1 ПРОГРАМНІ ЗАСОБИ ПІДТРИМКИ ПРОЕКТУВАННЯ ТА ВІДЛАГОДЖЕННЯ СИСТЕМ

1.1 Середовища розробки програмного забезпечення для мікроконтролерів AVR

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

– підготовка вихідного тексту програми якою-небудь мовою програмування;

– компіляція програми;

– налагодження й тестування програми;

– остаточне програмування й підготовка до серійного виробництва.

Мікропрограма пристрою має бути написана однією з мов програмування. На сьогодні для МК AVR існують 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 наголосити, 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.

1.1.1 Компілятори асемблера

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

Досить вдалим вибором при розробці програмного забезпечення для МК набору AVR може стати інтегроване середовище розробки AVR Studio фірми Atmel [2, 8, 15, 20, 21, 23, 27], що містить у собі текстовий редактор з підсвічуванням синтаксису, компілятор асемблера, симулятор, відлагоджувач й інтерфейс із апаратними емуляторами. Програма розрахована на роботу під керуванням операційних систем Windows 7, 8, 10. До недоліків AVR Studio можна віднести деяку нестабільність роботи налагоджувача, а також неповну симуляцію периферійних пристроїв (зокрема, відсутня симуляція АЦП). До плюсів же належить, насамперед, підтримка практично всіх МК набору AVR.

Atmel безкоштовно розповсюджує асемблер фірми IAR. Цей компілятор є частиною IAR Embedded Workbench [16, 27] – середовище розробки, що містить менеджер проектів, редактор, лінкер і менеджер бібліотек. Цей компілятор відрізняється розширеними можливостями по роботі з макросами.

GNU/Linux AVR Assembler повністю сумісний із компілятором фірми Atmel, має, порівняно з ним, кілька переваг. Він надається з відкритими текстами, так що користувач, якщо буде потреба, може легко додати нові можливості. І це один з декількох асемблерів для МК фірми AVR, що працюють під керуванням операційної системи Linux (до того ж, наявність вихідних текстів дозволяє легко перенести його на будь-яку іншу Unix-систему). У GNU AVR Assembler значно розширені, порівняно з асемблером фірми Atmel, можливості по роботі з макрокомандами (зокрема, допускаються макроси усередині макросів). Програма поширюється за умовами GNU Public License (GPL) [2, 21, 27].

1.1.2 Компілятори мови С

Останнім часом усе популярнішим стає використання компіляторів мов високого рівня під час написання програм для МПС. Найбільшого поширення при цьому одержали компілятори мови С, оскільки в цій мові найбільш просто реалізуються всі необхідні можливості з керування апаратними засобами МК.

Компілятор фірми IAR є одним із кращих компіляторів С для МК набору AVR. Пов'язано це з наявністю в ньому можливостей з оптимізації коду. Істотним його недоліком є те, що у демонстраційній версії накладаються значні обмеження на максимальний обсяг коду. Компілятор додається у склад інтегрованого середовища розробки IAR Embedded Workbench (EWB), що містить компілятор асемблера, лінкер, менеджер проектів і бібліотек, а також відлагоджувач. Він може працювати сумісно з AVR Studio.

Image Craft C Compiler [5, 14, 16] є другим за популярністю компілятором мови С для МК набору AVR. До переваг цього компілятора можна віднести можливість підтримки мікросхем набору FPSLIC фірми Atmel, непоганий рівень оптимізації коду й досить низька ціна. Демо-версія компілятора не містить функціональних обмежень і лімітована тільки часом роботи (30 днів).

Іншим популярним компілятором мови С для МК набору AVR є Code Vision AVR C Compiler [5, 14, 10, 16, 21, 27]. Компілятор підтримує МК ATTiny22, AT90S23x3, AT90S4433 та багато інших. Демонстраційна версія компілятора, так само, як і компілятор фірми IAR, має обмеження на максимальний обсяг коду програми. Компілятор додається до інтегрованого середовища розробки, у якому крім стандартних можливостей, включена досить цікава функція – Code Wizard AVR Automatic Program. Фактично це генератор стандартних блоків програми, що виконує такі функції:

– налаштування доступу до зовнішньої пам'яті;

– ініціалізація портів введення/виведення, зовнішніх переривань, таймерів, сторожового таймера;

– ініціалізація асинхронного порту послідовної передачі даних (UART) і прийом, передачу даних через UART;

– налаштування аналогового компаратора й АЦП;

– ініціалізація шини I2C і підключених до неї температурного датчика LM75 або годинника реального часу PCF8583, DS1302, DS1307;

– реалізація протоколу 1-Wire Bus й ініціалізація температурного датчика DS1820;

– керування LCD-індикатором.

Наявність цієї можливості спрощує написання програм. Наявність у середовищі розробки послідовного термінала дозволяє робити налагодження програм з використанням послідовного порту UART МК. Крім С, при розробці програмного забезпечення для МПС застосовуються й інші мови високого рівня. Так, для МК набору AVR існують також компілятори мов Basic, Pascal і Forth [5, 21, 27].

Компілятор мови Basic, розроблений фірмою MCS Electronics. Є демоверсія з обмеженням на обсяг коду програми. За вхідним кодом компілятор практично повністю сумісний з компіляторами VisualBasic/QuickBasic фірми Microsoft. У синтаксис мови додано кілька нових команд для забезпечення підтримки LCD-індикаторів, I^2C й 1-Wire інтерфейсів, оброблення переривань й інших специфічних для МК можливостей.

ABC Basic Compiler, що розроблений фірмою Investments Technologies PTY, дозволяє писати й налагоджувати програми мовою Basic в інтегрованому середовищі розробки. Існують версії компілятора як під Windows, так і під DOS.

Microbasic і Mikropascal є потужними середовищами програмування для МК AVR. Такі середовища створені для полегшення роботи користувача і поліпшення можливості роботи з МК набору AVR. Вони дозволяють:

– писати код Basic та Pascal за допомогою редактора коду;

– використовувати бібліотеки для прискорення роботи з даними, пам’яттю, індикаторами, периферійними пристроями;

– бачити структуру програми, змінні й функції у редакторі коду, генерувати прокоментовану, зрозумілу людині програму і сумісну з HEX-файлами для будь яких програм.

1.2 Інтегроване середовище розробки програм AVR Studio

Фірмою «Atmel» розроблено програмний пакет підтримки розробок на AVR-мікроконтролерах в середовищі Windows – AVR Studio. AVR Studio – це інтегроване середовище відлагодження розробки програм (Integrated Development Environment – IDE), що містить транслятор мови асемблера AVR-мікроконтролерів, відлагоджувач програмного забезпечення верхнього рівня для підтримки внутрішньосхемного програмування. Відлагоджувач AVR Studio підтримує всі типи мікроконтролерів AVR і має два режими роботи: режим програмної симуляції і режим управління різними типами внутрішньосхемних емуляторів виробництва фірми

«Atmel». Середовище відлагоджування підтримує виконання програм як у вигляді асемблерного тексту, так і у вигляді вихідного тексту мови С. AVR Studio поширюється вільно, його остання версія завжди доступна на сайті фірми «Atmel» [2, 8, 15, 20, 27]. Для запуску програми запустіть файл AVRStudio.exe. З'явиться основне діалогове вікно програми (рис. 1.1). У верхній частині програми знаходиться меню, у якому необхідно вибрати Project→New Project (рис. 1.2). У новому вікні обирається ім’я проекту (Project name), місце на диску, де зберігається проект (Location), а також тип проекту (Project type). Для цього мишею обираємо AVR Assembler та натискаємо кнопку Next.

