О. Є. Рубаненко, К. І. Кравцов, О. О. Рубаненко

# МІКРОПРОЦЕСОРНА ТЕХНІКА

## ВИКОРИСТАННЯ AVR МІКРОКОНТРОЛЕРІВ ATMEL

Міністерство освіти і науки України

Вінницький національний технічний університет

**О. Є. Рубаненко, К. І. Кравцов, О. О. Рубаненко**

# МІКРОПРОЦЕСОРНА ТЕХНІКА

# ВИКОРИСТАННЯ AVR МІКРОКОНТРОЛЕРІВ ATMEL

Лабораторний практикум

Вінниця
ВНТУ
2018

УДК 621.311(075)
Р40

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 10 від 29 травня 2014 р.)

Рецензенти:
**П. Д. Лежнюк**, д.т.н., професор
**В. М. Лисогор**, д.т.н., професор
**Ю. Д. Дементьєв**, к.т.н., доцент

**Рубаненко, О. Є.**

Р40 Мікропроцесорна техніка. Використання AVR мікроконтролерів ATMEL : лабораторний практикум / Рубаненко О. Є., Кравцов К. І., Рубаненко О. О. – Вінниця : ВНТУ, 2018. – 115 с.

Лабораторний практикум присвячено питанням використання однокристальних мікроконтролерів сімейства Mega фірми «ATMEL». Розглянуто архітектуру, її особливості. Описано внутрішню будову мікроконтролера, систему команд, периферію, а також програмне забезпечення, потрібне для конструювання і програмування схем на основі цих мікроконтролерів та способи програмування з прикладами реалізації деяких алгоритмів для конкретних енергетичних задач.

**УДК 621.311(075)**

ЗМІСТ

Список умовних позначень і скорочень .....6
Вступ .....7
1 Лабораторна робота № 1. AVR МІКРОКОНТРОЛЕРИ ТА РОБОТА З НИМИ .....12
1.1 Короткі теоретичні відомості .....12
1.1.1 Мікроконтролери Atmel AVR .....12
1.1.1.1 8-бітовий RISC FLASH-мікроконтролер .....14
1.1.1.2 Структура мікроконтролера .....17
1.1.1.3 Генератор тактового сигналу .....19
1.1.1.4 Процесор .....20
1.1.1.5 Регістри .....21
1.1.2 Опис мікроконтролера Atmega8515 фірми Atmel .....23
1.1.2.1 Опис виводів мікроконтролера Atmega8515 .....23
1.1.3 Процес розробки програм .....25
1.1.3.1 Вибір моделі .....25
1.1.3.2 Блок-схема алгоритму .....25
1.1.3.3 Написання програми .....26
1.1.3.4 Асемблювання .....27
1.1.3.5 Команди .....27
1.1.4 Загальна інформація про STK-500 .....31
1.2 Завдання .....32
1.3 Хід роботи .....32
1.4 Контрольні запитання .....35
2 Лабораторна робота № 2. ЗАПИС ТА ВИКОНАННЯ ПРОСТИХ ПРОГРАМ .....37
2.1 Короткі теоретичні відомості .....37
2.1.1 Мікроконтролери Atmel AVR .....37
2.2 Самостійна підготовка .....37
2.2.1 Завдання для самостійної роботи .....37
2.2.2 Виконання завдання .....38
2.2.3 Пояснення до тексту програми .....49
2.3 Робота в лабораторії .....49
2.4 Завдання .....49

2.5 Зміст до звіту до лабораторної роботи .....52
2.6 Контрольні запитання.....52
3 Лабораторна робота № 3. РОБОТА З ПОРТАМИ ВВЕДЕННЯ–ВИВЕДЕННЯ AVR.....53
3.1 Короткі теоретичні відомості .....53
3.1.1 Порти введення–виведення .....53
3.1.2 Загальні відомості про систему команд .....53
3.1.3 Програмування мікроконтролера.....55
3.1.4 Призначення та можливості STK-500 .....55
3.2 Самостійна підготовка.....61
3.2.1 Завдання для самостійної роботи.....61
3.2.2 Виконання завдання .....61
3.3 Хід роботи.....70
3.4 Зміст звіту .....71
3.5 Контрольні запитання.....71
4 Лабораторна робота № 4. ПІДПРОГРАМА І СТЕК.....72
4.1 Короткі теоретичні відомості .....72
4.2 Домашня підготовка .....74
4.3 Хід роботи.....74
4.4 Зміст звіту .....74
4.5 Контрольні запитання.....75
5 Лабораторна робота № 5. ВИКОРИСТАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ ДОДАВАННЯ ТА ВІДНІМАННЯ.....76
5.1 Короткі теоретичні відомості .....76
5.2 Домашня підготовка .....79
5.3 Хід роботи.....79
5.4 Зміст звіту .....79
5.5 Контрольні запитання.....80
6 Лабораторна робота № 6. ВИКОРИСТАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ МНОЖЕННЯ ТА ДІЛЕННЯ .....81
6.1 Короткі теоретичні відомості .....81
6.2 Хід роботи.....82
6.3 Домашня підготовка .....83
6.4 Зміст звіту .....83
6.5 Контрольні запитання.....83

7 Лабораторна робота № 7. ПРИЄДНАННЯ КЛАВІАТУРИ І ДИСПЛЕЯ ДО МК ....84
7.1 Короткі теоретичні відомості ....84
7.2 Домашня підготовка ....88
7.3 Хід роботи....89
7.4 Контрольні запитання....89
8 Лабораторна робота № 8. КЕРУВАННЯ ЕЛЕКТРИЧНИМИ ДВИГУНАМИ ЗА ДОПОМОГОЮ AVR МІКРОКОНТРОЛЕРІВ ....90
8.1 Опис апаратного та програмного забезпечення лабораторної роботи ....90
8.2 Домашня підготовка ....91
8.3 Хід роботи....91
8.4 Контрольні запитання....91
Література ....92
Додаток А. ....93
Додаток Б....96
Додаток В. ....107
Додаток Г. ....108
Додаток Д. ....110
Додаток Е. ....112
Додаток Ж. ....113

# СПИСОК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

| | | |
|---|---|---|
| МК | – | мікроконтролер |
| AT | – | Atmel |
| AVR | – | сімейство процесорів Atmel (Advanced Virtual RISC) |
| RISC | – | Reduced Instruction Set Computers |
| ISP | – | In System Programming |
| МП | – | мікропроцесор |
| ПЛК | – | програмні логічні контролери |
| АЛП | – | арифметичний логічний пристрій |
| РВВ | – | регістри введення–виведення |

# ВСТУП

Найтяжча частина – вступ.
(Марк Тулій Цицерон
греко-римський філософ та мислитель
(106-43 р.р. до н.е.))

Одна з характерних особливостей сучасного науково-технічного прогресу – надзвичайне прискорення темпів збільшення і оновлення знань. Розвиток людського суспільства, його культурний рівень безпосередньо пов'язані зі збільшенням кількості споживаної енергії, пошуком та обґрунтуванням нових, більш ефективних її видів. Нині науково-технічний прогрес неможливий без використання якісно нових видів енергії, в першу чергу, електричної [13].

Характерною рисою науково-технічного прогресу, яка визначає потужний подальший підйом суспільного виробництва, є широке впровадження електроніки в усі галузі народного господарства [12].

В даний час ми переживаємо справжню технічну революцію, пов'язану з приходом в енергетику нового покоління пристроїв – мікроелектронної і мікропроцесорної техніки. Це – мікропроцесорні пристрої релейного захисту та протиаварійної автоматики (наприклад, РС80, УЗА-10, «Діамант», УРЧ-3, МРЗС, Альтра – АСК), електровимірювальні прилади (наприклад, лічильник електричної енергії «Енергія-9»), вимірювальні системи, пристрої телемеханіки (наприклад, «Граніт-мікро»), пристрої технологічної автоматики і т. п. [13].

У наш час різко скоротився термін впровадження наукових ідей у практику і виробництво.

Часто перехід на мікропроцесорну елементну базу не приводить до зміни колишніх принципів роботи електричного обладнання, а тільки розширює його функціональні можливості, спрощує експлуатацію. Саме з цих причин мікропроцесорні пристрої дуже швидко займають місце застарілих видів устаткування.

Сучасна електронна цифрова обчислювальна техніка широко застосовується в енергетиці. В даний час нове покоління мікропроцесорів, мікроЕОМ і промислових мікроконтролерів має поліпшені техніко-економічні показники, що сприяє подальшому розширенню сфери їх застосування та ефективності їх використання в енергетичній галузі [1, 2].

Наприкінці 70-х років в результаті інтеграції всіх електронних пристроїв ЕОМ в одному кристалі були створені однокристальні мікроЕОМ, обчислювальна потужність яких не поступається обчислювальній потужності середніх ЕОМ початку 70-х років [3, 4].

Мікропроцесори і мікроЕОМ стали новим масовим класом ЕОМ внаслідок малої матеріалоємності і вартості, низького енергоспоживання і високої надійності [1 ÷ 4].

Масовість цього нового класу і його високі техніко-економічні параметри надають революціонізуючий вплив на ціле покоління приладів, обладнання, агрегати з вбудованими мікропроцесорними засобами [5].

На базі сучасних мікропроцесорів і мікроЕОМ створені високопродуктивні пристрої числового програмного керування складним енергетичним обладнанням [6 ÷ 8].

Будова сучасних мікропроцесорних систем контролю і управління в енергетиці на основі мікропроцесорних великих інтегральних схем (ВІС) дозволяє зменшити вартість таких систем, схожих за своїми параметрами з раніше створеними системами на основі застарілих ЕОМ, зменшити габаритні розміри, зменшити споживану потужність.

В основі нової мікропроцесорної техніки лежить мікропроцесор − функціонально закінчений пристрій обробки інформації, яка зберігається в пам'яті. Поява мікропроцесорів (МП) стала можливою завдяки розвитку інтегральної електроніки. Це дозволило перейти від схем малого і середнього ступеня інтеграції до великих і надвеликих інтегральних мікросхем (ВІС і НВІС) [9,10].

У наш час використовуються однокристальні МП з мікропрограмним управлінням.

Підвищення надійності, швидкодії і функціональної насиченості, а також зменшення габаритів сучасної обчислювальної техніки і споживаної нею електроенергії привели до широкого її використання для управління складними енергетичними об'єктами (електростанціями, високовольтними електричними підстанціями) з великою кількістю сенсорів фізичних величин і керованих агрегатів [11, 13] .

Але системи управління, в разі побудови їх на базі персональних комп'ютерів як найбільш універсальних засобів обчислювальної техніки, мають ряд суттєвих недоліків [6].

По-перше, виникає необхідність оснащення комп'ютерів спеціальними додатковими пристроями, що дозволяють підключати до них технологічне обладнання і відповідним чином перетворювати сигнали, як ті, що надходять на комп'ютер, так і ті, що виробляються комп'ютером для керування технологічними об'єктами.

По-друге, в більшості випадків одного разу налаштований комп'ютер керування конкретним технологічним об'єктом тривалий час не потребує втручання оператора, а тому немає необхідності мати такі компоненти, як дисплей і клавіатура, приводи дисків і самі диски.

По-третє, принцип управління з єдиного центру (від одного комп'ютера) має на увазі наявність загального алгоритму управління, що негативно може позначитися в процесі модернізації окремих складових технологічного обладнання при безперервному технологічному процесі.

Для усунення вищевказаних недоліків застосування комп'ютерів в системах управління технологічним обладнанням було розроблено спеціальний пристрій – мікроконтролер (МК).

Будь-яка машина, здатна автоматично виконувати деякі операції, має у своєму складі керуючий контролер – модуль, що забезпечує логіку роботи пристрою. Контролер – це мозок машини. Природно, чим складніша логіка роботи машини, тим «розумнішим» повинен бути контролер [6].

Технічно контролери реалізуються по-різному. Це може бути механічний пристрій, пневматичний або гідравлічний автомат, релейна або електронна схема, або навіть комп'ютерна програма.

У випадку, коли контролер вбудований в машину масового випуску, вартість його проектування розподілена на велику кількість виробів і мала відносно вартості виготовлення. У випадку машин, що виготовляються в одиничних екземплярах, ситуація протилежна. Вартість проектування контролера домінує відносно вартості його фізичної реалізації.

Контролери, виконані на основі реле або мікросхем з «жорсткою» логікою, неможливо навчити робити іншу роботу без істотної переробки. Очевидно, що таку можливість мають тільки програмовані логічні контролери (ПЛК). Ідея створення ПЛК народилася практично відразу з появою мікропроцесора [6].

Перший мікроконтролер був розроблений в 1976 р. і являв собою велику інтегральну схему (ВІС), виконану на одному кристалі. Мікроконтролер містив як основні елементи системної плати комп'ютера, так і пристроїв її сполучення з технологічним обладнанням: мікропроцесор, тактовий генератор, постійний і оперативний запам'ятовувальні пристрої, порти введення–виведення інформації, таймери, аналого-цифрові і цифро-аналогові перетворювачі, канали широтно-імпульсної модуляції сигналів і т. п.

Мікроконтролери знайшли широке застосування для інтелектуального управління різними об'єктами на транспорті, в машинобудуванні, енергетиці та інших галузях промисловості.

Мікроконтролер можна розглядати як мінікомп'ютер, оснащений периферійними пристроями.

Практичне застосування МК у складі промислових контролерів (програмно-логічних контролерів) на енергетичних підприємствах, в цехових умовах електричних станцій, у закритих, комплектних, відкритих розподільних пристроях пов'язано з підвищеною небезпекою. Помилки в зовнішніх електричних колах контролерів, некоректний розрахунок пристроїв живлення і силових блоків, неякісне заземлення, неправильно виконана система аварійного відключення, відсутність захисту механічних вузлів та інші порушення правил монтажу можуть призвести до важких наслідків. Монтаж МК і сполученого з ним устаткування повинен виконуватися тільки кваліфікованим персоналом, що має відповідні допуски.

Помилки в прикладному програмному забезпеченні ПЛК здатні призводити до втрати синхронізму у роботі механізмів, що може стати причиною їх пошкодження або призвести до травм обслуговуючого

персоналу. Правильно спроектована система повинна містити елементи блокування, що виключають таку можливість [6].

Програмований контролер – це програмно керований дискретний автомат, який має велику кількість входів, що підключені за допомогою сенсорів до об'єкта управління, і велику кількість виходів, підключених до виконавчих пристроїв. ПЛК контролює стан входів і виробляє певну послідовність програмно заданих дій, яка викликає зміну виходів.

ПЛК призначений для роботи в режимі реального часу в умовах промислового середовища і повинен бути доступним для програмування неспеціалістом в галузі інформатики.

До негативних факторів, які залежать від виробничого середовища, в якому експлуатується мікроконтролер, відносяться: температура і вологість, удари і вібрація, корозійно-активне газове середовище, мінеральний і металевий пил, електромагнітні завади. Перераховані фактори, досить характерні для виробничих умов, зумовлюють жорсткі вимоги, що визначають схемотехнічне рішення, елементну і конструктивну базу ПЛК. У процесі серійного виробництва ПЛК обов'язковим є технічний «прогін» готових виробів, під час якого виконуються кліматичні, вібраційні та інші випробування.

ПЛК – це конструктивно закінчений виріб, фізичне виконання якого визначається необхідним ступенем захисту, починаючи від контролерів в легких пластикових корпусах, призначених для монтажу в шафі (ступінь захисту IP20), та до герметичних пристроїв в литих металевих корпусах, призначених для роботи в особливо жорстких умовах.

В ідеальному випадку правильно підібраний за умовами експлуатації контролер не можна пошкодити ззовні без застосування екстремальних методів. Штатними для ПЛК є такі апаратні рішення, як повна гальванічна розв'язка входів-виходів, захист за струмом і напругою, дзеркальні вихідні канали, сторожовий таймер завдань і мікропроцесорного ядра.

Контролери традиційно працюють в нижній ланці *автома-тизованої системи управління енергетичним підприємством* (АСУЕП) – систем, безпосередньо пов'язаних з технологією виробництва (ТВ), транспортування, розподілу і споживання електричної енергії.

Лабораторний практикум присвячений вивченню AVR мікроконтролерів виробництва фірми «ATMEL», застосовуваних в обладнанні енергетичних підприємств України, наприклад, в лічильниках електричної енергії «Енергія 9» в ПАТ «Вінницяобленерго», в промисловому контролері «Логиконт S200» в ПЗЕС.

Ці 8-розрядні RISC-мікроконтролери є, мабуть, найбільш цікавим і прогресивним напрямком, який розвивається фірмою «ATMEL». Мікроконтролери цієї серії являють собою потужний інструмент, прекрасну основу для створення сучасних високопродуктивних і економічних вбудовуваних контролерів багатоцільового призначення.

Популярність мікроконтролерів AVR постійно збільшується. Не останню роль у цьому відіграє співвідношення показників «ціна/ швидкодія/енергоспоживання». Мікроконтролери AVR є одними з кращих на ринку 8-розрядних мікроконтролерів. Крім того, постійно зростає кількість випущених сторонніми виробниками різноманітних програмних і апаратних засобів підтримки розробок пристроїв на їх основі. Все це дозволяє говорити про мікроконтролери AVR як про новий індустріальний стандарт серед 8-розрядних мікроконтролерів загального застосування.

Фірма Atmel була заснована в 1984 р в знаменитій Селіконовій долині (Каліфорнія, США). У середині 90-х років її основною продукцією стали мікросхеми пам'яті і перепрограмовувані МК платформи MCS-51. Порівняно з подібними виробами фірм Intel, Philips, Temic, OKI, Siemens, мікросхеми Atmel були дешевшими, ні в чому не поступаючись їм за якістю. Одна зі складових успіху – створення філій виробництва в країнах Південно-Східної Азії.

Все в МК платформи MCS-51 було добре за винятком енергоспоживання і продуктивності. Там, де використовувалося мало-потужне (батарейне) живлення і була потрібна висока швидкість обробки даних, розробники використовували PIC-контролери фірми Microchip Technologies, МК серії H8/300 фірми Hitachi і МК фірми Dallas Semiconductor.

Ситуація докорінно змінилася в 1996 р., коли було оголошено про початок серійного виробництва принципово нових 8-розрядних контролерів платформи AVR.

У архітектури AVR скандинавський родовід. У 1995 р. два норвезьких винахідники Альф Боген і Вегард Воллей запропонували фірмі Atmel концепцію нового МК. Ідея була прийнята. Базові принципи і система команд розроблялися в норвезькому відділенні фірми Atmel спільно зі шведськими програмістами фірми IAR Systems.

Переваги AVR: швидкодійний RISC-процесор, FLASH – пам’ять з низьковольтною напругою програмування, внутрішнє перезаписування, потужні вихідні порти, широкий діапазон живильної напруги. І все це при малому споживанні струму, високій швидкості, а головне, при низькій вартості.

# 1 Лабораторна робота № 1
# AVR МІКРОКОНТРОЛЕРИ ТА РОБОТА З НИМИ

**Мета роботи:** ознайомитись з історією створення, призначенням, конструкцією, програмуванням мікроконтролерів AVR та з STK-500 - платформою розробки програм для восьмирозрядних RISC AVR МК фірми Atmel.

## 1.1 Короткі теоретичні відомості

### 1.1.1 Мікроконтролери Atmel AVR

Студенти Norwegian University of Science and Technology (NTNU) є авторами ідеї розробки нового прогресивного RISK-ядра (норвезьке місто Trondheim). Звали винахідників Альф Боген (Alf-Egil Bogen) і Вегард Воллен (Vegard Wollen). В місті Bakklandet майбутні директори Atmel Norway створили архітектуру, яка стала одною з найвдаліших на світовому ринку мікроконтролерів.

У 1995 році Боген і Воллен вирішили запропонувати американській корпорації Atmel випускати новий 8-бітовий RISC-мікроконтролер і забезпечити його Flash-пам'яттю програм на кристалі. Ідея настільки сподобалася керівництву Atmel Corp., що було ухвалено рішення негайно інвестувати даний проект.

У 1996 році був заснований дослідницький центр Atmel в Тронхеймі. Варто сказати, що 150-тисячний Тронхейм зусиллями свого університету щороку породжує до 20-ти нових компаній, які спеціалізуються в секторах ринку, починаючи з автоматизації і закінчуючи передаванням та обробленням даних. В кінці 1996 року був випущений дослідний кристал AT90S1200, а в другій половині 1997-го корпорація Atmel почала серійне виробництво нового сімейства мікроконтролерів, їх рекламну і технічну підтримку.

Нове ядро було запатентовано і одержало назву AVR, яка після вже декількох років стала трактуватися найрізноманітнішими способами. Хтось стверджує, що це не інакше як Advanced Virtual RISC, інші вважають, що не обійшлося тут без Alf Egil Bogen Vegard Wollan RISC. Утримувачами патенту при цьому є: Bogen, Alf-Egil (NO); Wollan, Vegard (NO); Myklebust, Gaute (NO); Bryant, John, D. (US).

Цікаво, що система команд і внутрішня будова чіпів AVR розроблялися спільно з фірмою IAR Systems – виробником компіляторів мов програмування C/C++, що забезпечило унікальні характеристики цих мікроконтролерів. В результаті для AVR стало можливим одержувати високу щільність коду при використанні мов високого рівня, практично не втрачаючи в продуктивності порівняно з програмами, написаними низькорівневою мовою Асемблера [1,2].

Крім того, використання прогресивної технології конвеєризації у AVR скорочувало цикл «вибірка – виконання» команди. Наприклад, у мікроконтролерів сімейства x51 коротка команда виконується за 12 тактів генератора. У PIC-контролерах фірми Microchip, де вже реалізований конвеєр, коротка команда виконується за 4 періоди тактової частоти. У мікроконтролерах AVR коротка команда в загальному потоці виконувалася всього за один період тактувального сигналу. Така побудова кристала забезпечила істотне підвищення продуктивності, яка стала досягати значення 1 MIPS на 1 МГц. Це у багатьох випадках, при заданій продуктивності, дозволяло зменшити тактову частоту, а тому і споживану потужність пристрою. AVR-мікроконтролери надавали ширші можливості з оптимізації параметрів продуктивності/енергоспоживання. Це особливо важливо при розробці пристроїв з батарейним живленням [1,2].

Звичайно ж, нові мікроконтролери від Atmel були зустрінуті з великою зацікавленістю. Їх продаж неухильно зростав, команда AVR, що складалась в 1997 році не більше, ніж з 10 чоловік, зараз перевищує сотню співробітників тільки в Норвегії, без урахування технічних фахівців з AVR в двох спеціалізованих центрах у Франції і Фінляндії.

Галузі застосування AVR багатогранні. Для «tiny» AVR це інтелектуальні автомобільні сенсори різного призначення, іграшки, ігрові приставки, материнські плати персональних комп'ютерів, контролери захисту доступу в мобільних телефонах, зарядні пристрої, детектори диму і полум'я, побутова техніка, різноманітні інфрачервоні пульти дистанційного керування. Для «classic» AVR (серія AT90) це модеми різних типів, сучасні зарядні пристрої, вироби класу Smart Cards і пристрої зчитування для них, супутникові навігаційні системи для визначення місцеположення автомобілів на трасі, складна побутова техніка, пульти дистанційного керування, мережеві карти, материнські плати комп'ютерів, стільникові телефони нового покоління, а також різноманітні промислові системи контролю і управління. Для «mega» AVR це аналогові (NMT, ETACS, AMPS) і цифрові (GSM, CDMA) мобільні телефони, принтери і контролери для них, контролери апаратів факсимільного зв'язку і ксероксів, контролери сучасних дискових накопичувачів, CD-ROM і т. п.

Фахівці вже оцінили високу швидкість роботи і потужну систему команд AVR, наявність двох типів незалежної пам'яті на одному кристалі і периферії, що активно розвивається. Важливу роль в цьому відіграє політика Atmel Corp. у питанні розвитку і розповсюдження різноманітних, доступних засобів підтримки розробок. Це дозволяє розробникам і виробникам електронної техніки сподіватися на збереження повноцінної підтримки для перспективного сімейства мікроконтролерів, закладаючи AVR в свої нові вироби [3,4].

AVR – це відносно молодий продукт корпорації Atmel, лінія вбудовуваних багатоцільових мікроконтролерів загального призначення, що активно розвивається. У цій лінії постійно з'являються нові сімейства і

кристали, оновлюються версії вже існуючих мікросхем, удосконалюється і розширюється програмне забезпечення підтримки.

Atmel виготовляє два сімейства мікроконтролерів з ядром AVR: Tiny і Mega. Мікроконтролери Classic (серії AT90), першого з сімейств AVR, поступово замінюються сучаснішими моделями. У 2003 році Atmel Corp. випустила 500 мільйонів штук мікросхем з ядром AVR.

#### 1.1.1.1 8-бітовий RISC FLASH-мікроконтролер

Мікроконтролер сімейства AVR фірми *Atmel* є восьмирозрядною однокристальною мікро-ЕОМ зі спрощеною (скороченою) системою команд — RISC *(Restricted (Reduced) Instruction Set Computer).*

Більшість команд, які входять в систему команд, вибираються з пам'яті за один такт і виконуються за один такт роботи мікроконтролера. При виконанні послідовності таких команд вибірка з пам'яті чергової команди поєднується в часі з виконанням раніше вибраної команди. При цьому кількість команд, які виконуються за одну секунду, збігається з тактовою частотою роботи мікроконтролера [5,6].

Мікроконтролери виготовляються за високоякісною КМОН (CMOS) технологією. Вони містять пристрої для зберігання програми і даних. Пам’ять цих МК виконана за Flash і EEPROM технологіями та відрізняється низьким енергоспоживанням при високій тактовій частоті. Запис програми і початкових даних в пам'ять може виконуватися після установлення мікроконтролера в апаратурі, де йому належить працювати.

Мікроконтролери AVR розроблені фірмою Atmel і мають такі основні характеристики:

– дуже швидка гарвардська RISC-архітектура завантаження і виконання більшості інструкцій протягом одного циклу тактового генератора. При цьому досягається швидкість роботи приблизно 1 MIPS на 1 МГц. Частота тактового генератора багатьох типів мікроконтролерів AVR може досягати 10 ... 16 МГц (10 ... 16 MIPS) (*MIPS* – Millions Instructions per Second – мільйонів операцій в секунду). Відсутнє внутрішнє ділення частоти, як, наприклад, в мікроконтролерах PIC. Таким чином, якщо використовується кварцовий резонатор з частотою 16 МГц, то мікроконтролер працює зі швидкодією майже 16 MIPS;

– програми розташовуються в електрично перепрограмованій постійній пам'яті програм FLASH ROM. Ця пам'ять може бути перепрограмована до 1000 разів. Це полегшує налаштування систем. Крім того, можливість внутрішньо-схемного програмування дозволяє не виймати мікроконтролер з плати пристрою (де цей контролер встановлений) в процесі програмування. Це значно прискорює процес розробки систем на основі AVR мікроконтролерів;

– система команд мікроконтролерів AVR спочатку проектувалася з урахуванням особливостей мови програмування високого рівня **C.** Це дозволяло одержувати після компіляції програм набагато ефективніший код, ніж для інших мікроконтролерів. А це вже виграш і у розмірі

одержаного коду (в обсязі пам'яті на кристалі), і в швидкості роботи мікроконтролера;

– мікроконтролери AVR мають 32 регістри, всі з яких безпосередньо працюють з АЛП (арифметичний логічний пристрій). Це значно зменшує розмір програм. У інших мікроконтролерах, як правило, для здійснення, наприклад, додавання, один з операндів обов'язково повинен знаходитися в спеціальному регістрі — акумуляторі. Таким чином, необхідно спочатку операнд занести в акумулятор. Потім, після виконання операції, результат з акумулятора потрібно переписати в регістр для зберігання результату. Разом виходить вже три команди. У мікроконтролерах AVR те ж саме займе лише одну команду;

– дуже невелике споживання енергії і наявність декількох режимів роботи із зниженим споживанням енергії робить ці мікроконтролери ідеальними для застосування в конструкціях, які отримують живлення від батарей;

– наявність дешевих і простих у використанні програмних засобів. Багато повноцінних програм доступні у вільно поширюваному варіанті, як, наприклад, програма AVR Studio (для налаштування кодів програм), асемблер Wavrasm, безліч версій програматорів і навіть компілятор мови С — avr gcc;

– вузли PWM (широтно-імпульсна модуляція), таймери/лічильники, аналоговий компаратор і послідовний порт UART розташовані в мікроконтролері. Ними можна керувати за допомогою переривань, що значно спрощує роботу;

– є умовні команди переходів і відгалужень;

– відсутня необхідність перемикати сторінки пам'яті (на відміну, наприклад, від мікроконтролерів PIC);

– всі мікроконтролери AVR мають електричноперепрограмовану постійну пам'ять даних EEPROM, яка може бути перепрограмована більше 1’000’000 раз!

Мікроконтролери AVR можна поділити на три серії:

1. Tiny AVR (ATtini) – недорогі мініатюрні мікроконтролери в 8-вивідному виконанні;

2. Classic AVR (серія AT90) — основна лінія мікроконтролерів з продуктивністю окремих модифікацій до 16 MIPS, FLASH-пам'яттю програм 2...8 Кб, пам'яттю даних EEPROM 64...512 байт, оперативною пам'яттю даних SRAM 128...512 байт;

3. Mega AVR (ATmega) – з продуктивністю 4 ... 16 MIPS для складних додатків, що потребує великого обсягу пам'яті, FLASH-пам'яттю програм до 128 кБ, пам'яттю даних EEPROM 64...512 байт, оперативною пам'яттю даних SRAM 2...4 кБ, вбудованим 10-розрядним 8-канальним ***АЦП***, апаратним помножувачем $8 \times 8$.

У кожну серію входять мікроконтролери декількох типів. Мікроконтролери серії AT90 за своїми структурними характеристиками (обсяг пам'яті, склад периферійних пристроїв) близькі до мікроконтролерів сімейств AT89 фірми *Atmel* і MCS-51 фірми *Intel.* За своїми

обчислювальними можливостями вони займають середнє положення між мікроконтролерами серій ATtiny і ATmega. Мікроконтролери серії ATtiny мають найменші, а мікроконтролери серії ATmega найбільші обчислювальні можливості в сімействі AVR.

Цікавою особливістю сімейства мікроконтролерів AVR є те, що система команд всього сімейства сумісна при перенесенні програми із слабкого на потужніший мікроконтролер.

У табл. 1.1 наведено повні і скорочені позначення типу мікроконтролера (МК), кількість виводів з корпусу мікроконтролера *NB*, кількість команд в системі команд *NK*, максимальне значення тактової частоти $F_{max}$ і струм споживання *I* мікроконтролерів різних типів сімейства AVR.

Струм споживання залежить від напруги живлення, тактової частоти і температури навколишнього середовища. У таблиці 1.1 вказано значення струму споживання при напрузі живлення +3 В, тактовій частоті 4 МГц і температурі навколишнього середовища +25 °С.

Таблиця 1.1 – Приклади позначення AVR контролерів

| Тип (позначення) МК | | NB | NK | ***F**** МГц | ***I*** мА |
|---|---|---|---|---|---|
| повне | скорочене | | | | |
| ATtiny 11 | t11 | 8 | 90 | 6 | 2,2 |
| ATtiny12 | t12 | 8 | 90 | 8 | 2,2 |
| ATtinyl 5 | t15 | 8 | 90 | 1,6 | 3,0* |
| AT90S2323 | 2323 | 8 | 118 | 10 | 2,4 |
| AT90S2343 | 2343 | 8 | 118 | 10 | 2,4 |
| AT90S1200 | 1200 | 20 | 89 | 12 | 2,0 |
| AT90S2313 | 2313 | 20 | 118 | 10 | 2,8 |
| ATtiny28 | t28 | 28 | 90 | 4 | 3,0 |
| AT90S4433 | 4433 | 28 | 118 | 8 | 3,4 |
| **AT90S8515** | 8515 | 40 | 118 | 8 | 3,0 |
| AT90S8535 | 8535 | 40 | 118 | 8 | 6,4 |
| ATmega163 | m163 | 40 | 130 | 8 | 5,0 |
| ATmega 103 | m103 | 64 | 121 | 6 | 5,5 |
| Atmega8515 | m8515 | 40 | 130 | 16 | 5,5 |

Примітка. * – при $\boldsymbol{F}_{max}$

Мікроконтролери одного типу випускаються в декількох варіантах допустимих значень напруги живлення, а ще вони відрізняються максимальними допустимими значеннями тактової частоти, типами корпусів і діапазонами допустимих значень температури навколишнього середовища [2, 5].

### 1.1.1.2 Структура мікроконтролера

Мікроконтролери сімейства AVR мають єдину базову структуру. Узагальнена структурна схема мікроконтролера зображена на рис. 1.1.

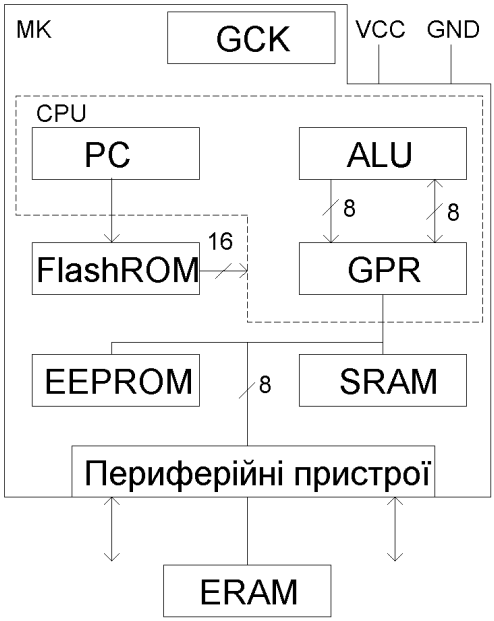



Рисунок 1.1 – Узагальнена структурна схема AVR мікроконтролера

До складу мікроконтролера входять:

- генератор тактового сигналу (GCK);
- процесор (CPU);
- постійний запам'ятовувальний пристрій (ПЗП) для зберігання програми, виконаний за технологією Flash (FlashROM);
- оперативний запам'ятовувальний пристрій статичного типу для зберігання даних (SRAM);
- постійний запам'ятовувальний пристрій для зберігання даних, виконаний за технологією EEPROM ;
- набір периферійних пристроїв для введення і виведення даних та сигналів управління і виконання інших функцій.

