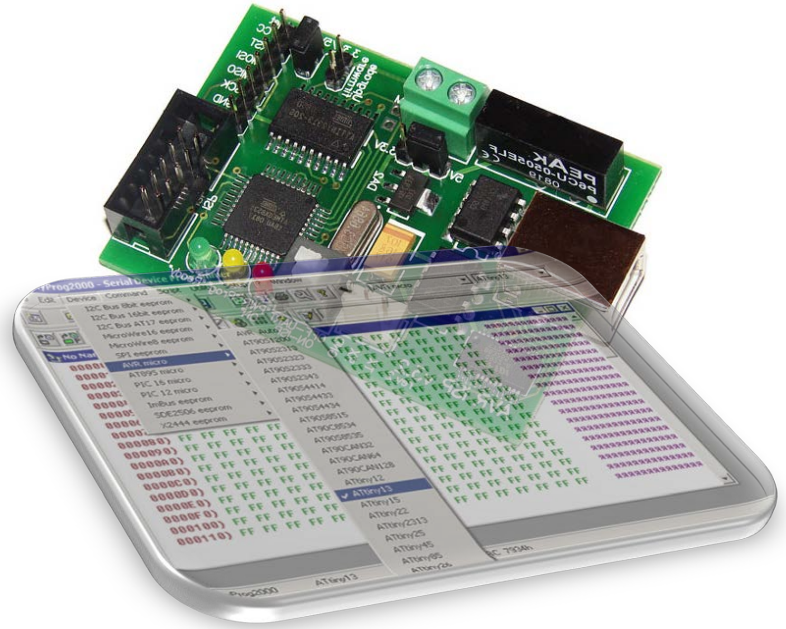
ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR



Міністерство освіти і науки України
Вінницький національний технічний університет

ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR

Навчальний посібник

Вінниця
ВНТУ
2018

УДК 004.451:004.354.7–022.513](075.8)
П78

Автори:
С. М. Цирульник, О. Д. Азаров, Л. В. Крупельницький, Т. І. Трояновська

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 7 від 22.02.2018 р.)

Рецензенти:
Л. Б. Ліщинська, доктор технічних наук, професор
В. А. Лужецький, доктор технічних наук, професор
А. А. Яровий, доктор технічних наук, професор

Програмування мікроконтролерів AVR : [навчальний посібник] / С. М. Цирульник, О. Д. Азаров, Л. В. Крупельницький, Т. І. Трояновська. – Вінниця : ВНТУ, 2018. – 111 с.
П78

У навчальному посібнику комплексно розглянуто та актуалізовано апаратно-програмні засоби підтримки мікроконтролерів AVR, що має вагоме значення для розробників вбудованих систем.

Посібник призначено для студентів спеціальностей 123 – «Комп'ютерна інженерія» та 125 – «Кібербезпека» під час вивчення дисциплін «Архітектура комп'ютерів», «Комп'ютерна схемотехніка» та суміжних апаратних дисциплін.

УДК 004.451:004.354.7–022.513](075.8)

ЗМІСТ

ВСТУП

Мікроконтролери (МК) можна зустріти практично в будь-якому технічному завданні, де потрібно вирішувати завдання вимірювання, оброблення інформації та управління. Це може бути електропобутова техніка, вимірювальні прилади або засоби комунікації, а також такі складні об'єкти управління, як автомобілі або літаки. Коло таких завдань дуже широке, починаючи від увімкнення звукової або світлової сигналізації та закінчуючи складною математичною обробкою й аналізом інформації, що надходить від різних давачів, з подальшим формуванням керуючих впливів на об'єкт управління.

Одним з найбільш популярних наборів мікроконтролерів, які застосовуються в системах оброблення даних, контролю і управління, є пристрої, що випускаються фірмою Microchip та відомі під абревіатурою PIC. Однак, останнім часом корпорація Atmel (США) стала виробляти набір мікросхем на ядрі AVR, яке має більш досконалу архітектуру й забезпечує мікроконтролерам цього набору високу швидкодію та низьке енергоспоживання, що дає їм відчутну перевагу, порівняно з контролерами PIC [15, 22, 24]. До того ж, цінова політика корпорації Atmel є більш привабливою для розробників таких систем. Порівняно з пристроями PIC, МК AVR мають більш розвинену систему команд, що налічує до 133 інструкцій, їхня продуктивність наближається до 1 MIPS/МГц, а flash-пам'ять програм має можливість внутрішньосхемного програмування. Архітектура ядра AVR оптимізована так, що дозволяє використовувати мову високого рівня С. На відміну від пристроїв PIC, де одна операція виконується за 4 такти, МК AVR можуть виконувати команди в кожному такті, тобто в 4 рази швидше на тій же тактовій частоті. А що менша частота, то менше енергоспоживання мікроконтролера.

Серед великих виробників мікроконтролерів потрібно виокремити такі фірми, як: Texas Instruments, Dallas Semiconductor, Philips, STMicroelectronics, Mitsubishi Electronics та ін. Однак корпорація Atmel є нині одним з визнаних світових лідерів у виробництві виробів сучасної мікроелектроніки. Вона добре відома на світовому ринку електронних компонентів. Заснована в 1984 році фірма Atmel визначила для своєї продукції такі сфери додатків: телекомунікації та мережі, обчислювальна техніка, вбудовані системи контролю та управління, побутова техніка та автомобільний транспорт [15, 22]. Тому методи та засоби програмування мікроконтролерів AVR є актуальними.

У навчальному посібнику комплексно розглянуті та актуалізовані апаратно-програмні засоби підтримки мікроконтролерів AVR, що має вагоме значення для розробників вбудованих систем.

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.

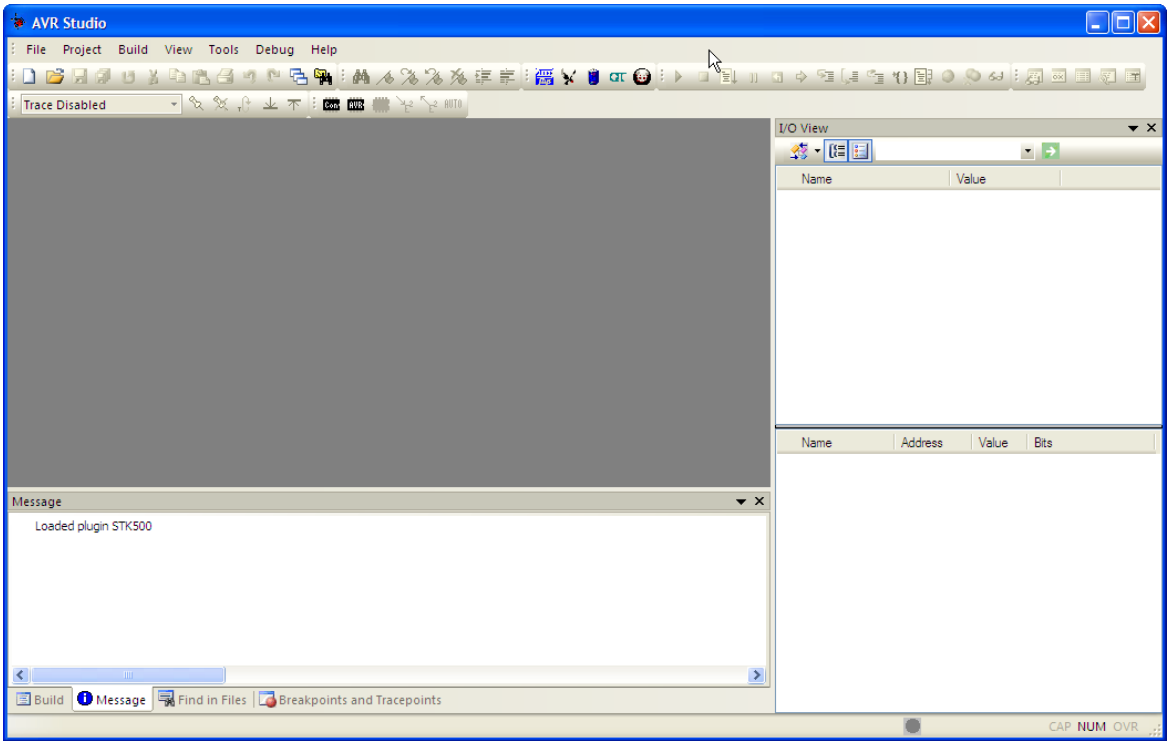


Рисунок 1.1 – Основне вікно програми AVR Studio

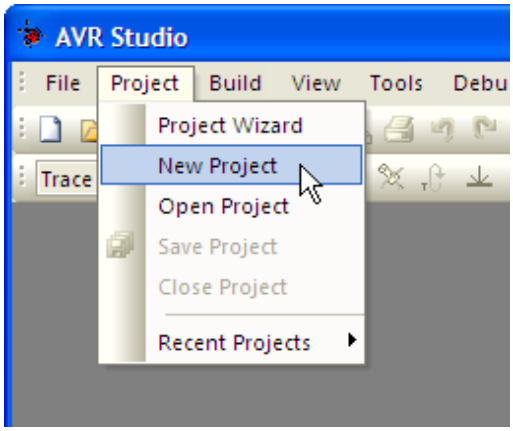


Рисунок 1.2 – Створення нового проекту
у програмі AVR Studio

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).

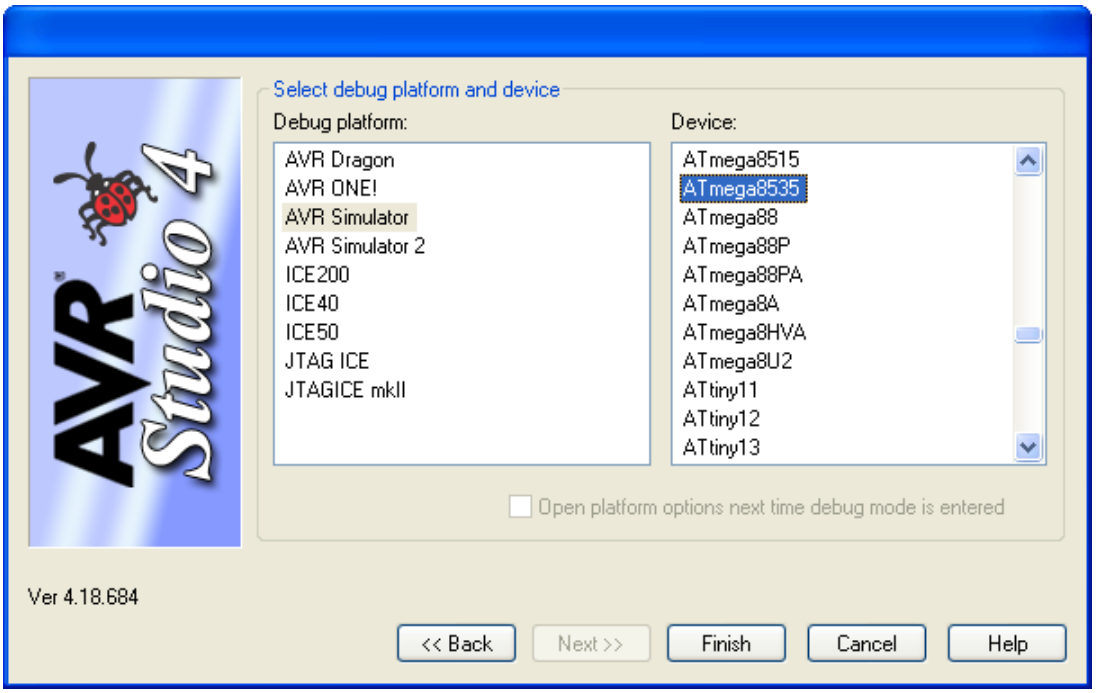


Рисунок 1.3 – Вибір платформи відлагодження та типу мікроконтролера у середовищі AVR Studio

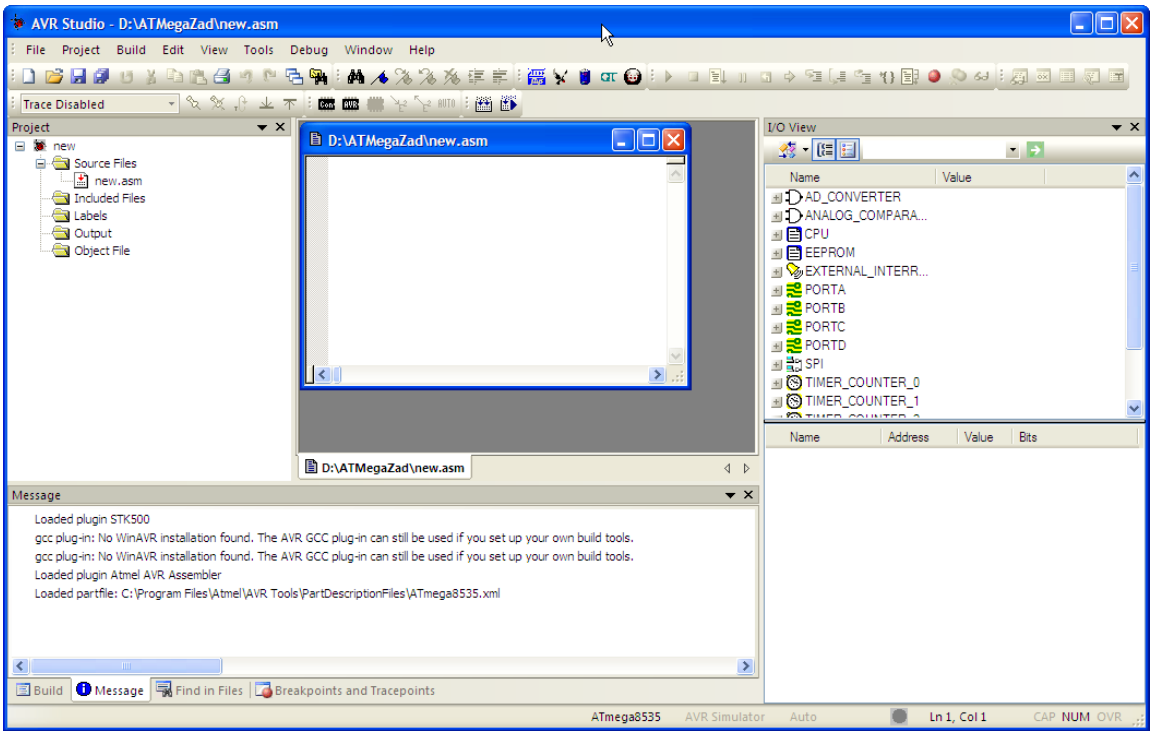


Рисунок 1.4 – Вікно нового проекту у програмі AVR Studio

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).

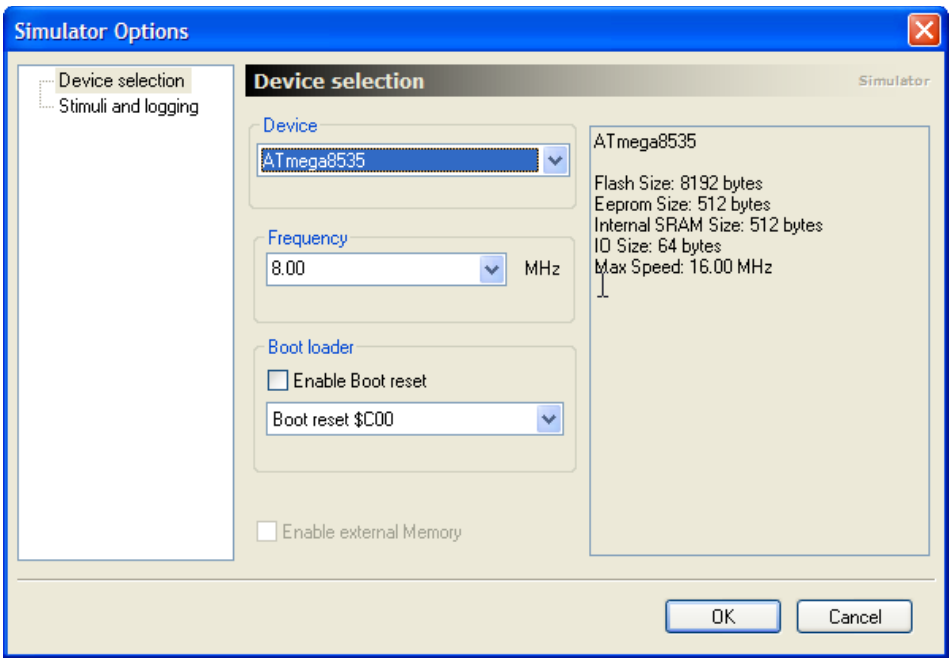


Рисунок 1.5 – Опції емулятора (Simulation Options) середовища AVR Studio

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.

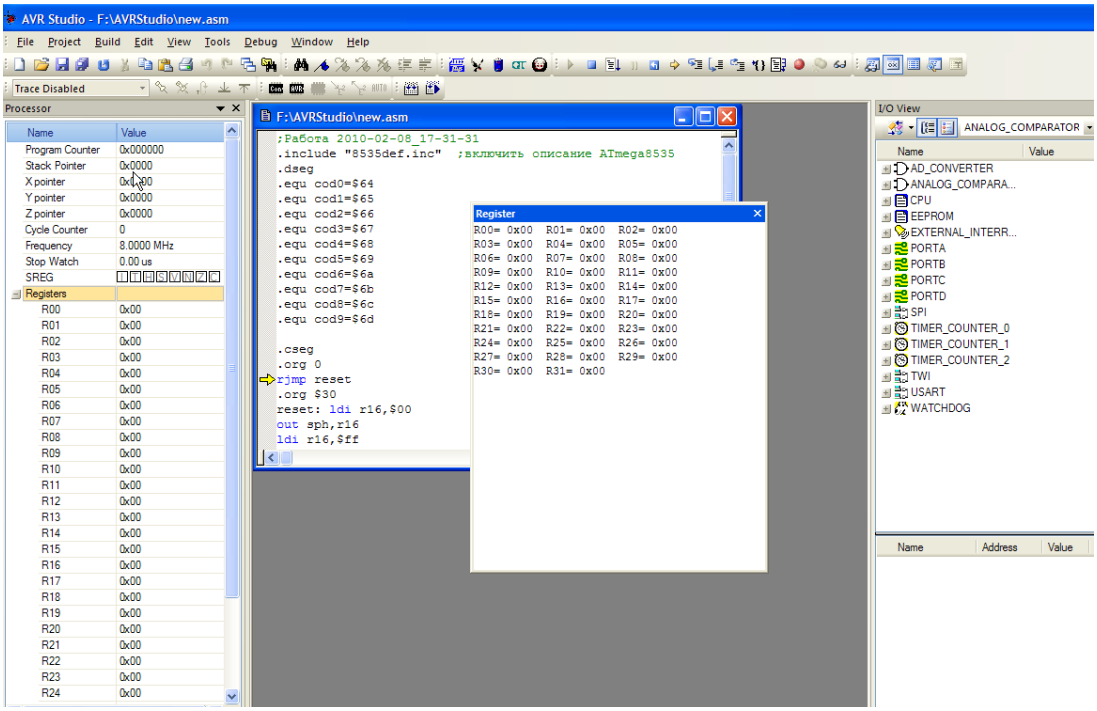


Рисунок 1.6 – Вікно емуляції роботи програми в AVR Studio

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).

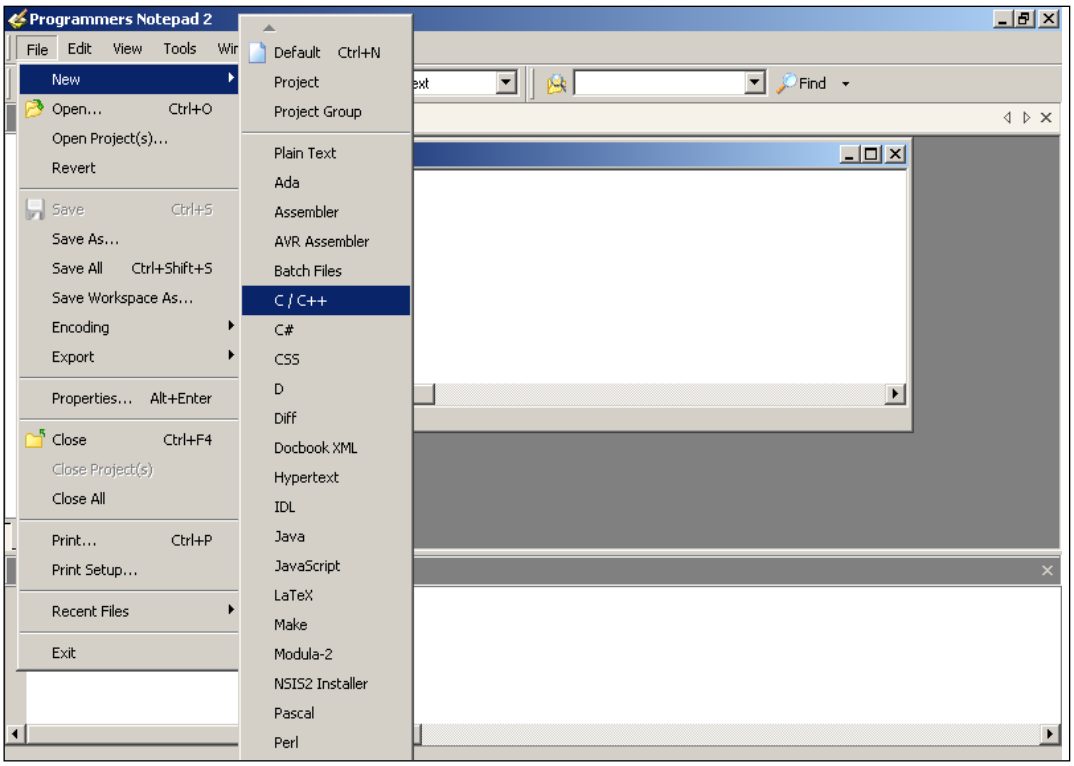


Рисунок 1.7 – Створення нового файлу
в редакторі Programmers Notepad

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.

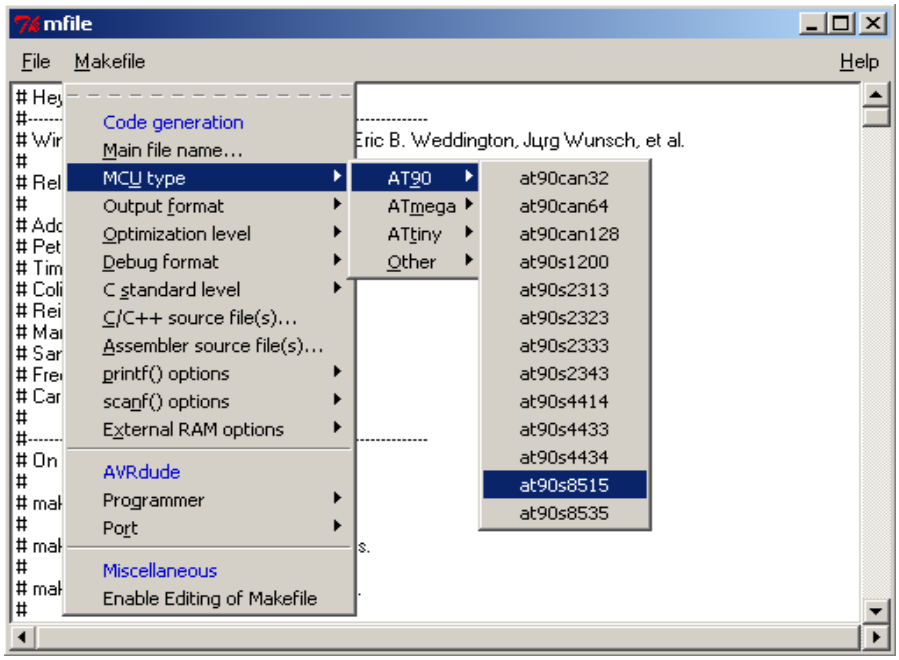


Рисунок 1.8 – Вибір типу мікроконтролера для створення makefile

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.

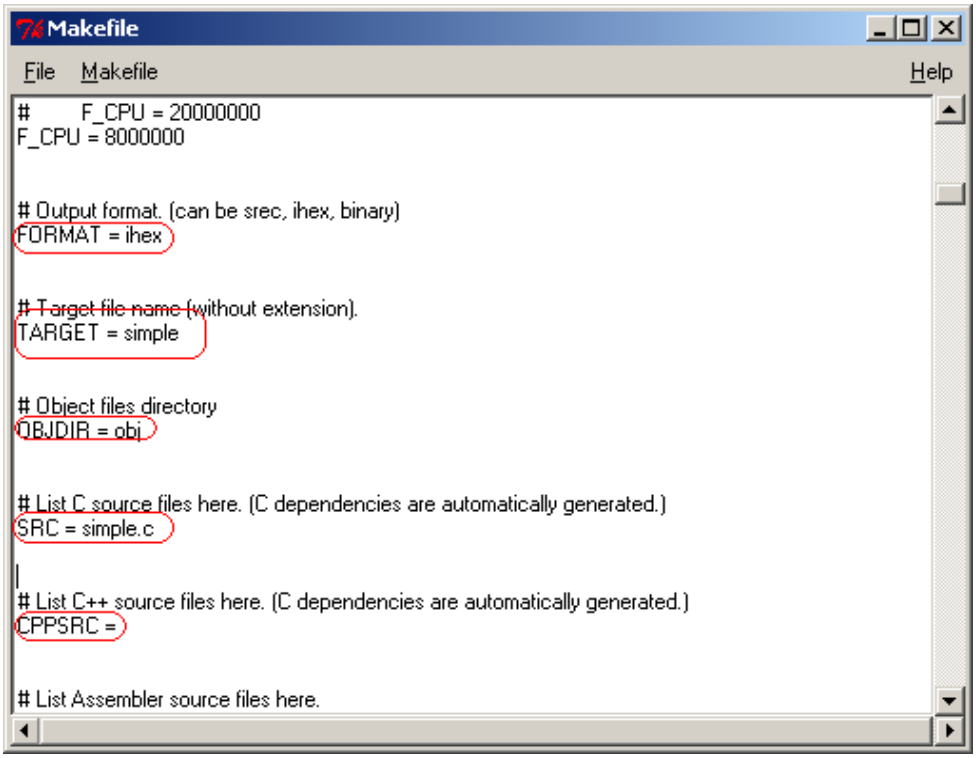


Рисунок 1.9 – Налаштування makefile

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.

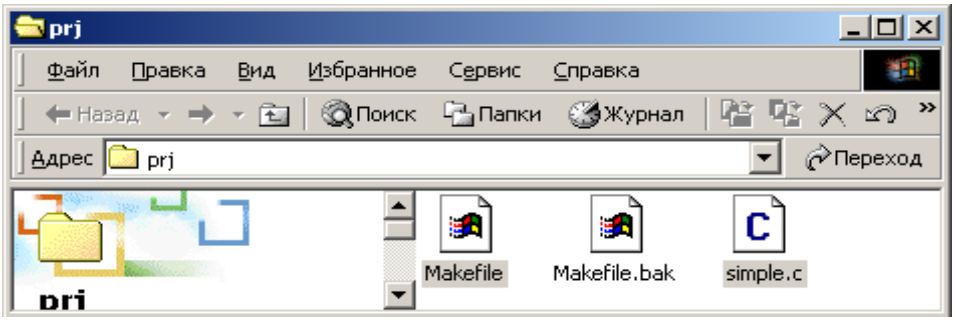


Рисунок 1.10 – Файли проекту

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).

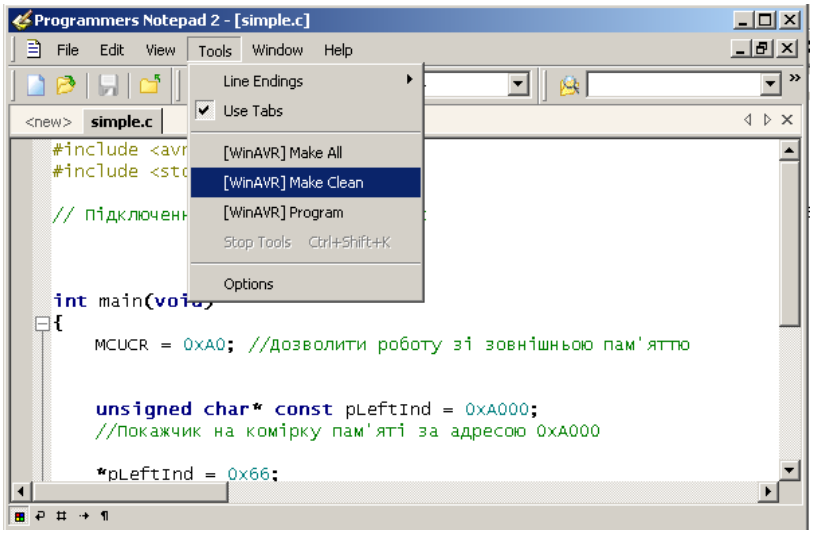


Рисунок 1.11 – Виконання make clean

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.

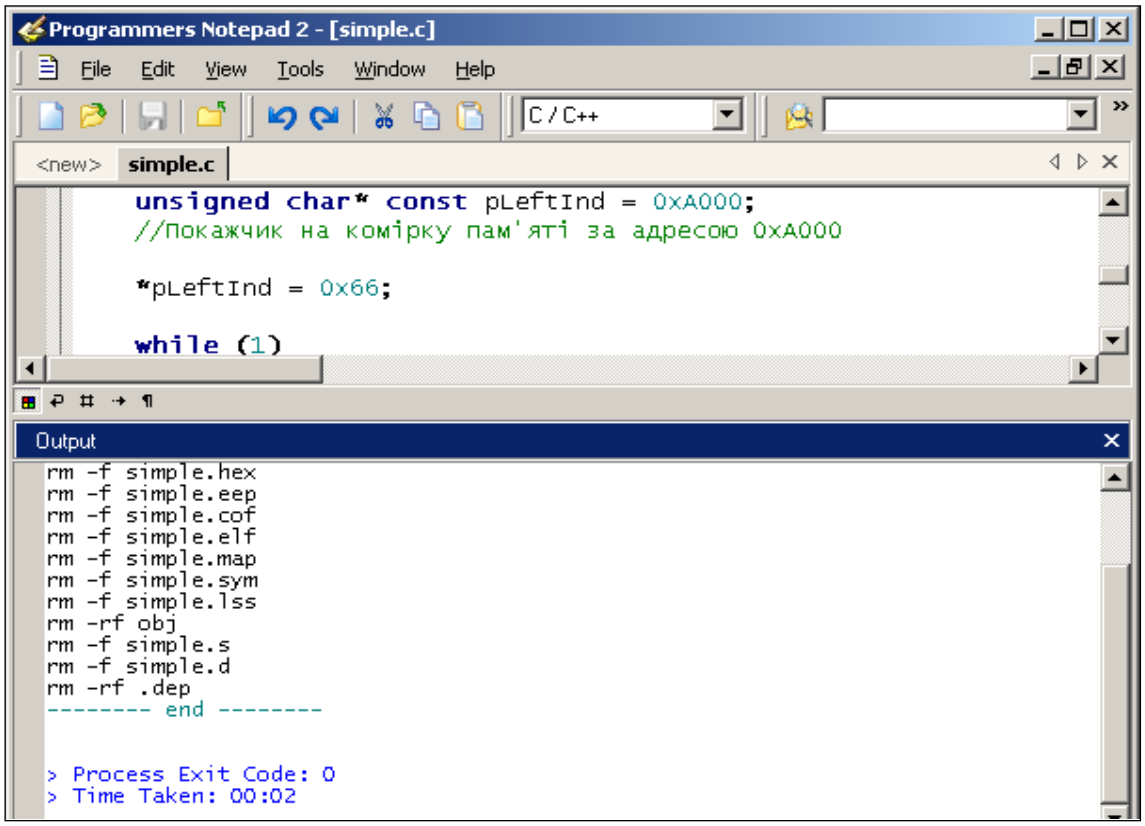


Рисунок 1.12 – Результати роботи операції "make.exe" clean

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.

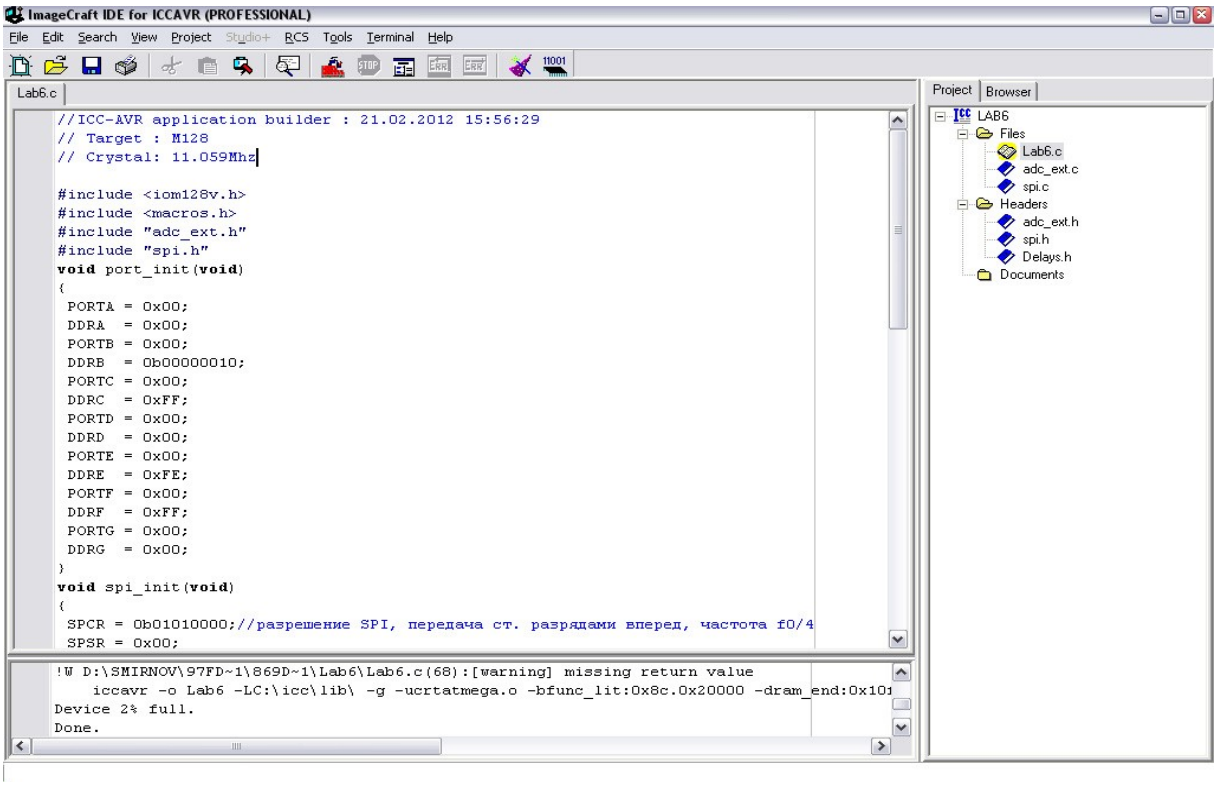


Рисунок 1.13 – Головне вікно ICCAVR

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.

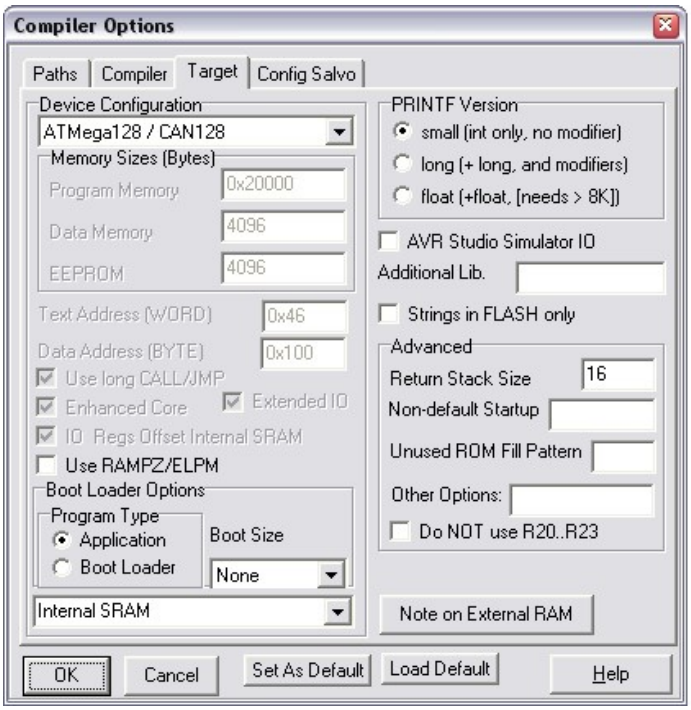


Рисунок 1.14 – Вікно при виборі пункту меню Project→Option

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.

