтролера для моделювання цифрових входів/виходів та аналогових входів;
● Змінна швидкість моделювання, моделювання статистики;
● Поточний менеджер для кодування з підтримкою точок зупинки;
● Асемблер AVR, диспетчер AVR;
● Потужний AVR Basic-компілятор з інтелектуальним основним редактором джерела;
Основні можливості компілятора AVR;
- три основних цілі типи даних (1-бітний, 1 байт, 2 байти);
- 4-байтовий (32-розрядний) довгий цілій тип даних з 32-розрядною арифметикою;
- 4-байтовий (32-розрядний) тип даних з єдиною точністю з плаваючою точкою з одиничними точністю математичних функцій, масивів;
- тип рядків даних з великим набором пов'язаних з рядками функцій;
- всі стандартні елементи базової мови, підтримка структурованої мови (процедури та функції), підтримка підтримки керування інтерфейсу Modbus master/slave;
Реалізація карт MMC/SD/SDSC/SDHC (з підтримкою файлової системи FAT16 та підтримкою файлової системи FAT32), підтримка мовлення на високому рівні для використання внутрішньої пам'яті EEPROM, використання внутрішнього модуля A/D конвертора, використання переривань, послідовний зв'язок за допомогою внутрішнього обладнання UART, програмне забезпечення UART, зв'язок I2C з зовнішніми пристроями I2C, зв'язок послідовного периферійного інтерфейсу (SPI), інтерфейсні символьні РК-дисплеї, взаємодія графічних РК-дисплеїв з точковою матрицею 128x64, сервоприводом R/C, керуванням кроковим двигуном, пристроями 1-провідного зв'язку, DS 18S20;
● Термінал послідовного порту ПК для зв'язку з реальними пристроями, підключеними до послідовного порту;
● Інтерфейс моделювання РК-модуля для персональних РК-модулів;
● Графічний інтерфейс моделювання ЖК-модуля для графічних ЖК-модулів 128x64;
● Інтерфейс моделювання ступеневої моторної фази для візуалізації керування кроковим двигуном;
● Модуль моделювання для зовнішніх EEPROM I2C від сімейства 24C;
● Апаратне моделювання інтерфейсу UART;
● Програмний інтерфейс моделювання UART для програмного забезпечення, що виконує UART процедуру;
● Осцилограф (з функцією масштабування) та інструменти для моделювання сигналів;
● 7-сегментний світлодіодний інтерфейс;
● DS18S20/DS18B20 цифровий термометр, інструмент моделювання;
● Пристрій моделювання Modbus (майстер/підлеглий);
● Підтримка зовнішніх модулів моделювання.

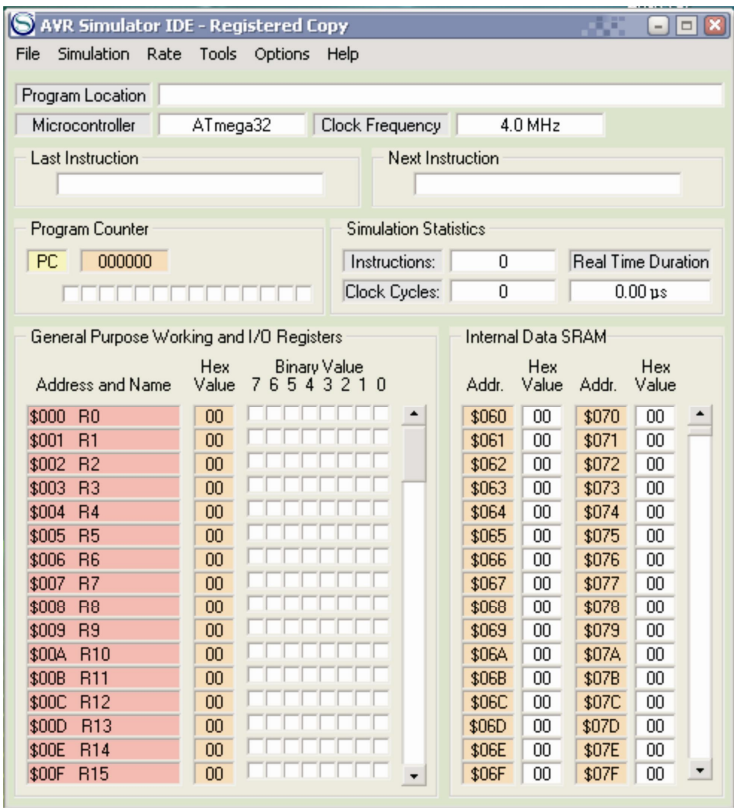

Рис. 1 Вигляд основного поля симулятора

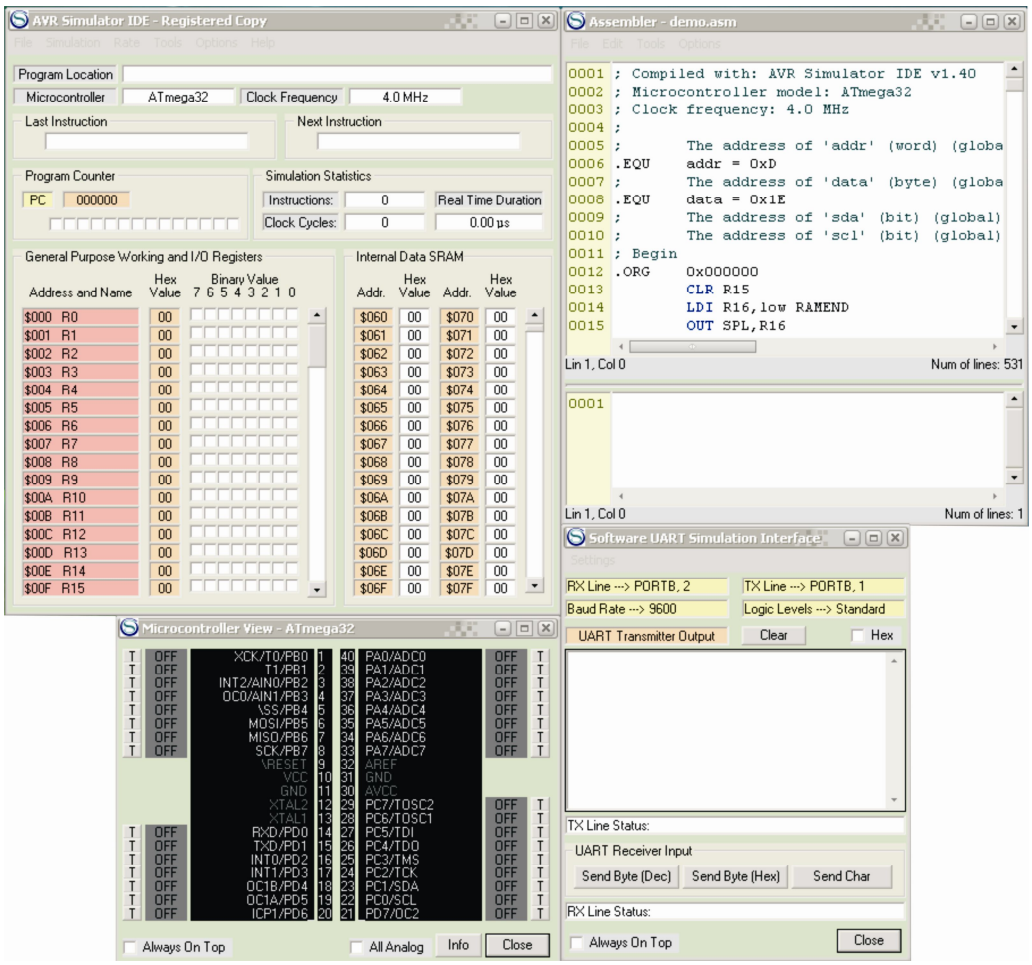

Рис. 2 Вигляд симулятора з полем компілятора Assembler, апаратними виводами контролера, полем послідовного інтерфейсу