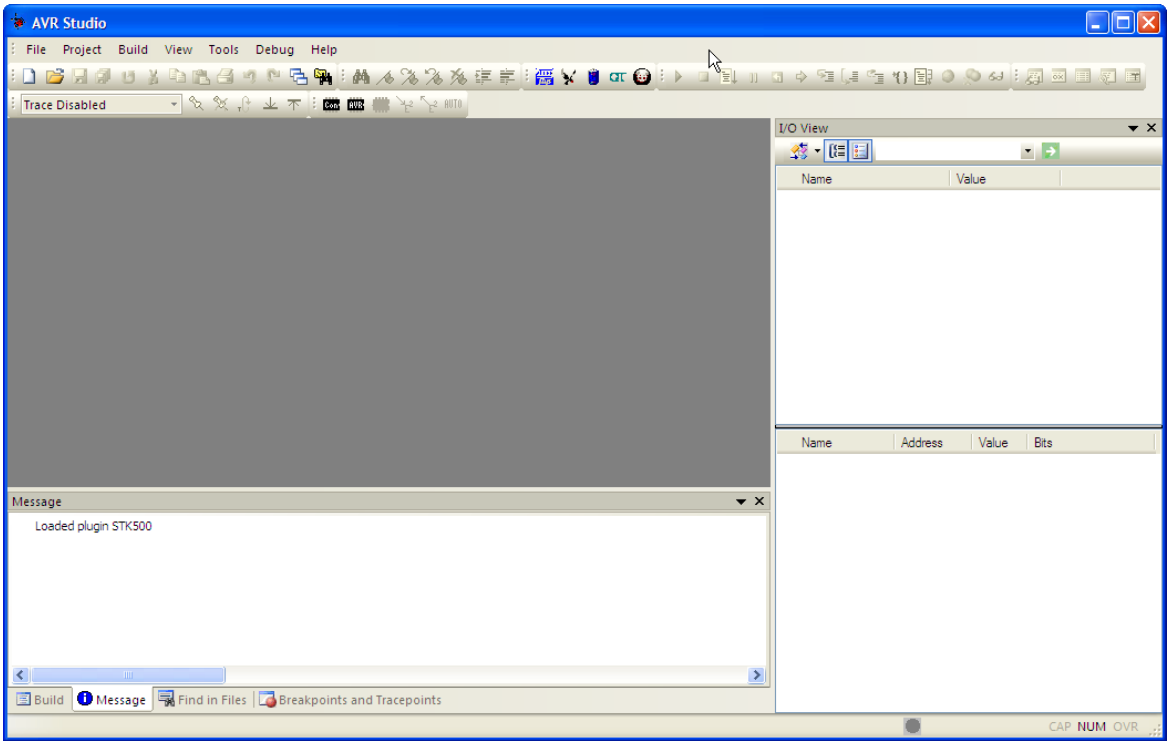

Рисунок 1.1 – Основне вікно програми AVR Studio

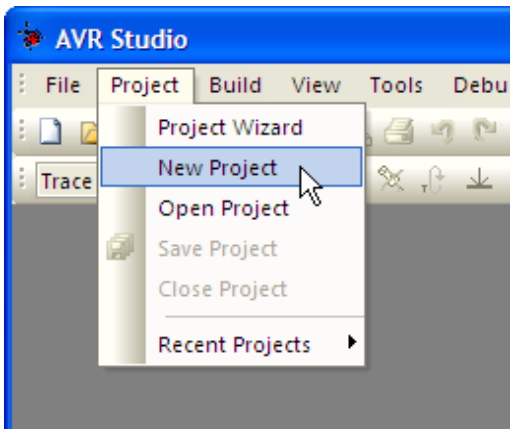

Рисунок 1.2 – Створення нового проекту
у програмі AVR Studio

У вікні вибору платформи відлагодження та пристрою (Select debug platform and device) обираємо AVR Simulator та мікроконтролер, наприклад, Atmega8535і завершуємо процедуру вибору кнопкою Finish (рис. 1.3).

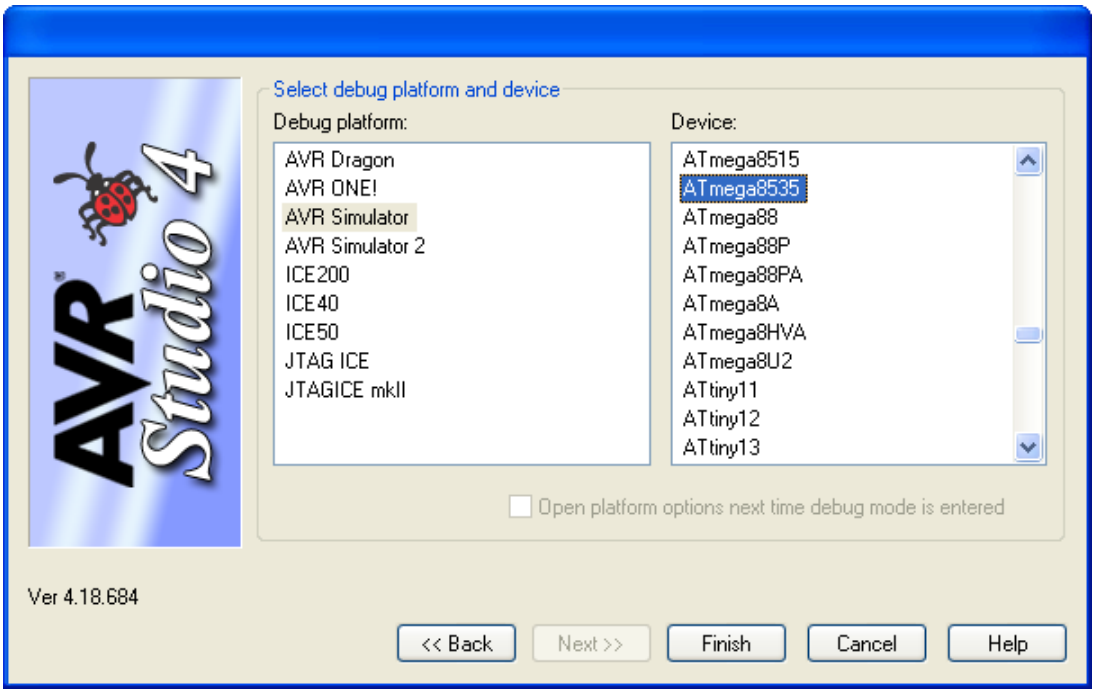

Рисунок 1.3 – Вибір платформи відлагодження та типу мікроконтролера у середовищі AVR Studio

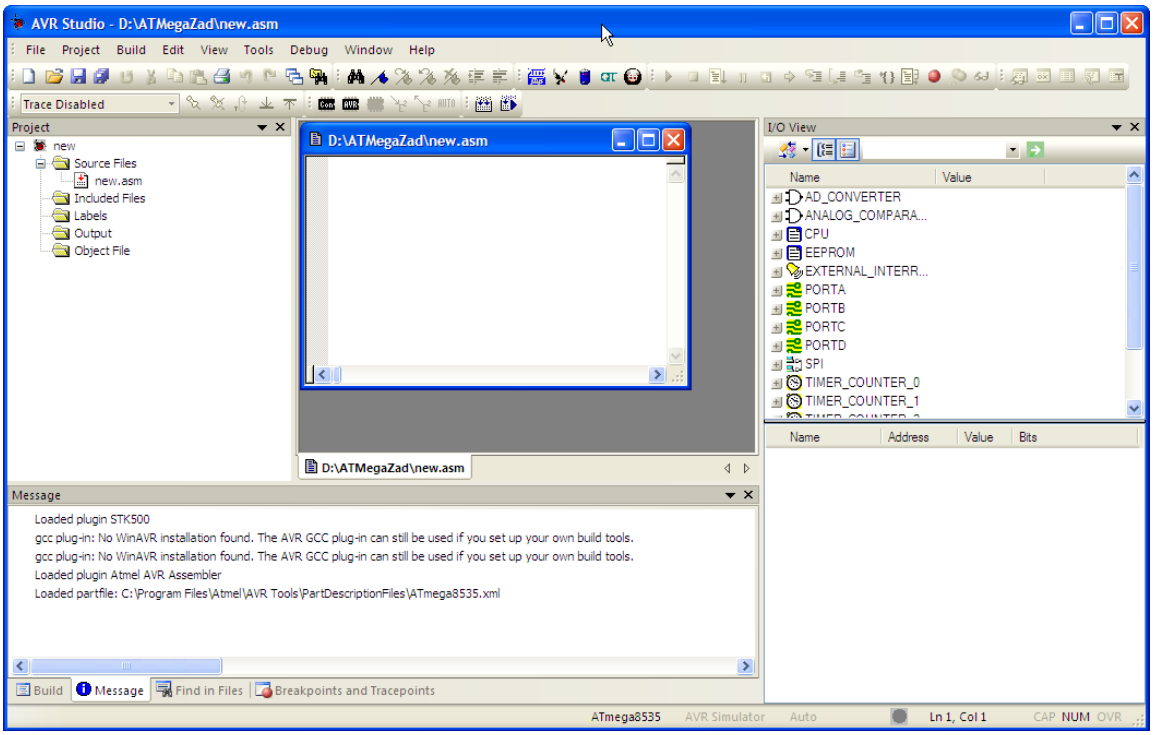

Рисунок 1.4 – Вікно нового проекту у програмі AVR Studio

У вікні Project (див. рис. 1.4) обираємо файл з розширенням *.asm та у вікні, що відкрилось, набираємо програму. Після того як програма написана, у верхньому меню обираємо Build та виконуємо її компіляцію, при цьому створюється файл з розширенням *.hex, який необхідно записати в мікроконтролер. Після компілювання з'явиться вікно Build, де зазначено, який файл асемблюється, файли бібліотеки, що використовуються, кількість слів у програмі та повідомлення про відсутність помилок Assembly complete with 0 errors, 0 warnings. Якщо є помилки, то в цьому вікні вказуються тип помилки, номер рядка з помилкою і в кінці загальне число помилок. Для їхнього виправлення необхідно повернуться до редагованого файлу, а потім знову відкомпілювати програму.