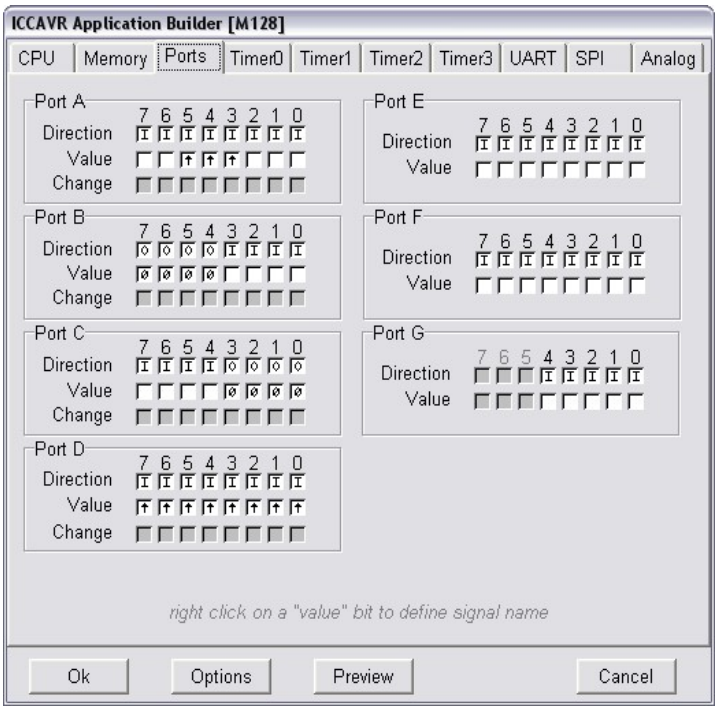


Рисунок 1.15 – Вікно Application Builder
з опціями налаштування портів введення/виведення

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).

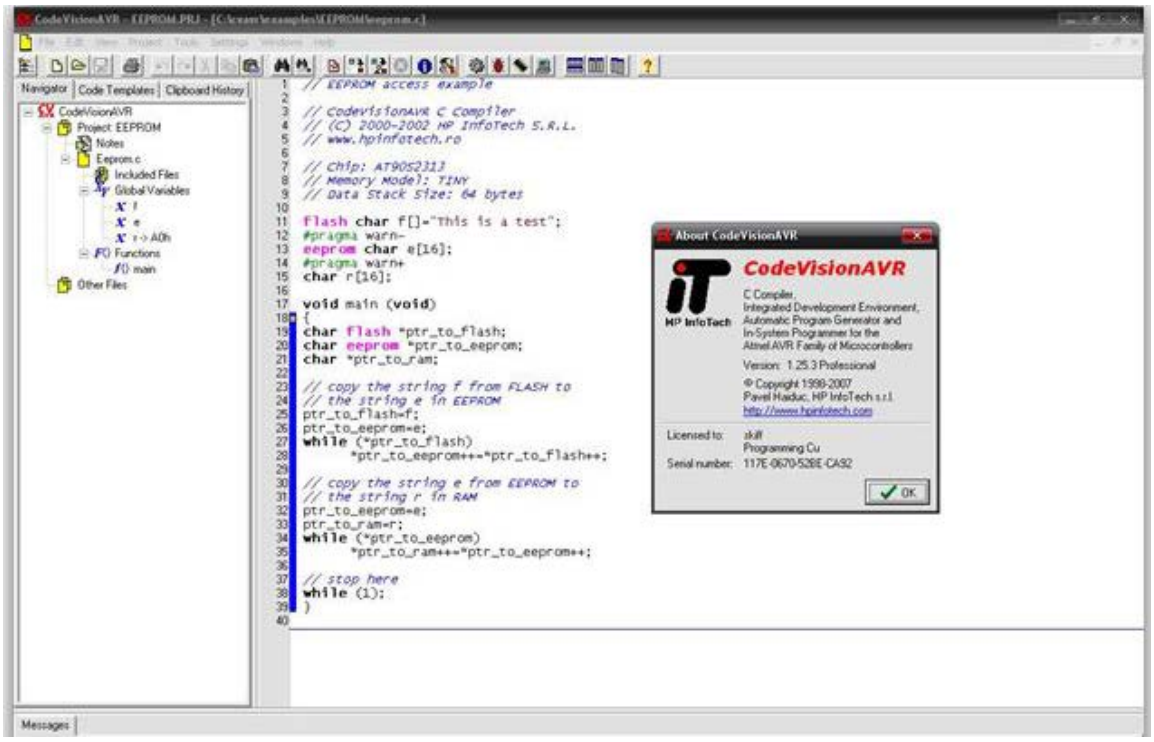


Рисунок 1.16 – Вікно програми CodeVisionAVR

CodeVisionAVR містить такі компоненти: компілятор мови С 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

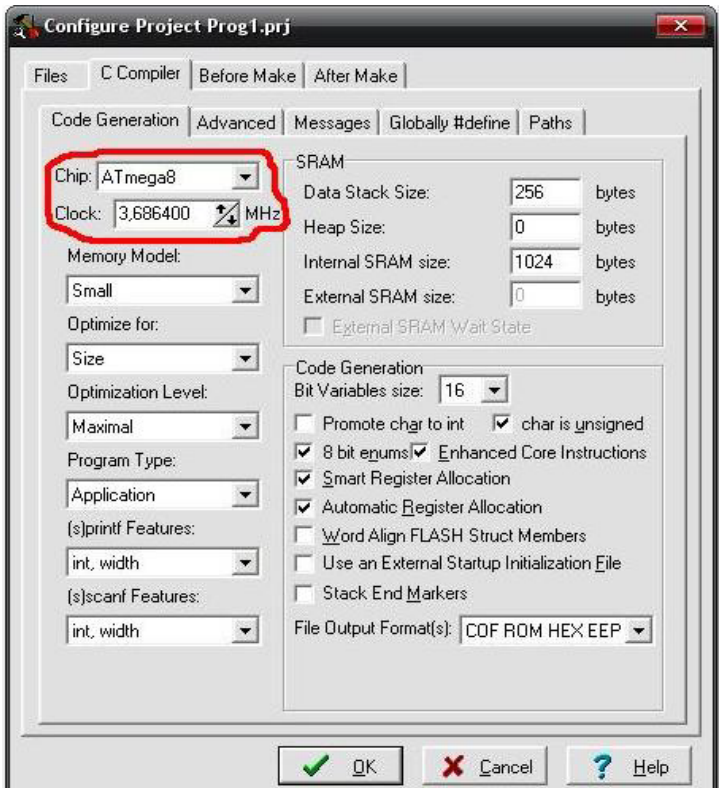


Рисунок 1.18 – Вікно налаштування властивостей проекту

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).

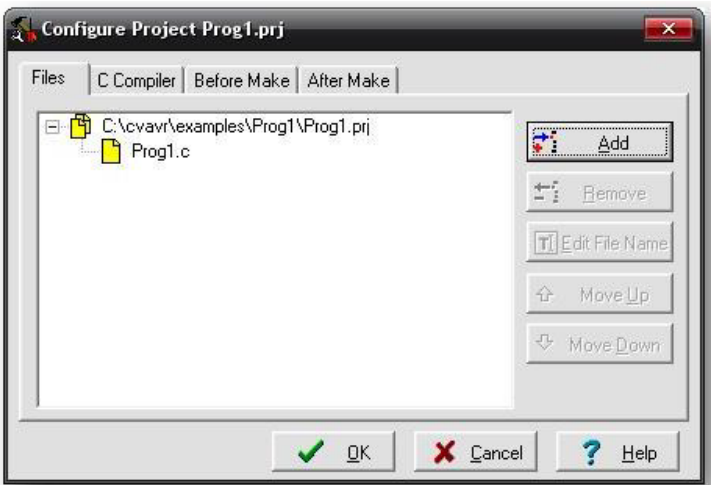


Рисунок 1.19 – вікно Configure Project

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

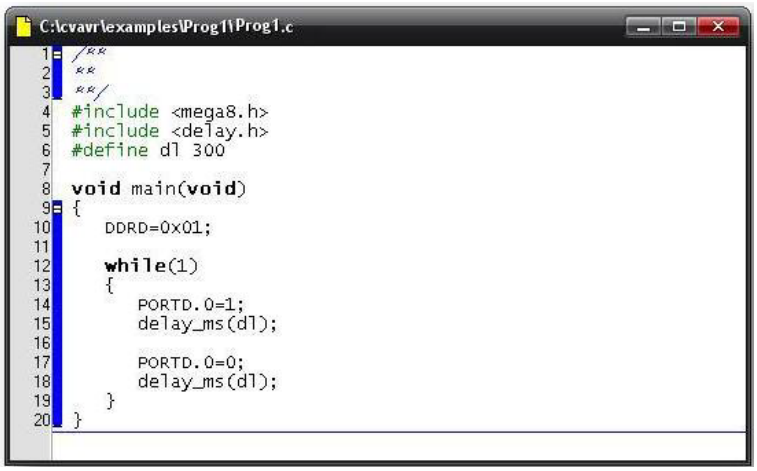


Рисунок 1.20 – Вікно коду програми Prog1.c.

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

цьому створюються готові до використання файли COF, ROM, HEX, EEP. Якщо проект створюється за допомогою майстра коду (CodeWizardAVR), то після необхідних установок на вкладках (рис. 1.21), то в меню File майстра обираємо Generate, Save and Exit (рис. 1.22), то виконуємо необхідні збереження. Переглянути попередній код, що створює майстер, можна шляхом вибору у меню File командного рядка Program Preview!

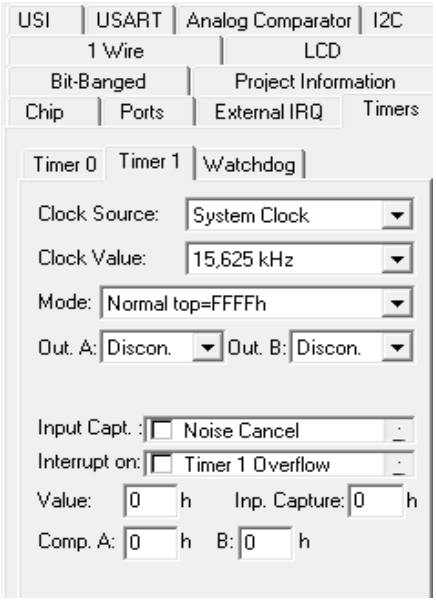


Рисунок 1.21 – Вікно налаштувань CodeWizardAVR

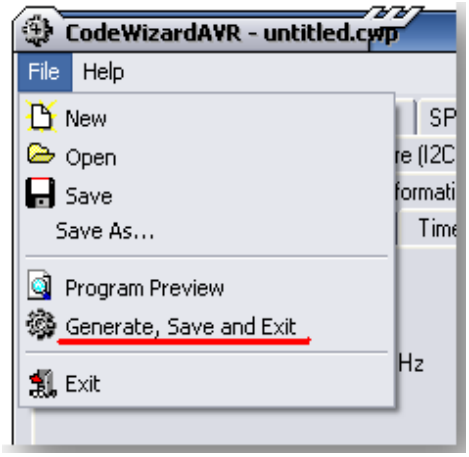


Рисунок 1.22 – Вікно налаштувань CodeWizardAVR; збереження та вихід з майстра коду

1.6 Робота в інтегрованому середовищі розробки AVR Builder

Середовище призначене для створення повного циклу розробки. Розробка програми може виконуватись як на рівні асемблера, так і на макрорівні з маніпуляцією багатобайтними величинами зі знаком. На відміну від класичного асемблера, програма вводиться у вигляді алгоритму з деревоподібними розгалуженнями і відображається на площині у двох вимірах. Мережа умовних і безумовних переходів відображається графічно у зручній векторній формі. Уся логічна структура програми стає наглядною. Візуальність логічної структури зменшує ймовірність помилок і скорочує строки розробки. Для налаштування периферійних пристроїв (таймери, UART, ADC, SPI) передбачений спеціальний елемент алгоритму – «налагоджувач» з віконним інтерфейсом. У ньому досить вибрати необхідні параметри роботи пристрою, а набір інструкцій, що забезпечують ці параметри, сформує компілятор (у правій частині вікна). Підтримується автоматичне перекодування рядків ANSI-кодів Windows у коди русифікованого буквено-цифрового LCD-індикатора [3, 27].

Середовище поєднує в собі графічний редактор, компілятор алгоритму, симулятор МК та програматор. При використанні програматора МК

підключається до COM порту комп'ютера через нескладний адаптер. Algorithm Builder забезпечує моніторне відлагодження на кристалі (On Chip debug), що дозволяє спостерігати вміст реального кристала в заданій точці зупину. При цьому, для зв'язку МК з комп'ютером використається тільки один вивід, на вибір користувача. Моніторне відлагодження може бути застосоване до будь-якого типу кристала, що має SRAM. Будь-яке програмне забезпечення можна розбити на окремі логічно завершені фрагменти. Як правило, останнім оператором цих фрагментів є такі оператори, як безумовний перехід або повернення з підпрограми, тобто оператори, після яких лінійне виконання програми однозначно припиняється. Розробка програмного забезпечення в середовищі Algorithm Builder зводиться до формування таких блоків, розміщення їх на площині й установлення між ними векторних зв'язків з умовних і безумовних переходів. Основний інтерфейс програми зображено на рис. 1.23.

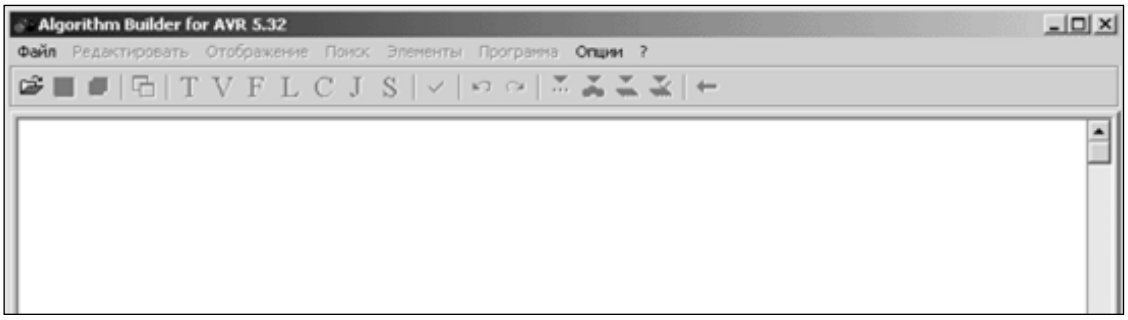


Рисунок 1.23 – Вікно програми AVR Builder

Для побудови алгоритму в Algorithm Builder передбачено 7 елементів:

– Text (рядок локального текстового редактора);
– Vertex (вершина блоку);
– Field (поле);
– Label (мітка);
– Condition (умовний перехід);
– Jmp Vector (відносний безумовний перехід);
– Setter (налагоджувач периферійних пристроїв).

Усі елементи знаходяться на панелі інструментів для швидкого доступу до них та зручного використання (рис. 1.24).

Елемент «Text» являє собою текстовий рядок, що починається від лівого краю поля алгоритму. Сукупність із декількох таких рядків утворює локальний текстовий редактор, обведений пунктирними лініями. Правила роботи в ньому, аналогічні іншим текстовим редакторам.

Рядки призначені для запису в них ряду директив компілятора, а також для коментарів. Для додавання нового рядка текстового редактора потрібно вибрати меню «Элементы→Текст», або натиснути клавіші «Alt+T». Також можна натиснути кнопку «T» на панелі інструментів. Коментарі мають позначатися двома слешами: «//».

Елемент «Label» являє собою вертикальний штрих, розташований усередині блоку операторів, і необов'язкове ім'я, що розташовується ліво-

руч або праворуч від штриха. Мітка призначена для позначення місць в алгоритмі, куди можливе здійснення умовних і безумовних переходів. Така реалізація значно спрощує розробку програми, так як код стає більш зрозумілим, ніж у звичайній програмі на Assembler.

Для додавання мітки в блок використовується пункт меню «Элементы→Метка», або ж натисканням клавіші «Alt+L», або кнопкою «L» на панелі інструментів.

Елемент «Vertex» за своїм змістом й призначенням повністю ідентичний мітці, але на відміну від неї, задає розташування блоку на робочій площині й завжди є його початком. Створення нової вершини виконується з пункту меню «Элементы→Вершина», або ж викликом гарячої клавіші «Alt+V», або натисканням кнопки «V» на панелі інструментів.

Елемент «Field» являє собою рядок у блоці. Об'єкт призначений для запису більшості операторів МК. Щоб додати поле, потрібно вибрати пункт меню «Элементы→Поле», або натиснути кнопку «F» на панелі інструментів, або комбінацією клавіш «Alt+F», або клавішею «Enter» у випадку, якщо курсор перебуває поза локальним текстовим редактором.

Елемент «Condition» призначений для реалізації умовних переходів. Являє собою овальний контур, усередині якого вписується умова переходу й можливий вектор у виді ламаної лінії зі стрілкою на кінці, біля якої можливо необов'язкове ім'я вектора. Кінець вектора має:

- закінчуватися на якій-небудь мітці;
- закінчуватися на вершині блоку;
- закінчуватися на відрізку іншого вектора;
- мати ім'я адресованої мітки.

Проведення вектору до потрібного місця відбувається натисканням лівої кнопки миші при натиснутій клавіші «Alt». Для редагування вектора використовуються клавіші напрямку в комбінації із клавішею «Alt». Щоб ввести новий об'єкт, варто обрати пункт меню «Элементы→Условие», натиснути клавіші «Alt+C» або кнопку на «C» панелі інструментів.

Елемент «JMP Vector» призначений для реалізації коротких безумовних переходів (у базовому асемблері це оператор «RJMP»). Являє собою ламану лінію, що виходить із середини блоку зі стрілкою на кінці, аналогічну вектору об'єкта «Condition». Щоб додати новий безумовний перехід, потрібно вибрати пункт меню «Элементы→Вектор б/у перехода», натиснути комбінацію клавіш «Alt+J» або кнопку «J» на панелі інструментів.

Елемент «Setter» являє собою сірий прямокутник, усередину якого вписане ім'я периферійного компонента МК такого, як таймер, АЦП, регістр маски переривань та ін. Настроювач призначений для формування послідовності операцій МК, які забезпечують завантаження необхідних констант у відповідні керуючі регістри введення/виведення відповідно до обраних властивостей.

Перед використанням цього елемента потрібно визначити тип МК. Для додавання настроювача в алгоритм потрібно обрати пункт меню «Элементы→Настройщик», натиснути клавіші «Alt+S» або кнопку «S» на панелі інструментів.

Об'єкт «Setter» є макро-оператором. Після компілювання він перетвориться в послідовність команд МК, які забезпечують завантаження необхідних констант у відповідні регістри керування. У цих операціях буде використаний регістр R16.

Для ряду компонентів, наприклад ADC, настроювач може впливати на декілька регістрів керування. У такому разі, за необхідності, вплив на кожний конкретний регістр можна заблокувати.

Для створення нового проекту в програмному середовищі AVR Builder потрібно вибрати пункт меню «Файл→Новый». При цьому в робочій області програми з’явиться елемент «Текст», «Вершина блоку» та елемент «Поле» з пустим оператором NOP (рис. 1.24).

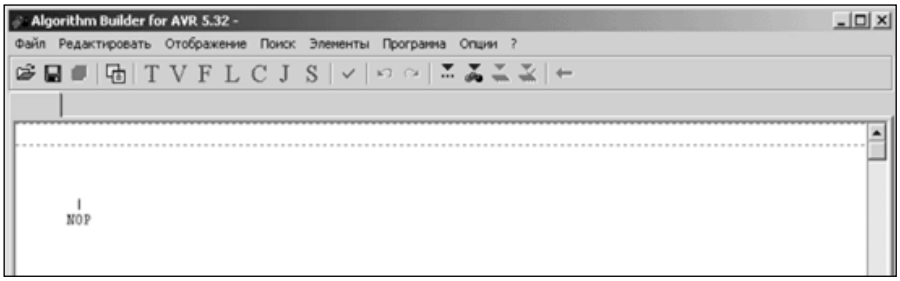


Рисунок 1.24 – Порожній проект AVR Builder

Розглянемо застосування AVR Builder на прикладі програми «бігучий вогонь». Алгоритми програми наведений на рис. 1.25. Цей алгоритм почергово виводить логічні одиниці в біти порту B. З лівої сторони відображено алгоритм основної програми, а з правої – алгоритм підпрограми затримки. Запис даних до регістрів відбувається за допомогою вказівки «->».

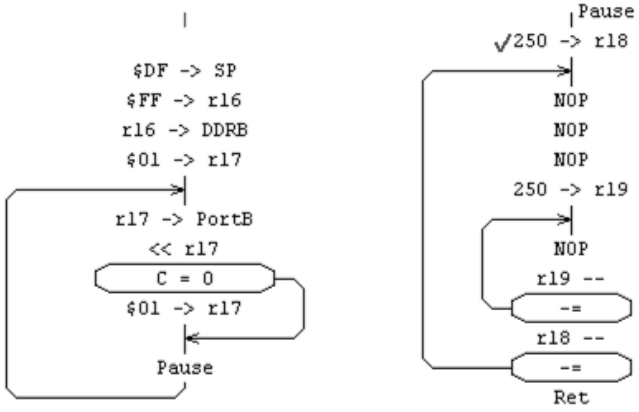


Рисунок 1.25 – Алгоритм програми «Бігучий вогонь» в AVR Builder

Для створення нового поля використовується клавіша «Enter», а для його видалення комбінацію клавіш «Ctrl+Del». Помітка біля оператора «250→r18» означає точку зупину для симулятора. Додається вона за допомогою клавіші «F5» або ж відповідною піктограмою на панелі інструментів. Після завершення роботи над алгоритмом програми необхідно визначити тип МК, після чого можна буде скомпілювати програму. Для цього потрібно скористатися пунктом меню «Опции→Опции проекта» та на закладці «Кристалл» обрати потрібний МК. Обираємо МК AT90S2313 (рис. 1.26).

Після виконання цих дій можна скомпілювати програму та запустити на виконання в симуляторі. Для цього використовується клавіша «F9». Якщо в програмі відсутні помилки, то запуститься симулятор. Якщо ж помилки є, то про це буде повідомлено.

Для спостереженням за станом МК під час виконання програми в AVR Builder передбачені вікна моніторингу (рис. 1.27). Їх можна викликати через пункт меню «Открыть». Для відстеження стану МК у цьому алгоритмі достатньо відкрити вікна «Processor», «Working Registers» та «I/O Registers→ Port B». При запуску симулятора вміст регістрів SRAM автоматично заповнюється випадковими числами, тому що в реальному часі їх вміст неможливо передбачити з появою напруги живлення. Синя мітка біля оператора (рис. 1.28) – це поточне положення програмного лічильника. Вона вказує на оператор, який буде виконаний далі. Після компілювання програми та запуску симулятора вона автоматично встановлюється біля першого оператора.

Рисунок 1.26 – Опції проекту AVR Builder

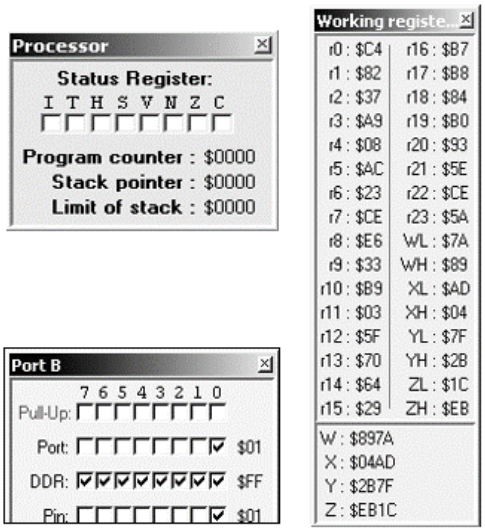


Рисунок 1.27 – Вікно стану процесора, стану регістрів, стану порту B

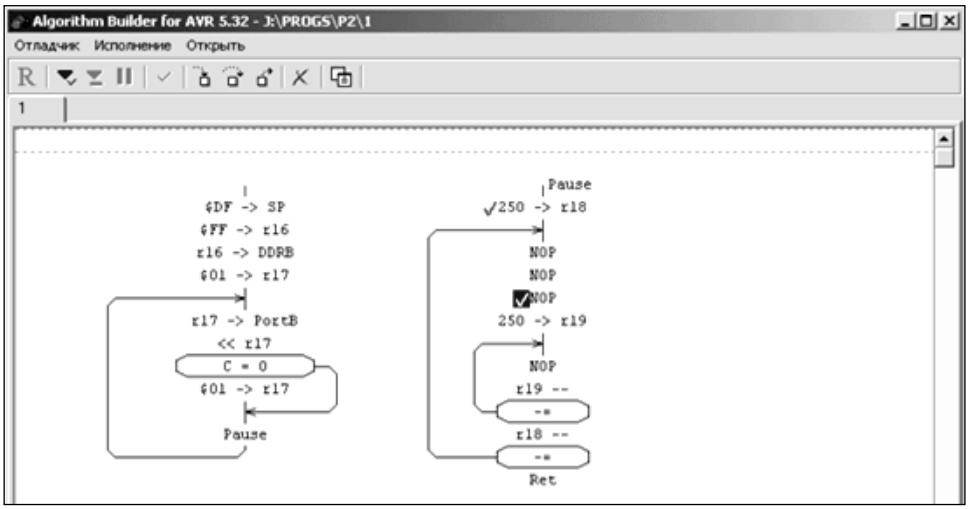


Рисунок 1.28 – Вікно симулятора AVR Builder

Для покрокового виконання програми використовується клавіша «F7». Покрокове виконання без входу в підпрограму виконується за допомогою клавіші «F8». Виконання алгоритму до виходу з підпрограми виконується клавішею «F6». Запуск на необмежене виконання до точки зупину виконується клавішею «F9», а виконання до виділеного елемента клавішею «F4». При цьому виконання алгоритму може бути призупинено натисканням клавіші «F2». Для більш зручного використання програми передбачені спеціальні кнопки на панелі інструментів симулятора, які відповідають вищезазначеним функціям. Під час виконання алгоритму у вікні моніторингу порту B спостерігається почергова поява логічних одиниць на виходах.

1.7 Робота в інтегрованому середовищі розробки FLOWCODE

У сучасному світі швидкість розробки програмного забезпечення є одним з головних чинників успішності продукту на ринку. З появою і широким упровадженням пристроїв на базі мікроконтролерів (MCU) з'явилася проблема прискорення процесу написання програмного забезпечення для таких систем. Одним із способів вирішення цієї задачі є застосування середовищ візуального програмування. Яскравим представником подібних засобів розробки є програмний комплекс FlowCode від компанії Matrix Multimedia [7, 25]. FlowCode – це середовище розробки із зрозумілим графічним інтерфейсом, що використовує мову програмування на основі об'єктів і блок-схем. Реалізація технології Drag and Drop дозволяє з легкістю створювати програми простим перетягуванням необхідних іконок, а блок-схемний підхід до написання програми підвищує її наочність та структурованість. Таке середовище програмування дозволяє створювати код для мікроконтролерів AVR, ARM і PIC, які є на сьогодні найпоширенішими. У ньому є готові бібліотеки програмного коду для різних периферійних модулів таких, як USART, SPI, ADC, а також різних компонентів, що, як правило, входять до складу пристроїв на основі мікропроцесорів (світлодіодні індикатори, LCD, крокові двигуни) (рис. 1.29).

Така можливість дозволяє зобразити ці блоки як окремі елементи (рис. 1.30), які мають відповідні входи і виходи, що істотно скорочує час на реалізацію коду. Досить просто додати звернення до потрібного модуля з програми. Ще однією корисною функцією такої програми є відкрита архітектура, яка дає можливість отримати лістинг мовою Асемблер і C. За необхідності
його можна відредагувати, що особливо актуально в додатках, де потрібна максимальна продуктивність. Також можна перевірити код на наявність логічних помилок за допомогою вбудованого відлагоджувача, який дозволяє візуалізувати такі процеси, як: вивід інформаціїї на LCD дисплей, обертання крокового двигуна і т. д.

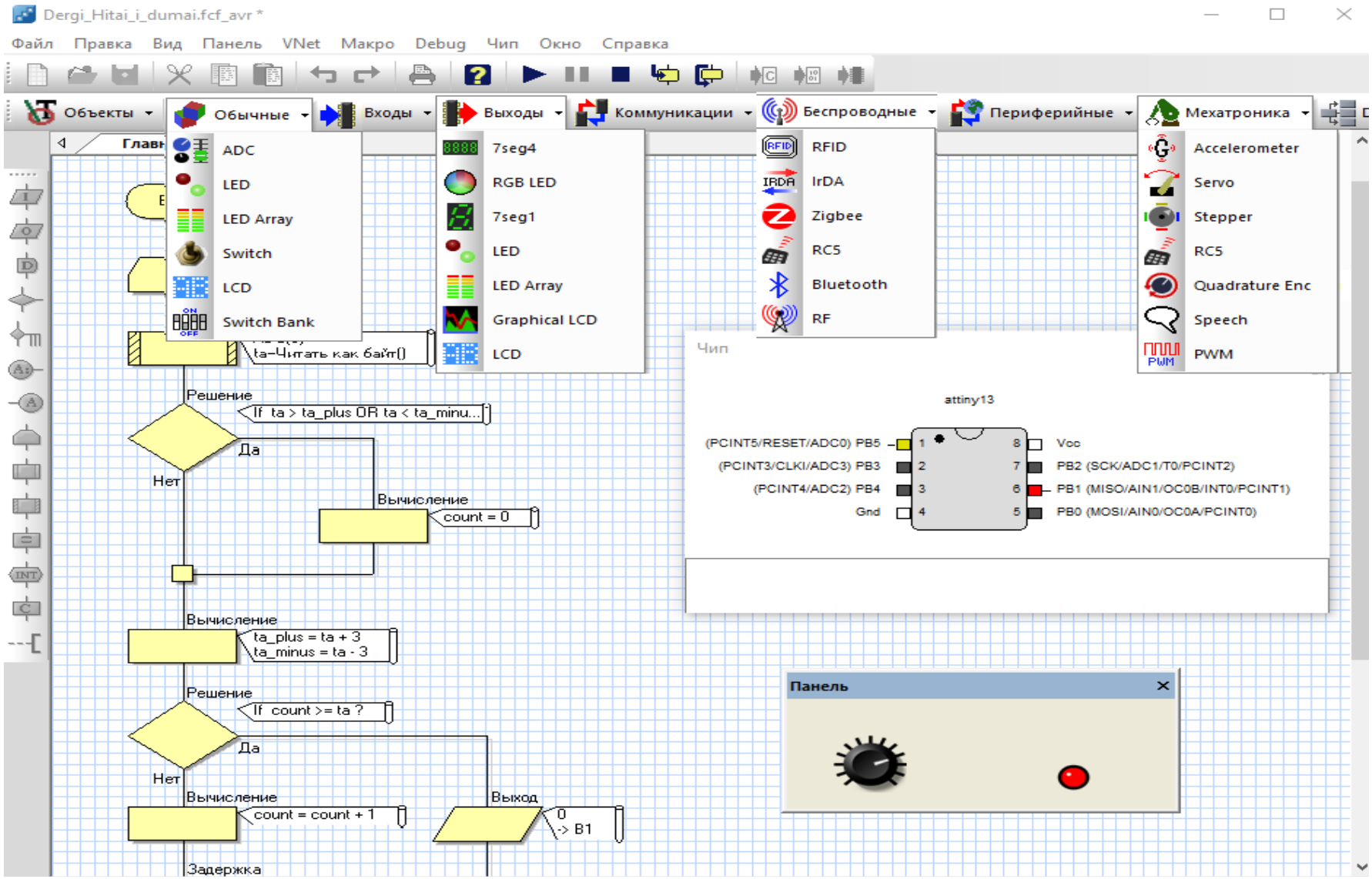


Рисунок 1.29 – Зовнішній вигляд програми FlowCode

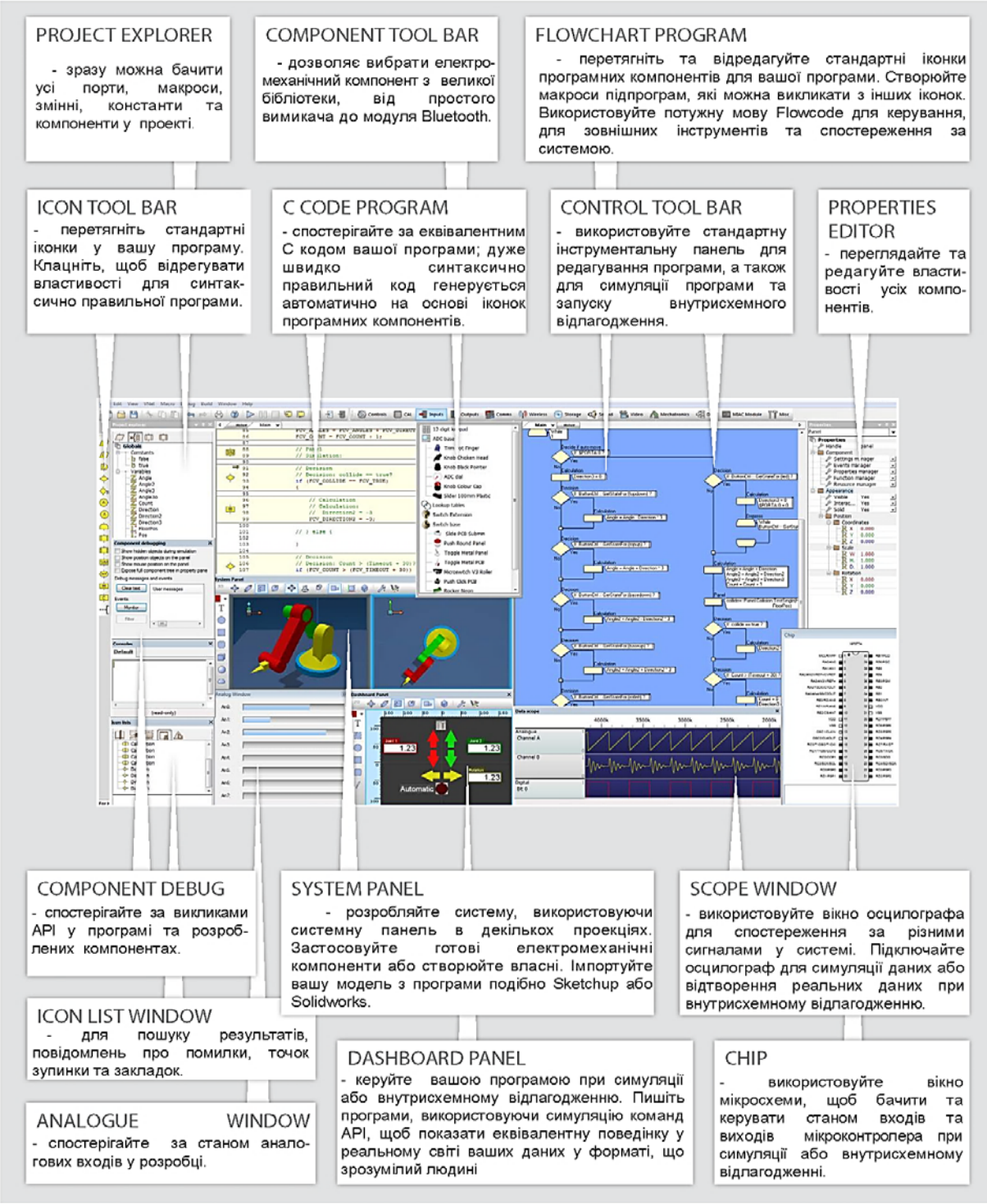


Рисунок 1.30 – Призначення елементів інтерфейсу програми FlowCode

1.8 Робота у програмному симуляторі Proteus VSM

Для розробки і налагодження МПС потрібні прилади, які здатні виконувати функції вимірювання напругу і струму, можуть відтворювати форму сигналу, подавати імпульси певної форми і т. д.; подавати послідовність сигналів одночасно на декілька входів відповідно до заданої часової діаграми або заданого алгоритму функціонування; збирати значення сигналів багатьох ліній протягом одного і того ж проміжку часу, який визначається

подіями, що задаються комбінацією або послідовністю сигналів на лініях; обробляти і представляти зібрану інформацію або у вигляді часової діаграми, або у вигляді таблиці логічних станів, або мовою високого рівня. Для автономного налагодження широко використовуються осцилографи, вольтметри, амперметри, частотоміри, генератори імпульсів і кодів, що дозволяють виконувати це для апаратури на схемному рівні. Для здійснення комплексного налагодження МПС використовують логічні аналізатори, налагоджувальні й діагностичні комплекси.