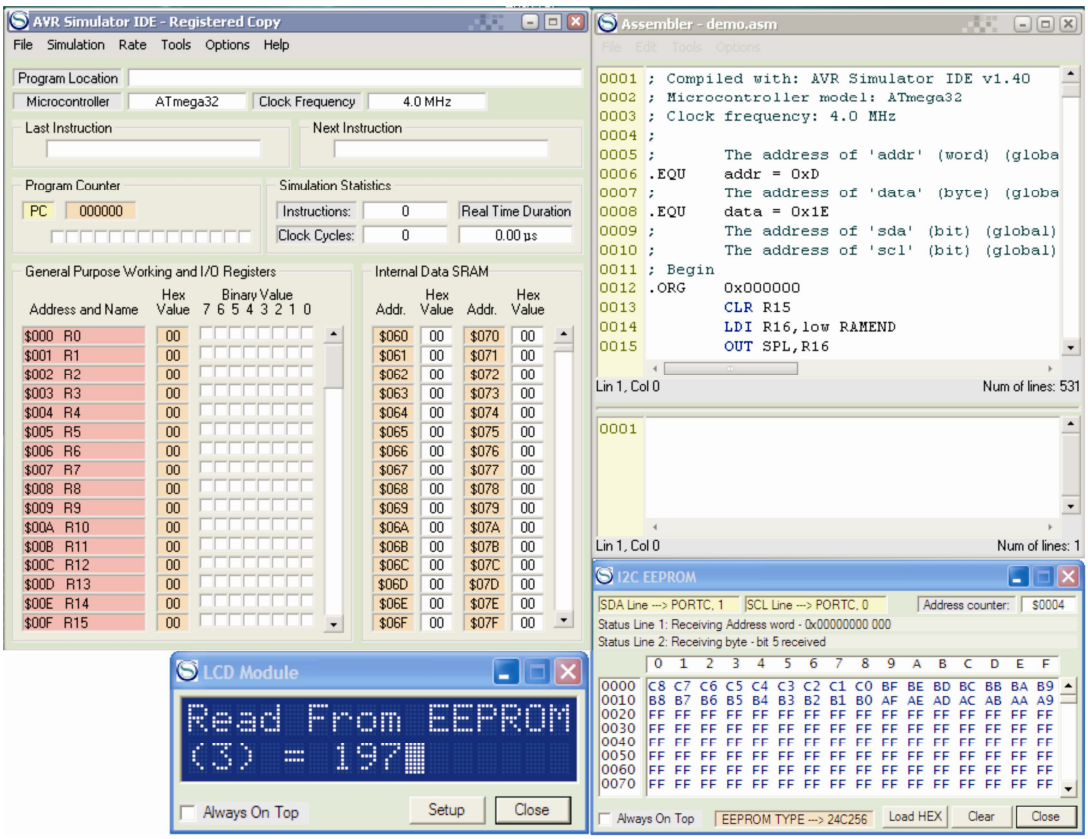

Рис. 3 Вигляд симулятора з полем компілятора Assembler, LED- модулем, I2C EEPROM

Призначення опцій інтерфейсу симулятора

У головному вікні програми відображається стан внутрішніх регістрів мікроконтролерів AVR (регістрів загального користування та регістрів вводу/виводу, SRAM внутрішніх даних та лічильника програм), мнемоніка останньої виконуваної інструкції, мнемоніка наступної інструкції, що буде виконуватися, цикли та інструкції лічильника і тривалість імітації в режимі реального часу.

Меню "Файл" (File)

● Очистити пам'ять

Ця команда скине симулятор до початкового стану та очищає робочі буфери пам'яті програми FLASH та пам'яті EEPROM.

● Програма завантаження

Ця команда завантажує файл програми у буфер пам'яті програми AVR Simulator FLASH. Файл програми має бути у форматі Intel HEX. Розширення за умовчанням - HEX. Після завантаження файл програми можна швидко перезавантажити, натиснувши на його поле розташування в основному інтерфейсі програми або за допомогою комбінації клавіш SPACE.

● Зберегти пам'ять

За допомогою цієї команди можна зберегти вміст робочого буфера пам'яті програми FLASH у файл HEX.

Меню моделювання (Simulation)

● Початок

AVR Simulator IDE входить в режим імітації і починає виконання інструкцій, починаючи з вектора скидання в пам'яті програми.

● Крок

Ця команда вмикається, якщо вибрано швидкість моделювання за кроком. Наступна інструкція виконується при кожному натисканні клавіші F2.

●СТОП

AVR Simulator IDE виходить з режиму симуляції та представляє інформацію про загальну кількість виконаних інструкцій, тривалість імітації та загальну тривалість симуляції в режимі реального часу в тактових циклах та μs на рівні мікроконтролера AVR.

● Виконати наступну BASIC команду

Ця команда доступна для програм, створених інтегрованим BASIC-компілятором. Почнеться швидке моделювання, доки не буде досягнуто наступної BASIC-команди, а потім автоматично переключиться на режим крок за кроком. Комбінація клавіш F4. Якщо доступна швидкість моделювання за кроком, то ця команда також відображатиметься як новий елемент головного меню на інтерфейсі програми.

Меню режимів роботи (Rate)

Це дозволяє користувачеві змінити швидкість моделювання. Воно також доступне під час моделювання.

● Крок за кроком

Інтервал між послідовними інструкціями визначається за бажанням користувача. Коли симулятор знаходиться в режимі крок за кроком, можна змінити значення у всіх регістрах загального користування та регістрі вводу-виводу, лічильника програм та регістра SRAM, натиснувши відповідне поле імені або значення у інтерфейсі програми. Значення в регістрах GPWR та вводу-виводу може бути змінено альтернативно шляхом перемикання окремих бітів графічного представлення, і ця функція може бути використана також з іншими функціями моделювання. Коли вибирається цей режим симуляції, на інтерфейсі програми з'явиться новий елемент головного меню «STEP». Це дозволить легко отримати доступ до команди «Крок» у меню «Симуляція».

● Повільно

Інтервал становить 1500 мсек.

● Нормальний

Інтервал становить 250 мсек.

● Швидко

Інтервал становить близько 50 мсек.

● Надзвичайно швидкий

Інтервал дуже короткий і лінійно залежить від загальної продуктивності комп'ютера.

● Остаточний

Головне вікно тренажера не постійно оновлюється після кожної симульованої інструкції, що значно покращує продуктивність моделювання. Інтервал оновлення можна змінити, скориставшись командою "Змінити інтервал оновлення інтервалів" у меню "Параметри".

Меню "Інструменти" (Tools)

● BASIC-компілятор

Буде відкрито вікно редактора компілятора BASIC. Більш детальна інформація доступна в довідковому посібнику BASIC Compiler. Доступ до нього можна отримати в меню «Довідка» головного вікна програми або у вікні редактора BASIC.

● Вид мікроконтролера

Ця команда відкриє вікно з вибраним типом мікроконтролера AVR. Логічні стани всіх ліній вводу-виводу відображаються графічно, і їх можна вручну змінити на вхідних лініях, натиснувши відповідні кнопки перемикання. Для того, щоб використовувати лінію як аналогову, на ній потрібно клацнути правою кнопкою миші. Після цього можна буде змінити аналогове значення, застосоване до лінії, використовуючи аналоговий слайдер. Якщо це вікно буде відкритим, його буде оновлено під час моделювання.

● Альтернативний переглядач регістрів

Ця команда відкриває альтернативне вікно перегляду для загальних робочих регістрів і регістрів вводу-виводу, які можуть бути змінені.

● Програмований редактор пам'яті

Це доступ до редактора буферу пам'яті програми AVR Simulator FLASH. Можна вручну ввести командний код інструкції, натиснувши на лінію відображення цільового розташування.

