ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра „Спеціалізовані комп'ютерні системи"

В.В. Нарожний

ЦИФРОВІ ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНІ МАШИНИ

Конспект лекцій

Харків – 2010

Нарожний В.В. Цифрові електронно-обчислювальні машини: Конспект лекцій. – Харків: УкрДАЗТ, 2010. – 105 с.

Конспект лекцій містить теоретичні відомості з архітектури, устрою та функціонування цифрових мікроконтролерів фірми Atmel, методику виконання лабораторних робіт мовою програмування Assembler, що дозволить подивитись на устрій та функціонування ЦЕОМ на базі мікроконтролерів з максимально глибоким дослідженням.

Призначений для студентів технічних спеціальностей при вивченні сучасних цифрових обчислювальних машин, архітектури ЦЕОМ та периферійних пристроїв ПЕОМ як матеріал лекційних і лабораторних занять, а також при курсовому і дипломному проектуванні.

Іл. 18, табл. 31, бібліогр.: 5 назв.

Конспект лекцій розглянуто та рекомендовано до друку на засіданні кафедри «Спеціалізовані комп'ютерні системи» 18 січня 2010 р., протокол №5/10.

Рецензент

доц. В.А. Книш

В.В. Нарожний

ЦИФРОВІ ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНІ МАШИНИ

Конспект лекцій

Відповідальний за випуск Нарожний В.В.

Редактор Ібрагімова Н.В.

Підписано до друку 14.04.10 р.
Формат паперу 60х84 1/16 . Папір писальний.
Умовн.-друк.арк. 5,5. Обл.-вид.арк. 5,75.
Замовлення № Тираж 50 Ціна

Видавництво УкрДАЗТу, свідоцтво ДК № 2874 від. 12.06.2007 р.
Друкарня УкрДАЗТу,
61050, Харків - 50, майд. Фейєрбаха, 7

ЗМІСТ

	ВСТУП ……..	4
1	Місце мікроконтролера в сучасних цифрових електронно-обчислювальних машинах ……	5
2	Мікроконтролери сімейства AVR ……...	9
3	Описання мікроконтролера ATtiny15L ……..	14
3.1	Цоколівка й описання виводів мікроконтролера ATtiny15L ……..	14
3.2	Організація пам'яті мікроконтролера ATtiny15L .	15
4	Регістри керування мікроконтролера ATtiny15L	17
4.1	SREG – регістр стану мікроконтролера ……...	17
4.2	MCUCR – регістр керування мікроконтролера ...	18
5	Програма-емулятор роботи мікроконтролерів фірми ATMEL – AVR-studio …….	19
5.1	Загальний опис вікон AVR-studio ……...	19
5.2	Програмування в середовищі AVR-studio ……...	20
6	Енергонезалежна пам'ять даних (EEPROM) …..	26
7	Режим зниженого енергоспоживання ……...	29
8	Скидання мікроконтролера …….	31
9	Порти введення-виведення ……	32
10	Переривання ……...	34
11	Таймери ……...	37
11.1	Переривання від таймера ……	38
11.2	Таймер-лічильник T0 ……	39
11.3	Таймер-лічильник T1 ……	41
11.4	Режим ШІМ таймера T1 ……...	45
11.5	Сторожовий таймер ……..	48
12	Аналоговий компаратор та аналого-цифровий перетворювач ……..	52
12.1	Аналоговий компаратор ……..	52
12.2	Аналого-цифровий перетворювач (АЦП) ……	54
	Список літератури ……..	62
	Додаток А РВВ мікроконтролера ATtiny15L ……	63
	Додаток Б Система команд ……..	64
	Додаток В Розширений опис команд ……	68
	Додаток Г Директиви Assembler ……...	105

ВСТУП

Центральним елементом кожної цифрової електронно-обчислювальної машини є центральний процесор. Вже багато років термін «центральний процесор» є близьким терміну «мікропроцесор». Це сталося, коли майже всі електронні схеми центрального процесора було вшито на кристалі однієї мікросхеми, яку назвали мікропроцесором. З часом у мікропроцесор почали додавати нові пристрої, що не входили до складу центрального процесора. Серед них постійний запам'ятовуючий пристрій, оперативний запам'ятовуючий пристрій, таймер, генератор послідовних імпульсів та ін. Переважна більшість процесорів, що випускаються у світі, — мікроконтролери (МК).

Мікроконтролер (**англ.** *microcontroller*), або цифрова однокришталева **мікро-ЕОМ**, виконана у вигляді **мікросхеми**, що включає **мікропроцесор**, блоки **пам'яті** для збереження **коду програм** і даних, **порти** введення-виведення і блоки зі спеціальними функціями (**лічильники**, **компаратори**, **АЦП** тощо). Використовується для керування **електронними** пристроями. По суті, це **комп'ютер**, здатний виконувати прості завдання. Використання однієї мікросхеми значно знижує розміри, **енергоспоживання** і вартість пристроїв, побудованих на базі мікроконтролерів.

Мікроконтролери можна зустріти в багатьох сучасних приладах, таких як **телефони**, **пральні машини**; вони відповідають за працю **двигунів** і систем **гальмування** сучасних **автомобілів**, за їх допомогою створюються **системи контролю** і **системи збору інформації**. Одним з найпопулярніших сімейств мікроконтролерів у світі є AVR — сімейство восьмибітових **мікроконтролерів** фірми **Atmel**, що розглядаються у конспекті лекцій. Мікроконтролери AVR мають **гарвардську архітектуру** (програма і дані знаходяться в різних адресних просторах) і систему команд, близьку до ідеології **RISC**. Архітектура AVR дозволяє застосовувати **операційні системи** при розробленні приладів, основними з яких є **FREERTOS** і **uOS**.

У конспекті лекцій розглядається найпростіший з мікроконтролерів сімейства ATtiny15L (хоча все сімейство є дуже схожим, проте деякі мікроконтролери відрізняються об’ємом пам'яті та присутністю більшої кількості периферійних пристроїв). Кожний розділ містить опис фрагментів лабораторних робіт, де застосовано мову програмування Assembler за допомогою емулятора AVR-studio. Під час циклу лабораторних робіт студент вивчить основні елементи і пристрої мікроконтролерів, різні периферійні пристрої, що присутні у мікроконтролерах.

Автор вісловлює подяку за допомогу Павлову В.А.

1 МІСЦЕ МІКРОКОНТРОЛЕРА В СУЧАСНИХ ЦИФРОВИХ ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНИХ МАШИНАХ

Електронно-обчислювальна машина (скорочено — ЕОМ) — загальна назва для обчислювальних машин, що є електронними (починаючи з перших лампових машин, включаючи напівпровідникові тощо), на відміну від електромеханічних (на електричних реле тощо) та механічних обчислювальних машин.

За часів широкого розповсюдження аналогових обчислювальних машин, що теж були у своїй переважній більшості електронними, для уникнення непорозумінь використовувалася назва «*цифрова* електронна обчислювальна машина» (ЦЕОМ) або «лічильна» (рос. *счётная*) машина (задля підкреслення того, що цифрова електронна машина саме реалізує безпосередньо обчислення результату, у той час, як аналогова машина фактично реалізує процес фізичного моделювання з отриманням результату вимірювання).

Унікальний винахід XX ст. — цифрову електронну обчислювальну машину — найчастіше називають англійським словом — комп'ютер (computer), що в перекладі дає те саме значення: обчислювач.

Гордон Мур (засновник корпорації INTEL, що першою у світі створила мікропроцесор) зазначив, що є два шляхи розвитку цифрової електронно-обчислювальної техніки:

1 Створення комп'ютерів все більшої потужності при постійній ціні.

2 Випускати одну модель при постійному зниженні ціни.

Сучасна комп'ютерна промисловість йде двома цими шляхами.

Але за останні 20-30 років з’явилась тенденція щодо розмежування різних за завданнями цифрових електронно-обчислювальних машин.

Одноразові комп'ютери

Маленькі, малофункціональні, спеціалізовані пристрої. Функції, які вони можуть виконувати: вбудовані у вітальні листівки, програвачі мелодій, радіопередавачі, які планується імплантувати тваринам, ідентифікатори товарів у супермаркетах, порогові пристрої в детекторній техніці, маркування багажу в аеропортах. Європейський центральний банк планує в найближчі роки налагодити випуск банкнот, які будуть запам'ятовувати всі інстанції проходження, та ін.

Такі комп'ютери можуть бути як пасивними (немає джерела живлення), так і активними (знаходяться в постійній дії).

Найбільш відома мікросхема товщиною близько півміліметра – RFID (Radio Frequency Identification – радіочастотна ідентифікація).

Мікроконтролери

Основне завдання таких комп'ютерів – виконання функцій управління пристроями і організації на їх базі інтерфейсів для користувача:

1) побутові прилади (будильники, пральні машини, мікрохвильові печі, охоронні сигналізації);

2) комунікатори (телефони, апарати факсиміле, маршрутизатори);

3) периферійні пристрої (принтери, сканери, модеми, приводи та ін.);

4) розважальні пристрої (відеомагнітофони, ДВД-програвачі, музичні центри, MP3-плеєри та ін.);

5) формувачі зображень (телевізори, цифрові фото- і відеокамери, копіювальні пристрої);

6) медичне обладнання;

7) військові комплекси;

8) торгове обладнання;

9) грашки;

10) автомобілі тощо.

На відміну від одноразових комп'ютерів, мікроконтролери є повноцінними комп'ютерами. Всі мікроконтролери поділяють на універсальні і спеціальні. Універсальні – комп'ютери, зменшені у розмірі. Спеціальні – обмежені архітектурою та системою команд і пристосовані для вирішення певного кола завдань.

Мікроконтролери бувають 4-, 8-, 16-, 32-розрядними.

Три основні напрями розвитку мікроконтролерів:

1) ціна;

2) робота в реальному масштабі часу;

3) розмір і енергоспоживання.

Ігрові комп'ютери

По суті це звичайні ПК, в яких розширені можливості графічних і звукових підсистем. При цьому використовуються не останні моделі процесорів: PlayStation-2 (процесор 296 МГц, але 128 розрядів даних, пам'ять 32 Мбайт, графічна мікросхема 160 МГц, 48-канальна звукова плата), XBOX (P3 733 МГц, пам'ять 64 Мбайт, графічна мікросхема 300 МГц, 256-канальна

звукова мікросхема).

Персональні комп'ютери

Два варіанти – настільні і портативні (ноутбуки).

Особливістю є наявність складної операційної системи (ОС).

Сервери

Основна відмінність від ПК – величезні об'єми пам'яті (ОЗУ, вінчестери). Архітектурно вони мало відрізняються від ПК.

Комплекси робочих станцій

На сучасному етапі розвитку – звичайні ПК, зібрані в загальну систему з розпаралелюванням обчислювальних можливостей.

Мейнфрейми

Справді величезні комп'ютери. Працюють не швидше за звичайні, але в них вища швидкість процесів введення-виведення і великі простори на диску. Деякий час тому з'явилася тенденція до витіснення таких машин **Серверами**, оскільки такі машини вимагають особливого підходу, спеціальних програмних засобів і дуже дорогі при купівлі й обслуговуванні. Але, на щастя, дуже скоро стало зрозумілим, що такі машини все ж таки необхідні для обробки величезних масивів даних і в популярних Інтернет-вузлах, де необхідно здійснювати величезну кількість трансакцій за секунду.

Суперкомп'ютер – це загальний термін, який використовується для позначення класу існуючих найпотужніших комп'ютерних систем. Суперкомп'ютери, зазвичай, використовуються при розв’язанні складних наукових та інженерних задач, які вимагають виконання великої кількості математичних операцій та(чи) працюють з великими об'ємами даних.

Усі поняття є відносними в часі. Потужність комп'ютерної системи оцінюється в порівнянні з існуючими

на певний момент комп'ютерними системами широкого використання та рівнем розвитку технологій.

2 МІКРОКОНТРОЛЕРИ СІМЕЙСТВА AVR

Популярність 8-розрядних RISC мікроконтролерів фірми Atmel викликана добрим співвідношенням «Ціна-швидкодія-енергозбереження», хоча експерти зв’язують популярність цього сімейства з наданням можливості прямого перепрограмування мікроконтролерів за допомогою звичайного ПК (LPT-Port), бо раніше мікроконтролери потребували ще одного пристрою – програмувача, який був недешевий. З появою мікроконтролерів фірми Atmel кожний розробник зміг навіть працювати вдома.

Існують три основні види мікроконтролерів фірми Atmel:

1) Classic AVR;
2) Tiny AVR;
3) Mega AVR.

Мікроконтролери Classic AVR не дарма мають таку назву, оскільки фірма на початковому етапі свого розвитку зайнялася виробництвом мікроконтролерів на базі класичного 8051 фірми INTEL. Істотною відмінністю таких МК була спрощена система прошиття, що не вимагає купівлі програматора, і використання вбудованої в мікроконтролері флеш-пам’яті, яка саме в цей час отримала помітне розповсюдження. І хоча було випущено величезну кількість мікроконтролерів даного виду, але надалі фірма перейшла на два інших сімейства, що було потребою часу.

Мікроконтролери Tiny AVR випущені в так званому малобюджетному виконанні. Можна навіть передбачити, що фірма планувала розвинути на їхній базі одноразові комп'ютери. Всі мікроконтролери цієї серії мають невелику пам'ять і призначені для невеликих і вузькоспеціалізованих пристроїв.

