

Лабораторна робота №3

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR

Мета роботи: вивчення способів програмування мікроконтролера AT90S2313, програмних засобів (AVR Studio) для відлагодження програмного забезпечення.

Теоретичні відомості

1. Способи програмування енергонезалежної пам'яті AVR

У процесі програмування AVR-мікроконтролерів виконують такі операції по стиранню, читанню і запису різних елементів енергонезалежної пам'яті кристала:

- операція "Chip erase" (стирання кристала);
- читання/запис FLASH-пам'яті програм;
- читання/запис EEPROM пам'яті даних;
- читання/запис конфігураційних FUSE-битий;
- читання/запис LOCK-біт захисту програмної інформації;
- читання SYGNATURE-біт ідентифікації кристала.

AVR-мікроконтролери поставляються зі стертими FLASH і EEPROM блоками пам'яті (зміст всіх комірок = \$FF), готовими до програмування.

У табл. 3.1 перераховані можливі способи програмування елементів енергонезалежної пам'яті AVR.

Таблиця 5.1 – Способи програмування AVR

	FLASH	LOCK-біти	FUSE-біти	EEPROM
Паралельне програмування	+	+	+	+
Послідовне програмування	+	+	+	+
Само-програмування	+	–	–	+

Паралельне програмування вимагає використання додаткового джерела підвищеної напруги (12 В), використовує велике число виводів мікроконтролера і виконується на спеціальних програматорах. Таке програмування зручне, коли при масовому виробництві необхідно "прошивати" велику кількість кристалів.

Послідовне програмування (Successive programming) не вимагає додаткового джерела живлення і може виконуватися безпосередньо в мікропроцесорній системі (In System Programming) через послідовний SPI-інтерфейс, що використовує всього чотири виводи AVR-мікроконтролера. Можливість внутрісистемного програмування є одним з найважливіших переваг AVR, тому що дозволяє значно спростити та здешевити процес розробки і модернізації програмного забезпечення. Паралельний і послідовний способи програмування припускають використання зовнішнього програмуючого процесора. EEPROM пам'ять може також програмуватися самим AVR під керуванням програми (самопрограмування).

LOCK-біти програмуються як паралельно, так і послідовно. FUSE-біти в молодших моделях AVR можуть програмуватися тільки послідовно, а в старших - і паралельно, і послідовно. SYGNATURE-байти доступні для читання при будь-якому способі програмування, якщо кристал не засекречений LOCK-бітами.

Операція "Chip erase" виконується в обох режимах програмування. Під час її стираються всі комірки FLASH і EEPROM пам'яті, а також LOCK-біти. Причому LOCK-біти стираються тільки після того, як буде очищена вся пам'ять програм. На стан FUSE біт операція "Chip erase" не робить впливу.

FLASH і EEPROM блоки пам'яті програмуються байт за байтом у кожному з режимів програмування.

Для EEPROM пам'яті в режимі послідовного програмування автоматично забезпечується цикл стирання. Таким чином, існує можливість побайтного перезапису окремих комірок EEPROM. Якщо ж потрібно змінити зміст яких-небудь комірок FLASH пам'яті, то необхідно виконувати операцію "Chip erase", що у всіх кристалів стирає не тільки цілком усю FLASH, але і вміст EEPROM.

LOCK-біти (LB1, LB2) призначені для захисту програмної інформації, що міститься в FLASH-пам'яті. Можливі режими захисту перераховані в табл. 5.2. Запрограмувавши біти захисту, стерти їх можна лише під час очищення FLASH-пам'яті (операція "Chip erase"), що знищує і всю програму.

Таблиця 3.2 - Режими захисту програмної інформації AVR

Режим	LB1	LB2	
1	1	1	Захист відсутній
2	0	1	Заборона програмування Flash
3	0	0	Заборона як програмування, так і читання Flash

FUSE-біти дозволяють задавати деякі конфігураційні особливості мікроконтролера. Склад FUSE біт кожного конкретного типу AVR обумовлений особливостями побудови вузлів скидання, тактування і програмування кристала.