Процес моделювання має бути максимально наближений до реальності. У такому разі користувач здійснює природну послідовність таких дій, як складання схеми, підключення вимірювальних приладів, встановлення режимів роботи вимірювальних приладів, отримання режимів роботи в звичній для нього формі. Таку можливість надає програма Proteus VSM [9, 11, 19, 26, 27].

На відміну від багатьох інших, ця програма здатна моделювати пристрої не тільки на дискретних компонентах, звичайних аналогових і цифрових мікросхемах, а й на МК серії AVR, 8051, PIC12, PIC16, PIC18, Z80, 68000. Програма містить велику кількість бібліотек напівпровідникових пристроїв, пасивних компонентів, ламп, індикаторів (світлодіоди, семисегментні, LCD), кнопок, клавіатур, динаміків, мікрофонів, джерел струму, напруги, генератори спеціальних сигналів тощо.

Пакет Proteus складається з двох програм: ISIS – моделювання електронних схем та ARES – програма створення друкованих плат. У ISIS передбачена покрокове налагодження МПС, можливість анімації елементів схем. Розглянемо налагодження пристрою на МК AVR AT90S8515 і інтерфейсом 1-Wire, по якому підключаються DS1990, DS18S20 та інші 1-Wire прилади [9, 11, 19, 27]. Запускаємо PROTEUS і в меню File→Load Design перейдемо на один рівень вверх – у папку, де встановлений PROTEUS і відкриваємо папку SAMPLES. У ній знаходимо і відкриваємо файл проекту One-Wire→NETWORK→1WIRE_NET.DSN. Розгорніть вікно PROTEUS на весь екран (рис. 1.31).

У лівій верхній області екрана відображається міні-макет сторінки й трохи нижче панель DEVICES – компоненти проекту (рис. 1.32). У цьому вікні відображаються всі елементи, що використовуються в схемі. Кнопка з буквою «P» відкриває форму пошуку компонента в бібліотеках PROTEUS для додавання в схему. Кнопка з буквою «L» відкриває менеджер бібліотек – з його допомогою можна підключати нові бібліотеки компонентів. На рис. 1.32 бачимо МК AVR AT90S8515, кілька компонентів компанії DALLAS-MAXIM (вони підключені за схемою інтерфейсу 1-Wire). Внизу списку є компонент PULLUP – це резистор, що підтягує напругу до + живлення МК і приладів DSXXXX (звичайно це +5 В).

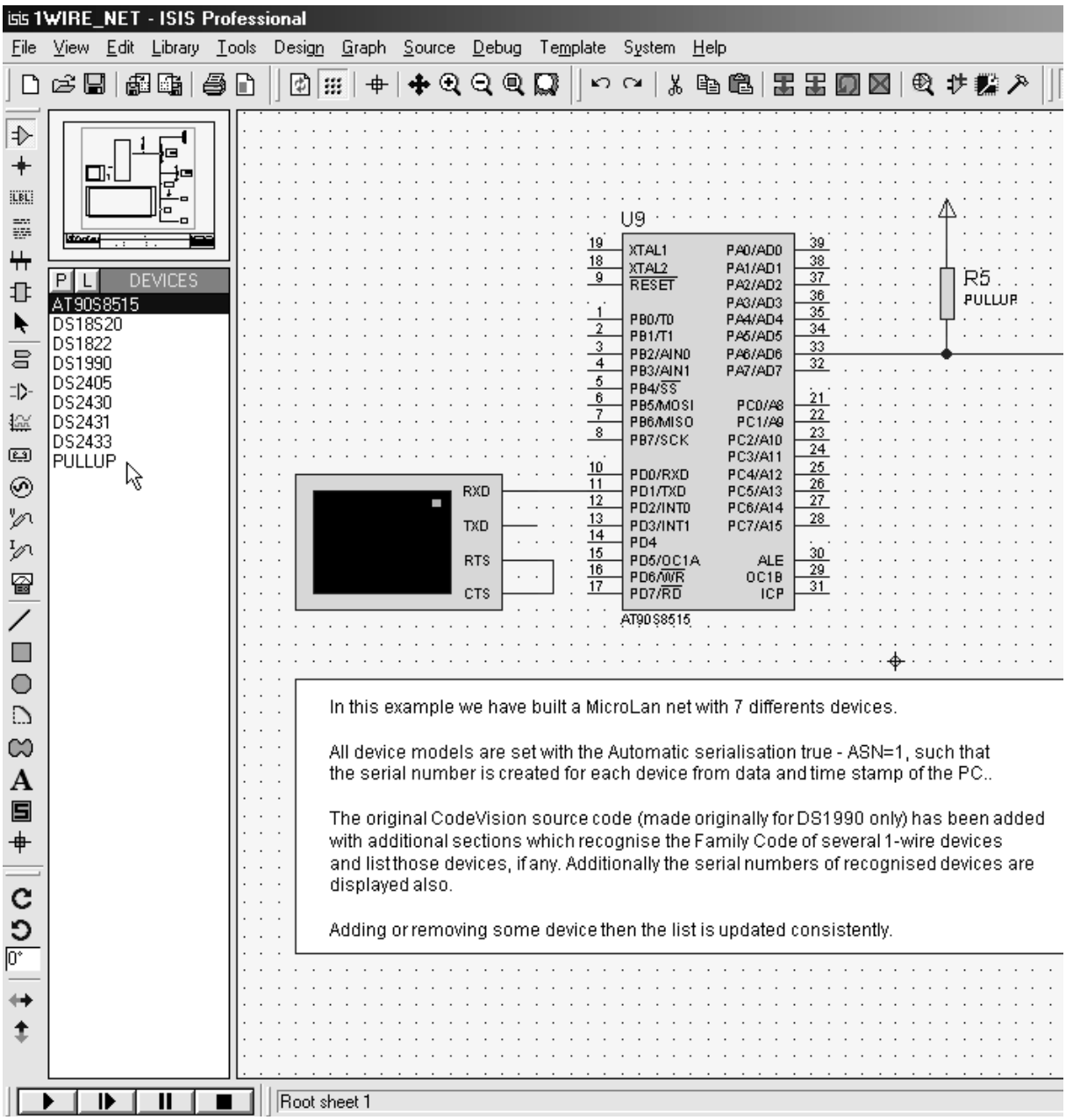


Рисунок 1.31 – Вікно проекту 1WIRE_NET у середовищі Proteus

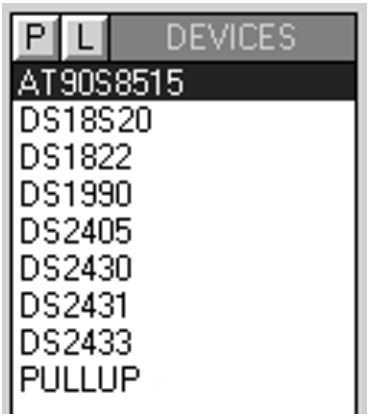


Рисунок 1.32 – Панель DEVICES

Натисніть кнопку з буквою «P». Відкриється форма, що містить меню пошуку і вибору компонентів «Pick Devices» (рис. 1.33).

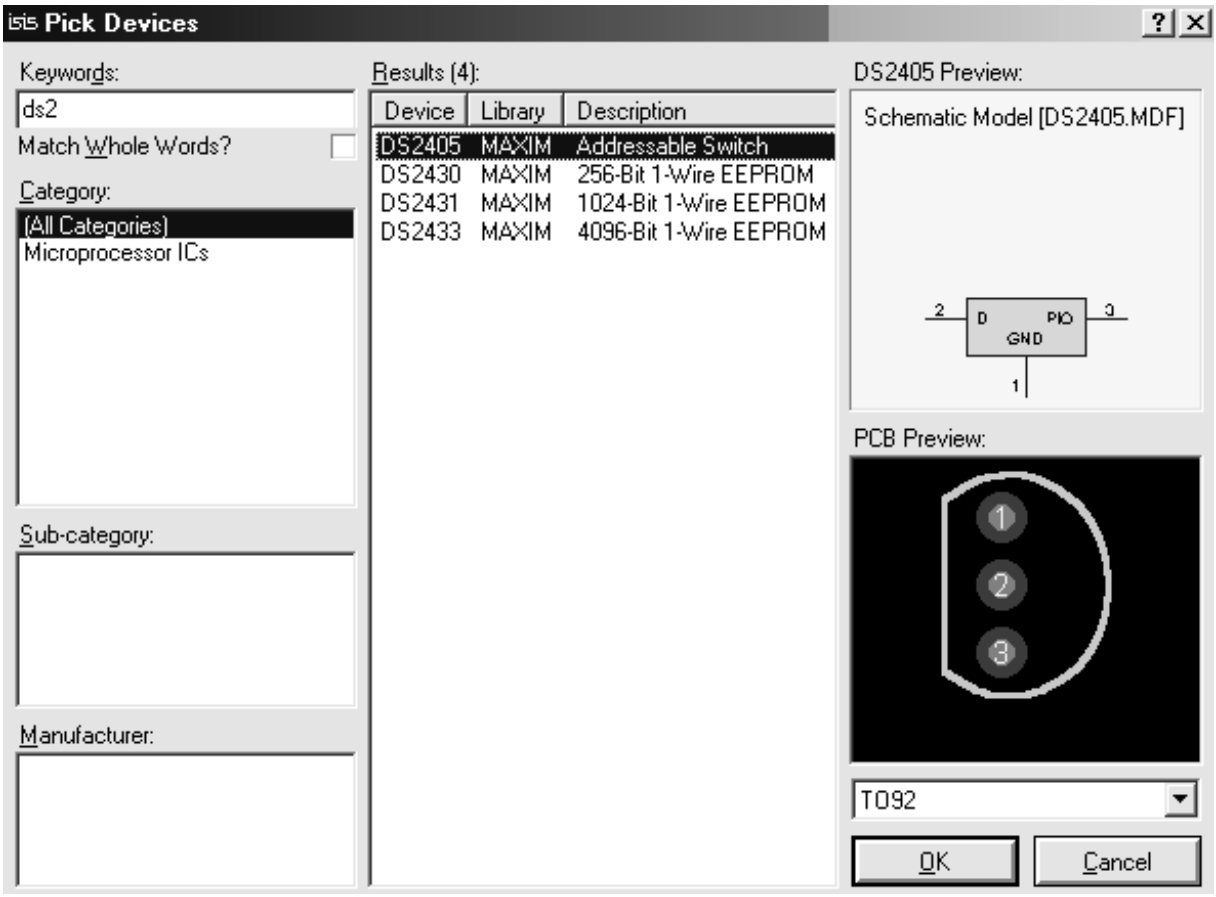


Рисунок 1.33 – Форма-меню пошуку і вибору компонентів «Pick Devices»; введений ключ пошуку DS2

У полі «Keywords» (ключові слова) введіть DS2 і виберіть верхній з 4-х знайдених компонентів – DS2405. У полі «Description» (опис) спостерігаємо «адресований перемикач», тобто до цього приладу можна звернутися за його адресою і «наказати» йому зробити на виході PIO логічний «1» або «0» і можливо перевести вихід у високо імпедансний Z-стан – вивід з дуже великим опором, який не впливає на те, що до нього підключено. У правій частині форми можна побачити назву моделі компоненту, його зображення на схемі, а нижче за нього «FootPrint» – його корпус. Ще нижче назва корпусу компонента – TO92.

Інша частина 1-Wire приладів на схемі починається на DS1 (рис. 1.34) – введіть ці символи у поле ключових слів. Тепер знайдено 8 приладів. Причому вони розташовані в 2-х категоріях. Обираємо мишкою DS18S20 – в описі написано: «Високоточний 1-Wire цифровий термометр». На схемному зображенні (рис. 1.35) цього компоненту видно деяке поле, що нагадує дисплей, у якому будуть виводиться дані в процесі симуляції.

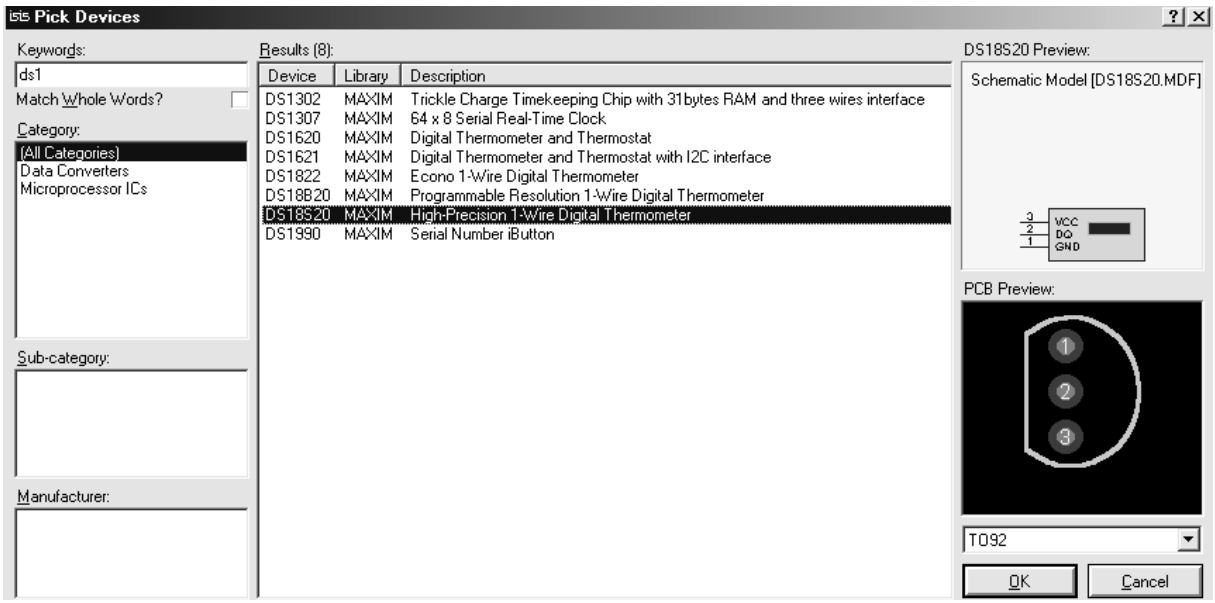


Рисунок 1.34 – Форма-меню пошуку та вибору компонентів «Pick Devices»; введений ключ пошуку DS1

Помістити компонент на схему можна натиснувши «ОК», потім помістите покажчик миші у потрібне місце на листі схеми і натиснути мишею. Компонент опиниться на схемі. Червоними стрілками під час симуляції можна змінювати температуру корпусу датчика – натиснувши по них мишею. Тут VCC – + живлення датчика, GND – «земля», DQ – лінія даних.

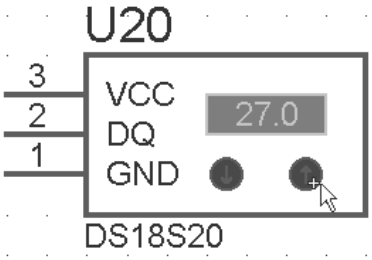


Рисунок 1.35 – Умовне позначення датчика DS18S20 у вікні проекту 1-WIRE_NET у середовищі Proteus

Рисунок 1.36 – Панель керування симуляцією

Панель керування симуляцією (рис. 1.36) складається з 4 кнопок: «Пуск» – запуск симуляції або продовження припиненої симуляції; «Крок» – виконати мінімальний крок за програмою МК, звичайно це одна інструкція на асемблері (цією кнопкою теж можна почати симуляцію); «Пауза» – пауза симуляції (можна продовжити кнопками «Пуск» або «Крок»); «Стоп» – зупинка симуляції.

Обираємо МК (рис. 1.31) натиснувши на ньому правою кнопкою миші. МК і підключені до нього провідники стануть червоними. Відкриваємо панель редагування властивостей компонента (Edit Component) натиснувши на МК лівою кнопкою миші. Натисніть кнопку "Hidden Pins" (приховані виводи) – відкриється додаткове меню (рис. 1.37), де за замовчуванням є назви вузлів схеми, до яких підключені живлення МК: VCC і GND. Зміна цих назв інколи необхідна при живленні МК або інших компонентів різною напругою або від різних джерел.

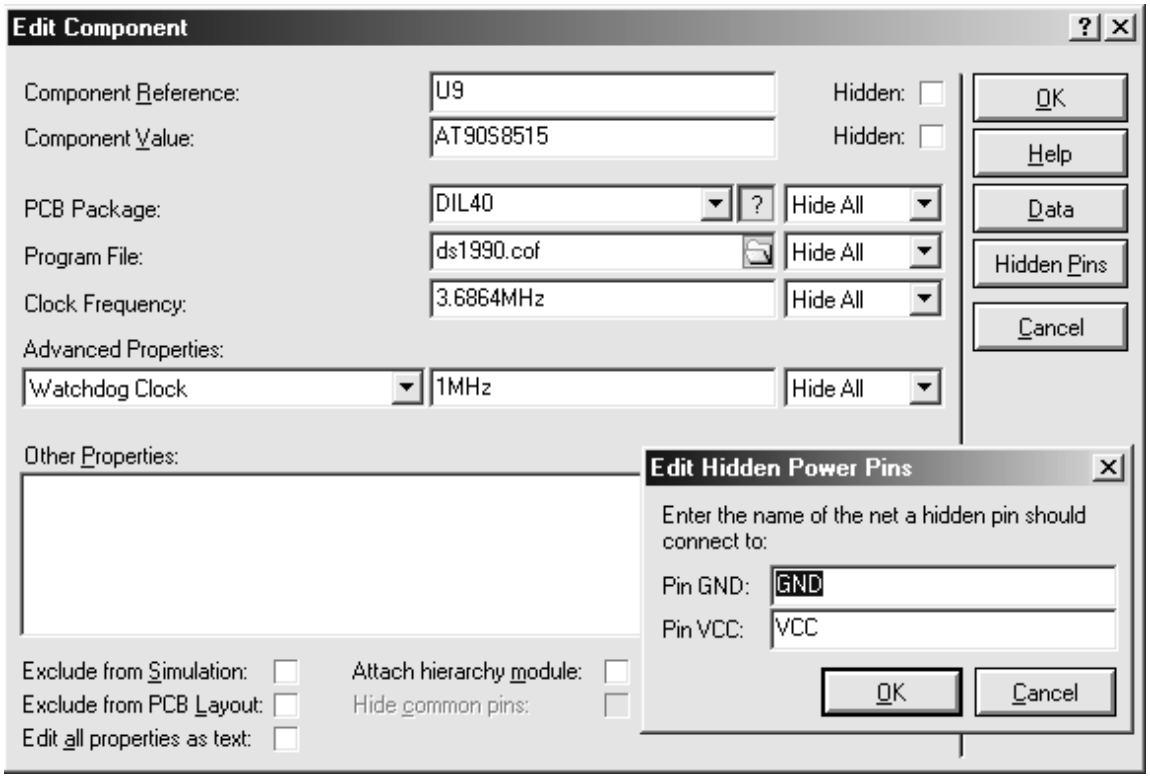


Рисунок 1.37 – Панель редагування властивостей компонента

Головне для МК це програма за якою він працює, тому у полі «Program File» потрібно вказати: .cof – файл, якщо ведеться налагодження програми (початкова програма) мовою С; .hex – файл прошивки, якщо немає ніякої початкової програми на С, ні на Асемблері. Якщо є початкова програма на Асемблері .asm, то потрібно вказати назву .hex файлу і через меню: Source →Add/remove Source File... додати назву файлу з текстом програми на асемблері і вибрати потрібний асемблер (рис. 1.38).

Щоб симулювати у Proteus роботу МК необхідно знайти його в бібліотеках і помістити на схему, указати яку програму він повинен виконувати, вказати тактову частоту МК. Що б побачити всі встановлення компоненту в панелі редагування (рис. 1.37) натисніть Edit All Prop. As Text (рис. 1.39).

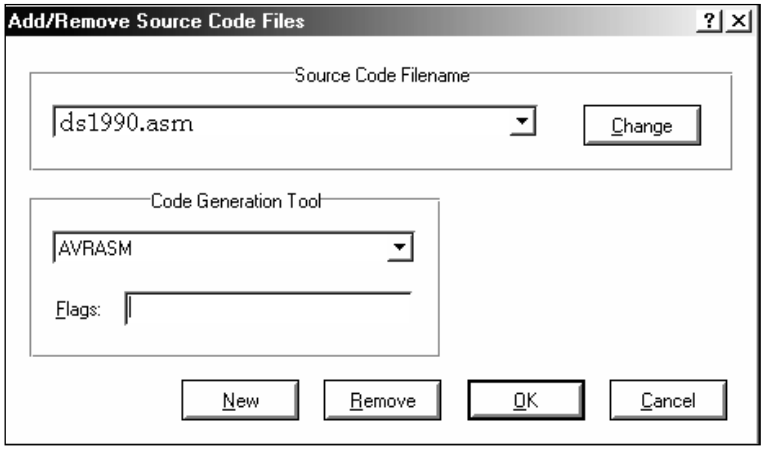


Рисунок 1.38 – Вікно Add/remove Source File

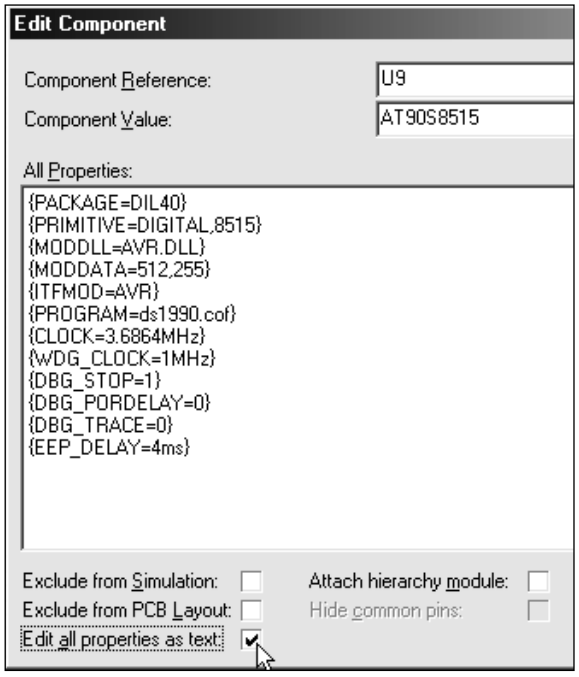


Рисунок 1.39 – Редагування властивостей МК у текстовому режимі

{CLOCK} – однозначно визначає тактову частоту МК при симуляції. Кварц і конденсатори не потрібні для симуляції, їх встановлюють на схему тільки для того, щоб врахувати при розводці друкованої плати пристрою.

{Wdg_clock}– показує частоту роботи генератора сторожового таймера. Хоча вона і позначена як 1MHz – сторожовий таймер не включений, поки не додамо у властивості МК стрічку: {Wdgon=1} (для даного проекту не потрібно її включати).

{Program} – показує, що МК працюватиме за програмою ds1990.cof (створив компілятор CVAVR).

Значення інших параметрів можна дізнатися в довідці, яка відкриється після натискання кнопки Help. Керувати процесом симуляції програми на асемблері можна за допомогою клавіш F10, F11, F12 і їх комбінацією з клавішами Alt і Ctrl і через меню DEBUG (рис. 1.40).

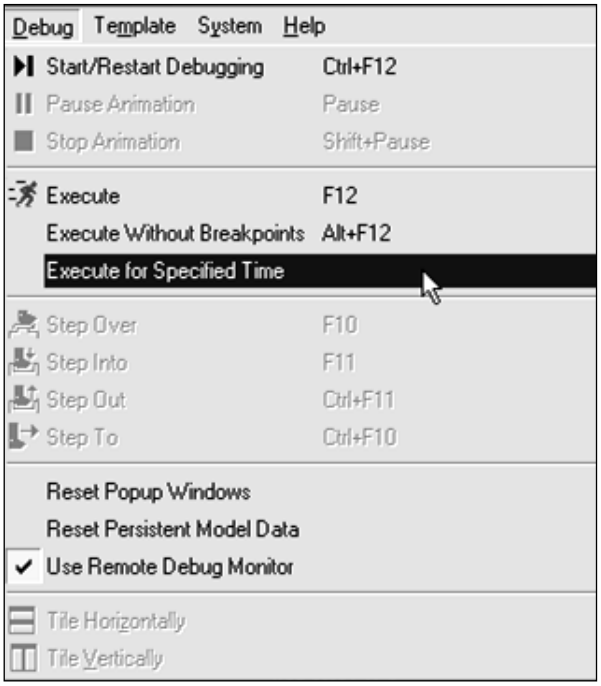


Рисунок 1.40 – Меню DEBUG

При виборі команди Execute for Specified Time виводиться поле введення для вказівки скільки часу має відпрацювати схема (n – наносекунди, u – мікросекунди, m – мілісекунди, секунди можна не позначати). Вказаний час буде використовуватись, якщо до його закінчення в програмі МК не зустрінеться точка зупинки (BP – Breakpoint) – симуляція зупиниться на ній.

Для меню DEBUG корисні такі команди:

- F12 еквівалентна кнопці «Пуск» – запускає або продовжує симуляцію до BP;
- Alt+F12 запускає або продовжує симуляцію, що не зупиняється на BP;
- F10 – симулювати не використовуючи процедури на асемблері або функцій на C (вони будуть виконані як 1 крок);
- F11 – симулювати крок за програмою із входом у процедуру або функцію;
- Ctrl+F11 – закінчити процедуру або функцію, у якій знаходимося;
- Ctrl+F10 – виконувати сумуляцію до місця в програмі, яке означене виділеним рядком в програмі на асемблері.

При симуляції пристрою на екрані спостерігаємо анімаційну картинку. Режим анімації можна встановити через меню System→Set Annimation Options (рис. 1.41).

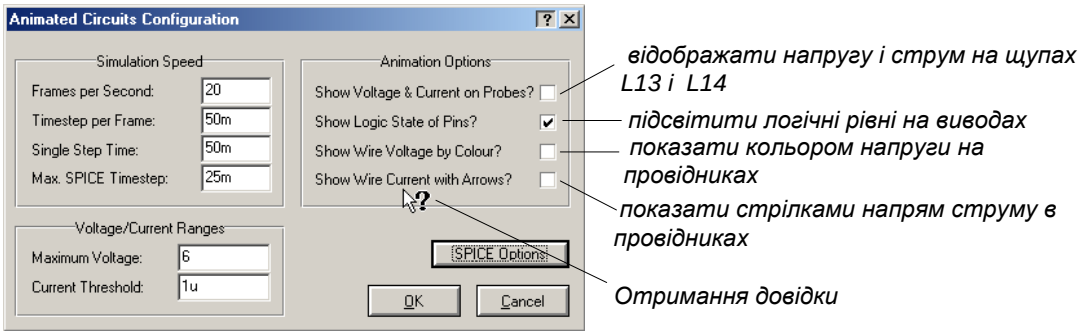


Рисунок 1.41 – Встановлення опцій анімації

Запускаємо симуляцію проекту 1WIRE_NET.DSN кнопкою «Пуск» (рис. 1.40). З'явиться вікно віртуального терміналу ПК (рис. 1.42) і приблизно за 1 секунду програма МК зробить все, що потрібно. При анімації логічні рівні на виводах вказуються кольоровими квадратиками: червоний – логічний рівень «1», синій – логічний рівень «0», сірий – Z стан.

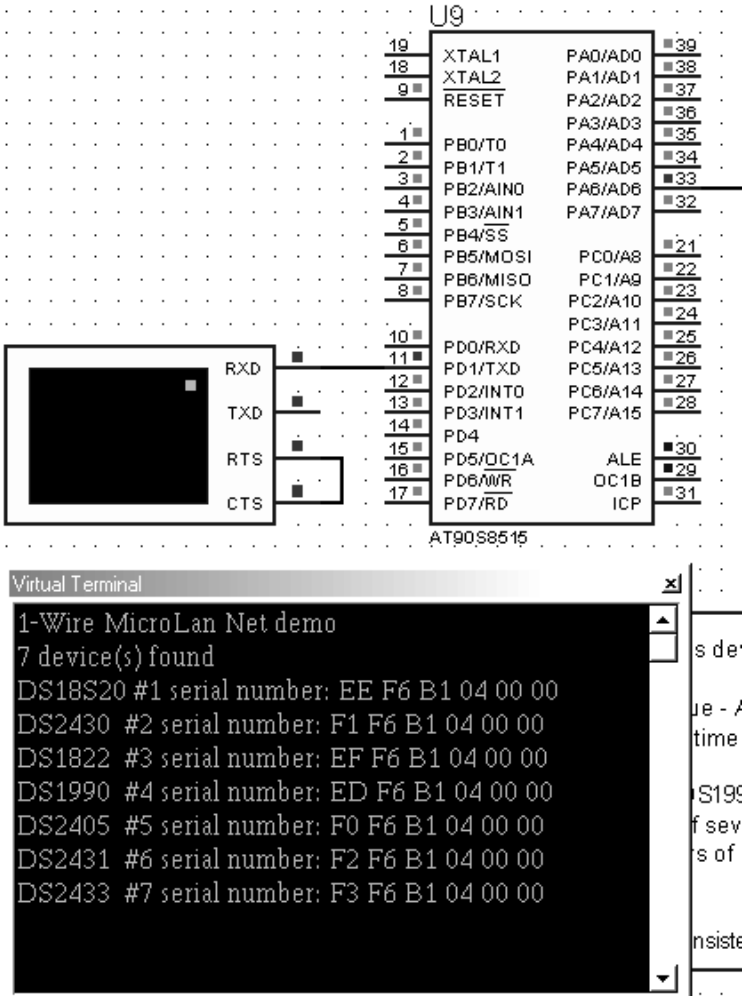


Рисунок 1.42 – Вікно симуляції проекту 1WIRE_NET.DSN; відкрито вікно Virtual Terminal

У вікні віртуального термінала (див. рис. 1.42) виводиться інформація з UART МК на віртуальний COM-порт ПК списку розпізнаних пристроїв на шині 1-Wire і їх номери: серійний номер DS1990, серійні номери приладів DSXXXX, що задаються на заводі й незмінні (у Proteus ці номери задаються у вікні редагування властивостей компонента).

У Proteus можна не використовувати узгоджувач рівнів RS232 з UART типу MAX232, а підключати термінал безпосередньо до МК. Термінал підтримує службові символи ASCII – управління виведенням тексту: CR (0DH), BS (0X08H) і BEL (0X07H). LF (0X0A) і решта службових кодів не підтримуються. У режимі паузи симуляції можна за натисненням правої кнопки миші на екрані терміналу відкрити меню, де можна скопіювати, вставити інформацію, можна змінити формат виводу числа (символи або 16 річний код), можна очистити екран даних, можна встановити режим «ехо» (прийняті на RXD терміналу символи будуть виводиться на його ніжку TXD).

Виводи RTS і CTS можна не з'єднувати і не підключати взагалі – хоча вони працюють як у справжньому COM-порту ПК. Термінал може передавати щось тільки при рівні «0» на виводі CTS. Можна використовувати декілька незалежних терміналів.

Якщо натиснути мишею у вікні терміналу, то він почне передавати символи, що набирають на клавіатурі ПК, на ніжку TXD. Можна вставити з буфера обміну інформацію, яку термінал так само виводитиме на ніжку TXD. Інформація, що передається не відображається у вікні терміналу. Курсор залишається на місці. У властивостях терміналу в полі додаткові властивості (Other Properties) можна ввести текст, який посилатиметься з терміналу при натисненні кнопки «Пуск» із затримкою на час передачі одного символу.

Компонент COMPIM дозволяє віртуальному пристрою підключитися до реального COM-порту ПК.

Натисніть «Стоп» потім «Старт» і «Пауза». На екрані монітора будуть відображатись вікна з налагоджувальною інформацією, перелік цих вікон задається у меню DEBUG (рис. 1.43).

Інші елементи схеми, у яких є цифрові дані або пам'ять, мають свої підменю з можливістю виведення інформаційних вікон. Наприклад, цифровий термометр DS18S20 має Scratch RAM і EEPROM. Якщо в схемі декілька МК, то для кожного буде своє підменю.

Watch Window – вікно стеження (рис. 1.44). Після налаштування воно не зникає з екрана при симуляції і знаходженні в паузі. У цьому вікні можна розмістити регістри МК і не тільки відстежувати їх вміст за ходом програми, а й задавати деякі умови і дії, при досягненні яких, наприклад, зупинити симуляцію. При виборі Add Items (By Address) з'являється панель спостереження. Натиснувши на ній правою кнопкою миші, з'являється меню для додавання регістрів, що будуть спостерігатись (за ім'ям або за їхньою адресою). Подвійний натиск мишею на назві регістра додає його у вікно спостереження.

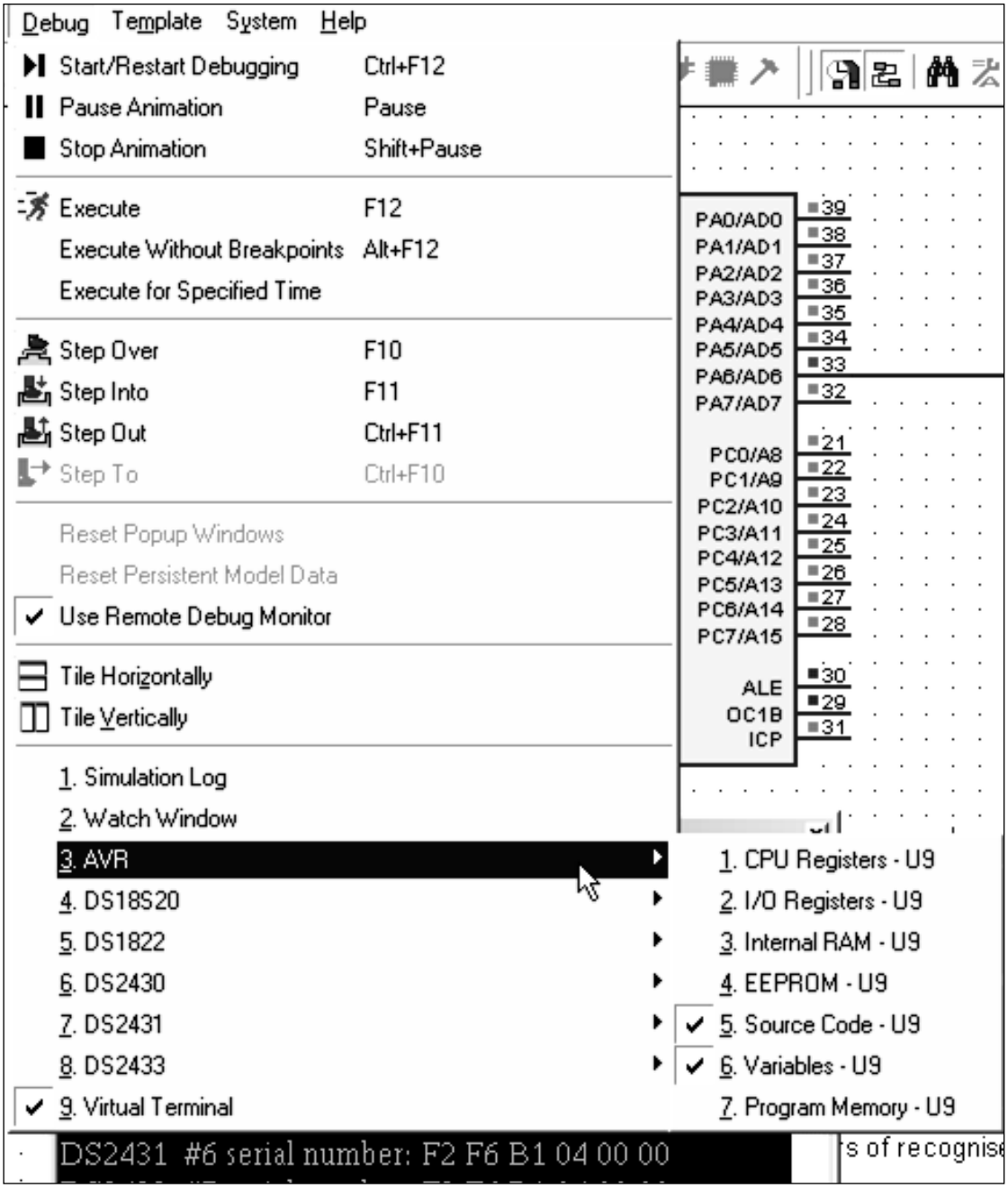


Рисунок 1.43 – Вибір коду програми та змінних у меню DEBUG для відображення їх під час налагодження роботи програми

Якщо задати умову у Watch Expression, то етапи процесу симулювання визначаються у вікні (рис. 1.45). Розглянемо уважно й інші пункти меню DEBUG (рис. 1.43). Вікно з текстом програми – AVR Source Code – U9. На жаль, при симуляції МК AVR в Proteus – це вікно не пам'ятає, яку програму показувати, якщо в ній немає активних ВР. Тому потрібно вказати, яку програму використовувати (рис. 1.46).