Виводи VCC і GND призначені для підключення позитивного та негативного полюсів джерела напруги живлення мікроконтролера. Рівень напруги всіх сигналів в мікроконтролері показують і вимірюють відносно рівня напруги на шині GND, яку приймають за 0 В;

- у мікроконтролерах типу t11, t12, t15, 1200 і t28 пристрій пам’яті SRAM відсутній. У мікроконтролерах типу t11 і t28 відсутній пристрій пам’яті EEPROM [4,5].

До мікроконтролерів типу 8515 і m103 може бути приєднано пристрій зовнішньої пам’яті для зберігання даних (ERAM). Команди коду програми зберігаються лише у внутрішній пам’яті FlashROM.

Таблиця 1.2 – Характеристики AVR мікроконтролерів

| Тип МК | Flash (байт) | ISP | SRAM (байт) | EEPROM (байт) | ERAM | IOR | P | I/O | ALT | SPI | UART | TWSI | T/CO | T/C1 | T/C2 | ADC | AC | PHM | IU | Тип МК |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t11 | 1K | | | | | 14 | 1 | 6 | 6 | | | | A | | | | + | | 5 | t11 |
| t12 | 1K | + | | 64 | | 18 | 1 | 6 | 6 | | | | A | | | | + | | 6 | t12 |
| t15 | 1K | + | | 64 | | 27 | 1 | 6 | 6 | | | | A | в | | 4 | + | | 9 | t15 |
| 2323 | 2K | + | 128 | 128 | | 17 | 1 | 3 | 2 | | | | A | | | | | | 3 | 2323 |
| 2343 | 2K | + | 128 | 128 | | 17 | 1 | 5 | 3 | | | | A | | | | | | 3 | 2343 |
| 1200 | 1K | + | | 64 | | 18 | 2 | 15 | 4 | | | | A | | | | + | | 4 | 1200 |
| t28 | 2K | | | | | 17 | 3 | 20 | 6 | | | | A | | | | + | + | 6 | t28 |
| 4433 | 4K | + | 128 | 256 | | 44 | 3 | 20 | 20 | + | + | | A | D | | 6 | + | | 14 | 4433 |
| 8515 | 8K | + | 512 | 512 | + | 45 | 4 | 32 | 31 | + | + | | A | E | | | + | | 13 | 8515 |
| 8535 | 8K | + | 512 | 512 | | 54 | 4 | 32 | 26 | + | + | | A | E | C | 8 | + | | 17 | 8535 |
| m163 | 16K | + | 1024 | 512 | | 63 | 4 | 32 | 28 | + | + | + | A | E | с | 8 | + | | 18 | m163 |
| m103 | 128K | + | 4000 | 4K | + | 60 | 6 | 48 | 47 | + | + | | C | E | в | 8 | + | | 24 | m103 |
| m8515 | 8K | + | 512 | 512 | + | 40 | 5 | 35 | | + | + | | C | B | | | + | | | m8515 |

До складу процесора (CPU) входять:

- лічильник команд (PC);
- арифметико-логічний пристрій (ALU);
- блок регістрів загального призначення (GPR, General Purpose Registers) і інші елементи, не показані на схемі рис. 1.1.

Окрім регістрів загального призначення в мікроконтролері є регістри спеціальних функцій, які в сімействі AVR називаються регістрами введення − виведення (I/O Registers, IOR). За участю цих регістрів здійснюються:

- управління роботою мікроконтролера та окремих його пристроїв;
- визначення стану мікроконтролера і окремих його пристроїв;
- введення даних в мікроконтролер і в окремі його пристрої, виведення даних;
- виконуються інші дії.

Для нумерації регістрів введення−виведення використовуються номери від 0 до 63. Номери записують у шістнадцятковому коді. Наприклад: від **$**00 до **$**3F або від **0x**00 до **0x**3F, де **$** або **0x** − покажчики шістнадцяткового коду. Кожному регістру надане ім'я, пов'язане з виконуваною цим регістром функцією. Мікроконтролери різних типів мають різний склад регістрів введення−виведення, при цьому регістри з однаковими номерами можуть мати різні імена. Кількість регістрів введення−виведення у мікроконтролерів різних типів наведено у табл. 1.2, в колонці IOR. Робота деяких пристроїв мікроконтролера залежить від стану додаткових однобітових елементів пам’яті − бітів конфігурації ***(Fuse Bits)***. Початкові значення настановних бітів записуються на підприємстві, яке виготовляє МК. Значення настановного біта може бути змінено тільки при програмуванні мікроконтролера.

### 1.1.1.3 Генератор тактового сигналу

Мікроконтролери сімейства AVR є пристроями синхронного типу. Дії, які виконуються в мікроконтролері, «прив'язані» до імпульсів тактового сигналу. Мікроконтролери мають повністю статичну структуру і можуть працювати при тактовій частоті від 0 Гц. Як генератор тактового сигналу (GCK) використовуються:

- внутрішній генератор з зовнішнім кварцовим або керамічним резонатором (XTAL);
- внутрішній RC-генератор (IRC);
- внутрішній генератор із зовнішнім RC-колом (ERC);
- зовнішній генератор (EXT).

У мікроконтролерів, які мають внутрішній генератор із зовнішнім резонатором (XTAL), резонатор приєднується до виводів XTAL1 і XTAL2, які через конденсатори малої ємності (20−30 пФ) приєднуються до шини GND. Тактова частота визначається робочою частотою резонатора. У

мікроконтролера типу t28 при нульовому значенні настановного біта INTCAP підключення виводів XTAL1 і XTAL2 до шини GND виконується через внутрішні конденсатори ємністю 50 пФ.

#### 1.1.1.4 Процесор

Процесор (CPU) формує адресу чергової команди, вибирає команду з пам'яті і організовує її виконання. Код команди має формат «слово» (16 біт) або «два слова».

До складу процесора окрім лічильника команд (PC), арифметико-логічного пристрою (ALU) і блоку регістрів загального призначення (GPR), входять:

- регістр стану мікроконтролера SREG;
- регістр-покажчик стека SP або SPL і SPH та інші елементи.

У лічильнику команд адреса чергової команди формується шляхом додавання 1 до числа, код якого зберігається в лічильнику команд. При пуску і перезапуску мікроконтролера в лічильник команд заноситься код числа 0 і перша команда вибирається з FlashROM за адресою 0.

У арифметико-логічному пристрої виконуються арифметичні і логічні операції. Операнди надходять з регістрів загального призначення (GPR). При виконанні простих операцій результат записується в регістр, з якого надійшов операнд. При виконанні складних операцій результат записується в регістр, з якого надійшов перший операнд.

Ми називаємо AVR 8-бітовим мікроконтролером. Це означає, що він оперує 8-бітовими числами. Двійкове число 11111111 є найбільшим 8-бітовим числом і дорівнює десятковому $255_{10}$ і шістнадцятковому $FF_{16}$. Для позначення конкретної системи числення в програмах використовуються різні способи запису (адже десяткове число $11111111_{10}$ значно відрізняється від двійкового числа 111111111). Двійкові числа записуються у вигляді 0b00101000 (тобто 0b...). Десяткова система числення використовується за замовчуванням і додаткових позначень не потребує, а шістнадцяткові числа починаються з символів 0x або знака грошової одиниці $ (0x3A, або $3A). Отже число 0b00101011 дорівнює числу 43 (десятковому), яке дорівнює числу 0x2B.

При роботі зі входами і виходами мікроконтролерів AVR використовують двійкову систему числення, при цьому кожен вхідний або вихідний вивід відповідає конкретному біту. Біт, встановлений в 1, відповідає стану, який називають логічною одиницею. Це означає, що напруга на виводі мікроконтролера дорівнює напрузі живлення (наприклад, +5 В). Біт, скинутий в 0, відповідає стану логічного нуля або 0 В. Для вхідних сигналів порогом спрацювання між станами логічного 0 і логічної 1 є половина напруги живлення (наприклад, +2,5 В).

### 1.1.1.5 Регістри

Регістри загального призначення

Одним з найбільш важливих аспектів програмування AVR і мікроконтролерів взагалі є регістри. Щоб було зрозуміліше, уявіть собі, що в мікроконтролері AVR є шафа з великою кількістю ящиків, в кожному з яких зберігається 8-бітове число (один байт). Ці ящики і є регістрами — точніше, ми називаємо їх регістрами введення-виведення (РВВ). Окрім цих регістрів введення/виведення, у нас є 32 регістри загального призначення. Вони відрізняються від регістрів введення−виведення, оскільки не є частиною шафи. Уявіть собі, що робочі регістри є службовцями, а ви — їхнім начальником. Якщо ви хочете покласти що-небудь в шафу, ви віддаєте це службовцю і наказуєте йому покласти це в шафу. Так само програміст не може помістити число безпосередньо в регістр введення−виведення. Натомість він повинен записати число в регістр загального призначення, а потім скопіювати регістр загального призначення в регістр введення/виведення. Ви можете також попросити службовців виконати яку−небудь операцію над числами, що є у них, тобто ви можете додавати числа, що знаходяться в робочих регістрах.

З таблиці 1.3 видно, що кожному регістру відповідає унікальний номер.

Таблиця 1.3 – Регістри моделі ATmega8515

| | |
|---|---|
| $3F | SREG |
| $10 | PIND |
| $11 | DDRD |
| $12 | POBTD |
| $16 | PINB |
| $17 | DDRB |
| $18 | PORTD |
| $1C | EECR |
| $1D | EEDR |
| $1E | EEARL |
| $21 | WDTCR |
| $32 | TCNT0 |
| $33 | TCCR0 |
| $35 | MCUCR |
| $38 | TIFR |
| $39 | TIMSK |
| $3B | GICR |

Регістри загального призначення позначаються як R0, R1, …, R31. Відмітимо, що регістри R30 і R31 трохи відрізняються від інших. Вони утворюють здвоєний регістр Z – регістр, який може містити 16-бітове значення (зване словом).

До цих регістрів можна звертатися окремо, як до регістрів ZL і ZH, але можна і об'єднати таким чином, що ZL (lower Z — молодший) міститиме біти 0...7 16-бітового числа, а ZH (higher Z — старший) — біти 8... 15.

Відмітимо, що таке об'єднання використовується тільки в деяких командах.

Блок регістрів загального призначення – GPR

Він містить 32 восьмирозрядних регістри, яким надані імена R0, R1, …, R31. У деяких операціях в ALU можуть брати участь лише регістри зі старшими номерами (від R16 до R31). Регістри з іменами від R24 до R31 можуть утворювати пари, використовувані для зберігання слів, при цьому регістр з парним номером зберігає молодший байт, а регістр з непарним номером — старший байт.

Для зберігання адрес, при зверненнях до пам'яті з непрямою адресацією, використовуються такі пари регістрів:

- R26, R27 (їм надане ім'я X),
- R28, R29 - ім'я Y,
- R30, R31 - ім'я Z.

Регістр стану мікроконтролера SREG ($3F).

Він містить вісім розрядів (SREG.7, …, SREG.6, …, SREG.0).

Розряд SREG, 7 (I) використовується для дозволу/заборони переривань. При I = 0 всі переривання заборонені. При I=1 будь-яке переривання може бути дозволено.

Розряд SREG.6 (T) використовується для зберігання біта при виконанні операцій з бітами. Решта розрядів регістра SREG використовується для зберігання ознак результатів арифметичних і логічних операцій, що виконуються в ALU [1].

Регістр-покажчик стека SP($3D)

Він зберігає і формує адресу при зверненні до стека типу LIFO. У мікроконтролерах типу t1l, t **12**, t **15**, t 200 і t28 як стек використовується пристрій пам’яті (апаратний стек). Цей стек використовується тільки для зберігання адрес повернення при перериваннях і зверненнях до підпрограм. У системі команд відсутні команди звернення до стеку.

У мікроконтролерах інших типів як стек використовується область, яка виділяється користувачам, в SRAM. У системі команд є команди для звернення до стека. Запис байтів в стек виконується у порядку зменшення адрес в SRAM. При пуску і перезапуску мікроконтролера в регістр-покажчик стека заноситься код числа 0. Для нормальної роботи стека в регістр-покажчик необхідно занести іншу початкову адресу. У мікроконтролерах з великою місткістю SRAM регістр-покажчик складається з двох регістрів:

- **SPL. L** – від слова low (**$**3D);
- **SPH. H** – від слова high (**$**3E).

### 1.1.2 Опис мікроконтролера ATmega8515 фірми Atmel

ATmegaS8515 – сучасний 8-бітовий КМОП – мікроконтролер. Він має продуктивність близько 1 MIPS на мегагерц за рахунок того, що майже всі команди він виконує за один період тактового генератора.

Мікроконтролери сімейства AVR побудовані на основі розширеної RISC-архітектури, об'єднуючої розвинений набір команд і 32 регістри загального призначення. Всі 32 регістри безпосередньо приєднані до арифметико-логічного пристрою, якій надає доступ до будь-яких двох регістрів протягом одного машинного циклу. Подібна архітектура забезпечує майже десятиразовий виграш в продуктивності порівняно з традиційними мікроконтролерами, наприклад, серії 8051[2].

Мікроконтролер ATmegaS8515 має такі характеристики:

- 8 кБ завантажуваної флеш-пам'яті;
- 512 байтів EEPROM;
- 35 ліній введення – виведення загального призначення;
- 32 восьмирозрядні регістри загального призначення;
- два таймери/лічильники: один 8-розрядний, другий 16-розрядний;
- зовнішні і внутрішні переривання;
- вбудований послідовний порт (UART);
- програмований сторожовий таймер з вбудованим генератором;
- послідовний порт ISP для завантаження програм;
- послідовний порт SPI;
- три канали ШІМ з програмним керуванням;
- внутрішній RC генератор з налаштуванням частоти програмним чином;
- три режими низького енергоспоживання, які налаштовуються програмно.

Флеш-пам'ять на кристалі може бути перепрограмована при встановленні мікроконтролера прямо на платі через послідовний інтерфейс **ISP**.

#### 1.1.2.1 Опис виводів мікроконтролера ATmegaS8515

**VCC** – виводи джерела живлення.

**GND** – загальний вивід («земля»).

**PA7...PA0 –** це виводи порту A (PORT A). Цей порт є восьмирозрядним двонаправленим портом введення−виведення. При відповідному налаштуванні порту він може використовуватися як порт адреси чи даних при підключенні зовнішньої пам’яті.

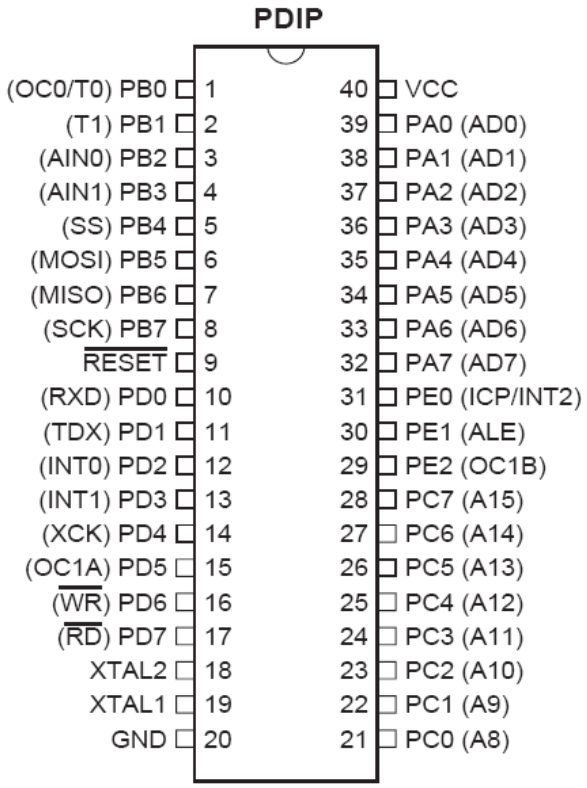



Рисунок 1.2 – Виводи мікроконтролера ATmega8515

**PB7...PB0** – це виводи порту **PORT B** (порт B). Цей порт є 8-бітовим двонаправленим паралельним портом введення/виведення з вбудованими підтягувальними резисторами. У виводів PB7...PB0 передбачені внутрішні підтягувальні резистори (їх можна вмикати або вимикати для кожного біта окремо). Виводи PB0 і PB1 є як позитивним (AIN0), так і негативним (AIN1) входами вбудованого аналогового компаратора. Вихідні буфери порту B можуть споживати струм 20 мА і безпосередньо керувати світлодіодами. Це означає, що мікроконтролер здатен керувати навантаженням до 20 мА. Таким чином, для керування світлодіодом його слід приєднувати одним виводом до виводу порту мікроконтролера, а іншим — до напруги живлення +VCC. Тому, якщо світиться світлодіод (а значить, і споживається струм), то це буде при значенні 0 на відповідній лінії порту.

**PD6...PD0** є виводами порту **PORT D (**порт D). Цей порт є 7-бітовим двонаправленим паралельним портом введення−виведення з вбудованими підтягувальними резисторами. Вихідні буфери порту D також можуть споживати струм 20 мА. Якщо входи встановлені в низький стан, то виводи порту D будуть джерелами струму, якщо будуть задіяні підтягувальні резистори.

**PC6...PC0 –** є восьмирозрядним, двонаправленим портом. Даний порт також може використовуватися, за відповідного налаштування, як порт адреси для зовнішньої пам’яті.

**PE6...PE0 –** трирозрядний, двонаправлений порт. Виводи даного порту можуть бути використані як вхід зовнішнього переривання, вихід таймера лічильника, дозвіл адреси, при роботі із зовнішньою пам’яттю, і як вхід таймера лічильника 1.

**RESET** — вхід скидання. Утримання на вході низького рівня протягом двох машинних циклів (якщо працює тактовий генератор) перезапускає мікроконтролер.

**XTAL1** — вхід інвертувального підсилювача генератора і вхід зовнішнього тактового сигналу.

**XTAL2** — вихід інвертувального підсилювача генератора.

### 1.1.3 Процес розробки програм складається з п'яти основних етапів:

1. Вибір конкретного мікроконтролера і складання блок-схеми програми.
2. Написання програми (AVR Studio або будь-якої іншої відповідної програми).
3. Асемблювання програми (перетворить написаний вами текст у форму, зрозумілу мікроконтролеру).
4. Симуляція або Емуляція програми, щоб переконатися в її роботоздатності (або нероботоздатності).
5. Програмування AVR. На цьому етапі написане вами заноситься в реальний мікроконтролер.

А зараз розглянемо деякі з цих етапів детальніше.

#### 1.1.3.1 Вибір моделі

Оскільки в сімейство AVR входить велика кількість різних моделей мікроконтролерів, необхідно гарненько подумати про те, яка з них краще найкраще підійде для вашого пристрою. Деяку інформацію про мікроконтролер можна одержати з його позначення:

ATmega8515 – Код обсягу ОЗП; 0 – ОЗП відсутній;

- Код обсягу ОЗП; – ОЗП 512 байтів;
- № моделі ЦПУ – Про;
- Код обсягу EEPROM; 5 – 512 байтів;
- 8 Кбайт FLASH-пам'яті програм.

Коди обсягу пам'яті:

Коди 0123456789AB

Обсяг [байт] 0 32 64 128 256 512 1К 2К 4К 8К 16К 32К

#### 1.1.3.2 Блок-схема алгоритму

У принципі, на цьому етапі формується основа програми, а написати програму, маючи перед собою блок-схему, набагато простіше, ніж без неї.

Блок-схема (рис. 1.3) повинна відображати основні етапи функціонування мікроконтролера, а також прояснювати структуру програми. Уявіть, що ваша програма є лабіринтом. В цьому випадку блок-схема буде картою, що позначає основні ділянки лабіринту. Маючи карту, вже легше вийти з лабіринту, так само і написання програм. Також при написанні програм і відповідно алгоритмів потрібно враховувати те, що програма не має мати кінця. Тобто програма має виконуватися у нескінченному циклі або після виконання певних дій очікувати переривань, які переведуть виконання програми на новий цикл.

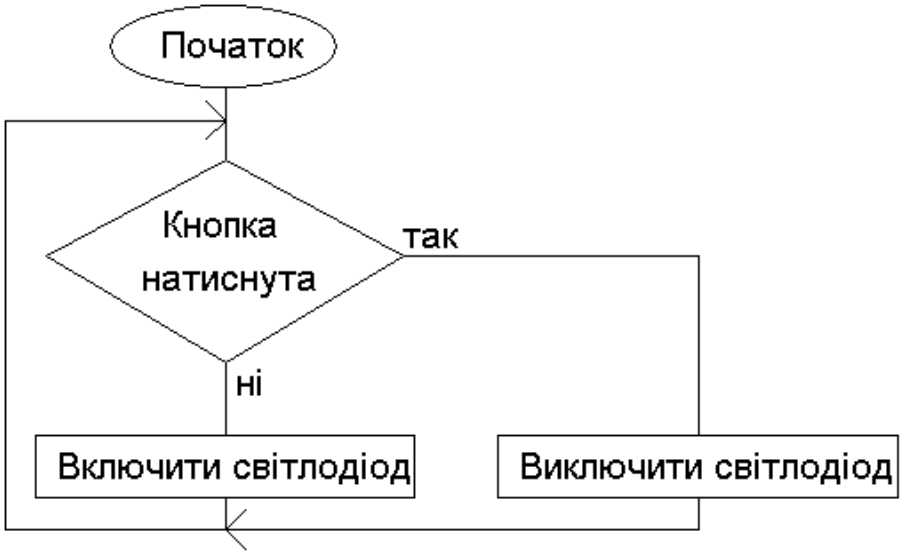



Рисунок 1.3 – Блок-схема програми, що включає і виключає світлодіоди

Ідея блок-схеми полягає у виділенні основних етапів виконання програми, а також в створенні діаграми, яку могла б зрозуміти будь-яка людина, навіть абсолютно не знайома з програмуванням. Надалі ви зрозумієте, що набагато легше писати програму на основі блок-схеми, оскільки в цьому випадку можна займатися кожним блоком окремо, абсолютно не замислюючись про загальну структуру програми.

### 1.1.3.3 Написання програми

Наступним етапом після розробки блок-схеми є завантаження шаблону і написання на його основі своєї програми. Це можна зробити за допомогою будь-якого простого текстового редактора або спеціалізованого середовища розробки, такого як AVR Studio.

### 1.1.3.4 Асемблювання

Щоб написану вами програму можна було записати в мікросхему, її необхідно асемблювати. Дана операція перетворить текст програми в послідовність чисел, яка може бути поміщена в FLASH-пам'ять програм мікроконтролера. В результаті асемблювання отримаємо файл з розширенням .hex і таким самим ім'ям, як і у файлу кода програми. Під час асемблювання асемблер перевіряє вашу програму рядок за рядком і намагається перетворити кожен рядок на відповідний шістнадцятковий код. Якщо він не може розпізнати, що написано в якому-небудь рядку, він реєструє в цьому рядку помилку (error). Помилкою є те місце в програмі, яке асемблер однозначно вважає за неправильне, тобто він не може зрозуміти, що там написано. Також асемблер може згенерувати попередження (warning), якщо зустрілося щось, що можливо є неправильним, тобто написане виглядає незвично, але не обов'язково неправильно.

### 1.1.3.5 Команди

Повний список команд кожного МК складається з близько 130 команд, в даному лабораторному практикумі всі не використовуються, тому розглянемо тільки використовувані.

Всі команди можна поділити на такі групи: логічні, арифметичні, галуження, передачі даних, бітові і команди контролю процесора.

Команди записуються кожна з нового рядка. Якщо команда має операнди то вони записуються відразу за командою відповідно до синтаксису команди.

Таблиця 1.4 – Команди мікроконтролера ATmega**8515**

| Команда | Опис | Синтаксис | Стан регістрів до (після) виконання команди | |
|---|---|---|---|---|
| | | | до | після |
| add | Додавання двох регістрів без переносу, і запис результату в перший регістр | Add r16,r17 | R16 -0x02<br>R17-0x06 | R16 -0x08<br>R17 -0x06 |
| icn | Збільшити вміст регістра на одиницю | inc r20 | R20- 0xAA | R20- 0xAB |
| dec | Зменшити вміст регістра на одиницю | dec r23 | R23 -0xFB | R23-0xFA |
| ldi | Завантажити константу в регістр (команда працює тільки з регістрами R16…R31) | Ldi r24,0x64<br>або<br>Ldi r26,0b01010110 | R24 – будь що<br>або<br>R26 – будь що | R24 – 0x64<br>або<br>R26-0b01010110 |

Продовження таблиці 1.4

| Команда | Опис | Синтаксис | Стан регістрів до(після) виконання команди | |
|---|---|---|---|---|
| | | | до | після |
| out | Виведення даних із регістра загального призначення в регістр порту | out DDRB,r21 | DDRB –будь що<br>R21 – 0x34 | DDRB – 0x34<br>R21 – 0x34 |
| in | Завантаження вмісту регістра введення/виведення в регістр загально призначення | In r18,PORTA | PORTA – 0xAC<br>R18 – будь що | PORTA – 0xAC<br>R18 – 0xAC |
| rjmp | Команда безумовного переходу до відповідної частини програми, яка позначена міткою (наприклад LOOP:) | rjmp LOOP | | |
| push | Збереження вмісту регістра в стек даних | push r26 | R26 – 0x34<br>Стек – будь що | R26 – 0x34<br>Стек – 0x34 |
| pop | Зчитування вмісту стека даних в регістр | pop r30 | R30 – будь що<br>Стек – 0x56 | R30 – 0x56<br>Стек – 0x56 |
| brne | Перейти за міткою, якщо вміст регістра не відповідає константі | brne loop | | |
| breq | Перейти за міткою, якщо вміст регістра дорівнює константі | breq start | | |
| cpi | Порівняння вмісту регістра з константою | cpi r30,0x23 | R30 - 0x23 | Результат порівняння 0 |

В лабораторних роботах курсу «Мікропроцесорна техніка» використовуються команди, наведені в таблиці 1.4.

В коді програм також зустрічається команда; include ця команда дозволяє додати у код програми файл. Синтаксис цієї команди: .include «C: \Program Files\Atmel\AVR Studio\Appnotes\m8515def.inc». Після виконання даної команди в код програми буде доданий текст файлу, в даному разі файл m8515def.inc.

При написанні коду програм для полегшення подальшого аналізу програми доцільно писати коментарі. Коментарі пишуться в рядку, що починається символом крапка з комою (;). Коментарі можуть містити будь-який текст. Коментарі при компіляції програми не впливають на розмір вихідного файлу і алгоритм виконання програми.

Розглянемо приклад будови коду програми (табл. 1.5).

Таблиця 1.5 – Код програми 1.1

| | | | |
|---|---|---|---|
| ;********** | | | |
| ;Автор: * | | | |
| ;Дата: * | | | |
| ;Ім'я файлу: * | | | |
| ;Для AVR: * | | | |
| ;Тактова частота: * | | | |
| ;********** | | | |
| | .device m8515 | | |
| | .nolist | | |
| | .include “C: \Program Files\Atmel\AVR Studio\ Appnotes\m8515def.inc” | | |
| | .list | | |
| | .def temp =r16 **може r17?** | | ; Оголошення: |
| | rjmp Init | | ; Початок програми |
| | | | ; Перша виконувана команда |
| | | | |
| Init: | ldi | temp,0b01010101 | ; Визначаємо входи і виходи порту B |
| | out | DDRB,temp | ; |
| | ldi | temp,0b10101010 | ; Визначаємо входи і виходи порту D |
| | out | DDRD,temp | |
| | ldi | temp,0b01010101 | ; Включаємо підтяжку для входів порту B |
| | out | PortB,temp | ; і задаємо початкові стани виходів |
| | ldi | temp,0b10101010 | ; Включаємо підтяжку для входів |
| | out | PortD,temp | ; порту D і задаємо початкові стани |
| | | | ; виходів |
| Start: | | | ; Основне тіло програми |
| | Inc | r17 | ; Збільшуємо r17 на одиницю |
| | out | PortD,r17 | ; Виводимо вміст r17 в порт D |
| | Rjmp | Start | ; Повертаємося до мітки Start |

Ця програма виводить на світлодіоди вміст регістра r17 через порт D. Враховуючи, що в регістрі напрямку (DDRD) записані 10101010 і в регістрі даних (PORTD) також записані 10101010, тому в регістрі стану (PIND) також з’являться 10101010. Саме вони викликають світіння 7, 5, 3, та 1 світлодіодів. Враховуючи те, що в регістрі PIND реалізується операція логічного множення, то в результаті виконання команди інкрементування, тобто збільшення на 1, вмісту регістра r17 (тобто регістра даних PORTD) вміст регістра PIND також зміниться, але візуально помітити це неможливо. Тому знову ми бачимо, як світяться 7, 5, 3 і 1 світлодіоди.

Для перевірки роботи цієї програми її потрібно ввести в контролер та з’єднати шлейфом роз’єм LEDS з роз’ємом PORTD або роз’єм LEDS з роз’ємом PORTB (тоді будуть світитися світлодіоди 6, 4, 2 і 0).

Рамка із зірочок розташована на самому початку шаблону і є заголовком програми (зірочки тут набрані виключно для краси). Заголовок заповнюється так, щоб можна було легко зрозуміти, що це за програма, не проглядаючи її цілком. Також завдяки заголовку можна переконатися, що ви працюєте з останньою версією програми. Відмітьте, що вміст цього

блоку абсолютно не впливає на реальну роботу програми, оскільки на початку кожного рядка стоїть крапка з комою. У рядку «Тактова частота:» указується частота джерела тактових сигналів (наприклад, кварцового резонатора), підключеного до мікроконтролера. Символ «*» після слів «Тактова частота:» означає те, що в реальній програмі замість «*» потрібно записати повідомлення про тактову частоту. Це повідомлення на роботу контролера не впливає, однак є підказкою для користувача. Мікроконтролеру AVR потрібен стабільний сигнал, що вказує, коли слід переходити до виконання наступної команди; таким чином мікроконтролер виконує команди в кожному періоді тактових імпульсів (або такті). Відповідно, якщо до мікроконтролера підключений резонатор з частотою 4 МГц, то мікроконтролер виконуватиме близько 4 мільйонів команд в секунду. Зверніть увагу на те, що близько 4 мільйонів, оскільки деякі команди (як правило, використовувані для переходів усередині програми) виконуються за два такти. У рядку «Для AVR:» вказується, для якої конкретної моделі мікроконтролера призначена програма.

Після заголовка починаються рядки, що дійсно виконують які-небудь функції. Важливою директивою є директива .include, яка дозволяє асемблеру використовувати так звані файли, що включаються. Вони виконують для асемблера роль словника. Асемблер зрозуміє більшість написаних вами виразів, а для інших йому, можливо, потрібно буде знайти переклад. Наприклад, всі імена регістрів введення−виведення і їх адреси зберігаються у файлах, що підключаються, тому, замість того щоб писати адресу $3F, ви можете вказати символічне ім'я регістра SREG. При установленні програми асемблера на комп'ютер файли, що підключаються, для різних моделей мікроконтролерів поміщаються в певну директорію. Отже, якщо передбачається використовувати модель m8515, повний рядок матиме вигляд:

.include «C:\Program Files\Atmel\AVR Tools\AvrAssembler\Appnotes\m8515 def.inc»

У наступному рядку розташовується перша команда, що виконується мікроконтролером при включенні живлення. У цьому рядку бажано помістити команду переходу до секції, поміченої міткою Init, в якій виконуються всі початкові налаштування AVR. Для цього використовується команда rjmp: rjmp Init.

Це команда відносного переходу (relative jump). Іншими словами, вона наказує мікроконтролеру перейти на ділянку програми, яку ви помітили міткою Init. Причина, з якої перехід називається відносним, пов'язана з тим, яким чином асемблер інтерпретує цю команду. Хай, наприклад, секція Init розташовується через 40 команд від команди rjmp Init. В цьому випадку асемблер інтерпретує цю команду як «перестрибнути вперед через 40 команд», тобто перейти вперед відносно поточної команди. Проте

набагато простіше вважати, що мікроконтролер просто переходить до мітки Init.

У першій частині секції Init задається, які з виводів порту працюватимуть як входи, а які — як виходи. Це здійснюється за допомогою регістрів введення−виведення DDRB і DDRD (регістри напряму передачі даних). Кожен біт цих регістрів відповідає одному з виводів мікроконтролера. Наприклад, біт 4 регістра DDRB відповідає виводу PB4, а біт 2 регістра DDRD — виводу PD2. Установлення відповідного біта регістра DDRx в 1 робить вивід виходом, а скидання біта в 0 робить вивід входом.

Якщо ми конфігуруємо вивід як вхід, то зможемо задати, буде до цього виводу підключений внутрішній підтягувальний резистор чи ні. Це може позбавити нас від необхідності використовувати зовнішні резистори. Щоб включити підтяжку входу, необхідно встановити в 1 відповідний біт регістра Portx; проте, якщо вам цього не потрібно, переконайтеся, що ви її відключили, скинувши відповідний біт регістра Portx в 0. Що ж до виходів, то при включенні мікроконтролера вони повинні знаходитися в певному початковому стані (наприклад, всі вимкнено).