AVR Studio дозволяє не тільки компілювати програми, а й налагоджувати їх на етапі розробки. При цьому AVR Studio емулює роботу мікроконтролера, всіх портів введення/виведення, лічильників/таймерів, переривань, ШІМ і АЦП. Емуляція роботи програми дозволяє розглянути її роботу, якщо б вона була записана в мікроконтролер.

Необхідно зазначити, що емулювати можна тільки роботу програми, яка не містить помилок. Тому перед емуляцією AVR Studio зробить компіляцію програми, і якщо є помилки, то емулювати (відлагодити) програму неможливо. Для налагодження програми, після того як вона написана, потрібно в меню Build вибрати пункт Build and run. Викликати вікно опцій емулятора (Simulation Options) у меню Debug→AVR Simulator Options. У пункті пристрій (Device) потрібно вибрати мікроконтролер ATmega8535, у пункті частота (Frequency) – частоту 8 МГц, натиснути кнопку ОК (рис. 1.5).

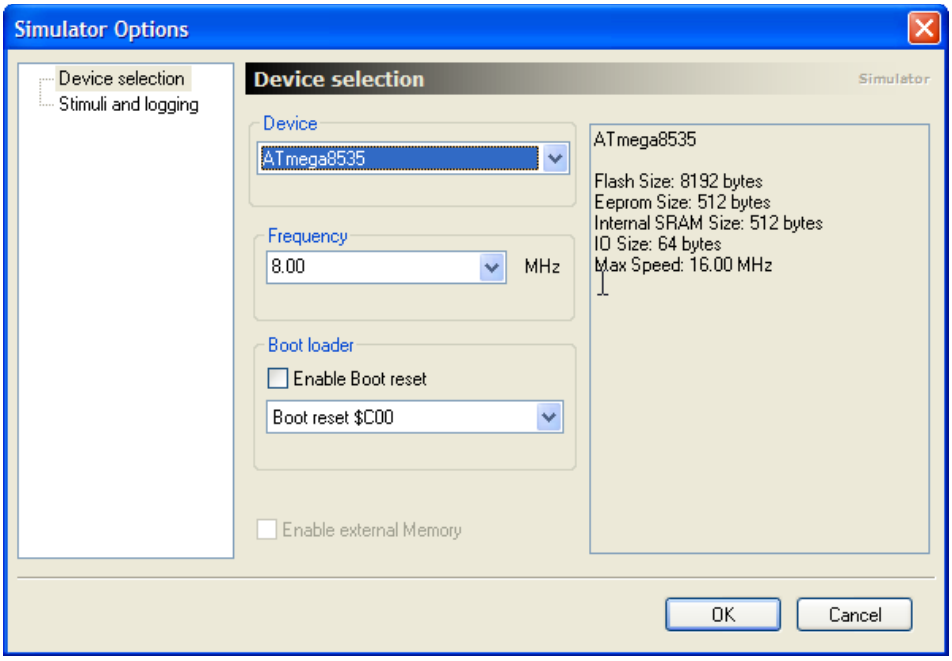

Рисунок 1.5 – Опції емулятора (Simulation Options) середовища AVR Studio

Після цього з'явиться вікно, де набиралася програма, початок програми буде зазначено жовтою стрілкою – це початок програми, вище вказано директиви компілятора. При емуляції роботи програми необхідно бачити стан регістрів, портів введення/виведення (рис. 1.6). Для перегляду регістрів у головному меню програми обираємо пункт перегляд (View), потім пункт регістри (Registers), для перегляду стану процесорного ядра використовується панель процесор (View→Toolbars→Processor), порти введення/виведення і периферійні модулі зручно спостерігати через панель введення/виведення (View→Toolbars→I/O). У меню View є й інші пункти, що можна використовувати: пам'ять (Memory) для перегляду пам'яті даних і програм.

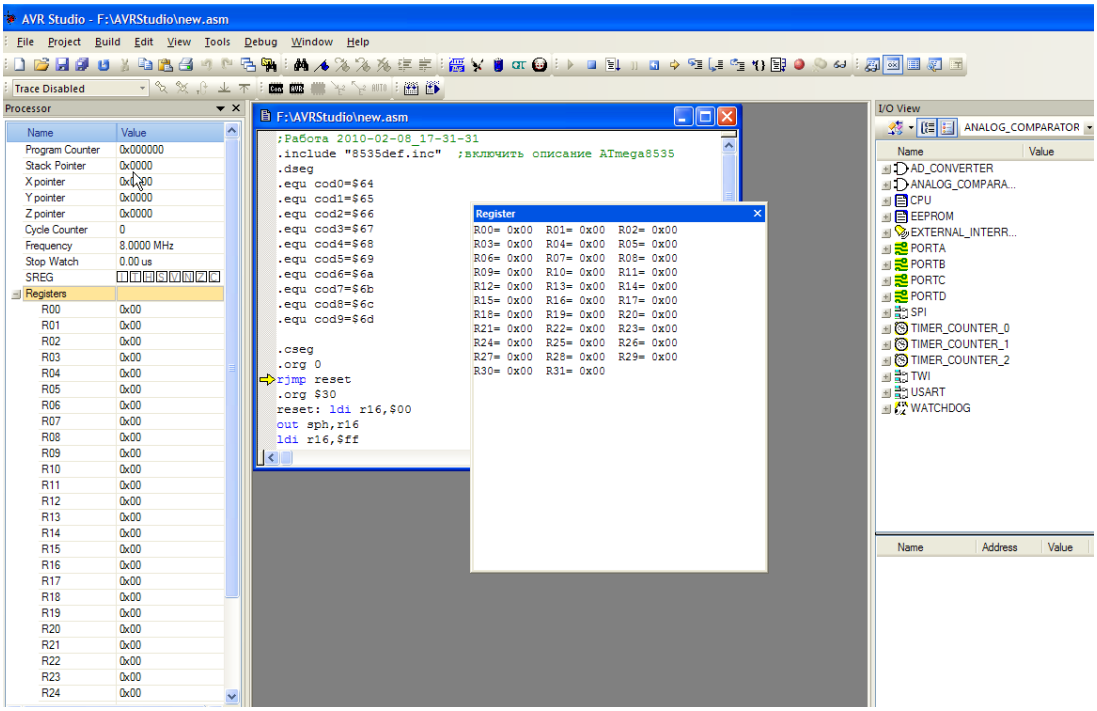

Рисунок 1.6 – Вікно емуляції роботи програми в AVR Studio

AVR Studio дозволяє запустити програму в реальному часі у покроковому режимі до покажчика. У головному меню в пункті налагодження (Debug) знаходяться всі варіанти запуску програми. Reset – скидання на початок програми (жовта стрілка покажчика показує на початок); Go – запуск у реальному часі (програма буде виконуватися до тих пір, поки не буде обраний пункт Break); Step over – покроковий режим (програма виконується через рядок, при цьому зупиняється після кожної команди, стрілка вказує на поточну команду); Run to cursor – виконувати до курсора (програма виконується до місця зазначеного курсором у вікні з редагованої програмою). Під час виконання програми можна спостерігати за станом регістрів після кожної команди – так перевіряється правильність операцій, що виконується мікроконтролер. Найбільш зручний режим для цього – покроковий.

На панелі Input/Output View, де наведені всі пристрої мікроконтролера, навпроти кожного пристрою є знак «+»; клацнувши на ньому мишкою, отримуємо вміст цього пристрою, тобто стан регістрів, регістрів даних і т. д. Два рази клацнувши на змісті якого-небудь регістра, можна змінити його стан в процесі виконання програми. У регістрах портів введення/виведення можна задати вхідні сигнали. Для цього позначають галочкою потрібний біт стану (логічна одиниця), так емулюється вплив зовнішніх сигналів.

