ЛАБОРАТОРНА РОБОТА №1

ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ AVR STUDIO

Ціль: навчитися використати для написання програм інтегроване середовище розробки AVR Studio IDE (Integrated Development Environment).

Завдання: скомпілювати й налагодити програму в середовищі програмування AVR Studio.

ЗАГАЛЬНІ ВІДОМОСТІ

AVR Studio – професійне інтегроване середовище розробки, призначене для написання й налагодження прикладних програм для AVR мікропроцесорів у середовищі Microsoft Windows.

Початок роботи

При написанні програмного забезпечення (ПО) для мікроконтролера (МК) необхідно виконати стандартну послідовність дій:

- ✓ створення проекту;
- ✓ написання програми;
- ✓ зборка проекту;
- ✓ налагодження:
  - ➢ виправлення синтаксичних помилок;
  - ➢ симуляція роботи програми на ПК;
  - ➢ виправлення логічних помилок;
- ✓ програмування мікроконтролера (МК);
- ✓ перевірка роботи програми на реальній електронній платі.

Написання програми процес ітеративний, тобто при необхідності деякий набір дій прийде повторювати кілька разів до одержання задовільного результату.

При написанні ПО розроблювач може зштовхнутися із двома типами помилок: апаратні (помилка або несправність у принциповій схемі пристрою) і програмні (помилки в тексті ПО).

Програмні помилки при розробці ПО підрозділяються в такий спосіб:

- синтаксичні помилки;
- логічні помилки.

Синтаксичні помилки – помилки в тексті програми, які знаходить компілятор. Виправляються швидко й безболісно в більшості випадків. Програма при цьому не може бути скомпільована й запущена.

Логічні помилки – помилки в логіку роботи програми. Це той тип помилок при

якому ПО компілюється й може бути запрограмоване в МК, але воно працює не вірно (не тому що замислювалося). Ці помилки перебувають і усуваються за допомогою різноманітних допоміжних, отладочных засобів (симулятор, висновок отладочной інформації на консоль, апаратний отладчик і ін.).

СТВОРЕННЯ НОВОГО ПРОЕКТУ

При запуску AVR Studio відкривається стартова сторінка (Start page) див. Рис. 1.