● Реєстр пам'яті EEPROM

Це відкриває в AVR Simulator буферний редактор пам'яті даних EEPROM. Якщо це вікно буде відкрито, його буде оновлено під час моделювання. Значення в конкретному місці пам'яті даних можна змінити, натиснувши на нього.

● Стековий редактор систем SRAM

Ця команда відкриє редактор для стеку області внутрішніх даних SRAM. Якщо це вікно буде відкрито, його буде оновлено під час моделювання. Вона також може змінювати значення покажчика стека та значення, що зберігаються в області стеку внутрішніх даних SRAM.

● Асемблер

Ця команда запускає інтегрований асемблер. Вихідні файли Assembler можна редагувати та збирати в одному графічному середовищі. Розширення за умовчанням - ASM. Після успішного процесу складання створюються два нові файли. Один із розширенням HEX, що є файлом програми в форматі Intel HEX, який можна завантажити в пам'ять програми мікроконтролера, а інший - з розширенням LST, що є асемблерним виходом списку. Цей асемблер є відмінним рішенням для складання вихідних файлів розміром до 20K. Для більших файлів процес збірки може тривати певний час, але у файлі не існує жодних обмежень. Їх обмежує лише те, що підтримуються директори асортименту ORG, .EQU, .DB, .DW та .END.

● Дизасемблер

AVR Simulator IDE має внутрішній дизасемблер, який запускається цією командою. Процес розбирання автоматично ініціюється відкриттям цього вікна. Дисасемблер завжди починається з місця розташування вектора скидання. Після завершення операції disassembler покаже файл вихідного списку. Сформований запис може бути збережений на диску. Користувачеві буде запропоновано ввести ім'я вихідного файлу. Розширення за умовчанням - LST.

● Менеджер точок зупинки

Ця команда запускає інтегрований налагоджувач, який може використовуватися для налагодження та моніторингу виконання програми. Файл списків налагоджувачів програми в пам'яті може бути створений внутрішнім дизасемблером. Можна визначити до 10 точок зупинки, натиснувши на окремі рядки у списку завантажених програм. Якщо симуляція починається в режимах більшої швидкості, він автоматично перемикається в режим «Крок за кроком», коли виходить один із цих точок зупинки. Точки зупинки позначені червоними точками, а поточне значення регістру ПК позначено жовтою стрілкою. Існує можливість зберегти вказівник ПК під час моделювання.

● Спеціальні точки зупинки

Цей інструмент моделювання забезпечує функцію для визначення спеціальних точок зупинки, які змінять швидкість моделювання в режимі "Крок за кроком", коли значення попередньо визначеного регістру змінилося або досягне заздалегідь заданого значення. Можна визначити до 5 спеціальних точок зупинки, які одночасно активні.

● 8-розрядний світлодіодний рядок

Цей простий модуль може бути використаний для приєднання до восьми світлодіодів до контактів мікроконтролера. Колір кожного світлодіода можна змінити, натиснувши поле кольорів світлодіодів, а призначення пікселів можна змінити, натиснувши мітку, що показує поточний вибір виводу.

● Матриця клавіатури

Це ще один простий модуль моделювання для матриці клавіатури до 4x4. Призначення ліній можна змінити, натиснувши відповідні їх мітки.

● ЖК-модуль

Ця команда запускає інтегрований симулятор ЖК-модуля. Перш ніж він може бути використаний для моделювання, користувач повинен налаштувати параметри інтерфейсу в діалоговому вікні «Налаштування».

● Графічний ЖК-модуль 128x64

Ця команда запускає інтегрований графічний симулятор ЖК-модуля 128x64. Перш ніж він може бути використаний для моделювання, користувач повинен налаштувати параметри інтерфейсу в діалоговому вікні «Налаштування».

● Моделювання крокового двигуна

Цей інструмент моделювання покаже спрощену (2-полюсну) графічну подачу однополярних фаз крокових двигунів як в автоматичних, так і в напівавтоматичних режимах. Під час імітаційного поля Step Counter буде відображатися поточна абсолютна позиція ротора, виміряна ступенями від першого визначеного положення ротора.

● I2C EEPROM.

Ця команда запускає інтегрований симуляційний модуль для зовнішніх EEPROM I2C з сімейства 24C.

● Апаратне моделювання інтерфейсу UART

Ця команда відкриває інтерфейс для апаратного UART симулятора.

● Термінал послідовного порту ПК

Це інтегрований інструмент, незалежний від симулятора. Цей термінал підключений до послідовного порту комп'ютера і може бути використаний для зв'язку з реальним мікроконтролером для тестування процедур послідовного зв'язку. Номер порту команд і швидкість передачі даних можна встановити за допомогою відповідних команд з меню.

● Програмне забезпечення UART Simulation Interface

Ця команда відкриває інтерфейс для програмного UART симулятора. Це серійний пристрій, який може взаємодіяти із запущеним програмним забезпеченням мікроконтролера, що реалізує процедури UART (висловлювання SERIN, SERININV, SEROUT та SEROUTINV), використовуючи послідовну комунікацію в режимі реального часу, що моделюється на контактах RX і TX.

● Осцилограф

Це дуже корисний інтегрований інструмент для відстеження рівня логіки на шпильках мікроконтролера під час моделювання. Це чотириканальний цифровий осцилограф. Користувач може призначати цільові штифти на канали осцилографа та змінювати довжину інтервалу відображення за допомогою команд в меню «Налаштування». Вхідні та вихідні контакти розфарбовані різними кольорами, які вибираються користувачем. Якщо для каналу осцилографа вибрано параметр Pull-up, при переході від вихідного сигналу до вхідного сигналу, шпильку буде встановлено високий рівень логіки за замовчуванням.

● Генератор сигналів

За допомогою цього інструменту моделювання користувач може визначити до чотирьох незалежних генераторів безперервних імпульсів з вибраними цільовими штифтами, періодами імпульсів та робочими циклами. Відповідна анімація покаже поточну фазу кожного покоління імпульсів.

● 7-сегментний світлодіодний дисплей панелі

Цей інтегрований інструмент моделювання дозволяє користувачеві визначати зв'язок з чотирма 7-сегментними світлодіодними індикаторами. Визначення з'єднання включає призначення призначень для всіх сегментів відображення та одну лінію включення для керування вибором дисплея, коли використовується декілька дисплеїв з паралельним підключенням сегментів - мультиплексування. Активні рівні для всіх ліній з'єднання можна перевернути, щоб відповідати вимогам до обладнання. Існує також можливість змінити колір,

який використовується для фарбування активних світлодіодів на дисплеях та опції Keep Last Display, щоб імітувати повільний ефект відгуку очей для програм, що використовують мультиплексування дисплеїв.

● Дивитись змінні

Під час моделювання програм, написаних з використанням інтегрованого Basic-компілятора, цей інструмент може використовуватися для перегляду поточних значень всіх змінних, оголошених в імітованій програмі. Ця функція корисна для моніторингу пам'яті для моделювання програмних файлів, які не компілюються в інтегрованому Basic- компіляторі. Змінні, додані користувачем, будуть запам'ятовуватись між сеансами, якщо той самий файл програми завантажений у симулятор. Змінні зі списку годинників можна легко видалити за допомогою команди Delete Variable, тому список може містити лише змінні, що мають особливий інтерес.

Інші команди та параметри включають:

- змінити значення змінної (також може бути запущений одним натисканням кнопки миші на змінній зі списку),
- значення показів HEX,
- підтвердження видалення.

● Цифровий термометр DS1820

Це інструмент для моделювання програм, що підтримують зв'язок з пристроєм DS18S20 або DS18B20 за допомогою 1-провідного протоколу. Він показує внутрішню пам'ять пристрою ROM і SRAM, а також функції функціонального генератора CRC. Користувач може змінити тип пристрою, встановити 1-провідний режим інтерфейсу, температуру, яка буде вимірюватися пристроєм, разом із часом перетворення температури, який буде використовуватися для моделювання. Модуль не імітує всі доступні команди ROM та Function. Список імітованих команд можна переглянути, натиснувши кнопку інформації.