До Atmel Studio 7 входить бібліотека Atmel Software Framework (ASF), і являє собою колекцію готового до використання початкового коду та більше 1600 прикладів проектів, що створені та оптимізовані експертами та випробувані в промислових системах. Наявні у складі колекції драйвера периферійних пристроїв, комунікаційні стеки та спеціальні прикладні бібліотеки прискорюють та спрощують розробку проектів.

1.3 Робота в інтегрованому середовищі розробки програм WinAVR

Відкрийте програму Programmers Notepad (PN), що входить до складу WinAVR та виберіть новий C/C++ файл (рис. 1.7).

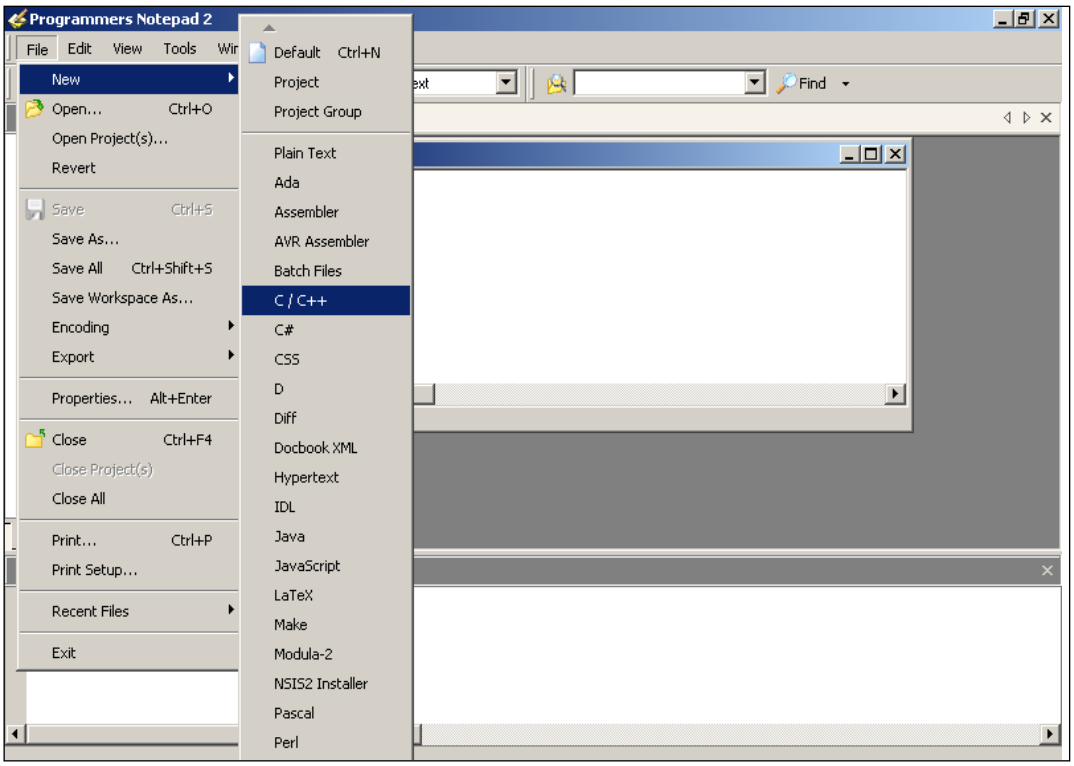

Рисунок 1.7 – Створення нового файлу в редакторі Programmers Notepad

Створіть папку проекту, наприклад D:\prj. У PN наберіть код і збережіть файл у папку проекту під ім’ям simple.c.

```
#include <AVR/io.h>
#include <stdint.h>
// Підключення необхідних бібліотек
void main(void)
{       MCUCR = 0xA0; //Дозволити роботу зі зовнішньою пам'яттю
        unsigned char* const pLeftInd = 0xA000;
        //Покажчик на комірку пам'яті за адресою 0xA000
                *pLeftInd = 0x66;
        while (1)
        {
        // Нескінчений цикл
        }
}
```

Для компілювання програми в середовищі WinAVR використовується один з найпотужніших і найгнучкіших компіляторів – GCC. І саме через свою універсальність він має дуже багато налаштувань, які для компілювання програми можна кожний раз вводити через командний рядок, а можна один раз записати у спеціальний makefile.

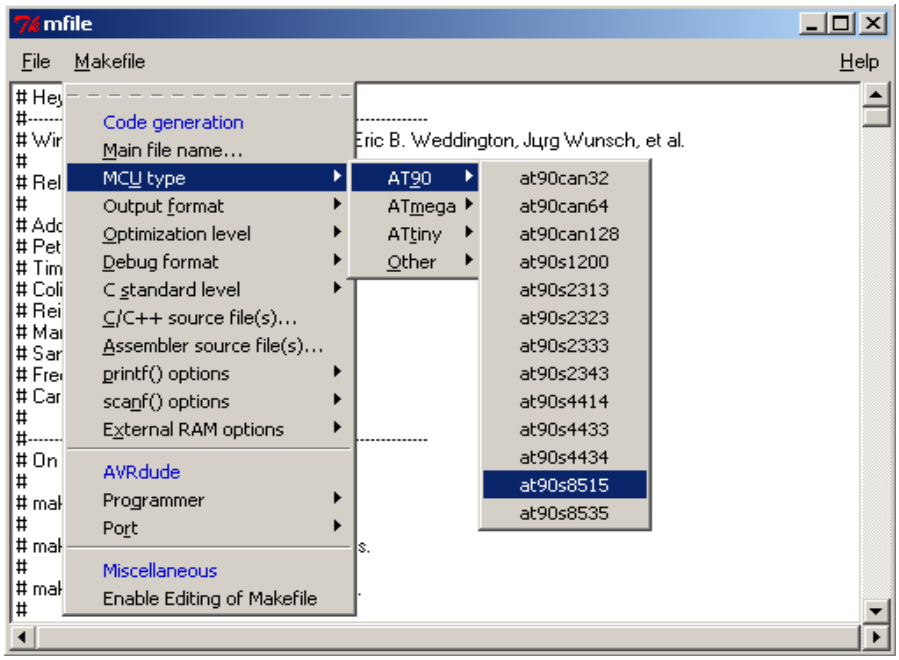

Рисунок 1.8 – Вибір типу мікроконтролера для створення makefile

Після компілювання буде створено об’єктний файл у форматі операційної системи Linux (ELF). Цей об’єктний файл необхідно перетворити у формат HEX. Ця дія забезпечується за допомогою лінковщика, який зокрема може бути викликаний з командного рядка, або керуватися налаштуваннями makefile. Тож найкращий вихід побудови файлу програми – це побудова з використанням makefile. Створити makefile можна або вручну,

або за допомогою спеціального генератора. У ролі генератора мейкфайлів буде використовуватись опція Mfile (рис. 1.9).

Для створення makefile проекту виконуються такі кроки:

1. Завантажуємо програму Mfile (входить до складу WinAVR) і вибираємо тип МК (див. рис. 1.8).
2. У меню Output format виберіть ihex (Intel HEX).
3. У меню C standard level – gnu99.
4. Збережіть файл у директорію проекту. Файл обов'язково має містити ім'я makefile.
5. Додайте до makefile запис про вихідний текст програми, що знаходиться у файлі simple.c.