Об'єктний файл, створюваний програмою WAVRASM, використовується надалі як вхідний для програми-відлагоджувача AVRSTUDIO і має спеціальний формат.

2. Створення програм

Процес написання програм для МК AVR, як і для будь-яких інших МК, складається з декількох етапів:

- підготовка вихідного тексту програми будь-якою мовою програмування;
- компіляція програми;
- налагодження й тестування програми;
- остаточне програмування й підготовка до серійного виробництва.

Мікропрограма пристрою повинна бути написана однією з мов програмування.

На даний час для МК AVR існує декілька мов програмування, а також різних засобів підтримки розробки, що використовують одну мову, але різняться за функціональністю.

На кожному з етапів необхідне застосування спеціальних програмних й апаратних засобів. Варто відзначити, що базовий набір програмного забезпечення (компілятор асемблера, ПЗ для програмування) поширюється фірмою Atmel безкоштовно. Однак за досить довгий період часу, що пройшов з моменту появи цих МК, з'явилася велика кількість програмного забезпечення сторонніх виробників.

Вибір компілятора асемблера є найменш принциповим питанням, оскільки сам процес компіляції (перетворення вихідного тексту програми в машинний код) виконується досить однозначно. Власне, всі відмінності між асемблерами полягають у можливостях процесора, що обробляє макрокоманди, наявності текстового редактора або середовища розробки й типу операційної системи, що підтримується.

Досить вдалим вибором при розробці програмного забезпечення для МК сімейства AVR може служити інтегроване середовище розробки AVR Studio фірми Atmel, що містить у собі текстовий редактор з підсвічуванням синтаксису, компілятор асемблера, симулятор, налагоджувач й інтерфейс із

апаратними емуляторами. Програма розрахована на роботу під керуванням операційних систем Windows 9x, 2000, XP, Vista. До недоліків AVR Studio можна віднести деяку нестабільність роботи налагоджувача, а також неповну симуляцію периферійних пристроїв (зокрема, відсутня симуляція АЦП). До плюсів належить, у першу чергу, підтримка практично всіх МК сімейства AVR.

3. Створення і трансляція проекту

Для створення нового проекту необхідно вибрати у головному меню **Project (Проект)|New... (Новий)**. У вікні, що з'явилося, в рядку **Project name (Ім'я проекту)** треба ввести ім'я нового проекту, наприклад "new". Далі, натиснувши лівою кнопкою мишки на рядку **AVR Assembler**, вибрати тип проекту (рисунок 3.1). Натиснути кнопку **OK**.

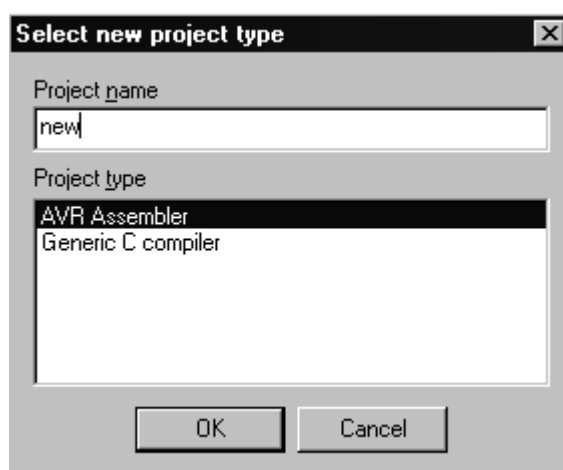


Рисунок 3.1 – Вікно вибору типу проекту

У вікні, що утворилось, необхідно натиснути правою кнопкою мишки на рядки **Assembler Files**; у меню, що з'явилося (рисунок 3.2) натиснути лівою кнопкою мишки на рядку **Add File...** (Додати файл).