### 1.1.4 Загальна інформація про STK-500

STK-500 є платформою розробки програм для восьмирозрядних RISC AVR МК фірми Atmel. Ця платформа дозволяє програмувати МК через послідовний (SPI) і паралельний інтерфейс, виконувати програми і досліджувати роботоздатність цих програм.

Загальний вигляд STK-500 наведено на рисунку 2.1.

Всю плату STK-500 можна поділити на три області. Перша область (ліва частина плати до білого виділення згідно з рисунком 1.4) містить: роз’єми програмування (RS232 CTRL) і живлення, вимикач живлення (POWER), кнопки скидання (RESET), кнопки програмування (PROGRAM) та кнопки мікросхеми.

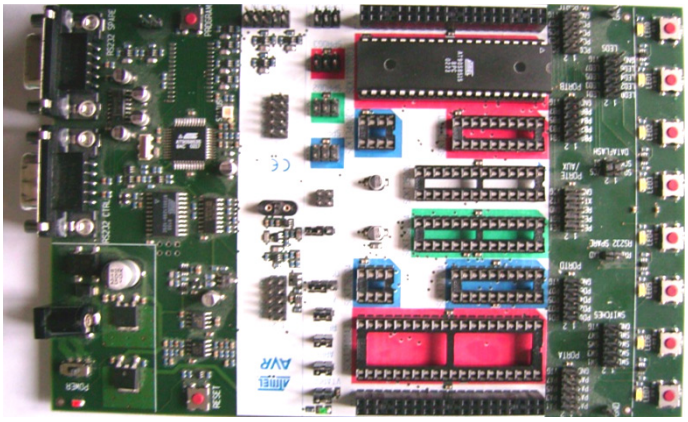


Рисунок 1.4 – Загальний вигляд STK-500

Роз’єм програмування (RS232 CTRL) призначений для підключення STK-500 до ПК з метою програмування МК. Цей роз’єм підключається за допомогою подовжувача до COM порту ПК.

Роз’єм живлення використовується для підключення джерела живлення до STK-500 з метою подати напругу живлення на схему.

Вимикач живлення (POWER) призначений для комутації кола живлення STK-500.

Кнопка скидання (RESET) призначена для подання сигналу скидання на МК, що може приводити до запуску програми з першої адреси пам’яті програм.

Кнопка програмування (PROGRAM) використовується для програмування STK-500.

На мікросхемах організовано тактовий генератор, що може подавати тактову частоту на МК. Також мікросхеми утворюють схему перетворення інтерфейсу RS232 (ПК) на інтерфейс програмування (ISP).

Наступна область STK-500, виділена білим кольором, містить роз’єми програмування шести (ISP6PIN) і десяти (ISP10PIN) виводів. Шестививідний роз’єм використовується для програмування МК на платі STK-500. Наявні також на білій області і роз’єми, що виділені червоним, зеленим і синім кольорами. Кожний роз’єм відповідає певній групі МК. Для того, щоб запрограмувати якийсь МК, потрібно з’єднати шестипіновий роз’єм з роз’ємом відповідної групи, позначеним відповідним кольором.

В третій області знаходяться роз’єми (PORTx, де x – буква відповідного порту), що дублюють виводи МК, роз’єми на які виведено виводи світлодіодів (LEDS) і кнопки (SWITCHES). Для керування світлодіодами потрібний порт МК (PORTx) приєднується до роз’єму світлодіодів (LEDS). Щоб працювати з кнопками, потрібно з’єднати роз’єм відповідного порту МК з роз’ємом кнопок за допомогою 10-провідного кабелю.

### 1.2 Завдання

Запишіть в контролер та виконайте тестову програму 1.2. Нарисуйте алгоритм програми та напишіть до нього пояснення. Зробіть висновки.

### 1.3 Хід роботи

В STK-500 має бути вставлений МК серії AT90S8515 (або ATmega8515) із запрограмованою програмою 1.1 (див. табл. 1.5). Щоб запустити цю програму на виконання і побачити її результат роботи потрібно виконати такі дії.

Для перевірки роботи цієї програми її потрібно ввести в контролер та з’єднати шлейфом роз’єм LEDS з роз’ємом PORTD (будуть світитися світлодіоди 7, 5, 3 і 1) або роз’єм LEDS з роз’ємом PORTB (тоді будуть світитися світлодіоди 6, 4, 2 і 0).

В STK-500 вставлений МК серії AT90S8515 (або ATmega8515) із запрограмованою програмою 1.2. Щоб запустити програму на виконання і побачити її результат роботи потрібно виконати такі дії:

1. Приєднайте до порту В МК світлодіоди, для чого 10-провідним кабелем з'єднати роз'єми PORTB і LEDS, не перекручуючи кабелі;

2. Приєднайте до порту D МК кнопки, для чого 10-провідним кабелем з'єднати роз'єми PORTD і SWITCHES.

3. Вставте роз'єм живлення від блока живлення в роз'єм живлення STK-500 як показано стрілкою на рисунку 1.5.

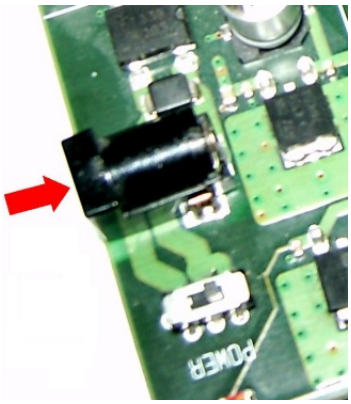


Рисунок 1.5 – Роз'єм живлення STK-500

**Код програми 1.2**

```
;***** STK500 LEDS and SWITCH demonstration
.include «m8515def.inc»
.def Temp =r16 ; Temporary register
.def Delay =r17 ; Delay variable 1
.def Delay2 =r18 ; Delay variable 2
;***** Initialization
RESET:
ser Temp
out DDRB,Temp ; Set PORTB to output
;**** Test input/output
LOOP:
out PORTB,temp ; Update LEDS
sbis PIND,0x00 ; If (Port D, pin0 == 0)
inc Temp ; then count LEDS one down
sbis PIND,0x01 ; If (Port D, pin1 == 0)
dec Temp ; then count LEDS one up
sbis PIND,0x02 ; If (Port D, pin2 == 0)
ror Temp ; then rotate LEDS one right
sbis PIND,0x03 ; If (Port D, pin3 == 0)
rol Temp ; then rotate LEDS one left
sbis PIND,0x04 ; If (Port D, pin4 == 0)
com Temp ; then invert all LEDS
```

```
sbis PIND,0x05 ; If (Port D, pin5 == 0)
neg Temp ; then invert all LEDS and add 1
sbis PIND,0x06 ; If (Port D, pin6 == 0)
swap Temp ; then swap nibbles of LEDS
;**** Now wait a while to make LED changes visible.
DLY:
dec Delay
brne DLY
dec Delay2
brne DLY
rjmp LOOP ; Repeat loop forever
```

4. Увімкніть вимикач живлення та подайте на плату STK-500 напругу.

Програма запустилась на виконання, але ми цього не бачимо.

5. Натисніть на будь-яку кнопку з номером від 0 до 6 (щоб побачити результат виконання програми).

При кожному натисканні на кнопку «0» очікується засвічення світлодіодів. На світлодіодах буде виводитися число, яке буде утворюватись шляхом додавання одиниці до (передуючого натисканню кнопки) значення числа. Тобто спочатку виводиться число 00000000 – світлодіоди не світять, потім потрібно натиснути на кнопку «0». Світлодіоди мають відобразити число 00000001. При утримуванні кнопки на світлодіоди послідовно будуть виводитись числа 00000010, 00000011, 00000100 і т. д.

6. Натисніть на кнопку «1».

Кожний раз, при натисканні на кнопку «1», на світло діоди має виводитись число, яке є результатом віднімання одиниці від числа, яке передувало натисканню на кнопку.

7. Натисніть на кнопку «2».

Якщо натиснути кнопку «2», то число, що відображається на світлодіодах зсунеться вправо на 1 розряд. Наприклад, на світлодіодах світилося число 00100010, після натискання на світлодіодах має відобразитися число 00010001. При подальшому утримуванні кнопки з’явиться число 00001000.

8. Натисніть на кнопку «3».

При натисканні кнопки «3» на світлодіоди буде виводитися число – результат зсування вліво.

9. Натисніть на кнопку «4».

В результаті натискання кнопки «4» число, що виводиться на світлодіоди, буде інвертуватися. Наприклад, світлодіоди відображають число 01100111, після натискання кнопки на світлодіодах відобразиться число 10011000.

10. Натисніть на кнопку «5».

Якщо натиснути кнопку «5», то на світлодіоди буде виводитись число, що утворилось при інвертуванні попереднього числа і додавання до нього одиниці. Наприклад, на світлодіодах відображається число 01110111, після натискання кнопки програма проінвертує число і отримаємо число 10001000, потім до цього числа додасть одиницю

10001000+1=10001001

і вже це число виведеться на світлодіоди.

11. Натисніть на кнопку «6».

В результаті натискання кнопки «6» на світлодіоди виведеться число, яке утвориться від переставляння чотирьох старших бітів з чотирма молодшими бітами числа. Наприклад, якщо на світлодіодах буде світитись число 01101111, то після натискання кнопки на світлодіоди виведеться число 11110110.

12. Запишіть результати виконаних дій за п.п. 1.3.4–1.3.11.

13. Зробіть висновки.

### 1.4 Контрольні запитання

1. Що є характерною рисою сучасного науково-технічного прогресу?
2. Що лежить в основі нової мікропроцесорної техніки?
3. Що вам відомо про однокристальні мікроЕОМ з мікропрограмним управлінням?
4. Що вам відомо про істотні недоліки систем управління, у разі їх побудови на базі персональних комп'ютерів як найбільш універсальних засобів обчислювальної техніки?
5. Що називається керувальним контролером?
6. Що називається програмованим контролером і для чого він призначений?
7. Для чого призначені 8-розрядні RISC-мікроконтролери?
8. Що вам відомо про фірму Atmel?
9. Що вам відомо про архітектуру AVR контролерів?
10. Які переваги у AVR мікроконтролерів?
11. Що таке RISC мікроконтролер?
12. Що таке CISC мікроконтролер?
13. З чого складається МК?
14. Що таке FLASH?
15. Що таке EEPROM?
16. Призначення виводу RESET.
17. Який результат виконання рядка ;inc r16 ?
18. Який результат виконання команди rjmp loop ?
19. Призначення регістра DDRB.
20. Для чого призначений STK-500?

21. Спільно з якою фірмою фірма ATMEL розробляла систему команд і внутрішню побудову чипів AVR?

22. Які переваги дає використання прогресивної технології конвеєризації у AVR МК?

23. Які вам відомі галузі застосування AVR МК?

24. Що вам відомо про мікроконтролер сімейства AVR фірми *Atmel* ?

25.Що входить до складу процесора (CPU) AVR МК?

26. Що вам відомо про генератор тактового сигналу?

27. Що означає, що AVR МК оперує 8-бітовими числами?

28. Що вам відомо про регістри загального призначення AVR МК?

29. Що вам відомо про блок регістрів загального призначення – GPR AVR МК?

30. Що вам відомо про блок регістрів загального призначення – GPR AVR МК?

31. Що вам відомо про регістр стану мікроконтролера SREG AVR МК?

32. Що вам відомо про AVR МК ATmegaS8515?

33. З яких основних етапів складається процес розробки програм?

34. Яку інформацію про МК можна отримати з його позначення?

# 2 Лабораторна робота № 2
# ЗАПИС ТА ВИКОНАННЯ ПРОСТИХ ПРОГРАМ

**Мета роботи:** ознайомитись із призначенням компілятора асемблера для МК AVR і написанням простих програм на асемблері.

## 2.1 Короткі теоретичні відомості

### 2.1.1 Мікроконтролери Atmel AVR

Написання програм для мікроконтролера (МК) в цій лабораторній роботі буде виконуватись мовою асемблера для МК AVR. Всі команди цієї мови можна поділити відповідно до їх призначення на такі групи: арифметичні, логічні, умовні, передачі даних і роботи з бітами.

Як відомо, програмою називається сукупність команд, які забезпечують виконання операцій згідно із алгоритмом. Для написання програм, як правило, дотримуються такої послідовності.

1. Постановка задачі.
2. Аналіз можливих програмних шляхів вирішення задачі.
3. Побудова алгоритму програми.
4. Написання програми.
5. Виправлення помилок.

При написанні програми для МК, яка буде працювати із зовнішніми пристроями, потрібно після першого пункту провести розробку блок-схеми алгоритму програми, на основі якої буде проводитись написання коду програми. Враховуючи те, що при роботі із зовнішніми пристроями МК використовує порти введення-виведення, то вони, як відомо, мають свої адреси, з якими вже і працює програміст. Також потрібно знати частоту тактового генератора МК. Оскільки при побудові інтерфейсів між МК і іншими пристроями проводиться «визначення» швидкості передачі даних, то особливість «вибору» цієї швидкості залежить від тактової частоти генератора.

## 2.2 Самостійна підготовка

Під час самостійної підготовки необхідно:

1. Ознайомитись з архітектурою МК AT90S8515 або ATmega8515.
2. Ознайомитись з командами асемблера і звернути увагу на ті команди, які підтримують ці МК.
3. Ознайомитись із AVRStudio.
4. Виконати завдання для самостійної роботи.

### 2.2.1 Завдання для самостійної роботи

Запишіть в комірку пам’яті даних за адресою **0x62** число, яке відповідає проінвертованому вмісту комірки пам’яті **0x60**.

Зверніть увагу на те, що адреси комірок пам’яті записуються в шістнадцятковому коді. Наприклад, запис десяткового числа $01_{10}$ в десятковому коді буде 01, в двійковому коді буде 0b01 (0b – ознака двійкового коду), в шістнадцятковому коді буде 0x01 (0x – ознака шістнадцяткового коду).

### 2.2.2 Виконання завдання

Для виконання цього завдання вам потрібно провести зчитування із комірки 0x60 значення числа, яке там зберігається, а потім знайти інверсне до цього числа інше число і записати результат інвертування у комірку за адресою 0x62.

Алгоритм програми контрольного прикладу наведений на рисунку 2.1.

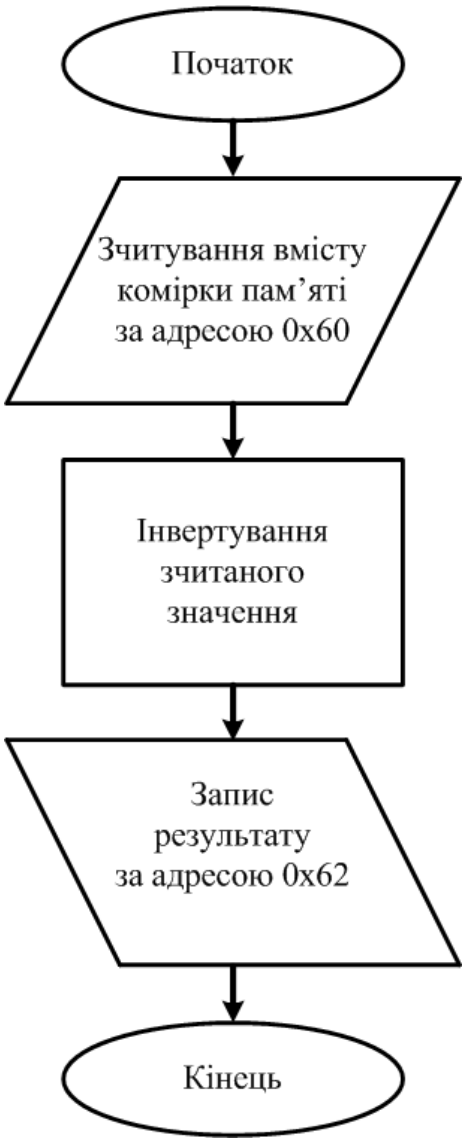



Рисунок 2.1 – Алгоритм програми для самостійної роботи

Наступний крок – це введення коду програми. Для цього перейдіть в меню програми AVRStudio, далі виберіть пункт меню Project, далі –

New Project (рис. 2.2). AVRStudio є інтегрованим програмним середовищем, яке призначене для написання, налаштування і вдосконалення кодів програм восьмирозрядних AVR МК фірми Atmel. AVRStudio є інструментом: проектування мікропроцесорних пристроїв, редагування файлу коду програм, емулятором 8-розрядних RISC мікроконтролерів.

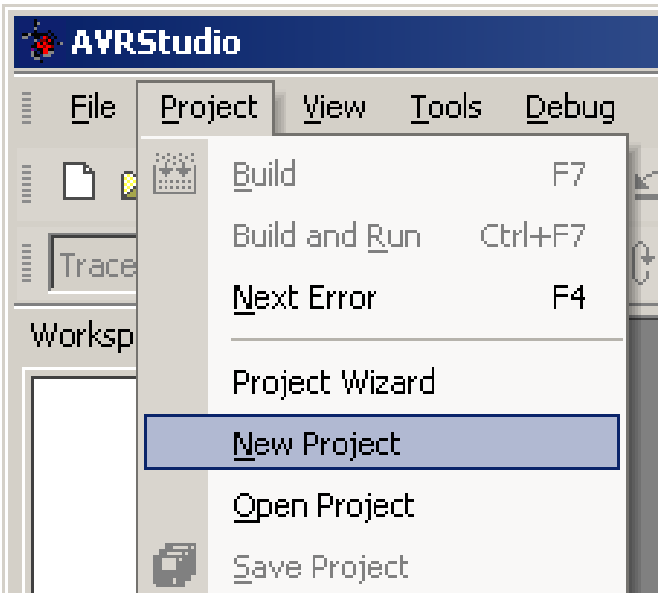



Рисунок 2.2 – Підготовка до введення коду програми

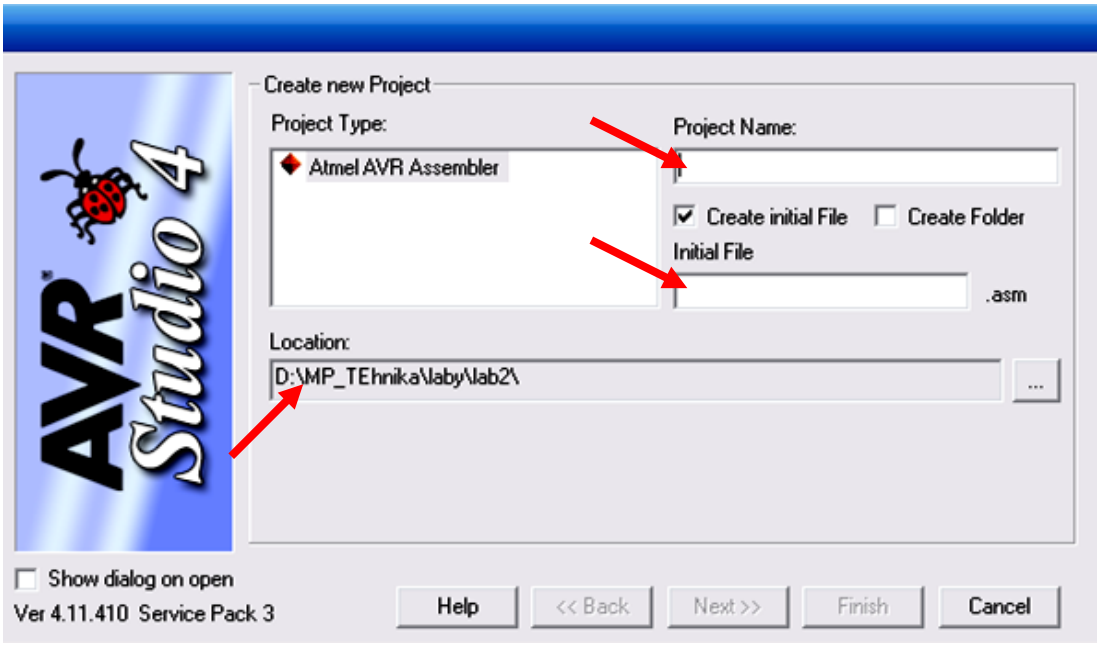



Рисунок 2.3 – Вікно помічника створення проекту

Після цього з'являється вікно помічника створення проекту, в якому пропонується ввести ім'я проекту (наприклад, lab2) і каталог, у якому будуть зберігатись всі файли проекту (d:\MP_Tehnika\laby\lab2\ (рис.2.3).

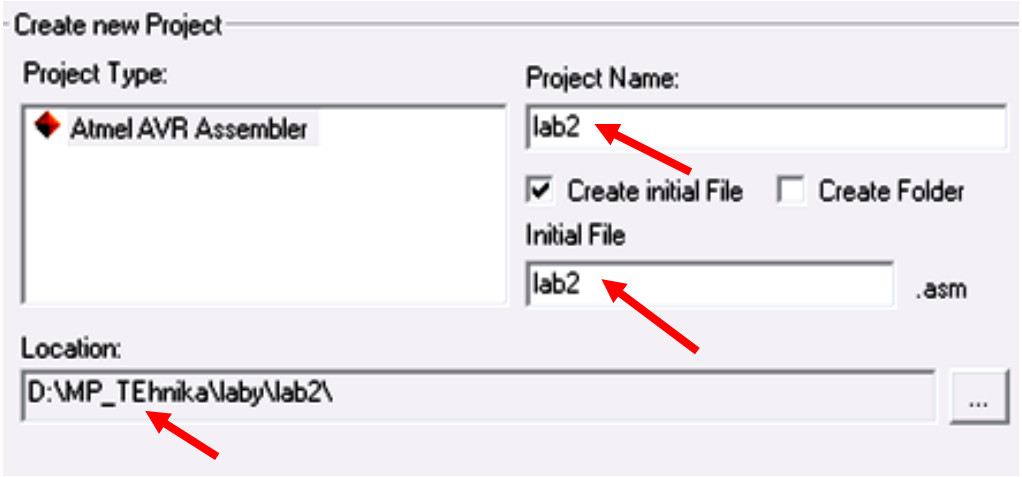



Рисунок 2.4 – Введення імені файлу коду програми

Ввівши відповідні дані про проект, натисніть кнопку Next >> для переходу до наступного кроку створення проекту і отримаєте вікно (рис. 2.5).

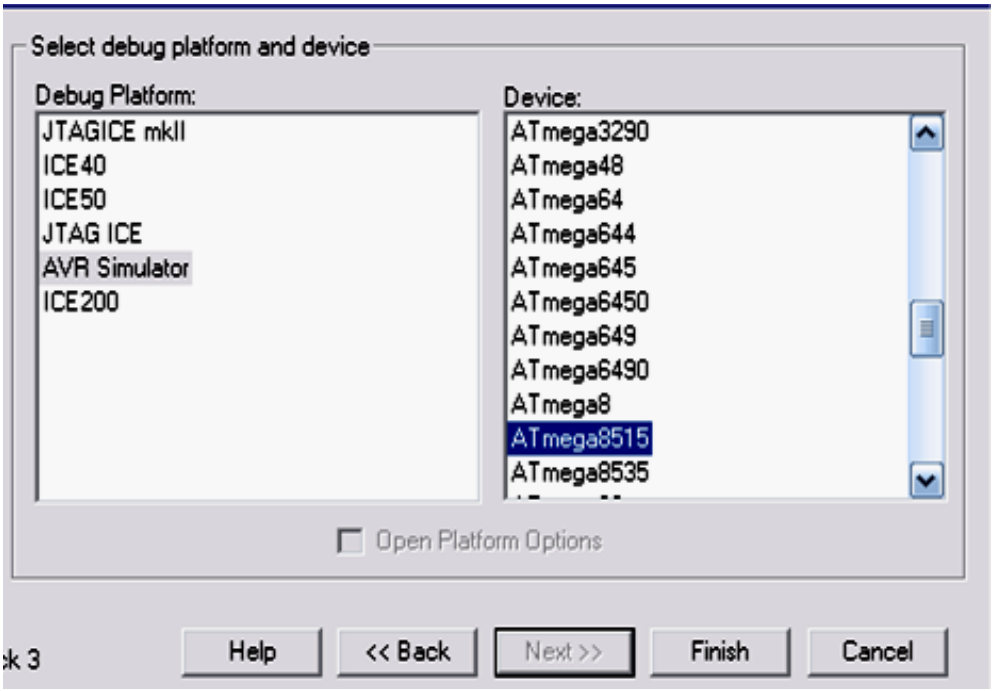



Рисунок 2.5 – Вікно вибору платформи на процесорі ATmega8515

У вікні (рис. 2.5) виберіть платформу для розробки програми і тип МК для якого буде проводитись написання програми. Як платформу (у вікні **Debug Platform**) для розробки виберіть **AVR Simulator**, тобто розташований в компіляторі симулятор МК. Як тип МК (у вікні **Device**) виберіть **AT90S8515** або **ATmega515** залежно від того, який тип контролера ви використовуєте. Після встановлення потрібних параметрів закінчіть створення проекту, натиснувши на кнопку Finish.

На екрані з’явиться вікно редактора тексту програми (рис. 2.6).

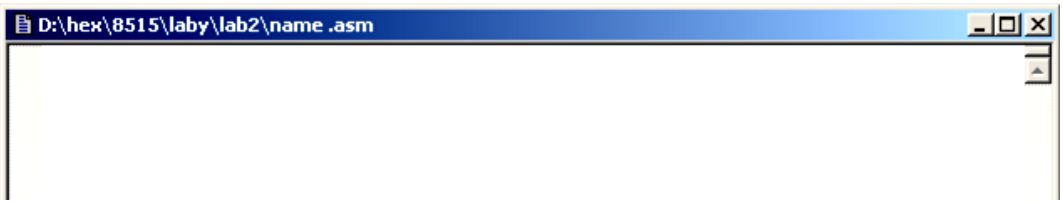



Рисунок 2.6 – Вікно редактора тексту програми

У цьому вікні проводиться написання коду програми і виправлення помилок.

Введіть такий код програми:

```
; Програма виконує читання даних з комірки пам’яті (при використанні
; непрямої адресації), проводить інверсію і записує в комірку пам’яті
; те ж при використанні непрямої адресації. Запис проводиться в область
;  даних
.include «8515def.inc»
ldi r30,0x60     ;записуємо в регістр z адресу комірки, з якої будуть братися дані
ldi r28,0x62     ;записуємо в регістр y адресу комірки, куди будемо записувати дані
ld r16,z  ;завантажуємо дані з комірки пам’яті за адресою  0x60 в r16
com r16 ;інвертуємо вміст регістра r16
st y,r16  ;записуємо вміст регістра r16 у комірку пам'яті із адресою 0x63
(в команді ldi r28,0x62 – потрібно 0x63 )
```

Після введення коду програми вікно редактора матиме вигляд, показаний на рис. 2.7.

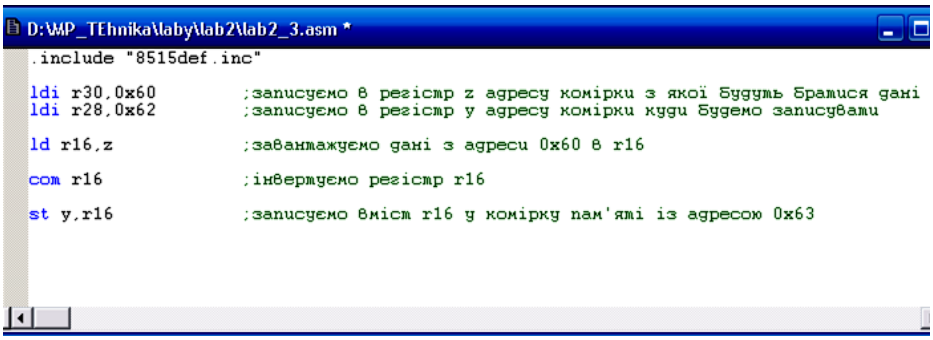



Рисунок 2.7 – Текст програми завдання для самостійної роботи (МК AT90S8515)

Для МК ATmega8515 в рядок коду програми:

.include «8515def.inc»

треба внести зміни, після яких рядок матиме такий вигляд:

.include «m8515def.inc».

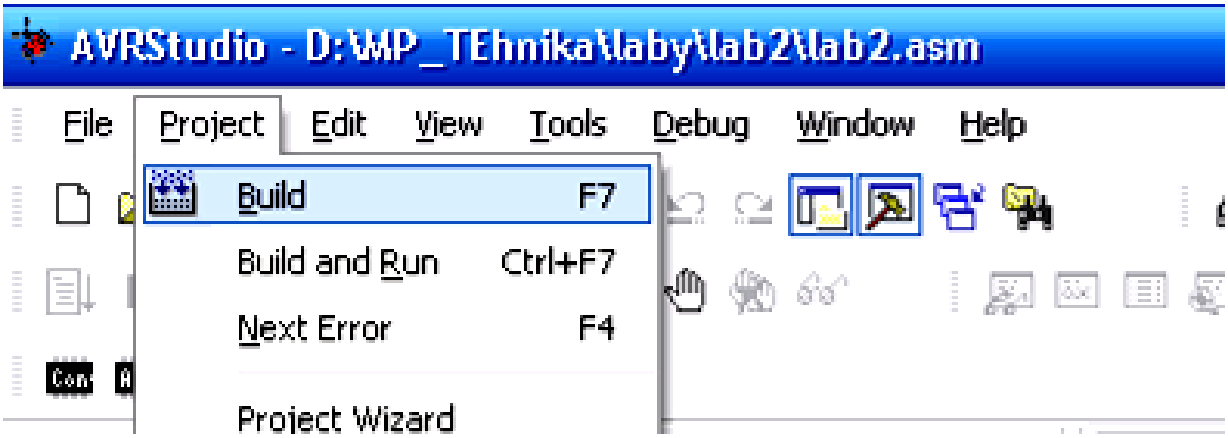



Рисунок 2.8 – Компіляція проекту

Після написання коду програми здійсніть компіляцію проекту. Для цього у головному меню виберіть вкладку «*Project*», а в ній пункт «*Build*» так, як це показано на рис. 2.8.

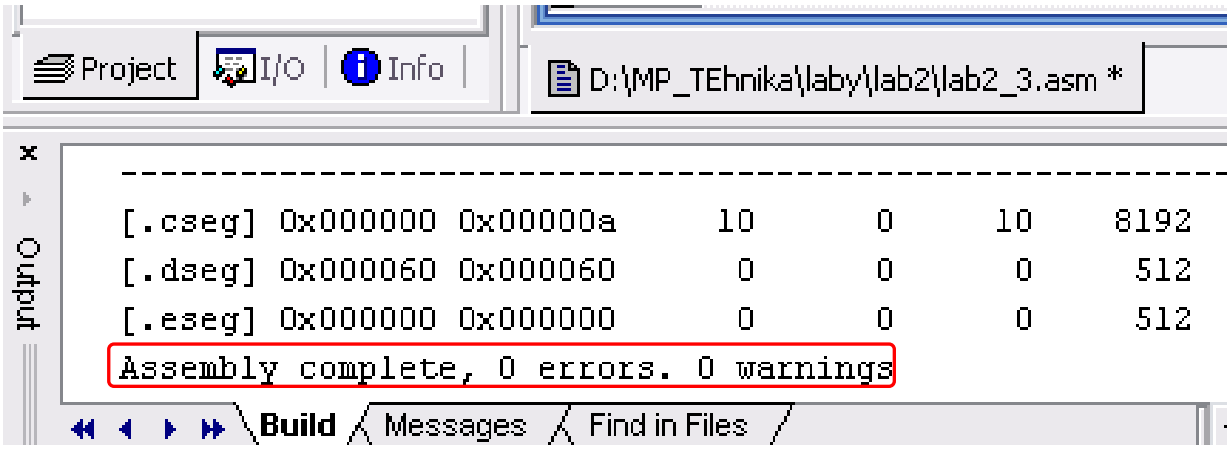



Рисунок 2.9 – Результат успішної компіляції програми

Далі отримаєте або створення виконуваного файлу (рис. 2.9), або вказівки компілятора про наявність помилок (рис. 2.10). За відсутності помилок буде повідомлення – «Assembly complete, 0 errors. 0 warnings.».

За наявності помилок (*errors*) та відсутності попереджень (*warnings*) буде повідомлення – «Assembly failed, 2 errors. 0 warnings.».

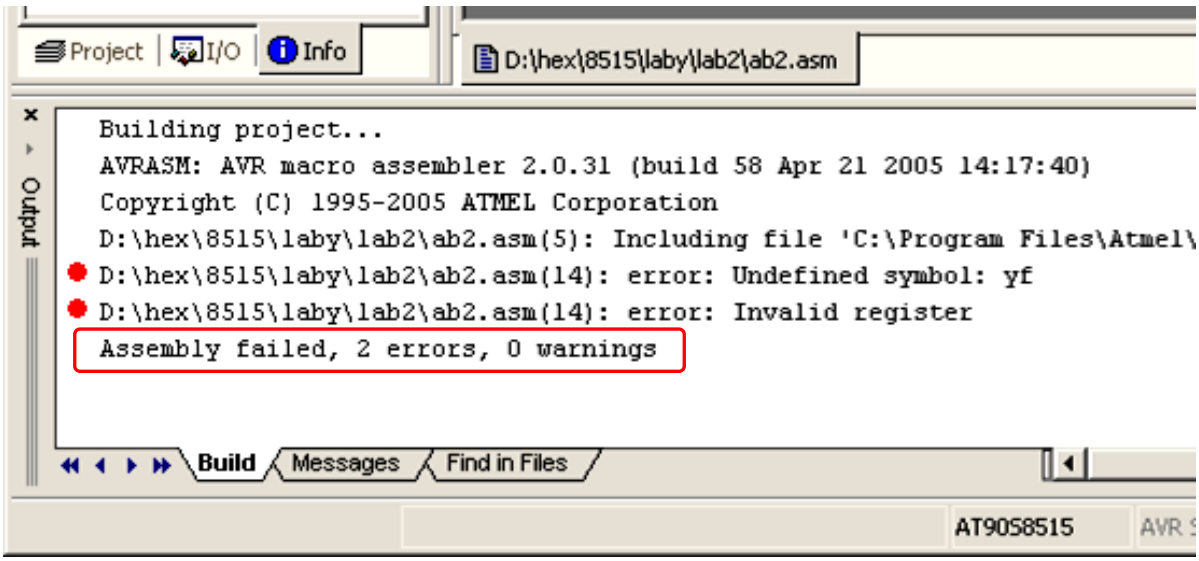



Рисунок 2.10 – Результат компіляції програми, яка містить помилки

Якщо після компіляції отримано позитивний результат, то потрібно перевірити, чи виконує дана програма поставлену задачу. Для цього скористайтеся емулятором МК.