● Модуль моделювання пристрою

За допомогою цього інструменту моделювання користувач може відслідковувати і втручатися в прошивку Modbus, що працює в симуляторі, а також створеного інтегрованим компілятором Basic. Більш детальна інформація доступна в Довіднику по основному компілятору.

● Зовнішні модулі

Цей інструмент повинен використовуватися для встановлення інтерфейсу автоматизації з максимум п'ятьма зовнішніми клієнтськими/серверними модулями. Потрібно ввести назву класу зовнішнього пристрою у формі ApplicationName.ObjectName, щоб встановити з ним зв'язок. Зовнішні клієнтські/серверні програми будуть запускатися та завершуються автоматично за допомогою IDE для програмування AVR Simulator. Додаткова інформація доступна в Посібнику з зовнішніх модулів. Доступ до нього можна отримати в меню "Довідка" головного вікна програми.

Меню опцій (Options)

● Виберіть мікроконтроллер

Ця команда використовується для зміни моделі мікроконтролера, яка буде використовуватися в IDE. Це призведе до скидання програми до початкового стану.

● Зміна частоти годинника

Ця команда дозволяє користувачеві змінювати параметр частоти, який використовується для розрахунку тривалості реального часу симуляції. Введене значення в МГц запам'ятовується для майбутніх сеансів. Цей параметр також використовується багатьма операторами в комбінаторі BASIC, які використовують деяку форму процедур синхронізації (WaitMs, WaitUs, Serin, Serout, ...). Значення за замовчуванням становить 4 МГц.

● Конфігурація панелі ярликів

Ця команда вибору відкриває простий у використанні інтерфейс для вмикання та налаштування панелі затишних комбінацій клавіш у головному вікні IDE для легкого доступу

до найбільш часто використовуваних команд меню. Панель може містити до трьох рядків ярликів пункту меню. Усі основні елементи меню вікна IDE доступні для розміщення на панелі.

● Зберегти позиції

За допомогою цієї опції позиції вікон на екрані будуть запам'ятовуватися.

● Перелік реєстру вводу / виводу спочатку

Вибір цієї опції призведе до інверсного порядку відображення списку реєстрів загального призначення та вводу / виводу.

● Зберегти завжди зверху

Якщо вибрано цей параметр, буде замінений параметр Завжди вгорі для всіх вікон із цією функцією.

● Автоматичні параметри запуску

За допомогою цієї утиліти користувачі можуть визначати дії, які будуть виконуватись при запуску додатка. Ці дії включають в себе автоматичне відкривання різних інструментів та моделюючих інтерфейсів з меню «Інструменти» та автоматичне завантаження останніх використаних файлів у симуляторі, асемблері та базовому компіляторі.

● Скидання статистики моделювання

Ця команда встановить нульовий тактовий цикл, лічильник інструкцій та тривалість імітації в режимі реального часу. Ця команда може бути використана для визначення тривалості конкретної частини симульованої програми.

● Змінити час запису EEPROM

Ця команда використовується для зміни кількості мікросекунд, які будуть використовуватися для інтервалу запису EEPROM під час моделювання. Значення за замовчуванням становить 3400 мікросекунд.

● Змінити час конвертації A/D

Ця команда використовується для зміни кількості мікросекунд, які будуть використовуватися для часу перетворення A / D під час моделювання. Значення за замовчуванням - 25 мікросекунд.

● Змінити час передачі/прийому UART

Ця команда використовується для зміни кількості мікросекунд, які будуть використовуватися для часу передачі / прийому UART, за допомогою внутрішнього апаратного симулятора UART. Реальне значення залежить від заданої швидкості передачі. Значення за замовчуванням становить 1000 мікросекунд.

● Змінити непрограмироване значення FLASH / EEPROM

Ця команда використовується для зміни нестандартних бітових значень для пам'яті програми FLASH та пам'яті EEPROM даних від 1 до 0 та навпаки від 0 до 1. Якщо непрограмоване значення бітового значення становить 1, то непрограмоване розташування FLASH і непрограмоване розташування EEPROM міститиме шістнадцяткове значення FF.

Якщо бітове значення становить 0, то вся непрограмована пам'ять заповнюється нулями.

● Компактний перегляд мікроконтролерів

Якщо цей параметр увімкнено, вікно Microcontroller View буде відображатися у більш компактній формі.

● Моделювання нескінченних циклів зупинки

Перевірка цієї опції змусить симулятор автоматично зупинити симуляцію при зустрічі нескінченного циклу.

● Основна програма відстеження

В даний час симульована основна заява з'явиться в основному вікні компілятора, коли цей параметр увімкнено.

● Показати вікна підтвердження

Якщо цей параметр увімкнено, буде показано вікно підтвердження, що показує результати операцій, і буде потрібно закрити відповідь користувача.

● Зберігати вхідні стану на початку моделювання

Якщо цей параметр увімкнено, то стани цифрових та аналогових входів у вікні перегляду мікроконтролера не будуть скидатися до вимкненого цифрового стану та нульового аналогового значення при повторному запуску симуляції.

● Використовувати напругу для аналогових входів

Використовуйте цю опцію, якщо ви хочете бачити значення напруги (0,00V-5,00 В) замість вихідного аналогового значення (0-1023) для аналогових вхідних станів у вікнах перегляду мікроконтролера.

● Постійний аналоговий введення повзунка оновлення

Якщо цей параметр увімкнено, тоді значення аналогового вводу будуть постійно оновлюватися за допомогою прокрутки слайдера аналогового вводу в вікні вікна перегляду мікроконтролера. В іншому випадку оновлення відбудеться після закриття аналогового спливаючого слайдера.

●Змінити інтервал оновлення останньої ставки

Ця команда дозволяє користувачеві змінювати інтервал оновлення (у мілісекундах) для основного інтерфейсу симуляції, коли симуляція працює на кінцевій швидкості. Однак його вартість значно не впливає на продуктивність моделювання. Значення за замовчуванням - 500 мс.

● Налаштування редактора

За допомогою цього інструмента настройки можна змінювати різні властивості основних редакторів кодів і асемблерів коду.

● Змінити колірну тему

Ця команда відкриє діалогове вікно із насиченим списком доступних кольорових тем, завдяки чому користувач може змінити зовнішній вигляд програми.

Меню "Довідка" (Help)

● Довідкові теми

Ця команда буде відображати довідкові теми. Цей файл довідки містить загальну інформацію про програму з описом всіх елементів меню.

Вікно довідки переглядача містить навігаційну панель, в якій показані теми та підтеми відображуваного файлу довідки. Клацніть правою кнопкою миші, на панелі навігації з'явиться спливаюче меню із пунктом Показати всі підтеми та Приховати всі підтепи.

Одним натисканням на елемент з панелі навігації буде переміщено фокус на панель дисплея до відповідної позиції.

Двічі клацніть по предмету, який буде показано / сховати його підтеми. На дисплеї відображається вміст завантаженого файлу довідки. Клацніть правою кнопкою миші, щоб відобразити спливаюче меню із різними параметрами та командами, включаючи копіювання, копіювання RTF, копіювання HTML, друк, збільшення шрифту, зменшення шрифту, скидання шрифтів, завжди на вершині. Вікно довідки переглядача змінювати розмір і буде пам'ятати як його позицію, так і розмір. Вертикальний сепаратор між панелями навігації та дисплеями є рухомим, і його позиція також буде збережена після того, як глядач буде закрито.

● Довідковий посібник для базового компілятора

Довідник з базового компілятора можна буде переглянути у переглядачі довідки.

● Зовнішні модулі вручну

Посібник з зовнішніх модулів можна буде переглянути у переглядачі довідки.

● Перевірити наявність оновлень

Цей інструмент дозволить користувачеві встановити зв'язок з веб-сайтом OshonSoft.com, щоб перевірити наявність нового завантаження програмного забезпечення. Файл журналу версії відображатиметься після отримання відповіді від веб-сайту.