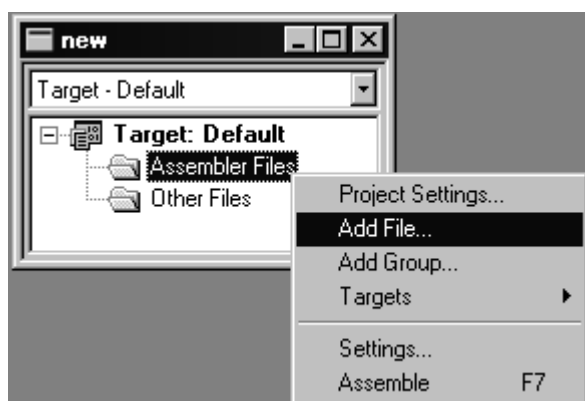


Рисунок 3.2 – Список файлів проекту

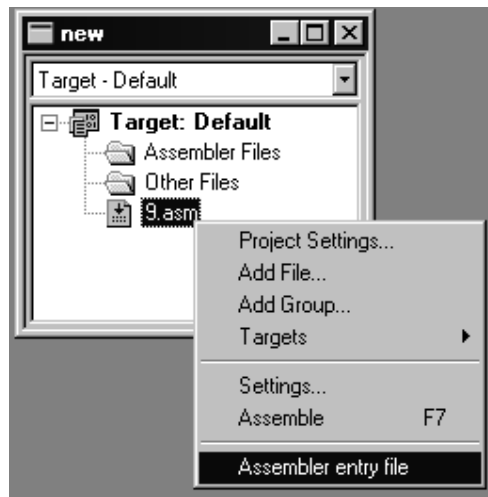


Рисунок3.3 – Вибір основного файлу

У вікні **Open file** необхідно створити порожній текстовий файл, змінити ім'я файлу майбутньої програми та надати йому розширення **.asm**. Таким чином буде створений порожній файл робочої програми з розширенням **.asm**. Якщо у вас вже існує файл програми (з розширенням **.asm**), то у вікні **Open file** треба вибрати цей файл.

Для того, щоб зробити створений файл основним, необхідно натиснути правою кнопкою мишки на імені створеного файлу в вікні проекту, а потім у меню, що з'явилося, натиснути лівою кнопкою мишки на рядку **Assembler entry file** (рис. 3.3). Тепер необхідно відкрити створений файл, двічі натиснувши на ньому лівою кнопкою мишки у вікні проекту. У вікні, що з'явилося, можна набирати текст програми.

Після введення тексту програми необхідно натиснути клавішу **F7**, щоб провести компіляцію програми. Після цього з'являється вікно (рис. 3.4), в якому міститься інформація про правильність написаної програми. При виявленні помилок повідомляється про всі рядки, в яких виявлено помилки. Натиснувши на рядок, в якому повідомляється про помилку, потрапимо в те місце програми, де виявлено помилку.

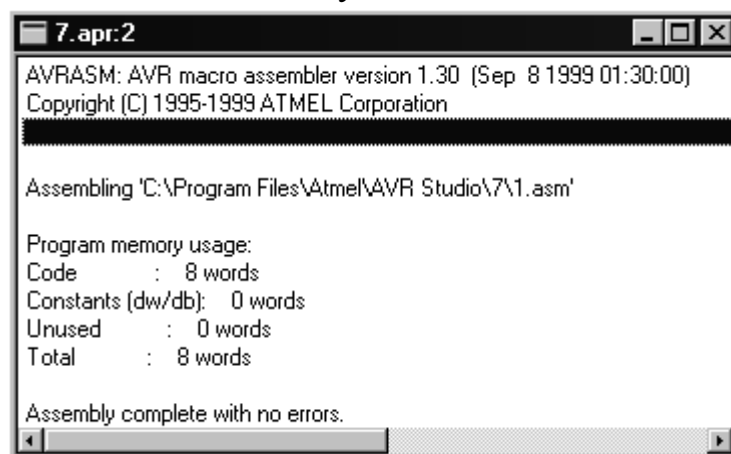


Рисунок 3.4 – Вікно результатів компіляції коду програми

Після завершення компіляції без помилок, тобто в вікні (див. рис. 3.4) нижній рядок буде мати вигляд **Assembly complete with no errors** (програма відкомпільована без помилок), можна приступити до відлагодження роботи програми.