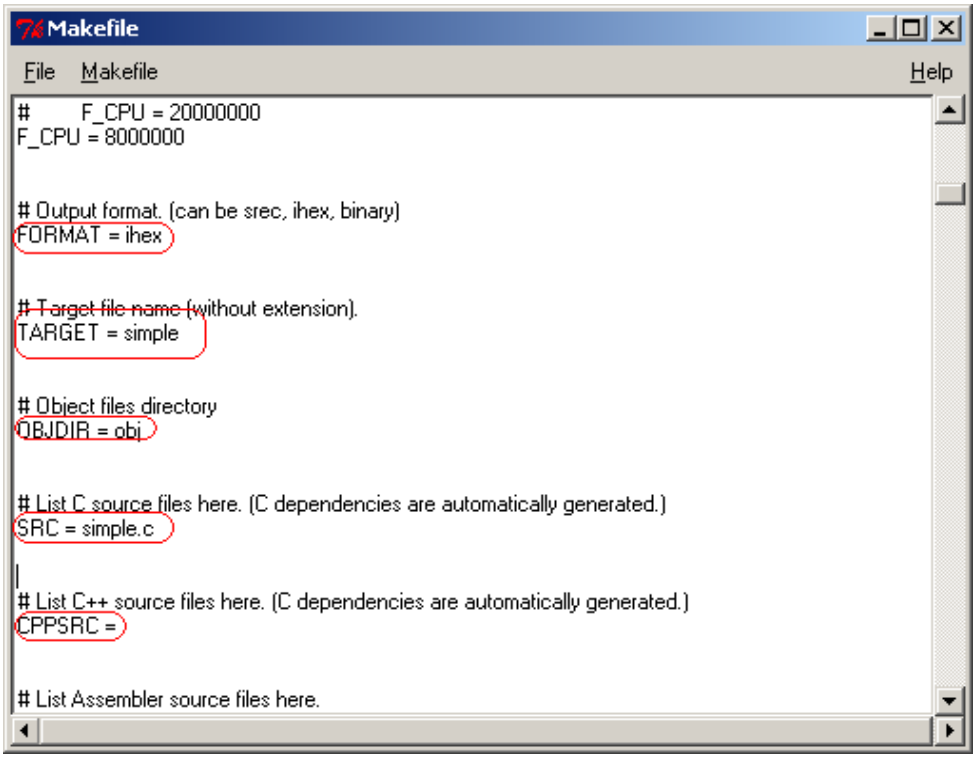

Рисунок 1.9 – Налаштування makefile

Зробити це можна за допомогою діалогу, що викликається при виборі пункту меню C/C++ source file(s). Але ця частина програми працює не завжди коректно, і тому, якщо змін файлу не сталося (зміни у makefile підсвічуються жовтим кольором), цей параметр потрібно встановити вручну. Для цього поставте прапорець на пункті меню Makefile→Enable Editing of Makefile.

Знайдіть параметр TARGET та встановіть його значення: TARGET = simple. Знайдіть параметр SRC і встановіть його значення: SRC = simple.c. Приберіть значення параметра CPPSRC. Необхідні налаштуваня наведені на рис. 1.9. Збережіть вміст makefile.

Компіляція та лінкування. Після виконаних дій у директорії проекту з'являться 2 потрібних файли (рис. 1.10): simple.c і makefile.

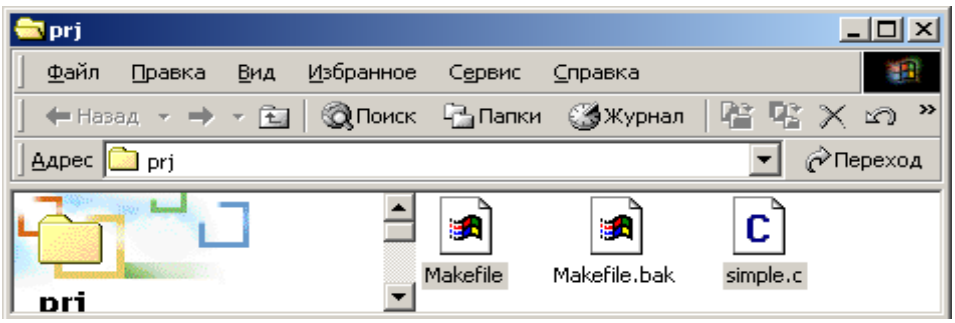

Рисунок 1.10 – Файли проекту

Тепер потрібно створити виконуваний файл програми (ihex), що зрозумілий МК:

1. Відкривається вихідний текст програми в редакторі PN.
2. Обираємо пункт меню Tools→[WinAVR] Make Clean (рис. 1.11).

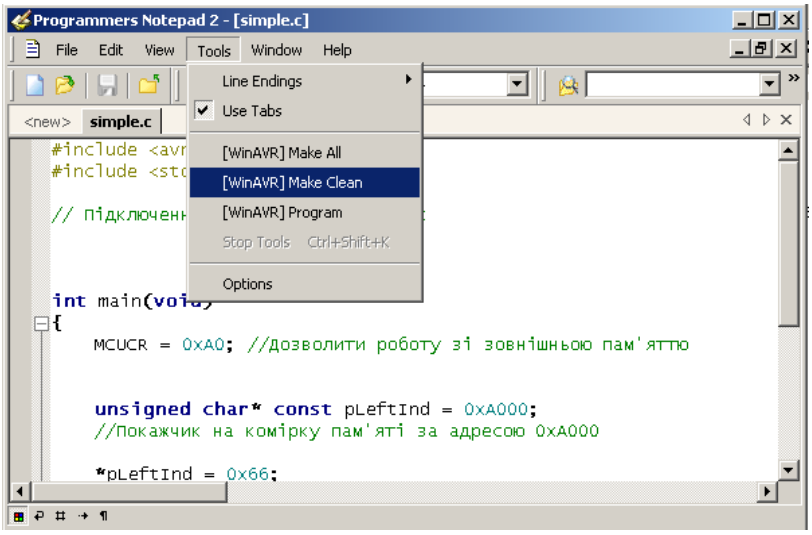

Рисунок 1.11 – Виконання make clean

Ця дія запускає спеціальну утиліту make, що згідно з параметрами має очистити всі зайві файли, що могли залишитися від попереднього компілювання. Під час виконання команди у нижній частині PN з’явиться вікно “Output” (рис. 1.12), що буде інформувати про перебіг виконання команди. Операція вважається успішною, якщо ви побачите внизу рядок:

“> Process Exit Code: 0”

Якщо значення Process Exit відмінне від нуля – це означає, що сталася помилка. Для операції "make.exe" clean помилку потрібно шукати у неправильній конфігурації мейкфайлу. Якщо операція пройшла успішно, то виконуємо побудову виконуваного файлу. Для цього виберіть пункт Tools→[WinAVR] Make all. Якщо результат компілювання успішний, і є таке повідомлення “Process Exit Code: 0”, це означає, що необхідний hex-файл був побудований без помилок. Інакше – помилки знаходяться у тексті програми. Усю необхідну інформацію про місце виникнення помил-

ки можна дізнатися з вікна “Output”. Компіляція та лінкування програми завершена. Якщо результат операції був успішним, то в директорії проекту з’являться такі нові файли:

- simple.eep – містить інформацію, що буде скопійована в EEPROM;
- simple.elf – об’єктний файл. Саме на основі інформації цього файлу буде створений виконуваний файл;
- simple.hex – виконуваний файл. Програма в кодах МК;
- simple.lss – містить інтерпретацію відкомпільованої програми мовою Асемблер;
- simple.map, simple.sym – допоміжні файли, що описують хід компілювання та лінкування проекту.