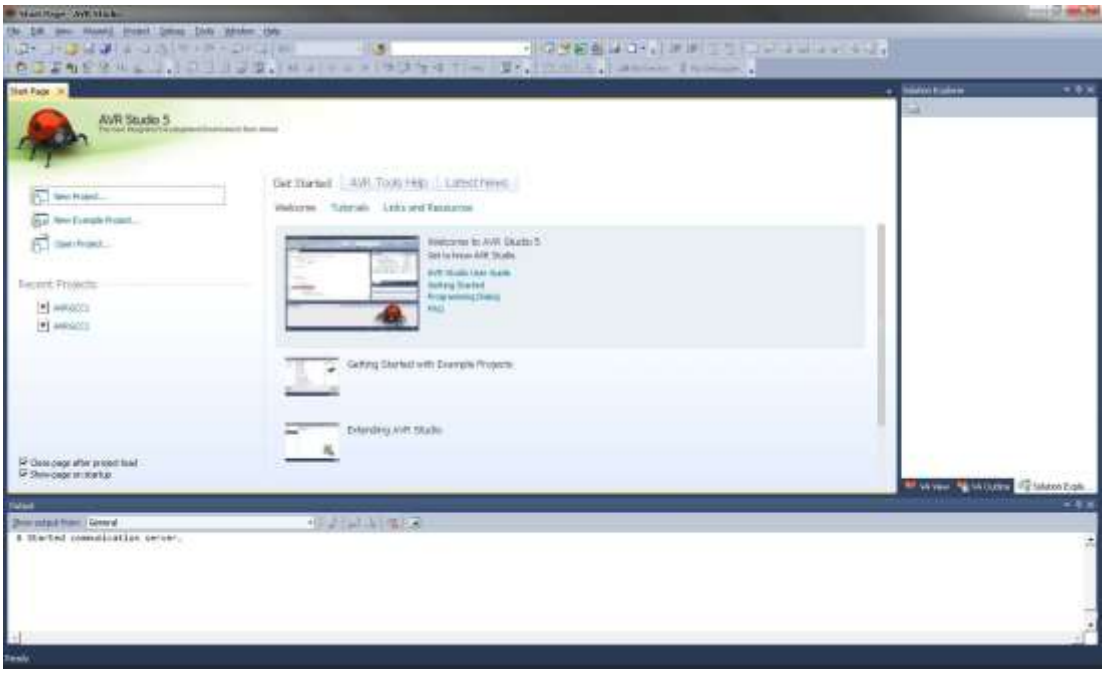

Рис. 1. Стартова сторінка AVR Studio

Дана сторінка дозволяє почати створювати новий проект (New Project),

відкрити проект із яким працювали раніше (Open Project), подивитися приклади проектів (New Example Projects), а також одержати різноманітну довідкову й допоміжну інформацію.

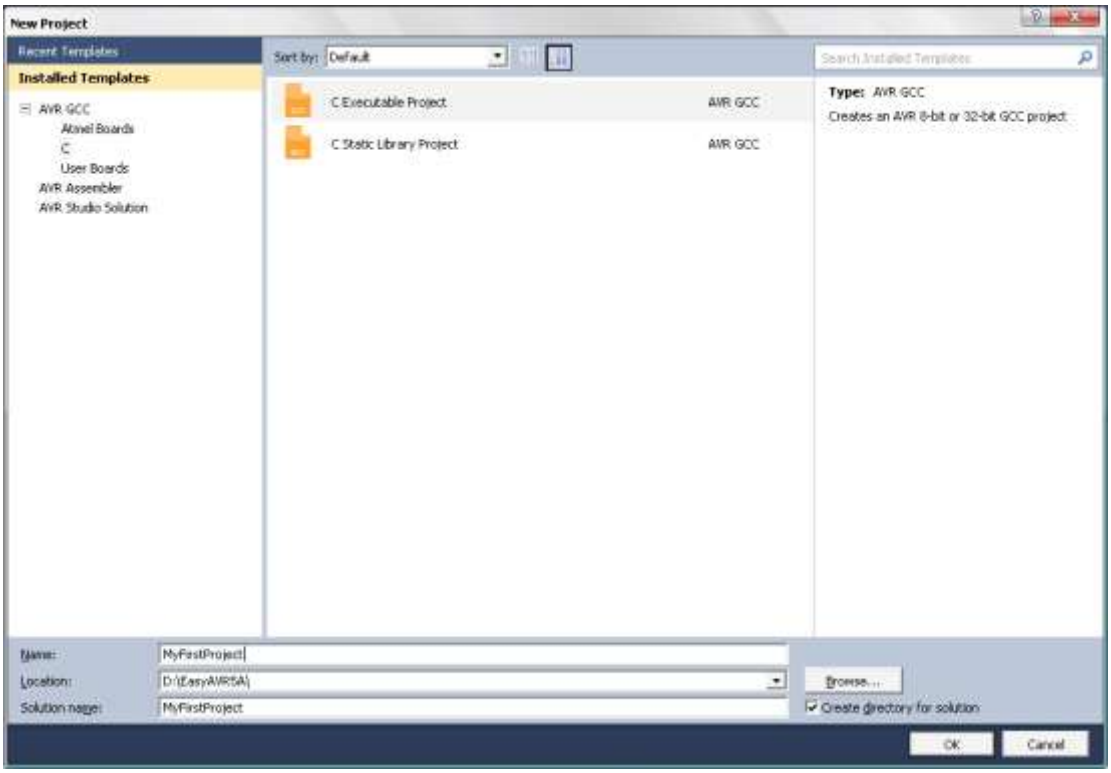

Рис. 2. Створення нового проекту

У вкладці встановлені шаблони (Installed Templates) вибираємо «З». А в допоміжному вікні праворуч - виконується программа, що, мовою З (З Executable Project).

У поле Name уводимо ім'я проекту. Ім'я проекту не повинне містити російських букв і пробілів. Поле Location містить шлях куди буде збережений проект (буде створена окрема папка при встановленому прапорі (Create directory for solution)).

Далі відкривається вікно вибору МК для якого пишеться ПО (див. Рис. 3).