Після натискання клавіші **F11** почнеться процес відлагодження. Для перегляду результатів роботи програми з меню **View** (Вид) треба викликати вікна **Registers** (Регістри), **Processor** (Процесор), **New Memory View** (Новий вид пам'яті). Вікно **Registers** відображає поточний зміст регістрів загального призначення **R0-R31**. Вікно **Processor** відображає поточне значення регістрових пар (X, Y, Z), значення програмного лічильника, значення покажчика стека та всіх восьми прапорців регістра стану.

Вікно **New Memory View**, залежно від обраного режиму, відображає поточний зміст пам'яті даних, пам'яті програм, пам'яті введення-виведення, пам'яті EEPROM. Робоче вікно програми набуде вигляду, показаного на рисунку 3.5.

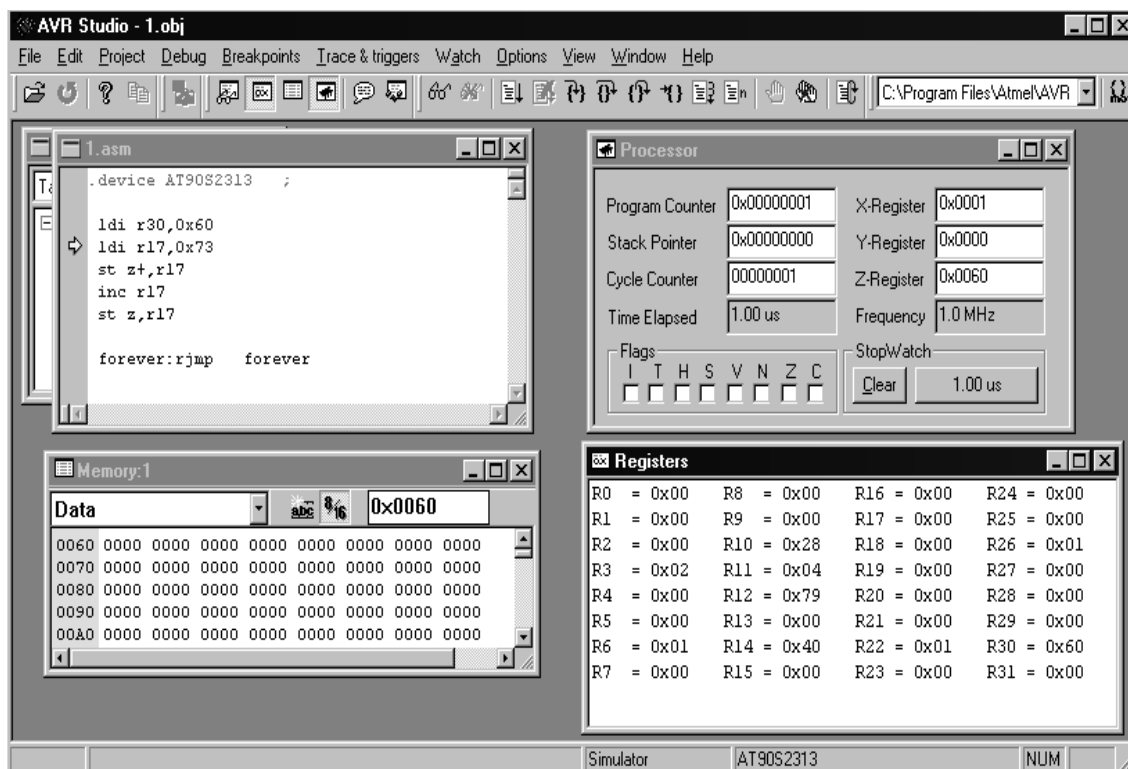


Рисунок 3.5 – Робоче вікно програми

При натисканні клавіші **F11** (**Trace into**) відбувається виконання поточних команд, що відображається відповідними змінами у вікнах.