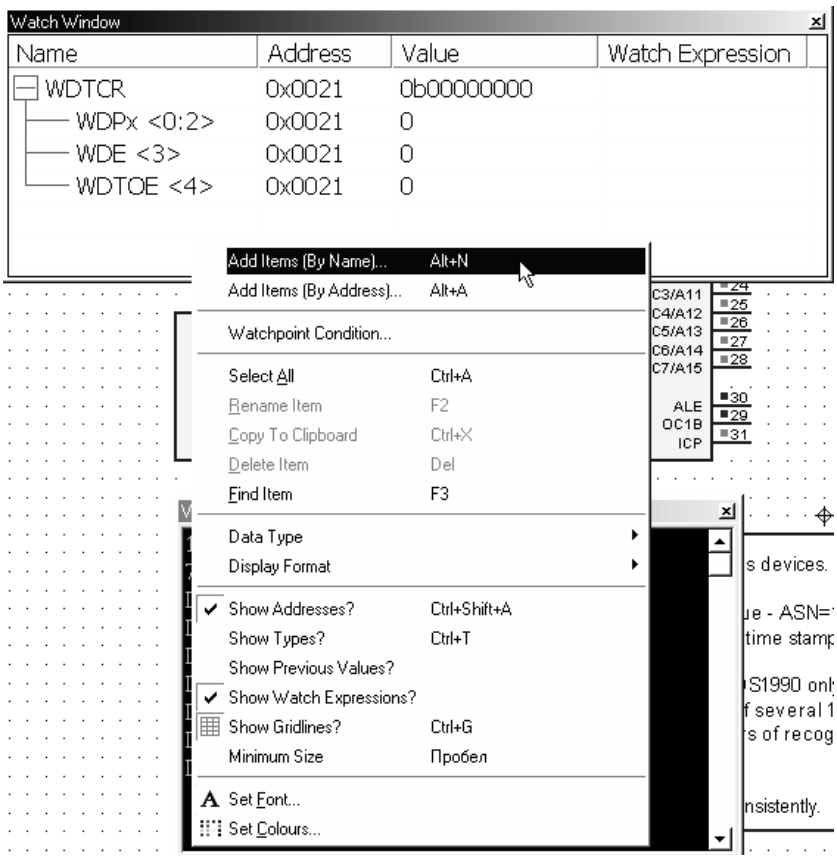


Рисунок 1.44 – Вікно Watch Window та його властивості

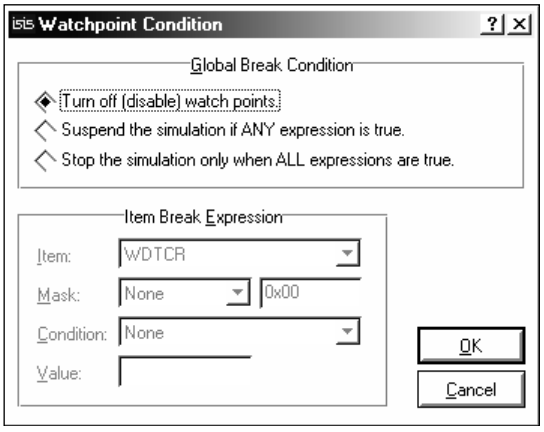


Рисунок 1.45 – Вікно, що задає, як працює симулятор при виникненні умов у Watch Expression

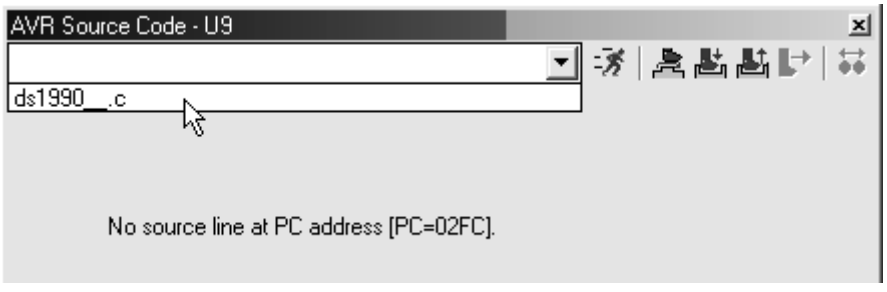


Рисунок 1.46 – Вибір тесту програми для режиму налагодження

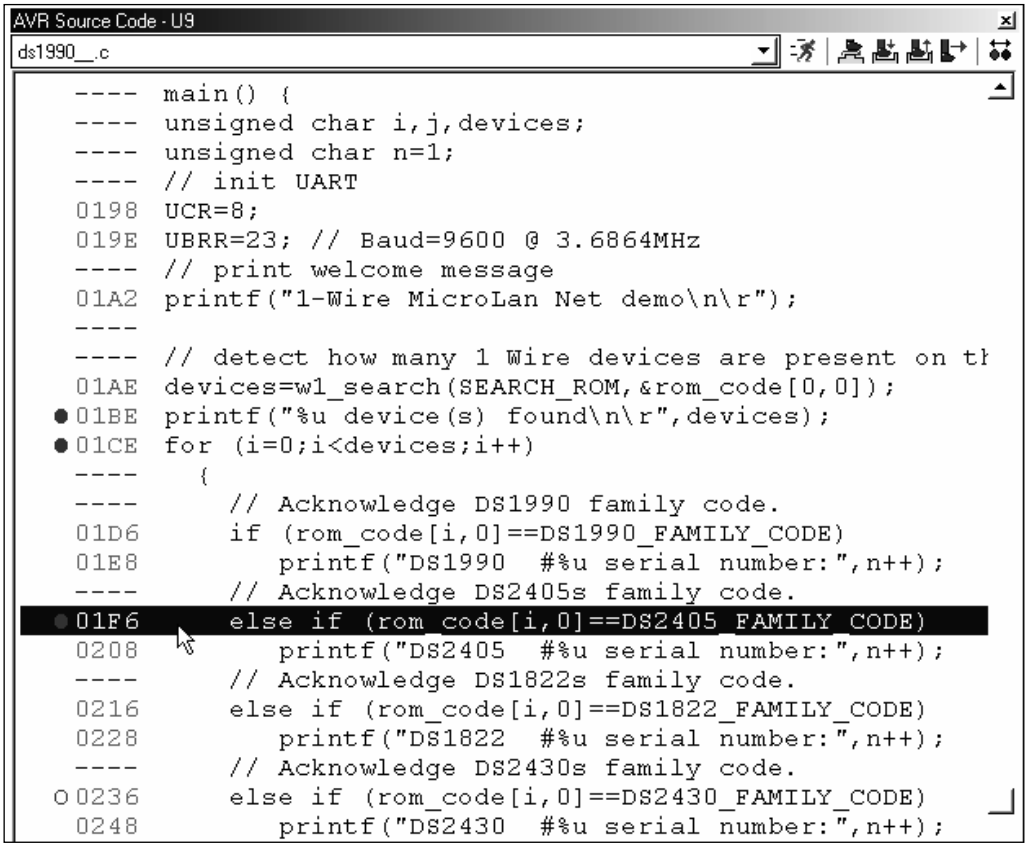


Рисунок 1.47 – Вікно з текстом програми

При виборі файлу програми відкривається вікно з текстом програми (рис. 1.47). Зелені цифри показують рядки програми, які можна виділити мишею і потім кнопкою F9 поставити ВР (точку зупинки) – червоний кружечок. Програма, дійшовши до цієї точки зупиниться, у режимі паузи – на екрані монітора з'являться вікна з налагоджувальною інформацією. Якщо натиснути F9 ще раз, то ВР дезактивувався – стане червоним колом і програма не зупиниться на ній. Наступне натиснення кнопки F9 приховає ВР з виділеного рядка.

Розставимо ВР, як на рис. 1.47. Натискаємо кнопку «Стоп», а потім «Пуск». Програма зупиниться на 1-ій точці ВР і з'явиться вікно з текстом програми мовою С (рис. 1.48).

Рядок, на якому зупинилася програма, підсвічується сірим кольором і на неї вказує червоний трикутник зліва від ВР. Натисніть «Старт», програма зупиниться на наступній точці ВР, тобто на наступному рядку програми. У вікні терміналу видно, що відбулося виконання попереднього рядка програми.

Натиснувши правою кнопкою миші у вікні програми відкривається меню, де можна побачити код мовою Assembler безпосередньо в тексті програми мовою С (рис. 1.49); включити нумерацію рядків програми; включити або виключити усі ВР; знайти щось у програмі; перейти на потрібний рядок програми; змінити шрифт у вікні.

Рисунок 1.48 – Процес налагодження програми;
зупинка на першій точці ВР

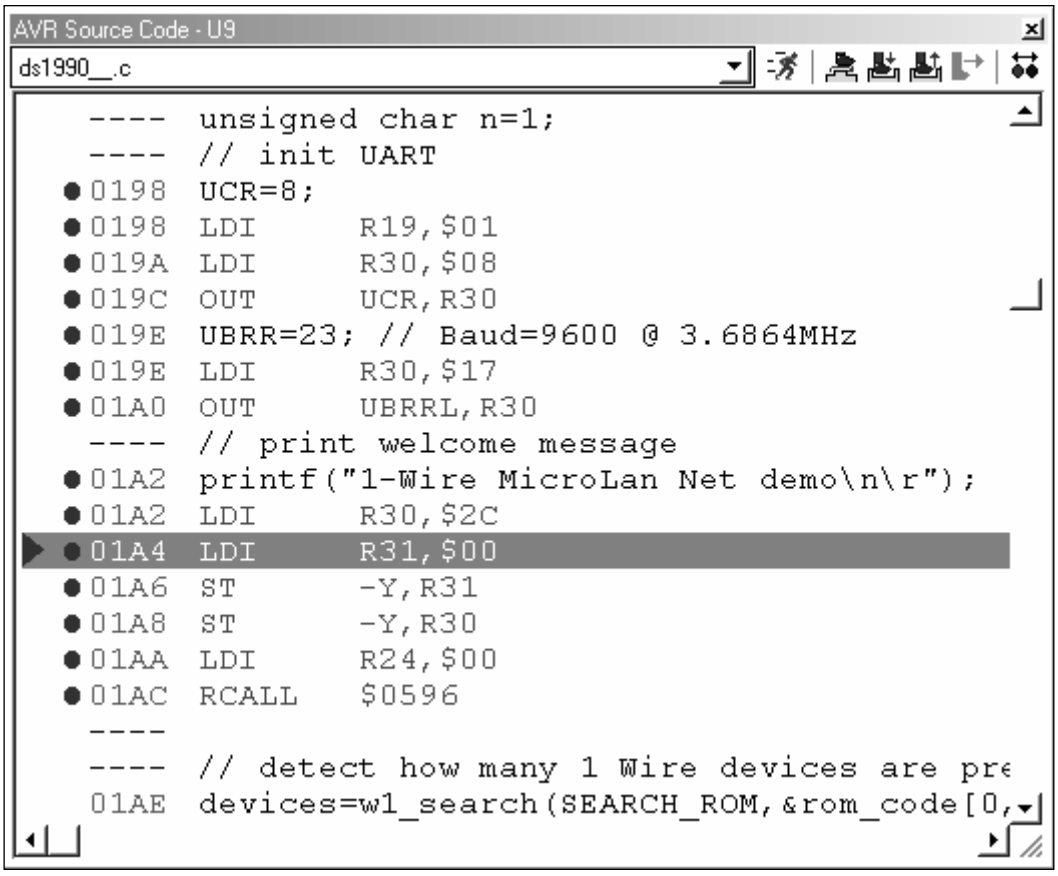


Рисунок 1.49 – Відображення у вікні програми мовою С
еквівалентного коду програми мовою Assembler

Розставивши ВР у потрібному місці програми можна покрокове налагодження за кожним рядком програми кнопкою F12 або «Пуск».

Навіть при покроковому налагодженні всі віртуальні прилади продовжують показувати вимірювані параметри і на віртуальному осцилографі або логічному аналізаторі можна побачити, після виконання якого рядка коду відбулося яка зміна на ніжках МК або в інших вузлах схеми. Детально про налагодження за початковим кодом програми написано у розділі допомоги: «SOURCE LEVEL DEBUGGING WITHIN PROTEUS VSM».

Дані симуляції можна зберегти у файл через меню Graph→Export Data. Отримаємо файл у стандартному форматі для вимірювальних приладів – CSV (дані розділені комами). Цей файл можна імпортувати до Excel або до Matlab. Графіки результатів симуляції можна експортувати до графічних та CAD форматів через меню File→Export Graphics.

1.9 Програмне забезпечення для програмування мікроконтролерів

1.9.1 Програмування у середовищі AVR Studio

Після того як мікроконтролер поміщений в гніздо плати програматора, що з'єднаний з ПК, і на нього подано живлення, у середовищі AVR Studio для налаштування з'єднання варто вибрати команду меню Tools→Program AVR→Connect або натиснути кнопку Display the Connect dialog панелі інструментів STK500. У результаті відкриється діалогове вікно Select AVR Programmer, де необхідно вибрати поточний програматор та порт, по якому він підключений до ПК (рис. 1.50).

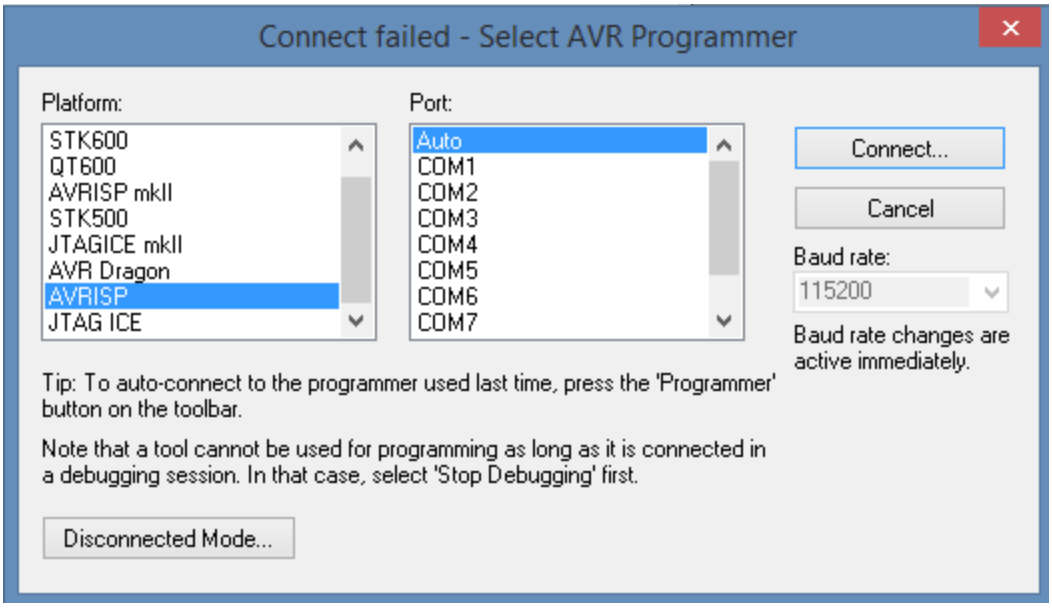


Рисунок 1.50 – Діалогове вікно Select AVR Programmer

Програматори, що використовують інтерфейси USB, тут у списку Platform за замовчуванням відповідають елементи JTAG ICE, AVR ISP та AVR Dragon. Після того як програматор і порт обрані, можна натиснути кнопку Connect. Якщо усі апаратні підключення були виконані правильно, то на екрані має з'явитися діалогове вікно, яке зображено на рис. 1.51.

На вкладці Main необхідно вибрати мікроконтролер (список Device), а також вказати виконувані HEX-файли для пам'яті программ (поле Flash – Input HEX File) та/або пам'яті EEPROM (поле EEPROM – Input HEX File). Перед програмуванням для контролю можна очистити пам'ять програм і даних мікроконтролера, натиснувши кнопку Erase Device. Власне програмування починається після натискання відповідної кнопки Program. Режим програмування (Programming mode) обирається на вкладці Main: ISP – внутрішньосхемний; PP/HVSP – високовольтний; JTAG – для програматорів AVR Dragon та JTAGICE mkll.

Установлення ознаки Erase Device Before Programming призводить до очищення пам'ять мікроконтролера перед записом.

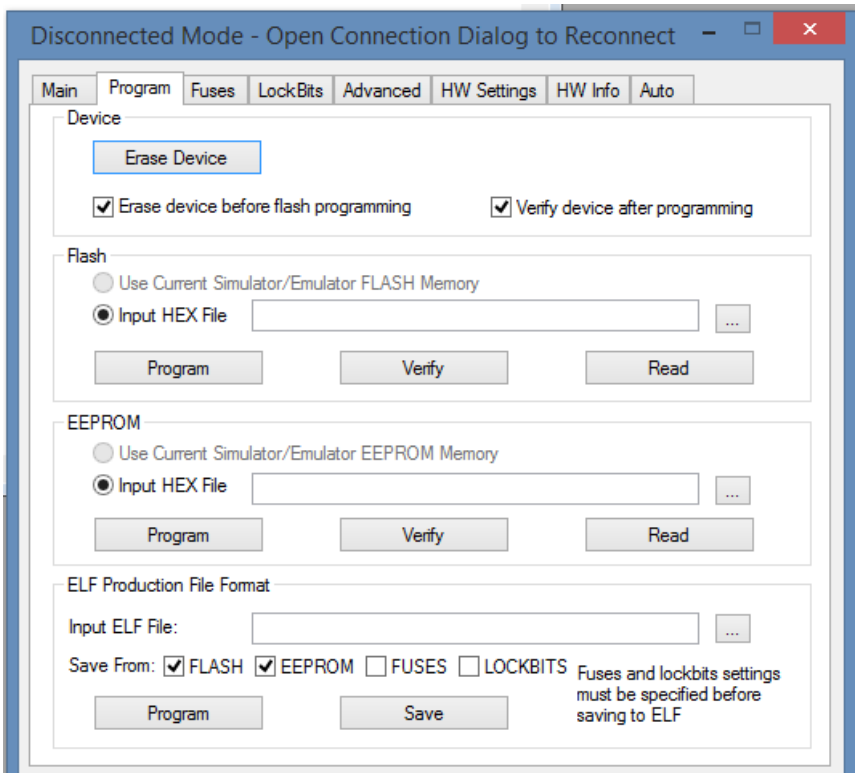


Рисунок 1.51 – Вікно керування процесом програмування

Установлення ознаки Verify Device After Programming призводить до перевірки на коректність вмісту пам'яті мікроконтролера після запису

Установлення ознаки Use Current Simulator/Emulator FLASH (EEPROM) Memory обирає для використання .hex файл з відкритого в цей момент проекту AVR Studio. Verify – кнопка перевірки записаних в пам'ять Flash або EEPROM даних. Read – кнопка читання записаних в па-

м'ять Flash або EEPROM даних. Програма AVR Programmer (рис. 1.51) дозволяє також програмувати розряди Fuses та LockBits. Для цього слугують вкладки Fuses і LockBits відповідно. Вкладка Advanced слугує для читання з мікроконтролера байтів сигнатури та калібровки осцилятора, а також для запису останнього за заданим користувачем адресою пам'яті Flash або EEPROM. Вкладка Board дозволяє змінити умови роботи плати відлагодження, а вкладка Auto – задати автоматичну виконувану послідовність операцій [2, 12, 13, 24].

1.9.2 Програмування у середовищі PonyProg2000

Завантажити і встановити програму PonyProg2000. Підключимо програматор до комп'ютера та запустимо PonyProg2000, з'явиться вікно програми (рис. 1.52) [2, 12, 13, 24, 30].

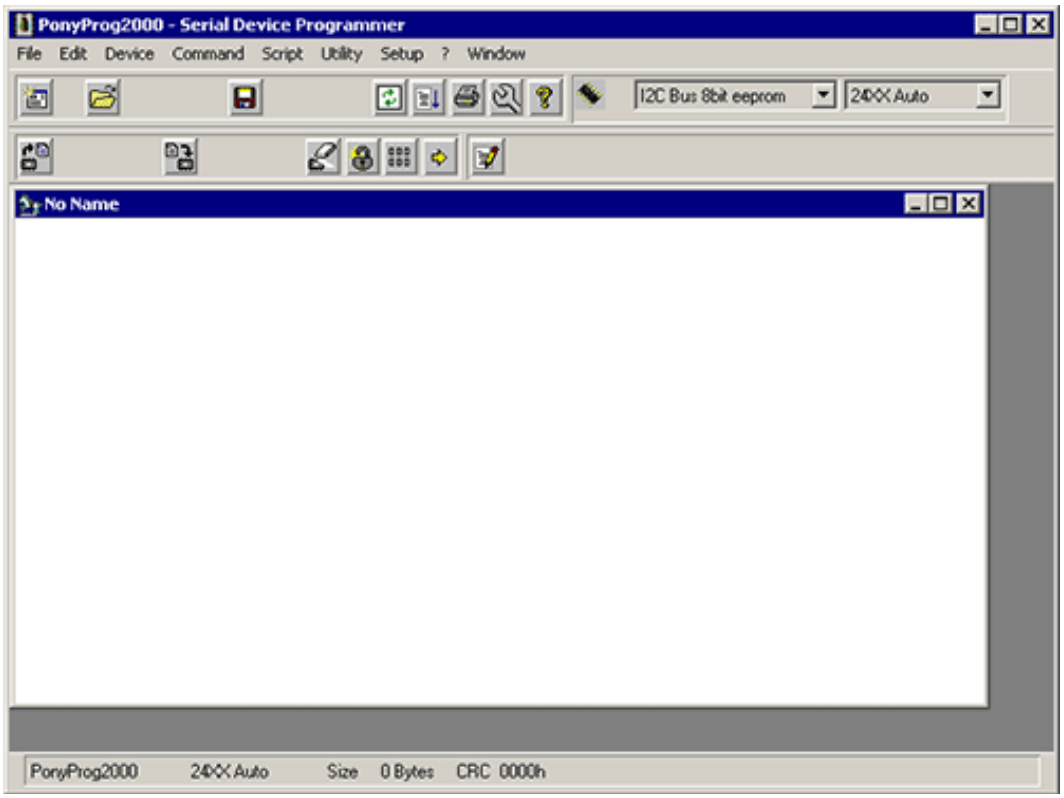


Рисунок 1.52 – Вікно програми PonyProg2000

Далі необхідно налаштувати програму для роботи з програматором. Обираємо меню Setup→Interface Setup. У вікні налаштувань потрібно установити тип та порт програматора (рис. 1.53). Далі необхідно провести калібровку програми, для цього обираємо меню Setup→Calibration. Натискаємо кнопку "Yes" (рис. 1.54) та очікуємо повідомлення (рис. 1.55) про завершення процесу калібрування.

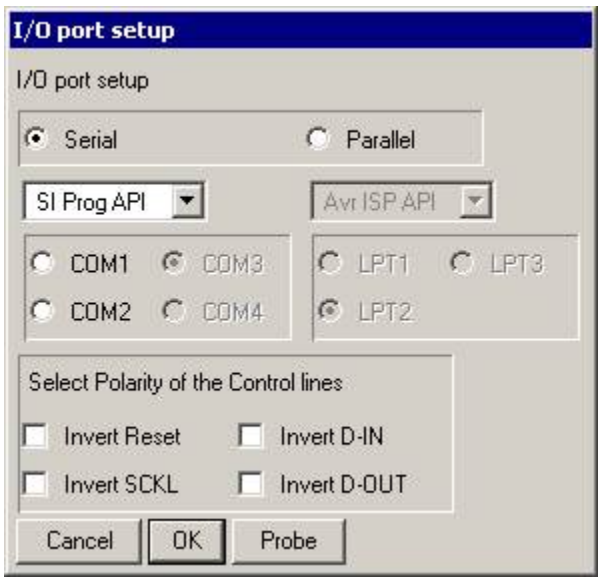


Рисунок 1.53 – Вікно налаштування типу підключення програматора у PonyProg

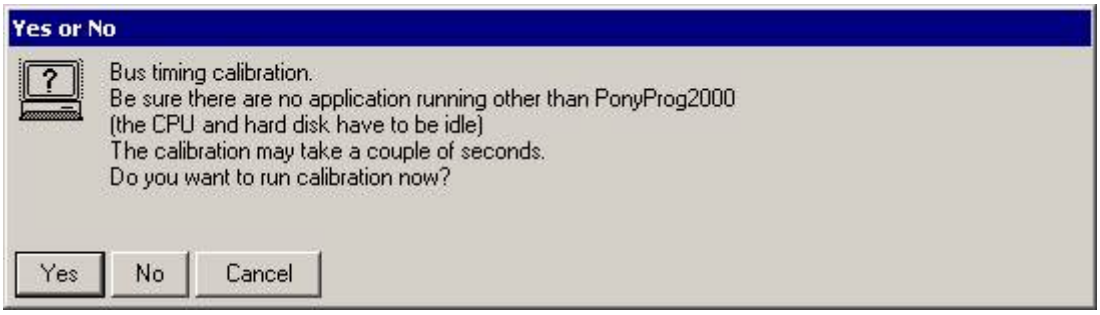


Рисунок 1.54 – Вікно початку режиму калібровки прграматора у PonyProg

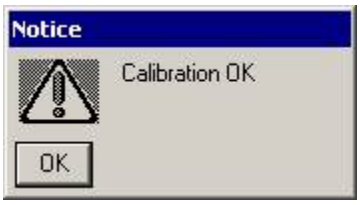
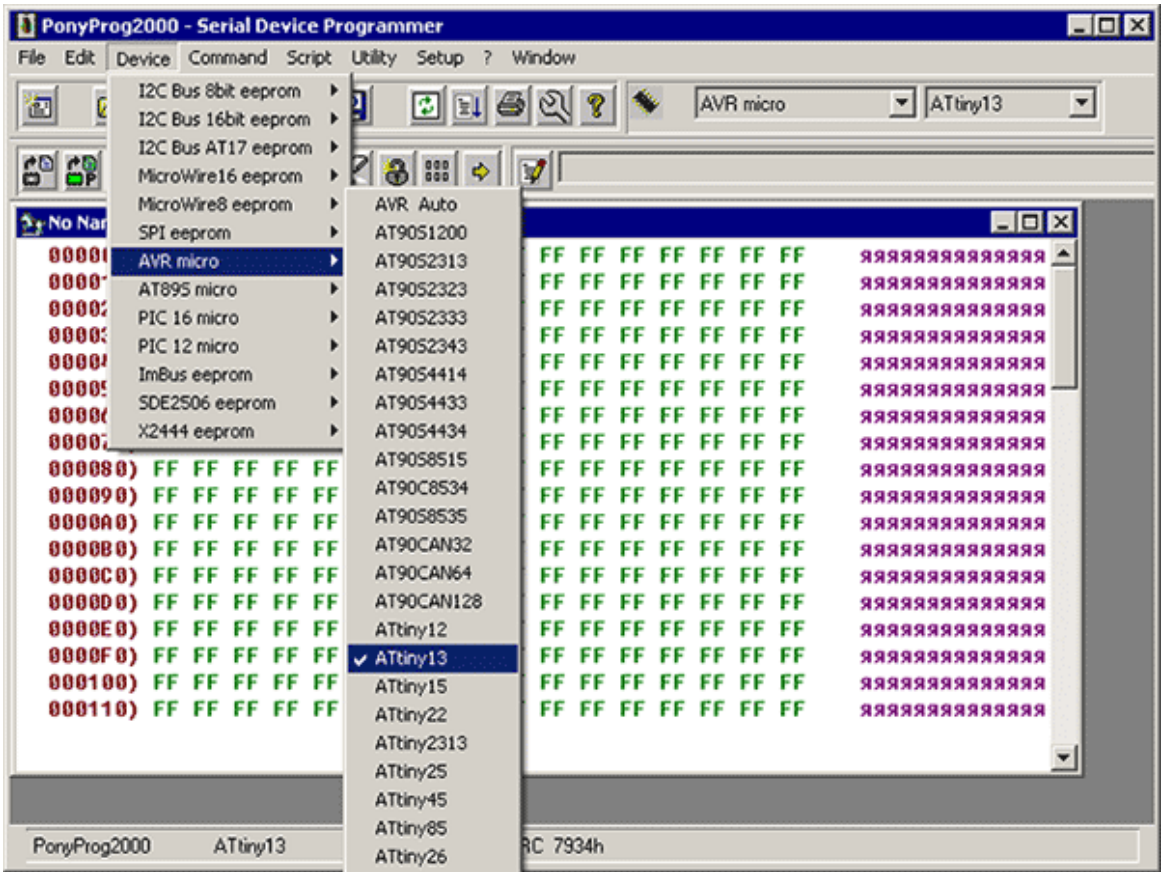


Рисунок 1.55 – Вікно завершення режиму калібровки

Обираємо AVR-мікроконтролер, з яким будемо працювати, переходимо до вкладки Device→AVR micro та обираємо мікроконтролер, наприклад Attiny13 (рис. 1.56).

Для читання «прошивки» мікроконтролера переходимо до вкладки Command→Read All (рис. 1.57) та очікуємо завершення процесу читання (рис. 13) «прошивки » мікроконтролера.

У вікні програми PonyProg2000 виводиться вміст «прошивки» мікроконтролера. Для збереження «прошивки», яку прочитали з мікроконтролера, обираємо режим File Save Device File As, обираємо ім’я файлу, тип файлу "*.HEX" та натискаємо кнопку "Сохранить".



Риснок 1.56 – Вибір типу мікроконтролера для роботи у PonyProg

Рисунок 1.57 – Вікно вибору режиму читання «прошивки»
з мікроконтролера

Рисунок 1.58 – Вікно процесу та результату читання «прошивки»
з мікроконтролера

Для запису «прошивки» у мікроконтролер відкриваємо файл «прошивки». Для цього обираємо File→Open Device File (рис. 1.59, а). У вікні, що відкриється, встановлюємо тип файлу "*.HEX" та відкриваємо файл з «прошивкою». Для програмування мікроконтролера обираємо Command→Write All (рис. 1.59, б). Процес програмування наведений на рис. 1.60.

а) б)

Рисунок 1.59 – Вибір «прошивки» (а) та режиму програмування (б)

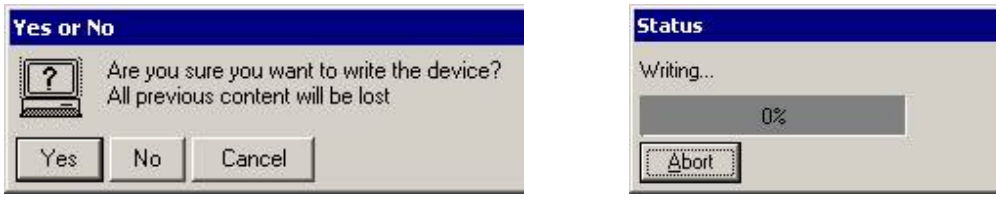


Рисунок 1.60 – Процес програмування мікроконтролера

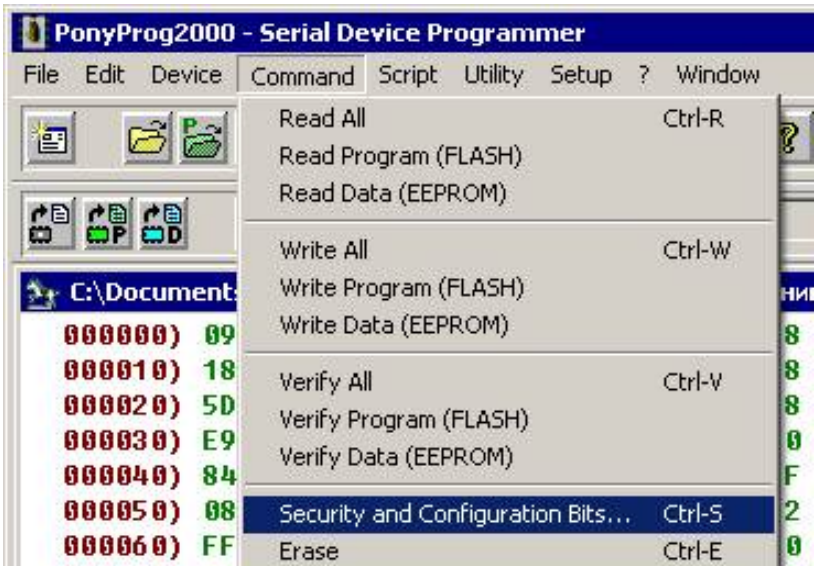


Рисунок 1.61 – Вибір режиму встановлення Fuse bits

Для знищення інформації з мікроконтролера обираємо Command→Erase. Щойно процес завершиться, PonyProg2000 повідомить повідомленням «Erase succesful».

Для встановлення **Fuse bits** мікроконтролера обираємо Command→Security and Configuration Bits (рис. 1.61). Для читання Fuse bits, що встановлені у мікроконтролері обираємо кнопку "Read" (рис. 1.62). Для запису Fuse bits встановлюємо необхідні біти у чек-боксах та натискаємо кнопку "Write" (рис. 1.62).

Рисунок 1.62 – Вікно встановлення Fuse bits у програмі PonyProg

1.9.3 Програмування у середовищі AVRDUDE та SinaProg

AVRDUDE PROG 3.1 призначена для програмування мікроконтролерів AVR під всі версії Windows через USB (рис. 1.63) [28–30].

Інтуїтивно зрозумілий інтерфейс, можливість вибору інверсних (PonyProg) та прямих (UniProf) Fuses біт, самостійне додавання мікроконтролерів та програматорів, що підтримують AVRDUDE .

Підтримує такі МК:

AT90CAN128,	ATmega325,
AT90CAN32,	ATmega3250,
AT90CAN64,	ATmega328p,
ATmega128,	ATmega329,
ATmega1280,	ATmega3290,
ATmega1281,	ATmega329p,
ATmega1284p,	ATmega3290p,
ATmega128RFA1,	ATmega32U4,
ATmega16,	ATmega48,
ATmega162,	ATmega8,
ATmega164p,	ATmega8515,
ATmega168,	ATmega8535,
ATmega169,	ATmega88,
ATmega2560,	ATtiny13,
ATmega2561,	ATtiny2313,
ATmega32,	ATtiny261.
ATmega324p,	

Підтримує такі програматори: USBasp, USBtiny, AVR ISP, SI-Prog, AVR910, STK200, STK500, JTAG ICE.