Для цього у головному меню виберіть вкладку «*Project*», а в ній пункт «*Build and Run*» так, як це показано на рис. 2.11.

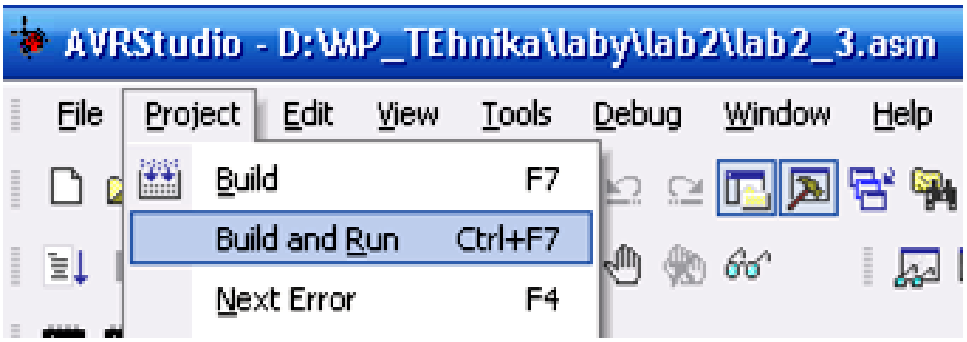



Рисунок 2.11 – Перевірка роботи програми

Через деякий час у полі коду програми з’явиться стрілка, яка вказуватиме на виконуваний рядок коду програми при покроковому виконанні (рис. 2.12).

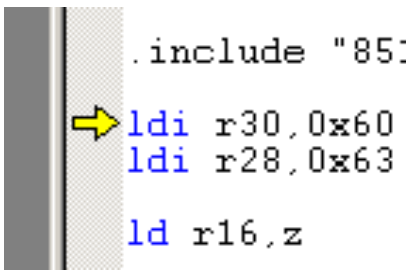



Рисунок 2.12 – Виконуваний рядок коду програми

Контрольний приклад працює із пам’яттю даних, тому відобразіть область пам’яті даних. Для цього у головному меню виберіть вкладку «*View*», а в ній пункт «*Memory*» так, як це показано на рис. 2.13.

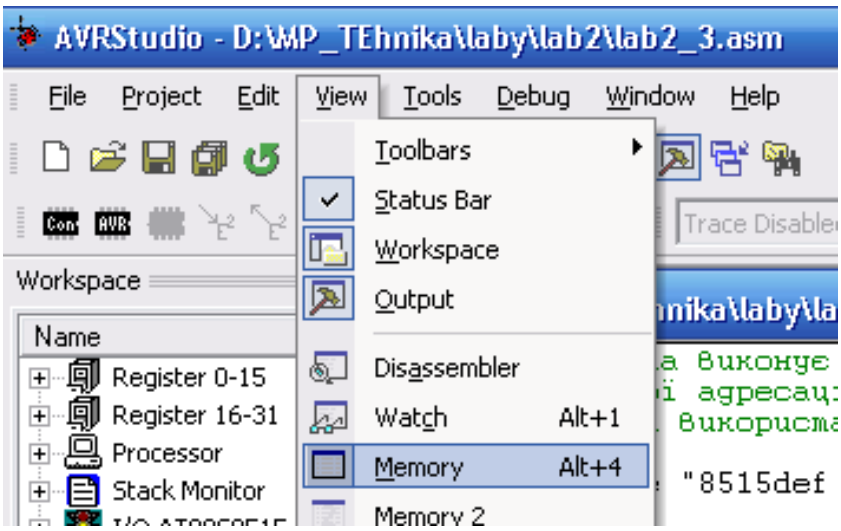



Рисунок 2.13 – Підготовка до відображення області даних

Далі з’явиться вікно, показане на рис. 2.14.

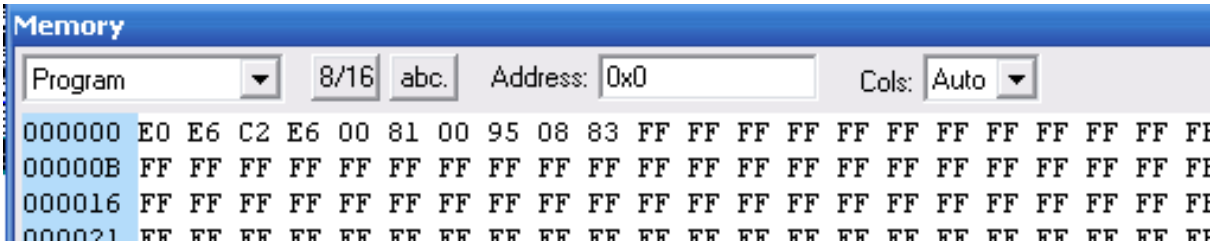



Рисунок 2.14 – Вікно стану пам’яті МК

У вікні (рис. 2.14) налаштуйте відображення області даних. В нашому випадку налаштовуємо відображення саме пам’яті даних (*Data*). Для цього натисніть на пункт «*Program*» (рис. 2.15).

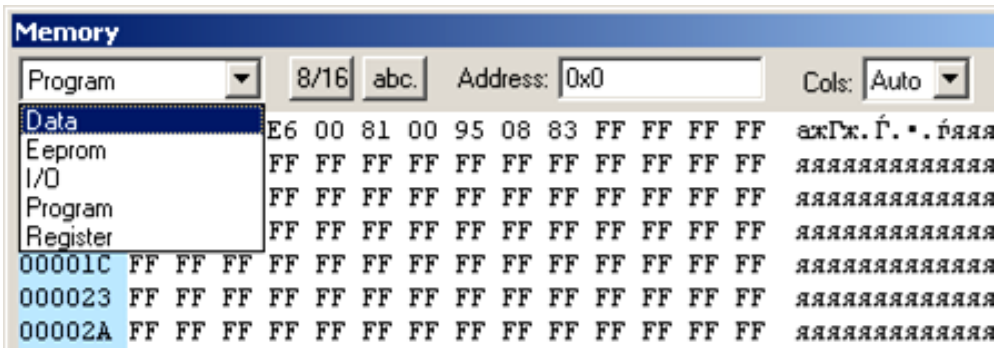



Рисунок 2.15 – Налаштування відображення вікна стану пам’яті МК

Відкриється меню з пунктами, які відображають розподіл комірок пам’яті за призначенням.

Наприклад:

*Data* - пам’ять даних;
*Eepr* - ;
*om*
*I/0* - ;
*Prog* - пам’ять програм;
*ram*
*Regi* - пам’ять універсальних регістрів.
*ster*

Виберіть пункт «*Data*». Виберіть рядок 000060. Цей рядок відповідає адресам регістрів, починаючи від 000060 і далі, залежно від розміру вікна по горизонталі. Наприклад до 000067, як це показано на рис. 2.16.


Рисунок 2.16 – Перевірка вмісту комірки пам’яті 000060

Виберіть комірку пам’яті 000060 і змініть її вміст з FF на 01, а потім вміст комірки 000062 з FF на 02 так, як це показано на рис. 2.17.

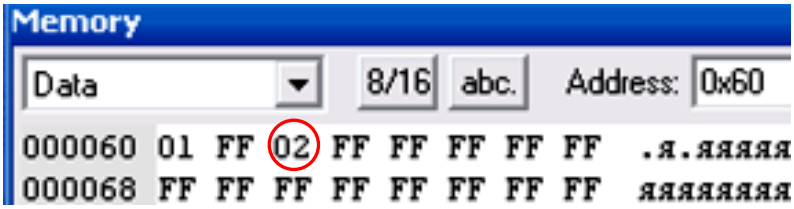



Рисунок 2.17 – Зміна вмісту комірок пам’яті 000060 та 000062

В головному меню виберіть вкладку «*Debug*» (рис. 2.18). Далі потрібно вибрати один з трьох варіантів виконання програми:

1. Звичайний режим – пункт меню «*Run*» вставки «*Debug*»;

2. Режим ручного покрокового виконання програми – пункт меню «*Step Into*» вставки «*Debug*»;

3. Режим з повільним (для кращого спостереження) автоматичним покроковим виконанням команд програми – пункт меню «*AutoStep*» вставки « *Debug*» .

Виберіть пункт меню «*Step Into*» вставки «*Debug*» і натисніть *F11*.

Після цього виконається рядок програми *ldi r30,0x60*. В результаті в регістр *r30* запишеться вміст комірки 000060, тобто число 01. Нагадаємо, що регістр *Z* містить в собі два регістри: регістр *r30* та *r31*.

Стрілка в полі коду програми переміститься на рядок нижче (рис. 2.19). Це вказує на готовність виконувати наступний рядок програми: ldi r28,0x62.

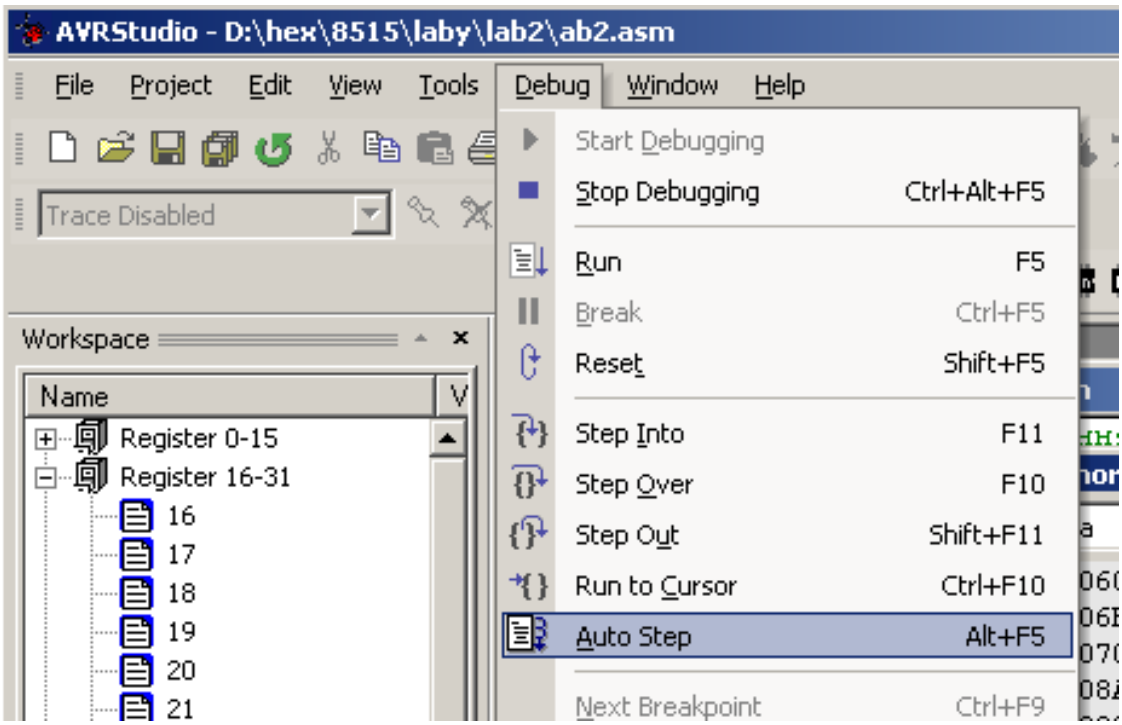



Рисунок 2.18 – Вибір режиму виконання програми

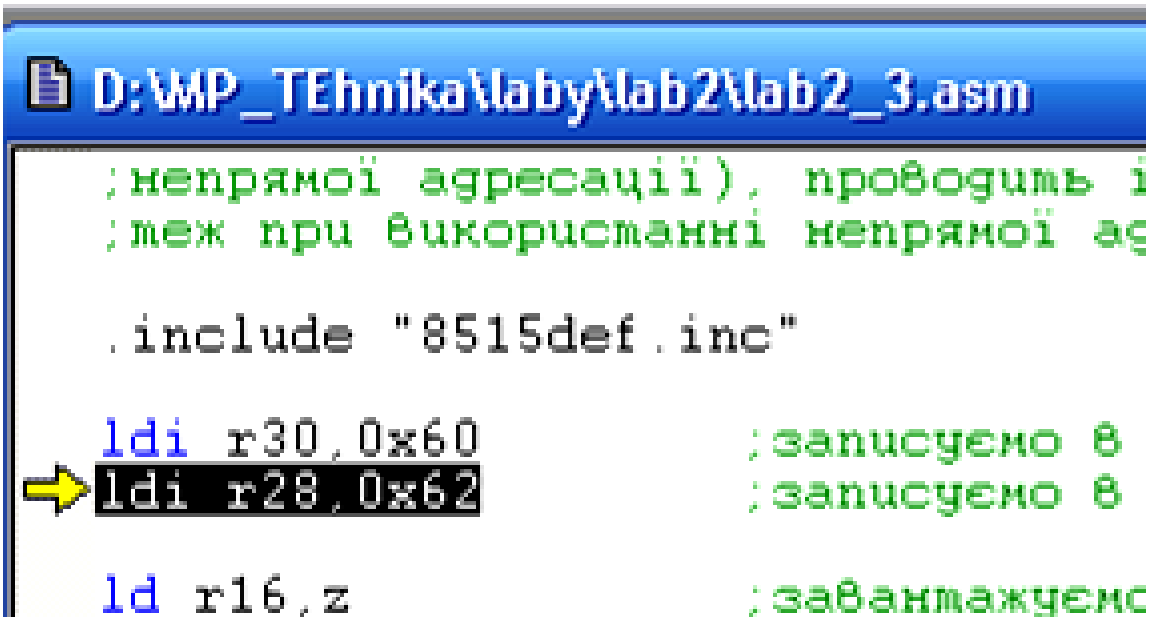



Рисунок 2.19 – Перехід на наступний рядок програми

Результат виконання команди ldi r30,0x60 можна побачити, натиснувши на «+» в рядку «Register 16-31», як це показано на рис. 2.20.

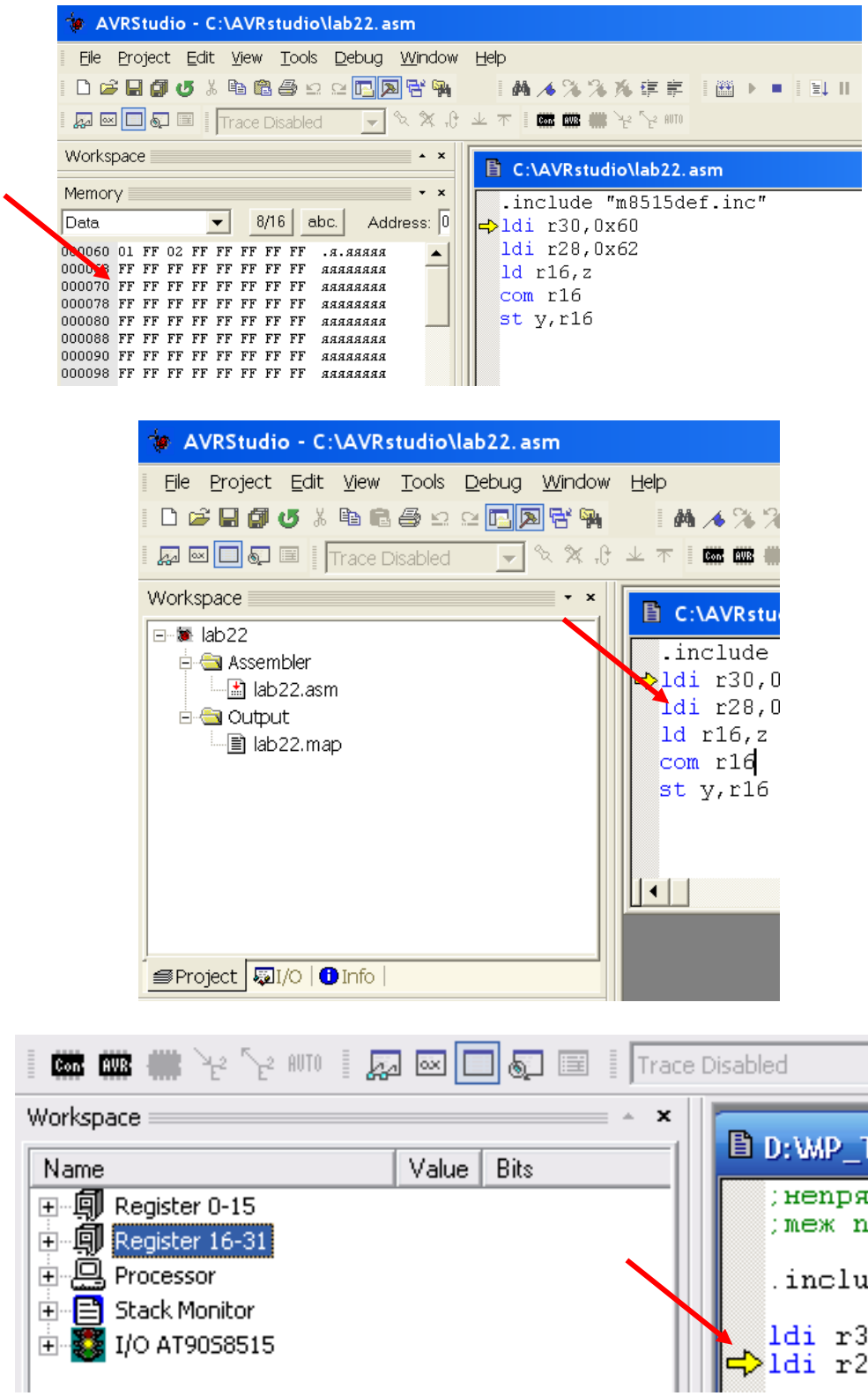



Рисунок 2.20 – Перехід до перегляду вмісту регістрів r30 та r31

Як бачимо на рис. 2.21 регістр *r30* змінив свій вміст з 0x00 на 0x60, що означає виконання першого рядка програми.

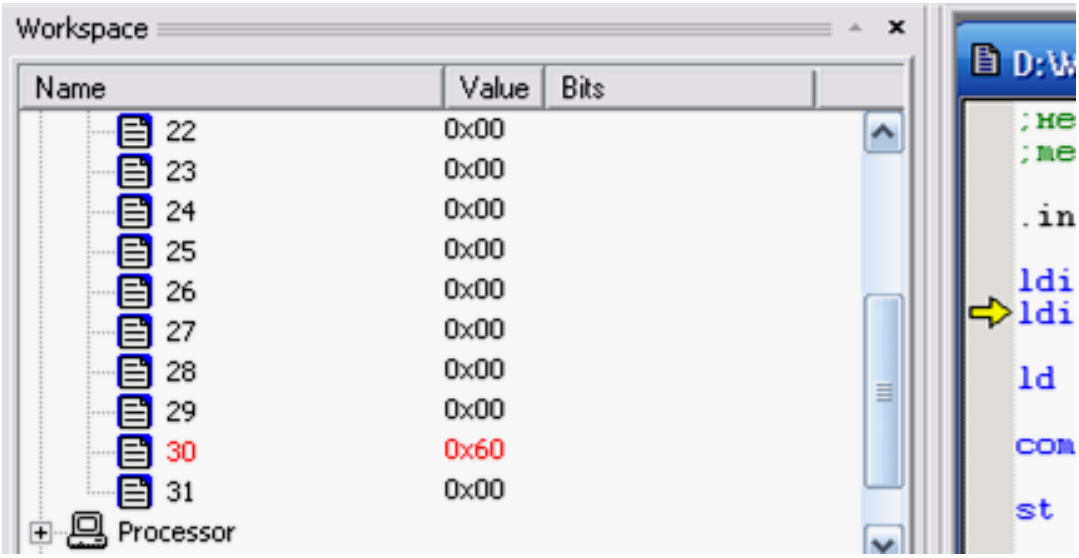



Рисунок 2.21 – Перегляд вмісту регістрів r30 та r31

Якщо далі натискати на кнопку F11, то є можливість відслідковувати виконання кожного наступного кроку програми.


Рисунок 2.22 – Результат виконання другого рядка програми

Для того, щоб побачити результат виконання цієї програми, вам потрібно натиснути на вкладку «*View*» в головному меню. На цій вкладці виберіть пункт меню «*Memory*». Результат побачите у вікні, фрагмент якого показаний на рис. 2.23.

Приклад висновку

В результаті роботи програми вміст комірки 000062 змінився з 02 (див. рис. 2.17) на FE **(**рис. 2.23**)**.

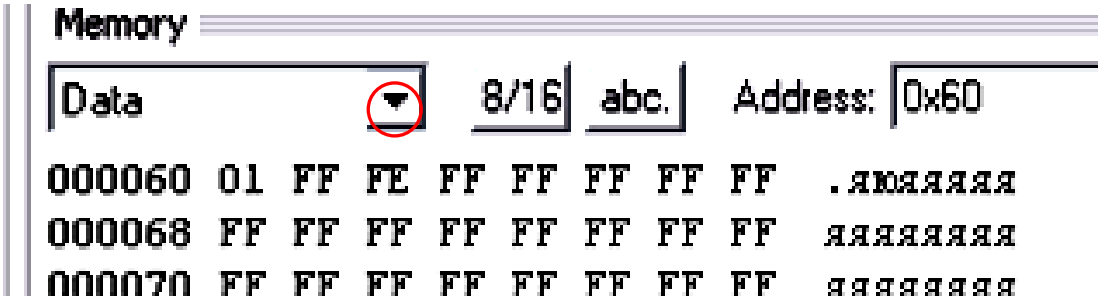



Рисунок 2.23 – Результат виконання всієї програми

Нагадуємо, що в шістнадцятковій системі числення інвертованим числом до числа 00 буде число FF, до числа 01 – число FE, до числа 02 – число FD, до числа 03 – число FC, до числа 04 – число FB, до числа 05 – число FA і т. п.

### 2.2.3 Пояснення до тексту програми

Команда Ldi – записує константу у регістр від r16 до r31.

Команда Ld – завантажує в регістр (наприклад, *r16*) значення числа – вмісту комірки пам'яті даних, адреса якої знаходиться в шістнадцятирозрядному розрядному регістрі, наприклад *Z*, який складається з двох восьмирозрядних регістрів *R30* та *R31*. В програмі до завдання для самостійної роботи – це повна адреса 0060. Це відповідає молодшим розрядам адреси – 60 та старшим розрядам – 00. Тобто адреса є шістнадцятирозрядною і розташована у двох восьмирозрядних регістрах *R30* та *R31* регістра *Z*. Існують регістри *r26* та *r27*, які входять до складу регістра *X*, і регістри *r28*, *r29*, що входять до регістра *Y*.

Команда Com r16 – інвертує число, яке зберігається в регістрі *r16,* і записує результат в регістр *r16*.

Команда St y, r16 – записує останнє значення вмісту комірки пам'яті регістра r16 в пам'ять даних за адресою, яка знаходиться в шістнадцятирозрядному регістрі ***Y***, тобто в регістр *r28*.

## 2.3 Робота в лабораторії

Хід роботи.

1. Ознайомтеся із призначенням і можливостями компілятора AVRStudio.
2. Напишіть прості алгоритм і програму згідно із індивідуальним завданням.
3. Введіть код програми в середовищі AVRStudio.
4. Виконайте програму в емуляторі AVRStudio.
5. Порівняйте отримані результати із завданням.
6. Зробіть висновки.

## 2.4 Завдання

Напишіть програму для МК AT90S8515 або ATmega8515, яка запише в комірку пам'яті даних за адресою 000064 число, що відповідає проінвертованому вмісту комірки пам'яті 000062, відповідно до варіанта таблиці 2.1.

Нарисуйте алгоритм програми. Зробіть висновки.

Таблиця 2.1 – Варіанти завдань

| Група 1 | | | Група 2 | | |
|---|---|---|---|---|---|
| № варіанта | Вміст комірки пам'яті 000061 | Початковий вміст комірки пам'яті 000064 | № варіанта | Вміст комірки пам'яті 000061 | Початковий вміст комірки пам'яті 000064 |
| 1 | 5 | 7 | 1 | 8 | 6 |
| 2 | 6 | 8 | 2 | 9 | 7 |
| 3 | 7 | 9 | 3 | 10 | 8 |
| 4 | 8 | 10 | 4 | 11 | 9 |
| 5 | 9 | 11 | 5 | 12 | 10 |
| 6 | 10 | 12 | 6 | 13 | 11 |
| 7 | 11 | 13 | 7 | 14 | 12 |
| 8 | 12 | 14 | 8 | 15 | 13 |
| 9 | 13 | 15 | 9 | 16 | 14 |
| 10 | 14 | 16 | 10 | 17 | 15 |
| 11 | 15 | 17 | 11 | 18 | 16 |
| 12 | 16 | 18 | 12 | 19 | 17 |
| 13 | 17 | 19 | 13 | 20 | 18 |
| 14 | 18 | 20 | 14 | 21 | 19 |
| 15 | 19 | 21 | 15 | 22 | 20 |
| 16 | 20 | 22 | 16 | 23 | 21 |
| 17 | 21 | 23 | 17 | 24 | 22 |
| 18 | 22 | 24 | 18 | 25 | 23 |
| 19 | 23 | 25 | 19 | 26 | 24 |
| 20 | 24 | 26 | 20 | 27 | 25 |
| 21 | 25 | 27 | 21 | 28 | 26 |
| 22 | 26 | 28 | 22 | 29 | 27 |
| 23 | 27 | 29 | 23 | 30 | 28 |
| 24 | 28 | 30 | 24 | 31 | 29 |
| 25 | 29 | 31 | 25 | 32 | 30 |
| 26 | 30 | 32 | 26 | 33 | 31 |
| 27 | 31 | 33 | 27 | 34 | 32 |
| 28 | 32 | 34 | 28 | 35 | 33 |
| 29 | 33 | 35 | 29 | 36 | 34 |
| 30 | 34 | 36 | 30 | 37 | 35 |

Продовження таблиці 2.1

| Група 3 | | | Група 4 | | |
|---|---|---|---|---|---|
| № варіанта | Вміст комірки пам'яті 000061 | Початковий вміст комірки пам'яті 000064 | № варіанта | Вміст комірки пам'яті 000061 | Початковий вміст комірки пам'яті 000064 |
| 1 | 10 | 19 | 1 | 16 | 25 |
| 2 | 11 | 20 | 2 | 17 | 26 |
| 3 | 12 | 21 | 3 | 18 | 27 |
| 4 | 13 | 22 | 4 | 19 | 28 |
| 5 | 14 | 23 | 5 | 20 | 29 |
| 6 | 15 | 24 | 6 | 21 | 30 |
| 7 | 16 | 25 | 7 | 22 | 31 |
| 8 | 17 | 26 | 8 | 23 | 32 |
| 9 | 18 | 27 | 9 | 24 | 33 |
| 10 | 19 | 28 | 10 | 25 | 34 |
| 11 | 20 | 29 | 11 | 26 | 35 |
| 12 | 21 | 30 | 12 | 27 | 36 |
| 13 | 22 | 31 | 13 | 28 | 37 |
| 14 | 23 | 32 | 14 | 29 | 38 |
| 15 | 24 | 33 | 15 | 30 | 39 |
| 16 | 25 | 34 | 16 | 31 | 40 |
| 17 | 26 | 35 | 17 | 32 | 41 |
| 18 | 27 | 36 | 18 | 33 | 42 |
| 19 | 28 | 37 | 19 | 34 | 43 |
| 20 | 29 | 38 | 20 | 35 | 44 |
| 21 | 30 | 39 | 21 | 36 | 45 |
| 22 | 31 | 40 | 22 | 37 | 46 |
| 23 | 32 | 41 | 23 | 38 | 47 |
| 24 | 33 | 42 | 24 | 39 | 48 |
| 25 | 34 | 43 | 25 | 40 | 49 |
| 26 | 35 | 44 | 26 | 41 | 50 |
| 27 | 36 | 45 | 27 | 42 | 51 |
| 28 | 37 | 46 | 28 | 43 | 52 |
| 29 | 38 | 47 | 29 | 44 | 53 |
| 30 | 39 | 48 | 30 | 45 | 54 |

## 2.5 Зміст звіту до лабораторної роботи

У звіті мають бути відображені: тема роботи, мета роботи, короткі теоретичні відомості, поставлена задача і можливі шляхи її вирішення, алгоритм програми, текст програми, висновки, використана література.

## 2.6 Контрольні запитання

1. Що вам відомо про шістнадцяткову систему числення?
2. Що вам відомо про призначення та можливості AVRStudio?
3. В якій системі числення записуються адреси комірок пам’яті та вміст комірок пам’яті в AVRStudio?
4. Як записати текст програми в AVRStudio?
5. Як змінити та як переглянути вміст комірок пам’яті даних МК AT90S8515 ATmega8515?
6. Як задати тип МК, з яким далі буде працювати STK-500 та AVRStudio?
7. Перетворіть шістнадцяткові числа (задаються викладачем) у двійкові і у десяткові числа.
8. Як виконати компіляцію набраного коду програми?
9. Що вам відомо про команду Ldi? Які дії виконуються під час роботи такого рядка програми: Ldi r30, 0x60 ?
10. Які дії виконуються під час роботи такого рядка програми: Ldi r28, 0x62 ?
11. Що вам відомо про команду Ld? Які дії виконуються під час роботи такого рядка програми: Ld r16, z ?
12. Що вам відомо про команду com? Які дії виконуються під час роботи такого рядка програми: com r16?
13. Що вам відомо про команду st y, r16? Які дії виконуються під час роботи такого рядка програми: st y, r16?

# 3 Лабораторна робота № 3
# РОБОТА З ПОРТАМИ ВВЕДЕННЯ–ВИВЕДЕННЯ AVR

**Мета роботи:** ознайомитись і отримати навички роботи з портами введення-виведення МК AVR.

## 3.1 Короткі теоретичні відомості

### 3.1.1 Порти введення-виведення

МК AVR фірми Atmel побудовані таким чином, що порти МК мають три регістри, за допомогою яких можна керувати портами МК. Це регістр керування DDRx (x – назва порту), що визначає, як працює порт ( на введення чи на виведення інформації). Регістр стану (PINx) відображає рівень сигналу на виводі порту (рівень логічної одиниці чи рівень нуля). Регістр даних (PORTx) зберігає дані, які передаються або приймаються портом.

Враховуючи будову порту можна сказати, що коли порт налаштований на виведення інформації (в регістр напрямку записана 1), то записування в регістр даних (PORTx) одиниці відповідає підключенню виводу порту через резистор на позитивну шину живлення.

Операції з портами проводяться за допомогою команд введення-виведення, а саме таких команд, як: **IN R16,PORTB** і **OUT DDRA,R17**. Також можуть застосовуватись команди роботи з бітами **CBI PORTC.PIN1.**

### 3.1.2 Загальні відомості про систему команд

В сімействі AVR система команд у мікроконтролерів різних типів містить від 89 до 130 команд. У мікроконтролерів типу 2323, 2343, 2313, 4433, 8515 і 8535 до системи команд входять 118 команд. Цю систему команд називають базовою.

Базова система команд містить:

-33 команди регістрових операцій, при виконанні яких використовуються тільки регістри загального призначення;

- 26 команд із зверненням за адресою в адресному просторі SRAM;

- 2 команди із зверненням до регістрів введення−виведення;

- 1 команду із зверненням до FlashROM;

- 22 команди операцій з бітами в розрядах регістрів загального призначення і регістрів введення−виведення;

- 34 команди управління ходом програми.

До системи команд мікроконтролерів типу t11, t 12 t15, 1200 і t28, у яких немає SRAM, не входять команди із зверненням за адресою в адресному просторі SRAM за винятком команд з мнемокодами LD Rd, Z і ST Z, Rr, по яких проводиться звернення до регістрів загального призначення і регістрів введення−виведення з використанням непрямої адресації. До системи команд у цих мікроконтролерів не входять також 2

команди регістрових операцій  і дві команди управління ходом програми. У мікроконтролерів типу 1200 до системи команд не входить також команда із зверненням до FlashROM.

До системи команд мікроконтролерів типу ml63 і ml03 входять додаткові команди. Поява деяких з них пов'язана із збільшеним обсягом FlashROM, а у мікроконтролера типу ml63, крім того, з наявністю апаратного помножувача.

При розробці програми роботи мікроконтролера окрім мнемокодів команд використовуються директиви асемблера і інші засоби асемблера.

Таблиця 3.1 – Команди операцій з бітами

| № | Операція | Мнемокод команди | № | Операція | Мнемокод команди |
|---|---|---|---|---|---|
| 1 | T-Rd.b | BLD Rd, b | 4 | Rr.b-T | BST Rr, b |
| 2 | 0-PrP.b | CBI P, b | 5 | 1-PrP.b | SBI P, b |
| 3 | 0-SREG.b | BCLR b | 6 | 1-SREG.b | BSETb |

d, r = 0—31; P = 0—31 (!); b = 0—7

Машинні коди всіх команд мають формат «слово». Команди №№ 5 і 6 виконуються за 2 такти, решта команд — за 1 такт. У табл. 3.1 використовується таке нове позначення:

**PrP**.b – розряд b (b = 0 – 7) регістра введення−виведення з номером P (P = $00 – $1F), біт в розряді PrP.b.

За командами №№ 3 і 4 виконується пересилка (копіювання) біта між вказаним розрядом регістра загального призначення і розрядом T регістра SREG.