Компілятор працює з вихідними файлами, що містять інструкції, мітки і директиви. Інструкції та директиви, як правило, мають один чи декілька операндів.

Примітка. Рядок коду не повинен бути довшим 120 символів. Будь-який рядок може починатися з мітки, що є набором символів та закінчується двокрапкою. Мітки використовуються для вказання місця, у яке передається керування при переходах, а також для задання імен змінних.

Вхідний рядок може мати одну з чотирьох форм:

- [мітка:] директива [операнди] [Коментар];
- [мітка:] інструкція [операнди] [Коментар];
- коментар;
- порожній рядок.

Коментар починається після крапки з комою (;) і до кінця рядка ігнорується компілятором.

Після завантаження **AVR STUDIO 4** з'явиться діалогове вікно. Для створення нового проекту треба натиснути [**Create New Project**] (Створити новий проект).

Другим етапом є встановлення налаштувань проекту. На даному етапі задаються розширення, назва й адреса файлу, що створюється (рис. 3.6).

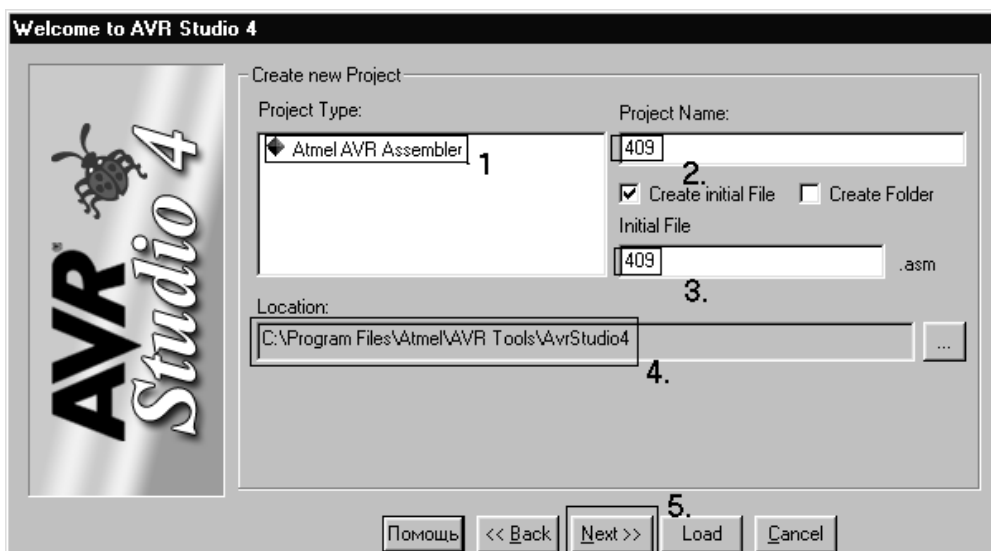


Рисунок 3.6 – Встановлення налаштувань проекту

Для цього необхідно виконати такі дії:

- у вікні **Project type** необхідно обрати Atmel AVR Assembler;
- у вікні **Project Name** необхідно ввести ім'я проекту та для створення нового файлу встановити прапорець **Create initial File**;

- вказати шлях збереження файлів проекту.
Третім етапом є вибір системи налагодження.

Програмне забезпечення **AVR STUDIO 4** має у своєму складі програми налагодження для широкого спектру задач (рис. 2.7), що передбачає:

- вибір вбудованого симулятора-налагоджувача;
- вибір пристрою, для якого створюється програма.