Мікроконтролери Mega AVR – пряма протилежність Tiny AVR, що позначається на їхній ціні, але і можливості цих мікроконтролерів помітно вище. Вони призначені для

використання в мобільних телефонах, контролерах периферійних пристроях (принтери, сканери, DVD та ін.), складній офісній техніці.

Мікроконтролери сімейства Tiny

8-розрядні мікроконтролери побудовані за КМОП-технологією, яка має вдосконалену RISC-архітектуру, що дозволяє досягти якнайкращого співвідношення швидкодія/ енергоспоживання. МК призначені для низьковартісних проектів і відповідно є найдешевшими. Не дивлячись на це МК даного сімейства мають дуже добрі характеристики:

- можливість обчислень до 1 МГц;
- FLASH-пам'ять програм 1-2 кбайт;
- ОЗУ до 2 кбайт;
- пам'ять даних (ЕСППЗУ-EEPROM) до 64 байтів;
- можливість захисту від зовнішнього читання і модифікації пам'яті програм і даних;
- можливість програмування безпосередньо в системі через послідовний інтерфейс;
- вбудований генератор;
- наявність двох або трьох режимів енергоспоживання;
- деякі моделі мікроконтролерів можуть працювати при напрузі живлення 1,8 В.

Характеристики процесора:

- повністю статична архітектура (мінімальна тактова частота – нуль);
- арифметико-логічний пристрій підключено безпосередньо до регістрів загального призначення;
- більшість команд виконується за один машинний цикл;
- багаторівнева система переривань і підтримка черги переривань;
- 5-8 джерел переривань;
- апаратний трирівневий стек.

Характеристики підсистем введення-виведення:

– програмне конфігурування і вибір портів введення-виведення;
– виводи можуть бути запрограмовані як вхідні або як вихідні;
– вхідні буфери з тригером Шмідта на всіх виводах;
– можливість програмного підключення внутрішніх підтягуючих резисторів.

Периферійні пристрої:

– 8-розрядний таймер-лічильник;
– сторожовий таймер WDT;
– одноканальний генератор сигналів ШІМ розрядністю 8 бітів;
– аналоговий компаратор;
– 10-розрядний АЦП (декілька каналів);
– апаратний модулятор.

Архітектура ядра

Ядро виконано за вдосконаленою RISC архітектурою, в якій використовується ряд розв'язань, направлених на підвищення швидкодії МК. Структурна схема мікроконтролера наведена на рисунку 2.1.

Арифметико-логічний пристрій (АЛП), підключений безпосередньо до 32 робочих регістрів загального призначення, об'єднаних у регістровий файл, виконує всі обчислення. Завдяки цьому арифметико-логічний пристрій виконує одну операцію за один машинний цикл. Крім того, кожна з команд займає тільки один елемент пам'яті програм.

У мікроконтролерах реалізована Гарвардська архітектура (роздільна пам'ять програм і даних). Розділення шин доступу до пам'яті програм і пам'яті даних дозволяє використовувати різну розрядність для кожного типу пам'яті, а також застосовувати конвеєризацію (при виконанні поточної команди проводиться вибірка з пам'яті і дешифрування коду наступної

команди). Завдяки цьому тривалість машинного циклу дорівнює лише одному періоду кварцового резонатора.

RISC – термін, що означає обчислення за скороченим набором команд.

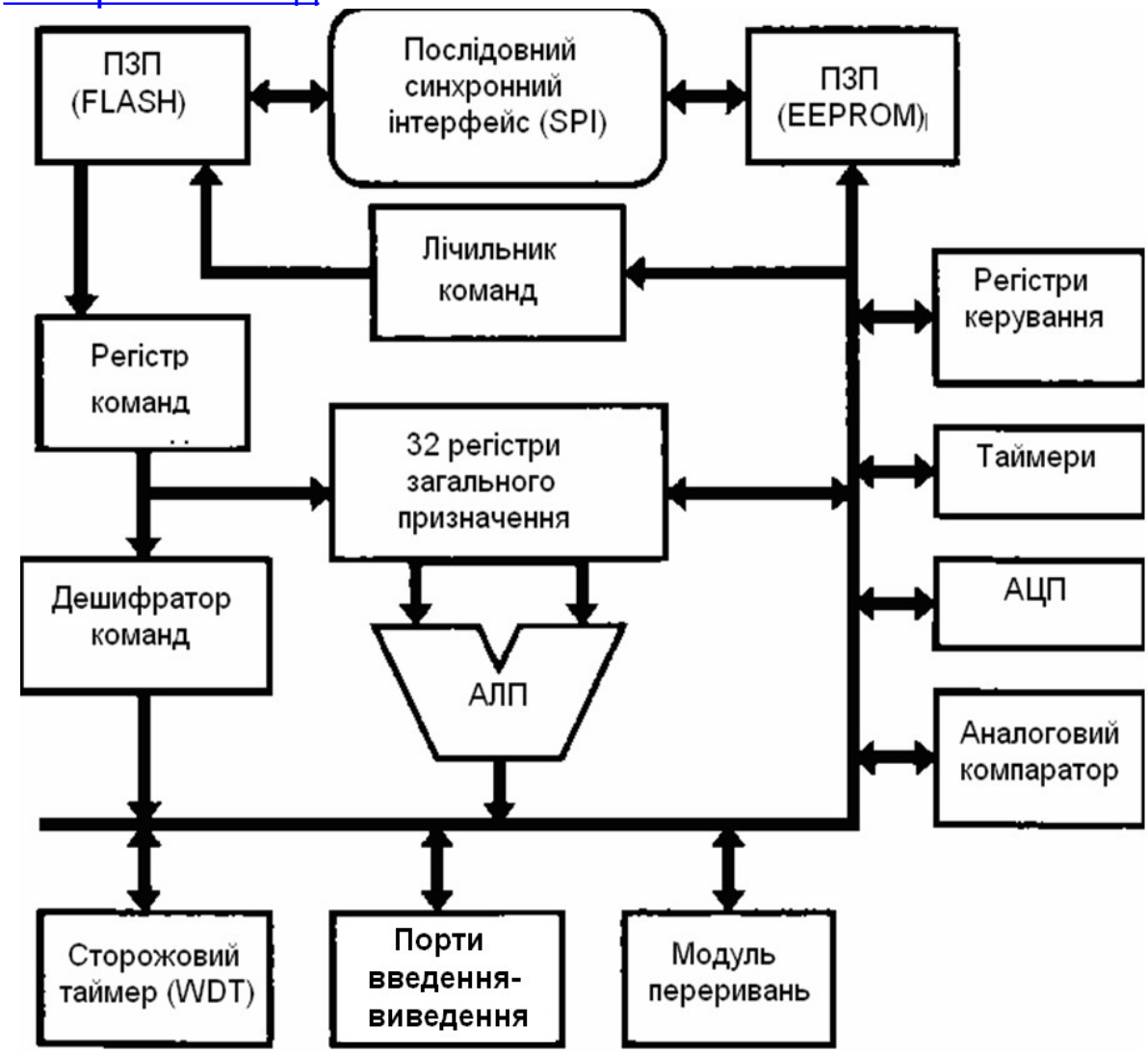

Рисунок 2.1 – Структурна схема мікроконтролера ATtiny15L

Це концепція проектування процесорів, яка має такий принцип: більш компактні і прості інструкції виконуються швидше. Проста архітектура дозволяє як здешевити процесор, так і підняти тактову частоту. Багато ранніх RISC-процесорів навіть не мали команд множення і ділення. Ідея створення RISC процесорів прийшла після того, як в 1970-х

роках вчені з компанії IBM виявили, що багато команд з великою функціональністю ігнорувались програмістами. Навпаки, програмісти навіть не знали про наявність таких команд. Частіше вони дублювали функції цих команд, використовуючи декілька простих команд.

Перші RISC-процесори були розроблені на початку 1980-х років у Стенфордському і Каліфорнійському університетах США.

Характерні особливості RISC-процесорів: фіксована довжина машинних інструкцій (наприклад, 32 біти) і простий формат команди; одна інструкція виконує тільки одну операцію з пам'яттю — читання або запис; операції вигляду «прочитати –змінити — записати» відсутні; велика кількість регістрів загального призначення (32 і більше).

3 ОПИСАННЯ МІКРОКОНТРОЛЕРА ATTINY15L

3.1 Цоколівка й описання виводів мікроконтролера ATtiny15L

Мікроконтролер має FLASH-пам'ять програм об'ємом 1 Кбайт і EEPROM-пам'ять даних об'ємом 64 байти. Максимальна кількість контактів вводу-виводу – 6. Два виводи живлення (рисунок 3.1, таблиця 3.1). Структурна схема мікроконтролера наведена на рисунку 3.2.

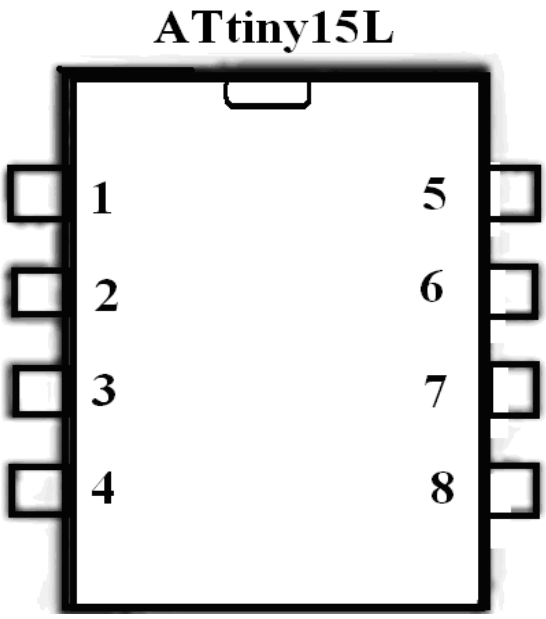

Рисунок 3.1 – Вигляд мікросхеми мікроконтролера ATtiny15L

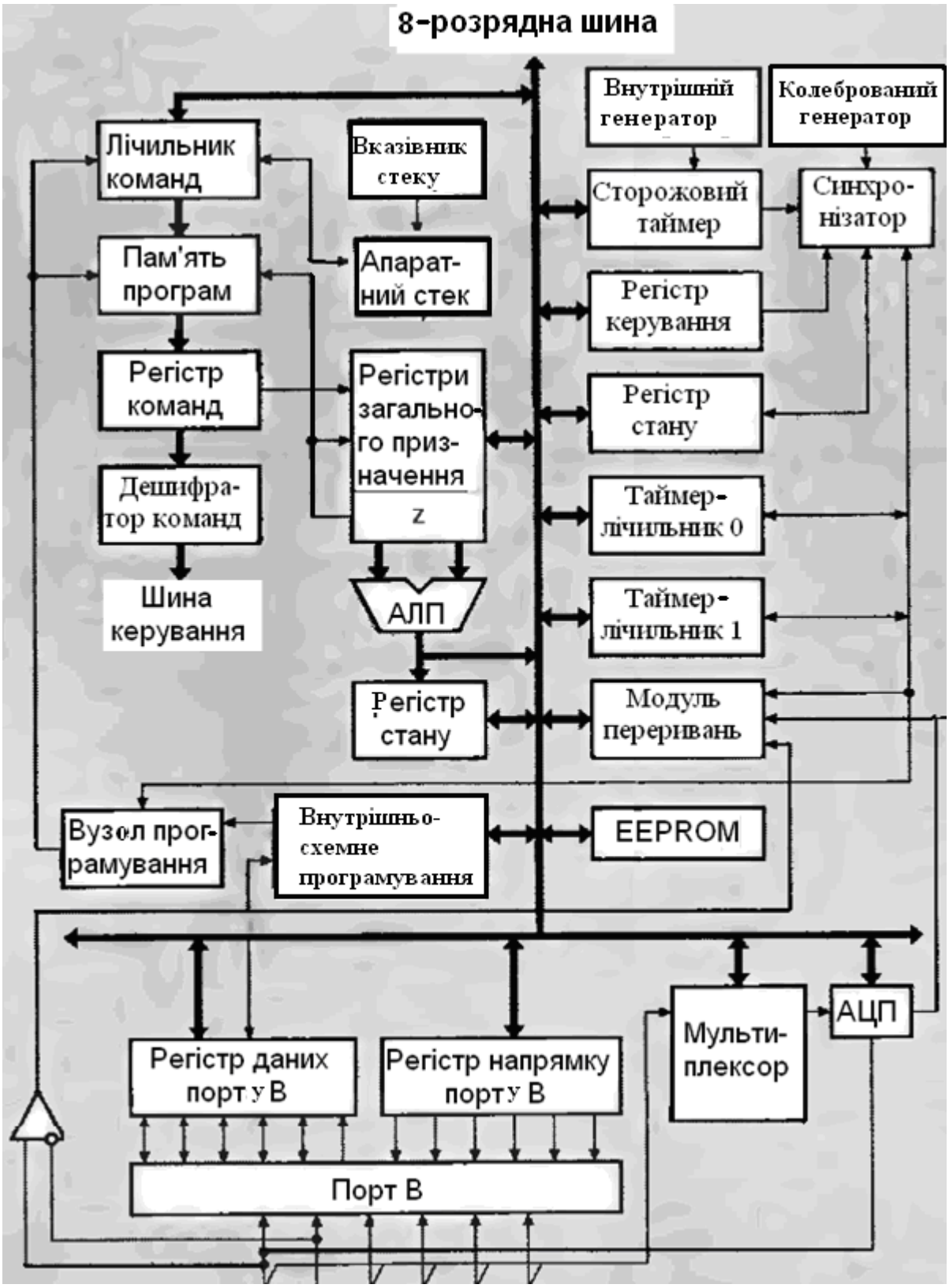

Рисунок 3.2 – Розширена структурна схема мікроконтролера ATtiny15L

Перелік регістрів мікроконтролера ATtiny15L наведено у додатку А.