● Інтерфейс звіту про помилку

Цей інтерфейс слід використовувати для відправки звітів про можливі помилки в програмне забезпечення на OshonSoft.com. Крім написаної частини користувача повний звіт містить частину, яка створюється програмним забезпеченням (системний звіт).

● Про...

Ця команда покаже основну інформацію про пакет програмного забезпечення.

● Переглянути інформацію про ліцензію

Ця команда покаже інформацію про встановлену ліцензію на програмне забезпечення. Кнопка "Інформація" відображатиме інформацію про модулі додаткового підключення, увімкнуті ліцензією.

Особливі примітки

● Інструкція SPM не моделюється даним симулятором

● Сторожовий таймер не моделюється (інструкція WDR виконується як NOP)

● Режим вимкнення живлення не моделюється (інструкція SLEEP зупинить симуляцію).

Перелік периферійних пристроїв, які моделюються симулятором

● Цифровий вхід / вихід
● Пам'ять EEPROM даних
● Переривання
● Зовнішні переривання INT0, INT1
● модуль перетворювача A/D
● Модуль аналогового компаратора
● Модуль USART
● Модуль таймера / Counter0 (з PWM)

2. **Завдання на лабораторну роботу:** ознайомлення з основними опціями програмного симулятора на прикладі виконання програми «demo.hex", яка здійснює програмування та вивід даних через порти та в пам'ять EEPROM даних.

1. Вивчити програмну модель AVR Simulator IDE.
2. Вивчити команди арифметичних, логічних операцій, операцій з портами та лічильником AVR – контролера.
3. Дослідити роботу програм за вказівкою викладача та вміст регістрів контролера, які використовуються при виконанні цієї програми.
4. Записати для команд асемблера, вказаних викладачем, коментар щодо їх призначення.

3. Послідовність роботи з симулятором при виконанні програм

Виконаємо програму відповідно до вказаного викладачем завдання в AVR Simulator ID, для чого необхідно:

1. Запустити AVR Simulator IDE;
2. Натиснути Options | Select Microcontroller;
3. Вибрати мікроконтролер Atmega32 і натиснути кнопку Select;
4. В папці «Program Files» Вашого комп'ютера знайти папку «AVR Simulator IDE» з інсталяцією симулятора;
5. Відкрити файл «demo.asm» і скопіювати його вміст;
6. Натиснути Tools і у вікні, що розкриється, вибрати «Assembler». Відкриється вікно компілятора «Assembler» (рис. 2);
7. Вставити скопійований раніше файл «demo.asm» у вікно «Assembler»;
8. Натиснути Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з'явиться лістинг відкомпільованого файлу (рис. 4). Одночасно буде створено об'єктний файл «demo.hex»;
9. Вибрати File | Load Program і завантажити створений файл «demo.hex»;

10. Натиснути Tools | LCD Module (відкриється вікно з LCD дисплеєм);

11. Натиснути Tools | 8xLED Board (відкриється вікно з дисплеєм, що містить вісім світлодіодів);

12. Натиснути Tools | Stepper Motor Phase Simulation (відкриється вікно симулятором уніполярного крокового двигуна) (рис. 5);

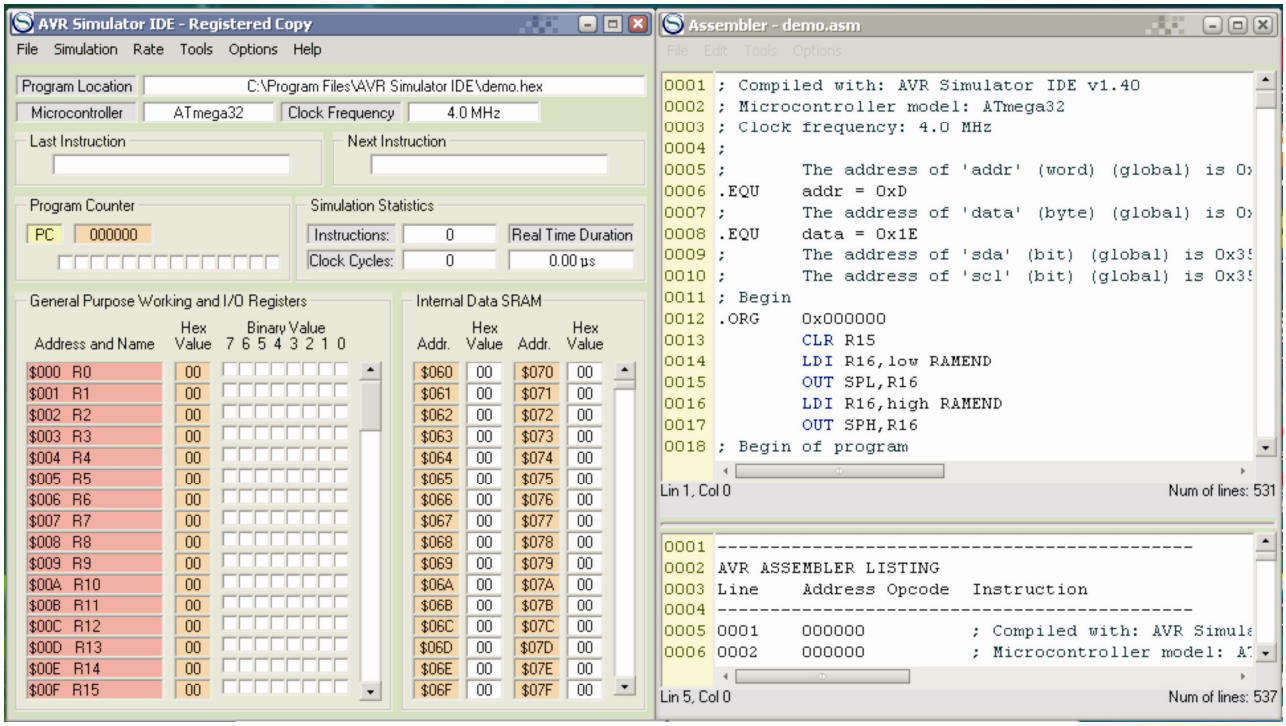

Рис. 4 Вигляд екрану з програмою

13. Якщо програма буде виконуватися в кроковому режимі, то в основному вікні симулятора натиснути Rate | Step By Step, а далі вибрати опцію Simulation і натиснути Start. Симулятор готовий до виконання програми в кроковому режимі. Для виконання наступної команди програми потрібно натиснути на закладку STEP, яка з'явиться справа від закладки HELP вгорі основного вікна симулятора після вибору крокового режиму його роботи;

14. Для виконання програми в автоматичному режимі потрібно вибрати Rate | Extremely Fast simulation rate, а далі вибрати опцію Simulation і натиснути Start;

15. Щоб зупинити виконання програми, потрібно натиснути Simulation | Stop.

Для того, щоб мати змогу контролювати вміст регістрів після виконання симулятором кожної команди, перейти на виконання програми в кроковому режимі роботи. Для цього:

1. В основному вікні симулятора натиснути Rate | Step By Step, а далі вибрати опцію Simulation і натиснути Start. Симулятор готовий до виконання програми в кроковому режимі;

2. Для виконання наступної команди програми потрібно натиснути на закладку STEP, яка з'явиться справа від закладки HELP вгорі основного вікна симулятора після вибору крокового режиму його роботи.

Вміст регістрів контролера, які використовуються при виконанні команд програми, знайти в області регістрів загального призначення та регістрів керування Adress and Name, яка розташована в лівій нижній частині основного вікна симулятора (виділені рожевим кольором). Всі регістри восьмирозрядні.

В процесі виконання програми по зміні кольору комірок видно, вміст яких регістрів змінюється. Забарвлення комірки відповідного розряду регістру помаранчевим кольором означає наявність "1", білим - "0".

В правій нижній частині основного вікна симулятора розташована область внутрішньої пам’яті даних Internal Data SRAM.