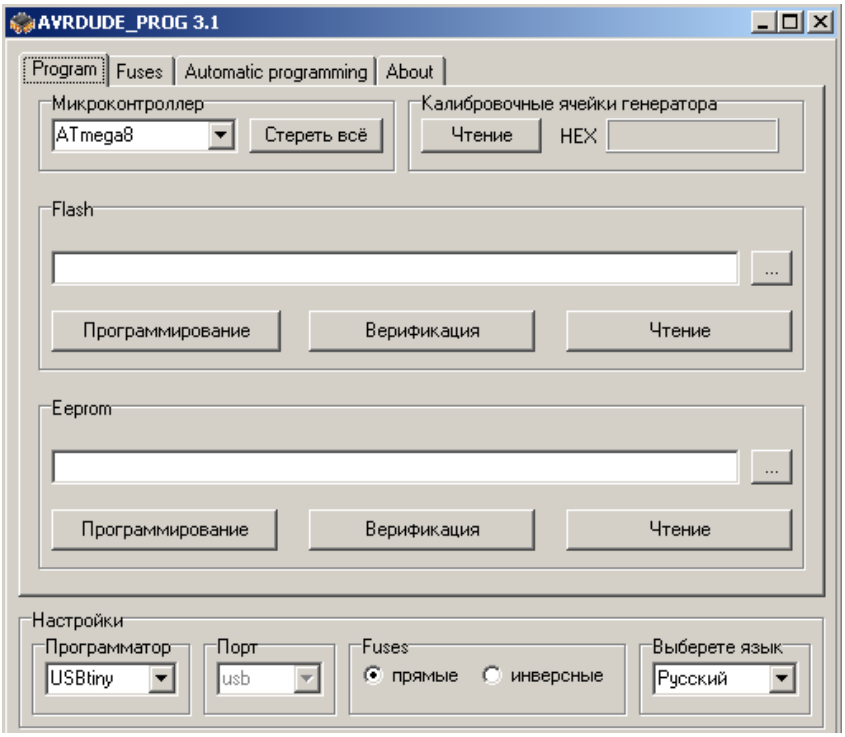


Рисунок 1.63 – Інтерфейс програми AVRDUDE PROG 3.1

Програма SinaProg портативна та не вимагає встановлення. До її складу входить AVRdude. Після запуску програми ми побачимо головне вікно (рис. 1.64) [28, 29, 30].

Опис інтерфейсу програми зверху–вниз:

- вікно вибору шляху та імені для HEX-файлу;
- вікно системних повідомлень;
- стрілочка показує/ховає вікно з логом повідомлень від AVRdude;
- прогрес-бар – відображає процес роботи з мікроконтролером;
- поле роботи з Flash-пам'яттю мікроконтролера. Програмування, перевірка (порівнюється вміст пам'яті з HEX-файлом) та читання (до процедури читання має бути вибраний HEX-файл у верхньому вікні – інакше буде виводиться помилка);
- поле роботи з EEPROM;
- поле вибору мікроконтролера – у списку обираємо мікроконтролер, з яким працюватиме програматор, кнопка «Search» запускає процес перевірки відповідності обраного мікроконтролера підключеному (корисно для перевірки працездатності ланки програматор-шлейф-мікроконтролер);
- робота з Fuse bits. Вікно пресетів та кнопку «Program» не чіпаємо – це для роботи з пресетами. Кнопка «Advanced ...» відкриває вікно встановлення Fuse bits (рис. 1.65);
- поле вибору програматора. У списку, що випадає, обираємо тип, наприклад, USBtiny, та порт – USB (у нових версіях SinaProg є можливість вибору різних портів та швидкості роботи з ними).

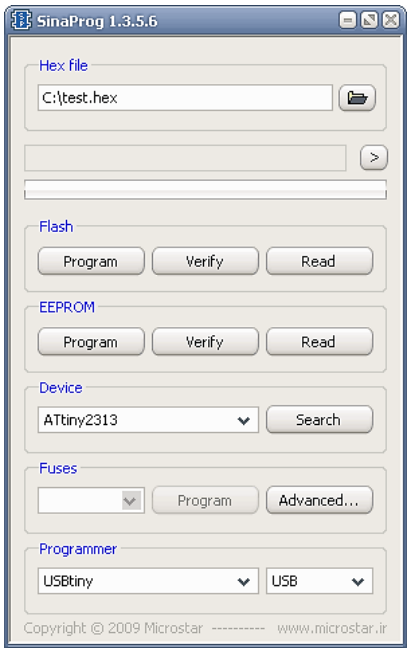


Рисунок 1.64 –
Інтерфейс програми Sina Prog

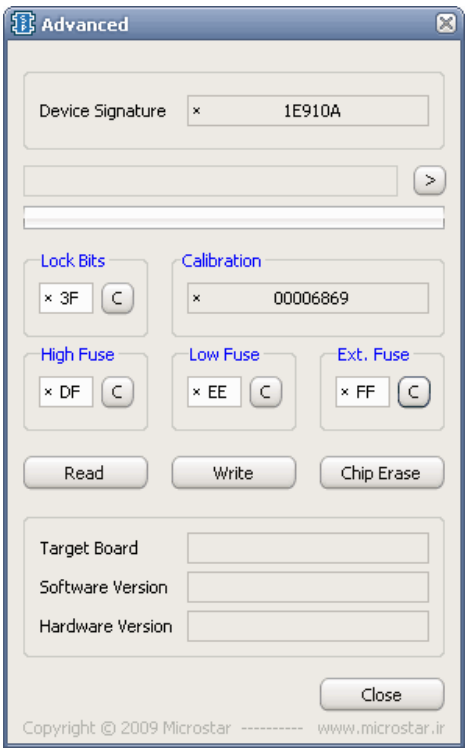


Рисунок 1.65 –
Вікно «Advanced» роботи з Fuse bits

Вікно «Advanced» містить сигнатуру мікроконтролера (Device Signature), значення калібровки для генератора (Calibration). Fuse bits розбиті на чотири байти (Lock Bits, High Fuse, Low Fuse, Ext. Fuse). При відкритті вікна дані автоматично зчитуються з мікроконтролера. Запис Fuse bits можливий у випадках:

1. Записом у відповідні вікна значень всього байту (шістнадцяткові числа).

2. За допомогою Fuse-калькулятора (рис. 1.66). Для переходу в цей режим тиснемо «C» біля потрібного байта Fuse. Відкриється вікно калькулятора, де і вибираються потрібні режими роботи. Після того як всі значення виставлені тиснемо «Write» (рис. 1.65). Значення Fuse bits тут не як по DataSheet, а інверсно. Чек-бокс дозволяється вибрану функцію. У такому разі це логічно, якщо сприймати калькулятор як настроювач конфігурації.

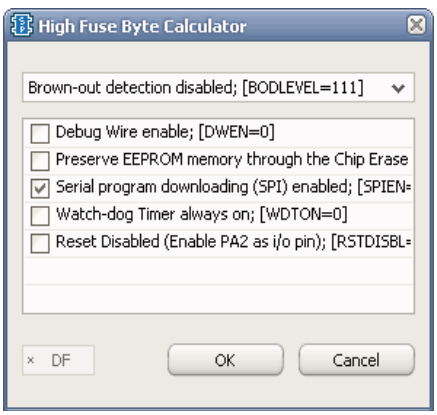


Рисунок 1.66 – Вікно Fuse-калькулятора програми SinaProg

Питання для самоконтролю

1. Опишіть основні етапи процесу написання програм для МК AVR.
2. Охарактеризуйте ключові переваги компілятора Асемблера.
3. Які Ви знаєте компілятори мови С?
4. Опишіть принципи роботи в інтегрованому середовищі розробки програм AVR Studio.
5. Які є особливості роботи в інтегрованому середовищі розробки програм WinAVR?
6. Дайте характеристику інтегрованому середовищу розробки програм ICCAVR.
7. Виконайте порівняльну характеристику інтегрованих середовищ розробки CodeVisionAVR та AVR Builder.
8. Які базові переваги роботи в інтегрованому середовищі розробки програм FLOWCODE?
9. Опишіть принципи роботи у програмному симуляторі Proteus VSM.
10. Якими ключовими характеристиками володіють особливості роботи в таких середовищах програмування: AVR Studio, PonyProg2000, AVRDUDE та SinaProg?

2 АПАРАТНІ ЗАСОБИ ПІДТРИМКИ МІКРОКОНТРОЛЕРІВ AVR

Донині поширеним способом створення прототипу майбутнього пристрою був такий: підібравши електронні компоненти, розробник брав у руки паяльник і збирав на макетних платах окремі вузли або пристрій в ціле. Далі починався процес налагодження: виправлення помилок принципової схеми, встановлення режимів роботи, уточнення параметрів компонентів, що застосовуються. Цей варіант не втратив своєї актуальності і нині, але застосовується тільки при розробках простих пристроїв, при виробництві одиничних екземплярів і при обмеженому бюджеті. Але якщо на перший план виходять такі критерії, як швидкість, зручність, надійність розробки, то без професійних засобів не обійтися.

До переваг використання засобів розробки можуть належати:

- зменшення часу виходу готової продукції;
- зменшення матеріальних витрат і ризику при розробці;
- використання власних ресурсів для прискорення розробки;
- вільне використання власних розробок надалі.

До спеціалізованих професійних інструментальних засобів налагоджування належать:

1. Оціночні та демонстраційні плати (Evaluation & demonstration board). Дозволяють швидко ознайомитися з тим чи іншим приладом або набором приладів та в короткі терміни розробити на ньому пристрій.

2. Внутрішньосхемні емулятори (In-circuit emulator). Найбільш потужні й універсальні налагоджувальні інструменти, що являють собою набір апаратно-програмних засобів, які дозволяють заміщувати собою емульований мікроконтролер у реальній схемі.

3. Програмні симулятори (Simulator). Програмні засоби, здатні імітувати роботу мікроконтролера і його пам'яті.

4. Відлагоджувачі (Debugger). Свого роду міст між розробником і пристроєм налагодження, що дозволяє користувачеві одночасно контролювати перебіг виконання програми і бачити відповідність між вихідним текстом, образом програми в машинних кодах і станом всіх ресурсів емульованого мікроконтролера.

5. Емулятори ПЗП (ROM emulator). Програмно-апаратні засоби, здатні заміщувати ПЗП на платі, що відлагоджується, шляхом підстановки замість нього ОЗП і завантаження програми за допомогою комп'ютера через один із стандартних інтерфейсів.

6. Програматори (Programmer). Пристрої, що дозволяють програмувати мікросхеми пам'яті, МК і програмовані логічні інтегральні схеми.

7. Інтегровані середовища розробки (Integrated Development Environment).

2.1 Внутрішньосхемні відлагоджувачі

Найважливіший інструмент будь-якого розробника програмного забезпечення – внутрішньосхемний відлагоджувач. Він з'єднує мікроконтролер з ПК через спеціальний інтерфейс для налагодження та програму-

вання, так відкривається доступ до внутрішніх алгоритмів функціонування мікроконтролера. Завантаження програми в мікроконтролер займає всього кілька секунд, при цьому не потрібно видаляти або перепаювати будь-які елементи. Запрограмований відлагоджувач може управляти процесором, запускаючи виконання програми, зупиняючи або реалізуючи її в покроковому режимі, а також зчитувати дані в режимі реального часу з усіх областей пам'яті мікроконтролера і регістрів введення/виведення. Для пристроїв AVR існує три внутрішньосхемних відлагоджувача [5, 14, 15, 17].

AVR Dragon. Внутрішньосистемний відлагоджувач для 8- і 32-розрядних мікроконтролерів AVR з підтримкою внутрішньосхемною налагодження (OCD). З його допомогою можна виконувати символьне налагодження на всіх пристроях з підтримкою внутрішньосхемною налагодження (OCD) з використанням режимів SPI, JTAG, PDI (на окремих пристроях), високовольтного послідовного програмування і паралельного програмування, а також стандартне налагодження через інтерфейси SPI, JTAG, PDI. Робоча область дозволяє створювати власні схеми або додавати гнізда для бажаних пристроїв. Відлагоджувач підтримує технологію NanoTrace (залежно від модуля OCD на пристрої AVR) з використанням пам'яті цільового пристрою.

JTAGICE 3. Внутрішньосистемний відлагоджувач та програматор середнього рівня для 8- та 32-розрядних мікроконтролерів AVR. Засіб розробки середнього рівня для 8- і 32-розрядних мікроконтролерів Atmel AVR з підтримкою внутрішньосхемною налагодження для проведення символьного налагодження на рівні вихідного коду, роботи в режимі NanoTrace (якщо підтримується пристроєм) і програмування.

AVR ONE!. Професійний інструмент розробки для всіх 8- та 32-розрядних пристроїв AVR з підтримкою внутрішньосхемного налагодження (OCD). Відлагоджувач Atmel AVR ONE! застосовується для символьного налагодження на рівні вихідного коду, трасування програм та програмування пристроїв. Він забезпечує повний цикл розробки і є найшвидшим налагоджувальним інструментом, що пропонується компанією Atmel, дозволяє програмувати в режимах SPI, JTAG, PDA і aWire, а також здійснювати налагодження через інтерфейси debugWIRE, JTAG, PDI і aWire. Також підтримує функцію LiveDebug, і допоміжний інтерфейс Nexus для високошвидкісної трасування програм, даних на частоті до 200 МГц в буферному або потоковому режимі. Заснований на потужній матриці FPGA, що забезпечує високошвидкісну передачу даних між ПК-хостом і пристроєм AVR, сприяючи швидкому завантаженню програм та зменшенню часу відгуку при налагодженні.

2.2 Засоби розробки для мікроконтролерів фірми Atmel

Сучасні засоби розробки для мікроконтролерів AVR фірми Atmel та сторонніх виробників розглянуто в [5, 14, 15, 17].

ICE 200 – внутрішньосхемний емулятор фірми Atmel для налагодження пристроїв на МК набору AVR (ATtiny12, AT90S2313, AT90S2333/4433, AT90S4414/8515, AT90S4434/8535). Емулятор підключається до пристрою

за допомогою спеціальної емуляційної головки і може використовуватися з платами, на яких встановлені панельки DIP8, DIP20, DIP28 і DIP40.

JTAGICE – внутрішньосхемний JTAG емулятор з фоновим налагодженням програми користувача, який може бути використаний як внутрішньосхемний програматор для мікроконтролерів AVR, що мають JTAG-інтерфейс.

STK500 – система відлагодження, яка полегшує роботу з AVR-мікроконтролерами та забезпечує підтримку програмування через паралельний та послідовний інтерфейси. Додатково система може бути використана як ISP-програматор. Використовуючи інтегровану середу розробки AVR Studio ATSTK500 забезпечує режими симуляції та емуляції, а також внутрішньосхемного програмування AVR-мікроконтролерів.

STK501 – модуль розширення до STK500, що дозволяє збільшити список пристроїв, що підтримуються: ATmega64, ATmega103, ATmega128.

STK502 – модуль розширення, розроблений для підтримки мікроконтролерів ATmega169. У ATSTK502 входять роз'єми, перемички і апаратні засоби для повної підтримки всіх особливостей mega169, а також управління вбудованим LCD.

STK600. Це повноцінний початковий комплект із системою розробки для 8- та 32-розрядних мікроконтролерів AVR. Комплект оснащений функціями моделювання та випробування нових проектів. Пристрої AVR взаємодіють з комплектом STK600 за допомогою інноваційної сендвіч-системи плат для підключення та маршрутизації, тобто передачі сигналів від пристрою до відповідного обладнання.

Сендвіч-система складається з базової плати з гніздами, на якій розміщується пристрій AVR, і плати маршрутизації сигналів (для кожного пристрою своя карта), яка направляє сигнали до різних блоків основної плати STK600 залежно від пристрою. Така система спрощує апаратне налаштування при переході від одного мікроконтролера AVR до іншого, оскільки всі сеанси зв'язку пристрою з материнською платою визначаються платою маршрутизації, специфічною для такого пристрою. Система маршрутизації повністю пасивна. У перенаправленні сигналів не беруть участь електронні компоненти, тому всі контакти введення/виведення на роз'ємах доступні безпосередньо, без необхідності зміни електричних схем.

Комплект забезпечує доступ до всіх виводів пристрою, а також підтримку корисних апаратних функцій таких, як: кнопки, світлодіодні індикатори та флеш-пам'ять даних Atmel DataFlash; являють собою повноцінну систему моделювання та випробування нових рішень. Безкоштовні програмні платформи AVR Studio і AVR32 Studio дозволяють розробникам писати і компілювати внутрішнє програмне забезпечення для мікроконтролерів, використовуючи асемблер або мову С, а також завантажувати готові програми в цільові пристрої AVR.

Плата також оснащена регульованим джерелом напруги і регульованим генератором тактових імпульсів. Рівень напруги і частота генератора

оперативно регулюються через середовища AVR Studio або AVR32 Studio, дозволяючи розробникам проводити випробування продуктивності при різних рівнях напруги і робочих частотах шляхом натискання всього однієї кнопки.

2.3 Апаратні засоби підтримки проектування та відлагодження систем

2.3.1 Логічні аналізатори та осцилографи змішаних сигналів

Логічні аналізатори є універсальними приладами для аналізу функціонування цифрових систем [14]. Вони дозволяють контролювати логічний стан декількох десятків точок системи протягом заданого проміжку часу і видати інформацію про їхній стан у візуальному (на екрані монітора) або друкованому вигляді. Форма подання може бути символьна або графічна (часові діаграми сигналів). Вхідні канали аналізатора підключаються до точок контролю за допомогою зондів-кліпсів. Число каналів у сучасних аналізаторах зазвичай становить від 16 до 200. Запуск аналізатора проводиться автоматично при надходженні на певні канали заданого коду (адреси, даних або комбінації керувальних сигналів) або послідовності кодів. Після запуску в пам'ять аналізатора записується послідовність значень логічних сигналів в точках контролю. Обсяг цієї пам'яті визначає число контрольованих точок на часовій осі (глибину контролю), яке може становити від декількох тисяч до сотень тисяч. На екран виводяться кілька десятків точок для кожного каналу з можливістю перегляду всієї записаної в пам'яті послідовності станів. Максимальна частота дискретизації часових інтервалів для різних моделей аналізаторів має значення від декількох десятків до сотень мегагерц.

Логічні аналізатори, які випускаються провідними виробниками електронно-вимірювальної апаратури: Hewlett-Packard, Tektronix, John Fluke і низкою інших, – реалізуються у вигляді автономних вимірювальних приладів з власним дисплеєм або аналізаторів блоків, що підключаються до персонального комп'ютера. Наведемо основні характеристики логічних аналізаторів компанії Hewlett-Packard, які широко використовуються для налагодження систем на базі мікроконтролерів [5].

Аналізатори серії HP1660 є автономні прилади, які дозволяють контролювати від 34 до 136 каналів з глибиною до 8К точок, забезпечуючи контроль логічних станів з частотою до 100 МГц при роздільній здатності 2 нс (частота дискретизації 500 МГц). Окремі моделі цієї серії містять також двоканальний осцилограф зі смугою пропускання 250 МГц або генератор послідовностей 32-розрядних тестових сигналів з частотою до 200 МГц. Аналізатори серії HP 1670 забезпечують глибину контролю до 128К точок при інших аналогічних параметрах. Ці серії логічних аналізаторів найбільш ефективні для налагодження систем на базі 8-розрядних мікроконтролерів набору AVR.

Ширші можливості налагодження реалізують аналізаторів HP 16600, HP 16700 компанії Hewlett-Packard, які працюють із зовнішніми монітором, клавіатурою і мишкою. Ці системи побудовані за модульним принципом, дозволяючи користувачеві вибрати необхідний йому набір стандартних функціональних модулів для виконання різних процедур контролю пристрою, що відлагоджуються. Системи серії HP 16600 мають від 68 до 204 каналів, що виконують аналіз логічних станів з частотою до 100 МГц при роздільній здатності 4 нс (частота дискретизації 250 МГц) і глибині контролю 128К точок. Є можливість підключення до системи вимірювального модуля, що виконує функції цифрового осцилографа або генератора тестових послідовностей, і модуля схемного емулятора.

Особливо корисним у процесі налагодження цифрових систем є використання осцилографів змішаних сигналів, які дозволяють одночасно контролювати логічний стан і значення аналогових сигналів в різних точках схеми. У цьому класі засобів налагодження оптимальним за комплексом технічних і експлуатаційних характеристик є прилад HP54645D компанії Hewlett-Packard [5], який містить двоканальний цифровий осцилограф із смугою пропускання 100 МГц (частота дискретизації 200 МГц), чутливістю 10 мВ/поділ, і 16-канальний логічний аналізатор з частотою дискретизації 400 МГц, глибиною контролю до 2М точок. Комбінація синхронно запуску осцилографа і логічного аналізатора з великою глибиною контролю відкриває нові можливості для виявлення причин виникнення збоїв і завад при роботі мікроконтролерних систем. Такий прилад дозволяє визначати причини виникнення завад на шинах живлення, виявляти імпульсні завади в лініях зв'язку, контролювати спільну роботу аналогових і цифрових блоків, налагоджувати процедури послідовного обміну даними при спотвореннях форми переданих сигналів, а також розв’язувати багато інших проблем, які досить важко вирішити іншими способами.

2.3.2 Схемні емулятори і схемні симулятори

Схемний емулятор (СЕ) являє собою програмно-апаратний комплекс, який у процесі налагодження заміщає мікроконтролер в реалізованої системі. У результаті такої заміни функціонування системи, що відлагоджується, стає наочною і контрольованою. Розробник отримує можливість візуального контролю за роботою системи на екрані дисплея та управління її роботою шляхом встановлення певних керуючих сигналів і модифікації вмісту регістрів та пам'яті. Завдяки наявності таких можливостей СЕ є найбільш універсальним та ефективним налагоджувальний засобом, що використовується на етапі комплексного налагодження системи [5, 14].

Найбільш широке застосування отримали СЕ, що підключаються до базового комп'ютера або робочої станції. Зазвичай, такі СЕ конструктивно оформлені у вигляді приладу, розміщеного в окремому корпусі з автономним джерелом живлення і з’єднаним з базовим комп'ютером. За допомогою плоского кабелю до СЕ підключається емуляторна головка, яка має вилку для ввімкнення в систему замість емульованого мікроконтролера. У голівці розміщується емулятор, який виконує ті ж функції, що і мікро-

контролер, але працює під управлінням комп'ютера. Більшість СЕ призначений для роботи з певним набором мікроконтролерів, причому для емуляції кожної моделі набору використовується відповідна головка.

До структури типового СЕ входять такі блоки:

– емулятор мікроконтролера (у голівці);

– пам'ять траси, яка зберігає значення сигналів, що встановлюються на виводах мікроконтролера в процесі виконання програми;

– блок контрольних переривань, який реалізує зупинки в контрольних точках, заданих користувачем з клавіатури комп'ютера;

– емуляційна пам'ять (ОЗП), яка замінює в процесі налагодження внутрішнє ПЗП мікроконтролерів або інші розділи пам'яті, зовнішній доступ до яких під час налагодження обмежений;

– таймер, який використовується для контролю часу виконання фрагментів програми.

СЕ дозволяє вводити в систему тестову або робочу програму і контролювати її виконання, забезпечуючи переривання в контрольних точках. Умовами переривання можуть бути різні комбінації значень адреси, даних і керувальних сигналів, що надходять на виводи мікропроцесора або мікроконтролера, що емулюється. Ці комбінації задаються користувачем з клавіатури комп'ютера, що управляє. Після зупинки користувач може отримати на екрані інформацію про поточний стан будь-яких регістрів і комірок пам'яті системи. За допомогою пам'яті траси можна переглянути стан системної шини для певної кількості попередніх циклів виконання програми. Дизасемблер дозволяє аналізувати виконання програми відповідно до її вихідного тексту мовою асемблера. Пам'ять траси працює аналогічно пам'яті логічного аналізатора, тому СЕ може виконувати також і його функції. Об'єм пам'яті траси в різних СЕ дозволяє контролювати від 4К до 512К програмних циклів. Таймер слугує для визначення часу виконання фрагментів програми з урахуванням реальної тактової частоти системи.

Програмне забезпечення СЕ складається з монітора – службової програми, що забезпечує роботу всіх блоків під керуванням базового комп'ютера, транслятора, що дозволяє програмувати роботу системи мовою високого рівня або Асемблера, і відлагоджувача. Ці програмні засоби зазвичай функціонують у складі інтегрованого середовища програмування-налагодження. Більшість сучасних СЕ використовують символьні відлагоджувачі і дизасемблери, застосування яких робить процес налагодження більш простим і наочним. Програмне забезпечення СЕ реалізує видачу даних на екран монітора в зручному для користувача форматі.

Деякі моделі СЕ надають можливості аналізу ефективності виконуваної програми, забезпечуючи інформацію про частоту звернення до певних її фрагментів, і дозволяють виконувати налагодження мультипроцесорних систем за допомогою організації багатоемуляторних комплексів. СЕ, що реалізують перераховані вище функції, називають налагоджувальними комплексами або системами розвитку (development system).

Як правило, симулятор складається з відлагоджувача, моделі ЦП і пам'яті. Більш досконалі пристрої містять у своєму складі моделі вбудованих периферійних пристроїв (таймерів, портів, АЦП і систем переривань).

Симулятор має вміти завантажувати файли програм в усіх популярних форматах, максимально повно відображати інформацію про стан ресурсів МК, а також надавати можливості по симуляції виконання програми, що завантажується, у різних режимах. Під час налагодження модель виконує програму, і на екрані монітора комп'ютера відображається поточний стан моделі.

Завантаживши програму в симулятор, користувач може запускати її у покроковому або безперервному режимі, задавати умовні або безумовні точки зупинки, контролювати і вільно модифікувати вміст комірок пам'яті і регістрів МК. Симулятор дозволяє швидко перевірити логіку виконання програми, правильність виконання арифметичних операцій. Залежно від класу відлагоджувача деякі моделі симуляторів підтримують високорівневе налагодження програм. Симулятор може містити і ряд додаткових програмних засобів, наприклад інтерфейс зовнішнього середовища. Наявність такого інтерфейсу дозволяє створювати і гнучко використовувати модель зовнішнього середовища МК, що функціонує і впливає на програму, яка відлагоджується за заданим алгоритмом.

У реальній системі МК зчитує інформацію з підключених до нього пристроїв (датчиків), обробляє її та видає сигнали керування на виконавчі пристрої. Для того, щоб в простому симуляторі змоделювати роботу датчика, потрібно вручну змінювати поточний стан моделі периферійного пристрою, до якого в реальній системі підключений датчик. Але існує низка сучасних розробок програмних симуляторів, у яких, щоб імітувати зовнішні умови і ситуації, зазвичай використовується спеціальний файл вхідних впливів. Цей файл задає послідовність вхідних сигналів, що надходять на пристрій, який моделюється.

Наприклад, для мікроконтролерів AVR цей вхідний файл програмного симулятора може виглядати так:

000000000:00
000000006:F1
000000015:18
000000109:1C
000000203:61
000000250:10
000000344:1F
000000391:71
999999999:FF,

де кожен рядок містить – цикл: дані, що надходять на будь-який вказаний порт.

У деяких моделях симуляторів ця проблема імітації зовнішніх сигналів вирішена так, що симулятор має вбудований засіб для створення моделей підключених до МК зовнішніх пристроїв, включаючи засоби графічного відображення інформації. Очевидна особливість програмних симулято-

рів в тому, що завантажені в них програми виконуються в масштабі часу, відмінному від реального. Однак низька ціна, можливість налагодження навіть за відсутності макета пристрою роблять програмні симулятори дуже привабливим засобом налагодження. Необхідно також відзначити, що існує цілий клас помилок, які можна виявити тільки за допомогою симулятора.

2.3.3 Плати розвитку

Цей клас засобів проектування мікроконтролерних систем є найбільш численним [14, 15]. Умовно їх можна розділити на такі типи:

– оціночні комплекти (Evaluation kit) – набір розміщених на платі апаратних засобів для реалізації нескладних систем;

– налагоджувальні плати і системи (Evaluation board, system) – розміщені на платі програмно-апаратні комплекси, що забезпечують моделювання та налагодження систем на базі певних моделей мікроконтролерів;

– одноплатні контролери (single-board controller) – конструктивні комплекси, що призначені для використання як базові модулі при реалізації цільових систем промислового застосування.

Ці засоби можуть використовуватися для таких цілей:

– тестування та налагодження програмного забезпечення систем на реальних зразках мікроконтролерів;

– комплексне налагодження макета системи, що використовується, як зразок для реалізації прототипу системи;

– реалізація цільової системи, до складу якої входять плати розвитку як базові модулі;

– вивчення функціонування мікроконтролерів, отримання навичок їхнього практичного застосування.

Практично всі типи плат розвитку містять у своєму складі порти для підключення персонального комп'ютера. Найчастіше для цієї мети використовується послідовний обмін за стандартом RS-232. Ряд типів налагоджувальних і цільових плат мають також окреме поле для макетування користувачем додаткових пристроїв за допомогою проводового монтажу.

Зважаючи на велику різноманітність галузей і способів застосування номенклатура плат розвитку дуже широка і чітка межа між їх типами відсутні. У багатьох випадках налагоджувальні плати можуть використовуватися в складі цільових систем, а одноплатні комп'ютери часто слугують засобами моделювання і налагодження прототипу систем.

Оціночні комплекти і типові проекти

Для інженерів, яким не потрібна гнучкість початкового комплекту STK600, хорошою альтернативою може стати будь-який з оціночних комплектів або типових проектів. Вони специфічні для кожного пристрою і постачаються з мікроконтролером AVR і всіма необхідними компонентами.

Для початківців рекомендуються такі оціночні комплекти для роботи з мікроконтролерами AVR: AVR Xplained та AVR UC3 EVK1104 [14, 15].

AVR Xplained. Комплект для створення проектів на базі Atmel AVR XMEGA; оснащений МК ATmega128A1 та додатковими компонентами, що

дозволяють оцінити функції пристрою. Комплект має всі необхідні апаратні засоби для оцінки високоточних аналогових функцій пристроїв Atmel AVR і демонстрації способів взаємодії із зовнішніми модулями пам'яті. Він містить пам'ять SDRAM 8 МБ, флеш-пам'ять DataFlash 8 МБ, підключені до Atmel ATxmega128A1, а також 8 кнопок і 8 світлодіодних індикаторів для взаємодії з користувачем. Як комунікаційний шлюз між портом USB COM МК ATxmega128A1 і джерелом живлення виступає інтерфейс USB. Для зчитування даних датчика температури і потенціометра можна використовувати АЦП, а для генерації звуку на монодинамік – ЦАП.

EVK1104. Оціночний комплект для 32-розрядного мікроконтролера Atmel AVR AT32UC3A3256 з комунікаційними інтерфейсами. Цей оціночний комплект поєднує 32-розрядний мікроконтролер Atmel AVR UC3 з багатим набором комунікаційних інтерфейсів, включаючи On-The-Go і SDcard, а також флеш-пам'ять NAND з ECC і 16-розрядний стерео-ЦАП.

QT600. Повноцінний комплект розробника для реалізації кнопок, повзунків і коліс прокрутки. Atmel QT600 дозволяє експериментувати з сенсорною технологією Atmel і є найпростішим способом аналізу та оцінки сенсорних рішень Atmel. Підтримуються обидві технології Atmel для введення даних: QTouch, QMatrix.

Як програмне забезпечення для взаємодії з комплектом використовується середу QTouch Studio. Комплект містить інтерфейсну плату з роз'ємом USB, 3 мікроконтролерні плати (ATtiny88, ATmega324PA, ATxmega128A1), 3 сенсорні плати, що підтримують до 64 каналів, і кабелі.

Програматори

Особливу групу засобів розробки становлять програматори. За функціональним можливостям програматори умовно можна розділити на такі групи:

– спеціалізовані програматори для мікросхем пам'яті (EPROM, EEPROM, FLASH);

– спеціалізовані програматори для мікросхем пам'яті і внутрішньої пам'яті окремих наборів мікроконтролерів;

– універсальні програматори мікросхем пам'яті, внутрішньої пам'яті мікроконтролерів, мікросхем програмованої логіки (PLD).

Основні функціональні можливості сучасних програматорів:

– тестова колодка з нульовим зусиллям (ZIF-socket), що забезпечує багаторазовий надійний контакт з програмованої мікросхемою в корпусі DIP;

– для програмування мікросхем з корпусами, відмінними від DIP, програматори забезпечуються спеціальними адаптерами під відповідний тип корпуса;

– можливість оновлення програмного забезпечення для розширення кількості програмованих мікросхем;

– програмна встановлення параметрів програмування: Vccp, Vvpp;

– самотестування при включенні живлення;

– тестування правильності встановлення мікросхем;

– перевірка якості контакту по всіх виводах запрограмованої мікросхеми;

– захист всіх виводів мікросхеми від перенапруги і статичної електрики.

Інтегровані середовища розробки

Ідея єдності програмного і апаратного забезпечення систем на базі МК є дуже важливою. Об'єднання інструментальних засобів розробки програмного забезпечення з інструментальними засобами розробки апаратного забезпечення може стати важливою перевагою при розробці пристроїв.

Істотно полегшують і прискорюють процес розробки та налагодження мікропроцесорних систем так звані інтегровані середовища розробки. Вони поєднують у собі текстовий редактор для написання вихідних текстів, транслятори з асемблера та С, лінкер, відлагоджувач, довідкову інформацію по МК і інші засоби, необхідні розробникові. Налаштування трансляторів, линкера й інших компонентів проводиться не методом вказівки ключів у командному рядку, а у вигляді діалогових вікон, де потрібно тільки розставити «галочки» у потрібних місцях. Перетворення початкового програмного коду в файл машинних кодів запускається натисканням однієї клавішею.

Поява інтегрованих середовищ розробки програм ще більше підвищило ефективність створення програм для МК, дозволило розробникові зосередитися на суті розв'язуваної задачі і відволіктися від конкретних деталей її реалізації. Інтегровані пакети для розробки програм випускають кілька фірм, пакети різних виробників схожі між собою за функціями, але розрізняються наданими сервісними можливостями, зручністю роботи і якістю генерованого машинного коду.

Як правило, хороше інтегроване середовище об'єднує наявні засоби налагодження (внутрішньосхемний емулятор, програмний симулятор і програматор) і забезпечує роботу програміста з текстами програм в стилі діалогових вікон.

Інтегроване середовище дозволяє:

– використовувати вбудований многофайловий текстовий редактор, спеціально орієнтований на роботу з вихідними текстами програм;

– спостерігати одночасно у багатовіконному режимі діагностику виявлених при компілюванні помилок і одночасне редагування вихідного коду програми;

– вести паралельну роботу над декількома проектами. Менеджер проектів дозволяє використовувати будь-який проект як шаблон для новостворюваного. Опції використовуваних компіляторів і список вихідних файлів проекту встановлюються в діалогових меню і зберігаються в межах проекту, усуваючи необхідність роботи з незручними bat-файлами: піддавати перекомпілюванню, лише редагувати модулі; завантажувати відлагоджену програму в наявні засоби налагодження і працювати з ними без виходу з оболонки; підключати до оболонки практично будь-які програмні засоби.

Останнім часом функції інтегрованих середовищ розробки стають приналежністю програмних інтерфейсів найбільш «просунутих» емуляторів і відлагоджувачів симуляторів. Такі функціональні можливості в поєднанні з дружнім інтерфейсом істотно прискорюють роботу програміста.

Так, вибираючи інструментальні засоби налагодження, доцільно брати до уваги такий комплекс показників:

- підтримка можливо більшої кількості мікроконтролерів;
- різноманітність вбудованих інтерфейсів (RS-232, IEEE1284/LPT, USB) і додаткових компонентів, що розширюють функціональні можливості;
- наявність на платі поля для макетування і контактів для безпосереднього підключення до виводів приладу;
- можливість підключення додаткових модулів;
- хороше співвідношення високої функціональності до ціни;
- наявність демонстраційних програм і прикладів використання;
- універсальне живлення (живлення від зовнішнього джерела, інтерфейсів ПК, батареї, наявність вбудованих регуляторів напруги).

Питання для самоконтролю

1. Дайте загальну оцінку апаратним засоби підтримки мікроконтролерів AVR.
2. Які є спеціалізовані професійні інструментальні засоби налагоджування?
3. Основна функція вутрішньосхемного відлагоджувача.
4. Наведіть кілька основних засобів розробки для мікроконтролерів фірми Atmel.
5. Які Ви знаєте апаратні засоби підтримки проектування та відлагодження систем?
6. Які Ви знаєте апаратні засоби аналізу функціонування цифрових систем?
7. Який програмно-апаратний комплекс під час налагодження заміщує мікроконтролер в реалізованої системі?
8. Розкажіть про типи плат розвитку.
9. Що таке оціночні комплекти і типові проекти?
10. Призначення програматорів.
11. Які переваги несе в собі використання інтегрованих середовищ розробки?

3 ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR

3.1 Загальна інформація

МК набору AVR підтримують такі режими програмування: режим послідовного програмування (по інтерфейсу SPI); режим високовольтного паралельного програмування (HVPP); режим програмування через інтерфейс JTAG.