Таблиця 3.1 – Опис функцій виводів мікросхеми мікроконтролера ATtiny15L

Ознака	Номер виводу	Тип виводу	Опис
PB0(AIN0\ AREF\ MOSI)	5	I\O	0-ий розряд порту B («+» вхід компаратора \Вхід опорної напруги АЦП \Вхід даних при програмуванні)
PB1(AIN1\ OC1A\ MISO)	6	I\O	1-ий розряд порту B («-» вхід компаратора \Вихід таймера-лічильника T1 \Вихід даних при програмуванні)
PB2(ADC1/ T0/ INT0/ SCK)	7	I\O	2-ий розряд порту B (Вхід АЦП1\ Вхід зовнішнього тактового сигналу \Лічильник T0 \Вхід зовнішнього переривання \Вхід тактового сигналу при програмуванні)
PB3(ADC2)	3	I\O	3-ий розряд порту B (Вхід АЦП2)
PB4(ADC3)	2	I\O	4-ий розряд порту B (Вхід АЦП3)
PB5(ADC0/ RESET)	1	I\O	5-ий розряд порту B (Вхід АЦП0\ Вхід скидання мікроконтролера)
GND	4	P	Загальний вивід живлення
Vcc	8	P	Вивід живлення

3.2 Організація пам'яті мікроконтролера ATtiny15L

Пам'ять організована за Гарвардською архітектурою (розділено адресний простір пам'яті даних і програм, а також шини доступу):

1) пам'ять програм призначена для зберігання команд. Також зберігаються константи, які не змінюються в процесі виконання програми. Кожна команда займає 16 бітів. Тобто якщо об'єм пам'яті – 1 кбайт, то це означає, що можна

записати не більше 512 команд. За адресою $000 знаходиться вектор скидання, де потрібно розміщувати команду переходу до основної програми. Далі йде таблиця векторів переривань максимум до адреси $008 (це залежить від марки мікроконтролера);

2) пам'ять даних у мікроконтролері ATtiny15L розділена на дві частини (регістрова – оперативний запам’ятовуючий пристрій (ОЗП) і енергонезалежна – ЕСППЗУ (EEPROM)).

Регістрова пам'ять включає 32 регістри загального призначення (РЗП) і регістри введення-виведення (РВВ) (кількість залежить від моделі). Всі регістри загального призначення можуть використовуватися для зберігання і пересилання даних у процесі виконання програми. Проте потрібно пам'ятати, що ряд команд (SBCI, SUBI, CPI, ANDI, ORI, LDI) не можуть працювати з РЗП R0-R15. РВВ поділяють на службові і такі, що належать до периферійних пристроїв. До будь-якого РВВ можна звернутися за допомогою команд IN, OUT, SBI, CBI, SBIS, SBIC.

Енергонезалежна пам'ять даних (EEPROM) розташована у власному адресному просторі, а її об'єм складає 64 байти.

Способи адресації:

1) пряма адресація. Адреси операндів містяться безпосередньо в команді: пряма адресація одного РЗП, пряма адресація двох РЗП, пряма адресація одного РВВ;

2) непряма адресація. Тільки проста непряма адресація. Команди непрямої адресації виконують звернення до регістра, адреса якого міститься в індексному регістрі Z (цей регістр знаходиться в регістровій парі R30, R31). При цьому вміст Z не змінюється. Використовуються дві команди непрямої адресації: LD Rd,Z (пересилання байта в РЗП) і ST Z,RD (пересилання байта з РЗП).

4 РЕГІСТРИ КЕРУВАННЯ МІКРОКОНТРОЛЕРА ATTINY15L

Як вже було зазначено, всі РВВ потрібні для керування або мікроконтролера в цілому, або кожного з його периферійних пристроїв. Такий підхід дозволяє мати велику кількість мікроконтролерів майже з однією архітектурою, але з різним набором функцій за рахунок присутності тих чи інших периферійних пристроїв.

4.1 SREG – регістр стану мікроконтролера

Цей регістр розташовується за адресою $3F. Містить набір прапорців, що відображують поточний стан мікроконтролера (таблиця 4.1). Ці прапорці автоматично встановлюються в 0 або 1 при настанні певної події.

Таблиця 4.1 – Опис регістра прапорців SREG

Розряд	Назва	Опис
1	2	3
7	I	Загальний дозвіл на переривання. Для дозволу на переривання цей прапорець повинен бути встановлений в «1». Також потрібно встановити прапорці переривань, які необхідно використовувати у програмі. Прапорець скидається автоматично при перериванні та встановлюється по його закінченні командою RETI
6	T	Зберігання скопійованого біта. Цей прапорець використовують команди копіювання бітів BLT та BST, які зберігають або виймають біти РЗП
5	H	Прапорець половинного перенесення. Цей прапорець стає в «1», якщо було переповнення молодших розрядів (0000 1111 – 0001 0000)
4	S	Прапорець знаку, що є результатом операції команди XOR. Прапорець встановлюється, якщо результат менше нуля
3	V	Прапорець переповнення доповнюючого коду. Цей прапорець стає в «1», якщо було переповнення розрядної сітки при роботі зі знаковими числами

Продовження таблиці 4.1

1	2	3
2	N	Прапорець 7-го розряду. Використовується при роботі з від'ємними числами. Цей прапорець встановлюється, якщо результат операції менше нуля
1	Z	Прапорець нуля. Прапорець встановлюється, якщо результат – нуль
0	C	Прапорець перенесення. Цей прапорець стає в «1», якщо було переповнення байта розрядів (з 11111111 у 00000000)

4.2 MCUCR – регістр керування мікроконтролера

Для керування загальними можливостями мікроконтролера використовується регістр MCUCR (таблиця 4.2), розташований за адресою $35.

Таблиця 4.2 – Опис регістра керування MCUCR

Розряд	Назва	Опис
1	2	3
7	-	Не використовується. Читається як «0»
6	PUD	Дозволити використання внутрішніх підтягуючих резисторів порту B, якщо встановлено «1», заборонити – якщо «0»
5	SE	Дозволити перехід у режим зниженого енергоспожи-вання, якщо встановлено «1», заборонити – якщо «0»
4,3	SM1, SM0	Вибрати режим зниженого енергоспоживання. «0» , «0» - IDLE, «0» , «1» - ADC-режим, «1» , «0» - Power Down, «1» , «1» - Резервоване
2	-	Не використовується. Читається як «0»
1,0	ISCO0, ISCO1	Вибрати умову зовнішнього переривання на виводі INT0 «0» , «0» - за низьким рівнем, «0» , «1» - у випадку будь-якої зміни, «1» , «0» - за спадним фронтом, «1» , «1» - за наростаючим фронтом

5 ПРОГРАМА-ЕМУЛЯТОР РОБОТИ МІКРОКОНТРОЛЕРІВ ФІРМИ ATMEL – AVR-STUDIO

Для подальшого вивчення роботи мікроконтролерів фірми ATMEL потрібно скористатися спеціалізованою програмою емуляції роботи мікроконтролерів фірми ATMEL – AVR-studio.

AVR-studio 4 – нове професійне інтегроване середовище (Integrated Development Environment - IDE), призначене для написання і налагоджування прикладних програм для AVR мікропроцесорів у середовищі Windows 9x/NT/2000/XP. AVR-studio 4 містить асемблер і емулятор. AVR-studio підтримує COFF як формат вихідних даних для символьного налагоджування. Інші програмні засоби третіх фірм також можуть бути конфігуровані для роботи з AVR-studio.

5.1 Загальний опис вікон AVR-studio

Ключове вікно в AVR-studio – це вікно початкового тексту програми. Коли об'єктний файл відкритий, автоматично створюється вікно початкового тексту програм. У вікні відображується код, який виконується в налагоджувальному оточенні (емуляторі або програмному симуляторі), а текстовий маркер завжди знаходиться на рядку, який буде виконаний у наступному циклі.

Користувач може виконувати програму повністю в покроковому режимі, трасуючи блоки функцій або виконуючи програму до місця, де є курсор. На додаток можна визначати необмежене число точок зупинок, кожна з яких може бути ввімкнена або вимкнена. Точки зупинок зберігаються між сесіями роботи.

У вікні початкового тексту програми виводиться інформація про процес виконання програми. На додаток, AVR-studio має багато інших вікон, які дозволяють управляти і відображувати інформацію про будь-який елемент мікроконтролера.

Список доступних вікон:

1) Watch window: вікно показує значення певних символів. У цьому вікні користувач може переглядати значення й адреси змінних;
2) Trace window: вікно показує хронологію програми, що виконується в даний час;
3) Register window: вікно показує вміст регістрів. Регістри можна змінювати під час зупинки програми;
4) Memory windows: вікна показують вміст пам'яті програм, даних, портів введення-виведення та енергонезалежного постійного запам’ятовуючого пристрою (ПЗП). Пам'ять можна переглядати в HEX, двійковому або десятковому форматах. Вміст пам'яті можна змінювати під час зупинки програми;
5) I/O window: показує вміст різних регістрів введення/виведення, EEPROM, I/O порти, таймери та ін.;
6) Message window: вікно показує повідомлення від AVR-studio;
7) Processor window: у вікні відображувалася важлива інформація про ресурси мікроконтролера, включаючи програмний лічильник, покажчик стеку, регістр статусу і лічильник циклу. Ці параметри можуть модифікуватися під час зупинки програми.

Настройки робочого оточення зберігаються при виході. При першому запуску вимагається настроювати вікна для управління і виведення необхідної інформації. Під час наступного завантаження настройки автоматично відновлюються.

5.2 Програмування в середовищі AVR-studio

Увага! Потрібно все робити латинськими буквами.

При програмуванні в середовищі AVR-studio треба виконати стандартну послідовність дій: створення проекту; завантаження файла; компіляція; симуляція; завантаження hex-коду в мікроконтролер. Створення проекту починається

з вибору рядка меню Project\New Project (рисунок 5.1).

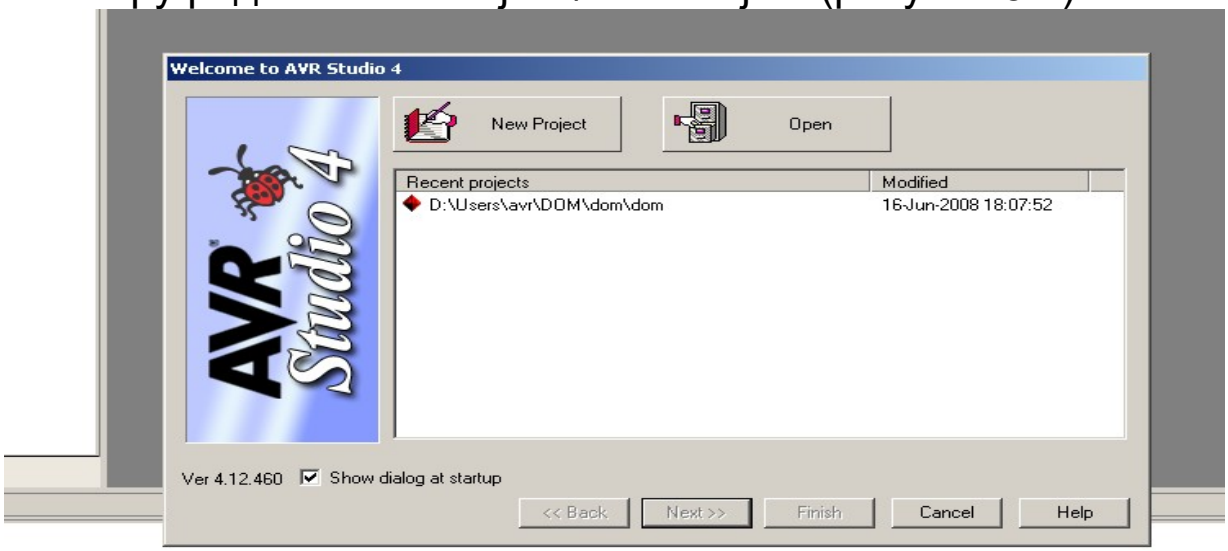

Рисунок 5.1 – Початкове вікно створення проекту

У вікні “Create new Project”, що відкрилося, треба вказати ім'я проекту (у нашому випадку – скс1) і ім'я файла ініціалізації. Вибрати папку для проекту (у нашому випадку 8_IV_скс). Натиснути кнопки “Next” (рисунок 5.2).