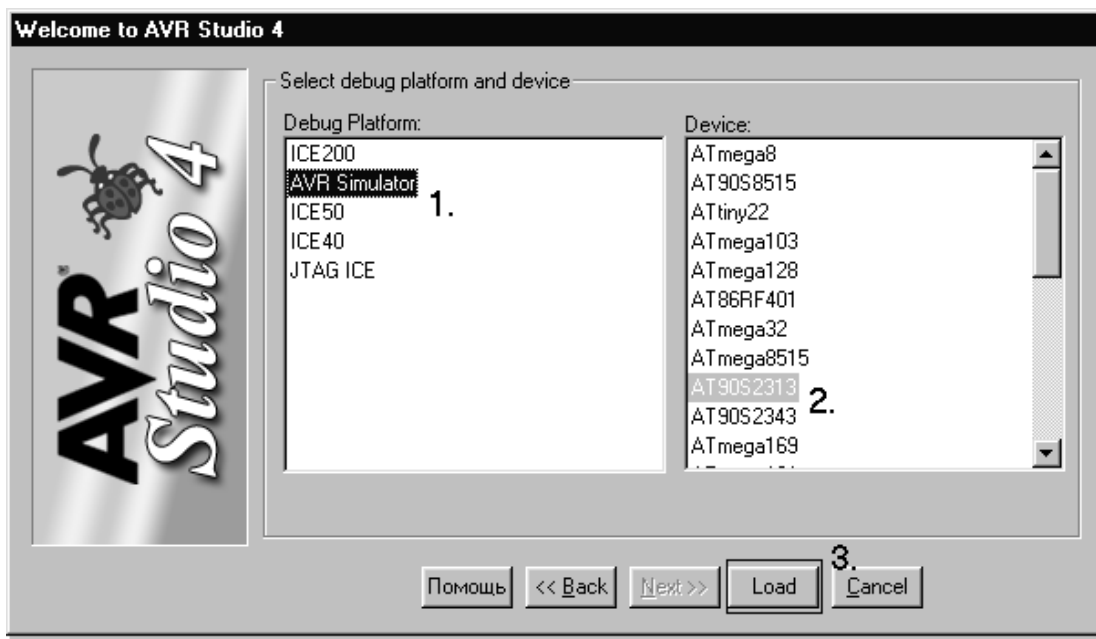


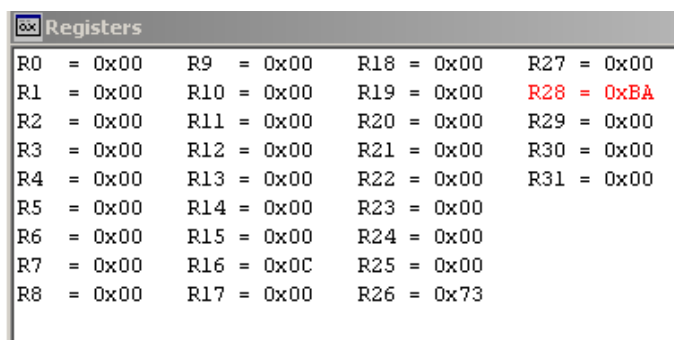
Рисунок 3.7 – Вибір симулятора та мікроконтролера

У процесі налагодження можна вибрати пункт меню **Debug -> Run To Cursor**. При виборі цього пункту код, що виконується, буде виконуватися до досягнення команди, позначеної курсором. При цьому, якщо налагоджувач виявляє точку зупинки, що встановлена раніше положення курсору, то зупинка буде виконана тільки у випадку його дозволу у вікні **Debug Option**, в іншому випадку виконання не припиняється. Якщо команда, позначена курсором, не досягається, налагоджувач продовжує виконувати код програми доти, поки виконання не буде перервано користувачем. Оскільки режим **Run To Cursor** залежить від позиції курсору, він доступний тільки при активному вікні вихідного тексту.

Для зупинки виконання програми користувачем служить команда **Break**. У стані зупинки ця команда недоступна. При налагодженні з використанням крапок зупинки, чи якщо адреса зупинки зазначена курсором у вікні вихідного тексту, модифікація інформації у всіх вікнах відбувається тільки при досягненні зупинки (чи при припиненні виконання програми користувачем).