Під високою напругою розуміють напругу керування (12 В), що подається на вивід RESET мікроконтролера для переведення останнього в режим програмування. Які з режимів програмування підтримують деякі МК AVR, можна дізнатися, звернувшись до табл. 3.1.

Таблиця 3.1 – Режими програмування мікроконтролерів набору AVR

Режим програмування	ATtiny13	ATtiny25	ATtiny2313	ATmega8515x/8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x
1. Послідовний (інтерфейс SPI)	+	+	+	+	+	+	+	+	+
2. Високовольтний паралельний	+	+	+	+	+	+	+	+	+
3. По інтерфейсу JTAG						+	+		+

До того ж, МК AVR мають можливість самопрограмування, тобто зміна вмісту пам'яті програм під керуванням самого мікроконтролера.

У процесі програмування можуть виконуватися такі операції:

- знищення кристала (chip erase);
- читання/запис FLASH-пам'яті програм;
- читання/запис EEPROM-пам'яті даних;
- читання/запис конфігураційних комірок;
- читання/запис комірок захисту;
- читання комірок ідентифікатора;
- читання байта калібрування.

Усі моделі мікроконтролерів поставляються зі стертою пам'яттю програм та пам'яттю даних (у всіх комірках знаходиться число $FF) і придатні до негайного програмування.

3.1.1 Захист коду і даних

Вміст FLASH-пам'яті програм, а також вміст EEPROM-пам'яті даних може бути захищеним від запису і/або читання за допомогою програмування комірок захисту (Lock Bits) LB1 і LB2. Можливі режими захисту, які відповідають різним станам цих комірок, наведені в табл. 3.2.

Таблиця 3.2 – Режими захисту коду і даних

№ режиму	Комірки захисту		Опис
	LB1	LB2	
1	1	1	Захист коду і даних відключений
2	0	1	Запис FLASH і EEPROM заборонений
3	0	0	Заборонені запис і читання FLASH і EEPROM

У режимах 2 та 3 забороняється також зміна конфігураційних комірок. Тому ввімкнення захисту потрібно виконувати в найостаннішу чергу, після програмування інших областей пам'яті мікроконтролера.

У мікроконтролерах AVR набору ATmega, крім ATmega48x, є чотири додаткові комірки захисту – BLB02, BLB01, BLB12 і BLB11. Комірки BLB02: BLB01 визначають рівень доступу із секції завантажувача до коду, розташованому в секції прикладної програми, а комірки BLB12: BLB11, навпаки, визначають рівень доступу з секції прикладної програми до коду, розташованому в секції завантажувача. Можливі режими захисту, які відповідають різним станам цих комірок, наведені в табл. 3.3 та табл. 3.4 відповідно. Усі перераховані комірки захисту згруповані в одному байті. Розташування комірок захисту в ньому для різних моделей мікроконтролерів AVR наведено на рис. 3.1.

Таблиця 3.3 – Режими захисту секції прикладної програми

№ режиму	Комірки захисту		Опис
	BLB02	BLB01	
1	1	1	Немає жодних обмежень з доступу до коду, що розташований у секції прикладної програми
2	1	0	Команда SPM не може здійснювати запис за адресами, які перебувають у межах секції прикладної програми
3	0	0	Команда SPM не може здійснювати запис за адресами, які перебувають у межах секції прикладної програми; команда LPM (ELPM), що викликається із секції завантажувача, не може здійснювати читання з секції прикладної програми. Якщо таблиця векторів переривань розташована в секції завантажувача, то при виконанні коду з секції прикладної програми переривання заборонені
4	0	1	Команда LPM (ELPM), що викликається із секції завантажувача, не може здійснювати читання із секції прикладної програми. Якщо таблиця векторів переривань розташована в секції завантажувача, то при виконанні коду з секції прикладної програми переривання заборонені

Таблиця 3.4 – Режими захисту секції завантажувача

№ режиму	Комірки захисту		Опис
	BLB12	BLB11	
1	2	3	4
1	1	1	Немає жодних обмежень по доступу до коду, що розташований в секції завантажувача
2	1	0	Команда SPM не може здійснювати запис за адресами, які перебувають у межах секції завантажувача
3	0	0	Команда SPM не може здійснювати запис за адресами, які перебувають у межах секції завантажувача, і команда LPM (ELPM), що викликається із секції прикладної програми, не може здійснювати читання із секції завантажувача. Якщо таблиця векторів переривань розташована в секції прикладної програми, то при виконанні коду із секції завантажувача переривання заборонені
4	0	1	Команда LPM (ELPM), що викликається із секції прикладної програми, не може здійснювати читання із секції завантажувача Якщо таблиця векторів переривань розташована в секції прикладної програми, то при виконанні коду із секції завантажувача переривання заборонені

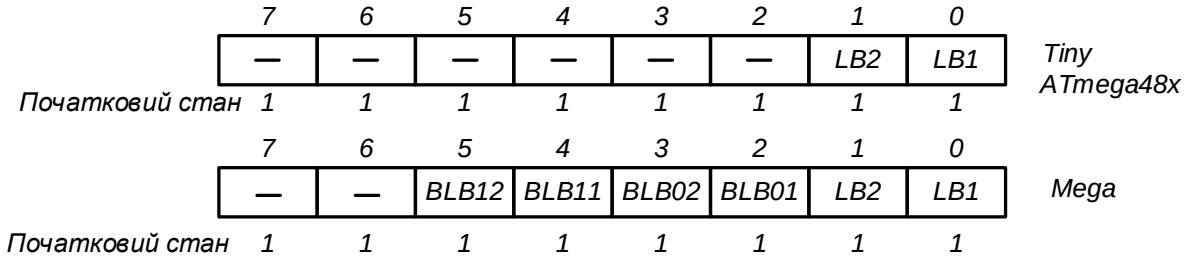


Рисунок 3.1 – Байт комірок захисту

У незапрограмованому стані у всіх комірках захисту міститься *1*, після програмування – *0*. Знищення комірок (запис в них логічної *1*) може бути зроблено тільки при виконанні команди «Знищення кристала», яка знищує також вміст FLASH- і EEPROM-пам'яті.

3.1.2 Конфігураційні комірки

Конфігураційні комірки (Fuse Bits) визначають різні параметри конфігурації мікроконтролера. Ці комірки розташовані в окремому адресному просторі, доступному тільки при програмуванні. Усі конфігураційні комірки згруповані в кілька байтів, а склад цих комірок залежить від конкретної моделі мікроконтролера. Наявність тих чи інших комірок у конкретному

мікроконтролері можна визначити з табл. 3.5, де в стовпцях, позначених «зірочкою», указані стани конфігураційних комірок за замовчуванням.

Основне призначення всіх конфігураційних комірок наведено в табл. 3.6. Для зміни вмісту конфігураційних комірок використовуються спеціальні команди програмування. Команда «Знищення кристала» на стан цих комірок не впливає. При програмуванні комірки захисту LB1 конфігураційні комірки блокуються. Тому конфігурацію мікроконтролера необхідно виконувати до програмування комірок захисту.

Таблиця 3.5 – Конфігураційні комірки мікроконтролерів AVR

Біт	ATmega8515		ATmega 8535		ATmega8x		ATmega16x/ 32x		ATmega64x/ 128x		ATmega48x	
	Назва	*	Назва	*	Назва	*	Назва	*	Назва	*	Назва	*
Молодший конфігураційний байт												
7	BODLEVEL	1	BODLEVEL	1	BODLEVEL	1	BODLEVEL	1	BODLEVEL	1	CKDIV8	0
6	BODEN	1	BODEN	1	BODEN	1	BODEN	1	BODEN	1	CKOUT	1
5	SUT1	1	SUT1	1	SUT1	1	SUT1	1	SUT1	1	SUT1	1
4	SUT0	0	SUT0	0	SUT0	0	SUT0	0	SUT0	0	SUT0	0
3	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0
2	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0
1	CKSEL1	0	CKSEL1	0	CKSEL1	0	CKSEL1	0	CKSEL1	0	CKSEL1	1
0	CKSEL0	1	CKSEL0	1	CKSEL0	1	CKSEL0	1	CKSEL0	1	CKSEL0	0
Старший конфігураційний байт												
7	S8515C	1	S8535C	1	RSTDISBL	1	OCDEN	1	OCDEN	1	RSTDISBL	1
6	WDTON	1	WDTON	1	WDTON	1	JTAGEN	0	JTAGEN	0	DWEN	1
5	SPIEN	0	SPIEN	0	SPIEN	0	SPIEN	0	SPIEN	0	SPIEN	0
4	CKOPT	1	CKOPT	1	CKOPT	1	CKOPT	1	CKOPT	1	WDTON	1
3	EESAVE	1	EESAVE	1	EESAVE	1	EESAVE	1	EESAVE	1	EESAVE	1
2	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	1
1	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	1
0	BOOTRST	1	BOOTRST	1	BOOTRST	1	BOOTRST	1	BOOTRST	1	BOOTRST	1
Додатковий конфігураційний байт												
7	—								—	1	—	1
6									—	1	—	1
5									—	1	—	1
4									—	1	—	1
3									—	1	—	1
2									—	1	—	1
1									M103C	0	—	1
0									WDTON	I	SELFPRGEN	1
* Значення біта												
Молодший конфігураційний байт												
7	CKDIV8	0	CKDIV8	0	CKDIV8	0	CKDIV8	0	CKDIV8	0	CKDIV8	0
6	CKOUT	i	CKOUT	1	CKOUT	1	CKOUT	1	CKOUT	1	CKOUT	1
5	SUT1	i	SUT1	1	SUT1	1	SUT1	1	SUT1	1	SUT1	1
4	SUT0	0	SUT0	0	SUT0	0	SUT0	0	SUT0	0	SUT0	0
3	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0
2	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0
1	CKSEL1	1	CKSEL1	1	CKSEL1	1	CKSEL1	1	CKSEL1	1	CKSEL1	1
0	CKSEL0	0	CKSEL0	0	CKSEL0	0	CKSEL0	0	CKSEL0	0	CKSEL0	0

Продовження таблиці 3.5

Біт	ATmega8515		ATmega 8535		ATmega8x		ATmega16x/ 32x		ATmega64x/ 128x		ATmega48x	
	Назва	*	Назва	*	Назва	*	Назва	*	Назва	*	Назва	*
Старший конфігураційний байт												
7	RSTDISBL	1	OCDEN	1	OCDEN	1	OCDEN	1	OCDEN	1	OCDEN	1
6	DWEN	1	JTAGEN	0	JTAGEN	0	JTAGEN	0	JTAGEN	0	JTAGEN	0
5	SPIEN	0	SPIEN	0	SPIEN	0	SPIEN	0	SPIEN	0	SPIEN	0
4	WDTON	1	WDTON	1	WDTON	1	WDTON	1	WDTON	1	WDTON	1
3	EESAVE	1	EESAVE	1	EESAVE	1	EESAVE	1	EESAVE	1	EESAVE	1
2	BODLEVEL2	1	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0
1	BODLEVEL1	1	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0
0	BODLEVEL0	1	BOOTRST	1	BOOTRST	1	BOOTRST	1	BOOTRST	1	BOOTRST	1
Додадковий конфігураційний байт												
7	–	1	–	1	–	1	–	1	–	1	–	1
6	–	1	–	1	–	1	–	1	–	1	–	1
5	–	1	–	1	–	1	–	1	–	1	–	1
4	–	1	M161C	1	–	1	–	1	–	1	–	1
3	–	1	BODLEVEL2	1	–	1	BODLEVEL2	1	–	1	–	1
2	BOOTSZ1	0	BODLEVEL1	1	BODLEVEL2	1	BODLEVEL1	1	BODLEVEL1	1	BODLEVEL2	1
1	BOOTSZ0	0	BODLEVEL0	1	BODLEVEL1	1	BODLEVEL0	1	BODLEVEL1	1	BODLEVEL1	1
0	BOOTRST	1	-	1	BODLEVEL0	1	RESERVED	1	RSTDISBL	1	BODLEVEL0	1
* Значення біта												

Таблиця 3.6 – Призначення конфігураційних комірок

Назва	ATmega8515x	ATmega8535x	ATmega 8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega 162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x ATmega645x/6450x	ATmega640x/1280x/1281x ATmega2560x/2561x	Призначення
1	2	3	4	5	6	7	8	9	10	11	12	13
RSTDISBL			+			+				+		Визначає функціонування виводу МК, поєднаного з виводом RESET (0 – контакт порту введення/виведення, 1 – вивід скидання)

Продовження таблиці 3.6

1	2	3	4	5	6	7	8	9	10	11	12	13
CKSEL	+	+	+	+	+	+	+	+	+	+	+	Визначає режим роботи тактового генератора, а також тривалість затримки скидання, T_{OUT}
BODLEVEL	+	+	+	+	+	+	+	+	+	+	+	Визначає поріг спрацьовування схеми BOR
BODEN	+	+	+	+	+							Дозволяє/забороняє функціонування схеми BOR (0 – дозволено, 1 – заборонено)
SPIEN [1)]	+	+	+	+	+	+	+	+	+	+	+	Дозволяє/забороняє програмування по інтерфейсу SPI (0 – дозволено, 1 – заборонено)
SUT	+	+	+	+	+	+	+	+	+	+	+	Визначає тривалість затримки скидання, T_{OUT}
WDTON	+	+	+		+	+	+	+	+	+	+	Визначає режим роботи сторожового таймера (0 – завжди включений, 1 – може бути виключений програмно)
CKOPT	+	+	+	+	+							Визначає функціонування тактового генератора, дія залежить від установок комірок CKSEL
EESAVE [2)]	+	+	+	+	+	+	+	+	+	+	+	Визначає вплив команди «Знищення кристала» на EEPROM-пам'ять (0 – не стирає, 1 – стирає)
BOOTSZ	+	+	+	+	+	+	+	+	+	+	+	Визначає розмір секції завантажувача
BOOTRST	+	+	+	+	+	+	+	+	+	+	+	Визначає положення вектора скидання
DWEN						+						Дозволяє/забороняє роботу інтерфейсу debugWire (0 – дозволено, 1 – заборонено)
OCDEN				+	+		+	+	+	+	+	Дозволяє/забороняє внутрішньосхемне налагодження (0 – дозволено, 1 – заборонено)
JTAGEN				+	+		+	+	+	+	+	Дозволяє/забороняє використання інтерфейсу JTAG (0 – дозволено, 1 – заборонено)
CKDIV8						+	+	+	+	+	+	Визначає початковий стан подільника системного тактового сигналу

Продовження таблиці 3.6

1	2	3	4	5	6	7	8	9	10	11	12	13
CKOUT						+	+	+	+	+	+	Визначає стан вихідного буфера системного тактового сигналу (0 – підключений до виводу МК, 1–відключений)
SELFPRGEN[3)]						+						Дозвіл самопрограмування (0 – дозволено)
S8515C	+											Вмикає/вимикає режим сумісності з МК AT90S4414/8515 набору Classic (0 – включений, 1 – виключений)
S8535C		+										Вмикає/вимикає режим сумісності з МК AT90S8535 набору Classic (0 – включений, 1 – виключений)
M161C							+					Вмикає/вимикає режим сумісності з МК ATmega161x набору Mega (0 – включений, 1 – виключений)
M103C					+							Вмикає/вимикає режим сумісності з МК ATmega103x набору Mega (0 – включений, 1 – виключений)

3.1.3 Ідентифікатор

Усі МК фірми Atmel мають три 8-бітові комірки, вміст яких дозволяє ідентифікувати пристрій.

У першій клітинці міститься код виробника, у другій – код обсягу FLASH-пам’яті, а в третій – код пристрою. Як і конфігураційні комірки, комірки ідентифікатора розташовані в окремому адресному просторі, доступ до якого можливий тільки в режимі програмування.

Однак, на відміну від конфігураційних комірок, комірки ідентифікатора доступні тільки для читання.

Вміст комірок ідентифікатора для всіх мікроконтролерів набору Mega приведено в табл. 3.7.

Як видно з табл. 3.7, значення коду пристрою (комірка $ 02) у різних моделей може збігатися. Тому пристрій варто ідентифікувати тільки за сукупністю значень комірок $ 01 і $ 02, так як саме ця пара чисел є унікальною для кожного мікроконтролера.

Таблиця 3.7 – Комірки ідентифікатора мікроконтролерів набору Mega

Модель	Комірка ідентифікатора		
	$ 00	$ 01	$ 02
ATmega 8515x	$ 1E	$ 93	$ 06
ATmega 8535x	$ IE	$ 93	$ 08
ATmega 8x	$ IE	$ 93	$ 07
ATmega 16x	$ IE	$ 94	$ 03
ATmega 32x	$ 1E	$ 95	$ 02
ATmega 64x	$ 1E	$ 96	$ 02
ATmega 128x	$ 1E	$ 97	$ 02
ATmega 48x	$ 1E	$ 92	$ 05
ATmega 88x	$ 1E	$ 93	$ 0A
ATmega 168x	$ 1E	$ 94	$ 06
ATmega l62x	$ 1E	$ 94	$ 04
ATmega 164x	$ 1E	$ 94	–
ATmega 324x	$ 1E	$ 95	–
ATmega 644x	$ 1E	$ 96	–
ATmega 165x	$ 1E	$ 94	$ 07
ATmega 325x	$ 1E	$ 95	$ 05
ATmega 3250x	$ 1E	$ 95	$ 05
ATmega 645x	$ 1E	$ 96	$ 06
ATmega 6450x	$ 1E	$ 96	$ 06
ATmega 640x	51E	$ 96	$ 08
ATmega l280x	$ 1E	$ 97	$ 03
ATmega l281x	$ 1E	$ 97	$ 04
ATmega 2560x	$ 1E	$ 98	$ 01

3.1.4 Комірки калібрування

У калібрувальні комірки при виготовленні мікроконтролера заносяться константи калібрування, які призначені для підстроювання на номінальну частоту внутрішнього RC-генератора. Кількість цих комірок залежить від того, на скількох частотах може працювати внутрішній RC-генератор. У моделях ATmega8515x/8535x і ATmega8x/16x/32x/64x/128x є чотири 8-бітних комірки, а в інших моделях – одна комірка. Знаходяться вони в старших байтах адресного простору комірок ідентифікатора (одна комірка – за адресою $ 000, чотири комірки – за адресами $ 000, $ 001, $ 002 і $ 003 для частот 1, 2, 4 і 8 МГц відповідно).

Завантаження калібрувальної константи в регістр OSCCAL здійснюється апаратно при знаходженні мікроконтролера в стані скидання. Однак у моделях ATmega8515x/8535x і ATmega8x/16x/32x/64x/128x генератор автоматично калібрується тільки на частоту 1 МГц. Тому при використанні іншої частоти RC-генератора його калібрування необхідно здійснювати вручну. Для цього вибору програматор під час програмування має прочитати вміст комірки калібрування і занести його за будь-якою адресою FLASH-пам'яті програм. А програма має після старту зчитати це значення з пам'яті програм і занести його в регістр OSCCAL.

3.1.5 Організація пам'яті програм і даних

У мікрококнтролерах набору Classic, Mega використовується сторінкова організація пам'яті програм. При програмуванні весь обсяг FLASH-пам'яті розбивається на 16-бітові сторінки, розмір яких, а також їхня кількість залежать від обсягу пам'яті програм мікроконтролера (табл. 3.8).

Таблиця 3.8 – Параметри сторінкової організації пам'яті програм

Параметр	Обсяг пам'яті програм [байт]						
	4К	8К	16К	32К	64К	128К	256К
Розмір сторінки, слів	32	32	64	64	128	128	128
Кількість сторінок	64	128	128	256	256	512	1024

При програмуванні пам'яті програм мікроконтролерів набору Classic, Mega дані спочатку завантажуються до буферу сторінки і тільки потім заносяться безпосередньо в пам'ять програм. Прошивка всіх комірок сторінки при цьому здійснюється одночасно.

Аналогічно організована і EEPROM-пам'ять. Розмір 8-бітних сторінок EEPROM-пам'яті, а також їхня кількість для моделей Mega мікроконтролерів наведені в табл. 3.9.

Таблиця 3.9 – Параметри сторінкової організації EEPROM-пам'яті

Параметр	Обсяг EEPROM-пам'яті [байт]				
	256	512	1К	2К	4К
Розмір сторінки, байт	4	4	4	8	8
Кількість сторінок	64	128	256	256	512

Однак варто зазначити, що в багатьох моделях сторінкова організація EEPROM-пам'яті використовується тільки при програмуванні в паралельному режимі, а програмування по послідовному каналу здійснюється побайтно.

3.2 Програмування по послідовному каналу

У режимі програмування по послідовному каналу програмування пам'яті програм і даних здійснюється по послідовному інтерфейсу SPI. Як правило, цей режим використовується для програмування (перепрограмування) мікроконтролера безпосередньо в пристрої.

Схема ввімкнення мікросхем у режимі програмування по послідовному каналу наведено на рис. 3.2. На цьому ж рисунку наведено два варіанти розведення колодки для підключення програматора, рекомендовані компанією Atmel. Як видно з рис. 3.2, для обміну даними між програматором і

пристроєм використовуються три лінії: SCK (тактовий сигнал), MOSI (вхід даних) і MISO (вихід даних). Відповідність між лініями інтерфейсу і контактами портів введення/виведення мікроконтролерів набору Mega зображено в табл. 3.10. Параметри сигналів при програмуванні по послідовному каналу зображено в табл. 3.11.

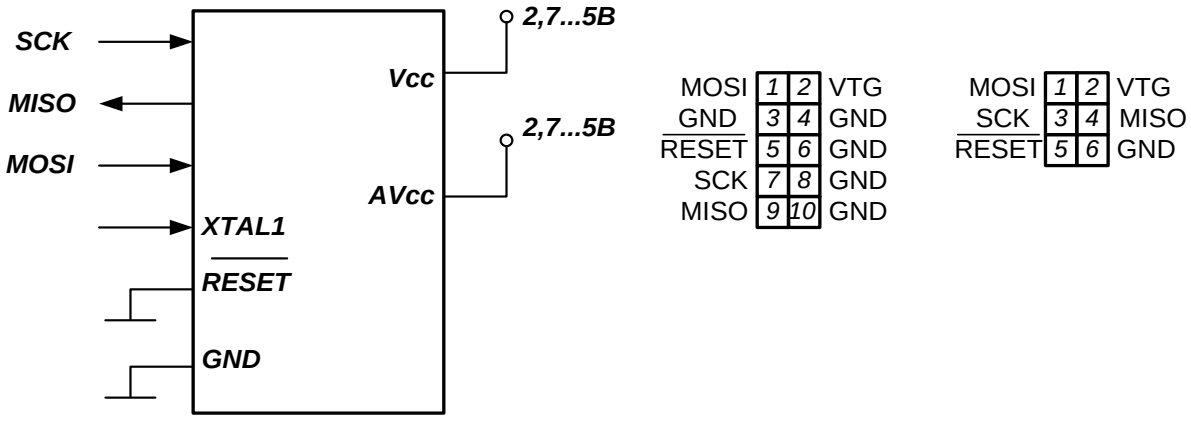


Рисунок 3.2 – Увімкнення мікроконтролерів у режимі програмування по послідовному каналу

Таблиця 3.10 – Виводи мікроконтролерів Mega, які використовуються при програмуванні по послідовному каналу

Назва лінії інтерфейсу	ATmega8515x/8535x	ATmega8x	ATmega16x/32x	ATYnega64x/128x	ATmega48x/88x/168x	ATmegal62x	ATmega164x/324x/644x	ATmega165x/325x/3250x ATmega645x/6450x	ATmega640x/1280x/2560x	ATmega1281x/2561x	Призначення виводів
SCK	PB7	PB5	PB7	PB1	PB5	PB7	PB7	PB1	PB1	PB1	Вхід тактового сигналу
MISO (PD0)	PB6	PB4	PB6	PE1	PB4	PB6	PB6	PB3	PE1	PB3	Вихід даних
MOSI (PD1)	PB5	PB3	PB5	PE0	PB3	PB5	PB5	PB2	PE0	PB2	Вхід даних

Таблиця 3.11– Параметри сигналів при програмуванні по послідовному каналу

Позначення	Параметр			min	typ	max
$1/_t$ CLCL	частота тактового сигналу, МГц	V_{CC} = = 1.8...5.5 В	ATmega 162V	0	–	1
			ATmega 48V/88V/168V, ATmega 164V/324V/644V, ATmega 165V/325V/3250V/645V/6450V, ATmega 640V/1280V/1281V/2560V/2561V	0	–	4
		V_{CC} = = 2.7...5.5В	ATmega 8515L/8535L ATmega 8L/16L/32L/64L/128L, ATmega 162L ATmega165x/325x/3250x/645x/6450x, ATmega640x/l280x/г/2560x/2561x	0	–	8
			ATmega48x/88x/168x, ATmega164x/324x/644x	0	–	10
		V_{CC} = = 4.5...5.5 В	ATmega 8515/8535, ATmega 8/1б/32/64/128, ATmega l62, ATmega 165/325/3250/645/6450, ATmega 640/1280/1281/2560/2561	0	–	16
			ATmega 48/88/168, ATmega 164/324/644	0	–	20
t_{SHSL}	Тривалість позитивного імпульсу сигналу SCK, нс			$2t_{CLCL}$ [1)]	–	–
t_{SLSH}	Тривалість негативного імпульсу сигналу SCK, нс			$2t_{CLCL}$ [1)]	–	–
t_{OVSH}	Затримка наростаючого фронту сигналу SCK щодо встановлення сигналу MOSI, нс			t_{CLCL}	–	–
t_{SHOX}	Час утримання сигналу MOSI щодо наростаючого фронту сигналу SCK, нс			$2t_{CLCL}$ [1)]	–	–
t_{SLIV}	Затримка встановлення сигналу MISO щодо спадаючого фронту сигналу SCK, нс			10	16	32
t_{WD_ERASE}	Період очікування після завантаження команди «Знищення кристала», мс			9	–	–
t_{WD_FLASH}	Період очікування після запису FLASH-пам'яті, мс			4.5	–	–
t_{WD_EEPROM}	Період очікування після запису EEPROM-пам'яті, мс			9	–	–
t_{WD_FUSE}	Період очікування після запису конфігураційних комірок або комірок захисту, мс			4.5	–	–

Як і в робочому режимі, при програмуванні по послідовному каналу мікроконтролеру потрібно джерело тактового сигналу. Як джерела тактового сигналу може використовуватися будь-яке таке джерело. При цьому має виконуватися така умова: тривалість імпульсів як «0», так і «1» рівня сигналу SCK має бути більше 2 (при $f_{ск}$ <12 МГц) або 3 (при $f_{ск}$ > 12 МГц) періодів тактового сигналу мікроконтролера.

Програмування здійснюється шляхом посилки 4-байтних команд на вивід MOSI мікроконтролера. Результат виконання команд читання знімається з виводу MISO мікроконтролера. Передача команд і видача результатів їхнього виконання здійснюються від старшого біта до молодшого. Формат усіх команд, які використовуються для програмування мікроконтролерів набору в цьому режимі, наведено в табл. 3.12.

Таблиця 3.12 – Команди режиму програмування по послідовному каналу

Команда	Формат команди				Опис команди
	1-й байт	2-й байт	3-й байт	4-й байт	
Дозвіл програмування	$ AC	$ 53	$ 00	$ 00	Дозволяє програмування МК, допоки на виводі $\overline{\text{RESET}}$ присутня напруга НИЗЬКОГО рівня
Знищення кристала	$ AC	$ 80	$ 00	$ 00	Очищення вмісту FLASH- і EEPROM-пам'яті
Читання прапорця готовності RDY/$\overline{\text{BSY}}$ [1)]	$ F0	$ 00	$ 00	дані	Якщо 0-й біт прийнятого байта скинутий в 0, то попередня операція програмування не завершена. Перед посилкою нової команди програмування необхідно дочекатися встановлення цього біта в 1
Завантаження додаткового байта адреси	$ 40	$ 00	адреса	$ 00	Завантаження старшого байта адреси слова даних в моделях ATmega2560x/2561x (використовується тільки молодший біт)
Команди завантаження					
Завантаження сторінки FLASH-пам'яті, старший байт	$ 48	$ 00	адреса	старший байт	Запис молодшого/старшого байта в буфер сторінки пам'яті програм. Адреса є адресою слова в межах сторінки. Молодший байт має завантажуватися першим
Завантаження сторінки FLASH-пам'яті, молодший байт	$ 40	$ 00	адреса	молодший байт	
Завантаження сторінки EEPROM-пам'яті (сторінковий доступ) [1)]	$ C1	$ 00	адреса	дані	Запис байта в буфер сторінки EEPROM-пам'яті Адреса є адресою байта в межах сторінки

Продовження таблиці 3.12

Команда	Формат команди				Опис команди
	1-й байт	2-й байт	3-й байт	4-й байт	
команди читання					
читання FLASH-пам'яті, старший байт	$28	адреса (ст. байт)	адреса (мол. байт)	дані	Читання старшого/молодшого байта пам'яті програм. Адреса є адресою слова в межах адресного простору пам'яті програм. У моделях ATmega 2560x/2561x попередньо повинен бути завантажений старший біт адреси (команда завантаження додаткового байта адреси)
читання FLASH-пам'яті, молодший байт	$20	адреса (ст. байт)	адреса (мол. байт)	дані	
читання EEPROM-пам'яті	$A0	адреса (ст. байт)	адреса (мол. байт)	дані	Читання вмісту комірки EEPROM-пам'яті. Адреса є адресою байта в межах адресного простору EEPROM
Читання комірок захисту	$58	$00	$00	дані	Читання байта, що містить значення комірок захисту
читання ідентифікатора	$30	$00	адреса	дані	Читання байта ідентифікатора з заданим адресою (тільки в режимах захисту №1 і №2, див. табл. 3.2)
Читання молодшого конфігураційного байта	$50	$00	$00	дані	Читання конфігураційних комірок. Скинутий біт означає, що відповідна конфігураційна комірка запрограмована
Читання старшого конфігураційного байта	$58	$08	$00	дані	
Читання додаткового конфігураційного байта	$50	$08	$00	дані	Читання конфігураційних комірок. Скинутий біт означає, що відповідна конфігураційна комірка запрограмована
Читання калібрувальної комірки	$38	$00	адреса	дані	Читання калібрувальної константи за заданою адресою
Команди запису					
Запис сторінки FLASH-пам'яті	$4C	адреса (ст. байт)	адреса (мол. байт)	$ 00	Запис сторінки пам'яті програм за заданою адресою. Адреса являє собою значення старших бітів слів завантаженої сторінки. У моделях ATmega25б0x/2561x попередньо повинен бути завантажений старший біт адреси (команда завантаження додаткового байта адреси)

Продовження таблиці 3.12

Команда	Формат команди				Опис команди
	1-й байт	2-й байт	3-й байт	4-й байт	
запис EEPROM-пам'яті	$C0	адреса (ст. байт)	адреса (мол. байт)	дані	Запис значення в клітинку EEPROM-пам'яті по заданій адресі
Запис сторінки EEPROM-пам'яті (сторінковий доступ)	$C2	адреса (ст. байт)	адреса (мол. байт)	$ 00	Запис сторінки EEPROM-пам'яті за заданою адресою
Запис комірок захисту	$AC	$ E0	$ 00	дані	Запис комірок захисту. Для програмування комірки відповідний біт байта даних має бути скинутий
Запис молодшого конфігураційного байта	$ AC	$ A0	$ 00	дані	Запис конфігураційних комірок. Для програмування комірки відповідний біт байта даних повинен бути скинутий
Запис старшого конфігураційного байта	$ AC	$ A8	$ 00	дані	
Запис додаткового конфігураційного байта	$ AC	$ A4	$ 00	дані	

3.2.1 Перемикання в режим програмування

Для переведення мікроконтролера в режим програмування по послідовному каналу необхідно виконати такі дії:

1. Подати на мікроконтролер напругу живлення, при цьому на виводах SCK і $\overline{\text{RESET}}$ має бути присутньою напруга «0» рівня.

2. Зачекати не менше 20 мс.

3. Послати на вивід MOSI команду «Дозвіл програмування».

Для контролю проходження команди при посиланні 3-го байта повертається значення 2-го байта ($53). Якщо значення, що повертається відмінно від зазначеного, необхідно подати на вивід $\overline{\text{RESET}}$ позитивний імпульс і знову послати команду «Дозвіл програмування». Причому незалежно від значення, що повертається необхідно передавати всі 4 байти команди. Відсутність повернення числа $53 після кількох спроб вказує на відсутність зв'язку між програматором і мікросхемою або на несправність мікросхеми.

Після завершення програмування на вивід $\overline{\text{RESET}}$ можна подати напругу рівня «1» для переведення мікроконтролера в робочий режим або вимкнути його. В останньому випадку необхідно виконати таку послідовність дій:

1) Подати на вивід XTAL1 напругу рівня «0», якщо тактування мікроконтролера здійснюється від зовнішньої схеми.

2) Подати на вивід $\overline{\text{RESET}}$ напругу рівня «1».

3) Відключити напругу живлення від мікроконтролера.

3.2.2 Управління процесом програмування FLASH-пам'яті

Програмування пам'яті програм мікроконтролерів набору Mega здійснюється посторінково. Спочатку вміст сторінки побайтно заноситься в буфер по командах «Завантаження сторінки FLASH-пам'яті». У кожній команді передаються молодші біти адреси змінною комірки (положення комірки всередині сторінки) і записується значення. Вміст кожної комірки має завантажуватися в такій послідовності: спочатку молодший байт, потім старший.

Фактичне програмування сторінки FLASH-пам'яті здійснюється після завантаження буфера сторінки за командою «Запис сторінки FLASH-пам'яті». У команді передаються старші біти адреси комірок (номер сторінки).

Подальше програмування пам'яті можна буде виконувати тільки після завершення запису сторінки. Визначити момент закінчення запису можна трьома способами. Перший і найбільш простий спосіб – витримувати між посиланням команд паузу тривалістю не менше T_{WD_FLASH} (табл. 3.11). Другий спосіб полягає в контролюванні вмісту будь-якої із записуваних комірок після посилання команди запису: до завершення запису комірки при її читанні повертається значення $ FF? а після завершення – записане значення.Третій спосіб – аналіз стану прапорця готовності RDY/$\overline{BSY}$ за допомогою відповідної команди і відновлення процесу програмування після встановлення прапорця в «1».

3.2.3 Управління процесом програмування EEPROM-пам'яті

У всіх старих моделях програмування EEPROM-пам'яті здійснюється звичайним способом – побайтно. У нових моделях з'явився альтернативний спосіб запису EEPROM-пам'яті – посторінковий. Вміст сторінки побайтно заноситься в буфер по командах «Завантаження сторінки EEPROM-пам'яті», а потім здійснюється фактичне програмування сторінки EEPROM-пам'яті за командою «Запис сторінки EEPROM-пам'яті». Значення адрес, що передаються в цих командах, визначаються так само, як і при програмуванні EEPROM-пам'яті. Для визначення моменту закінчення запису можна використовувати будь-який з описаних вище способів.

3.3 Паралельне програмування

У режимі паралельного програмування, як випливає з його назви, від програматора до мікроконтролеру передаються одночасно всі біти коду команди або байта даних. Цей режим задіє велику число виводів мікроконтролера і, до того ж, вимагає використання додаткового джерела підвищеної напруги (12 В). Тому програмування в паралельному режимі здійснюється спеціалізованими програматорами. Основне застосування цього режиму – «прошивка» мікроконтролерів перед установкою їх на плату в умовах масового виробництва.

Схема ввімкнення мікросхем у режимі паралельного програмування зображено на рис. 3.3. Призначення сигналів, присутніх на виводах мікроконтролера в цьому режимі, наведено в табл. 3.13–3.15.