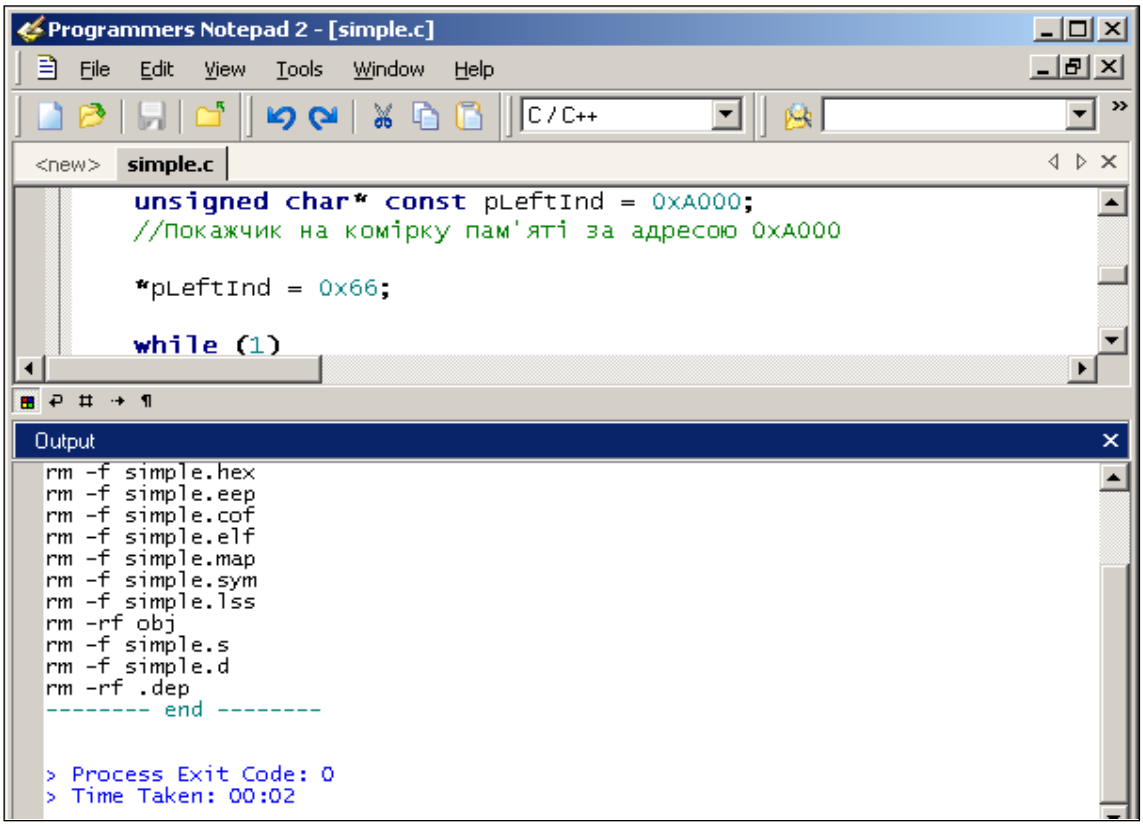

Рисунок 1.12 – Результати роботи операції "make.exe" clean

1.4 Робота в інтегрованому середовищі розробки ICCAVR

Середовище програмування ICCAVR (рис. 1.13) фірми ImageCraft Creations Inc призначена для розробки програмного забезпечення (проекту) та його компілювання е виконуваний AVR-мікроконтролерів файл. ICCAVR є одним з найпростіших і дуже зручних компіляторів. Він має набір стандартних бібліотечних функцій і низку підпрограм, спеціально призначених для AVR-мікроконтролерів. Їх можна використовувати у своєму проекті, попередньо підключивши їх до проекту директивою #include. У роботі ICCAVR використовуються такі типи файлів:

– C – вихідний текст мовою C;
– S – вихідний текст мовою Assembler, що генерується для кожного вихідного C-файлу;
– H – заголовки (header) файл;
– PRJ – файл проекту;
– SRC – список файлів проекту;
– O – об'єктний файл, який утворюється після компілювання асемблерного файлу;
– HEX – вихідний файл у форматі Intel HEX для завантаження у мікроконтролер;
– EEP – вихідний файл у форматі Intel HEX для завантаження в ПЗП даних мікроконтролера;
– COF – вихідний файл у форматі COFF, використовується при налаштуванні проекту в AVR Studio або інших середовищах програмування;
– LST – файл-лістинг, що містить інформацію про адреси;
– MP – MAP-файл, який містить символічну інформацію;
– DBG – файл з налаштування;
– A – бібліотечний файл.

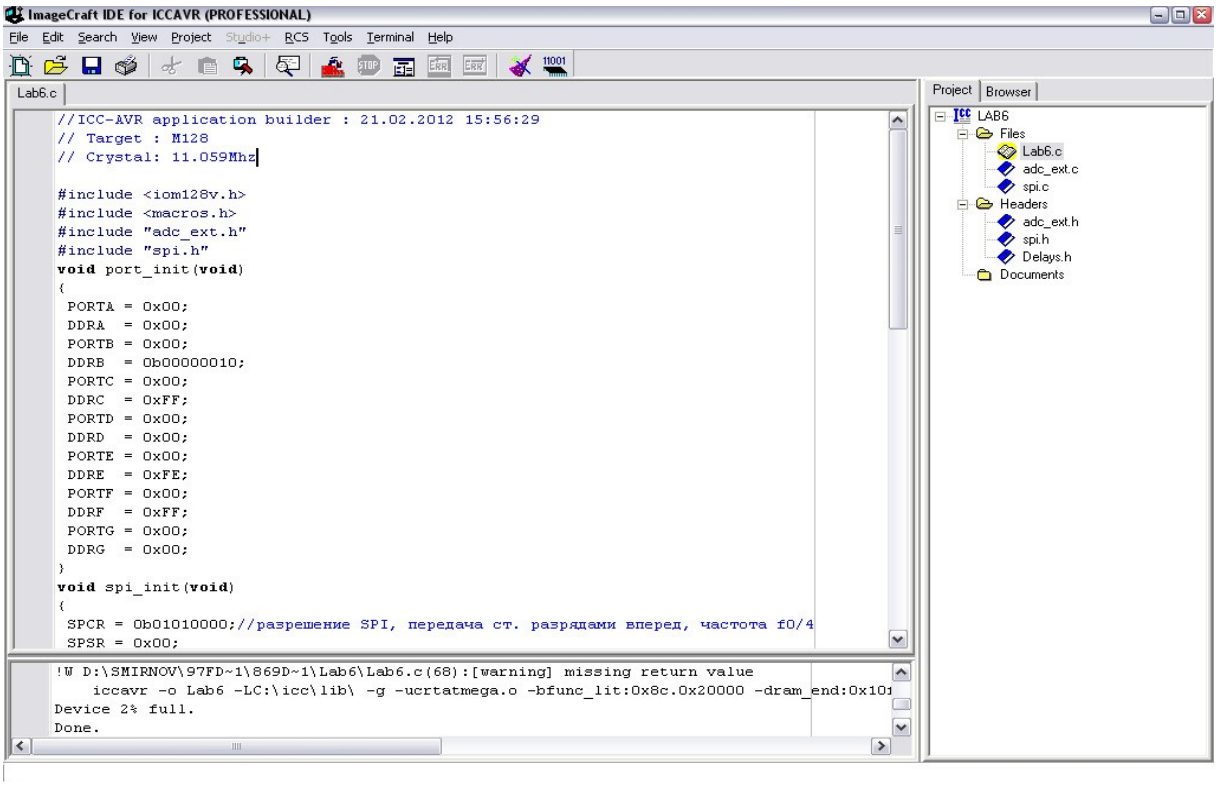

Рисунок 1.13 – Головне вікно ICCAVR

Після запуску ICCAVR на екрані монітора з'являється головне вікно, яке розділене на три частини (вікна). Ліва верхня частина – вікно редактора. У ньому представлені тексти програм, які утворюють проект. Саме з цими текстами і працює програміст, редагуючи їх або додаючи нові рядки та блоки програм. При першому запуску ICCAVR вікно, як правило, порожнє.

Права частина – вікно менеджера проекту, що містить дві вкладки: Project та Browser. Перша містить список файлів проекту, кожен з яких можна відкрити у вікні редактора подвійним клацанням миші. Друга вкладка – оглядач коду, що показує список функцій та змінних, визначених у проекті. Нижня частина – вікно стану. У ньому показуються результати компілювання окремого файлу або всього проекту в цілому. Якщо є помилки, то після компілювання з'являться повідомлення про виявлені помилки із зазначенням їхнього місця розташування і підказкою про тип помилки. Усі свої дії з управління проектом або окремими файлами користувач здійснює, використовуючи пункти меню у верхній частині головного вікна, кнопки панелі інструментів (під пунктами меню) або праву кнопку миші й спливаюче при цьому контекстне меню.

Меню Project дозволяє відкрити будь-який проект або створити новий; відкрити або закрити всі файли проекту; відкрити один з останніх проектів, з яким працював користувач; закрити проект або зберегти його під новим ім'ям; додати відкритий у вікні файл у проект або видалити з проекту файл, який обраний у вікні менеджера проекту. Пункт Make Project дозволяє здійснити компіляцію відкритого у вікні редактора файлу, а пункт Rebuild All – компіляцію всіх файлів проекту. Важливим пунктом є Option, який відкриває діалогове вікно (рис. 1.14). У пункті Target у вікні Device Configuration необхідно задати мікроконтролер, для якого розробляється проект, наприклад ATmega128/CAN128 (рис. 1.14). У пункті Paths необхідно встановити шляхи до бібліотечних файлів і файлів, які користувач підключає до проекту, наприклад, для бібліотечних файлів c:\icc\lib\. У пункті Compiler у вікні Output Format встановити формат вихідних файлів – COFF/HEX.