За командами №№ 5 і 6 встановлюється в необхідний стан (0 або 1, відповідно) вказаний розряд регістра введення−виведення, а за командами №№ 7 і 8 — вказаний розряд регістра SREG.

Під час запису мнемокодів команд №№ 5 і 6 замість номера регістра (P) може бути вказано його символічне ім'я, а замість номера розряда (b) - символічне ім'я розряду. У мікроконтролерах сімейства AVR розряди багатьох регістрів введення–виведення мають штатні імена. У мікроконтролерах деяких типів розряди однойменних регістрів з однаковими іменами мають різні номери.

При використанні штатних імен регістрів введення−виведення і штатних імен розрядів в них необхідно використовувати версію AVR Асемблера для мікроконтролера відповідного типу.

Під призначенням штатного імені розряду потрібно розуміти те, що цей розряд належить певному регістру введення–виведення, проте в мнемокоді команд №№ 5 і 6 необхідно вказувати і ім'я/номер регістра і ім'я розряду.

Необхідне значення біта (0 або 1) в розрядах регістрів введення−виведення з номерами від $20 до $3F встановлюється з використанням команд регістрових операцій з мнемокодами **CBR** і **SBR,** відповідно.

### 3.1.3 Програмування мікроконтролера

В лабораторній використовується програмування МК за допомогою інтерфейсу **ISP** (In System Programming). Даний інтерфейс програмування дозволяє програмувати МК, які вже змонтовані в схему приладу. При використанні такого інтерфейсу для програмування готових МК потрібно врахувати в схемі приладу ту умову, що виводи, які використовуються для програмування, не повинні бути приєднані до корпусу.

### 3.1.4 Призначення та можливості STK-500

В лабораторній роботі для програмування і налаштування програми використовується **STK-500**. STK500 – завершений стартовий набір і система проектування для AVR флеш-мікроконтролерів корпорації Atmel. Він розроблений для проектувальників, які бажають швидко розпочати розробку програмного коду і виконати тестування нової розробки.

**Можливості STK-500**

STK-500 має такі можливості:

- сумісність із програмою **AVR Studio;**
- зв'язок із ПК через інтерфейс RS-232 для програмування і керування;
- стабілізоване джерело живлення з входом 10 – 15 В;
- 8-вивідні, 20-вивідні, 28-вивідні, 40-вивідні панелі для установки DIP-корпусів AVR-мікроконтролерів;
- підтримка паралельного і послідовного програмування підвищеною напругою всіх AVR-мікроконтролерів;
- послідовне внутрішньо-системне програмування (ISP) усіх AVR-пристроїв;
- внутрішньо-системний програматор для програмування мікроконтролера безпосередньо в готовому приладі;
- перепрограмування AVR-мікроконтролерів;
- 8 кнопок загального призначення;
- 8 світлодіодів загального призначення;
- всі порти уведення–виведення виведені на роз’єми;
- додатковий порт RS-232 загального призначення;
- роз’єми розширення для підключення зовнішніх модулів і плат для макетування.

Системне програмне забезпечення STK-500 підтримує різні типи мікроконтролерів з урахуванням їх швидкодії.

Таблиця 3.2 – Типи мікроконтролерів з урахуванням їх швидкодії

| | | |
|---|---|---|
| ATtiny11 | AT90S1200 | AT86RF401 |
| ATtiny12 | AT90S2313 | |
| ATtiny15 | AT90S2323 | |
| ATtiny22 | AT90S2333 | |
| ATtiny26 | AT90S2343 | |
| ATtiny28 | AT90S4414 | |
| | ATmega323 | |
| | ATmega8515 | |
| | ATmega8535 | |
| | ATmega64 (1) | |
| | ATmega103 (1) | |
| | ATmega128 (1) | |

Крім того, STK500 підтримує ISP-програмування мікроконтролерів AT89S51 і AT89S52.

Набір STK500 постачається з мікроконтролером AT90S8515-8PC на панелі з маркуванням SCKT3000D3. Вихідні установлення перемичок забезпечують роботу мікроконтролера разом з тактовим генератором і стабілізатором напруги, установленими на платі STK500.

Треба використовувати 10-провідний шнур з комплекту для підключення штирів роз’єму з маркуванням «PORTB» з роз’ємом позначеним, як «LEDS», а потім аналогічно треба з'єднати роз’єми «PORTD» з «SWITCHES». Результат виконання з'єднань зображено на рис. 3.1. На рис. 3.1 показані: 1 – вимикач живлення, 2 – роз’єм живлення, 3 – статусний світлодіод.

Для роботи потрібне зовнішнє джерело живлення 10–15 В. Вхідна схема виконана як двопівперіодний випрямляч, тому STK500 розпізнає полярність прикладеної напруги.

Приєднайте джерело живлення до STK500. Переміщення перемикача живлення дозволяє увімкнути або вимкнути STK500. Світіння червоного світлодіода сигналізує про подачу живлення, а стан статусного світлодіода буде змінюватися від червоного до жовтого, а потім до зеленого. Зелений колір світлодіода сигналізує про наявність напруги живлення VCC.

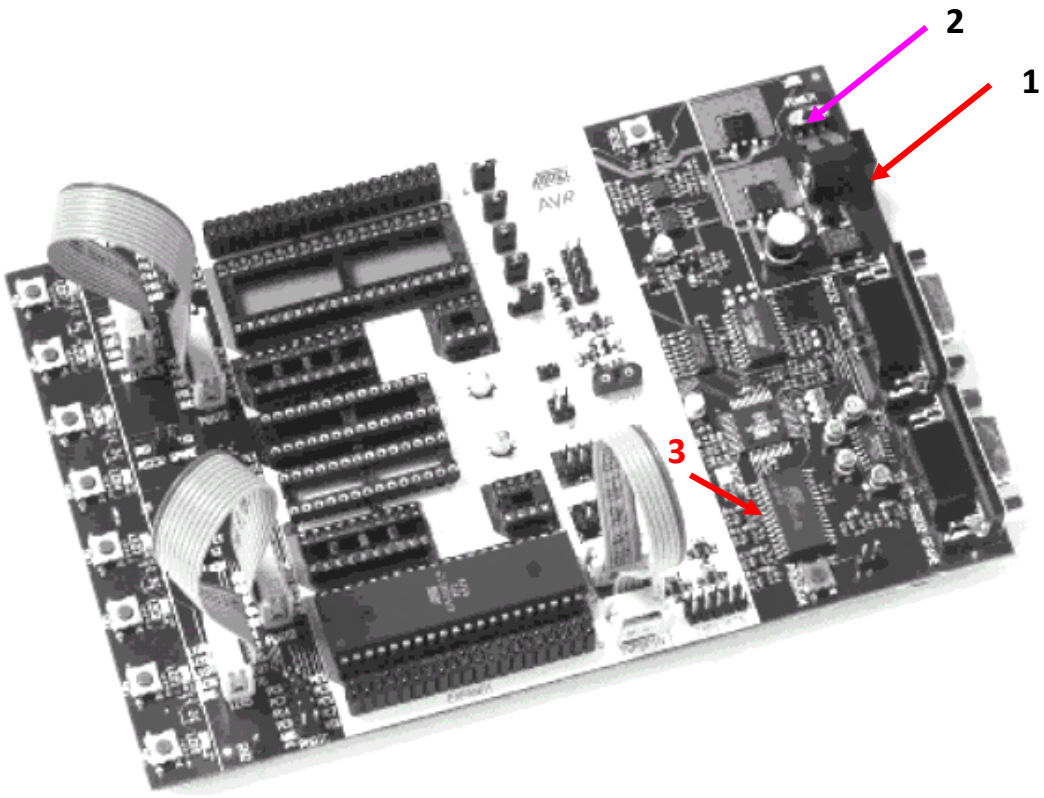



Рисунок 3.1 – Початкові налаштування набору STK500

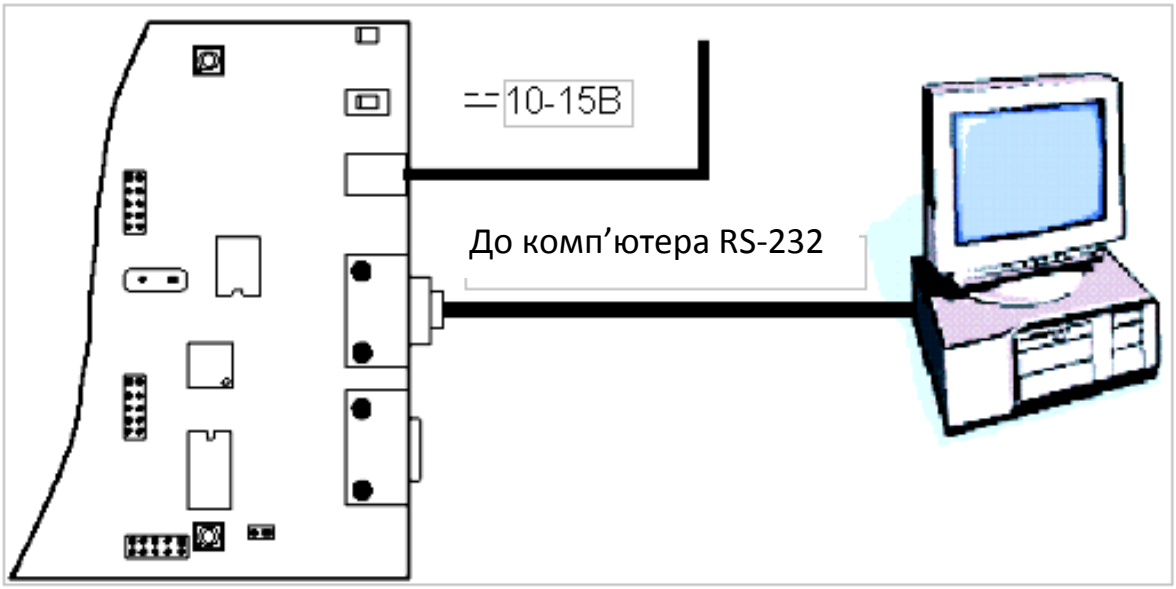



Рисунок 3.2 – Підключення до STK500

До набору STK500 входять 8 жовтих світлодіодів і 8 кнопок без фіксації. Світлодіоди і кнопки електрично відділені від іншої частини плати за рахунок приєднання до власних роз’ємів.

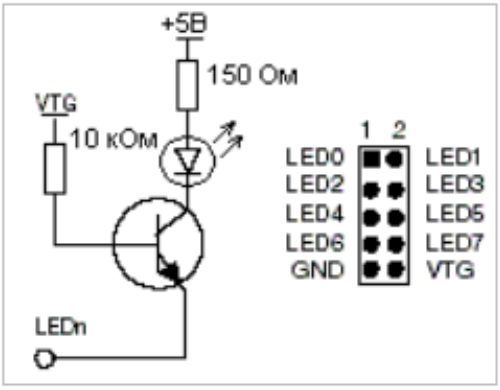



Рисунок 3.3 – Схема приєднання світлодіода

Таким чином, вони можуть бути приєднані до AVR-мікроконтролера через 10-провідний шнур і роз'єм портів введення−виведення. На рисунку 3.3 показано як світлодіоди і кнопки можуть бути приєднані до роз'ємів портів введення−виведення. Шнури мають бути підключені безпосередньо між роз'ємами портів і роз'ємами світлодіодів або кнопок. Шнур не повинен скручуватися. Червоний провід шнура вказує на 1 вивід. Переконайтеся, що шнур приєднаний до першого виводу кожного роз'єму. На рисунку 3.3 показано як реалізовано управління світлодіодом. Таке рішення дозволяє отримати однакову інтенсивність світіння світлодіода при напрузі живлення мікроконтролера в діапазоні від 1,8 до 6,0 В.

Порти AVR-мікроконтролерів можуть керувати безпосередньо світлодіодним навантаженням, у якому струм протікає як в одному напрямку, так і в іншому. Однак, у STK500 використовуються транзистор і два резистори для підтримки постійної яскравості світіння світлодіодів при будь-якому значенні напруги живлення мікроконтролера (VTG), а також для вимикання світлодіодів, коли VTG відсутня.

Кнопки підключені до роз'єму за схемою, зображеною на рисунку 3.4. При натисканні на кнопку на виводі $SW_n$ буде низький рівень напруги, а при відпусканні – високий (VTG). Робочий діапазон напруги VTG= =1,8...6,0 В.

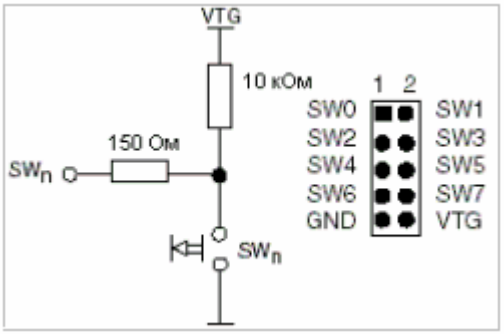



Рисунок 3.4 – Схема приєднання кнопок і підключення до роз'єму

На лініях портів введення–виведення AVR мікроконтролерів є можливість активізації вбудованих резисторів, які приєднані між плюсом джерела живлення і виводом порту. Цю властивість можна використовувати з метою виключення із схеми зовнішнього підтягувального резистора. У STK500 додано зовнішні підтягувальні резистори 10 кОм для формування логічної «1» на виводах $SW_n$ при віджатому стані кнопок. Резистор 150 Ом виконує функцію захисного струмообмежувача (наприклад, у випадку помилкового налаштування ліній введення-виведення, приєднаних до кнопок) на виводі.

Кожний порт введення−виведення AVR-мікроконтролера може бути підключений до світлодіодів і кнопок, використовуючи 10-провідний шнур, що входить до складу набору. На роз'єм додатково до ліній портів виводяться напруга живлення мікроконтролера VTG (VCC) і загальний провід GND.

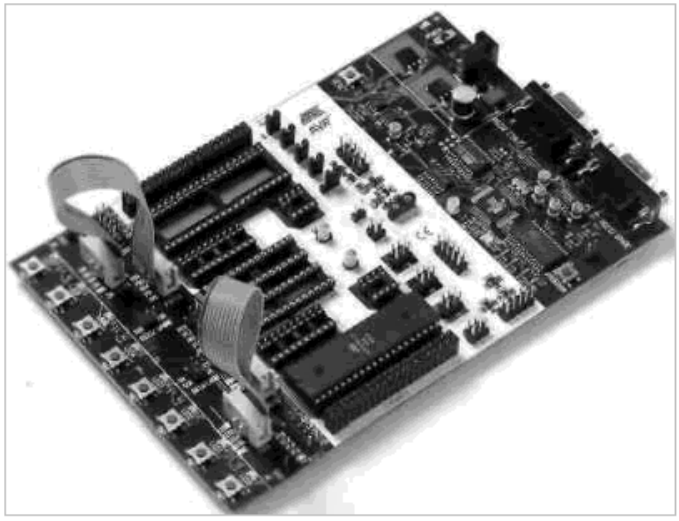


Рисунок 3.5 – Підключення світлодіодів і кнопок до портів введення–виведення

Розташування виводів роз'єму і їхня відповідність лініям портів введення–виведення показані на рисунку 3.6. Вивід із квадратним маркуванням указує на вивід 1.

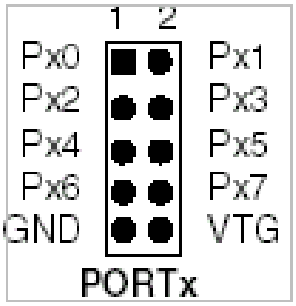
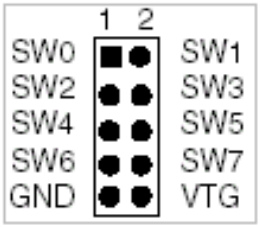


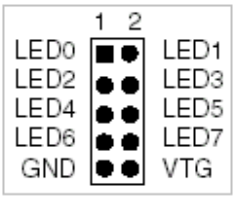

Рисунок 3.6 – Розташування і призначення виводів роз'єму портів введення–виведення

Роз’єм підключення кнопок і світлодіодів мають аналогічне роз’ємам портів введення-виведення розташування і призначення виводів, що показано на рисунках 3.7 і 3.8, відповідно. Квадратне маркування вказує на вивід 1.




SW0-SW7- Кнопка 0-7

Рисунок 3.7 – Розташування і призначення виводів роз’єму кнопок




LED0 – LED7 – світлодіоди 0 – 7

Рисунок 3.8 – Розташування і призначення виводів роз’єму світлодіодів

Внутрішнє системне програмування використовує вбудований у AVR-МК послідовний периферійний інтерфейс SPI для завантаження коду у флеш-пам’ять і **ЕЗППЗП** мікроконтролера. ISP-програмування потребує підключення кіл VCC, GND, RESET і трьох сигналів програмування. Усі AVR-мікроконтролери, за винятком AT90C8534, ATtiny11 і ATtiny28, підтримують ISP-програмування. Програмування може бути виконане при нормальній робочій напрузі, зазвичай 2.7–6.0 В. Сигнали з підвищеними рівнями в даному випадку не потрібні. ISP-програматор програмує як вбудовану флеш-пам’ять програм, так і ЕСППЗП для збереження даних. Він також дозволяє програмувати конфігураційні біти (fuse) для вибору налаштувань тактування, тривалості затримки при старті і порогу детектора зниження напруги (BOD) для більшості мікроконтролерів.

Оскільки інтерфейс програмування розміщається на різних виводах для різних типів мікроконтролерів реалізовано три роз’єми для коректного розведення сигналів програмування. 6-провідний шнур з комплекту STK-500 використовується для з'єднання ISP-сигналів з роз’ємом ISP-програмування МК. Кольорове кодування і номер роз’єму використовуються для визначення, який роз’єм ISP-програмування цільового МК використовується для кожної панелі.

У процесі ISP-програмування 6-пров. шлейф повинен бути постійно зв'язаний з роз’ємом, позначеним як «ISP6PIN». Якщо програмувальний пристрій знаходитися в синій панелі, то під'єднайте інший кінець шлейфа до синього роз’єму ISP-програмування SPROG1. Якщо програмувальний

пристрій знаходиться в зеленій панелі, то використовуйте зелений роз’єм SPROG2.

6-провідний шлейф повинен з'єднувати безпосередньо роз’єм ISP6PIN з роз’ємом цільового ISP-програмування SPROG. Шлейф не повинен перекручуватися. Кольорове маркування проводу шлейфа вказує на вивід 1. Переконайтеся в правильності з'єднань роз’ємів.

### 3.2 Самостійна підготовка

Під час самостійної підготовки необхідно:

1. Ознайомитись із призначенням STK-500;
2. Ознайомитись із будовою МК AT90S8515 та ATmega8515;
3. Навчитись програмувати МК за допомогою STK-500.

### 3.2.1 Завдання для самостійної роботи

Вивчіть алгоритм і код програми та запрограмуйте МК AT90S8515 або ATmega8515 так, щоб МК (розташований в STK-500) після увімкнення живлення виводив число у двійковій системі числення на індикатори STK-500, періодично збільшуючи його на одиницю. При натисканні на кнопку «1» і утримуванні її у натисненому стані потрібно виводити число, яке буде періодично зменшуватись на одиницю.

### 3.2.2 Виконання завдання

Ознайомтесь з алгоритмом програми, який наведений на рис. 3.9.

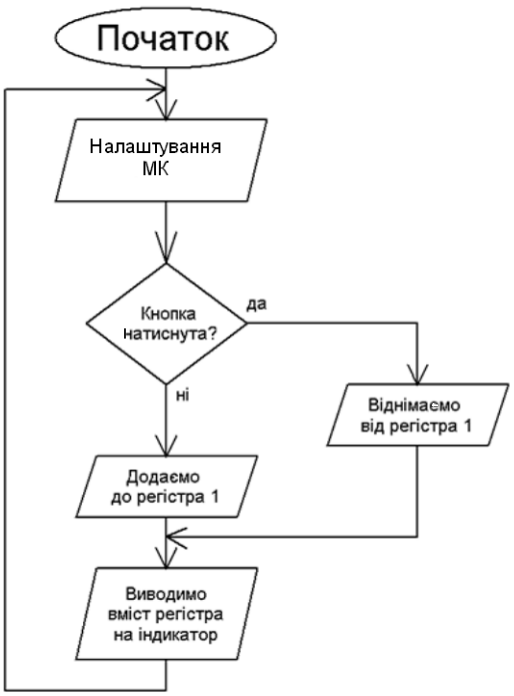



Рисунок 3.9 – Алгоритм програми

Для зміни типу контролера скористайтесь порадами додатку Г.

Цей алгоритм реалізовано в такому коді програми:

```
.include «8515def.inc»
;клавіатура PORTA
;світлодіоди PORTB
ldi r16,0xff
out ddrb,r16
out porta,r16

ldi r17,low(ramend)
out spl,r17
ldi r17,high(ramend)
out sph,r17

start:
sbic pina,0
inc r18
sbis pina,0
dec r18
out portb,r18
rcall delay
rjmp start

delay:
ldi r21,0xff            ;Затримка
zatr1:
ldi r20,0xff
zatr:
dec r20
cpi r20,0x00
brne zatr
dec r21
cpi r21,0x00
brne zatr1
ret
```

Як видно із коду програми, в перших рядках програми налаштовуються порти (порт А і В налаштовані на вихід), і виводиться на порт А одиниці. Це свідчить про те, що між виводами порту і позитивною шиною живлення вмикається резистор. Тепер, якщо кнопки на порті розташувати таким чином, що у замкненому положенні кнопка забезпечує на виводі порту нульовий потенціал, то при розмиканні кнопки на виводі порту встановиться високий потенціал. Цей високий потенціал забезпечить запис 1 в регістр даних.

Запис програми в пам’ять МК AT90S8515

Виконайте дії із створення проекту в послідовності, яка наведена в лабораторній роботі 2. Для цього записуємо текст програми у вікно редактора коду AVR Studio (**потрібна версія AVR Studio 4.11.b401**). Компілюємо програму. Якщо компіляція пройшла без помилок, то переходимо до програмування МК.

На платі STK-500 з’єднайте роз’єми: **Switches** з **PortA**, **Leds** з **PortB** 10-провідними шнурами з комплекту, а роз’єми (для програмування) **ISP6PIN** з **SPROG3** 6-провідним шнуром. З'єднаєте шнур послідовного зв'язку між роз’ємом з маркуванням «**RS232 CTRL**» на платі і роз’ємом **COM**-порту персонального комп'ютера, як показано на рис. 3.2. Увімкніть вимикач живлення «**Power**» на платі STK-500.

Для того, щоб запрограмувати hex-файл у AVR-мікроконтролер, виконуємо команду з меню “Tools” програми AVR Studio (рис. 3.10).

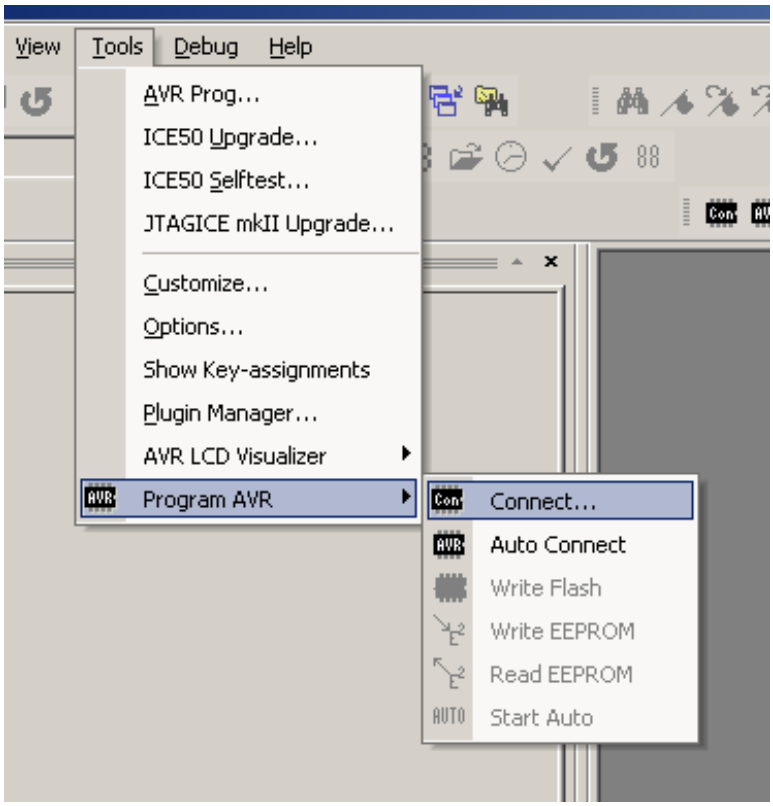



Рисунок 3.10 – Команда меню «Tools»

Натисніть «Program AVR», «Connect» після чого виберіть тип програматора у вікні Select AVR Programmer як показано на рисунку 3.11.

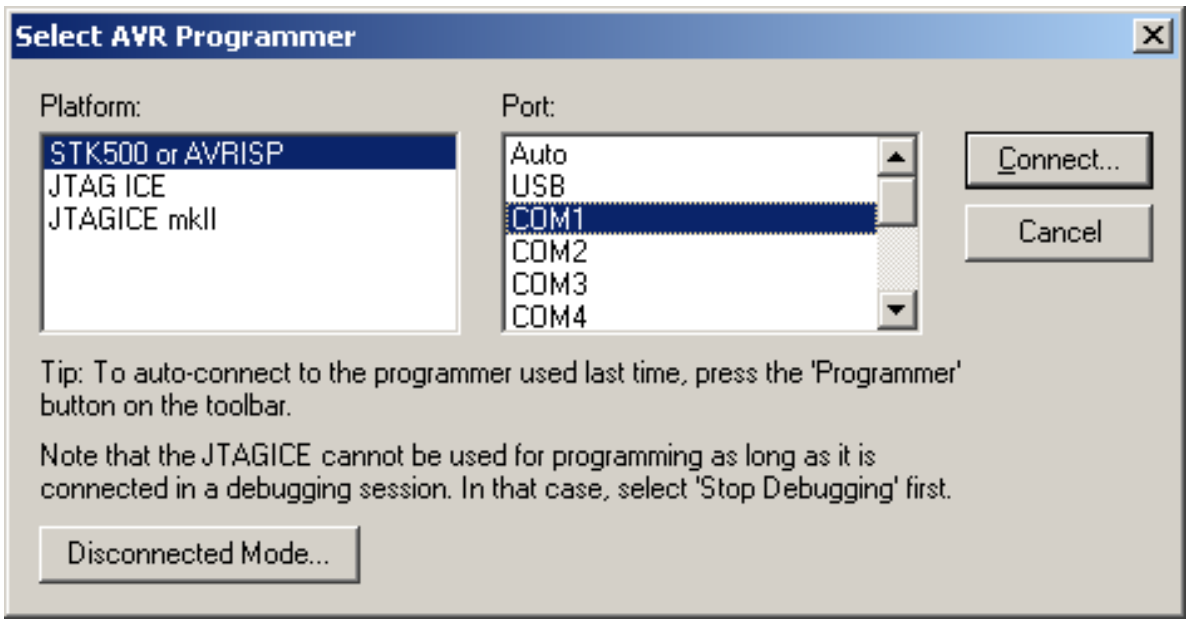



Рисунок 3.11 – Вибір програматора

Натисніть кнопку «**Connect**» і отримайте повідомлення, яке показане на рисунку 3.12.

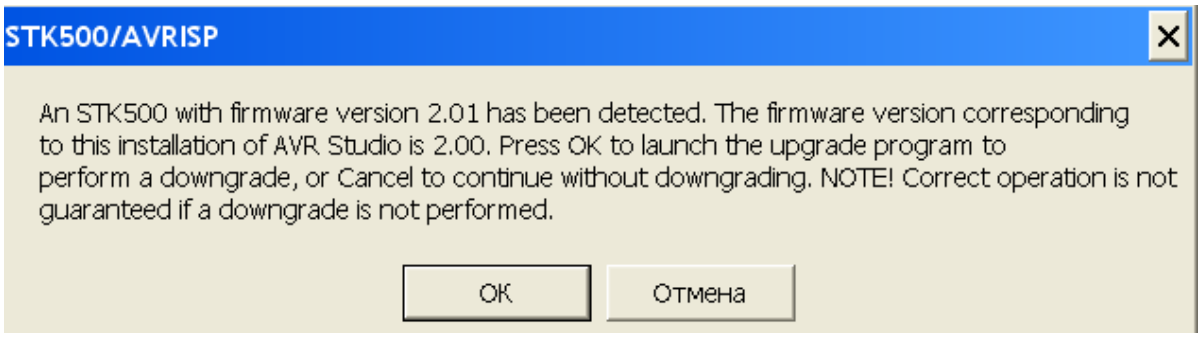



Рисунок 3.12 – Повідомлення після натискання кнопки «Connect»

У повідомленні, яке показано на рис. 3.12, говориться про те, що був знайдений пристрій STK500 з мікропрограмною версією 2.01. Мікропрограмна версія, яка відповідає нині діючій версії програми AVR Studio, має бути 2.00. Натисніть «**ОК**», якщо ви бажаєте встановити модернізовану версію програми, або відмініть заміну існуючої версії програми на новішу, щоб продовжувати роботу без заміни версії програми. Однак, пам’ятайте про те, що коректна робота програми не гарантована, якщо не здійснити заміну версії програмного забезпечення.

Натисніть кнопку «**Отмена**» і перейдіть у віконну заставку «**Disconnected Mode – Open Connection Dialog to Reconnect**» або «**STK500**» залежно від версії програми AVR Studio (рис. 3.13 та рис. 3.14).

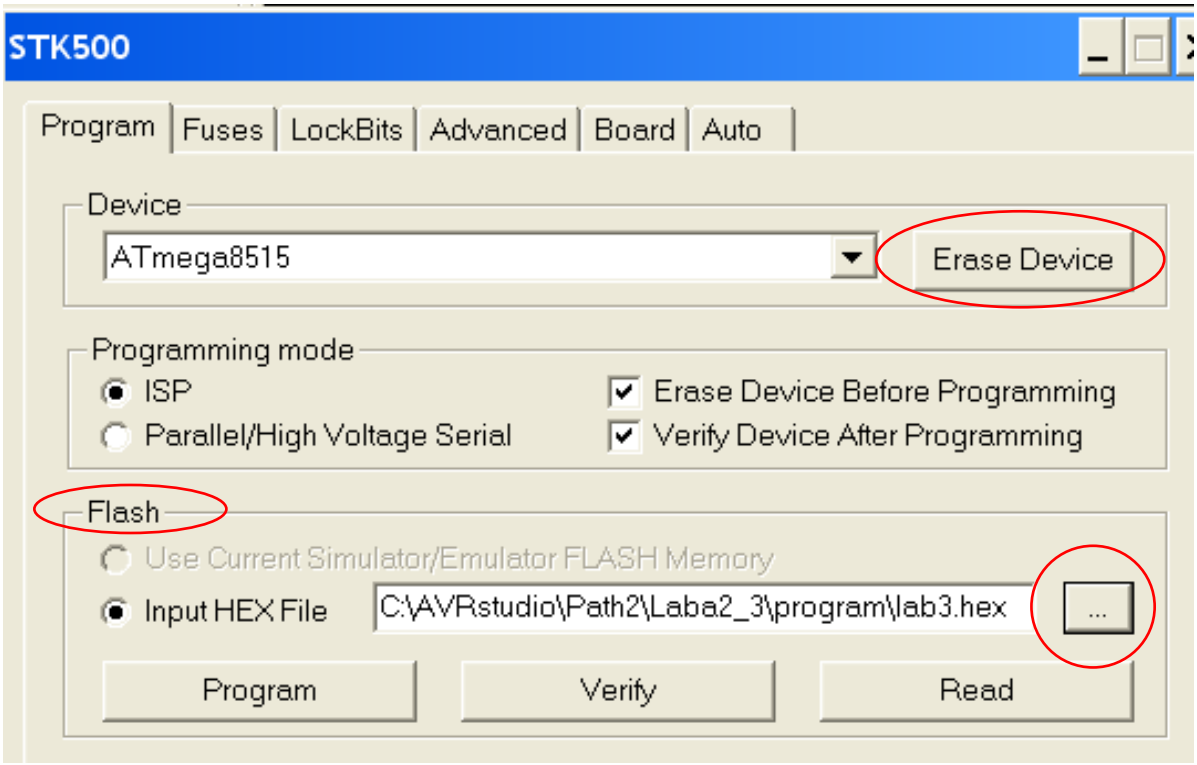



Рисунок 3.13 – Віконна заставка STK500 для AVR Studio 4.11.b401

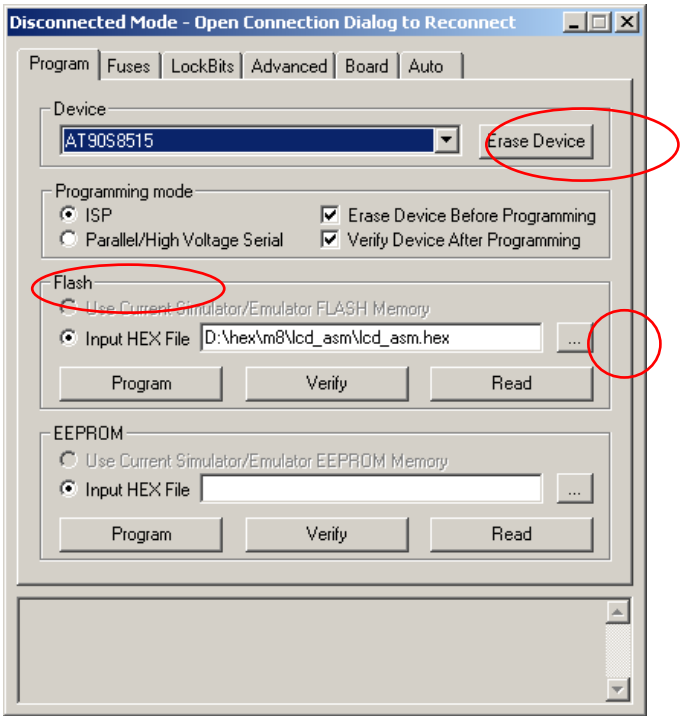



Рисунок 3.14 – Віконна заставка STK500 для AVR Studio 4.11.b410SP3

Виберіть тип AVR-мікроконтролера зі списку «Device», що розкривається, на закладці «Program» і вкажіть шлях до записуваного intel-hex-файлу в полі «Input HEX File» (рис. 3.13, рис. 3.14).