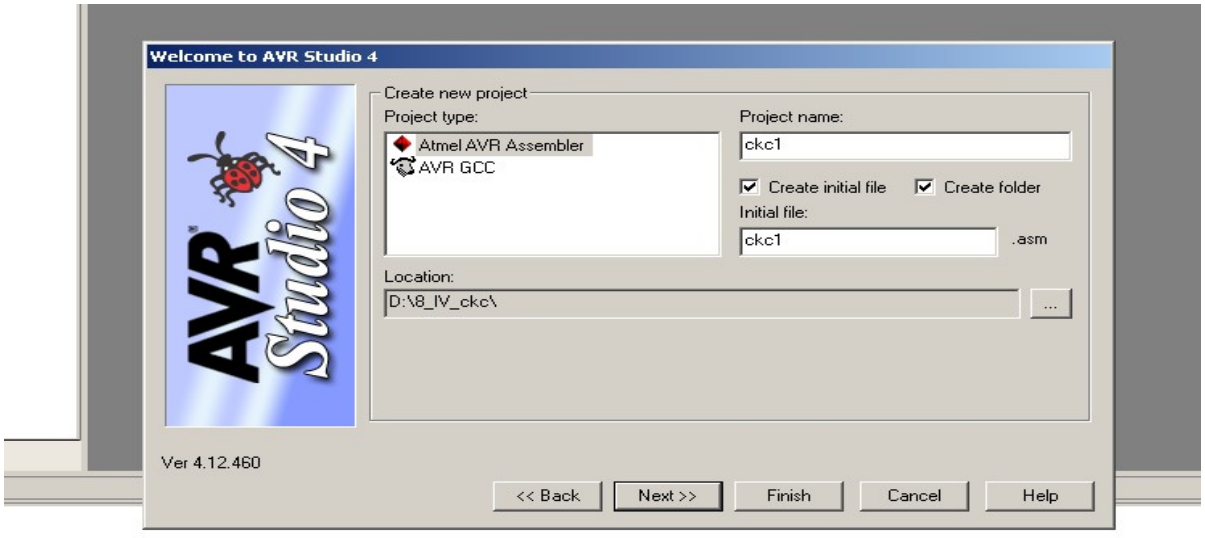

Рисунок 5.2 – Початкове вікно створення назви проекту

Після натиснення кнопки “Next” відкривається вікно “Select debug platform and device” (рисунок 5.3), де вибирається налагоджувальна платформа (симулятор або емулятор) і тип мікроконтролера. Можна вибрати один з пропонованих внутрішньосхемних емуляторів. Зазначимо, що у кожного емулятора свій список підтримуваних мікросхем. Для даного прикладу ми вибираємо платформу AVR Simulator і мікросхему ATiny15. Натиснути кнопку «Finish».

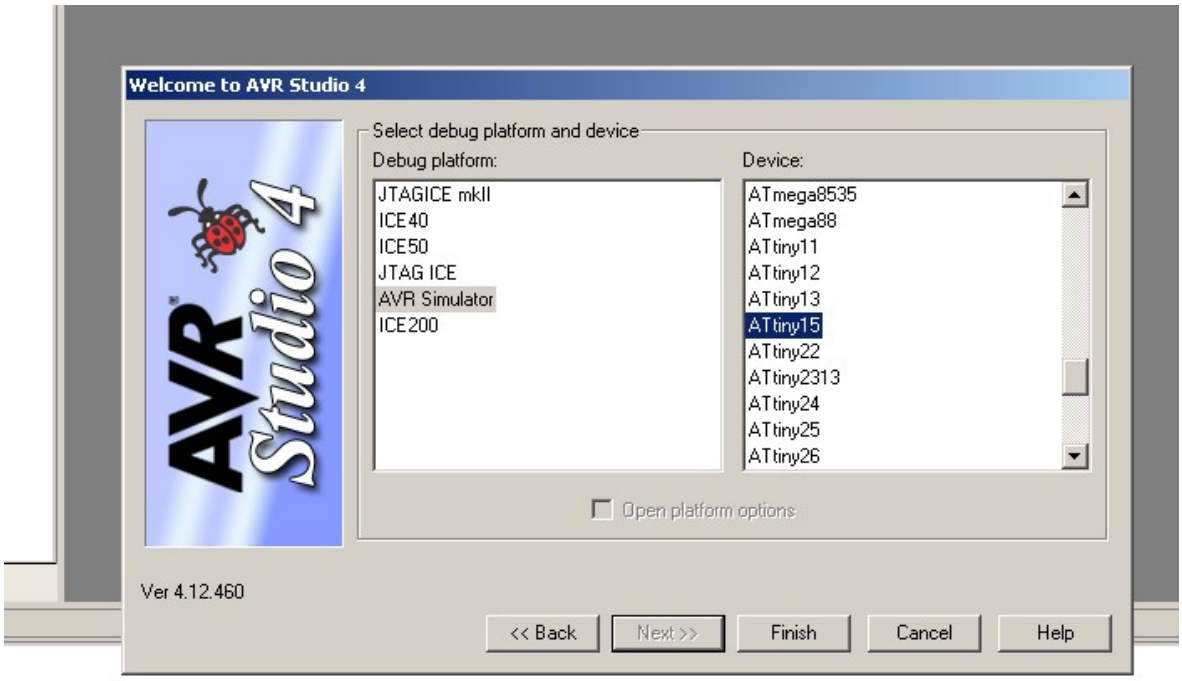

Рисунок 5.3 – Початкове вікно вибору платформи

Після натиснення кнопки “Finish” з’являються робочі вікна пакета AVR-studio, поки порожні.

Для подальшої роботи необхідно розглянути найпростіший приклад для ознайомлення з МК ATtiny15. Тому поки що обмежимося написанням і налагоджуванням найпростішої програми, що виводить на ніжку 2 логічні «0» і «1».

Потрібно в праве вікно помістити початковий текст програми. Це можна зробити двома способами: або

набрати весь текст безпосередньо у вікні редактора, або завантажити вже існуючий файл. Нижче наведений повний текст простої програми з коментарями (рисунок 5.4). Коментарі можна не вписувати у вікно. Вони потрібні для розуміння тих команд, що набираються.

Перелік команд Assembler наведено у додатках Б, В.

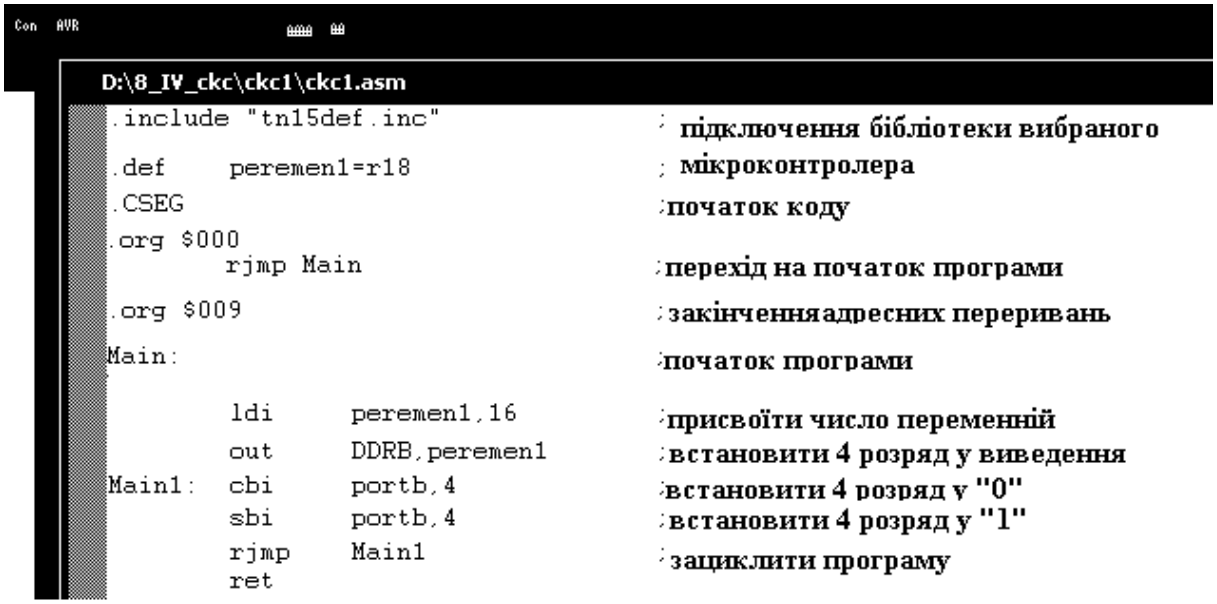

Рисунок 5.4 – Вікно набору програм

Записати програму та натиснути на значок . Якщо ви все зробили правильно, знизу у вікні є зелений значок (рисунок 5.5).

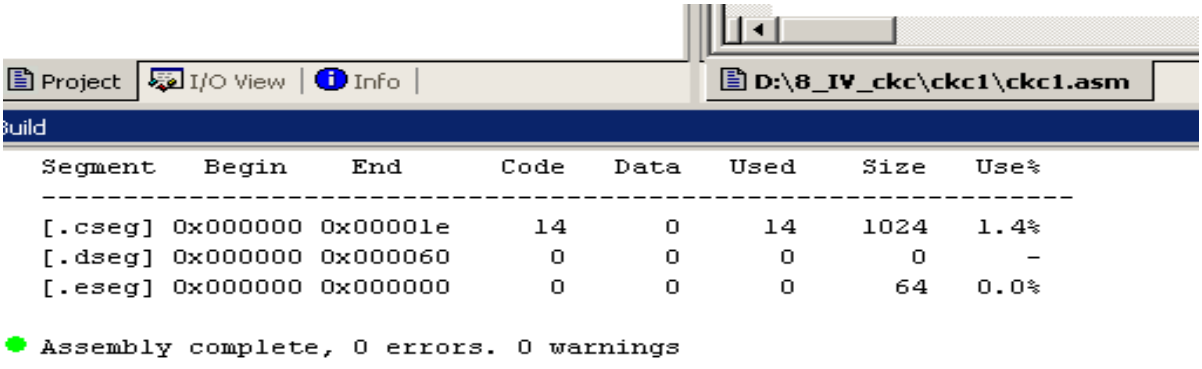

Рисунок 5.5 – Вікно попереджень без помилок

Якщо десь було зроблено помилку, наприклад замість команди ldi записано ld, то у вікні є попередження (рисунок 5.6).

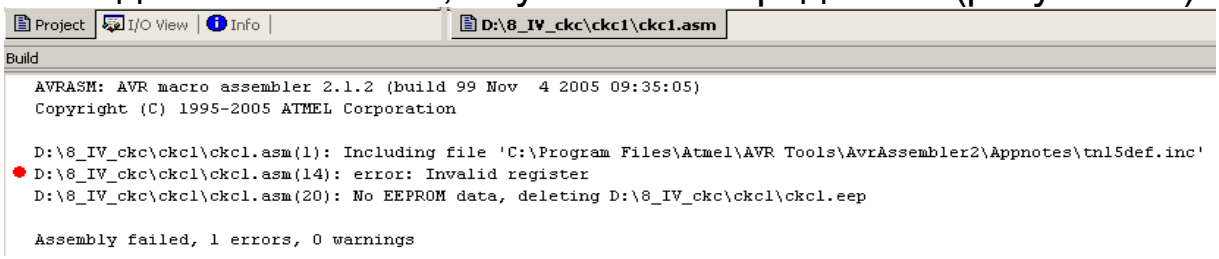

Рисунок 5.6 – Вікно попереджень з помилкою

Після проведених дій на диску з'явився файл у форматі .hex, який вже можна завантажувати в мікросхему мікроконтролера. Але повний цикл роботи в середовищі AVR-studio був би неповним, тому ми переходимо до стадії налагоджування програм. Це робиться командою \Debug\ Start Debugging (рисунок 5.7).

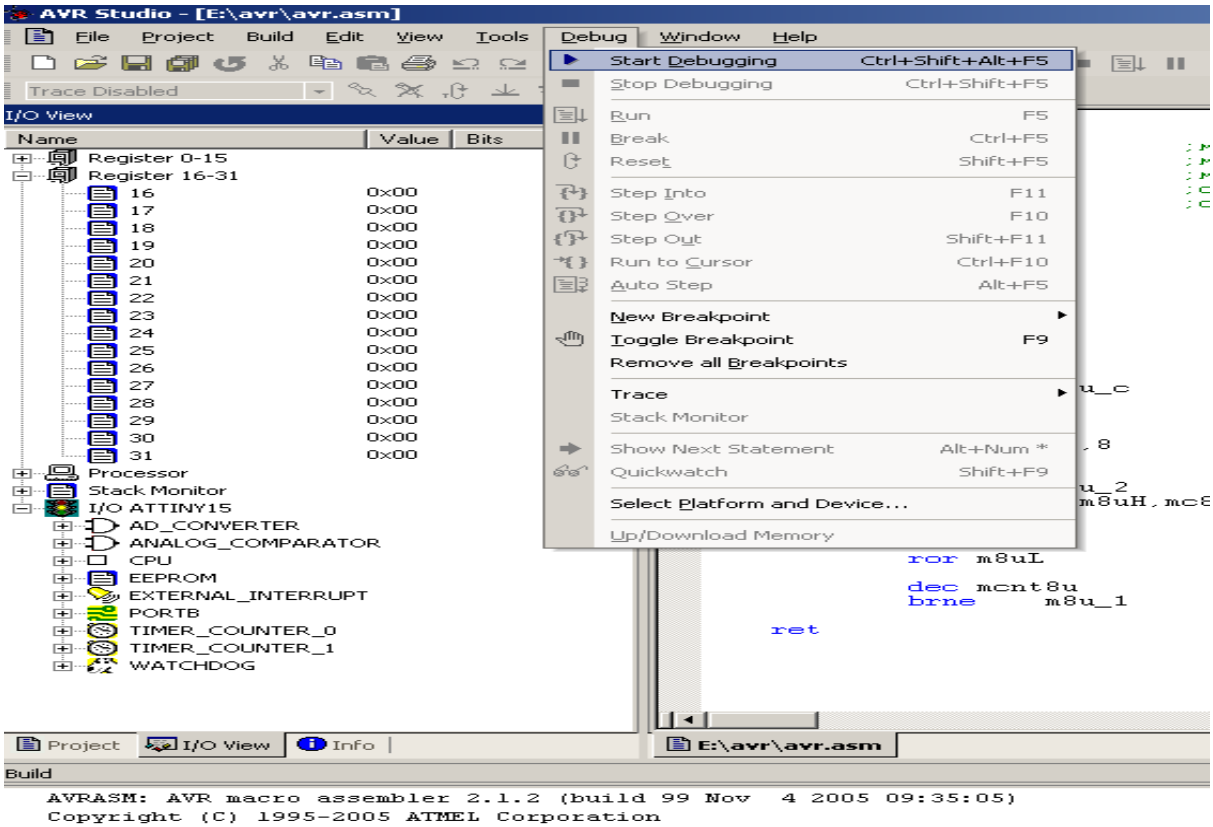

Рисунок 5.7 – Початок налагоджування програми

Пакет AVR-studio містить могутні засоби для перегляду і редагування стану внутрішніх регістрів і портів введення-виведення мікроконтролера, а також час виконання програми. Доступ до них здійснюється через вікно “I/O”. Насправді, кількість інформації, доступна через вікна перегляду пакета AVR-studio, настільки велика, що для отримання максимального комфорту потрібно використовувати комп'ютер у двомоніторній конфігурації.