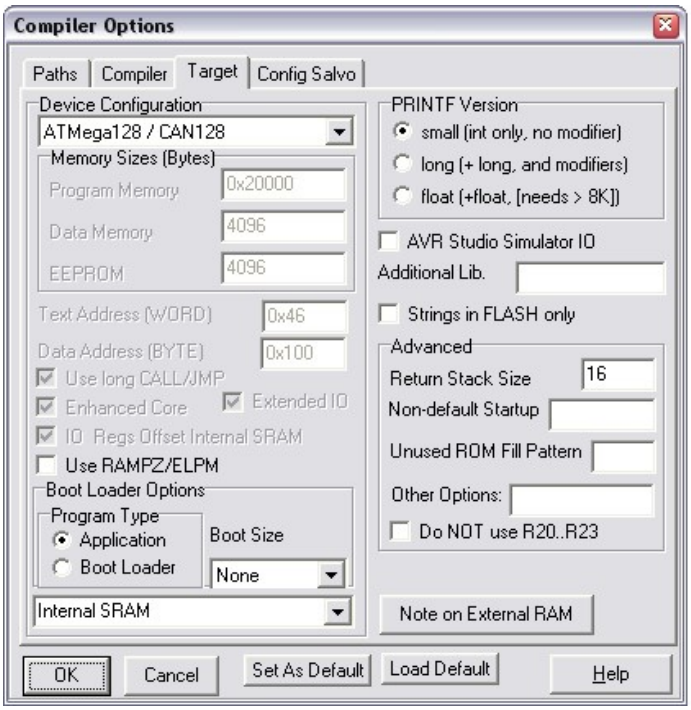

Рисунок 1.14 – Вікно при виборі пункту меню Project→Option

У пункті Tools головного вікна важливими є три пункти: Editor Options, Application Builder і In System Programmer. Пункт Tools→Editor Options дозволяє встановити необхідні параметри для редактора тексту. Тут краще все залишити без змін. Але якщо виникає необхідність писати коментарі в програмі російською мовою, то в пункті Tools→Editor Options→Highlighting необхідно у вікні Characters установити опцію Russian. Пункт Tools→Application Builder призначений для ініціалізації периферійних пристроїв мікроконтролера, а також для налаштування системи зовнішніх переривань. Налаштування периферійних пристроїв можна виконати і вручну, присвоївши в програмі відповідним регістрів управління і статусу потрібні значення. Однак за допомогою модуля Application Builder ця ініціалізація здійснюється набагато швидше і надійніше. Вікно Application Builder з опціями налаштування портів введення/виведення зображено на рис. 1.15.

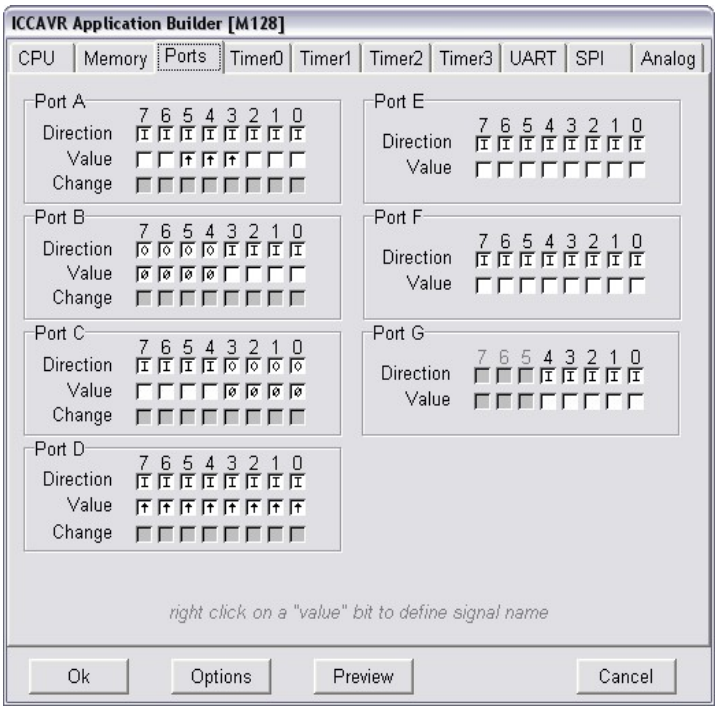

Рисунок 1.15 – Вікно Application Builder
з опціями налаштування портів введення/виведення

Як приклад на рис. 1.15 показана ініціалізація портів A, B, C і D, а для портів E, F і G – стан регістрів напрямку DDRx та даних PORTx (x = E, F, G) залишено без змін. Видно, що всі виводи портів (за винятком 4-х старших у порту B і 4-х молодших у порту C) налаштовані на вхід. До того ж, у виводів 3, 4 та 5 порту A та у всіх виводів порту D підключені резистори PushUp. У результаті Application Builder згенерує програмний код з функцією void port_init (void), при виконанні якої у всіх регістрах DDRx і PORTx будуть присвоєні значення, що відповідатимуть заданим користувачем встановленням (рис. 1.15).

Так, за допомогою Application Builder можна налаштувати роботу системи зовнішніх переривань, таймерів/лічильників, усіх інтерфейсів (USART, SPI, TWI), АЦП, компаратора, а також вирішити низку інших завдань, пов'язаних з роботою пам'яті EEPROM, сторожового таймера. Якщо дозволені якісь переривання, наприклад, від таймерів або USART, то Application Builder генерує програмний код, що містить шаблони функцій для оброблення цих переривань.

Пункт Tools→In System Programmer призначений для «прошивки» HEX-коду програми в пам'ять мікроконтролера за допомогою програматорів.

Розглянемо послідовність дій при створенні нового проекту. Для початку створимо робочу папку, в якій будуть зберігатися всі файли проекту. Потім виберемо пункт меню Project→New і у вікні, задаємо ім'я проекту та шлях до нього. У вікні менеджера з'явиться ім'я нового проекту і порожні папки для зберігання С-файлів, Н-файлів та файлів для документів, наприклад, у текстовому форматі.

Далі необхідно провести налаштування проекту, установивши за допомогою Project→Options→Target у вікні Device Configuration потрібний тип мікроконтролера, наприклад, ATMega128/CAN128. За допомогою Project→Options→Paths можна у вікні Library Path установити шлях до стандартних бібліотечних файлів, а у вікні Include Paths – шляхи підключення файлів до проекту. Ці файли знаходяться в папках icc\lib та icc\include, які, насамперед, знаходяться в папці, де встановлений компілятор ICCAVR. Якщо це не перший проект, який створюється на такому комп'ютері, то правильні шляхи до бібліотек вже встановлені, треба тільки це перевірити.

Підключаємо файл з вихідним кодом, попередньо обравши пункт File→New. Можна написати вихідний код безпосередньо у вікні редактора. У цьому випадку дуже корисна комбінація клавіш Ctrl + J, яка відкриває вікно з шаблонами операторів. Використання цієї комбінації клавіш допоможе уникнути синтаксичних помилок при розробці програм.

Якщо текст програми (С-файл) вже є і його треба лише відредагувати, то можна відкрити цей файл та зробити необхідні виправлення. Якщо програма складна, то варто скористатися Tools→Application Builder. Він здійснить початкову ініціалізацію периферійних пристроїв та згенерує початковий фрагмент програмного коду, який користувач може допрацьовувати на свій розсуд. Необхідно пам'ятати, що після виклику Application Builder в пункті CPU необхідно встановити тип мікроконтролера, наприклад, M128 (скорочення від ATMega128) та частоту синхронізації Xtal speed, наприклад, 11.059MHz. Усі інші периферійні пристрої налаштовуються (або не налаштовуються) відповідно до вирішуваних в такій програмі завданнями. Зазвичай, цей фрагмент коду закінчується функцією init_devices. Ім'я цієї функції необхідно вставити в текст головної функції програми main однією з перших (відразу після оголошення змінних у

функції main), так як саме з ініціалізації периферійних пристроїв починається виконання програми. Варто пам'ятати, що в програмі обов'язково має бути головна функція з ім'ям main, інакше компілятор буде видавати повідомлення про помилку.

Після створення вихідного файлу, розробленого мовою С, його необхідно зберегти в робочій папці. Після цього потрібно додати файл до проекту, використовуючи, наприклад, праву кнопку миші й контекстне меню. У результаті у вікні менеджера проекту в папці Files з'явиться ім'я першого файлу проекту. До проекту можна додавати нові С-файли або Н-файли, попередньо обравши папку Headers. Так, створюється проект, що містить кілька файлів. Така модульна структура проекту дозволяє краще сприймати його алгоритм функціонування. Для контролю помилок при розробці програми необхідно періодично здійснювати компіляцію окремого файлу або всіх файлів проекту. Успішне компілювання закінчується повідомленням Done, неуспішне – повідомленнями про помилки та додатковою інформацією про суть цих помилок і їхнє місцезнаходження в програмі.