У всіх лабораторних роботах ми будемо вивчати МК ATmega16 - вибираємо МК.

Після всіх перерахованих дій проект створений і можна приступати до написання програми.

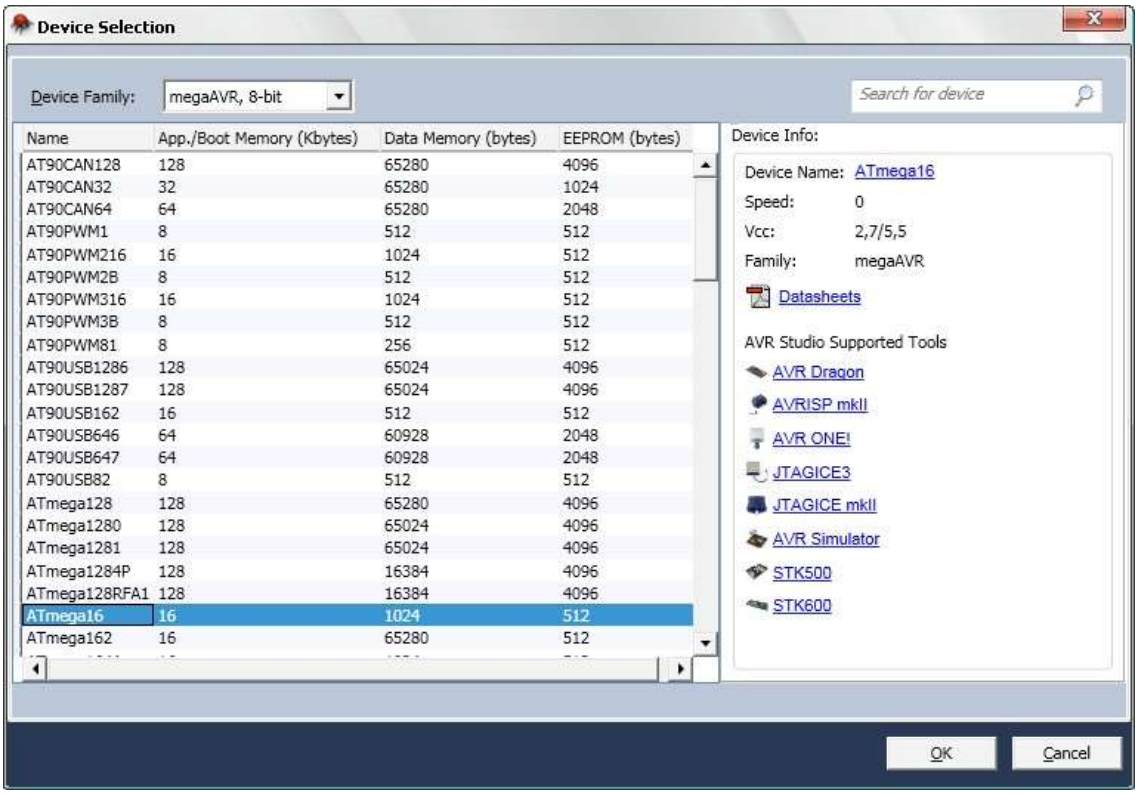

Рис. 3. Вікно вибору МК

ВВЕДЕННЯ ТЕКСТУ ПРОГРАМИ

Основне вікно розробки програми представлено на Рис. 4.