Натисніть кнопку «Erase Device» (стирання МК), розташовану на вкладці програмування «Program» (див. рис. 3.13, рис. 3.14) і отримайте повідомлення, яке показане на рис. 3.15.

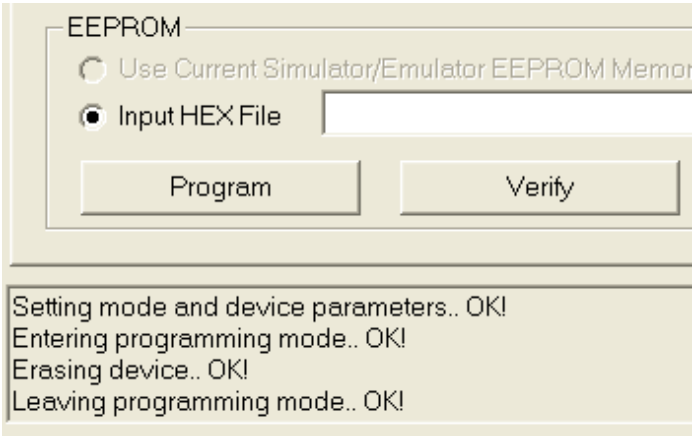



Рисунок 3.15 – Повідомлення про успішне стирання програми з «Flash» пам’яті мікроконтролера

Перейдіть в поле «Flash» (див. рис. 3.13, рис. 3.14) і введіть шлях до потрібного вам файлу, але з розширенням «hex». Для цього натисніть на кнопку «…» (див. рис. 3.13, рис. 3.14) і виберіть шлях до потрібного вам файлу (рис. 3.16).

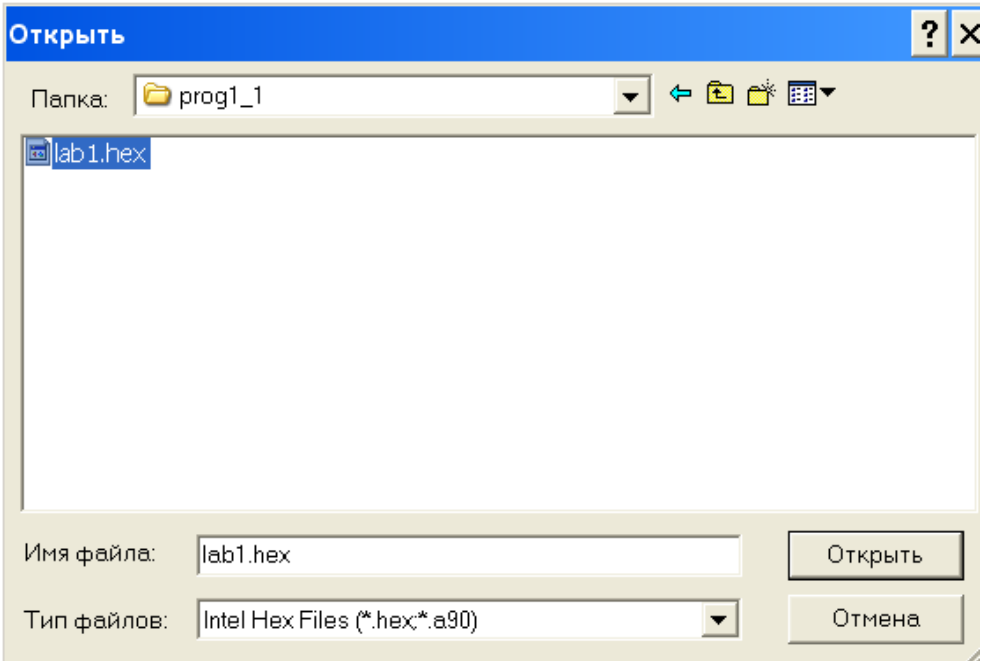



Рисунок 3.16 – Вибір шляху до файлу

Після вибору шляху до файлу, натисніть кнопку «**Открыть**» (рис. 3.16).

Перейдіть на вкладку «Advanced» (рис. 3.17), натисніть «Read Signature» і в полі «Colibrate for frequency:» виберіть «1 Mhz».

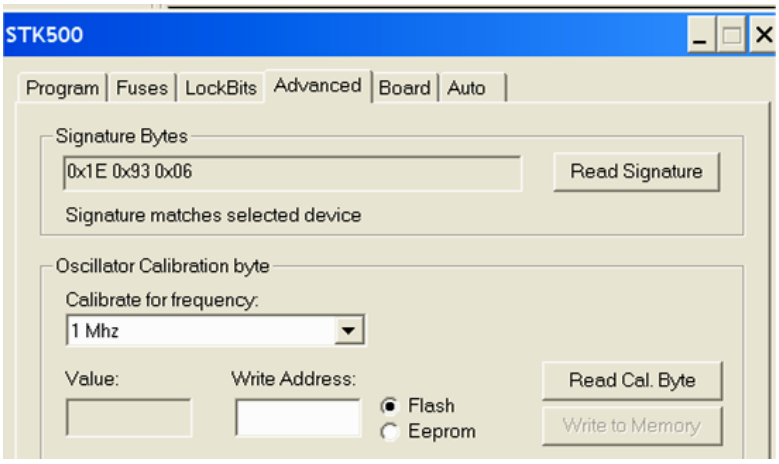



Рисунок 3.17 – Віконна заставка «STK500»

На вкладці «Board» мають бути параметри, які показані на рис. 3.18. Зверніть увагу на параметр «ISP Freq.». Перевірено, що при частоті передавання даних по протоколу «ISP» – 28,36 кГц STK500 працює без помилок.

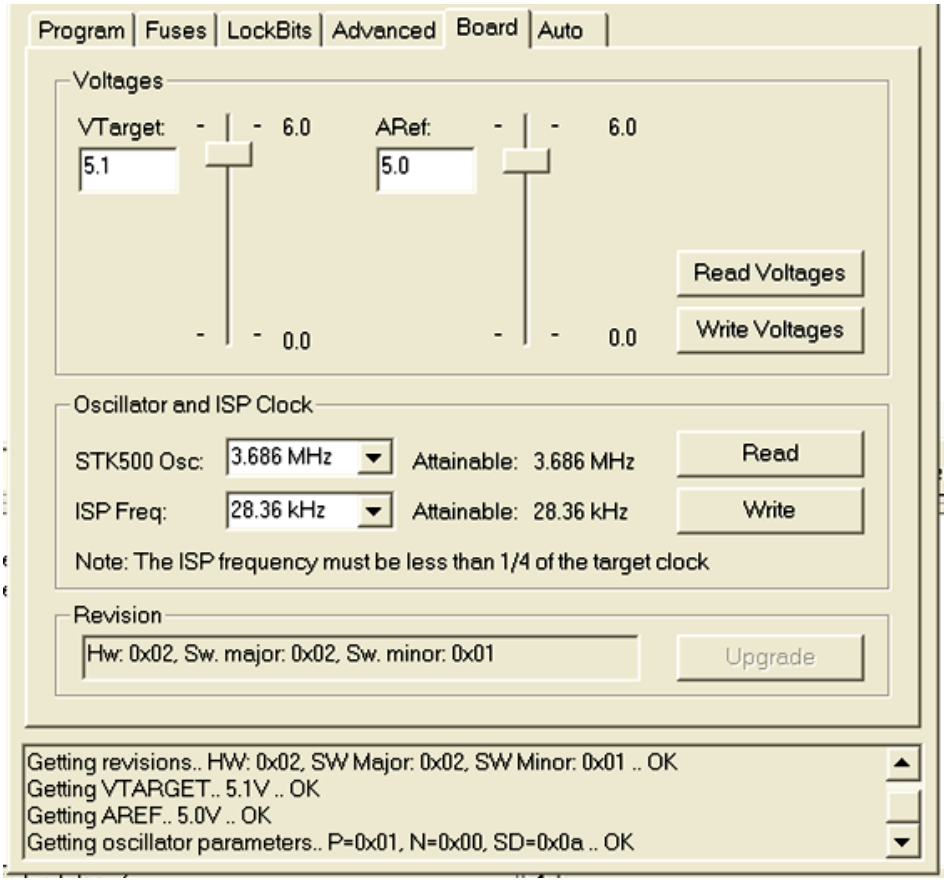



Рисунок 3.18 – Параметри вкладки «Board»

Частоту передавання даних по протоколу «ISP» − 28,36 кГц, STK500 можна змінити на 4 кГц, однак робити це потрібно лише в разі необхідності, наприклад, якщо передавання даних відбувається нестабільно.

Далі перейдіть на вкладку «Auto» і виберіть пункт «Programm FLASH».

Натисніть кнопку «Start» і отримайте повідомлення про успішне завершення програмування (рис. 3.20).

В разі необхідності виберіть пункт «Read FLASH» (рис. 3.21) і прочитайте програму, яка записана у «FLASH» пам'яті мікроконтролера. Далі вам буде запропоновано ввести ім'я файлу та місце на ПЕОМ, куди цю програму можна записати (рис. 3.22).

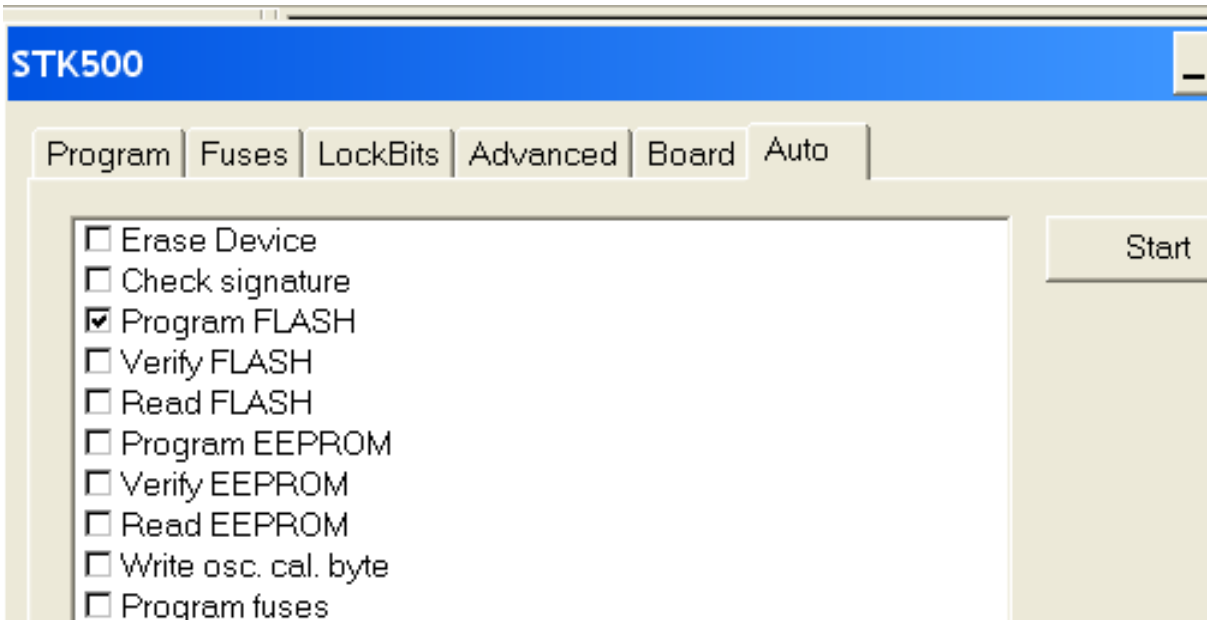



Рисунок 3.19 – Вкладка «Auto» віконної заставки «STK500»

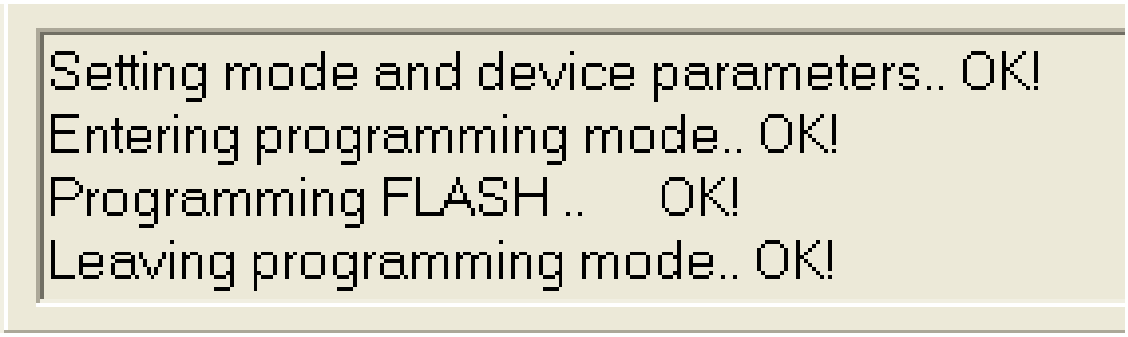



Рисунок 3.20 – Повідомлення про результат успішного програмування

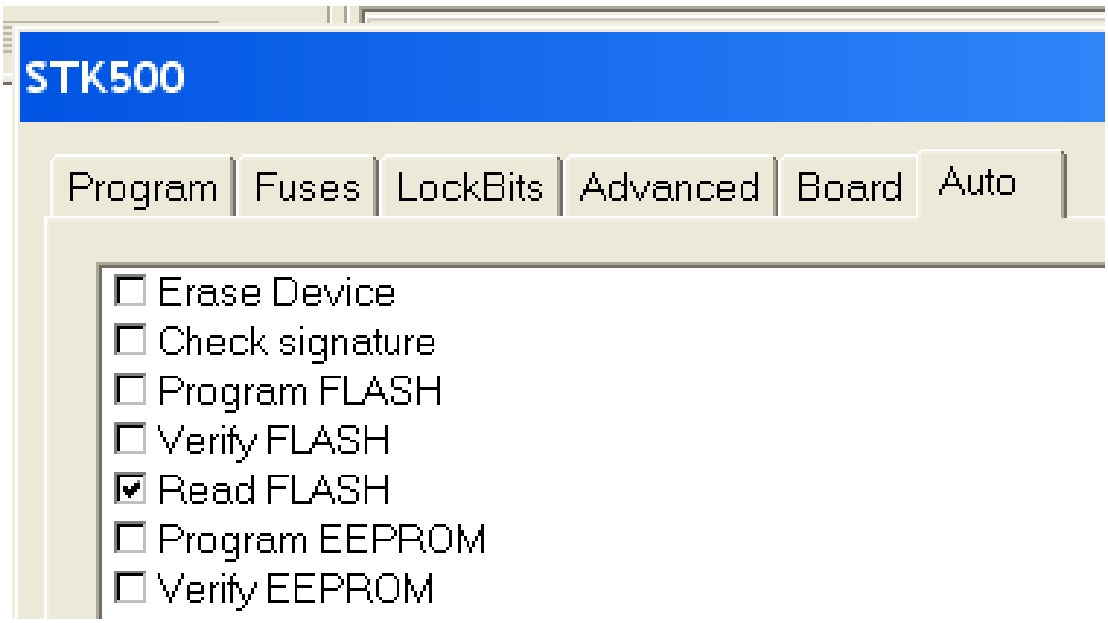



Рисунок 3.21 – Вибір команди зчитування «Read FLASH»

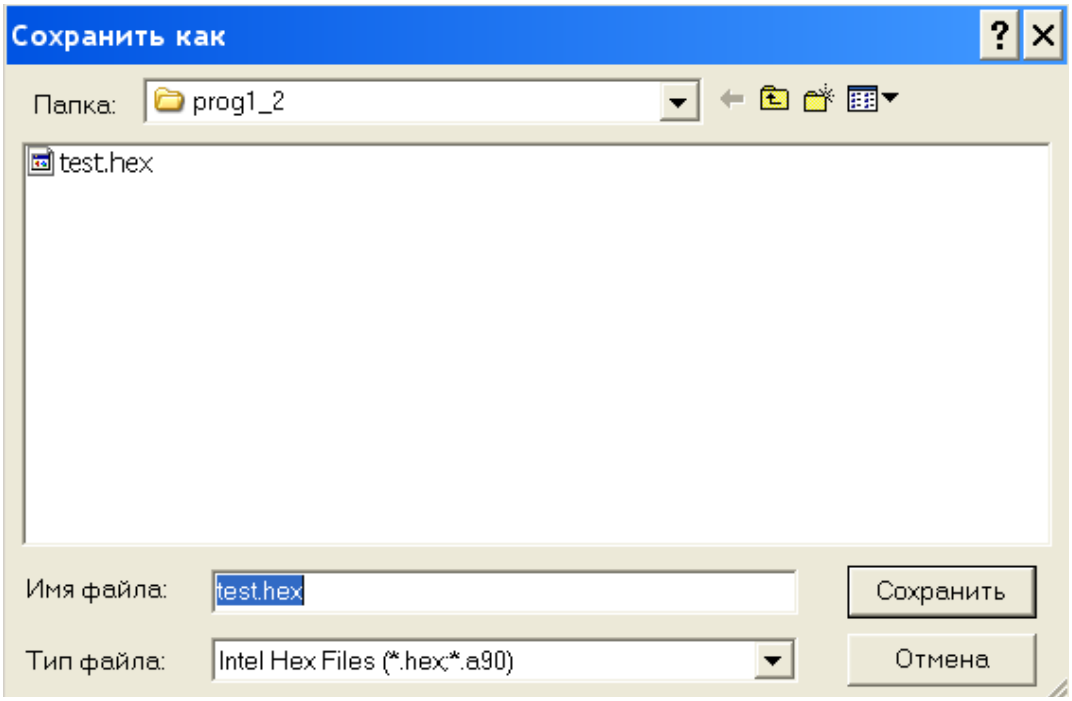



Рисунок 3.22 – Вибір місця для зберігання коду програми

Після успішного програмування відразу запуститься програма на виконання. На панелі світлодіодів з'явиться відображення двійкового числа («1» – світлодіод світиться, «0» – світлодіод не світиться). Число буде змінюватися в сторону збільшення шляхом додавання одиниці до попереднього. При натисканні на кнопку SW0 на платі STK500, число буде змінюватись за законом віднімання.

Код програми наведено в таблиці 3.3.

Таблиця 3.3 – Код програми

| Команда | Коментар |
|---|---|
| .include «8515def.inc» | Підключення бібліотеки 8515def.inc, в якій адресам МК присвоєні відповідні імена |
| ;клавіатура PORTA<br>;світлодіоди PORTB | Коментарі програми |
| ldi r16**,0xff** | Завантажуємо в регістр r16 константу 0xFF |
| out ddrb,r16 | Виводимо в регістр стану порту B значення 0xFF, налаштовуючи всі виводи порту на виведення |
| out porta,r16 | Включаємо «підвішувальні» резистори на порту A |
| ldi r17,low(ramend)<br>out spl,r17<br>ldi r17,high(ramend)<br>out sph,r17 | Проводимо налаштування програмного стека, записуючи в нього кінець пам'яті програми «ramend». Запис проводиться в два прийоми, оскільки адреса 16-розрядна |
| start: | Мітка |
| sbic pina,0 | Пропустити наступну команду, якщо нульовий біт регістра стану порту A очищений |
| inc r18 | Збільшити вміст регістра 18 на 1 |
| sbis pina,0 | Пропустити наступну команду, якщо біт 0 в регістрі стану встановлений |
| dec r18 | Зменшити вміст регістра 18 на одиницю |
| out portb,r18 | Вивести вміст реістра 18 на порт B |
| rcall delay | Виклик підпрограми затримки для формування паузи при індикації числа |
| rjmp start | Повернення на початок програми |
| delay:<br>ldi r21,**0xff** ;Затримка<br>zatr1:<br>ldi r20,0xff<br>zatr:<br>dec r20<br>cpi r20,0x00<br>brne zatr<br>dec r21<br>cpi r21,0x00<br>brne zatr1<br>ret | |
| out portb,r18 | Вивести вміст регістра 18 на порт B |

### 3.3 Хід роботи

1. Ознайомтеся з будовою і призначенням основних частин STK-500.
2. Напишіть програми роботи із портами відповідно до індивідуального завдання.
3. Відкомпілюйте програму у програмному комплексі AVRStudio.
4. Запрограмуйте програму в МК.

5. Запустіть програму на виконання на платі розробки STK-500 (під наглядом викладача) і переконайтеся у роботоздатності програми.
6. Зробіть висновки.
7. Оформіть звіт.

### 3.4 Зміст звіту

1. Алгоритм програми.
2. Текст програми.
3. Результати роботи програми.
4. Висновки.

### 3.5 Контрольні запитання

1. Як побудовані порти МК AVR фірми Atmel?
2. Які вам відомі регістри МК AVR фірми Atmel?
3. За допомогою яких команд проводяться операції з портами?
4. Що вам відомо про базову систему команд?
5. Які команди не входять до базової системи команд мікроконтролерів типу t1l, t 12 t15, 1200 і t28, у яких немає SRAM?
6. Які вам відомі команди операцій з бітами?
7. Що вам відомо про інтерфейс ISP?
8. Що вам відомо про можливості STK-500?
9. Що вам відомо про призначення STK-500?
10. Що входить до набору STK500?
11. Що використовує внутрішнє системне програмування?
12. Як записати програму в пам’ять МК AT90S8515?

# 4 Лабораторна робота № 4

# ПІДПРОГРАМА І СТЕК

**Мета роботи**: ознайомитись із принципом будови підпрограми і використання при цьому стека підпрограм.

## 4.1 Короткі теоретичні відомості

У МК AVR наявні два типи стека – програмний стек і стек даних. Програмний стек надає можливість створювати переходи до підпрограм і після виконання підпрограм повертатися в початкове положення. При зверненні основної програми до підпрограми МК проводить запис у програмний стек адреси, яка знаходиться у програмному лічильнику. Після закінчення виконання підпрограми проводиться відновлення із стека вмісту програмного лічильника. Далі проводиться додавання одиниці до попередньо збереженого у стеку значення.

Стек даних використовується для тимчасового збереження вмісту регістрів. Така необхідність виникає при потребі використання регістрів у підпрограмі, але за умови, що раніше ці регістри вже використовувалися у основній програмі. Тоді для використання їх у підпрограмі їхній вміст заносять в стек і після виконання підпрограми проводять відновлення вмісту відповідних регістрів. Слід пам’ятати те, що при організації записуванні і читання стека в ньому зберігаються дані за принципом «перший прийшов – перший вийшов». Тобто при записуванні в стек вмісту регістрів 5, 12 і 8, відновлення проводиться у зворотній послідовності 8, 12 і 5.

Програма 4.1 за допомогою світлодіодів STK 500 імітує «біжучі вогні». Якщо до порту підключити світлодіоди і виводити на них певну послідовність сигналів, то світлодіоди, блимаючи, будуть створювати відповідний ефект. Враховуючи те, що частота тактового генератора дорівнює 4 МГц, то частота відображення ефекту дорівнює приблизно 4 МГц. Враховуючи інерційність людського ока, потрібно встановити затримку між відображеннями окремих елементів ефекту, тобто частину програми, яка створює паузи між окремими елементами відображення. При написанні такої програми після кожного елементу потрібно встановлювати таку затримку. Тобто код програми збільшиться на обсяг коду затримки, помноженого на кількість елементів відображення. Тобто при великій кількості символів, що відображаються, буде великий розмір коду. Для того, щоб зменшити величину коду, в програмі створюють підпрограми, до яких звертається головна програма. Таким чином, код програми, затримки записаний в пам’ять МК один раз і до нього проводиться звернення кожен раз, коли потрібно виконати затримку.

Під час виконання в МК програми, кожна наступна команда викликається відповідно до адреси, яка записана в лічильнику команд. При виконанні програми з лінійним алгоритмом, кожне наступне значення адреси знаходиться шляхом додавання одиниці до попереднього значення адреси. Якщо ж виконується програма з розгалужуваннями, то наступне

значення адреси лічильника команд отримується теж шляхом додавання одиниці до попереднього (при виконанні лінійної частини програми). Якщо програма «зустрічає» розгалуження, то наступне значення адреси береться із коду програми (з аргументу команди розгалуження). При переході за адресою розгалуження починає виконуватися підпрограма (в нашому випадку підпрограма затримки). Після завершення виконання підпрограми проводиться повернення на місце, з якого відбувся виклик підпрограми. Для того, щоб процесор «знав» куди йому повернутись після виконання підпрограми, проводиться записування в область пам'яті (стек) адреси, яка відповідає теперішньому значенню адреси плюс один. Після цього програма забезпечує виклик підпрограми. Текст програми наведено в таблиці 4.1.

Таблиця 4.1 – Код програми 4.1

| Команди | Коментарі | |
|---|---|---|
| .include «8515def.inc» | Підключення бібліотеки для МК AT90S8515 | |
| ldi r16,low(RAMEND); | Завантаження в регістр r16 молодшої частини 16-розрядної адреси кінця пам'яті програм МК | Ініціалізація стека |
| out spl,r16 | Виведення вмісту регістра r16 в молодшу частину регістра стека | |
| ldi r16,high(ramend) | Завантаження в регістр r16 старшого розряду адреси кінця пам'яті програм МК | |
| out sph,r16 | Виведення вмісту регістра r16 в старшу частину регістра стека | |
| ldi r16,$ff | Завантаження в регістр r16 константи 0xff | Налаштування порта D на виведення |
| out ddrd,r16 | Виведення вмісту регістра r16 в регістр керування порту D (DDRD) | |
| ldi r16,0xaa | Завантаження в регістр r16 константи 0xAA(0b10101010) | |
| ldi r17,0x55 | Завантаження в регістр r17 константи 0x55(0b01010101) | |
| **start:** | Мітка «**start**». Символізує початок основної програми | Основна програма |
| out portd,r16 | Виводимо в регістр даних порту D вміст регістра r16 | |
| rcall **delay** | Виклик підпрограми «**delay**» | |
| out portd,r17 | Виводимо в регістр даних порту D вміст регістра r17 | Основна програма |
| rcall delay | Виклик підпрограми «delay» | |
| rjmp **start** | Перехід до мітки «**start**» | |

Продовження таблиці 4.1

| Команди | Коментарі | |
|---|---|---|
| delay: | Мітка «delay». Початок підпрограми затримки | Підпрограма «delay» |
| push r16 | Збереження в стек вмісту регістра r16 | |
| push r17 | Збереження в стек вмісту регістра r17 | |
| ldi r16,$02 | Завантаження в регістр r16 числа 0x02 | |
| zovn: | Мітка «zovn» | |
| ldi r17,$02 | Завантаження в регістр r17 числа 0x02 | |
| vnutr: | Мітка «vnutr» | |
| dec r17 | Зменшення на одиницю реістра r17 | |
| brne vnutr | Якщо вміст регістра r17 не дорівнює нулю, то перейти до мітки «vnutr» | |
| dec r16 | Зменшити вміст регістра r16 на одиницю | |
| brne zovn | Якщо регістр r16 не дорівнює нулю, то перейти до мітки «zovn» | |
| pop r17 | Відновити значення регістра r17 зі стека | |
| pop r16 | Відновити значення регістра r16 зі стека | |
| ret | Повернення із підпрограми | |

Для перевірки роботоздатності програми використовуємо STK-500. Компілюємо програму і програмуємо МК (приклад програмування наведено в лабораторній роботі № 2). При відключеному живленні STK-500 підключаємо 10-провідниковим кабелем порт D (роз'єм PORTD) до світлодіодів (роз'єм LED). Після вмикання живлення світлодіоди починають блимати.

### 4.2 Домашня підготовка

1. Ознайомитеся з принципом ініціалізації стека.
2. Розгляньте команди виклику і повернення з підпрограми.
3. Запропонуйте варіанти вирішення задачі утворення затримки.

### 4.3 Хід роботи

1. Ознайомтеся з будовою та призначенням основних частин **STK-500**.
2. Текст коду програми відповідно до завдання вашого варіанта.
3. Скомпілюйте і перевірте правильність програми.

### 4.4 Зміст звіту

1. Напишіть можливі шляхи вирішення поставленої задачі.
2. Алгоритм програми відповідно до свого варіанта.
3. Текст програми з коментарями.

4. Результати роботи програми.
5. Висновки.

### 4.5 Контрольні запитання

1. Що називається стеком і яке його призначення?
2. Які ви знаєте два види стека у МК AVR? За яким принципом зберігаються дані у стеку?
3. Що імітує програма 4.1 в лабораторній роботі?
4. Поясніть призначення команд в коді програми лабораторної роботи № 4?
5. Як програмно створити затримку на світіння світлодіодів?
6. Яким шляхом знаходиться кожне наступне значення адреси при виконанні програми з лінійним алгоритмом?
7. Яким шляхом знаходиться кожне наступне значення адреси при виконанні програми з розгалуженнями?
8. Як здійснюється перевірки роботоздатності програми?
9. Як ініціалізувати стек?
10. Які ви знаєте команди виклику і повернення з підпрограми?
11. Запропонуйте варіанти вирішення задачі утворення затримки.

# 5 Лабораторна робота № 5

# ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ ДОДАВАННЯ ТА ВІДНІМАННЯ

**Мета роботи**: отримати навички у написанні програм під МК для вирішення задач віднімання і додавання.

## 5.1 Короткі теоретичні відомості

Організація додавання і віднімання чисел на базі МК організовується при використанні команд відповідно віднімання і додавання. Тільки слід враховувати під час написання програм, що МК містить 8-розрядні регістри, тобто може оперувати числами, які не перевищують 255 (0xff). При потребі проведення операцій із більшими числами проводимо використання біта переповнення і додатково один чи більше регістрів. Тобто при додаванні потрібно відслідковувати стан біта переповнення і відповідного до цього стану проводити виконання відповідних операцій.

Особливістю є те, що при відніманні двох чисел меншого і більшого відслідковувати від'ємний знак результата можна при встановленні біта нуля. Тобто при «переході через нуль» буде встановлюватись біт нуля і, як результат, програма «буде знати» про від'ємність результата.

Приклад програми, яка виконує додавання двох 8-розрядних чисел без перенесення, наведено у таблиці 5.1.

Таблиця 5.1 – Текст програми 5.1 додавання двох 8-розрядних чисел

| Команди | Коментарі |
|---|---|
| .include «8515def.inc» | Підключення бібліотеки |
| .def operand_1=r16 | Присвоєння назви «operand_1» регістру r16 |
| .def operand_2=r17 | Присвоєння назви «operand_2» регістру r17 |
| lds operand_1,$60 | Завантажуємо в operand_1 вміст комірки даних із адресою $60 |
| lds operand_2,$61 | Завантажуємо в operand_2 вміст комірки даних із адресою $61 |
| add operand_1,operand_2 | Додаємо вміст регістра operand_2 до вмісту регістра operand_1. |
| sts $63,operand_1 | Зберігаємо вміст регістра operand_1(результат додавання) за адресою $63 |

Ця програма проводить читання числа з пам'яті даних за адресою 0x60 і 0x61. Після додавання проводить записування за адресою 0x63. Програма не проводить перевірку на переповнення. Тобто при результаті додавання, який буде перевищувати 255, результат не буде відповідати реальному значенню.

Для перевірки роботи програми використовуємо програму AVRStudio (див. ЛР № 2).

Скомпілювавши програму, проводимо емуляцію її роботи. Перейдемо в меню «Project», «Build and Run» (рис. 5.1).

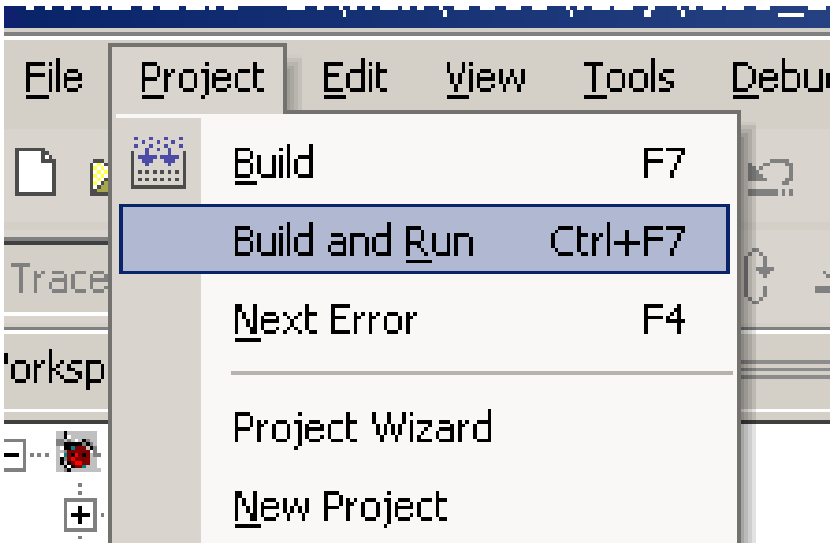



Рисунок 5.1 – Запуск на виконання програми

Виконуємо команду меню «View», «Memory» (рис. 5.2).