Тепер встановлюємо у вікні “Simulator Options” частоту кварцу 1,0 МГц для точного вимірювання часу виконання програми (рисунок 5.8).

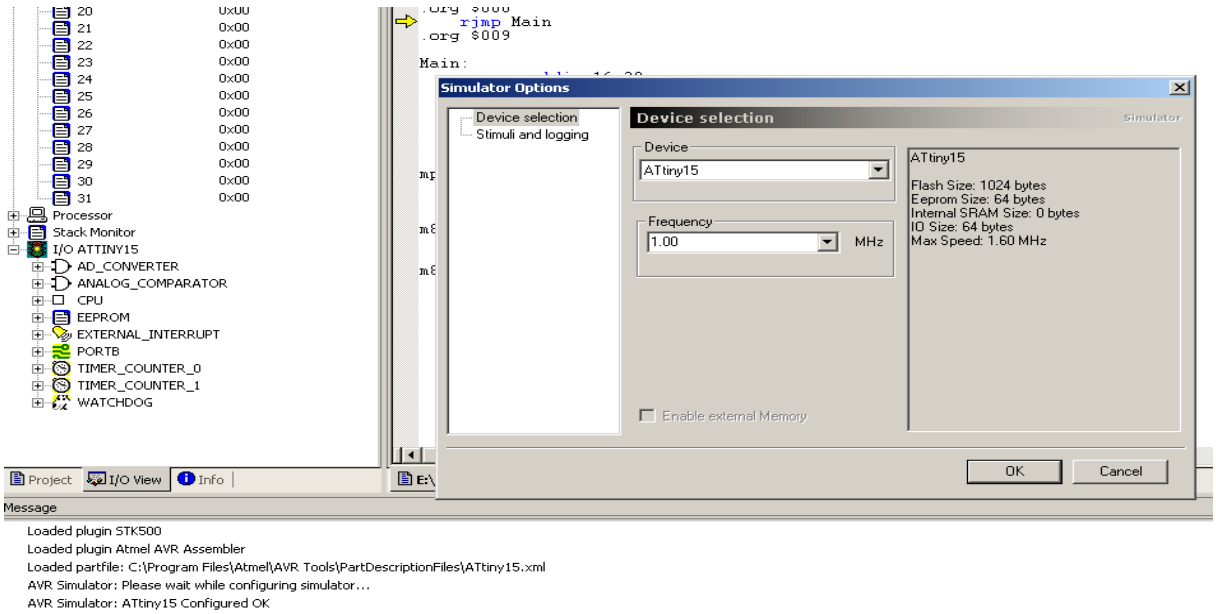

Рисунок 5.8 – Встановлення частоти роботи мікроконтролера

Інші опції потрібно залишити без зміни. Тепер можна виконувати програму в покроковому режимі за допомогою миші або кнопки F11.

Щоб отримати доступ до бітів порту B, треба розкрити рядок I/O, а потім рядок PORTB. Тепер видно всі три

регістри цього порту – PORTB, DDRB і PINB. Щоб побачити поля Value, Bits і Address, доведеться розширити праву межу вікна, потіснивши при цьому вікно з початковим текстом програми. Тепер, проходячи програму в покроковому режимі, можна бачити зміну поточних станів цих регістрів у полі Bits. Є можливість оперативної зміни стану будь-якого біта регістрів порту, причому це можна робити або записом нового коду у полі Value, або безпосередньо натиснувши мишею на потрібному біті регістра.

Увага! Директиви Assembler не те саме, що команди. Директиви вказують Assembler, що в написаній програмістом програмі є деякі особливості. При компіляції Assembler, спираючись на директиви, вносить зміни. Так, .include «tn15def.inc» – директива, що вказує на необхідність підключити бібліотеку для мікроконтролера ATtiny15L. Перелік директив наведено у додатку Г.

Таким чином, у розділі наведено всі загальні положення з роботи в середовищі AVR-studio. Подальший розгляд можливостей пакета краще проводити під час роботи.

6 ЕНЕРГОНЕЗАЛЕЖНА ПАМ'ЯТЬ ДАНИХ (EEPROM)

МК ATTINY 15 має EEPROM. Ця пам'ять розташована у власному адресному просторі, а її об'єм складає 64 байти.

Доступ до EEPROM. Пам'ять вимагає використання трьох РВВ: регістра адреси (EEAR), регістра даних (EEDR), регістра управління (EECR).

У регістр управління (EECR), розташований за адресою $1C, завантажують команди керування, використовують тільки 4 біти (таблиця 6.1), цей регістр доступний для читання і запису.

Регістр адреси (EEAR) розташовано за адресою $1E. У цей регістр завантажують адресу EEPROM, до якої буде звертання. Регістр доступний для читання і запису. Оскільки всього адрес EEPROM – 64, то достатньо перших 6 бітів

регістра. Інші ігноруються.

Таблиця 6.1 – Регістр управління EEPROM

Розряд	Назва	Опис
7…4	-	Не використовується. Читається як «0»
3	EERIE	Дозволити переривання від EEPROM. Якщо цей розряд встановлений в «1», переривання дозволено. Під час запису в EEPROM переривання заборонені
2	EEMWE	Керування дозволом запису в EEPROM. Якщо цей розряд встановлений в «1» та EERIE встановлений в «1», то дозволено запис в EEPROM. Апаратно скидається в «0» через 4 цикли генератора
1	EEWE	Запис в EEPROM. Якщо цей розряд та EEMWE встановлені в «1», то починається запис в EEPROM
0	EERE	Читання з EEPROM. Якщо цей розряд встановлено в «1», то починається запис в EEPROM. Після закінчення читання розряд скидається

Регістр даних (EEDR) розташований за адресою $1D. У цей регістр завантажують дані, які будуть поміщені в пам'ять, до якої буде звертання програми. Доступний для читання і запису.

Етапи запису в EEPROM:

1) дочекатися готовності EEPROM до запису даних (чекати поки не скинеться прапорець EEWE регістра управління);

2) завантажити байт даних у регістр даних, а адресу в регістр адреси;

3) встановити в «1» прапорець EEMWE регістра

управління EEPROM;

4) протягом 4 машинних циклів після встановлення прапорця EEMWE записати «1» в розряд EEWE регістра управління.

Після запису (декілька мікросекунд – залежить від мікроконтролера) прапорець EEWE апаратно скинеться і EEPROM готовий до наступного запису.

Важливо пам'ятати, що якщо в процесі запису в EEPROM відбудеться переривання, то запис буде зірвано. Тому необхідно забороняти переривання на період запису в EEPROM. Нижче приклад програми.

```
.include «tn15def.inc»
.CSEG
.ORG $000
      Rjmp      Main
.ORG $009
Main:
      Ldi   r20,16
      Ldi   r21,4
      Ldi   r22,4
      Rcall writeEEPROM
      Rcall readEEPROM
Ret
WriteEEPROM:
      Sbic $1C,1
      Rjmp writeEEPROM
      Cli
      Out  $1D,r20
      Out  $1E,r21
      Sbi  $1C,2
      Sbi  $1C,1
      Sei
Ret
ReadEEPROM:
      Sbic $1C,0
      Rjmp ReadEEPROM
```

```
        Out  $1C,r22
        Sbi  $1C,0
        In   r23,$1d
 Ret
```

Процес читання набагато простіший.

Недоліки EEPROM. При зниженні напруги живлення, збереженої в EEPROM, дані можуть бути втрачені або пошкоджені із таких причин:

1) якщо напруга живлення під час запису буде нижче від певної величини, то запис буде проведено некоректно;
2) мікроконтролер сам виконує команди запису некоректно при зниженому живленні.

Причина цього – електричне програмування вимагає підвищеної потужності живлення в процесі запису. Якщо живлення недостатньо, напруга під час запису може знизитися нижче робочого мінімуму.

7 РЕЖИМ ЗНИЖЕНОГО ЕНЕРГОСПОЖИВАННЯ

Режим зниженого енергоспоживання необхідний для різних випадків використання пристроїв, що містять мікроконтролер. Якщо узяти мобільний телефон або годинник, то стає зрозуміло, що зниження енергоспоживання дозволить довше працювати елементам живлення без заміни або заряджання. Але в сучасних мікроконтролерах є ще додаткові функції, такі як вимірювання (аналого-цифровий перетворювач). Будь-який електричний фон впливає на якість вимірювань, оскільки всі пристрої знаходяться на одному кришталі. Відключивши всі пристрої мікроконтролера, які не потрібні для вимірювань, можна провести ці вимірювання точніше. У різних мікроконтролерах різні варіанти режимів зниженого енергоспоживання. У нашому їх три.

Перемикання мікроконтролера в будь-який з можливих

сплячих режимів здійснює команда SLEEP, при цьому прапорець SE регістра керування MCUCR повинен бути встановлений в «1». Щоб уникнути ненавмисного перемикання мікроконтролера в сплячий режим, рекомендується цей прапорець встановлювати безпосередньо перед командою SLEEP. Вибір сплячого режиму, в який буде переведено мікроконтролер, встановлюється програмістом у регістрі MCUCR (таблиця 7.1).

Таблиця 7.1 – Режим зниженого енергоспоживання

PBB MCUCR

SM1	SM0	Режим
0	0	IDLE
0	1	ADC Noise Redaction
1	0	Power Down
1	1	Зарезервовано

Вихід зі сплячого режиму здійснюється перериванням або скиданням. При скиданні мікроконтролер почне роботу з початку програми.

Режим IDLE

У цьому режимі роботу припиняє тільки центральний процесорний пристрій. Всі інші периферійні пристрої (таймери-лічильники, аналоговий компаратор, АЦП, сторожовий таймер), а також система переривань продовжують функціонувати. З цієї причини вихід з даного режиму можливий по зовнішньому і внутрішньому перериваннях.

Режим Power Down

У цьому режимі припиняється функціонування всіх систем мікроконтролера включаючи тактовий генератор. Продовжують працювати тільки сторожовий таймер (якщо його включено) і підсистема обробки зовнішніх переривань.

Вихід з даного стану можливий такий: зовнішнє

скидання; скидання сторожового таймера; генерація зовнішнього переривання по рівню; генерація переривання через зміну стану виводу.

Режим ADC Noise Reduction

Цей режим є тільки в мікроконтролерах з вбудованими АЦП. У роботі залишаються тільки АЦП, підсистема обробки зовнішніх переривань, сторожовий таймер і тактовий генератор. За рахунок вимкнення ЦПУ і ряду внутрішніх пристроїв перешкоди на входах АЦП знижуються.

Вихід з даного стану можливий такий: зовнішнє скидання; скидання сторожового таймера; генерація зовнішнього переривання по рівню; генерація переривання через зміну стану виводу; генерація переривання від АЦП.

Приклад програми.

```
.include «tn15def.inc»
.CSEG
.ORG $000
      Rjmp      Main
.ORG $009
Main:
      Ldi       r19,49
      Out       MCUCR,r19
      Sleep
Ret
```

Ця програма використовує сплячий режим Power Down.

8 СКИДАННЯ МІКРОКОНТРОЛЕРА

Скидання переводить мікроконтролер у початковий стан, тобто всі РВВ встановлюються в стан за умовчанням, а в лічильник команд завантажується нульова адреса. Скидання відбувається:

1) включенням напруги живлення;

2) зниженням напруги живлення нижче заданої величини;

3) скиданням сторожового таймера;

4) поданням сигналу низького рівня на вивід RESET.

Причина останнього скидання залишається у РВВ MCUSR (таблиця 8.1).

Таблиця 8.1 – Опис РВВ MCUSR

Розряд	Назва	Опис
7…4	-	Не використовується. Читається як «0»
3	WDRF	Прапорець скидання сторожового таймера встановлюється в «1», якщо джерелом скидання був сторожовий таймер
2	BORF	Прапорець скидання зі зниження напруги живлення встановлюється в «1», якщо джерелом скидання було критичне зниження напруги
1	EXTRF	Прапорець апаратного скидання встановлюється в «1», якщо джерелом скидання було подача «0» на вивід скидання мікроконтролера
0	PORF	Прапорець скидання при включенні мікроконтролера встановлюється в «1», якщо мікроконтролер було ввімкнено

9 ПОРТИ ВВЕДЕННЯ-ВИВЕДЕННЯ

У мікроконтролерах AVR порти введення–виведення мають двонаправлену структуру. Це означає, що залежно від стану відповідних бітів регістрів керування порт буде виводити інформацію з мікроконтролера або приймати. Взагалі порти введення-виведення – це фактично ніжки мікросхеми, підключені до кришталю.