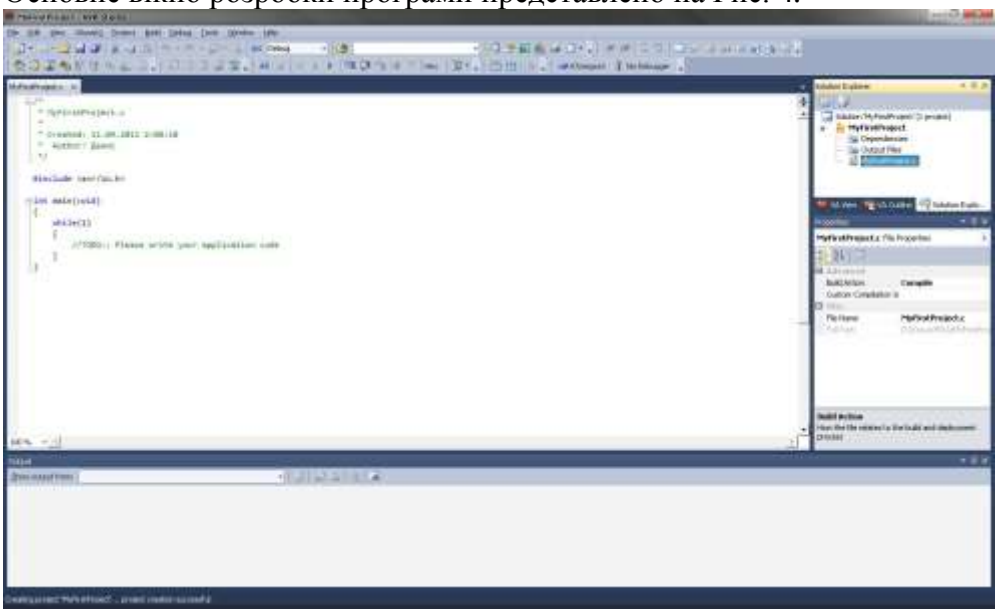

Рис. 4. Основне вікно розробки програми

У верхній частині вікна розташований - рядок меню й різні панелі інструментів (Toolbars).

У центрі вікна розміщене вікно для редагування/перегляду файлів проекту. У цей момент відкритий файл тексту програми з розширенням *.c

Праворуч розміщене вікно проекту (Solution Explorer). У даному вікні можна побачити файли підключені до проекту, які файли з'являються в результаті компіляції й ін.

Унизу розташоване вікно висновку діагностичної інформації (Output). У дане вікно середовище розробки виводить інформацію про процес компіляції й зборки проекту, а також інформацію про знайдені помилки.

При створенні проекту середовище розробки створює заготівлю вихідного тексту основної процедури ПО (функція **main**), а також довідковий блок коментарів.

При написанні програми зарезервовані слова виділяються синім кольором, коментарі - зеленим, основний текст - чорним і ін., тобто виробляється підсвічування синтаксису.

При написанні програми дуже рекомендується дотримуватися «гарного стилю написання коду»!:

- ✓ робити відповідні відступи в тексті для виділення логічних блоків коду (як горизонтальні - пробіли, так і вертикальні - порожні рядки);
- ✓ давати осмислені імена змінним і функціям на англійському мові;
- ✓ постачати програму коментарями.

Додаткові підказки:

- ✓ за відкриваючою дужкою негайно ставити закриваючу й убивати потрібний текст між ними;
- ✓ при написанні програми користуватися методом послідовного наближення, тобто писати програму вроздріб (завершеними блоками) і періодично перевіряти (компілюючи текст програми й/або перевіряючи його на реальному пристрої).

ЗБІРКА ПРОЕКТУ

Після написання тесту програми, його варто відкомпілювати й зробити зборку проекту. Для зборки проекту існує панель зборки (Build), див. Рис. 5.

Рис. 5. Панель зборки проекту

На панелі присутні 3 кнопки:

- ✓ зборка проекту (Build);
- ✓ зборка рішення (Build Solution);
- ✓ скасування (Cancel).

Всі панелі середовища розробки є що набудовуються, тобто їхній вид може відрізнятися від наведеного вище.

Зборка проекту (Build) складається із двох основних фаз:

✓ компіляція вихідних текстів програми написаних Вами (Compiling);

✓ підключення відкомпільованих раніше стандартних бібліотек функцій і т.д. - називається лінковка (Linking).

Компіляція – процес перекладу тексту програми, написаної мовою програмування, в об'єктний модуль, що містить машинні команди конкретного процесора.

Компоновщик (лінкер) – програма, що робить компонування: приймає на вхід один або кілька об'єктних модулів і збирає по них виконується модуль, що.

За результатами зборки проекту у вікні Output буде наданий звіт про хід роботи, а в Solution Explorer будуть показані файли отримані в результаті зборки. Програма втримується в ELF-файлі й HEX-файлі (вони дублюють один одного, а розроблювач використає необхідний з них).