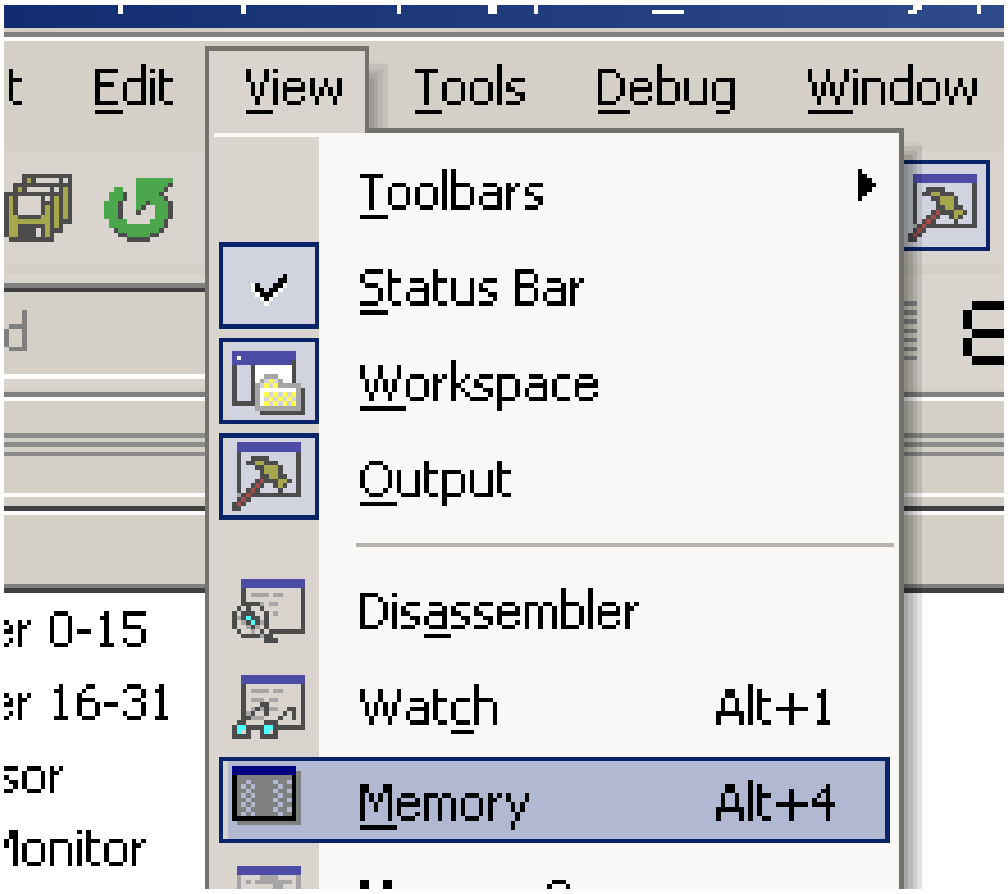



Рисунок 5.2 – Меню відображення

У вікні «Memory» у лівому низхідному меню змінюємо значення «Program» на «Data».

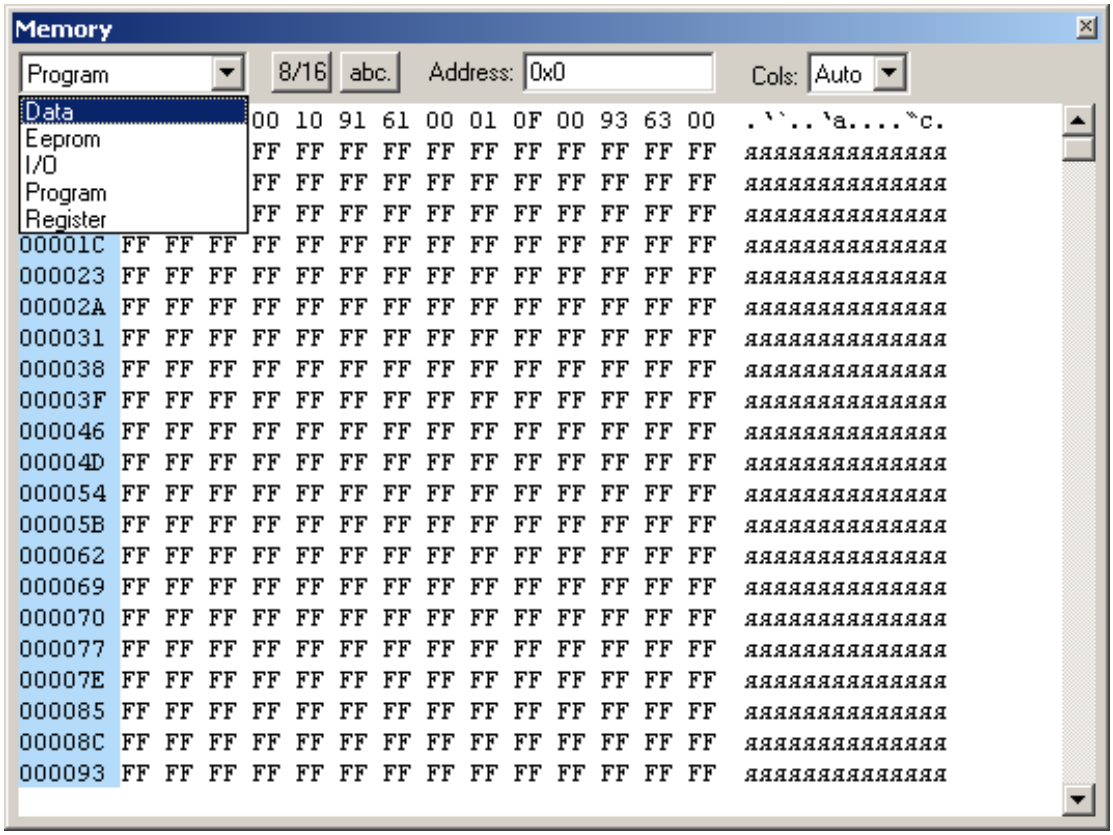



Рисунок 5.3 – Вікно відображення стану пам’яті МК

Змінюємо вміст комірки з адресою 60 на 02, а вміст комірки за адресою 61 на 04 (рис. 5.4).

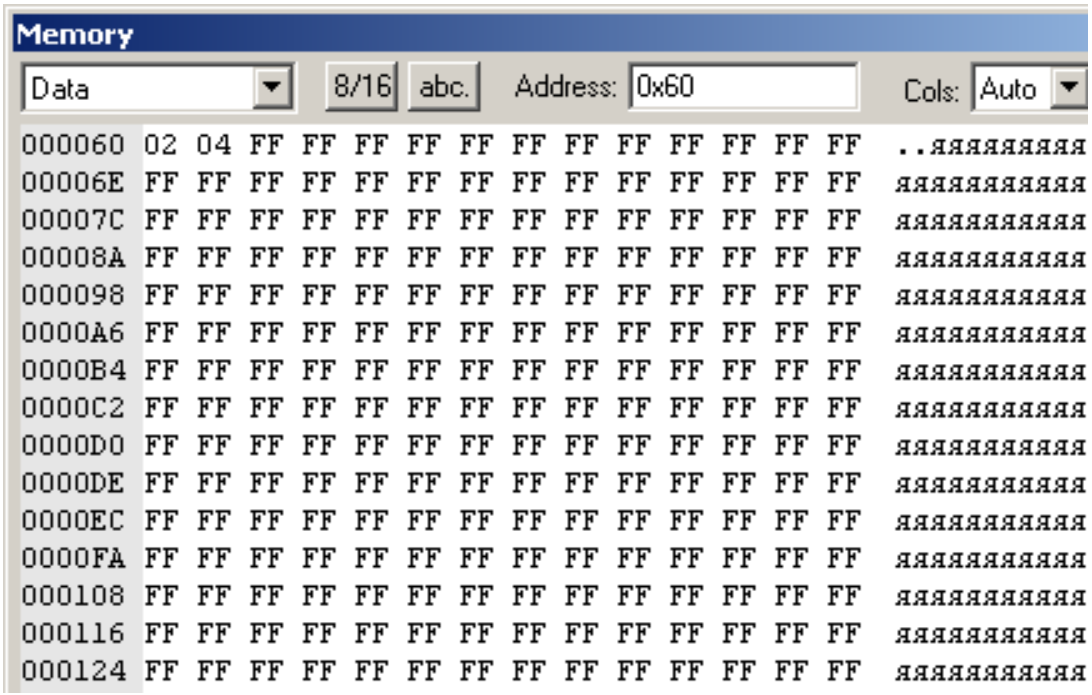



Рисунок 5.4 – Вигляд пам’яті даних до виконання програми

Запускаємо програму на виконання в покроковому режимі. Для цього натискаємо кнопку F11 на клавіатурі ПК і слідкуємо за змінами у вікні «Memory» (рис. 5.5).

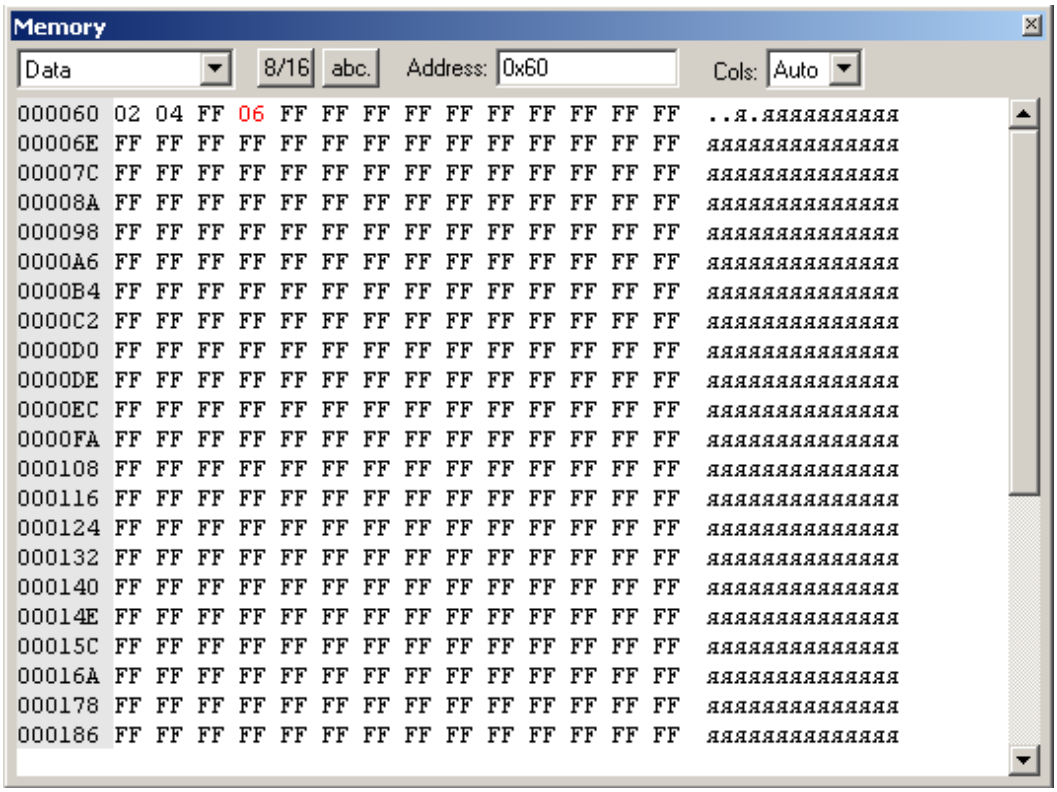



Рисунок 5.5 – Результат роботи програми

### 5.2 Домашня підготовка

1. Ознайомитись із командами асемблера для виконання арифметичних дій (додаток Д).
2. Ознайомитись із призначенням біта переповнення C і біта нульового результату Z.
3. Запропонувати можливі шляхи вирішення задачі відповідно до індивідуального завдання.
4. Розробити алгоритм за одним із варіантів шляху вирішення задачі. Пояснити вибір такого шляху вирішення задачі.

### 5.3 Хід роботи

1. Написати програму згідно із індивідуальним завданням.
2. Виконання програму в емуляторі AVRStudio.
3. Порівняти отримані результати із завданням.
4. Зробити висновки.

### 5.4 Зміст звіту

1. Алгоритм програми відповідно до індивідуального завдання.
2. Текст програми із коментарями.
3. Результат виконання програми.

### 5.5 Контрольні запитання

1. За допомогою яких команд здійснюється віднімання і додавання чисел?
2. Що потрібно враховувати під час написання програми на віднімання і додавання чисел?
3. Що потрібно використовувати в разі операцій з великими числами під час написання програми на віднімання і додавання чисел?
4. Яка є особливість в написанні програм на віднімання чисел?
5. Поясніть призначення команд коду програми на додавання двох 8-розрядних чисел (код програми 5.1).
6. Нарисуйте блок-схему алгоритму програми 5.1.
7. Поясніть призначення команд коду програми 5.2 (додаток Д).
8. Нарисуйте блок-схему алгоритму програми 5.2 (додаток Д).
9. Поясніть призначення команд коду програми 5.3 (додаток Д).
10. Нарисуйте блок-схему алгоритму програми 5.3 (додаток Д).
11. Які ви знаєте команди асемблера для виконання арифметичних дій?
12. Яке призначення біта переповнення C і біта нульового результату Z?

# 6 Лабораторна робота № 6
# ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ МНОЖЕННЯ ТА ДІЛЕННЯ

**Мета роботи:** отримати навички із виконання операцій ділення і множення на базі МК AVR.

## 6.1 Короткі теоретичні відомості

**Множення чисел**. Існує декілька алогритмів множення чисел. При першому алгоритмі множення можна замінити багаторазовим додаванням, наприклад, $14 \cdot 3 = 14 + 14 + 14$. Існує один недолік цього способу – значна тривалість процесу обчислення. При другому алгоритмі множення здійснюється в стовпець. Цей алгоритм застосовується і для множення двійкових чисел, наприклад

$0110 = 6_{10}$
$0011 = 3_{10}$

0110
0110
0000
0000

$00010010 = 18_{10}$

При обчисленні результату за другим алгоритмом необхідно здійснити багаторазове додавання зі зсувом вліво множеного при одночасній перевірці вмісту розрядів множника, починаючи зі сторони його розряду. При цьому якщо в черговому розряді множника записана 1, то множене додається до суми і зсувається вліво на один розряд, а якщо в розряді записаний 0, то відбудеться тільки зсув множеного. Зсув множеного вліво можна замінити зсувом суми вправо.

Програма−приклад завантажує число із пам’яті даних і виконує додавання до початкового пустого регістра доданку, що відповідає зсунутому першому множнику вправо. Додавання не відбувається, якщо розряд другого множника відповідає нулю. Перевірка «на нуль» проводиться із застосуванням маскування. Тобто множимо на число, в якому один тільки біт дорівнює одиниці і перевіряємо результат на нуль.

Текст коду програми−прикладу наведено в таблиці 6.1. Ця програма проводить множення двох восьмирозрядних чисел, які знаходяться у комірках пам’яті даних 0x60 і 0x61. Результат множення зберігається в пам’яті даних за адресою 0x6e.

Таблиця 6.1 – Код програми 6.1 множення

| .include «8515def.inc» | Включення бібліотеки |
|---|---|
| lds r16,0x60 | Завантаження в регістр r16 вмісту комірки пам’яті даних з адресою 0x60 |
| lds r17,0x61 | Завантаження в регістр 17 вмісту комірки пам’яті даних за адресою 0x61 |
| clr r19 | Очистити регістр r19 |
| ldi r20,0b00000001 | Завантажити в регістр r20 константу 0b00000001 |
| multi: | Мітка |
| mov r18,r17 | Копіювання вмісту регістра r17 у регістр 18 |
| and r18,r20 | Логічне І регістра r18 і регістра r20 |
| rol r16 | Зсув регістра r16 вліво на одиницю |
| rol r20 | Зсув регістра r20 вліво на одиницю |
| cpi r20,0 | Порівняння вмісту регістра r20 із константою |
| breq end | Якщо однакові, то перейти за міткою «end» |
| cpi r18,0 | Порівняння вмісту регістра r18 із константою 0 |
| breq multi | Якщо однакові, то перейти за міткою «multi» |
| add r19,r16 | Додавання до вмісту регістра r19 вміст регістра r16 (без переносу) |
| cpi r20,0 | Порівняння вмісту регістра r20 із константою 0 |
| brne multi | Якщо не однакові, то перейти за міткою «multi» |
| end: | Мітка «end» |
| ror r19 | Зсув вмісту регістра r19 вправо |
| sts 0x6e,r19 | Збереження вмісту регістра r19 в комірку пам’яті даних з адресою 0x6e |

Написавши програму, виконуємо її компілювання (див. ЛР № 2). Для отримання результатів виконання програми використовується симулятор програми AVRStudio (ЛР № 5).

## 6.2 Хід роботи

1. Написати програму відповідно до завдання.
2. Скомпілювати програму.
3. Виконати програму в симуляторі AVRStudio.
4. Для перевірки роботи програми перейти в головне меню, потім на вкладку Veiw і потім на Memory.
5. У вікні Memory змінити Program на Data.

6. В комірках за адресами 0x60 та 0x61 вписати числа, які будуть множитись (краще 02 та 03).

7. Виконати програму, а саме: Project – Build and Run – Memory – Data – F11 (F11 повторюється до закінчення виконання програми).

8. Прочитати результат (він має з'явитися в комірці 0x6E, наприклад, 06).

9. Такі самі дії виконати з програмою ділення.

10. Зробити висновки щодо роботоздатності програми за результатам роботи програми.

### 6.3 Домашня підготовка

1. Ознайомитись команди для виконання арифметичних дій.
2. Ознайомитись із призначенням бітів переповнення і переходу через нуль.
3. Запропонувати можливі способи вирішення задачі.
4. Вибрати один із методів і навести для нього алгоритм.

### 6.4 Зміст звіту

1. Алгоритм програми.
2. Текст програми.
3. Результат роботи програми.
4. Висновки.

### 6.5 Контрольні запитання

1. Які ви знаєте алгоритми множення чисел?
2. Напишіть код програми множення числа 02 на число 05 за першим (запропонованим в описі лабораторної роботи) алгоритмом.
3. Напишіть код програми множення числа 03 на число 03 за другим (запропонованим в описі лабораторної роботи) алгоритмом.
4. Нарисуйте блок-схему алгоритму програми множення двох чисел (числа 03 на число 03) за другим (запропонованим в описі лабораторної роботи) алгоритмом.
5. Поясніть роботу програми 6.1.
6. Нарисуйте блок-схему алгоритму програми 6.1.
7. Поясніть призначення команд коду програми 6.1.
8. Поясніть роботу програми 6.2 (додаток Е).
9. Нарисуйте блок-схему алгоритму програми 6.2 (додаток Е).
10. Поясніть призначення команд коду програми 6.2 (додаток Е).
11. Які ви знаєте команди для виконання арифметичних дій?
12. Яке призначення бітів переповнення і переходу через нуль?

# 7 Лабораторна робота № 7
# ПРИЄДНАННЯ КЛАВІАТУРИ І ДИСПЛЕЯ ДО МК

**Мета роботи**: вивчення програмно-апаратних методів приєднання дисплея і клавіатури до МК.

## 7.1 Короткі теоретичні відомості

У раніше запропонованих лабораторних роботах проводилось виконання різних операції за допомогою МК, а про результат виконання програми дізнавалися із станів бітів регістрів, що відображав на екрані ПК емулятор МК. Тобто, якщо запрограмувати МК попередніми програмами і увімкнути STK-500, то ми нічого не побачимо без застосування спеціальних пристроїв. Для того, щоб побачити результат виконання програми, потрібно використати пристрої виведення інформації. Такі пристрої називаються індикаторами, тобто за допомогою них ми можемо отримати інформацію про стан МК або результат виконання, чи невиконання операції.

МК не завжди працює із пам’яттю. Він може отримувати інформацію про зовнішнє середовище за допомогою сенсорів. Такими сенсорами можуть бути як примітивні кнопки, так і складні системи із складними законами перетворення.

В даній лабораторній роботі розглядається робота з семисегментним індикатором як з відображенням даних і з примітивною клавіатурою, як з сенсором.

Семисегментний індикатор є таким індикатором, який складається із семи сегментів (рис. 7.1).

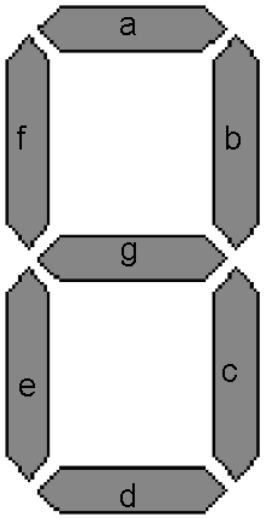



Рисунок 7.1 – Розташування сегментів семисегментного індикатора

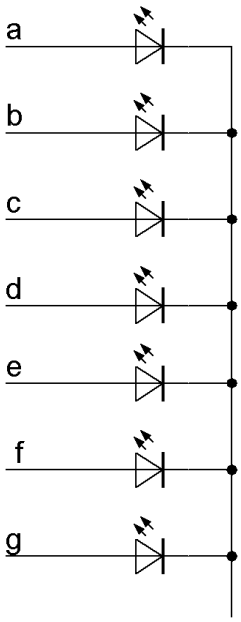



Рисунок 7.2 – Схема семисегментного індикатора із загальним катодом

Кожний сегмент пронумерований літерами латинської абетки від a до g. Якщо розглянути схему такого індикатора, виготовленого на світлодіодах, із загальним катодом, то вона матиме вигляд (рис. 7.2).

Як видно із схеми для відображення певного символа, потрібно подати на відповідні аноди позитивні потенціали, а на катод подати негативні.

Так при підключенні до МК одного семисегментного індикатора використовується 7 виводів МК. Вже при підключенні 2 і більше індикаторів кількість виводів, які будуть використовуватися, буде дорівнювати 7, помноженому на кількість індикаторів (рис. 7.3).

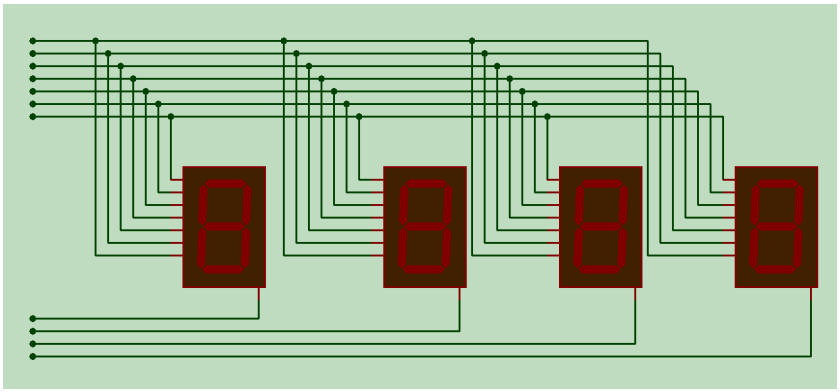


Рисунок 7.3 – Приєднання семисегментних індикаторів

Як бачимо, такий спосіб приєднання не є раціональним. Тому для підключення двох і більше індикаторів до МК використовують так звану динамічну або мультиплексну індикацію. Схематичне рішення такої індикації полягає в тому, що всі індикатори підключаються паралельно виводами a…g і кожний окремо (в нашому випадку) одним виводом –

катодом. В такому випадку кількість задіяних виводів дорівнює семи плюс кількість індикаторів (катодів). Принцип такої індикації полягає в послідовному виведенні інформації на кожен одинарний індикатор окремо і за рахунок інерції людського ока ми бачимо статичне зображення.

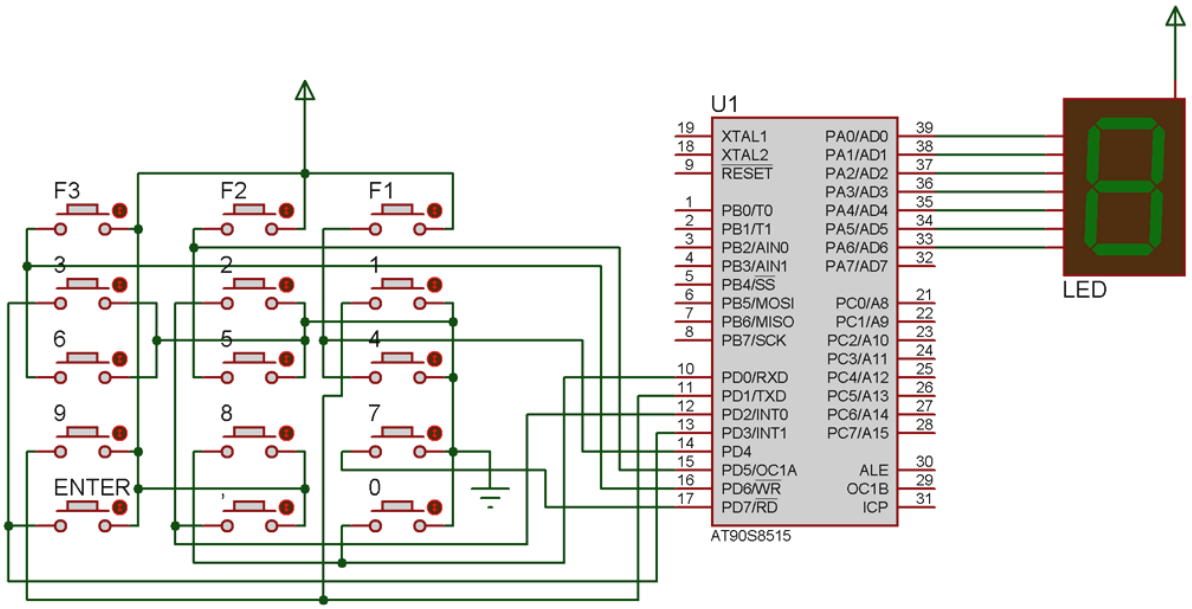



Рисунок 7.4 – Схема для програми прикладу

Приклад програми 7.1, яка здійснює виведення символа натиснутої кнопки на семисегментний індикатор.

Таблиця 7.1 – Код програми для виведення символа натиснутої кнопки на семисегментний індикатор

| | |
|---|---|
| .include «8515def.inc» | Включення бібліотеки |
| .cseg | Початок сегмента коду програми |
| ldi r16,0xff | Завантаження в регістр 16 константи 0xFF |
| out ddra,r16 | Виведення вмісту регістра 16 у регістр напрямку порту A |
| ldi r16,low(ramend) | Завантаження молодшої частини адреси кінця пам’яті програми |
| out spl,r16 | Виведення вмісту регістра 16 в молодшу частину регістра стека |
| ldi r16,high(ramend) | Завантаження старшої частини адреси кінця пам’яті програми |
| out sph,r16 | Виведення вмісту регістра r16 в старшу частину регістра |
| **main:** | Початок основної програми |

Продовження таблиці 7.1

| | |
|---|---|
| rcall **keyboard** | Виклик підпрограми опитування клавіатури |
| rcall **led** | Виклик підпрограми індикації **led** |
| rjmp **main** | Повернення до початку основної програми |
| **led:** | Початок підпрограми індикації **led** |
| clr r30 | Очистити регістр r16 |
| ldi zl,low(led_code*2) | Завантажити в регістр z молодшу частину адреси першої константи, записаної в пам'яті програми і позначеної міткою led_code |
| ldi zh,high(led_code*2) | Завантажити в регістр z старшу частину адреси першої константи, записаної в пам'яті програми і позначеної міткою led_code |
| add r30,r17 | Додавання до вмісту регістра r30 вмісту регістра r17 |
| lpm | Зчитування констант із пам'яті програми |
| out PORTA,r0 | Виведення на порт A вмісту регістра r0 |
| ret | Повернення із підпрограми **led** |
| **keyboard:** | Початок підпрограми оброблення клавіатури |
| clr r16 | Очистити регістр r16 |
| out PORTD,r16 | Вивести на порт D вміст регістра r16 |
| sbic PIND,0 | Якщо в порт D скинуто біт 0, то пропустити наступний рядок |
| ldi r17,8; 8 | Завантажити константу $8_{10}$ в регістр r17 |
| sbic PIND,1 | Якщо в порт D скинутий біт 1, то пропустити наступний рядок |
| ldi r17,9; 9 | Завантажити константу $9_{10}$ в регістр r17 |
| sbic PIND,2 | Якщо в порт D скинуто біт 2, то пропустити наступний рядок |
| ldi r17,0x0a; , | Завантажити константу 0x0A в регістр r17 |
| sbic PIND,3 | Якщо в порт D скинуто біт 3, то пропустити наступний рядок |
| ldi r17,0x0b; enter | Завантажити константу 0x0B в регістр r17 |
| sbic PIND,4 | Якщо в порту D скинуто біт 4, то пропустити наступний рядок |
| ldi r17,0x0c; F1 | Завантажити константу 0x0C в регістр r17 |
| sbic PIND,5 | Якщо в порту D скинуто біт 5, то пропустити наступний рядок |
| ldi r17,0x0d; F2 | Завантажити константу 0x0D в регістр r17 |
| sbic PIND,6 | Якщо в порту D скинуто біт 6, то пропустити наступний рядок |
| ldi r17,0x0e; F3 | Завантажити константу 0x0E в регістр r17 |
| ser r16 | Встановити регістр r16 |
| out PORTD,r16 | Вивести вміст регістра r16 на порт D |
| OUT DDRD,r16 | Вивести вміст регістра r16 в регістр напрямку порту D |

Продовження таблиці 7.1

| | |
|---|---|
| SBIS PIND,0 | Пропустити наступний рядок, якщо біт 0 порту D встановлений |
| LDI r17,0; 0 | Завантажити в регістр r17 константу 0 |
| SBIS PIND,1 | Пропустити наступний рядок, якщо біт 1 порту D встановлений |
| LDI r17,1; 1 | Завантажити в регістр 17 константу 1 |
| SBIS PIND,2 | Пропустити наступний рядок, якщо біт 2 порту D встановлений |
| LDI r17,2; 2 | Завантажити в регістр r17 константу 2 |
| SBIS PIND,3 | Пропустити наступний рядок, якщо біт 3 порту D встановлений |
| LDI r17,3; 3 | Завантажити в регістр r17 константу 3 |
| SBIS PIND,4 | Пропустити наступний рядок, якщо біт 4 порту D встановлений |
| LDI r17,4; 4 | Завантажити в регістр r17 константу 4 |
| SBIS PIND,5 | Пропустити наступний рядок, якщо біт 5 порту D встановлений |
| LDI r17,5; 5 | Завантажити в регістр r17 константу 5 |
| SBIS PIND,6 | Пропустити наступний рядок, якщо біт 6 порту D встановлений |
| LDI r17,6; 6 | Завантажити в регістр r17 константу 6 |
| SBIS PIND,7 | Пропустити наступний рядок, якщо біт 7 порту D встановлений |
| LDI r17,7; 7 | Завантажити в регістр r17 константу 7 |
| ret | Повернення з підпрограми **keyboard** |
| led_code: | Мітка, що вказує на початок констант, збережених в пам’яті програми |
| .db<br>0b00000110,0b11011110,<br>0b00001101,0b00011100,<br>0b11010100,0b00110100,<br>0b00100100,0b01011110,<br>0b00000100,0b00010100,<br>0b01000100,0b10100100,<br>0b00100111,0b10001100,<br>0b00100101,0b01100101 | Константи, збережені в пам’яті програми (використовуються для формування знаків на індикаторі) |

Для перевірки правильності виконання програми використовується STK-500. Після програмування МК (див. ЛР № 2) приєднуємо клавіатуру і індикатор відповідно до порту PORTD і PORTA (див. ЛР № 3), вмикаємо живлення і спостерігаємо результат.

### 7.2 Домашня підготовка

1. Розглянути можливі методи підключення клавіатури і індикатора до МК.

2. Ознайомитись із можливістю підключення зовнішніх клавіатури і індикатора до плати STK-500.

3. Запропонувати програмні і апаратні способи вирішення задачі відповідно до варіанта.

4. Вибрати один із способів і побудувати алгоритм його організації.

### 7.3 Хід роботи

1. Визначити схему підключення клавіатури і індикатора.
2. Написати програму відповідно до варіанта.
3. Відкомпілювати програму і перевірити правильність виконання у програмному емуляторі.
4. Запрограмувати МК.
5. Скласти схему.
6. На платі STK 500 шлейф від PORTA приєднати до роз'єму A1 на платі індикаторів, шлейф від PORTD приєднуємо до плати індикаторів.
7. Перевірити правильність роботи програми на STK 500.
8. Написати звіт з лабораторної роботи.

### 7.4 Контрольні запитання

1. Які вам відомі програмно-апаратні методи підключення дисплея і клавіатури до МК?
2. Що потрібно використати для того, щоб побачити результат виконання програми?
3. Як називаються пристрої, за допомогою яких можна побачити результат виконання програми чи невиконання операції?
4. Звідки МК може отримувати інформацію про зовнішнє середовище?
5. Що може бути сенсорами для отримання мікроконтролером інформації про зовнішнє середовище?
6. Які пристрої введення та виведення інформації використовуються в лабораторній роботі № 7?
7. Поясніть схему підключення семисегментних індикаторів.
8. Поясніть схему підключення клавіатури.
9. Поясніть роботу програми 7.1.
10. Нарисуйте блок-схему алгоритму програми 7.1.
11. Поясніть призначення команд коду програми 7.1.
12. Поясніть роботу програми 7.2 (додаток Ж).
13. Нарисуйте блок-схему алгоритму програми 7.2 (додаток Ж).
14. Поясніть призначення команд коду програми 7.2 (додаток Ж).
15. Які вам відомі методи підключення клавіатури і індикатора до МК?

# 8 Лабораторна робота № 8

# КЕРУВАННЯ ЕЛЕКТРИЧНИМИ ДВИГУНАМИ ЗА ДОПОМОГОЮ AVR МІКРОКОНТРОЛЕРІВ

**Мета роботи:** вивчити методи програмування AVR мікроконтролерів для вирішення задач автоматизації процесів управління електричними двигунами.