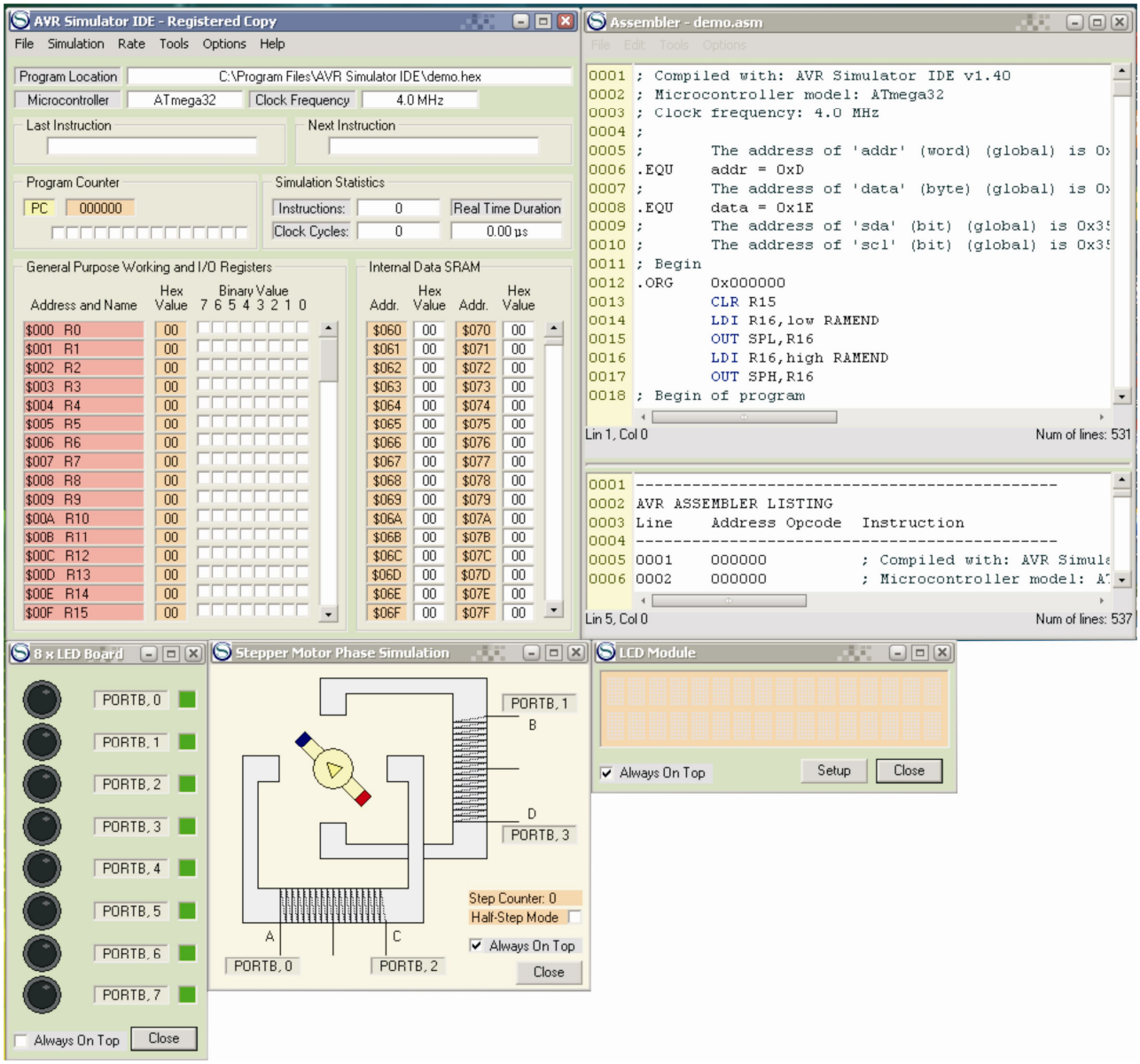

Рис. 5 Вигляд симулятора з завантаженою програмою, вікном LCD Module, вікном з дисплеєм 8xLED Board та вікном Stepper Motor Phase Simulation

Вигляд стимулятора з виведенною поточною інформацією на дисплеї в процесі виконання програми показано на рис. 6.

Завдання

1. Вміст регістрів, значення яких змінюється в процесі виконання команд програми, записати в шістнадцятковому коді в табл.1.

Таблиця 1

Регістр	PC	Ri	TMR0	SREG	SP	ID SRAM	EEAR	EEDR	EECR	PORTx	DDRx
Команда 1											
Команда 2											
Команда n											

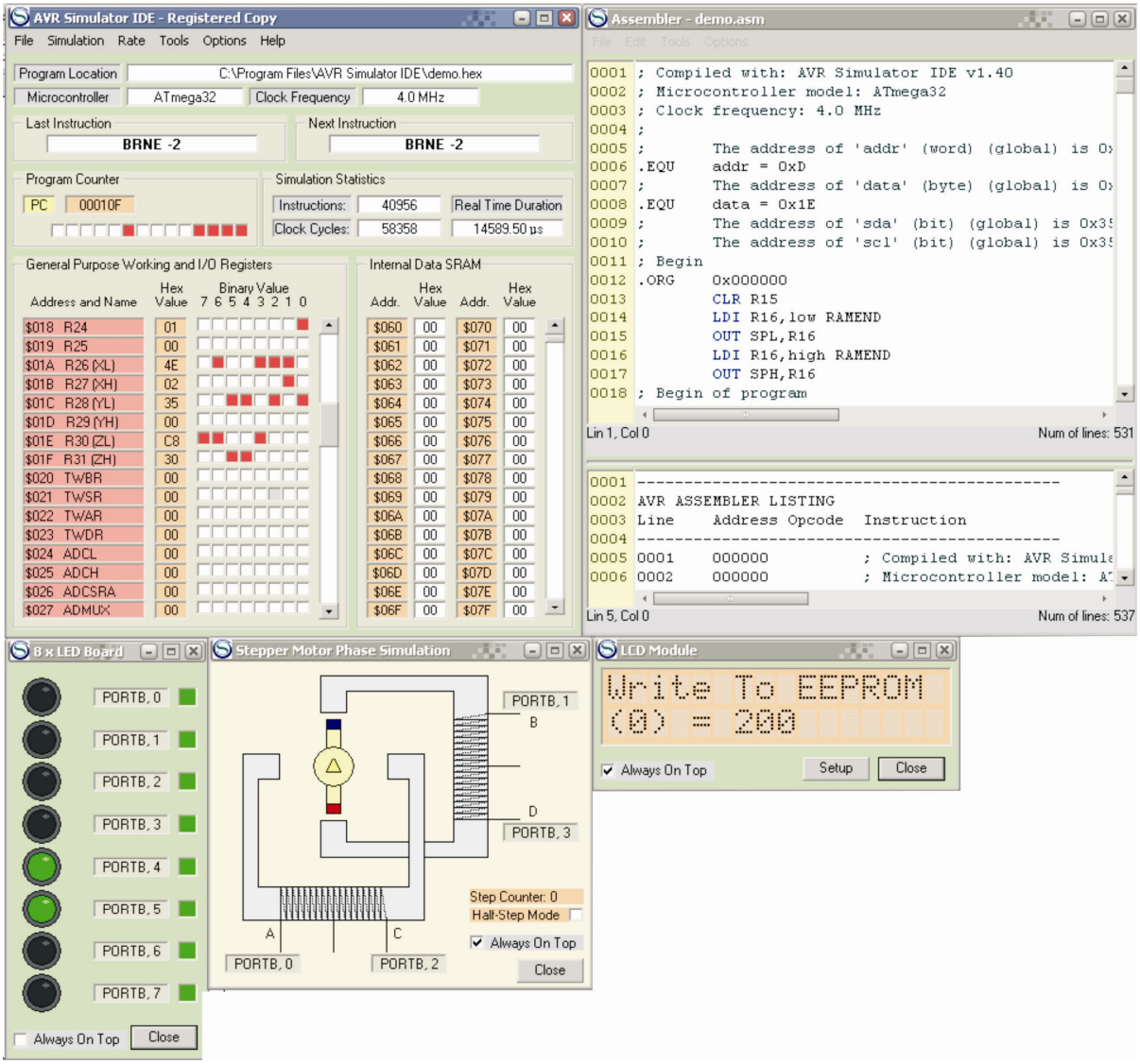

Рис. 6 Вигляд симулятора з виконуваною програмою, станом вікон LCD Module, дисплею 8xLED Board та емулятора уніполярного крокового двигуна Stepper Motor Phase Simulation

З вікна з програмою (рис. 4) вибрати десять команд і за таблицею команд асемблера для AVR - контролера (табл. 1) записати коментар щодо призначення цих команд (див. Приклад, де наведено такий запис для однієї команди).