НАЛАГОДЖЕННЯ

На написання тексту програми йде 20% часу, а інші 80% - займає процес налагодження проекту (Debug) і доведення його до стану випуску (Release).

Налагодження – етап комп'ютерного рішення завдання, при якому відбувається усунення явних помилок у програмі. Часто виробляється з використанням спеціальних програмних засобів - відлагоджувальників.

Виправлення синтаксичних помилок

Якщо при написанні ПО минулому допущені синтаксичні помилки, то компілятор не зможе відкомпілювати вихідний текст програми й видасть список знайдених помилок (Error List), див. Рис. 6.

Рис. 6. Помилки й попередження компілятора

У списку помилок (Errors) показується їхній опис і можливе місце розташування.

Також можуть бути присутнім попередження (Warnings) - місця в коді на які варто звернути увагу, тому що вони можуть викликати помилку під час виконання програми. І повідомлення (Messages).

Після вдалої зборки проекту можна приступитися до перевірки логіки роботи програми за допомогою симулятора або на реальному пристрої.

Симуляція роботи програми на ПК

Симуляція – моделювання процесу виконання програми мікроконтролером на персональному комп'ютері. Один із самих потужних методів налагодження програм.

Для керування процесом симуляції існує панель налагодження (Debug Toolbar), див Рис. 7.

Рис. 7. Панель налагодження проекту

Для керування ходом виконання програми призначені наступні кнопки:

	запуск/пауза налагодження програми (Start Debugging and Break)
	зупинка налагодження (Stop Debugging)
	дана стрілка показує команду, що буде виконана наступної (Show Next Statement)
	виконання команди із заходом у підпрограму
	(Step Into)
	виконання команди без заходу в підпрограму (Step Over)
	виконання коду до кінця поточної підпрограми (Step Out)
	виконання коду до місця розташування курсору (Run To Cursor)
	скидання - перехід програми в початковий стан (Reset)
	виконання програми без зупинок (Continue)
	пауза виконання програми (Break All)
	додаткові функції (малюнок може відрізнятися)

Для зупинки програми в певній крапці коду використаються, так звані, контрольні крапки.

Контрольна крапка – інструкція, у програмі дійшовши до якої виконання програми призупиниться. Установлена контрольна крапка відзначена червоним кружком.

Список установлених контрольних крапок (Breakpoints) можна одержати в додаткових функціях.

Для контролю значення змінних оголошених у програмі використається вікно Locals, див. Рис. 8. Контролюючи значення змінних ми можемо зрозуміти логіку роботи ЗД. (Кнопка перебуває в додаткових функціях).

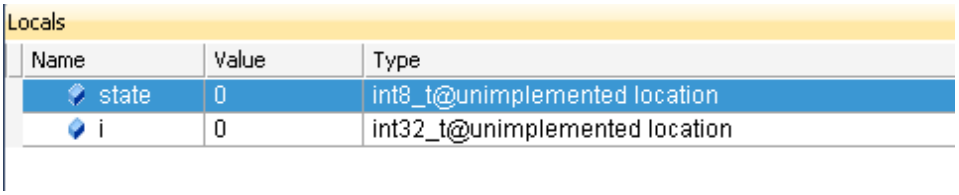

Рис. 8. Вікно контролю локальних змінних

У ході покрокового виконання програми (т.зв. трасування) розроблювач контролює стан МК, його периферії й модулів і т.д. у ході роботи програми. Для цього призначена панель налагодження AVR (AVR Debug Toolbar), див. Рис. 9.

Рис. 9. Панель налагодження AVR

На панелі є наступні команди:

	перегляд стану процесора (Processor View)
	перегляд регістрів загального призначення (Registers)
	перегляд пам'яті (Memory)
	перегляд периферії МК (I/O View)
	перегляд дізассемблованого коду (Disassembly)

У вікні стану процесора можна побачити наступне (див. Рис. 10)...

Програмний лічильник (Program Counter) – адреса машинної інструкції, що буде виконана в наступний такт (цикл) роботи МК.

Покажчик стека (Stack Pointer) – адреса вершини стека в пам'яті МК.

Регістр статусу (Status Register) – спеціальний регістр утримуючий статус результату виконання попередньої машинної інструкції. Інакше називається - регістр прапорів.