## 8.1 Опис апаратного і програмного забезпечення лабораторної роботи

На лабораторному стенді встановлено два двигуни (зі шківом і вентилятор).

Схема стенду дозволяє виконувати реверс двигуном зі шківом і вмикати та вимикати вентилятор.

На стенді розташований роз'єм, через який відбувається керування двигунами. Перших два його піна (0 і 1) відповідають за напрямок обертання двигуна із шківом. Третій пін відповідає за подачу живлення на двигун зі шківом. Четвертий пін керує живленням вентилятора.

Активним рівнем для керування є логічна одиниця (+5В). Тобто, щоб увімкнути вентилятор, потрібно подати на роз'єм відповідно код 00001000 (табл. 8.1).

Таблиця 8.1 – Коди керування двигунами

| Стан двигунів | Керувальний код |
|---|---|
| Двигун зі шківом вперед | 00000101 |
| Двигун зі шківом назад | 00000110 |
| Вентилятор обертається | 00001000 |
| Всі двигуни вимкнені | 00000000 |

Програма керування написана мовою асемблера і має такий вигляд:

```
.include «m8515def.inc»
;програма керування двигунами
ldi r16,$ff
;portA – кнопки
out ddrb,r16;portB – двигун
out portb,r16
start:
sbis pina,0;двигун зі шківом вперед
ldi r17,0b00000101
sbis pina,1;двигун зі шківом назад
ldi r17,0b00000110
sbis pina,2;увімкнути вентилятор
ldi r17,0b00001000
```

```
out portb,r17
rjmp start
```

Ця програма перевіряє стан кнопок, які приєднані до порту А (кнопки SW0, SW1, SW2). При натисканні на кнопку SW0 двигун зі шківом почне обертатися вперед. При натисканні на кнопку SW1 двигун із шківом змінить напрямок обертання. Кнопка SW3 відповідає увімкненню вентилятора. При натисканні будь-якої наступної кнопки попередня команда відміняється.

### 8.2 Домашня підготовка

Розглянути можливі методи підключення електричних двигунів до МК.

Ознайомитись із можливістю підключення стенду з електричними двигунами до плати STK-500.

Запропонувати програмні і апаратні способи вирішення задачі відповідно до варіанта.

Вибрати один із способів і побудувати алгоритм його організації.

### 8.3 Хід роботи

1. Визначити схему підключення стенду з електричними двигунами до плати STK-500.

**Увага!** При приєднанні схеми до STK-500 потрібно звернути увагу на розташування першого піна на платі і на STK-500. На платі двигуна перший пін позначений трикутником, а на платі STK-500 перший пін має назву PB0. При приєднанні зовнішніх елементів до плати STK-500 потрібно слідкувати за тим, щоб червоний край шлейфа був направлений в бік першого виводу.

2. Написати програму відповідно до варіанта.
3. Відкомпілювати програму і перевірити правильність виконання у програмному емуляторі.
4. Запрограмувати МК.
5. Скласти схему.
6. Перевірити правильність роботи програми на STK 500.
7. Натисніть на кнопку SW0, потім на кнопку SW1, а потім на кнопку SW3. Запишіть у звіт реакцію електричних двигунів, які розташовані на лабораторному стенді, на натискання кнопок SW0, SW1, SW2.
8. Написати звіт з лабораторної роботи.

### 8.4 Контрольні запитання

1. Поясніть роботу програми 8.1.
2. Нарисуйте блок-схему алгоритму програми 8.1.
3. Поясніть призначення команд коду програми 8.1.
4. Як підключити джерело живлення до лабораторного стенда?

## Література


1. Гребнев В. В. Микроконтроллеры семейства AVR фирмы Atmel / Гребнев В. В. – М. : ИП РадиоСофт, 2002. – 176 с.
2. Баранов В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / Баранов В. Н. – М. : Издательский дом «Додэка-XXI», 2004. – 288 с.
3. Евстифеев А. В. Микроконтроллеры AVR семейства Classic фирмы ATMEL / Евстифеев А. В. – М. : Издательский дом «Додэка-XXI», 2006. – 288 с.
4. Евстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL» / Евстифеев А. В. – М. : Издательский дом «Додэка-XXI», 2004. – 560 с.
5. Голубцов М. С. Микроконтроллеры AVR: от простого к сложному / Голубцов М. С. – М. : СОЛОН–Пресс, 2003. – 288 с.
6. Петров И. В. ПЗО Программируемые контроллеры. Стандартные языки и инструменты / Петров И. В. ; под ред. проф. В. П. Дьяконова. – М. : СОЛОН-Пресс, 2003. – 256 с.
7. Тавернье К. PIC–микроконтроллеры. Практика применения / Тавернье К. ; пер. с франц. – М. : ДМК Пресс, 2004. – 272 с.
8. Ульрих В. А. Микроконтроллеры PIC16X7XX / Ульрих В. А. – СПб. : Наука и Техника, 2002. – 320 с.
9. Рюмик С. М. Микроконтроллеры AVR. Ступень 1 / Рюмик С. М. // Радиоаматор. – 2005. – № 1. – 35–39 с.
10. Рюмик С. М. Микроконтроллеры AVR. Ступень 5 / Рюмик С. М. // Радиоаматор. – 2005. – № 1. – 35–39 с.
11. Башнин О. И. Микропроцессоры в энергетике / Башнин О. И., Буевич В. В., Каштелян В. Е. – Л. : Наука, Ленинградское отделение, 1982. – 189 с.
12. Рубаненко О. Є. Мікропроцесорна техніка : лабораторний практикум / О. Є. Рубаненко, В. О. Комар. – Вінниця : ВНТУ, 2005. – 83 с.
13. Стогний Б. С. Микропроцесорные системы в электроэнергетике / Б. С. Стогний, В. В. Рогоза, А. В. Кириленко. – Киев : Наукова думка, 1988. – 232 с.

## Додаток А

| | |
|---|---|
| Стек | Особлива, фізично локалізована на кристалі мікропроцесора, група комірок пам'яті даних. Типовий восьмирозрядний мікропроцесор містить вказівник стека – спеціалізований шістнадцятирозрядний регістр – лічильник. Стек типового мікропроцесора буде розташовуватись в ОЗП і його положення визначається програмістом. Вказівник стека завантажується старшою адресою, яка є вершиною стека. |
| Регістр команд | Восьмирозрядний регістр, який містить перший байт команди (її код операції). |
| КОП | Код операції. |
| Дешифратор команд | Це пристрій, який декодує вміст регістра команд, визначає мікрокоманду для виконання потрібної команди із всієї множини команд і послідовно вводить в дію секцію управління. |
| Акумулятор | Універсальний восьмирозрядний регістр, в якому концентрується більшість результатів виконання команд: арифметичних, логічних, завантаження, запам'ятовування результату, введення–виведення. |
| Лічильник команд (PC) | Різновид шістнадцятирозрядної пам'яті, яка постійно вказує на наступну виконувану програму. Він завжди містить шістнадцятирозрядну адресу. Він завжди може бути інкрементований (збільшений на одиницю), «скинутий» (занулений) пристроєм управління або змінений командою передачі даних. PC – Programm Counter. |
| increment | Приріст |
| Вказівник стека (SP) | Подібний до лічильника команд. В ньому міститься адреса вершини стека, яку він встановлює або забирає. Місткість вказівника стека – 16 біт. |
| Операція, операнд | Команда складається з двох частин: оператора і операнда. Оператор відповідає на питання – що зробити, а операнд – звідки взяти дані і куди передати. |
| MIPS - | Millions Instructions per Second |

| | |
|---|---|
| FLASH ROM | Flash memory – флеш-пам'ять є незалежною електрично перепрограмованою постійною пам'яттю, яка може бути записана і прочитана так само, як і динамічний ОЗП, але зберігає свій вміст без живлення і регенерації, як EPROM. Застосовується в дуже багатьох типах електронних пристроїв. |
| ROM | Енергонезалежна пам'ять призначена для одноразового записування інформації і багаторазового читання (Read only memory). |
| PWM | Широтно–імпульсна модуляція |
| EEPROM (ЕСППЗП) | Electrical Erasable Programmable Read-Only Memory електронностираний запам’ятовувальний пристрій, який може бути перепрограмований більше 1’000’000 раз. Цей пристрій є рідко використовуваним типом напівпровідникового ППЗП, в якому для стирання і перепрограмування використовуються електричні сигнали, що подаються або з комп'ютера, або із зовнішнього пристрою (програматора) на спеціальні виводи мікросхеми. Він допускає від 10 до 100 тис. циклів перезаписування. ЕСППЗП було помітно витіснено флеш-пам'яттю. |
| PROM (ППЗП) | Programmable read only memory, програмований постійний запам’ятовувальний пристрій. |
| МК | Мікроконтролер. |
| NB | Кількість виводів з корпусу мікроконтролера |
| NK | Кількість команд в системі команд. |
| $F_{max}$ | Максимальне значення тактової частоти. |
| I | Струм споживання. |
| GCK | Генератор тактового сигналу. |
| CPU | Central processing unit, процесор. |
| ПЗП | Постійний запам'ятовувальний пристрій для зберігання програми, виконаний за технологією Flash (FlashROM). |
| ОЗП | Оперативний запам'ятовувальний пристрій статичного типу для зберігання даних (SRAM). |

| | |
|---|---|
| SRAM | Static random access memory, статичний. запам'ятовувальний пристрій з довільною вибіркою, статичний ОЗП. |
| VCC | Вивід, призначений для підключення позитивного полюса джерела напруги живлення. |
| GND | Вивід, призначений для підключення негативного полюса джерела напруги живлення. |
| RISC | Restricted (Reduced) Instruction Set Computer. |
| CISC | Complete Instruction Set Computer. |
| AT | Atmel. |
| AVR | сімейство процесорів фірми Atmel (Advanced Virtual RISC). |
| ISP | In System Programming. |
| IAR | IAR Systems − провідний світовий виробник апаратних та програмних засобів розробки вбудованих систем, що дозволяють великим та малим фірмам створювати високоякісні продукти на базі 8-, 16-, 32-бітових мікроконтролерів в галузі промислової автоматики, медичного обладнання, автомобільної електроніки. Фірма має налагоджені партнерські стосунками з найбільшими виробниками напівпровідників. Заснована у 1983 році, фірма IAR Systems входить у групу компаній Inoi. |
| RAM | Random Access Memory – пам’ять з довільною вибіркою. |
| ПЗЕС | Південно-Західна електроенергетична система. |
| АСУ | Автоматизовані системи управління. |

## Додаток Б

```
;***********************************************************
;* A P P L I C A T I O N  N O T E  F O R  T H E  A V R  F A M I L Y
;*
;* Number                :AVR000
;* File Name             :»m8515def.inc»
;* Title                 :Register/Bit Definitions for the ATmega8515
;* Date                  :April 16th, 2002
;* Version               :1.00
;* Support E-mail        :support@atmel.no
;* Target MCU            :ATmega8515
;*
;* DESCRIPTION
;* When including this file in the assembly program file, all I/O register
;* names and I/O register bit names appearing in the data book can be used.
;* In addition, the six registers forming the three data pointers X, Y and
;* Z have been assigned names XL - ZH. Highest RAM address for Internal
;* SRAM is also defined
;*
;* The Register names are represented by their hexadecimal address.
;*
;* The Register Bit names are represented by their bit number (0-7).
;*
;* Please observe the difference in using the bit names with instructions
;* such as «sbr»/»cbr» (set/clear bit in register) and «sbrs»/»sbrc»
;* (skip if bit in register set/cleared). The following example illustrates
;* this:
;*
;* in   r16,PORTB                ;read PORTB latch
;* sbr r16,(1<<PB6)+(1<<PB5)     ;set PB6 and PB5 (use masks, not bit#)
;* out  PORTB,r16                    ;output to PORTB
;*
;* in   r16,TIFR                 ;read the Timer Interrupt Flag Register
;* sbrc      r16,TOV0                ;test the overflow flag (use bit#)
;* rjmp      TOV0_is_set                 ;jump if set
;* ...                           ;otherwise do something else
;***********************************************************

;***** Specify Device
.device ATmega8515

;***** I/O Register Definitions
```

```
.equ   SREG      =$3f
.equ   SPH       =$3e
.equ   SPL       =$3d
.equ   GIMSK     =$3b
.equ   GICR =$3b
.equ   GIFR =$3a
.equ   TIMSK     =$39
.equ   TIFR =$38
.equ   SPMCR     =$37
.equ   EMCUCR    =$36
.equ   MCUCR     =$35
.equ    MCUSR   =$34          ; For compatibility,
.equ    MCUCSR  =$34          ; keep both names until further
.equ   TCCR0     =$33
.equ   TCNT0     =$32
.equ   OCR0      =$31
.equ   SFIOR   =$30
.equ   TCCR1A    =$2f
.equ   TCCR1B    =$2e
.equ   TCNT1H    =$2d
.equ   TCNT1L    =$2c
.equ   OCR1AH    =$2b
.equ   OCR1AL    =$2a
.equ   OCR1BH    =$29
.equ   OCR1BL    =$28
.equ   ICR1H     =$25
.equ   ICR1L     =$24
.equ   WDTCR     =$21
.equ   UCSRC     =$20          ; Note! UCSRC equals UBRRH
.equ   UBRRH     =$20          ; Note! UCSRC equals UBRRH
.equ   EEARH     =$1f
.equ   EEARL     =$1e
.equ   EEDR      =$1d
.equ   EECR      =$1c
.equ   PORTA     =$1b
.equ   DDRA      =$1a
.equ   PINA =$19
.equ   PORTB     =$18
.equ   DDRB      =$17
.equ   PINB =$16
.equ   PORTC     =$15
.equ   DDRC      =$14
.equ   PINC =$13
.equ   PORTD     =$12
```

```
.equ   DDRD       =$11
.equ   PIND =$10
.equ   SPDR=$0f
.equ   SPSR =$0e
.equ   SPCR=$0d
.equ   UDR        =$0c
.equ   UCSRA      =$0b
.equ   UCSRB      =$0a
.equ   UBRR       =$09          ; for AT90S8515
.equ   UBRRL      =$09
.equ   ACSR       =$08
.equ   PORTE      =$07
.equ   DDRE       =$06
.equ   PINE =$05
.equ   OSCCAL     =$04          ; New

;***** Bit Definitions
;GIMSK
.equ   INT1 =7
.equ   INT0 =6
.equ   INT2 =5
.equ   IVSEL      =1            ; interrupt vector select
.equ   IVCE =0              ; interrupt vector change enable


;GIFR
.equ   INTF1      =7
.equ   INTF0      =6
.equ   INTF2      =5

;TIMSK
.equ   TOIE1      =7
.equ   OCIE1A     =6
.equ   OCIE1B     =5
.equ   TICIE1     =3
.equ   TOIE0      =1
.equ   OCIE0      =0

;TIFR
.equ   TOV1       =7
.equ   OCF1A      =6
.equ   OCF1B      =5
.equ   ICF1 =3
.equ   TOV0       =1
```

```
.equ   OCF0=0

;SPMCR
.equ   SPMIE      =7
.equ   RWWSB     =6
.equ   ASB        =6          ; old
.equ   RWWSRE   =4
.equ   ASRE       =4          ; old
.equ   BLBSET    =3
.equ   PGWRT     =2
.equ   PGERS     =1
.equ   SPMEN     =0

;EMCUCR
.equ    SM0     =7
.equ   SRL2 =6
.equ   SRL1 =5
.equ   SRL0 =4
.equ   SRW01      =3
.equ   SRW00      =2
.equ   SRW11      =1
.equ   ISC2  =0

;MCUCR
.equ   SRE   =7
.equ   SRW  =6
.equ   SRW10      =6
.equ   SE     =5
.equ   SM    =4
.equ   SM1   =4
.equ   ISC11      =3
.equ   ISC10      =2
.equ   ISC01      =1
.equ   ISC00      =0

;MCUSR
.equ   SM2     =5
.equ   WDRF       =3
.equ   BORF       =2
.equ   EXTRF      =1
.equ   PORF=0

;TCCR0
.equ    FOC0    =7
```

```
.equ   WGM00  =6
.equ   COM01     =5
.equ   COM00     =4
.equ   WGM01     =3
.equ   CS02 =2
.equ   CS01 =1
.equ   CS00 =0

;TCCR1A
.equ   COM1A1    = 7
.equ   COM1A0    = 6
.equ   COM1B1    = 5
.equ   COM1B0    = 4
.equ   FOC1A     = 3
.equ   FOC1B     = 2
.equ   PWM11     = 1          ; OBSOLETE! Use WGM11
.equ   PWM10     = 0          ; OBSOLETE! Use WGM10
.equ   WGM11     = 1
.equ   WGM10     = 0

;TCCR1B
.equ   ICNC1     = 7
.equ   ICES1     = 6
.equ   CTC11     = 4          ; OBSOLETE! Use WGM13
.equ   CTC10     = 3          ; OBSOLETE! Use WGM12
.equ   WGM13     = 4
.equ   WGM12     = 3
.equ   CS12 = 2
.equ   CS11 = 1
.equ   CS10 = 0


;SFIOR
.equ   TSM  =7
.equ   XMBK      =6
.equ   XMM2      =5
.equ   XMM1      =4
.equ   XMM0      =3
.equ   PUD    =2
.equ   PSR10     =0

;WDTCR
.equ   WDTOE     =4
.equ   WDCE      =4
```

```
.equ   WDE =3
.equ   WDP2        =2
.equ   WDP1        =1
.equ   WDP0        =0

;EECR
.equ    EERIE   =3
.equ   EEWEE      =2
.equ   EEMWE     =2
.equ   EEWE        =1
.equ   EERE=0

;PORTA
.equ   PA7   =7
.equ   PA6   =6
.equ   PA5   =5
.equ   PA4   =4
.equ   PA3   =3
.equ   PA2   =2
.equ   PA1   =1
.equ   PA0   =0

;DDRA
.equ   DDA7        =7
.equ   DDA6        =6
.equ   DDA5        =5
.equ   DDA4        =4
.equ   DDA3        =3
.equ   DDA2        =2
.equ   DDA1        =1
.equ   DDA0        =0

;PINA
.equ   PINA7       =7
.equ   PINA6       =6
.equ   PINA5       =5
.equ   PINA4       =4
.equ   PINA3       =3
.equ   PINA2       =2
.equ   PINA1       =1
.equ   PINA0       =0

;PORTB
.equ   PB7   =7
```

```
.equ   PB6   =6
.equ   PB5   =5
.equ   PB4   =4
.equ   PB3   =3
.equ   PB2   =2
.equ   PB1   =1
.equ   PB0   =0

;DDRB
.equ   DDB7        =7
.equ   DDB6        =6
.equ   DDB5        =5
.equ   DDB4        =4
.equ   DDB3        =3
.equ   DDB2        =2
.equ   DDB1        =1
.equ   DDB0        =0

;PINB
.equ   PINB7       =7
.equ   PINB6       =6
.equ   PINB5       =5
.equ   PINB4       =4
.equ   PINB3       =3
.equ   PINB2       =2
.equ   PINB1       =1
.equ   PINB0       =0

;PORTC
.equ   PC7   =7
.equ   PC6   =6
.equ   PC5   =5
.equ   PC4   =4
.equ   PC3   =3
.equ   PC2   =2
.equ   PC1   =1
.equ   PC0   =0

;DDRC
.equ   DDC7        =7
.equ   DDC6        =6
.equ   DDC5        =5
.equ   DDC4        =4
.equ   DDC3        =3
```

```
.equ   DDC2       =2
.equ   DDC1       =1
.equ   DDC0       =0

;PINC
.equ   PINC7      =7
.equ   PINC6      =6
.equ   PINC5      =5
.equ   PINC4      =4
.equ   PINC3      =3
.equ   PINC2      =2
.equ   PINC1      =1
.equ   PINC0      =0

;PORTD
.equ   PD7   =7
.equ   PD6   =6
.equ   PD5   =5
.equ   PD4   =4
.equ   PD3   =3
.equ   PD2   =2
.equ   PD1   =1
.equ   PD0   =0

;DDRD
.equ   DDD7       =7
.equ   DDD6       =6
.equ   DDD5       =5
.equ   DDD4       =4
.equ   DDD3       =3
.equ   DDD2       =2
.equ   DDD1       =1
.equ   DDD0       =0

;PIND
.equ   PIND7      =7
.equ   PIND6      =6
.equ   PIND5      =5
.equ   PIND4      =4
.equ   PIND3      =3
.equ   PIND2      =2
.equ   PIND1      =1
.equ   PIND0      =0
```

```
;PORTE
.equ   PE2   =2
.equ   PE1   =1
.equ   PE0   =0

;DDRE
.equ   DDE2        =2
.equ   DDE1        =1
.equ   DDE0        =0

;PINE
.equ   PINE2       =2
.equ   PINE1       =1
.equ   PINE0       =0

;UCSRA
.equ   RXC  =7
.equ   TXC  =6
.equ   UDRE        =5
.equ   FE     =4
.equ   OR     =3            ; old name kept for compatibilty
.equ   DOR  =3
.equ   PE     =2
.equ   UPE  =2
.equ   U2X  =1
.equ   MPCM        =0

;UCSRB
.equ   RXCIE       =7
.equ   TXCIE       =6
.equ   UDRIE       =5
.equ   RXEN        =4
.equ   TXEN        =3
.equ   CHR9        =2            ; old name kept for compatibilty
.equ   UCSZ2       =2
.equ   RXB8        =1
.equ   TXB8=0

;UCSRC
.equ   URSEL       =7
.equ   UMSEL       =6
.equ   UPM1        =5
.equ   UPM0        =4
.equ   USBS=3
```

```
.equ   UCSZ1       =2
.equ   UCSZ0       =1
.equ   UCPOL       =0

;SPCR
.equ   SPIE  =7
.equ   SPE   =6
.equ   DORD        =5
.equ   MSTR        =4
.equ   CPOL        =3
.equ   CPHA        =2
.equ   SPR1 =1
.equ   SPR0 =0

;SPSR
.equ   SPIF  =7
.equ   WCOL        =6
.equ   SPI2X       =0

;ACSR
.equ   ACD  =7
.equ   AINBG       =6
.equ   ACBG        =6
.equ   ACO  =5
.equ   ACI   =4
.equ   ACIE =3
.equ   ACIC =2
.equ   ACIS1       =1
.equ   ACIS0       =0

.def   XL    =r26
.def   XH    =r27
.def   YL    =r28
.def   YH    =r29
.def   ZL    =r30
.def   ZH    =r31

.equ   RAMEND =$25F
.equ   EEPROMEND = $1FF
.equ   FLASHEND = $FFF

;  byte groups
;  /\--\/--\/--\
```

```
.equ SMALLBOOTSTART =0b00111110000000 ;($0F80) smallest boot block is 128W
.equ SECONDBOOTSTART =0b00111100000000 ;($0F00) 2'nd boot block size is 256W
.equ THIRDBOOTSTART =0b00111000000000 ;($0E00) third boot block size is 512W
.equ LARGEBOOTSTART =0b00110000000000 ;($0C00) largest boot block is 1KW
.equ BOOTSTART =THIRDBOOTSTART ;OBSOLETE!!! kept for compatibility
.equ PAGESIZE =32 ;number of WORDS in a page


.equ INT0addr=$001 ;External Interrupt0 Vector Address
.equ INT1addr=$002 ;External Interrupt1 Vector Address
.equ ICP1addr=$003 ;Input Capture1 Interrupt Vector Address
.equ OC1Aaddr=$004 ;Output Compare1A Interrupt Vector Address
.equ OC1Baddr=$005 ;Output Compare1B Interrupt Vector Address
.equ OVF1addr=$006 ;Overflow1 Interrupt Vector Address
.equ OVF0addr=$007 ;Overflow0 Interrupt Vector Address
.equ SPIaddr =$008 ;SPI Interrupt Vector Address
.equ URXCaddr=$009 ;UART Receive Complete Interrupt Vector Address
.equ UDREaddr=$00a ;UART Data Register Empty Interrupt Vector Address
.equ UTXCaddr=$00b ;UART Transmit Complete Interrupt Vector Address
.equ ACIaddr =$00c ;Analog Comparator Interrupt Vector Address

.equ INT2addr=$00d ;External Interrupt2 Vector Address
.equ OC0addr= $00e ;Output Compare0 Interrupt Vector Address
.equ ERDYaddr=$00f ; EEPROM Interrupt Vector Address
.equ SPMaddr =$010 ; SPM complete Interrupt Vector Address
.equ SPMRaddr =$010 ; SPM complete Interrupt Vector Address
```

## Додаток В

```
;***** STK500 LEDS and SWITCH demonstration
.include «m8515def.inc»
.def Temp =r16 ; Temporary register
.def Delay =r17 ; Delay variable 1
.def Delay2 =r18 ; Delay variable 2
;***** Initialization
RESET:
ser Temp
out DDRB,Temp ; Set PORTB to output
;**** Test input/output
LOOP:
out PORTB,temp ; Update LEDS
sbis PIND,0x00 ; If (Port D, pin0 == 0)
inc Temp ; then count LEDS one down
sbis PIND,0x01 ; If (Port D, pin1 == 0)
dec Temp ; then count LEDS one up
sbis PIND,0x02 ; If (Port D, pin2 == 0)
ror Temp ; then rotate LEDS one right
sbis PIND,0x03 ; If (Port D, pin3 == 0)
rol Temp ; then rotate LEDS one left
sbis PIND,0x04 ; If (Port D, pin4 == 0)
com Temp ; then invert all LEDS
sbis PIND,0x05 ; If (Port D, pin5 == 0)
neg Temp ; then invert all LEDS and add 1
sbis PIND,0x06 ; If (Port D, pin6 == 0)
swap Temp ; then swap nibbles of LEDS
;**** Now wait a while to make LED changes visible.
DLY:
dec Delay
brne DLY
dec Delay2
brne DLY
rjmp LOOP ; Repeat loop forever
```

## Додаток Г

Для зміни типу контролера після завантаження раніше написаної програми потрібно в головному меню вибрати вкладку Debug, а потім Select Platform and Device. Подальша послідовність дій описана раніше.

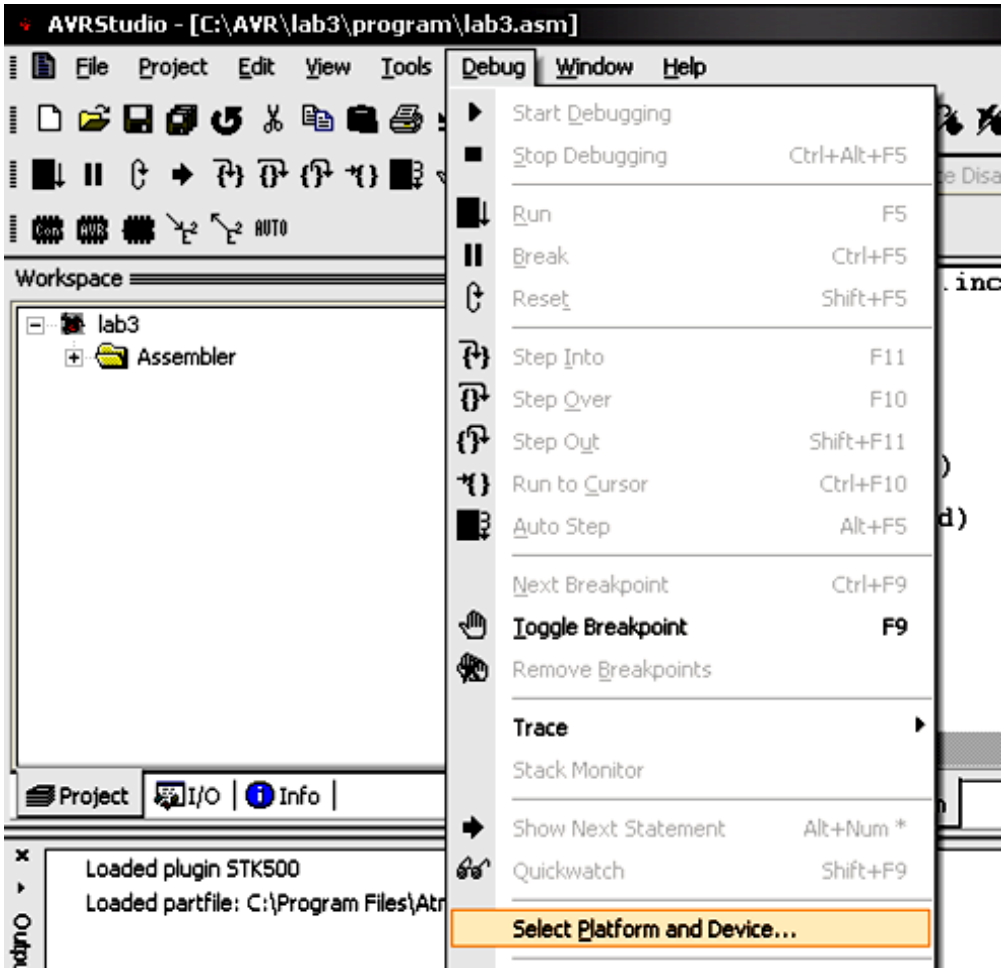



Рисунок Г.1 – Вибір команди «Select Platform and Device»

Програма виводить двійкове число на світлодіоди, що видно по засвічуванню світлодіодів (табл. Г.1).

Таблиця Г.1 – Відповідність засвічування світлодіодів двійкому числу

| Число | Світлодіоди | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $0_{10}$=00000 | | | | | | | | |
| $1_{10}$=00000 | | | | | | | | |
| $2_{10}$=00000 | | | | | | | | |
| $3_{10}$=00000 | | | | | | | | |
| $4_{10}$=00000 | | | | | | | | |
| $5_{10}$=00000 | | | | | | | | |

Продовження таблиці Г.1

| Число | Світлодіоди | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| … | . | . | . | . | . | . | . | . |
| $7_{10}$=00000 $111_2$ | | | | | | | | |
| $8_{10}$=00001 $000_2$ | | | | | | | | |
| $9_{10}$=00001 $001_2$ | | | | | | | | |
| … | . | . | . | . | . | . | . | . |

При натисканні на кнопку 0 і її утриманні відлік здійснюється у зворотному напрямку, а саме, відбувається зменшення вмісту регістра на 1.

## Додаток Д

```
;Програма  додавання з урахуванням перенесення

.include «m8515def.inc»

.def operand_1=r16
.def operand_2=r17
.def rez_l=r18
.def rez_h=r19
.def odynycja=r20

ldi odynycja,1
lds operand_1,$60
lds operand_2,$61

mov rez_l,operand_1

dodavanja:
        adc rez_l,odynycja
        brcc propusk
        inc rez_h
        clc
        propusk:

        dec operand_2
        cpi operand_2,0
brne dodavanja

sts $63,rez_h
sts $64,rez_l

end:
nop
rjmp end
```

```
;Програма віднімання без перенесення
.include «8515def.inc»

.def operand_1=r16
.def operand_2=r17

lds operand_1,$60
lds operand_2,$61
sub operand_1,operand_2
sts $63,operand_1
```

## Додаток Е

```
.include «8515def.inc»

;*********************************************
;       Ділення  восьмирозрядних чисел, записаних
;у пам’яті даних МК.
;       Проводиться відніманням діленого від
;самого себе.
;*********************************************

lds r16,0x60
lds r17,0x61

vidn:

sbc r16,r17
inc r18

brcc vidn

subi r18,1
sts 0x6e,r18
```

## Додаток Ж

```
.include «m8515def.inc»

;Клавіатура - PORTD
;Індикатор - PORTA

.cseg
ldi r16,0xff
out ddra,r16
ldi r16,low(ramend)
out spl,r16
ldi r16,high(ramend)
out sph,r16

main:
rcall keyboard
rcall led
rjmp main

led:
clr r30
ldi zl,low(led_code*2)
ldi zh,high(led_code*2)

add r30,r17
lpm
out PORTA,r0
ret

keyboard:
clr r16
out PORTD,r16
out DDRD,r16
sbic PIND,0
ldi r17,8; 8
sbic PIND,1
ldi r17,9; 9
sbic PIND,2
ldi r17,0x0a  ; ,
sbic PIND,3
ldi r17,0x0b ; enter
sbic PIND,4
```

```
ldi r17,0x0c  ; F1
sbic PIND,5
ldi r17,0x0d  ; F2
sbic PIND,6
ldi r17,0x0e  ; F3

ser r16
out PORTD,r16
OUT DDRD,r16
SBIS PIND,0
LDI r17,0; 0
SBIS PIND,1
LDI r17,1; 1
SBIS PIND,2
LDI r17,2; 2
SBIS PIND,3
LDI r17,3; 3
SBIS PIND,4
LDI r17,4; 4
SBIS PIND,5
LDI r17,5; 5
SBIS PIND,6
LDI r17,6; 6
SBIS PIND,7
LDI r17,7; 7
;in r17 code of pressed key
ret

;  0 1 2 3 4 5 6 7 8 9 a b c d e f
led_code:
 .db
0b00000110,0b111011110,0b100001101,0b000011100,0b111010100,0b1
00110100,0b000100100,0b101011110,0b000000100,0b000010100,0b010
00100,0b10100100,0b00100111,0b10001100,0b00100101,0b01100101
```

*Навчальне видання*

**Олександр Євгенійович Рубаненко**
**Костянтин Іванович Кравцов**
**Олена Олександрівна Рубаненко**

# Мікропроцесорна техніка
# Використання AVR мікроконтролерів ATMEL

Лабораторний практикум

Редактор Т. Старічек
Оригінал-макет підготовлено О. Рубаненком

Підписано до друку 02.01.2018.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 6,61.
Наклад 50 (1-й запуск 1-20) пр. Зам. № 2018-009.

Видавець та виготовлювач
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.