Приклад

Команда **Виконувана операція (коментар)**

OUT SPL, R16 ; вивести вміст регістра R16 в молодший байт вказівника стеку SPL і т.д.

1. Контрольні запитання

1. Будова контролера Atmega32.
2. Призначення регістрів загального призначення мікроконтролера.
3. Формат та призначення регістрів керування.
4. Як програмуються лінії портів на ввід та на вивід?
5. Формат регістра SREG.
6. Призначення та позначення основних елементів програмної моделі мікроконтролера.

Система команд мікроконтролерів AVR

Система команд AVR мікроконтролерів включає команди арифметичних і логічних операцій, команди передачі даних, команди, що керують послідовністю виконання програми і команди операцій з бітами.

Для зручності написання й аналізу програм всім операціям із системи команд крім двійкового коду зіставлені мнемокоди Ассемблера (символічні позначення операцій), що використовуються при створенні вихідного тексту програми.

Спеціальні програми-транслятори переводять потім символічні позначення в двійкові коди.

Спеціальна директива ассемблера **.device** забезпечує контроль відповідності команд, використовуваних у тексті програми, типу зазначеного процесора.

Під час виконання арифметичних, логічних чи операцій роботи з бітами ALU формує ознаки результату операції, тобто встановлює чи скидає біти в регістрі стану **SREG** (Status Register).

Регістр статусу - SREG - розміщений у просторі I/O за адресою $3F ($5F).

Таблиця 1 – Регістр статусу - SREG

Біти	7	6	5	4	3	2	1	0	
$3F ($5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**REG**
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Bit 7 - I: Global Interrupt Enable - Дозвіл глобального переривання. Біт дозволу глобального переривання для дозволу переривання повинний бути встановлений у стан 1. Керування дозволом конкретного переривання виконується регістрами маски переривання GIMSK і TIMSK. Якщо біт глобального переривання очищений (у стані 0), то жодне з дозволів конкретних переривань, встановлених у регістрах GIMSK і TIMSK, не діє.

Біт I апаратно очищається після переривання і встановлюється для наступного дозволу глобального переривання командою RETI.

Bit 6 - T: Bit Copy Storage - Біт збереження копії. Команди копіювання біта BLD (Bit Load) і BST (Bit STore) використовують біт T, як біт джерело і біт призначення при операціях з бітами. Командою BST біт регістра копіюється до біту T, командою BLD біт T копіюється до регістру.

Bit 5 - H: Half Carry Flag - Прапор напівпереносу. Прапор напівпереносу вказує на напівперенос у ряді арифметичних операцій.

Bit 4 - S: Sign Bit, S = N V - Біт знаку. Біт S завжди знаходиться в стані, обумовленому логічною функцію АБО (OR) між прапором негативного значення N і доповненням до двох прапора переповнення V.

Bit 3 - V: Two's Complement Overflow Flag. Доповнення до двох прапора переповнення. Доповнення до двох прапора V підтримує арифметику доповнення до двох.

Bit 2 - N: Negative Flag – Прапор негативного значення. Прапор негативного значення N вказує на негативний результат ряду арифметичних і логічних операцій.

Bit 1 - Z: Zero Flag – Прапор нульового значення. Прапор нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій.

Bit 0 - C: Carry Flag – Прапор переносу. Ознаки результату операції можуть бути використані в програмі для виконання подальших арифметично-логічних операцій чи команд умовних переходів.

Арифметичні і логічні конструкції

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
ADD	Rd,Rr	Підсумовування без переносу	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Підсумовування з переносом	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
SUB	Rd,Rr	Вирахування без переносу	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Вирахування константи	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Вирахування з переносом	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Вирахування константи з переносом	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Логічне И	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Логічне И с константою	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Логічне АБО	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Логічне АБО з константою	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Логічне що виключає АБО	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	Побітна Інверсія	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Зміна знака (Доп. код)	Rd = $00 - Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	Установити біт (біти) у регістрі	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Скинути біт (біти) у регістрі	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Інкрементувати значення регістра	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	Декрементувати значення регістра	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Перевірка на нуль або заперечність	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Очистити регістр	Rd = 0	Z,C,N,V,S	1
SER	Rd	Установити регістр	Rd = $FF	None	1
ADIW	Rdl,K6	Скласти константу і слово	Rdh:Rdl=Rdh:Rdl+ K6	Z,C,N,V,S	2
SBIW	Rdl,K6	Вичитати константу зі слова	Rdh:Rdl=Rdh:Rdl - K 6	Z,C,N,V,S	2

Інструкції розгалуження

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
RJMP	k	Відносний перехід	PC = PC + k +1	None	2
IJMP	Немає	Непрямий перехід на (Z)	PC = Z	None	2
EIJMP	Немає	Розширений непрямий перехід на (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	None	2
JMP	k	Перехід	PC = k	None	3
RCALL	k	Відносний виклик підпрограми	STACK=PC+1, PC=PC + k+ 1	None	3/4*
ICALL	Немає	Непрямий виклик (Z)	STACK = PC+1, PC = Z	None	3/4*
EICALL	Немає	Розширений непрямий виклик (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND	None	4*
RET	Немає	Повернення з підпрограми	PC = STACK	None	4/5*
RETI	Немає	Повернення з переривання	PC = STACK	I	4/5*
CPSE	Rd,Rr	Порівняти, пропустити якщо рівні	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Порівняти	Rd -Rr	Z,C,N,V,H,S	1
CPC	Rd,Rr	Порівняти з переносом	Rd - Rr - C	Z,C,N,V,H,S	1
CPI	Rd,K8	Порівняти з константою	Rd - K	Z,C,N,V,H,S	1
SBRC	Rr,b	Пропустити якщо біт у регістрі очищений	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3

SBRS	Rr,b	Пропустити якщо біт у регістрі встановлений	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Пропустити якщо біт у порту очищений	if(I/O(P,b)==0) PC=PC + 2 or 3	None	1/2/3
SBIS	P,b	Пропустити якщо біт у порту встановлений	if(I/O(P,b)==1) PC=PC + 2 or 3	None	1/2/3
BRBC	s,k	Перейти якщо прапор у SREG очищений	if(SREG(s)==0) PC=PC+ k + 1	None	1/2
BRBS	s,k	Перейти якщо прапор у SREG установлений	if(SREG(s)==1) PC = PC+k+ 1	None	1/2
BREQ	k	Перейти якщо дорівнює	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Перейти якщо не дорівнює	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Перейти якщо перенос установлений	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Перейти якщо перенос очищений	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Перейти якщо дорівнює чи більше	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Перейти якщо менше	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Перейти якщо мінус	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Перейти якщо плюс	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Перейти якщо більше чи дорівнює (зі знаком)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Перейти якщо менше (зі знаком)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Перейти якщо прапор внутрішнього переносу встановлений	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Перейти якщо прапор внутрішнього переносу очищений	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Перейти якщо прапор T встановлений	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Перейти якщо прапор T очищений	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Перейти якщо прапор переповнення встановлений	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Перейти якщо прапор переповнення очищений	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Перейти якщо переривання дозволені	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Перейти якщо переривання заборонені	if(I==0) PC = PC + k + 1	None	1/2

Виконувати арифметико-логічні операції й операції читання безпосередньо над змістом комірок пам'яті не можна. Не можна також записати константу чи очистити вміст комірки пам'яті.