Лічильник циклів (Cycle Counter) – лічильник циклів минулих з моменту запуску програми.

Час зупинки (Stop Watch) – час у плині якого програма виконувалася.

Частота (Frequency) – тактова частота МК. Чим вище тактова частота, тим більше інструкцій МК може виконати за од. часу.

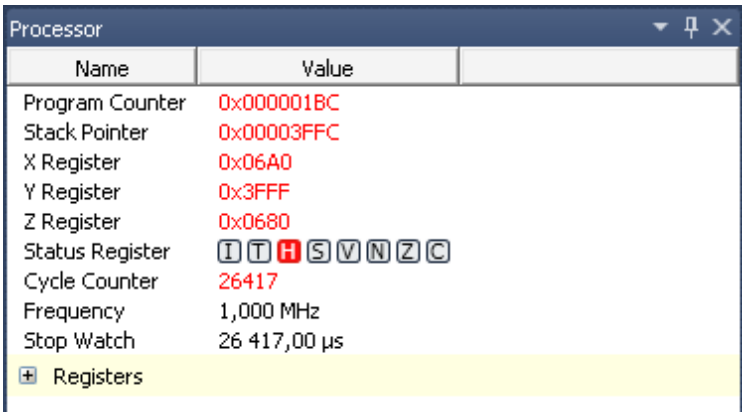

Рис. 10. Вікно стану процесора

Архітектура МК AVR припускає наявність 32-х 8-бітних регістрів загального призначення. Останні 6 регістрів формують, так звані, 16- бітні регістри непрямої адресації (X, Y, Z Registers) - вони використаються для доступу до пам'яті МК.

Вікно перегляду пам'яті дозволяє переглянути, а при необхідності й модифікувати, будь-яку пам'ять МК (пам'ять програм, пам'ять даних, EEPROM).

Вікно дизасемблювання показує, як програма мовою високого рівня була переведена в машинні коди компілятором.

Вікно перегляду периферії МК представлено на Рис. 11.

У даному вікні показані регістри, відповідальні за роботу убудованої в МК периферії, тобто порти введення/виводу, таймери, компаратори, комунікаційні інтерфейси й ін..