Пункт меню **Debug -> Reset** виконує скидання мікроконтролера. Якщо програма при цьому виконується, то її виконання буде зупинено. Після скидання інформація у всіх вікнах модифікується. Для спостереження за роботою програми можна відкрити кілька вікон, що відображають стан різних вузлів мікроконтролера. Вікна відкриваються натисканням відповідних кнопок на панелі чи інструментів при виборі відповідного пункту меню **View**.

Файл реєстрів мікроконтролера AVR відображається у вікні **Registers** (рис. 3.8). Якщо в процесі виконання програми значення якого-небудь реєстра зміниться, то цей реєстр буде виділений червоним кольором. При цьому, якщо в наступному циклі значення реєстра залишиться незмінним, то кольорове виділення буде зняте. Також кольорове виділення реалізоване у вікнах пристроїв введення/виведення, пам'яті і змінних.



Registers			
R0 = 0x00	R9 = 0x00	R18 = 0x00	R27 = 0x00
R1 = 0x00	R10 = 0x00	R19 = 0x00	R28 = 0xBA
R2 = 0x00	R11 = 0x00	R20 = 0x00	R29 = 0x00
R3 = 0x00	R12 = 0x00	R21 = 0x00	R30 = 0x00
R4 = 0x00	R13 = 0x00	R22 = 0x00	R31 = 0x00
R5 = 0x00	R14 = 0x00	R23 = 0x00	
R6 = 0x00	R15 = 0x00	R24 = 0x00	
R7 = 0x00	R16 = 0x0C	R25 = 0x00	
R8 = 0x00	R17 = 0x00	R26 = 0x73	

Рисунок 3.8 – Вікно стану реєстрів

Перегляд комірок пам'яті програм, пам'яті даних, EEPROM і реєстрів портів введення/виведення в ході виконання програми можливо також за допомогою діалогового вікна **Memory**. Падаюче меню діалогового вікна дозволяє вибрати один з чотирьох масивів комірок пам'яті: *Data*, *I/O*, *EEPROM*, *Program Memory*. Для одночасного перегляду декількох областей вікно **Memory** може бути відкрито кілька разів. Інформація в діалоговому вікні може бути представлена у виді байтів чи у вигляді слів у шістнадцятиричній системі числення, а також у вигляді ASCII-символів.

У процесі налагодження користувач може ініціалізувати внутрішню ПЗП (EEPROM) мікроконтролера (наприклад, даними, що містяться в отриманому при трансляції файлі *.vpr*) чи зберегти вміст ПЗП і EEPROM у виді файлів у форматі Intel Hex. Для цього служить пункт меню **File -> Up/Download Memory**.

Для внесення змін у програму в процесі налагодження необхідно редагувати її вихідний текст. При спробі запуску симулятора на виконання

програми після редагування на екрані з'являється вікно, що повідомляє про зміну програми і необхідності її компіляції.

При закритті проекту зберігаються всі його настройки. Під час наступного завантаження настройки будуть автоматично відновлені.

Завдання до лабораторної роботи

Створити проект з ім'ям група_N.obj, де N – номер варіанту за списком у журналі. В проект занести програму на мові Assembler з лабораторної роботи №2. Виконати симуляцію роботи мікроконтролера AT90S2313 в AVR STUDIO. Перевірити вірність виконання програми. При необхідності внесіть корекцію в програму.

Зміст звіту

Текст програми з поясненнями.

Зміст Program memory в 16 річних кодах.

Зміст даних DATA (на початку і по закінченні програми).

Зміст REGISTERS (на початку і по закінченні програми).

Зміст FLAGC (на початку і по закінченні програми).

Контрольні запитання

1. Які Ви знаєте програмні засоби для налагодження програмного забезпечення?
2. Перерахуйте операції для стирання, читання і запису різних елементів енергонезалежної пам'яті кристала мікроконтролера.
3. Які Ви знаєте способи програмування мікроконтролерів? Наведіть їхні переваги та недоліки.
4. Як створити проект за допомогою програмного симулятора?
5. Які вихідні формати підтримує AVR Studio?