Система команд AVR дозволяє лише виконувати операції обміну даними між осередками SRAM і регістрами загального призначення.

Перевагами системи команд можна вважати різноманітні режими адресації комірок пам'яті.

Усі регістри введення/виведення можуть зчитуватися і записуватися через регістри загального призначення за допомогою команд IN, OUT.

Безпосередня установка і скидання окремих розрядів цих регістрів виконується командами SBI і CBI. Команди умовних переходів у якості своїх операндів можуть мати як біти-ознаки результату операції, так і окремі розряди регістрів введення/виведення, що побітно адресуються.

Інструкції передачі даних

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
MOV	Rd,Rr	Скопіювати регістр	Rd = Rr	None	1
LDI	Rd,K8	Завантажити константу	Rd = K	None	1
LDS	Rd,k	Пряме завантаження	Rd = (k)	None	2*
LD	Rd,X	Непряме завантаження	Rd = (X)	None	2*
LD	Rd,X+	Непряме завантаження з пост-інкрементом	Rd=(X), X=X+1	None	2*
LD	Rd,-X	Непряме завантаження з пре-декрементом	X=X-1, Rd=(X)	None	2*
LD	Rd,Y	Непряме завантаження	Rd = (Y)	None	2*
LD	Rd,Y+	Непряме завантаження з пост-інкрементом	Rd=(Y),Y=Y+1	None	2*
LD	Rd,-Y	Непряме завантаження з пре-декрементом	Y=Y-1,Rd= (Y)	None	2*
LDD	Rd,Y+q	Непряме завантаження з заміщенням	Rd = (Y+q)	None	2*
LD	Rd,Z	Непряме завантаження	Rd = (Z)	None	2*
LD	Rd,Z+	Непряме завантаження з пост-інкрементом	Rd= (Z), Z=Z+1	None	2*
LD	Rd,-Z	Непряме завантаження з пре-декрементом	Z=Z-1, Rd = (Z)	None	2*
LDD	Rd,Z+q	Непряме завантаження з заміщенням	Rd = (Z+q)	None	2*
STS	k,Rr	Пряме збереження	(k) = Rr	None	2*
ST	X,Rr	Непряме збереження	(X) = Rr	None	2*
ST	X+,Rr	Непряме збереження з пост-інкрементом	(X)=Rr, X=X+1	None	2*
ST	-X,Rr	Непряме збереження з пре-декрементом	X=X-1, (X)=Rr	None	2*
ST	Y,Rr	Непряме збереження	(Y) = Rr	None	2*
ST	Y+,Rr	Непряме збереження з пост-інкрементом	(Y)=Rr, Y=Y+1	None	2
ST	-Y,Rr	Непряме збереження з пре-декрементом	Y=Y-1, (Y)= Rr	None	2
ST	Y+q,Rr	Непряме збереження з заміщенням	(Y+q) = Rr	None	2
ST	Z,Rr	Непряме збереження	(Z) = Rr	None	2
ST	Z+,Rr	Непряме збереження з пост-інкрементом	(Z)= Rr, Z=Z+1	None	2
ST	-Z,Rr	Непряме збереження з пре-декрементом	Z=Z-1, (Z) = Rr	None	2
ST	Z+q,Rr	Непряме збереження з заміщенням	(Z+q) = Rr	None	2
LPM	Нет	Завантаження з програмної пам'яті	R0 = (Z)	None	3
LPM	Rd,Z	Завантаження з програмної пам'яті	Rd = (Z)	None	3
LPM	Rd,Z+	Завантаження з програмної пам'яті з пост-інкрементом	Rd=(Z), Z=Z+1	None	3
SPM	Нет	Збереження в програмній пам'яті	(Z) = R1:R0	None	-
IN	Rd,P	Читання порту	Rd = P	None	1
OUT	P,Rr	Запис у порт	P = Rr	None	1
PUSH	Rr	Занесення регістра в стек	STACK = Rr	None	2
POP	Rd	Витяг регістра зі стека	Rd = STACK	None	2

Інструкції роботи з бітами

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
LSL	Rd	Логічний зсув вліво	Rd(n+1)=Rd(n),Rd(0)=0,C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	Логічне зрушення вправо	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Циклічне зрушення вліво через C	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	Z,C,N,V,H,S	1
ROR	Rd	Циклічне зрушення вправо через C	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Арифметичне зрушення вправо	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Перестановка тетрад	Rd(3..0)=Rd(7..4),Rd(7..4)=Rd(3..0)	None	1
BSET	s	Установка прапора	SREG(s) = 1	SREG(s)	1
BCLR	s	Очищення прапора	SREG(s) = 0	SREG(s)	1
SBI	P,b	Установити біт у порту	I/O(P,b) = 1	None	2
CBI	P,b	Очистити біт у порту	I/O(P,b) = 0	None	2
BST	Rr,b	Зберегти біт з регістра в T	T = Rr(b)	T	1
BLD	Rd,b	Завантажити біт з T у регістр	Rd(b) = T	None	1
SEC	Hi	Установити прапор переносу	C =1	C	1
CLC	Hi	Очистити прапор переносу	C = 0	C	1
SEN	Hi	Установити прапор негативного числа	N = 1	N	1
CLN	Hi	Очистити прапор негативного числа	N = 0	N	1
SEZ	Hi	Встановити прапор нуля	Z = 1	Z	1
CLZ	Hi	Очистити прапор нуля	Z = 0	Z	1
SEI	Hi	Встановити прапор переривань	I = 1	I	1
CLI	Hi	Очистити прапор переривань	I = 0	I	1
SES	Hi	Установити прапор числа зі знаком	S = 1	S	1
CLN	Hi	Очистити прапор числа зі знаком	S = 0	S	1
SEV	Hi	Установити прапор переповнення	V = 1	V	1
CLV	Hi	Очистити прапор переповнення	V = 0	V	1
SET	Hi	Установити прапор T	T = 1	T	1
CLT	Hi	Очистити прапор T	T = 0	T	1
SEH	Hi	Установити прапор внутрішнього переносу	H = 1	H	1
CLH	Hi	Очистити прапор внутрішнього переносу	H = 0	H	1
NOP	Hi	Немає операції	Hi	None	1
SLEEP	Hi	Спати (зменшити енергоспоживання)	Дивитися опис інструкції	None	1
WDR	Hi	Скидання сторожового таймера	Дивитися опис інструкції	None	1

Асемблер не розрізняє регістр символів. Операнди можуть бути таких видів:

- Rd: результуючий і вихідний регістр;
- Rr: вихідний регістр;

- b: константа (3 біти), може бути константний вираз;
- s: константа (3 біти), може бути константний вираз;
- P: константа (5-6 біт), може бути константний вираз;
- K6: константа (6 біт), може бути константний вираз;
- K8: константа (8 біт), може бути константний вираз;
- k: константа, може бути константний вираз;
- q: константа (6 біт), може бути константний вираз;
- Rdl: R24, R26, R28, R30 для інструкцій ADIW і SBIW;
- X,Y,Z: регістри непрямої адресації (X=R27:R26, Y=R29:R28, Z=R31:R30).

Література

1. Евстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL, – [5-е изд., стер.] / Евстифеев А. В. – М.: Издательский дом «Додэка-XXI», 2008. 560 с.

2. Програмування мікроконтролерів систем автоматики: конспект лекцій для студентів базового напряму 050201 “Системна інженерія” / Укл.: А.Г. Павельчак, В.В. Самотий, Ю.В. Яцук – Львів: Львівська політехніка. – 2012. – 143 с.