1.5 Робота в інтегрованому середовищі розробки CodeVisionAVR

CodeVisionAVR – інтегроване середовище розробки програмного забезпечення для мікроконтролерів набору Atmel AVR (рис. 1.16).

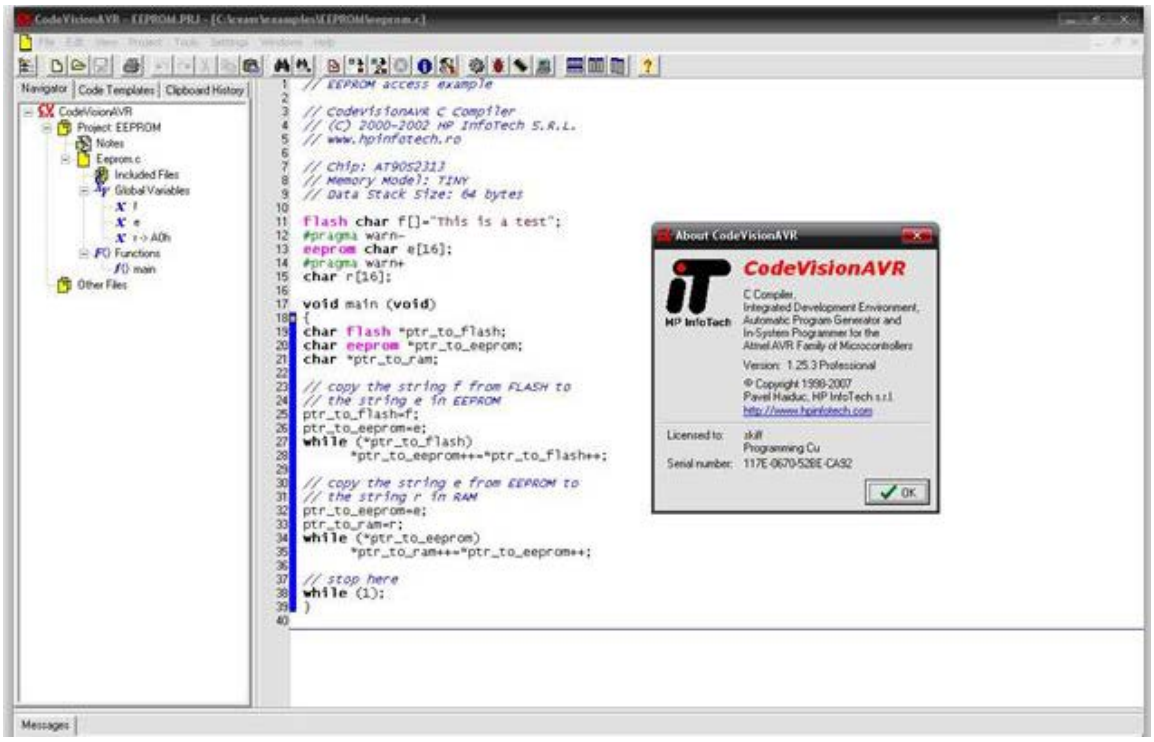

Рисунок 1.16 – Вікно програми CodeVisionAVR

CodeVisionAVR містить такі компоненти: компілятор мови C для AVR; компілятор мови асемблер для AVR; генератор початкового коду програми, що дозволяє провести ініціалізацію периферійних пристроїв; модуль взаємодії налагоджування платою STK-500; модуль взаємодії з програматором; термінал. Вихідними файлами CodeVisionAVR є: HEX, BIN або ROM-файл для завантаження в мікроконтроллер за допомогою програматора; COFF-файл, що містить інформацію для усунення несправностей; OBJ-файл. Середовище CodeVisionAVR є дуже популярним середовищем програмування мікроконтролерів AVR. Проект можна створювати як з нуля, так й за допомогою майстру коду (Code Wizard AVR).

Для створення проекту запускаємо середовище CodeVisionAVR C Compiler та обираємо меню File→New. У формі Create New File обираємо Project (проект) та на пропозицію створити код за допомогою майстра – обираємо No (рис. 1.17). Зберігаємо проект як Prog1. Обираємо мікроконтролер, визначаємо тактову частоту (рис. 1.18).

Рисунок 1.17 – Створення нового проекту в CodeVisionAVR

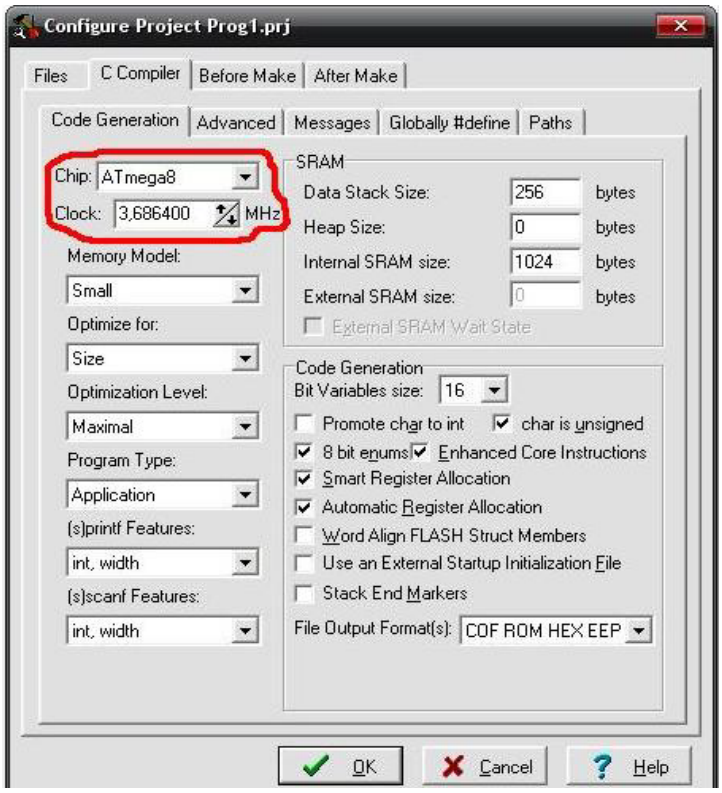

Рисунок 1.18 – Вікно налаштування властивостей проекту

Список File Output Format(s) (див. рис. 1.18) визначає, які файли будуть створені при компілюванні проекту. Цікавлять два файли: файл HEX, який записується в мікроконтролер, та файл COF, що відкривається в середовищі AVR Studio та за допомогою симулятора дозволяє проаналізувати роботу програми. Знов обираємо File→New та File Type→Source. З'явилось пусте вікно коду, зберігаємо його як Prog1.c. Відкриваємо вікно Configure Project та на вкладці Files додаємо створений файл Prog1.c (рис. 1.19).

Рисунок 1.19 – вікно Configure Project

Далі у вікні коду набираємо код програми залежно від задачі, що розв'язується (рис. 1.20).

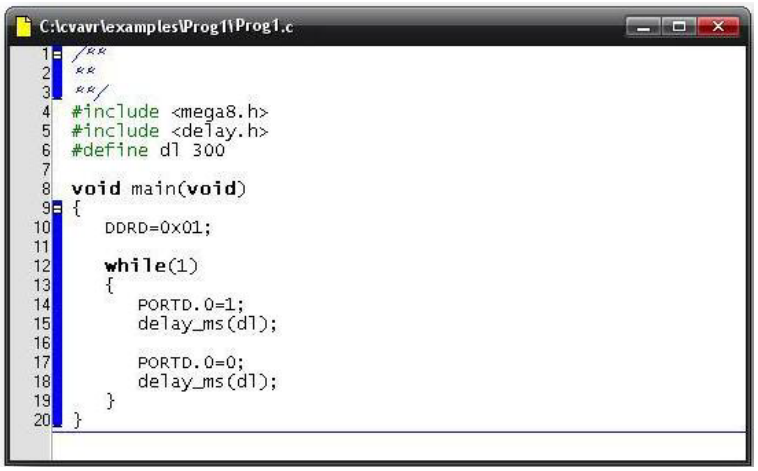

Рисунок 1.20 – Вікно коду програми Prog1.c.

Компілюємо проект, для виявлення можливих помилок та попереджень компілятора. У процесі компілювання створюється початковий файл асемблера Prog1.asm, який можна відкрити редактором для перегляду та внесення необхідних змін. На стадії компілювання файли COF, ROM, HEX, EEP не створюються. Виконуємо остаточне збирання проекту, при