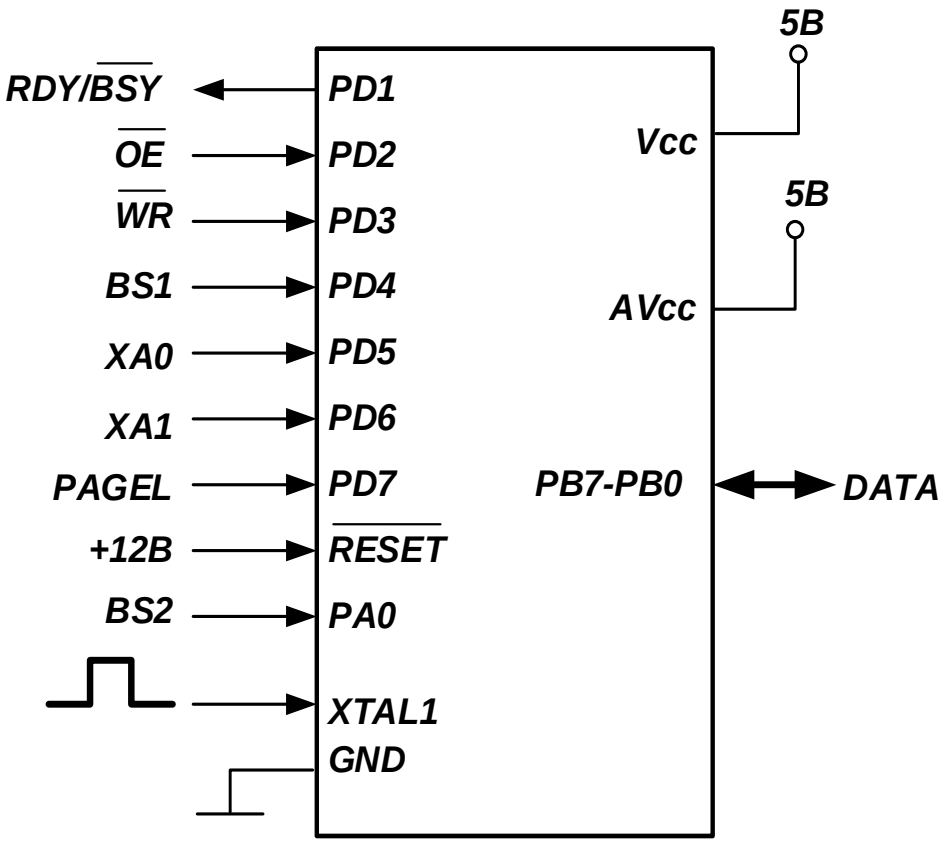


Рисунок 3.3 – Увімкнення мікроконтролерів
у режимі паралельного програмування

Таблиця 3.13 – Позначення та функції виводів, що використовуються при програмуванні в паралельному режимі

Сигнал	Вивід	Вхід/вихід	Призначення
RDY/$\overline{\text{BSY}}$	PD1	вихід	Стан пристрою: 0 – зайнято (виконується попередня команда); 1 – готово до прийому наступної команди
OE	PD2	Вхід	Управління режимом роботи шини даних PB7 ... PB0: 0 – вихід, 1 – вхід
WR	PD3	Вхід	Сигнал запису (активний рівень – лог. 0)
BS1	PD4	Вхід	Вибір байта (табл. 3.14)
XA0	PD5	Вхід	Визначають дію, яка виконується по позитивному імпульсу на виводі XTAL1 (табл. 3.15)
XA1	PD6	Вхід	
PAGEL	PD7	Вхід	Сигнал завантаження сторінки пам'яті
BS2	PA0	Вхід	Вибір байта (табл. 3.14)
DATA	PB7...PB0	Вхід/вихід	Двонаправлена шина даних

Таблиця 3.14 – Функції сигналів BS1 і BS2

BS2	BS1	Адреси FLASH/EEPROM	Завантаження/читання FLASH	Програмування конфігураційних комірок	Читання конфігурацій-них комірок і комірок захисту
0	0	молодший байт	молодший байт	молодший байт	молодший байт
0	1	старший байт	старший байт	старший байт	байт комірок захисту
1	0	додатковий байт	зарезервовано	додатковий байт	додатковий байт
1	1	зарезервовано	зарезервовано	зарезервовано	старший байт

Таблиця 3.15 – Функції сигналів XA0 і XA1

XA1	XA0	Дія, що виконується по тактовому імпульсу
0	0	Завантаження адреси комірки пам'яті (молодшого або старшого байта залежно від рівня сигналу BS1)
0	1	завантаження даних (молодшого або старшого байта залежно від рівня сигналу BS1)
1	0	Завантаження команди
1	1	Немає дії, режим очікування

Таблиця 3.16 – Параметри сигналів (програмування в паралельному режимі)

Позна-чення	Параметр	Min	Max	Одиниці виміру
V_{pp}	Напруга дозволу програмування	11,5	12,5	В
I_{pp}	Струм, споживаний від джерела +12 В $_{(V\ pp)}$	–	250	мкА
t_{DVXH}	Затримка сигналу XTAL1 відносно моменту встановлення сигналів управління і даних	67	–	нс
t_{XLXH}	Інтервал між імпульсами сигналу XTAL1	200	–	нс
t_{XHXL}	Тривалість імпульсів сигналу XTAL1	150	–	нс
t_{XLDX}	Час утримання сигналів управління і даних щодо заднього фронту сигналу XTAL1	67	–	нс
t_{XLWL}	Затримка сигналу відносно заднього фронту сигналу XTAL1	0	–	нс
t_{XLPH}	Затримка сигналу PAGEL відносно заднього фронту сигналу XTAL1	0	–	нс
t_{PLXH}	Затримка сигналу XTAL1 відносно заднього фронту сигналу PAGEL	150	–	нс
t_{BVPH}	Затримка сигналу PAGEL відносно сигналу BS1	67	–	нс
t_{PHPL}	Тривалість імпульсу сигналу PAGEL	150	–	нс
t_{PLBX}	Час утримання сигналу BS1 щодо заднього фронту сигналу PAGEL	67	–	нс
t_{WHBX}	Час утримання сигналів BS1/BS2 щодо заднього фронту сигналу $\overline{WR}$	67	–	нс
t_{PLWL}	Затримка сигналу щодо заднього фронту сигналу PAGEL	67	–	нс
t_{BVWL}	Затримка сигналу щодо моменту встановлення сигналу BS1	67	–	нс
t_{RHBX}	Час утримання сигналу BS1 щодо заднього фронту сигналу RDY/$\overline{BSY}$	67	–	нс

Продовження таблиці 3.16

Позначення	Параметр	Min	Max	Одиниці виміру
t_{WLWH}	Тривалість імпульсу сигналу	150	–	Нс
t_{WLRL}	Затримка появи сигналу RDY/$\overline{BSY}$щодо переднього фронту сигналу $\overline{WR}$	0	1	мкс
t_{WLRH}	Затримка зняття сигналу RDY/$\overline{BSY}$ щодо переднього фронту сигналу	3.4	4.5	мс
t_{WLRH_CE}	Затримка зняття сигналу RDY/$\overline{BSY}$ щодо переднього фронту сигналу для команди «Знищення кристала» $\overline{WR}$	7.5	9	7 мс
t_{XLOL}	Затримка сигналу OE щодо заднього фронту сигналу XTAL1	0	–	нс
t_{BHDV}	Час встановлення сигналів даних щодо наростаючого фронту сигналу BS 1	0	250	нс
t_{OLDV}	Час встановлення сигналів даних щодо переднього фронту сигналу $\overline{OE}$	–	250	нс
t_{OHDZ}	Затримка перемикання шини даних в третій стан щодо заднього фронту сигналу $\overline{OE}$	–	250	нс

У загальних рисах процес програмування в цьому режимі складається з багаторазового виконання таких операцій:

– завантаження команди;

– завантаження адреси;

– завантаження даних;

– виконання команди.

Послідовність подачі сигналів на виводи мікроконтролера при виконанні різних базових операцій наведено в табл. 3.17.

Таблиця 3.17 – Базові операції програмування в паралельному режимі

№	Назва операції	Дії
1	Завантаження команди	1. Установити виводи XA1, XA0 у стан 10 (завантаження команди) 2. Подати на вивід BS1 напругу логічний 0 3. Виставити на шину DATA код команди (табл. 3.18) 4. Подати на вивід XTAL1 позитивний імпульс
2	Завантаження адреси	1. Установити висновки XA1, XA0 в стан 00 (завантаження адреси) 2. Установити виводи BS2: BS1 у стан 00 (завантаження молодшого байта), 01 (завантаження старшого байта) або 10 (завантаження додаткового байта) 3. Виставити на шину DATA байт адреси 4. Подати на вивід XTAL1 позитивний імпульс
3	Завантаження даних	1. Установити виводи XA1, XA0 у стан 01 (завантаження даних) 2. Подати на вивід BS1 напругу логічний 0 (завантаження молодшого байта) або логічна 1(завантаження старшого байта) 3. Виставити на шину DATA вміст байта даних 4. Подати на вивід XTAL1 позитивний імпульс

Продовження таблиці 3.17

№	Назва операції	Дії
4	Запис даних у буфер сторінки	1. Подати на вивід BS1 напругу логічна 1 2. Подать на вивід PAGEL позитивний імпульс
5	Запис байта конфігурації	1. Установити виводи BS2: BS1 у стан 00 (запис молодшого байта), 01 (запис старшого байта) або 10 (запис додаткового байта) 2. Подать на вивід $\overline{WR}$ негативний імпульс; при цьому на виводі RDY/$\overline{BSY}$ з'являється сигнал рівня «0» 3. Очікувати появи на виводі RDY/$\overline{BSY}$ сигналу рівня «1»
6	Запис сторінки	1. Установити виводи BS2: BS1 у стан 00 2. Подать на вивід $\overline{WR}$ негативний імпульс; при цьому на виводі RDY/$\overline{BSY}$ з'являється сигнал рівня «0» 3. Очікувати появи на виводі RDY/$\overline{BSY}$ сигналу рівня «1»

У цьому режимі використовується 9 команд, коди яких наведені в табл. 3.18.

Таблиця 3.18 – Команди програмування в паралельному режимі

Код команди	Опис
1000 0000	Знищення кристала
0100 0000	Запис конфігураційних комірок
0010 0000	Запис комірок захисту
0001 0000	Запис FLASH-пам'яті
0001 0001	Запис EEPROM-пам'яті
0000 1000	Читання ідентифікатора
0000 0100	Читання конфігураційних комірок і комірок захисту
0000 0010	Читання FLASH-пам'яті
0000 0011	Читання EEPROM-пам'яті

3.3.1 Перемикання в режим паралельного програмування

Першою операцією при програмуванні мікроконтролера є його переведення в режим програмування. Для переведення мікроконтролера в режим програмування необхідно виконати таки дії:

1. Подати на мікроконтролер напругу живлення.

2. Подати на вивід $\overline{RESET}$ напругу рівня «0» і сформувати не менше трьох імпульсів на виводі XTAL1.

3. Подати на виводи PAGEL, XA1, XA0, BS1 напругу рівня «0» на час не менше 100 нс.

4. Подати напругу 11,5...12,5 В на вивід $\overline{RESET}$ і утримувати напругу рівня «0» на виводах PAGEL, XA1, XA0, BS1 протягом, як мінімум, 10 мкс. Будь-яка активність на зазначених виводах протягом цього часу призведе до того, що мікроконтролер не перейде в режим програмування.

5. Зачекати не менше 300 мкс перед посиланням наступної команди.

Окремо варто сказати про мікроконтролери ATmega8515x/8535x, ATmega8x і ATmega48x/88x/168x, так як в цих моделях вивід скидання може бути задіяний під лінію введення/виведення (якщо конфігураційна комірка RSTDISBL запрограмована). У цьому випадку перед виконанням дій, описаних вище, необхідно:

1. Подати на виводи PAGEL, XA1, XA0, BS1 напругу рівня «0».
2. Подати на мікроконтролер напругу живлення (VCC), а на вивід $\overline{\text{RESET}}$ – напругу 11,5...12,5 В (V_{pp}).
3. Зачекати не менше 100 нс.
4. Перепрограмувати комірку RSTDISBL (якщо встановлений захист, то спочатку необхідно виконати знищення кристала).
5. Вийти з режиму програмування (вимкнути живлення або подати на вивід $\overline{\text{RESET}}$ напругу рівня «1»).
6. Перевести мікроконтролер у режим програмування, як було описано раніше.

3.3.2 Знищення кристала

Команда «Знищення кристала» має виконуватися перед кожним перепрограмуванням мікроконтролера. Така команда повністю знищує вміст FLASH- і EEPROM-пам'яті, а потім скидає комірки захисту (записує в них логічну 1). Однак на стан конфігураційних комірок дана команда не впливає. До того ж, у низці моделей мікроконтролерів набору Mega можна запобігти цьому EEPROM-пам'яті шляхом програмування конфігураційної комірки EESAVE.

Для виконання команди «Знищення кристала» необхідно виконати такі дії:

1. Завантажити команду «Знищення кристала» (код 1000 0000).
2. Подати на вивід $\overline{\text{WR}}$ негативний імпульс; при цьому на вивід RDY/$\overline{\text{BSY}}$ з'являється сигнал рівня «0».
3. Чекати появи на виводі RDY/$\overline{\text{BSY}}$ сигналу рівня «1».

3.3.3 Програмування FLASH-пам'яті

Запис FLASH-пам'яті проводиться в такій послідовності (реалізація кожного етапу наведена в табл. 3.17):

1. Завантажити команду «Запис FLASH-пам'яті» (код 0001 0000).
2. Завантажити молодший байт адреси (положення комірки всередині сторінки).
3. Завантажити молодший байт даних.
4. Завантажити старший байт даних.
5. Запам'ятати дані в буфері.
6. Повторити п. 2...5 до повного заповнення буфера сторінки.
7. Завантажити старший байт адреси (номер сторінки).
8. Завантажити додатковий байт адреси (у моделях ATmega2560x/2561x).
9. Зберегти сторінку.
10. Повторити п. 2...9 для запису інших сторінок пам'яті програм.

11. Завершити програмування, завантаживши команду «Немає операції» (код 0000 0000).

Для читання FLASH-пам'яті необхідно виконати такі дії (реалізація кожного етапу наведена в табл. 3.17):

1. Завантажити команду «Читання FLASH-пам'яті» (код 0000 0010).
2. Завантажити додатковий байт адреси (у моделях ATmega 2560/2561x).
3. Завантажити старший байт адреси.
4. Завантажити молодший байт адреси.
5. Установити $\overline{OE}$ і BS1 в 0, після цього з шини даних DATA можна буде зчитати значення молодшого байта вмісту комірки пам'яті.
6. Установити BS1 в 1, після цього з шини даних DATA можна буде зчитати значення старшого байта вмісту комірки пам'яті.
7. Установити $\overline{OE}$ в 1.

3.3.4 Програмування EEPROM-пам'яті

Запис EEPROM-пам'яті проводиться у такій послідовності (реалізація кожного етапу наведена в табл. 3.17):

1. Завантажити команду «Запис EEPROM-пам'яті» (код 0001 0001).
2. Завантажити старший байт адреси.
3. Завантажити молодший байт адреси.
4. Завантажити байт даних.
5. Запам'ятати дані в буфері.
6. Повторити п. 3...5 до повного заповнення буфера.
7. Зберегти сторінку.

Для читання вмісту EEPROM-пам'яті необхідно виконати такі дії (реалізація кожного етапу наведена в табл. 3.17):

1. Завантажити команду «Читання EEPROM-пам'яті» (код 0000 0011).
2. Завантажити старший байт адреси.
3. Завантажити молодший байт адреси.
4. Встановити $\overline{OE}$ і BS1 в 0, після цього з шини даних DATA можна буде зчитати вміст комірки пам'яті.
5. Установити $\overline{OE}$ в 1.

3.3.5 Програмування конфігураційних комірок

Програмування байтів конфігурації мікроконтролерів набору Mega здійснюється так.

Молодший конфігураційний байт:

1. Завантажити команду «Запис конфігураційних комірок» (код 0100 0000).
2. Завантажити молодший байт даних. Якщо біт скинутий в 0, виконується програмування відповідної комірки, якщо встановлений в 1 – її скидання.
3. Записати молодший байт конфігурації.

Старший конфігураційний байт:

1. Завантажити команду «Запис конфігураційних комірок» (код 0100 0000).

2. Завантажити молодший байт даних. Якщо біт скинутий в 0, виконується програмування відповідної комірки, якщо встановлений в 1 – її скидання.

3. Записати старший байт конфігурації.

Додатковий конфігураційний байт записується аналогічно.

3.3.6 Програмування, читання комірок захисту та конфігурації

Програмування комірок захисту виконується аналогічно програмуванню конфігураційних комірок (реалізація кожного етапу наведена в табл. 3.17):

1. Завантажити команду «Запис комірок захисту» (код 0010 0000).

2. Завантажити молодший байт даних. Для програмування комірки відповідний біт має бути скинутий в 0. Невикористані біти мають бути завжди встановлені в 1.

3. Записати молодший байт конфігурації.

Читання комірок конфігурації та захисту виконується у такій послідовності (реалізація кожного етапу наведена в табл. 3.17):

1. Завантажити команду «Читання конфігураційних комірок та комірок захисту» (код 0000 0100).

2. Установити $\overline{OE}$, BS1 і BS2 в «0», після цього з шини даних DATA можна буде зчитати значення молодшого конфігураційного байта.

3. Установити $\overline{OE}$ в 0, а BS1 і BS2 – у «1». Після цього з шини даних DATA можна буде зчитати значення старшого конфігураційного байта.

4. Установити $\overline{OE}$ в 0, BS1 – у «0», а BS2 – у «1». Після цього з шини даних DATA можна буде зчитати значення додаткового байта конфігурації.

5. Установити $\overline{OE}$ в 0, BS1 – у «1», а BS2 – у «0». Після цього з шини даних DATA можна буде зчитати значення байту захисту.

6. Встановити $\overline{OE}$ в «1».

3.3.7 Читання комірок ідентифікатора і комірок калібрування

Читання комірок ідентифікатора здійснюється у такій послідовності:

1. Завантажити команду «Читання комірок ідентифікатора» (код 0000 1000).

2. Завантажити молодший байт адреси ($00...$02).

3. Установити $\overline{OE}$ і BS1 в «0», після цього з шини даних DATA можна буде зчитати вміст вибраної комірки ідентифікатора.

4. Установити $\overline{OE}$ в «1».

5. Читання калібрувальних констант здійснюється аналогічно і за допомогою тієї ж команди:

6. Завантажити команду «Читання комірок ідентифікатора» (код 0000 1000).

7. Завантажити молодший байт адреси ($00 ... $03).

8. Установити $\overline{OE}$ в «0», а BS1 – в «1», після цього з шини даних DATA можна буде зчитати значення обраної константи калібрування.

9. Установити $\overline{OE}$ в «1».

3.4. Програмування по інтерфейсу JTAG

3.4.1 Використання інтерфейсу JTAG для програмування кристала

Інтерфейс JTAG був розроблений групою провідних фахівців з проблем тестування електронних компонентів (Joint Test Action Group). Вбудований у більшість мікроконтролерів набору Mega, інтерфейс JTAG може бути використаний для:

– тестування друкованих плат;

– конфігурації (програмування) кристала;

– внутрішньосхемного налагодження.

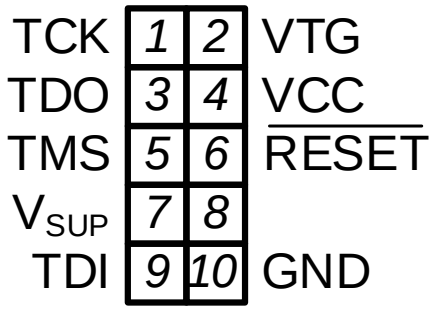


Рисунок 3.4 – Розводка роз'єму JTAG

Доступ до модуля JTAG здійснюється через чотири виводи мікроконтролера, що складають так званий «порт тестового доступу» (Test Access Port – TAP): TMS, TCK, TDI і TDO. Відповідність цих виводів контактам портів введення/виведення мікроконтролера, а також їхні функції наведені в табл. 3.19. Стандартний роз'єм для підключення пристроїв наведений на рис. 3.4.

Роботою модуля JTAG керує TAP-контролер, який є скінченним автоматом зі 16 станами. Перехід між станами здійснюється за наростаючим фронтом сигналу TCK відповідно до сигналу, що присутній на виводі TMS. Після ввімкнення живлення контролер знаходиться в стані Test-Logic-Reset.

Дозвіл/заборона інтерфейсу JTAG здійснюється за допомогою конфігураційної комірки JTAGEN. Якщо вона не запрограмована (1), то виводи TAP працюють як звичайні контакти портів введення/виведення, а TAP-контролер знаходиться в стані скидання. Для ввімкнення інтерфейсу комірка JTAGEN має бути запрограмована (стан за замовчуванням). До того ж, має бути скинутий біт JTD регістра MCUCSR або MCUCR (рис. 3.5). Причому, для зміни стану цього біта нове значення необхідно записати в нього двічі протягом чотирьох тактів.

Таблиця 3.19 – Виводи, які використовуються інтерфейсом JTAG

Назва лінії інтерфейсу	ATmega16x/32x	ATYnega64x/128x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega 325x/3250x	ATmega640x/1280x/2560x	Призначення виводів
TCK	PC2	PF4	PC4	PC2	PF4	PF4	PF4	Вхід тактового сигналу
TMS	PC3	PF5	PC5	PC3	PF5	PF5	PF5	Вхід вибору режиму
TDO	PC4	PF6	PC6	PC4	PF6	PF6	PF6	Вихід даних
TDI	PC5	PF7	PC7	PC5	PF7	PF7	PF7	Вхід даних

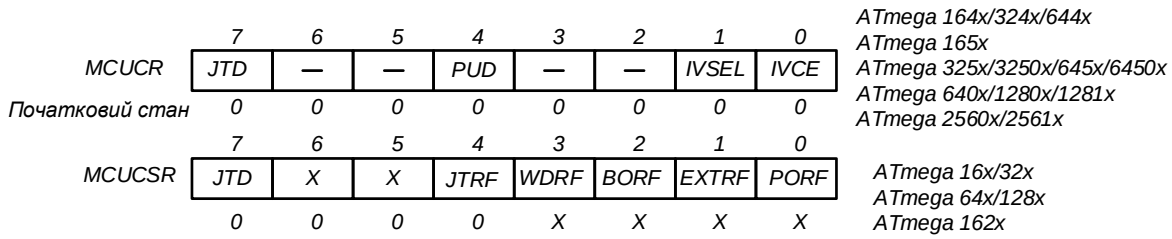


Рисунок 3.5 – Регістри MCUCSR/MCUCR
щодо інтерфейсу JTAG

3.4.2 Команди JTAG для програмування

З 16 команд, підтримуваних інтерфейсом, при програмуванні використовуються тільки п'ять. Описи цих команд наведені нижче.

AVR_RESET (код команди $ 0C). Ця команда призначена для переведення мікроконтролера в стан скидання і відповідно виведення його з цього стану. Як регістру даних обирається 1-бітний регістр скидання (Reset Register). Запис «1» у цей регістр еквівалентна подачі на вивід RESET мікроконтролера напруги рівня «0». У стані скидання мікроконтролер буде перебувати до тих пір, поки в регістр скидання не буде записаний «0».

Активний стан: Shift-DR – здійснюється завантаження регістра скидання.

PROG_ENABLE (код команди $ 04). Ця команда призначена для дозволу програмування кристала через порт JTAG. Як регістр даних обирається 16-бітовий регістр дозволу програмування (Programming Enable Register). При запису в цей регістр числа $A370 (сигнатура дозволу програмування) дозволяється програмування мікроконтролера по інтерфейсу JTAG. При виході з режиму програмування цей регістр має скидатися.

Активні стани:

– Shift-DR – здійснюється завантаження регістра дозволу переривання;

– Update-DR – здійснюється порівняння вмісту регістра з числом $A370 і в разі збігу – переведення мікроконтролера в режим програмування.

PROG_COMMANDS (код команди $ 05). Ця команда призначена для завантаження команд програмування і видачі результатів їхнього виконання (якщо вони є). Як регістр даних обирається 15-бітний регістр команд (Programming Command Register).

Активні стани:

– Capture-DR – результат виконання попередньої команди завантажується в регістр;

– Shift-DR – за наростаючим фронтом сигналу TCK здійснюється видача результату з одночасним завантаженням нової команди;

– Update-DR – завантажена команда подається на блок пам'яті мікроконтролера;

– Run-Test/Idle – генерується один тактовий імпульс, необхідний для виконання команди.

PROG_PAGELOAD (код команди $06). Ця команда призначена для безпосереднього завантаження сторінки пам'яті програм через порт JTAG. Реалізація цієї команди залежить від моделі мікроконтролера. У «старих» моделях (ATmegal6x/32x/64x/128x і ATmegal62x) як регістр даних обирається регістр завантаження віртуальної сторінки (Virtual FLASH Page Load Register), розмір якого дорівнює розміру однієї сторінки пам'яті програм. Власне зсувний регістр є 8-бітовийм, а пересилання по байтах даних в буфер здійснюється автоматично.

Активний стан: Shift-DR – здійснюється побітова завантаження даних і побайтне їхнє пересилання в буфер сторінки FLASH-пам'яті.

В інших моделях як регістр даних обирається 8-бітовий регістр даних FLASH-пам'яті. Активні стани:

– Shift-DR – здійснюється побітне завантаження байта даних;

– Update-DR – вміст регістра даних FLASH-пам'яті копіюється у тимчасовий регістр в буфері сторінки. Чергування старшого і молодшого байтів при кожному проходженні стану Update-DR здійснюється автоматично, починаючи з молодшого байта при завантаженні команди.

PROG_PAGEREAD (код команди $07). Ця команда призначена для зчитування вмісту сторінки пам'яті програм через порт JTAG. Реалізація цієї команди також залежить від моделі мікроконтролера. У «старих» моделях (ATmcgal6x/32x/64x/128x і ATmegal62x) як регістр даних обирається регістр читання віртуальної сторінки (Virtual FLASH Page Read Register), розмір якого на 1 байт більше розміру сторінки пам'яті програм. Побайтне пересилання вмісту сторінки у зсувний регістр здійснюється автоматично.

Активні стани: Shift-DR – здійснюється побайтне пересилання вмісту сторінки FLASH-пам'яті у зсувний регістр і побітова його видача на вивід TDO. Перші 8 тактів використовуються для первинного завантаження зсувного регістру, оскільки біти, що вивантажуються в цей час, необхідно ігнорувати.

В інших моделях як регістр даних обирається 8-бітний регістр даних FLASH-пам'яті. Активні стани:

– Capture-DR – вміст обраного байта замикається в регістрі даних FLASH-пам'яті. Чергування старшого і молодшого байтів при кожному проходженні стану Capture-DR здійснюється автоматично, починаючи з молодшого байта при завантаженні команди.

– Shift-DR – здійснюється побітове завантаження байта даних.

3.4.3 Алгоритм програмування

У розділі описуються дії, які необхідно виконувати для програмування мікроконтролерів через порт JTAG. Формат усіх команд, які використовуються при програмуванні, наведено в табл. 3.20.

Таблиця 3.20 – Команди програмування по інтерфейсу JTAG

Команда		TDI	TDO	Примітка
1	a. Знищення кристала	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	b. Запит стану операції	0110011_10000000	xxxxx0x_xxxxxxxx	2
2	a. Вхід у режим запису FLASH-пам'яті	0100011_00010000	xxxxxxx_xxxxxxxx	–
	b. Завантаження додаткового байта адреси	0001011_cccccccc	xxxxxxx_xxxxxxxx	–
	c. Завантаження старшого байта адреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	d. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	e. Завантаження молодшого байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	–
	f. Завантаження старшого байта даних	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	–
	g. Фіксація даних	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	h. Запис сторінки FLASH-пам'яті	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	-

Продовження таблиці 3.20

Команда		TDI	TDO	Примітка
	i. Запит стану операції	0110111_00000000	xxxxxxx_xxxxxxxx	2
3	a. Вхід в режим читання FLASH-пам'яті	0100011_00000010	xxxxxxx_xxxxxxxx	–
	b. Завантаження додаткового байта адреси	0001011_cccccccc	xxxxxxx_xxxxxxxx	–
	c. Завантаження старшого байта адреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	d. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	e. Читання молодшого і старшого байтів даних	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_00000000 xxxxxxx_00000000	9 8
4	a. Вхід в режим запису EEPROM-пам'яті	0100011_00010001	xxxxxxx_xxxxxxxx	–
	b. Завантаження старшого байта адреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	c. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	d. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	–
	e. Фіксація даних	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx xxxxxxxx xxxxxxx xxxxxxxx xxxxxxx xxxxxxxx	–
	f. Запис сторінки EEPKOM-пам'яті	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	g. Запит стану операції	0110011_00000000	xxxxx0x_xxxxxxxx	2
5	a. Входу режим читання EEPROM-пам'яті	0110011_00000011	xxxxxxx_xxxxxxxx	–
	b. Завантаження старшого байтуадреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	c. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	d. Читання байта даних	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	–

Продовження таблиці 3.20

Команда		TDI	TDO	Примітка
6	a. Вхід у режим запису конфігураційних комірок	0100011_01000000	xxxxxxx xxxxxxxx	–
	b. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	3,5
	c. Запис додаткового конфігураційного байта	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	d. Запит стану операції	0110111_00000000	xxxxxox_xxxxxxxx	2
	e. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	3,5
	f. Запис старшою конфігураційного байта	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	g. Запит стану операції	0110111_00000000	xxxxxxx_xxxxxxxx	2
	h. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	3, 5
	i. Запис молодшого конфігураційного байта	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
6	j. Запит стану операції	0110011_00000000	xxxxxxx_xxxxxxxx	2
7	a. Вхід у режим запису комірок захисту	0100011_00100000	xxxxxxx_xxxxxxxx	–
	b. Завантаження байта даних	001001_11iiiiii	xxxxxxx_xxxxxxxx	3,6
	c. Запис байта зашитий	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	d. Запит стану операції	0110011_00000000	xxxxxxx_xxxxxxxx	2
8	a. Вхід у режим читання конфігураційних комі-рок і комірок захисту	0100011_00000100	xxxxxxx_xxxxxxxx	–
	b. Читання додаткового конфігураційного байта	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	5
	c. Читання старшого конфігураційного байта	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	5

Продовження таблиці 3.20

Команда		TDI	TDO	Примітка
	d. Читання мол. конфігураційного байта	0110010_00000000 0110011_00000000	xxxxxxx xxxxxxxx xxxxxxx_oooooooo	5
	e. Читання байта захисту	0110110_00000000 0110111_00000000	xxxxxxx xxxxxxxx xxxxxxx xxoooooo	4, 6
	f. Читання конфігураційних комірок і комірок захисту	0111010 00000000 0111110 00000000 0110010 00000000 0110110 00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxxoooooooo xxxxxxx oooooooo xxxxxxx oooooooo xxxxxxx xxoooooo	4 4, 7 4, 8 4, 9 4, 10
9	a. Вхід у режим читання комірок ідентифікатора	0100011 00001000	xxxxxxx_xxxxxxxx	–
	b. Завантаження байта адреси	000001_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	c. Читання байта ідентифікатора	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	–
10	a. Вхід до режиму читання комірки калібрування	0100011_00001000	xxxxxxx_xxxxxxxx	–
	b. Завантаження байта адреси	000001_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	c. Читання константи калібрування	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxxoooooooo	–
11	a. Завантаження команди «Немає операції»	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–

Примітки:

1. Умовні позначення, що використовуються в таблиці:
a – біти старшого байта адреси;
b – біти молодшого байта адреси;
i – дані, що посилаються до мікроконтролера;
o – дані, що зчитуються з мікроконтролера;
x – довільний стан біта.
2. Повторювати до тих пір, поки o = 1.
3. Для програмування комірки відповідний їй біт має бути скинутий, для знищення – встановлений.
4. 0 – комірка запрограмована, 1 – незапрограмована.
5. Відповідність біт коміркам конфігурації (табл. 3.5).
6. Відповідність біт коміркам захисту (рис. 3.5).
7. Додатковий байт.
8. Старший байт.
9. Молодший байт.
10. Байт захисту.

Операції програмування (номера команд відповідно до табл. 3.20). Операція «Вхід до режиму програмування»:

1. Завантажити команду AVR_RESET і занести «1» у регістр скидання.

2. Завантажити команду PROG_ENABLE і занести в регістр дозволу програмування значення 1010_0011_0111_ 0000 ($ A370).

Операція «Вихід з режиму програмування»:

1. Завантажити команду PROG_COMMANDS.
2. Заборонити виконання всіх команд програмування, запустивши команду «Немає операції».
3. Завантажити команду PROG_ENABLE і занести в регістр дозволу програмування значення 0000_0000_0000_0000.
4. Завантажити команду AVR_RESET і занести «0» у регістр скидання.

Операція «Знищення кристала»:

1. Завантажити команду PROG_COMMANDS.
2. Почати знищення кристала, запустивши команду 1a.
3. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 1b. Як альтернативу можна просто зачекати (див. табл. 3.16).

Операція «Програмування FLASH-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування FLASH-пам'яті (команда 2a).
3. Завантажити додатковий байт адреси (команда 2b).
4. Завантажити старший байт адреси (команда 2c).
5. Завантажити молодший байт адреси (команда 2d).
6. Завантажити дані (команди 2e, 2f і 2g).
7. Повторити п. 5 та 6 для кожної комірки сторінки пам'яті програм.
8. Зберегти сторінку (команда 2h).
9. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 2h.
10. Повторити п. 3...9 для програмування інших сторінок пам’яті програм.

Операція «Читання FLASH-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Перейти в режим читання FLASH-пам'яті (команда 3a).
3. Завантажити адресу комірки пам'яті (команди 3b, 3c та 3d).
4. Прочитати вміст комірки (команда 3c).
5. Повторити п. 3 та 4 для читання інших комірок пам'яті програм.

Операція «Програмування EEPROM-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування EEPROM-пам'яті (команда 4a).
3. Завантажити старший байт адреси (команда 4b).
4. Завантажити молодший байт адреси (команда 4c).
5. Завантажити дані (команди 4d і 4e).
6. Повторити п. 4 і 5 для всіх комірок сторінки.
7. Зберегти сторінку (команда 4f).

8. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 4g.

9. Повторити п. 3...8 для програмування інших сторінок EEPROM-пам'яті.

Операція «Читання EEPROM-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання EEPROM-пам'яті (команда 5a).
3. Завантажити адресу комірки пам'яті (команди 5b та 5c).
4. Зчитати вміст комірки (команда 5d).
5. Повторити п. 3 та 4 для читання інших комірок EEPROM-пам'яті.

Операція «Програмування комірок конфігурації»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування конфігураційних комірок (команда 6-a).
3. Завантажити байт даних (команда 6b).
4. Записати додатковий байт конфігурації (команда 6c).
5. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 6d.
6. Завантажити байт даних (команда 6e).
7. Записати старший байт конфігурації (команда 6f)
8. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 6g.
9. Завантажити байт даних (команда 6h).
10. Записати молодший байт конфігурації (команда 6i).
11. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 6j.

Операція «Програмування комірок захисту»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування комірок захисту (команда 7a).
3. Завантажити байт даних (команда 7b).
4. Записати байт захисту (команда 7c).
5. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 7d.

Операція «Читання конфігураційних комірок та комірок захисту»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання конфігураційних комірок та комірок захисту (команда 8a).
3. Для читання всіх комірок виконати команду 8f.
4. Для читання тільки додаткового байта конфігурації виконати команду 8b.

5. Для читання тільки старшого байта конфігурації виконати команду 8c.

6. Для читання тільки молодшого байта конфігурації виконати команду 8d.

7. Для читання тільки байту захисту виконати команду 8e.

Операція «Читання комірок ідентифікатора»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання комірок ідентифікатора (команда 9a).
3. Завантажити адресу $00, використовуючи команду 9b.
4. Прочитати перший байт ідентифікатора (команда 9c).
5. Повторити п. 3 та 4 для читання другого (адреса $01) та третього (адреса $02) байта ідентифікатора.

Операція «Читання комірок калібрування»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання комірок калібрування (команда 10a).
3. Завантажити адресу комірки (команда 10b).
4. Зчитати значення константи калібрування (команда 10c).

3.5 Самопрограмування мікроконтролерів набору Mega

Усі МК набору Mega мають можливість самопрограмування, тобто самостійно змінювати вміст своєї пам'яті програм.