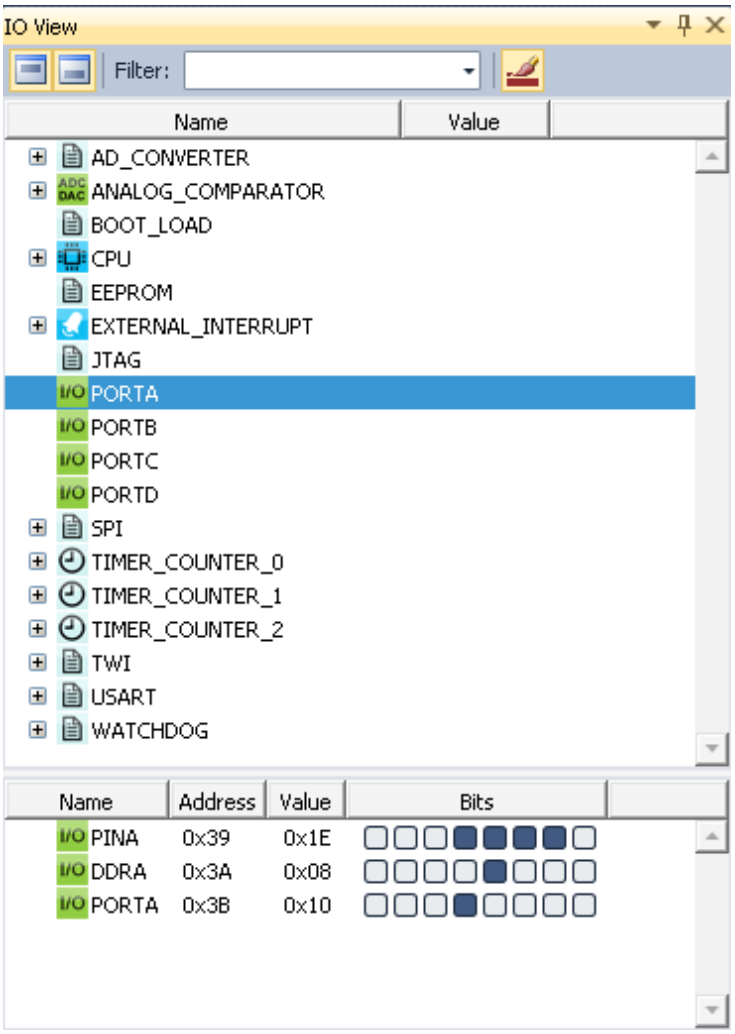

Рис. 11. Вікно перегляду периферії МК

ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРА

Коли програма налагоджена й працює відповідно до задуманої ідеї її можна прошивати в МК і тестувати на реальному МК у реальному оточенні.

Для прошивання МК може бути використана програма AvrFlash, див. Рис. 12.