Звернення до портів введення-виведення проводиться

через РВВ. У моделі ATtiny15 використовується три РВВ (таблиця 9.1).

Таблиця 9.1 – РВВ порту введення-виведення

Порт	Регістр	Назва	Адреса
B	PORTB	Регістр даних порту B	$18
	DDRB	Регістр напрямку порту B	$17
	PINB	Регістр виводів порту B	$16

PINB – доступний тільки для читання, оскільки за адресою цього регістра здійснюється доступ до фізичних значень сигналів на виводах порту.

Оскільки порти є двонаправленими, то напрям вказують у спеціальному РВВ DDRB. Якщо розряд DDRB встановлений в «1», то відповідний вивід є виходом порту. Якщо розряд DDRB встановлений в «0», то відповідний вивід є входом порту.

Приклад програми.

```
.include «tn15def.inc»
.def        peremen1=r18
.CSEG
.ORG $000
      Rjmp      Main
.ORG $009
Main:
      Ldi       peremen1,16
      Out       DDRB,peremen1
Main1:
      Cbi       PORTB,4
      Sbi       PORTB,4
      Rjmp            Main1
Ret
```

Максимальна здатність навантаження виводу складає 20 мА (можна підключати світлодіоди без додаткових підсилювачів).

10 ПЕРЕРИВАННЯ

Переривання припиняють нормальний хід виконання роботи програми для виконання пріоритетної задачі. При виникненні переривання мікроконтролер зберігає в стеку вміст лічильника команд і завантажує в нього адресу відповідного вектора переривання. За цією адресою повинна знаходитись команда відносного переходу до підпрограми обробки переривання RJMP, а підпрограма обробки переривань повинна закінчуватися командою RETI.

Кожний мікроконтролер має свій набір переривань. Кожне переривання знаходиться за своєю адресою. Так, для мікроконтролера, що розглядається, переривання наведені в таблиці 10.1.

Для того щоб мікроконтролер зміг обробити переривання, необхідно встановити дозвіл переривання, встановивши в «1» відповідний біт PBB SREG (прапорець I). Після виконання переривання прапорець апаратно буде скинений (якщо необхідно дозволити нове переривання, то потрібно встановити прапорець знов) або прапорець дозволу переривань буде апаратно встановлений після закінчення виконання підпрограми переривання командою RETI. У лічильнику команд зі стеку завантажується команда, яка була перервана.

Таблиця 10.1 – Переривання мікроконтролера ATtiny15L

Джерело	Опис	Адреса
INT0	Зовнішнє переривання 0	$001
PIN_CHANGE	Зміна сигналу на виводах	$002
T1 COMPA	Збіг таймера-лічильника T1	$003
T1 OVF	Переповнення таймера-лічильника T1	$004
T0 OVF	Переповнення таймера-	$005

	лічильника T0	
EE_RDY	EEPROM готовий	$006
ANA_COMP	Аналоговий компаратор	$007
ADC	Перетворювання АЦП закінчено	$008

Важливо пам'ятати, що при роботі в підпрограмі переривання регістр SREG може бути змінений, а тоді після повернення з підпрограми переривання може бути порушена логіка виконання програми. Тому у випадку, якщо регістр буде змінюватися підпрограмою обробки переривання, необхідно зберегти його вміст, а наприкінці відновити.

Найменший час відгуку для будь-якого переривання – 4 машинні цикли. Якщо переривання відбувається під час виконання команди, то спочатку буде закінчено виконання команди, а потім відбудеться переривання. Повернення до основної програми відбувається через 4 машинні цикли. Процесор завжди виконує одну команду основної програми, перш ніж перейти до наступного переривання.

Дозволити зовнішнє переривання можливо використанням PBB GIMSK, що розташований за адресою $3B (таблиця 10.2).

Таблиця 10.2 – Опис PBB GIMSK

Розряд	Назва	Опис
7	-	Не використовується. Читається як «0»
6	INT0	Дозвіл зовнішнього переривання по виводу INT0. Якщо цей прапорець встановлено в «1» та прапорець I регістра SREG також встановлено у «1», то дозволяється зовнішнє переривання за встановленою подією у регістрі MCUCR розрядів ISC01, ISC00
5	PCIE	Дозвіл зовнішнього переривання за зміною сигналу на будь-якому виводі. Якщо цей прапорець встановлено в «1» та прапорець I регістра SREG також встановлено в «1», то дозволяється зовнішнє переривання за кожною зміною на виводах

		мікроконтролера
4…0	-	Не використовується. Читається як «0»

Зовнішні переривання можливо відслідкувати за допомогою PBB GIFR (таблиця 10.3), розташованого за адресою $3A.

Таблиця 10.3 – Опис PBB GIFR

Розряд	Назва	Опис
7	-	Не використовується. Читається як «0»
6	INTF0	Прапорець зовнішнього переривання по виводу INT0. Якщо є переривання за вказаною подією, цей прапорець встановлюється в «1» і скидається на початку підпрограми обробки переривання. Прапорець постійно в «0», якщо переривання спрацьовує за низьким рівнем сигналу
5	PCIF	Прапорець зовнішнього переривання за зміною сигналу на будь-якому виводі. Якщо є переривання за вказаної подією, цей прапорець встановлюється в «1» і скидається на початку підпрограми обробки переривання
4…0	-	Не використовується. Читається як «0»

Дозволити переривання можливо, якщо встановити в «1» відповідний біт PBB SREG (прапорець I).

Переривання виникає тільки при тривалості зовнішнього імпульсу довше за один період тактового сигналу мікроконтролера (в нашому випадку 1 мкс).

Приклад програми.

```
.include «tn15def.inc»
.def        peremen1=r18
.CSEG
.ORG $000
      Rjmp          Main
```

```
.ORG $001
      Rjmp          Perer
.ORG $009
Main:
      Ldi       r19,49
      Ldi       r20,64
      Sei
      Out       MCUCR,r19
      Out       GIMSK,r20
      Ldi       peremen1,16
      Out       DDRB,peremen1
Main1:

      Sbi       PORTB,4
      Sleep
      Rjmp          Main1
Ret
Perer:
      Cbi       PORTB,4
Reti
```

Інші переривання будуть розглянуті далі, під час вивчення периферійних пристроїв мікроконтролера.

11 ТАЙМЕРИ

Всі мікроконтролери мають у своєму складі один і більше таймерів-лічильників загального призначення. У мікроконтролера ATtiny15 – два 8-розрядні таймери-лічильники і сторожовий таймер.

Залежно від конструкції мікроконтролера таймери можуть:

1) проводити підрахунок зовнішніх імпульсів;
2) генерувати сигнал ШІМ;
3) генерувати переривання при переповнюванні регістра-лічильника;

4) генерувати переривання, досягнувши заданого в регістрі-лічильнику значення.

Сторожовий таймер використовується для запобігання зациклення програми.

11.1 Переривання від таймера

Дозволити переривання від таймера можливо при використанні РВВ TIMSK (таблиця 11.1), розташованого за адресою $39.

Таблиця 11.1 – Опис РВВ TIMSK

Розряд	Назва	Опис
7	-	Не використовується. Читається як «0»
6	OCIE1A	Прапорець дозволу переривання за збігом значення таймера-лічильника Т1 з OCR1A
5…3	-	Не використовується. Читається як «0»
2	TOIE1	Прапорець дозволу переривання за переповнюванням таймера-лічильника Т1
1	TOIE0	Прапорець дозволу переривання за переповнюванням таймера-лічильника Т0
0	-	Не використовується. Читається як «0»

Індикація переривання від таймера можлива при використанні РВВ TIFR (таблиця 11.2), розташованого за адресою $38.

Таблиця 11.2 – Опис РВВ TIMSK

Розряд	Назва	Опис
7	-	Не використовується. Читається як «0».
6	OCIF1A	Прапорець індикації переривання за збігом значення таймера-лічильника Т1 з регістром OCR1A
5…3	-	Не використовується. Читається як «0»

2	TOIF1	Прапорець індикації переривання за переповнюванням таймера-лічильника T1
1	TOIF0	Прапорець індикації переривання за переповнюванням таймера-лічильника T0
0	-	Не використовується. Читається як «0»

Далі, під час розгляду таймерів мікроконтролера, буде розглянуто роботу цих переривань більш детально.

11.2 Таймер-лічильник T0

Використовується для формування тимчасових інтервалів (T0) та рахування числа зовнішніх імпульсів (СК), наприклад натискання кнопки. Структурна схема наведена на рисунку 11.1.

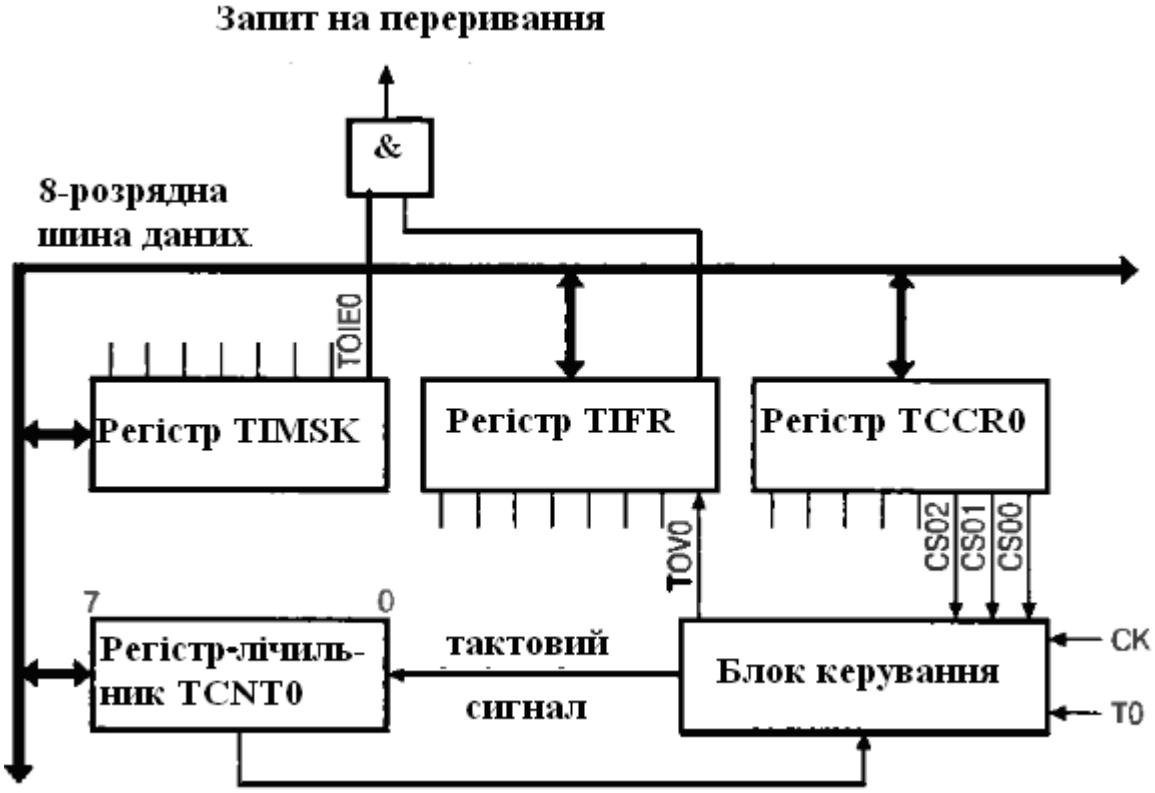

Рисунок 11.1 – Структурна схема таймера T0

TCNT0 розташований за адресою $32 і доступний у

будь-який момент часу для читання і запису. При переповнюванні регістра встановлюється прапорець T0V0 регістра TIFR і генерується запит на переривання. Дозвіл переривання здійснюється встановленням в «1» розряду T0IE0 регістра TIMSK.

Увага! Повинен бути встановлений прапорець загального дозволу переривань I регістра SREG.

У режимі таймера на внутрішній вхід T0 надходять імпульси тактового сигналу мікроконтролера безпосередньо або через роздільник. У режимі лічильника збільшення величини регістра-лічильника відбувається за появи активного фронту сигналу на вході T0 мікроконтролера. Регістр керування таймера T0 TCCR0 (таблиці 11.3, 11.4) розташований за адресою $33.

Таблиця 11.3 – Опис РВВ TCCR0

Розряд	Назва	Опис
2…0	CS02…CS00	Вибір джерела тактового сигналу

Таблиця 11.4 – Опис роздільника РВВ TCCR0

CS02	CS01	CS00	Джерело тактового сигналу
0	0	0	Таймер-лічильник зупинено
0	0	1	СК (тактовий сигнал мікроконтролера)
0	1	0	СК/8
0	1	1	СК/64
1	0	0	СК/256
1	0	1	СК/1024
1	1	0	Вивід T0. Збільшення лічильника настає за спадним фронтом імпульсу
1	1	1	Вивід T1. Збільшення лічильника настає за наростаючим фронтом імпульсу

У мікроконтролері ATtiny15 можливо скинути коефіцієнт

передподільника. Скидання здійснюється записом «1» у відповідний розряд регістра SFIOR (таблиця 11.5), що знаходиться за адресою $2C.

Таблиця 11.5 – Опис PBB SFIOR

Розряд	Назва	Опис
7…3	-	Не використовується. Читається як «0»
2	FOC1A	Якщо прапорець встановлено в «1», то настає примусове формування події збігу
1	PSR1	Якщо прапорець встановлено в «1», то настає скидання передподільника таймера-лічильника T1
0	PSR0	Якщо прапорець встановлено в «1», то настає скидання передподільника таймера-лічильника T0

Увага! Збільшення регістра-лічильника відбувається як для входу відповідного виводу мікроконтролера, так і для виходу. Проміжок між імпульсами повинен бути більше періоду тактового сигналу мікроконтролера.