У всіх мікроконтролерах набору, за винятком моделі ATmega48x, уся область пам'яті програм логічно розділена на дві секції – секцію прикладної програми (Application Section) і секцію завантажувача (Boot Loader Section). Зміна параметрів пам'яті програм здійснюється програмою-завантажувачем, що розташована в одноіменній секції завантаження нового вмісту пам'яті програм. Для вивантаження пам’яті програма-завантажувач може використовувати будь-який інтерфейс передачі даних (USART, SPI, TWI), що наявний у складі конкретного мікроконтролера. Завантажувач може змінювати вміст обох секцій. Це дозволяє йому модифікувати власний код і навіть видаляти себе з пам'яті, якщо в ньому не буде потреби. Рівень доступу (читання/запис) до кожної з секцій задається користувачем за допомогою комірок захисту BLB02: BLB01 та BLB12: BLB11 (див. табл. 3.3, табл. 3.4). У моделі ж ATmega48x зміна вмісту пам'яті програм може бути здійснена з будь-якого її місця, тобто секцією завантажувача, по суті справи, є вся пам'ять програм. Перехід до програми-завантажувача може здійснюватися різними способами. Зокрема, вона може бути викликана з основної програми командами CALL/JMP. Іншим способом є переміщення вектора скидання в початок секції завантажувача. У цьому випадку запуск програми-завантажувача буде здійснюватися автоматично після кожного скидання мікроконтролера. Положення вектора

скидання визначається станом комірки конфігурації BOOTRST. Якщо в ній міститься 1, вектор скидання розташовується на початку пам'яті програм, за адресою $0000. Якщо комірка запрограмована (0), то вектор скидання розташовується на початку секції завантажувача (див. табл. 3.21). Розмір секції завантажувача і відповідно розмір секції прикладної програми практично у всіх мікроконтролерах задається за допомогою двох конфігураційних комірок – BOOTSZ1: BOOTSZ0. Виняток становлять лише моделі ATmega48x, у яких поділ на секції відсутній. Можливі конфігурації пам'яті програм всіх мікроконтролерів набору Mega наведені в табл. 3.21.

Таблиця 3.21 – Конфігурація пам'яті програм

Модель	BOOTSZ1	BOOTSZ0	Розмір завантажувача [слів]	К-сть сторінок	Секція прикладної програми	Секція завантажувача
ATmega8515x/8535x, ATmega8x, ATmega88x	1	1	128	4	$000...$F7F	$F80...$FFF
	1	0	256	8	$000...$EFF	$F00...$FFF
	0	1	512	16	$000...$DFF	$E00...$FFF
	0	0	1024	32	$000...$BFF	$C00...$FFF
ATmegal6x, ATmegal62x, ATmegal64x, ATmegal65x, ATmegal68x	1	1	128	2	$0000...$1F7F	$1F80...$1FFF
	1	0	256	4	$0000...$1EFF	$1F00...$1FFF
	0	1	512	8	$0000...$1 DFF	$ 1 E00...$ 1FFF
	0	0	1024	16	$000...$1BFF	$1C00...$1FFF
ATmega32x, ATmega324x, ATmega325x/3250x	1	1	256	4	$0000...$3EFF	$3F00...$3FFF
	1	0	512	8	$0000...$3DFF	$3E00...$3FFF
	0	1	1024	16	$0000...$3BFF	$3C00...$3FFF
	0	0	2048	32	$0000...$37FF	$3800...$3FFF
ATmega64x, ATmega640x, ATmega645x/6450x	1	1	512	4	$0000...$7DFF	$7E00...$7FFF
	1	0	1024	8	$0000...$7BFF	$7C00...$7FFF
	0	1	2048	16	$0000...$77FF	$7800...$7FFF
	0	0	4096	32	$0000...$6FFF	$7000...$7FFF
ATmegal28x, ATmegal280x/1281x	1	1	512	4	$0000...$FDFF	$FE00...$FFFF
	1	0	1024	8	$0000...$FBFF	$FC00...$FFFF
	0	1	2048	16	$0000...$F7FF	$F800...$FFFF
	0	0	4096	32	$0000...$EFFF	$F000...$FFFF
ATmega2560x/2561x	1	1	512	4	$0000...$1FDFF	$1FE00...$FFFF
	1	0	1024	8	$0000...$! FBFF	$1FC00...$FFFF
	0	1	2048	16	$0000...$ 1F7FF	$1F800...$FFFF
	0	0	4096	32	$0000...$1EFFF	$1F000...$FFFF

Зміна вмісту пам'яті програм і комірок захисту завантажувача здійснюється за допомогою команди SPM (Store Program Memory). Параметрами цієї команди є адреса області пам'яті, що завантажується попередньо в індексний регістр Z, і, за необхідності, дані, що знаходяться в регістровій парі R1: R0. Управління процесом програмування і, зокрема, визначення опера-

ції, що виконується при виклику команди SPM, здійснюється за допомогою регістра введення/виведення SPMCR (моделі ATmega8515x/8535x, ATmega8x/16x/32x/64x/128x і ATmegal62x) або SPMCSR (в інших моделях). У всіх мікроконтролерах, за винятком моделей ATmega64x/128x, цей регістр розташований за адресою $ 37 ($ 57). У моделях ATmega64x/128x цей регістр розташований в просторі додаткових регістрів введення/виведення за адресою ($68). Формат цих регістрів для різних моделей набору показаний на рис. 3.6, а опис його біт наведено в табл. 3.22.

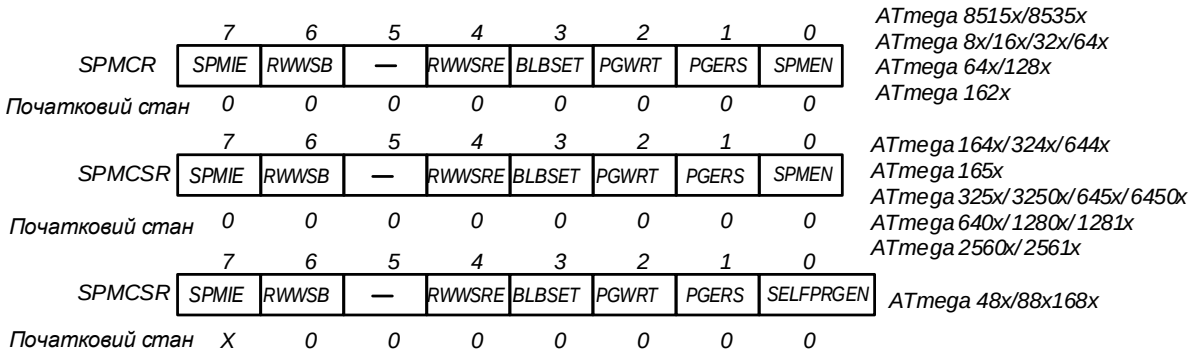


Рисунок 3.6 – Формат регістра SPMCR/SPMCSR

Таблиця 3.22 – Біти регістра SPMCR/SPMCSR

Біт	Назва	Опис
1	2	3
7	SPMIE	Дозвіл переривання «Готовність SPM». Якщо в цьому біті записана «1» і прапорець I регістра SREG також встановлено в «1», то дозволяється переривання «Готовність SPM». Переривання генерується весь час, поки біт SPMEN регістра скинутий в «0»
6	RWWSB	Заборонено доступ до області RWW. Цей прапорець показує можливість звернення за адресами, розташованим в області RWW. Якщо прапорець встановлений в «1», доступ до області RWW заборонений, якщо скинутий в «0» дозволено. Установлення цього прапорця здійснюється апаратно при виконанні операцій запису або знищення сторінки пам'яті. Скидання прапорця здійснюється або програмно, записом «1» в біт RWWSRE після закінчення операції, або апаратно, при запуску операції завантаження сторінки
5	–	Зарезервовано
4	RWWSRE	Читання області RWW дозволено. Одночасне встановлення цього біта і біта SPMEN дозволяє доступ до області RWW. Дозвіл доступу до цієї області здійснюється запуском команди SPM протягом 4 тактів після встановлення зазначених бітів. Дозвіл доступу до області RWW може здійснюватися тільки після завершення операції програмування (після скидання прапорця SPMEN)

Продовження таблиці 3.23

3	BLBSET	Зміна комірок захисту завантажувача. При одночасній установці цього біта і біта SPMEN команда SPM, запущена протягом таких 4 тактів, здійснить установку захисних комірок завантажувача відповідно до вмісту регістра R0. Скидання біта BLBSET здійснюється апаратно після встановлення комірок захисту або після закінчення зазначеного часу. За командою (E)LPM, запущеної протягом 3 тактів після встановлення зазначених біт, буде здійснено читання або конфігураційних комірок або комірок захисту (залежить від значення біта Z0 регістра Z)
2	PGWRT	Запис сторінки. При одночасній установці цього біта і біта SPMEN команда SPM, запущена протягом таких 4 тактів, здійснить запис сторінки пам'яті програм з тимчасового буфера. Адреса сторінки має бути завантажена в старший байт регістра Z (R31). Скидання біта PGWRT здійснюється апаратно після закінчення запису сторінки або після закінчення зазначеного часу. При запису в секцію NRWW центральний процесор зупиняється на час виконання
1	PGERS	Знищення сторінки. При одночасній установці цього біта і біта SPMEN команда SPM, запущена протягом таких 4 тактів, здійснить знищення сторінки пам'яті програм. Адреса сторінки має бути завантажена в старший байт регістра Z (R31). Скидання біта PGERS здійснюється апаратно після закінчення знищення сторінки або після закінчення зазначеного часу. При запису в секцію NRWW центральний процесор зупиняється на час виконання операції
0	SPMEN (SELFPRGEN)	Дозвіл виконання команди SPM. Установлення цього біта дозволяє запуск команди SPM протягом таких 4 тактів. Якщо біт SPMEN встановлюється одночасно з одним з бітів RWWSRE, BLBSET, PGWRT або PGERS, виконується операція, яка визначається цим бітом. Якщо встановлюється тільки біт SPMEN, здійснюється збереження вмісту регістрів R1: R0 у тимчасовому буфері за адресою, що знаходиться в регістрі Z. Скидання біта SPMEN здійснюється апаратно після завершення операції або після закінчення зазначеного часу

Під час запису в EEPROM-пам'ять змінювати регістр SPMCR неможливо. Тому, перед тим як записати будь-яке значення в регістр SPMCR, рекомендується дочекатися скидання прапорця EEWE регістра EECR.

Для адресації пам'яті програм при використанні команди SPM використовується індексний регістр Z, який складається з R30 (молодший байт) і R31 (старший байт), а в моделях з об'ємом пам'яті програм більше 64 Кбайт – ще і регістр введення/виведення RAMPZ. Оскільки пам'ять програм в мікроконтролерах набору має сторінкову організацію, лічильник команд можна умовно розбити на дві частини. Перша частина (молодші біти) адресує комірка на сторінці, а друга частина визначає сторінку. Після запуску операції програмування вміст регістра Z фіксується і його можна використовувати для інших цілей.

Зміна вмісту пам'яті програм здійснюється в такій послідовності: заповнення тимчасового буфера сторінки новим вмістом; видалення сторінки; вміст буфера в пам'ять програм.

Для визначення моменту закінчення виконання операцій можна або опитувати стан прапорця SPMEN регістра SPMCR, чекаючи його скидання, або скористатися перериванням «Готовність SPM». Це переривання генерується весь час, поки прапорець SPMEN скинутий. В останньому випадку таблиця векторів переривань має знаходитися в секції завантажувача, а це переривання має бути дозволено установкою прапорця SPMIE регістра SPMCR.

Для знищення сторінки пам'яті програм необхідно занести адресу сторінки в регістр Z (секція PCPAGE), записати значення x0000011 в регістр SPMCR і протягом таких чотирьох тактів виконати команду SPM. Вміст регістрів R1 і R0 при цьому ігнорується.

Для занесення слова команди в буфер варто завантажити адресу комірки в регістр Z (секція PCWORD), а код операції – у регістри R1: R0. Після цього необхідно записати значення x0000011 в регістр SPMCR і протягом таких чотирьох тактів виконати команду SPM. Очищення буфера здійснюється автоматично, після закінчення запису сторінки, або вручну, записом «1» в біт RWWSRE регістра SPMCR.

Запис вмісту буфера в пам'ять програм здійснюється аналогічно. У регістр Z (секція PCPAGE) заноситься адреса сторінки, у регістр SPMCR записується значення x0000101, і протягом таких чотирьох тактів виконується команда SPM. Вміст регістрів R1 і R0 при цьому ігнорується.

Зміна комірок захисту завантажувача BLB12: BLB11 і BLB02: BLB01 також здійснюється командою SPM. Для цього необхідно завантажити в регістр R0 необхідне значення відповідно до рис. 3.7 (скинутий біт означає програмування відповідної комірки).

7	6	5	4	3	2	1	0
1	1	BLB12	BLB11	BLB02	BLB01	1	1

Рисунок 3.7 – Вміст R0 при зміні комірок захисту завантажувача

Після цього необхідно записати значення x0001001 у регістр SPMCR і протягом таких чотирьох тактів виконати команду SPM. Уміст регістра Z при цьому ігнорується, проте для сумісності з майбутніми пристроями рекомендується записувати в нього значення $ 0001. Під час програмування комірок захисту можна звертатися до будь-якої області пам'яті програм.

Крім програмування мікроконтролера, завантажувач може також зчитувати вміст конфігураційних комірок і комірок захисту. Так, для читання байту захисту варто завантажити в регістр Z число $0001 записати в регістр SPMCR значення x0001001 і протягом трьох таких тактів виконати команду LPM. У результаті вміст байту захисту буде занесено в заданий регістр загального призначення. Відповідність бітів регістра коміркам наведено на рис. 3.1.

Читання конфігураційних байтів здійснюється аналогічно. У регістр Z завантажується адреса байта ($0000 – молодший байт, $0003 – старший байт, $0002 – додатковий байт), після чого в регістр SPMCR потрібно записати значення x0001001 і протягом трьох таких тактів виконати команду LPM. У результаті виконання команди вміст обраного байту конфігурації буде занесено в регістр загального призначення.

3.6 Розробка комутатора панелі ZIF для програмування мікроконтролерів Atmel

Паралельне програмування вимагає використання додаткового джерела підвищеної напруги (12В), використовує велике число виводів мікроконтролера і виконується на спеціальних програматорах. Таке програмування зручне, коли при масовому виробництві необхідно "прошивати" велику кількість кристалів. Послідовне програмування не вимагає додаткового джерела живлення і може виконуватися безпосередньо в мікропроцесорній системі (In System Programming) через послідовний SPI – інтерфейс (рис. 3.2), що використовує всього чотири виводи AVR-мікроконтролера. Можливість внутрішньосистемного програмування є однією з найважливіших переваг AVR, тому що дозволяє значно спростити й здешевити процес розробки і модернізації програмного забезпечення. Паралельний і послідовний способи програмування припускають використання зовнішнього пристрою програмування.

Для внутрішньосхемного програмування мікроконтролерів AVR існує велика кількість різноманітних програматорів. Широко поширення набув програматор USBASP [12, 18, 24, 28, 30], який буде використовуватись разом з комутатором ZIF-панелі. Інформація між мікроконтролером та програматором передається порозрядно послідовним кодом, причому кожний розряд супроводжується по лінії синхронізації (SCK) імпульсом, який генерує програматор. Для передачі даних з програматора в мікроконтролер фірми Atmel слугує ланка MOSI (Master-Out Slave-In), а у зворотному напрямку MISO (Master-In Slave-Out). Схема адаптера для програмування ІМС фірми Atmel зображена на рис. 3.8. При розробці комутатора був проведений аналіз розташування виводів найбільш популярних мікроконтролерів фірми Atmel. Його результати відображені в табл. 3.24. МК розподілені на п'ять груп, кожна з яких об'єднує мікросхеми з однаковим числом виводів (8, 20, 28 або 40) та однаковим розташуванням тих з них, що використовуються для програмування, у тому числі виводів живлення та загальної шини. У табл. 3.25 наведено, з якими контактами панелі ZIF необхідно з'єднати відповідні виводи програматора за умови, що перший вивід мікроконтролера вставлений в перше гніздо панелі ZIF. Лінії для програмування MOSI, RST, SCK, VCC, GND, що є вхідними для мікроконтролерів можливо під’єднати до відповідних виводів ZIF-панелі через 5 буферних регістрів (табл. 3.25). Буферні регістри знаходяться в z-стані. Один з буферних регістрів вмикається залежно від типу мікроконтролера, що буде програмуватись, і з’єднує його виводи MOSI, RST, SCK, VCC, GND з відповідними виводами ZIF-панелі.

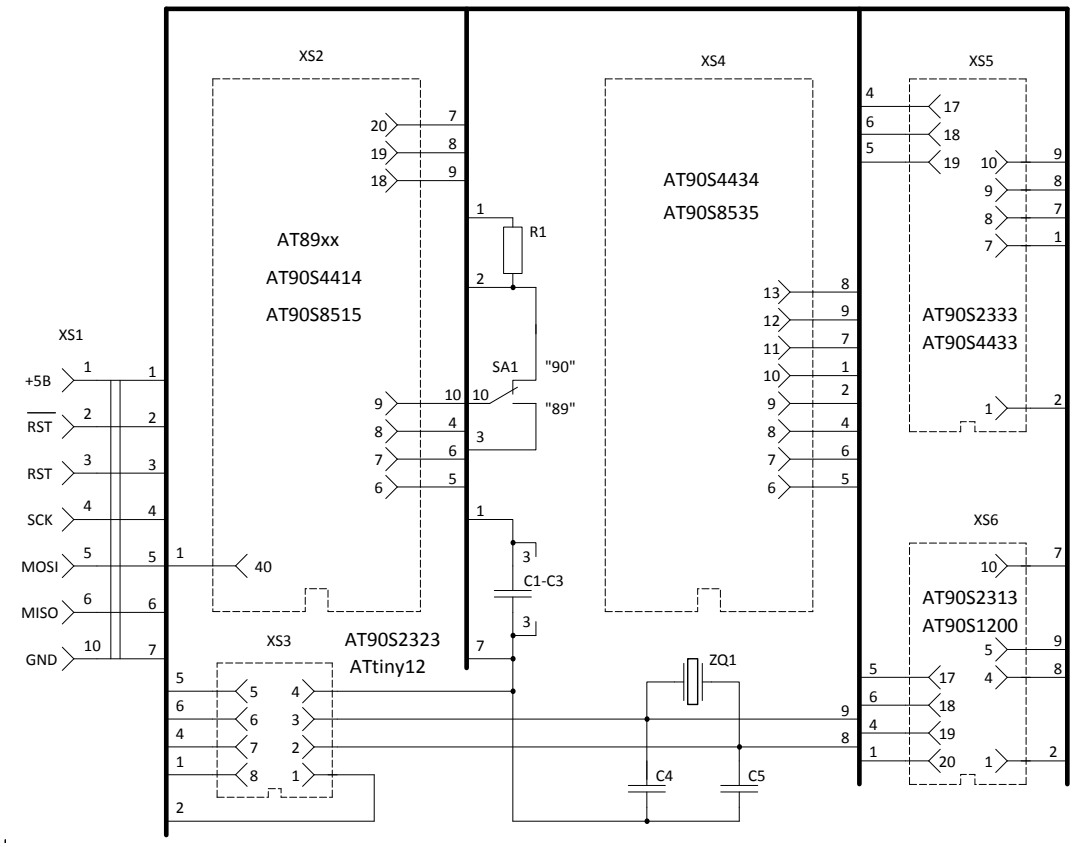


Рисунок 3.8 – Схема адаптера для програмування ІМС фірми Atmel

Таблиця 3.24 – Розташування виводів мікроконтролерів AVR, що використовуються для внутрішньосхемного програмування

Група	Мікроконтролер	Номера виводів МК для ISP програмування						
		MOSI	MISO	SCK	RES	XTAL	GND	U_{CC}
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	AT90S2313 ATtiny2313	17	18	19	1	5	10	20
2	AT90S8515 AT89S8252 ATmega8515	6	7	8	9	19	20	40
3	AT90S8535 ATmega16 ATmega32 ATmega8535	6	7	8	9	13	11, 31	10,30
4	ATmega8 ATmega48 ATmega88 ATmega168 ATmega328	17	18	19	1	9	8, 22	7, 20
5	ATtiny12 ATtiny13 ATtiny15	5	6	7	1	2	4	8

Таблиця 3.25 – Відповідність виводів для ISP програмування у ZIF-панелі

Група	Мікроконтролер	Номера виводів ZIF-панелі						
		MOSI	MISO	SCK	RES	XTAL	GND	U_{CC}
1	AT90S2313 ATtiny2313	37	38	39	1	5	10	40
2	AT90S8515 AT89S8252 ATmega8515	6	7	8	9	19	20	40
3	AT90S8535 ATmega16 ATmega32 ATmega8535	6	7	8	9	13	11, 31	10,30
4	ATmega8 ATmega48 ATmega88 ATmega168 ATmega328	29	30	31	1	9	8, 22	7, 32
5	ATtiny12 ATtiny13 ATtiny15	37	38	39	1	2	4	40

Обираємо мікросхему 74HC541 як буферний регістр для роботи у складі комутатора ZIF-панелі. Функціональна схема 74HC541 наведена на рис. 3.9 [1]. Мікросхема складається з восьми буферів D0-D7. Коли на обох входах OEA, OEB буде рівень логічного «0», мікросхема виходить з високоімпедансного стану, що можна використати для вибору типу мікроконтролера, який буде програмуватись.

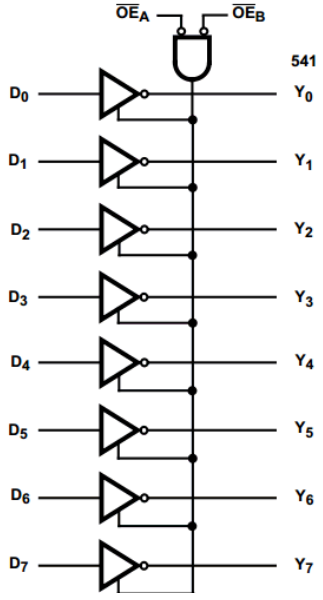


Рисунок 3.9 – Функціональна схема мікросхеми 74HC541

Основні експлуатаційні параметри мікросхеми 74HC541 [11, 19, 20]: напруга живлення – 2–6 В, максимальний вихідний струм – 75 мА, максимальна частота – 50–150 МГц, затримка розповсюдження сигналу – 3,5–5 нс, рівень логічного «0» – менше 0,1 В, рівень логічної «1» – U жив. Для правильного з'єднання вихідного сигналу MISO з програматором буде використаний селектор-мультиплексор CD4051BCN. Сигнали на адресних входах такої мікросхеми формує шифратор на діодах VD1–VD7 залежно від вибраної мікросхеми для програмування (рис. 3.10).

Таймер DA1 генерує імпульси з частотою 1 МГц, які необхідні для тактування мікроконтролера при програмуванні. Вони поступають, якщо перемичкою S2 з'єднати контакти 1, 2 з'єднувача XP3.

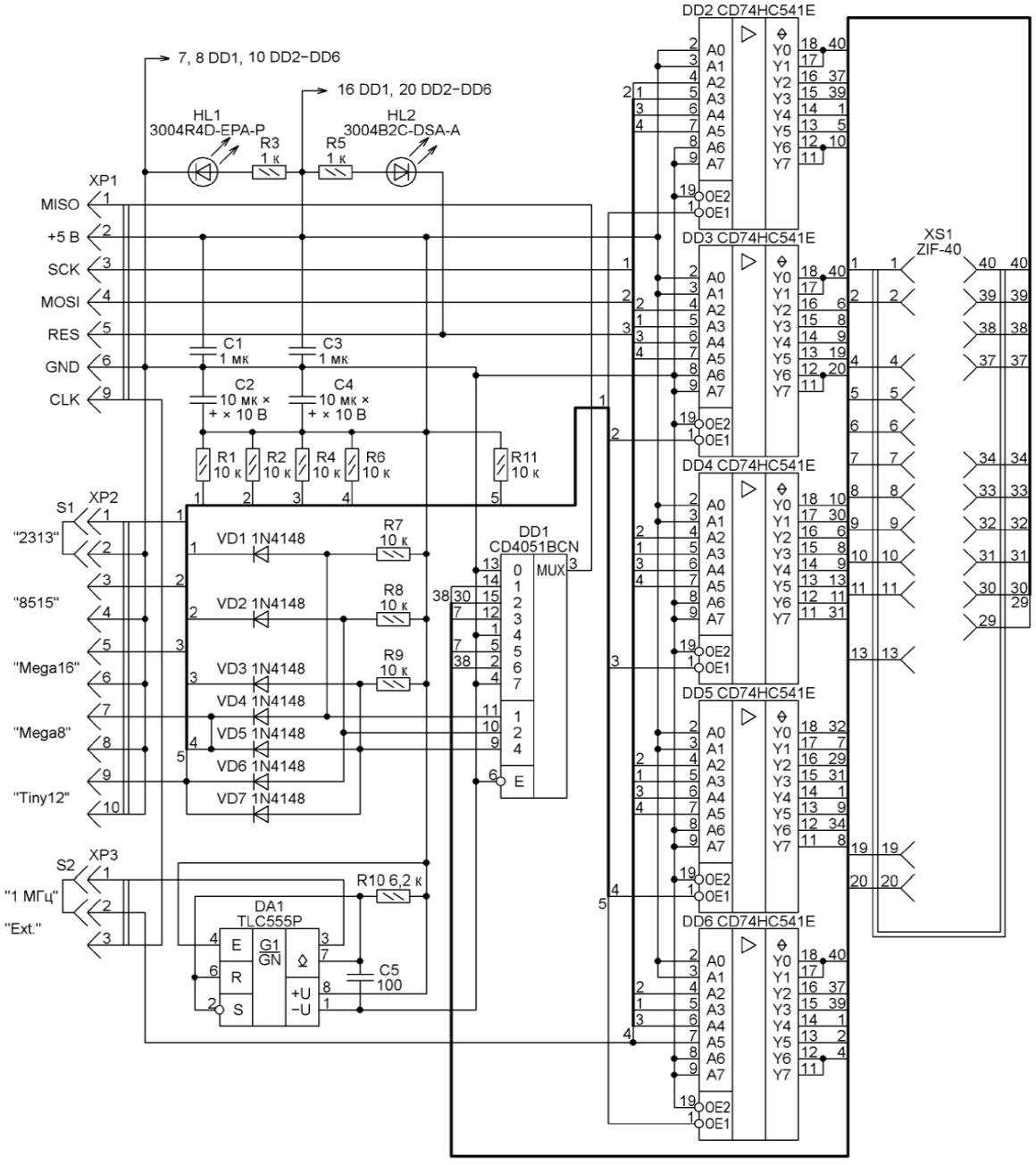


Рисунок 3.9 – Схема комутатора ZIF-панелі

Питання для самоконтролю

1. Які режими програмування підтримують МК набору AVR?
2. У чому полягає захист коду і даних?
3. Яке призначення конфігураційних комірок?
4. Як називаються 8-бітові комірки, вміст яких дозволяє ідентифікувати пристрій?
5. Поняття комірок калібрування та констант калібрування.
6. Як організовано пам'ять програм і даних?
7. Розкажіть про особливості програмування по послідовному каналу.
8. Які дії необхідно виконати для переведення мікроконтролера в режим програмування по послідовному каналу?
9. Опишіть процес управління програмування FLASH-пам'яті.
10. Опишіть процес програмування EEPROM-пам'яті.
11. Опишіть особливості паралельного програмування.
12. Які дії необхідно виконати для перемикання в режим паралельного програмування?
13. Яка команда повністю знищує вміст FLASH- і EEPROM-пам'яті?
14. Послідовність програмування FLASH-пам'яті.
15. Послідовність програмування EEPROM-пам'яті.
16. Послідовність програмування конфігураційних комірок.
17. Як відбувається програмування, читання комірок захисту та конфігурації?
18. Опишіть процес читання комірок ідентифікатора і комірок калібрування.
19. Дайте коротку характеристику програмуванню по інтерфейсу JTAG.
20. Опишіть використання інтерфейсу JTAG для програмування кристала.
21. Перелічіть команди JTAG для програмування.
22. Дайте визначення поняттю самопрограмування мікроконтролерів набору Mega.
23. Які етапи розробки комутатора панелі ZIF для програмування мікроконтролерів Atmel?

ВИСНОВКИ

Системи з мікроконтролерами належать до класу цифрових обчислювальних систем, тому методологія їхнього проектування має містити елементи апаратної та програмної підтримки. Увесь технологічний процес створення програмного продукту можна розбити на кілька етапів. Черговість проектування полягає в складанні технічного завдання, розробці та деталізації алгоритму, наборі лістинга програми, компілюванні, моделюванні, програмуванні та реальній роботі. За результатами моделювання або реальної роботи можуть бути змінені вихідні дані технічного завдання, відкоригований алгоритм функціонування.

Набір тексту лістинга, компілювання, моделювання та програмування можуть здійснюватися як окремими комп'ютерними програмами, так і в комплексі.

Компілятори 8-бітових МК є вузькоспеціалізовані комп'ютерні програми. Вони можуть бути автономними або входити до складу пакетів, що мають власну графічну оболонку, редактор тексту, засоби моделювання та програмування. Переважна більшість таких комплексів комерційні, платні.

При розробці програмного забезпечення мікроконтролерів необхідно звернути увагу на безкоштовні програмні засоби (AVR Studio, WinAVR) підтримки проектування та налагодження систем на мікроконтролерах AVR, робота з якими економить гроші та забезпечує ліцензовану чистоту кінцевого програмного продукту.

Обираючи інструментальні засоби налагодження, доцільно брати до уваги: підтримка можливо більшої кількості мікроконтролерів; різноманітність вбудованих інтерфейсів (RS-232, IEEE1284/LPT, USB) та додаткових компонентів, що розширюють функціональні можливості; наявність на платі поля для макетування і контактів для безпосереднього підключення до виводів приладу; можливість підключення додаткових модулів; універсальне живлення.

МК AVR при програмуванні застосовують такі технології: низьковольтне послідовне програмування (ISP); низьковольтне шлейфове програмування з можливістю відлагодження (JTAG); низьковольтне однопровідне програмування з можливістю відлагодження (debugWire); високовольтне паралельне програмування (HVPP), самопрограмування (BootLoader). Кількість виводів мікроконтролерів для їхнього програмування зводиться до мінімуму. Для програмування достатньо ліній синхронізації, однієї – двох ліній для передачі послідовним кодом команд, адрес й даних та лінії для подачі сигналу, який переводить мікроконтролер у режим програмування. Тому для програмування мікроконтролерів AVR усе частіше використовують інтерфейс USB та низьковольтне послідовне програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абакумов А. Коммутатор панели ZIF для программирования микроконтроллеров Atmel / А. Абакумов // Радио. – 2014. – № 2. – С. 20–21.

2. Белов А. В. Разработка устройств на микроконтроллерах AVR: шагаем от «чайника» до профи / А. В. Белов. – СПб. : Наука и Техника, 2013. – 528 с.

3. Величко В. Є. Використання технології візуального програмування в університетській освіті засобами вільного програмного забезпечення / В. Є. Величко // Вісник Житомирського державного університету імені Івана Франка. – 2014. – № 4. – С. 51–55.

4. Внутрисхемное программирование [Электронный ресурс]. – Режим доступа :
http:kit-e.ru/articles/circuit/2003_8_114.php.

5. Гурин А. Программно-инструментальные средства разработки и отладки / А. Гурин, П. Перевозников // Chip News. – 2004. – № 4 (87). – С. 12–16

6. Гёлль П. Как превратить персональный компьютер в универсальный програма тор / Гёлль П. – М. : ДМК ПРЕСС, 2002. – 168 с.

7. Гладун М. Комплект Formula Flowcode Buggy як засіб підвищення інтересу до навчання / М. Гладун // Інформатика та інформаційні технології в навчальних закладах. – 2015. – № 4. – С.11–16.

8. Евстифеев А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / Евстифеев А. В. – М. : Издательский дом «Додэка-XXI», 2007. – 592 с.

9. Краткий учебный курс PROTEUS [Электронный ресурс] / Русское руководство для начинающих. – Режим доступа :
http://proteus123.narod.ru.

10. Лебедев М. Б. CodeVisionAVR: пособие для начинающих / Лебедев М. Б. – М. : Додэка-XX1, 2008. –592 с.

11. Максимов А. Моделирование устройств на микроконтроллерах с помощью программы ISIS из пакета PROTEUS VSM / А. Максимов // Радио. –2005. –№ 4, 5, 6. – С. 30–33, 31–34, 30–32.

12. Меандр [Электронный ресурс] / Урок 5. Программирование AVR микроконтроллеров. – Режим доступа :
http://meandr.org.

13. Методичні вказівки до виконання студентами-магістрантами наукового напрямку економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський. – Вінниця : ВНТУ, 2012. – 12 с.

14. Островский К. В. Сравительный анализ средств разработки и отладки программного обеспечения для различных типов МК / К. В. Островский // Радиоэлектроника и информатика. – 2007. – № 2. – С. 79–84.

15. Офіційна web-сторінка ATMEL [Електронний ресурс]. – Режим доступу :
http://www.atmel.ru.

16. Программирование на языке С для AVR и PIC микроконтроллеров / Ю. А. Шпак. – К: «МК-Пресс», 2006. – 400 с.– ISBN 966-8806-16-6.

17. Протасов А. Г. Инструментальное средство для изучения микроконтроллеров семейства AVR / А. Г. Протасов, А. Л. Дугин // Вісник НТУУ «КПІ». Серія Приладобудування. – 2015. – Вип. 50(2). – С. 134–138.

18. Радиокот [Электронный ресурс] / «РЕАНИМАТОР» для AVR. – Режим доступа :

http://radiokot.ru/lab/controller/48.

19. Радиокот [Электронный ресурс] / Proteus – первое знакомство. – Режим доступа :

http://radiokot.ru/start/soft/proteus/01.

20. Ревич Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке асемблера / Ю. В. Ревич. – СПб. : БХВ-Петербург, 2008. – 384 с.

21. Рюмик С. М. 1000 и одна микронтроллерная схема / С. М. Рюмик. – М. : Додэка-XXI, 2010. –356 с.

22. Соммер У. Программирование микроконтроллерных плат Arduino/ Freeduino / Соммер У. – СПб. : БХВ-Петербург, 2012. – 256 с.

23. Хартов В. Я. Микроконтроллеры AVR. Практикум для начинающих / Хартов В. Я. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2012. – 280 с.

24. Хлюпин Н. Два универсальных програматора/ Н. Хлюпин // Радио. – 2006. – № 6. – С. 28–30.

25. Цирульник С. М. Візуальне програмування мікроконтролерів у системі FLOWCODE / С. М. Цирульник, А. О. Метелиця, В. А. Вознюк // VI Міжнародна науково-практична конференція «Актуальні проблеми наукового й освітнього простору в умовах поглиблення євроінтеграційних процесів» (14–15 травня 2015 р., Мукачево, Україна). – Мукачево: Вид-во «Карпатська вежа», 2015. – С. 374–375.

26. Цирульник С. М. Застосування програми ISIS пакету Proteus VSM при вивченні курсу «Мікропроцесорна техніка» // С. М. Цирульник, В. К. Задорожний // Матеріали XIII міжнародної конференції з автоматики управління (Автоматика 2006). – Вінниця : Універсум-Вінниця. – 2007. – С. 526–530.

27. Цирульник С. М. Проектування мікропроцесорних систем / С. М. Цирульник, Г. Л. Лисенко. – Вінниця : ВНТУ, 2012. – 191 с.

28. Easyelectronics [Электронный ресурс] / AVR. Учебный курс. Трактат о програматорах. – Режим доступа :

http://easyelectronics.ru/category/arm-uchebnyj-kurs.

29. GetChip [Электронный ресурс] / 024-Что такое Fuse bits AVR микроконтроллеров. – Режим доступа :

http://www.getchip.net/posts/024-chto-takoe-fuse-bits-AVR-mikrokontrollerov.

30. GetChip [Электронный ресурс] / 057-Программируем AVR-микроконтроллеры USBtiny + AVRdude + SinaProg. – Режим доступа :

http://www.getchip.net/posts/057-programmiruem-AVR-mikrokontrollery-usbtiny-AVRdude-sinaprog.

Навчальне видання

Цирульник Сергій Михайлович
Азаров Олексій Дмитрович
Крупельницький Леонід Віталійович
Трояновська Тетяна Іванівна

ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR

Навчальний посібник

Рукопис оформлено: Т. Трояновська

Редактор: О. Ткачук

Оригінал-макет виготовлено: О. Ткачук

Підписано до друку 31.05.2018.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 6,41.
Наклад 50 (1-й запуск 1–20) пр Зам. № 2018-104.

Видавець та виготовлювач
Вінницький національний технічний університет,
інформаційний редакційно-видавничий центр.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: kivc.vntu@gmail.com.

Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.