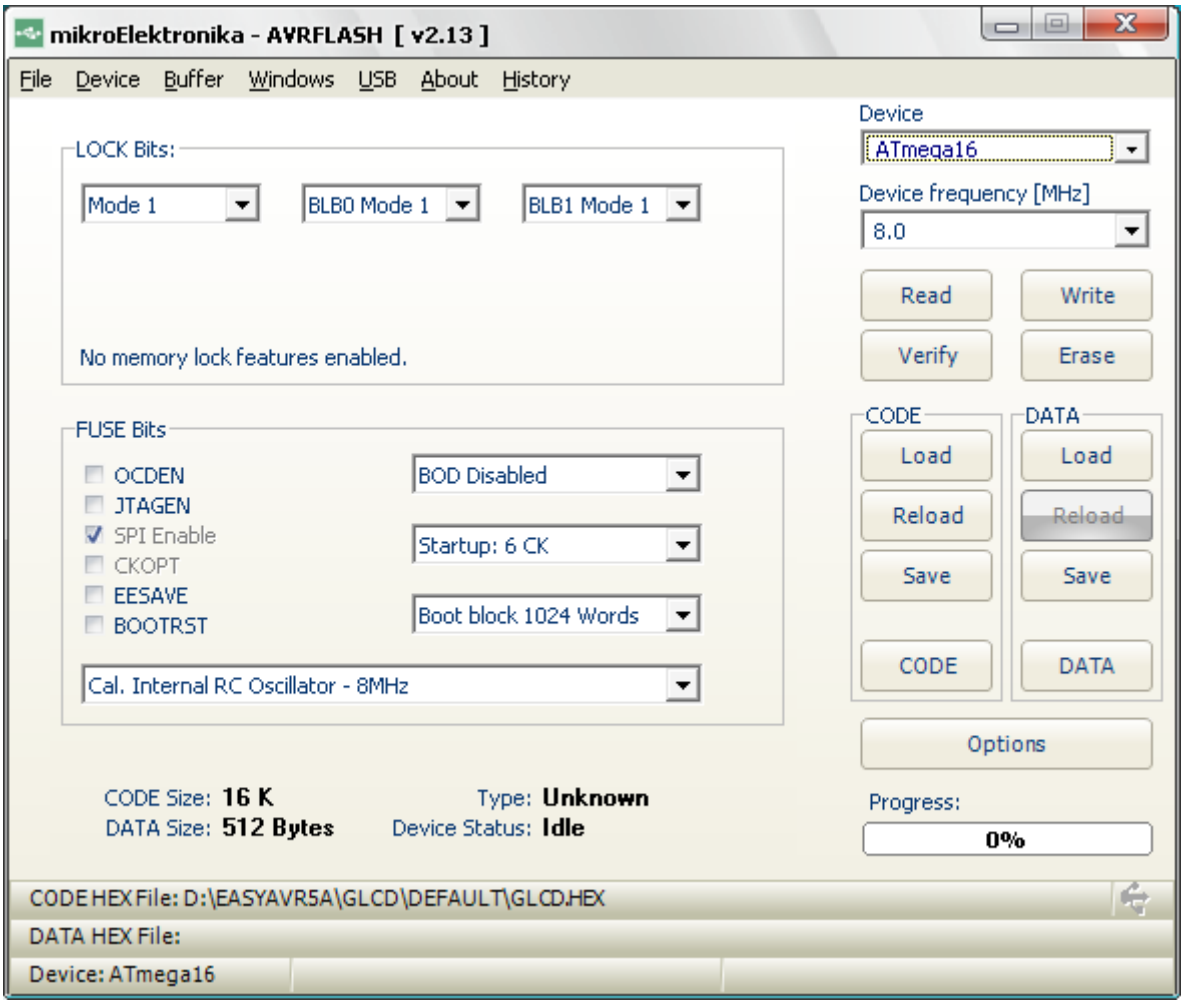

Рис. 12. Прошивання МК

ПОЛЯ В ОБЛАСТІ «LOCK BITS» НЕ ЗМІНЮВАТИ!!!

Це спеціальні біти призначені для уведення МК у різні режими (тільки читання, неможливість скидання й т.п.). Зміна даних полів приводить у деяких випадках до блокування МК (неможливо стерти й т.п.). Використати їх можна тільки з розумінням що відбувається.

Область Fuse Bits - треба виставити робочу частоту і її джерело (випадаючий список унизу групи). Device - виставити використовуваний МК. Device Frequency - частота на якій МК працює.
Область «CODE» - відповідає за роботу з кодом програми:

- ✓ LOAD - завантажити код програми на згадку ПК із HEX-файлу (*.hex);
- ✓ RELOAD - перезавантажити код програми на згадку ПК;
- ✓ SAVE - зберегти код програми на ПК.

Кнопки READ, WRITE, ERASE, VERIFY відповідають за читання, запис, стирання й перевірку МК відповідно.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке AVR Studio.

2. Яка стандартна послідовність дій при розробці програми для мікроконтролера.
3. Що таке зборка.
4. Чим компілювання відрізняється від лінкування.
5. Що таке симуляція.
6. Навіщо роблять налагодження програми.
7. Що таке контрольна крапка.
8. Що таке дизасемблювання.
9. Навіщо призначена вкладка I/O View.

Текст програми

```
// Підключаємо зовнішні бібліотеки
#include <AVR/io.h>
#include <stdint.h>
#include <util/delay.h>


// Основна програма
int main(void)
{
unsigned int line;
unsigned int line;
unsigned int line;
unsigned int line;
short i;

// Налаштовуємо порти введення/виводу
DDRA   = 0xFF;
DDRB   = 0xFF;
DDRC   = 0xFF;
DDRD   = 0xFF;

// Вічний цикл
while (1)
{
// Формуємо вихідну картину
line       = 0b1100000000000000;
line       = 0b0110000000000000;
line       = 0b0110000000000000;
line       = 0b1100000000000000;

// Виводимо її на екран
for (i = 0; i < 20; i++)
{
// Вивід на світлодіоди
PORTA = line;
PORTB  = line;
PORTC        = line;
PORTD        = line;

// Зсування зображення line = line >> 1; line = line >> 1;
line = line >> 1; line = line >> 1;

// Затримка
_delay_ms(300);
}
}
}
```

Текст програми

```
// Константи для включення індикаторів
#define DIS3 0x08
#define DIS2 0x04
#define DIS1 0x02
#define DIS0 0x01

// Затримка зміни лінії
#define delay_const 400

// Підключаємо використовувані бібліотеки
#include <AVR/io.h>
#include <util/delay.h>

void move_line(void)
{
// Виводимо першу лінію
PORTC = 0x30;
_delay_ms(delay_const);

// Виводимо другу лінію
PORTC = 0x06;
_delay_ms(delay_const);

// Виключаємо висновок
PORTC = 0x00;
}

// Основна програма
int main(void)
{
// Настроювання портів
DDRA  = 0xFF;
DDRB  = 0xFF;
DDRC  = 0xFF;
DDRD  = 0xFF;

// Висновок лінії, що біжить
//  (вічний цикл) while(1)
{
// Включаємо перший символ
PORTB = DIS3;

// Зсуваємо лінію
move_line();

// Включаємо перший символ
PORTB = DIS2;
```

```
// Зсуваємо лінію
move_line();

// Включаємо перший символ
PORTB = DIS1;

// Зсуваємо лінію
move_line();

// Включаємо перший символ
PORTB = DIS0;

// Зсуваємо лінію
move_line();
}
}
```