11.3 Таймер-лічильник T1

Відсутня можливість підрахунку зовнішніх імпульсів.
Важливою властивістю T1 є:

1) здатність виконувати певні дії при дорівнюванні рахункового регістра заданому значенню;
2) можливість працювати як генератор ШІМ.

До складу T1 включено три 8-розрядні регістри (регістр-лічильник TCNT1 і два регістри порівняння OCR1A, OCR1B), два 8-розрядні компаратори, два регістри керування (регістр керування TCCR1 і регістр спеціальних функцій SFIOR), а також блок керування таймером.

Всі прапорці стану (переповнювання і збігу) знаходяться у регістрі TIFR, а дозвіл і заборона переривань від таймера здійснюється встановленням (скиданням) прапорців регістра TIMSK. Структурна схема таймера Т1 наведена на рисунок 11.2.

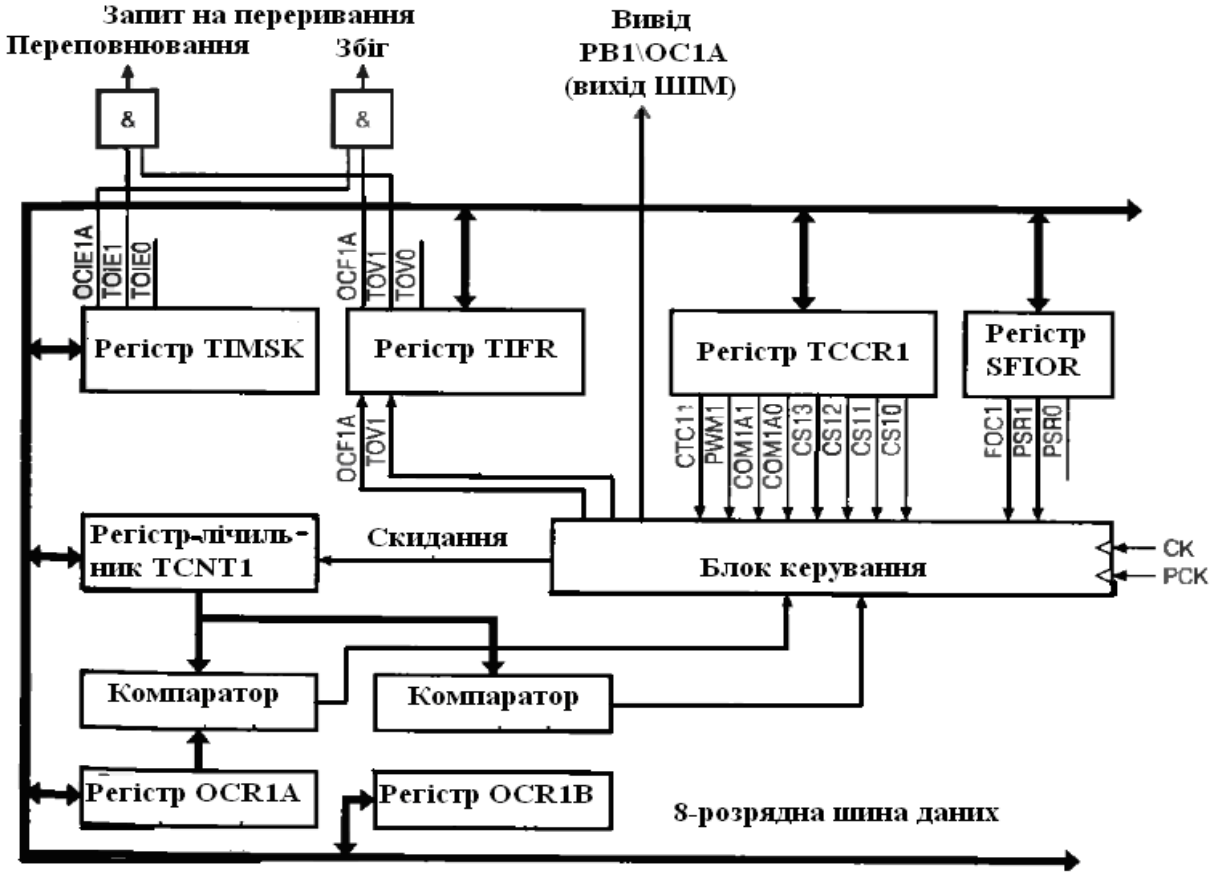

Рисунок 11.2 – Структурна схема таймера Т1

Регістр-лічильник TCNT1 знаходиться за адресою $2F. Регістр керування TCCR1 (таблиці 11.6, 11.7) знаходиться за адресою $30.

Таблиця 11.6 – Опис РВВ TCCR1

Розряд	Назва	Опис
7	CTC1	Скидання таймера-лічильника за збігом («1» – скинути якщо подія наступила)

6	PWM1	Дозвіл ШІМ («1» – увімкнути режим ШІМ)
5,4	COM1A1,0	Режим роботи виводу PB1 (OC1A)
3…0	CS13…CS10	Вибір джерела тактового сигналу

Таблиця 11.7 – Опис передподільника РВВ TCCR1

CS02	CS01	CS01	CS00	Джерело тактового сигналу
0	0	0	0	Таймер-лічильник зупинено
0	0	0	1	СК*16 (РСК)
0	0	1	0	СК*8
0	0	1	1	СК*4
0	1	0	0	СК*2
0	1	0	1	СК (тактовий сигнал мікроконтролера)
0	1	1	0	СК/2
0	1	1	1	СК/4
1	0	0	0	СК/8
1	0	0	1	СК/16
1	0	1	0	СК/32
1	0	1	1	СК/64
1	1	0	0	СК/128
1	1	0	1	СК/256
1	1	1	0	СК/512
1	1	1	1	СК/1024

Регістр спеціальних функцій SFIOR (таблиця 11.8) знаходиться за адресою $2C.

Таблиця 11.8 – Опис РВВ SFIOR

Розряд	Назва	Опис
7…3	-	Не використовується. Читається як «0»
2	FOC1A	Якщо прапорець встановлено в «1», то настає примусове формування події збігу
1	PSR1	Якщо прапорець встановлено в «1», то настає

		скидання передподільника таймера-лічильника T1
0	PSR0	Якщо прапорець встановлено в «1», то настає скидання передподільника таймера-лічильника T0

При переповнюванні регістра-лічильника встановлюється прапорець T0V1 регістра TIFR і генерується запит на переривання. Дозвіл переривання здійснюється встановленням в «1» розряду T0IE1 регістра TIMSK. Увага! Повинен бути встановлений прапорець загального дозволу переривань I регістра SREG.

Важливе додавання до функцій T1 відносно T0. При кожній зміні регістра-лічильника здійснюється порівняння його вмісту з числом, що знаходиться в регістрі OCR1A. При збігу значень встановлюється прапорець OCF1A регістра TIFR і генерується запит на переривання. Дозвіл переривання здійснюється встановленням в «1» розряду OCIE1A регістра TIMSK. Увага! Повинен бути встановлений прапорець загального дозволу переривань I регістра SREG.

Крім генерації переривання, можуть також виконуватися скидання таймера-лічильника або зміни стану виводу PB1 (OC1A) МК. Ці дії визначаються станом розрядів TCCR1 (таблиця 11.9).

Таблиця 11.9 – Можливі стани розрядів PBB TCCR1

Розряд	Назва	Опис
7	CTC1	Скидання таймера-лічильника за збігом. Якщо в «1» скинути при настанні події, то при збігу регістра-лічильника та регістра OCR1A буде зроблено скидання регістра-лічильника в «$00»
5,4	COM1A1,0	Керування виводом PB1 (**не для режиму ШІМ**). Стан цих розрядів визначає поведінку виводу PB1 при збігу регістра-лічильника та регістра OCR1A. Вивід PB1 потрібно сконфігурувати як вихід. Поведінка виводу задається таким чином: «0»,«0» – таймер-лічильник вимкнено від PB1 «0»,«1» – стан виводу змінюється на

		протилежний «1»,«0» – вивід скидається в «0» «1»,«1» – вивід встановлюється в «1»

Стан виводу PB1 може бути змінено і примусово. Для цього необхідно в розряд FOCIA регістра SFIOR записати «1». Тоді стан PB1 змінюється у відповідності з вмістом COM1A0 і COM1A1. Переривання при цьому не генерується.

11.4 Режим ШІМ таймера T1

У цьому режимі T1 перетворюється на генератор ШІМ (рисунок 11.3) і використовується для генерації сигналу з програмованою частотою і шпаруватістю.

Рисунок 11.3 – Приклад шпаруватості ШІМ

Для переведення T1 в режим ШІМ необхідно встановити в «1» розряд PWM1 регістра TCCR1. При роботі T1 в режимі ШІМ стан регістра-лічильника змінюється від $00 до значення, що знаходиться в регістрі OCR1B. Після чого регістр-лічильник скидається та цикл повторюється. При рівності регістра-лічильника, що міститься, та регістра OCR1A стан виводу PB1 (OC1A) мікроконтролера змінюється відповідно до значення розрядів COM1A1:COM1A0 регістра TCCR1 (таблиці 11.10, 11.11).

Таким чином, вміст регістра OCR1A визначає шпаруватість ШІМ-імпульсів, а вміст регістра OCR1B – його частоту.

Таблиця 11.10 – Можлива частота ШІМ-сигналів

Частота тактового сигналу таймера-лічильника	Регістр OCR1B	Частота ШІМ сигналу, кГц
СК	159	10
РСК/8	159	20
РСК/4	213	30
РСК/4	159	40
РСК/2	255	50
РСК/2	213	60
РСК/2	181	70
РСК/2	159	80
РСК/2	141	90
РСК	255	100
РСК	231	110
РСК	213	120
РСК	195	130
РСК	181	140
РСК	169	150

Таблиця 11.11 – Можливі стани виводу РВ1 в режимі ШІМ

COM1A1	COM1A0	Стани виводу PB1
0	0	Таймер-лічильник вимкнено від PB1
0	1	Таймер-лічильник вимкнено від PB1
1	0	Неінвертований ШІМ
1	1	Інвертований ШІМ

Відповідно до величини сигналу різниця між ним та OCR1A або OCR1B дорівнює $00, вихід схеми порівняння перемкнеться в стійкий стан (рисунок 11.4, таблиця 11.12).

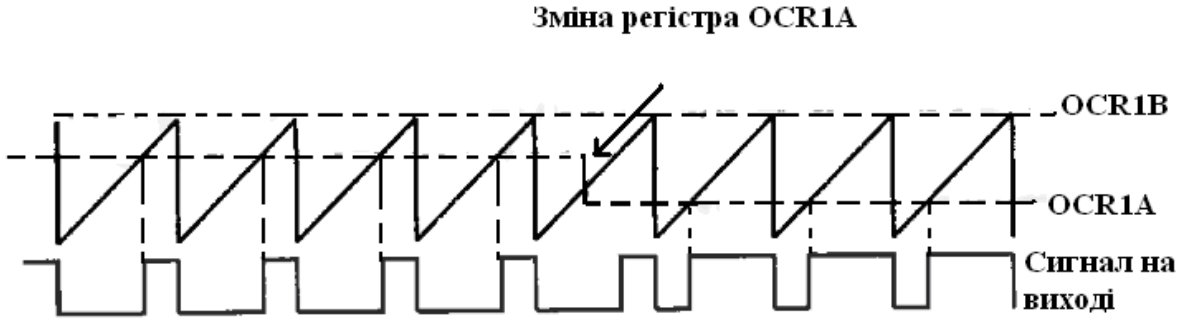

Рисунок 11.4 – Формування ШІМ-сигналів

Таблиця 11.12 – Можливі стани виводу PB1 в режимі ШІМ

COM1A1	COM1A0	Регістр OCR1A	Стани виводу PB1
1	0	$000	0
1	0	OCR1B	1
1	1	$000	1
1	1	OCR1B	0

У схемі ШІМ ухвалені заходи, що дозволяють уникнути появи несиметричних викидів. Для цього зміна регістра OCR1A відбувається тільки у момент досягнення лічильником максимального значення. До цього моменту записане у регістр OCR1A число зберігається в спеціальному тимчасовому регістрі. Відповідно при читанні регістра порівняння в проміжку між записом у нього і його дійсними змінами повертається тимчасовий вміст регістра.

Таким чином, при читанні регістра завжди повертається значення, записане останнім.

І останнє. При роботі Т1 в режимі ШІМ може генеруватися переривання за переповнюванням регістра-

лічильника таймера-лічильника, а також переривання від схеми порівняння. Дозвіл і обробка відповідних переривань виконується звичайним способом.

11.5 Сторожовий таймер

Основна функція сторожового таймера – захист пристрою від збоїв. Завдяки сторожовому таймеру можна перервати виконання програми, що зациклилася, або вийти з інших непередбачених ситуацій, перешкоджаючих нормальному виконанню програми. Особливо важливо це в автоматичних системах, де зациклювання може призвести до виходу з ладу всього обладнання. Структурна схема сторожового таймера наведена на рисунку 11.5.

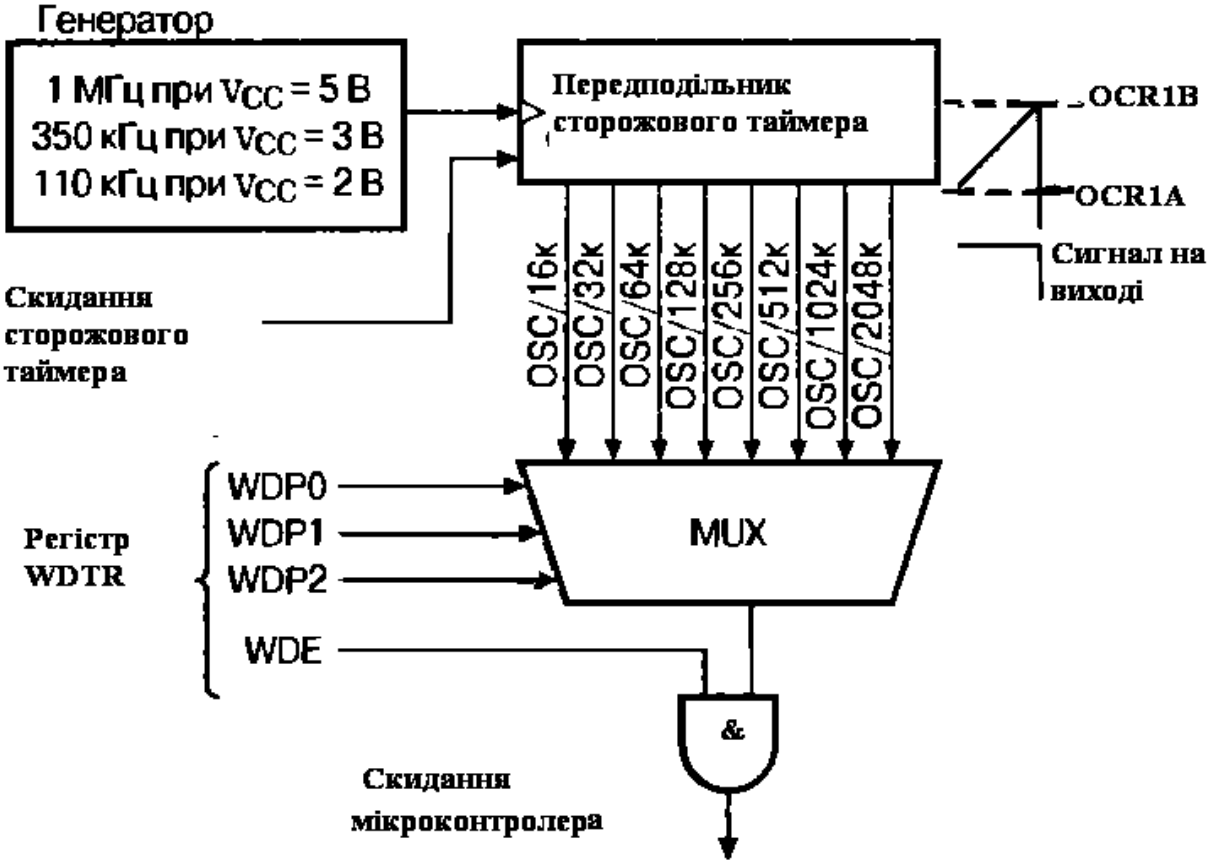

Рисунок 11.5 – Структурна схема сторожового таймера

Сторожовий таймер має незалежний тактовий генератор і працює навіть у сплячому режимі POWER DOWN. Частота цього генератора залежить від напруги живлення пристрою, температури, технологічного розкиду. Типові значення частот: 1МГц при напрузі 5 В; 350 кГц – 3 В; 110 кГц – 2 В.

Якщо сторожовий таймер увімкнено, то через певні проміжки часу (настання тайм-ауту) виконується скидання мікроконтролера. При нормальному виконанні програми сторожовий таймер повинен періодично (через проміжки часу, менші за його період) скидатися командою асемблера WDR.

Для управління сторожовим таймером призначений регістр WDTCR (таблиця 11.13), розташований за адресою $21.

Таблиця 11.13 – Опис РВВ WDTCR

Розряд	Назва	Опис
7…5	-	Не використовується. Читається як «0»
4	WDTOE	Дозволити вимкнення сторожового таймера
3	WDE	Вимкнути сторожовий таймер («1»)
2…0	WDP2…WDP0	Коефіцієнт поділення передподільника сторожового таймера

Для вмикання-вимикання сторожового таймера використовуються два розряди регістра WDTCR. Якщо розряд WDE встановлено в «1», сторожовий таймер ввімкнено, «0» – вимкнено. Безпосередньо перед вмиканням таймера рекомендується зробити його скидання командою WDR, для випадку, якщо в лічильника таймера виявилося не нульове значення, таймер спрацює раніше передбачуваного програмістом терміну (таблиця 11.14).

Бувають випадки, коли таймер може бути ненавмисно

вимкнений. Щоб цього не сталося, використовується розряд WDTOE. Це додатковий захист. Вимкнення сторожового таймера можливе тільки при послідовному скиданні розрядів WDTOE, а потім WDE. Причому скидання WDE повинне відбутися протягом 4 машинних циклів, інакше WDTOE знов скинеться в «0» і процедуру вимкнення доведеться почати знов. Така процедура дозволяє практично повне виключення випадкового вимкнення сторожового таймера.

Період настання тайм-ауту сторожового таймера задається за допомогою розрядів WDP2…WDP0 регістра WDTR (таблиця 11.14).

Таблиця 11.14 – Періоди настання тайм-аутів

WDP2	WDP1	WDP0	Число тактів генератора	Time-out при Vcc=2 В	Time-out при Vcc=3 В	Time-out при Vcc=5 В
0	0	0	16 К	0,15 с	47 мс	15 мс
0	0	1	32 К	0,3 с	94 мс	30 мс
0	1	0	64 К	0,6 с	0,19 с	60 мс
0	1	1	128 К	1,2 с	0,38 с	120 мс
1	0	0	256 К	2,4 с	0,75 с	240 мс
1	0	1	512 К	4,8 с	1,5 с	490 мс
1	1	0	1024 К	9,6 с	3 с	970 мс
1	1	1	2024 К	19 с	6 с	1,9 с

Приклад програми для таймера Т1.

```
.include «tn15def.inc»
.CSEG
.ORG $000
      Rjmp      Main
.ORG $003
      Rjmp      ObrobkaT1
.ORG $009
Main:
```

```
        Ldi     r25,138
        Out     TCCR1,r25
        Ldi     r24,64
        Out     TIMSK,r24
        Ldi     r26,249
        Out     OCR1A,r26
        Ldi     r20,5
        Ldi     r21,1
        Ldi     r23,150
        Sei
Main1:
        Nop
        Rjmp    Main1
Ret
WriteEEPROM:
        Sbic $1C,1
        Rjmp writeEEPROM
        Cli
        Out     $1D,r20
        Out     $1E,r21
        Sbi     $1C,2
        Sbi     $1C,1
        Sei
        Inc     r21
        Ldi     r22,0
Ret
ObrobkaT1:
        Inc     r22
        Cpse    r22,r23
        Rjmp    M1
        Rcall   WriteEEPROM
M1:
Reti
```

Таким чином, таймери є важливим периферійним пристроєм, який значно розширює можливості використання мікроконтролерів та значно підвищує їх надійність.

12 АНАЛОГОВИЙ КОМПАРАТОР ТА АНАЛОГО-ЦИФРОВИЙ ПЕРЕТВОРЮВАЧ

12.1 Аналоговий компаратор

Аналоговий компаратор є у складі всіх мікроконтролерів фірми Atmel. Аналоговий компаратор дозволяє порівнювати значення напруги, що приходять на два виводи мікроконтролера. Результатом порівняння є логічне значення, яке може бути прочитане програмно, та генероване переривання. У будь-якого компаратора є інвертований і неінвертований входи (принцип складання). AIN0 – неінвертований, AIN1 – інвертований. Щоб ці виводи можна було використовувати як компаратор, необхідно їх встановити як входи. Внутрішні підтягуючі резистори вимкнуться автоматично при дозволі роботи компаратора. Структурна схема аналогового компаратора наведена на рисунку 12.1.

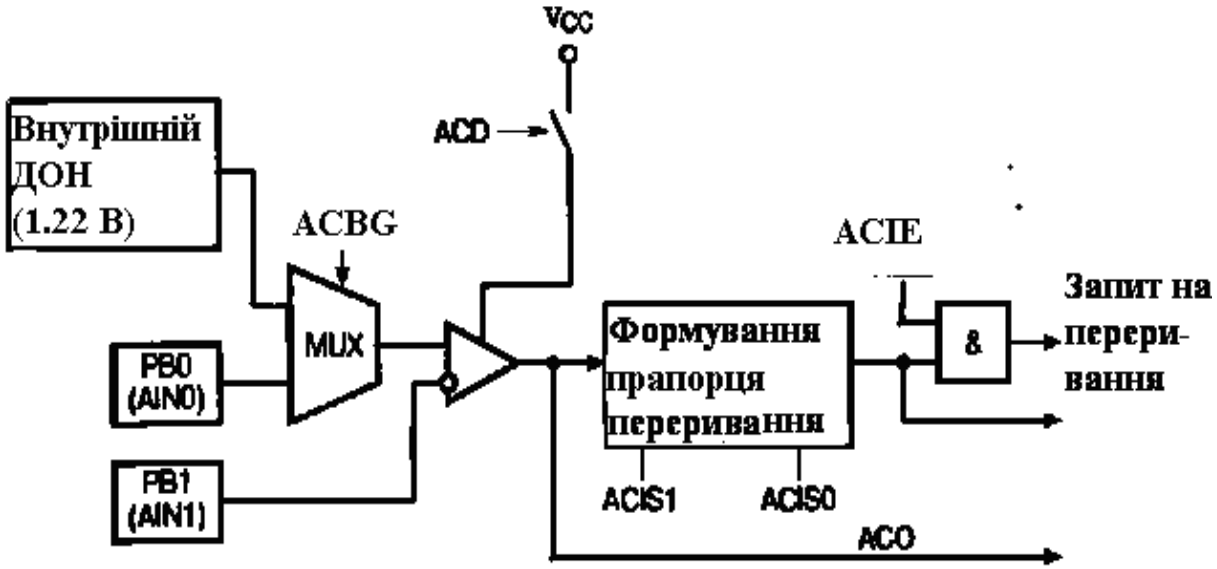

Рисунок 12.1 – Структурна схема аналогового компаратора

Керування компаратором здійснюється через регістр ACSR (таблиця 12.1), розташований за адресою $08.

У компаратора є внутрішнє джерело опорної напруги (ДОН).

Таблиця 12.1 – Опис РВВ ACSR

Розряд	Назва	Опис
7	ACD	Вимкнення компаратора («1»)
6	ACBG	Підключення до неінвертованого входу внутрішнього ДОН («1»)
5	ACO	Результат порівняння (вихід компаратора)
4	ACI	Прапорець переривання від компаратора
3	ACIE	Дозвіл переривання від компаратора
2	-	Не використовується. Читається як «0»
1,0	ACIS1,0	Умова виникнення переривання від компаратора

При підключенні живлення мікроконтролера всі розряди скидаються в «0», а отже, компаратор вмикається. При роботі з 7-мим розрядом необхідно заборонити переривання від компаратора (3-й розряд). За своєю дією компаратор працює як будь-який інший компаратор. Якщо напруга на AIN0 більше, ніж на AIN1, то результат порівняння «1», інакше «0». Результат порівняння знаходиться в 5-му розряді ACSR. Якщо за наслідками порівняння стан компаратора змінився, то встановлюється прапорець переривання (4-й розряд) і генерується запит на переривання (див. таблицю 10.1). Для дозволу переривань необхідно встановити в «1» 3-й розряд і прапорець дозволу переривань регістра SREG. Умови генерації запиту на переривання від компаратора визначаються станом розрядів 0 та 1 (таблиця 12.2).

У мікроконтролері до неінвертованого входу компаратора замість зовнішньої напруги можна підключити внутрішнє ДОН величиною 1,22 В. Для цього необхідно встановити в «1» 6-й розряд.

Таблиця 12.2 – Опис 0 та 1 розрядів РВВ ACSR

ACIS1	ACIS0	Умова виникнення переривання від компаратора
0	0	Будь-яка зміна стану виходу
0	1	Зарезервовано
1	0	Зміна стану виходу компаратора з «1» на «0»
1	1	Зміна стану виходу компаратора з «0» на «1»

12.2 Аналого-цифровий перетворювач (АЦП)

Важливою характеристикою АЦП є швидкодія. У мікроконтролері ATtiny15 вбудований АЦП послідовного наближення. Швидкодія 15 тис. вибірок за секунду. Абсолютна похибка $\pm$ 2 МЗР (молодше значення розряду).

Структурна схема АЦП наведена на рисунку 12.2.

Хоча в мікроконтролері вбудовано один фізичний АЦП, завдяки мультиплексору можна до мікроконтролера підключити до 4-х каналів аналогового сигналу з несиметричними входами або 1 канал з диференційним входом з можливістю 20-кратного попереднього посилення вхідного вимірювального сигналу. Для несиметричних входів діапазон вхідних напруг складає 0…Vcc (напруга живлення мікроконтролера), а для диференціального входу – 0…2,6 В. ДОН для АЦП може використовуватися внутрішнє (2,6 В), зовнішнє або напруга живлення мікроконтролера. АЦП може працювати в режимі поодинокого перетворення або режимі безперервного через певні інтервали часу. Керування модулем АЦП та контроль за його станом здійснюється регістром ADCSR (таблиця 12.3), розташованим за адресою $06.

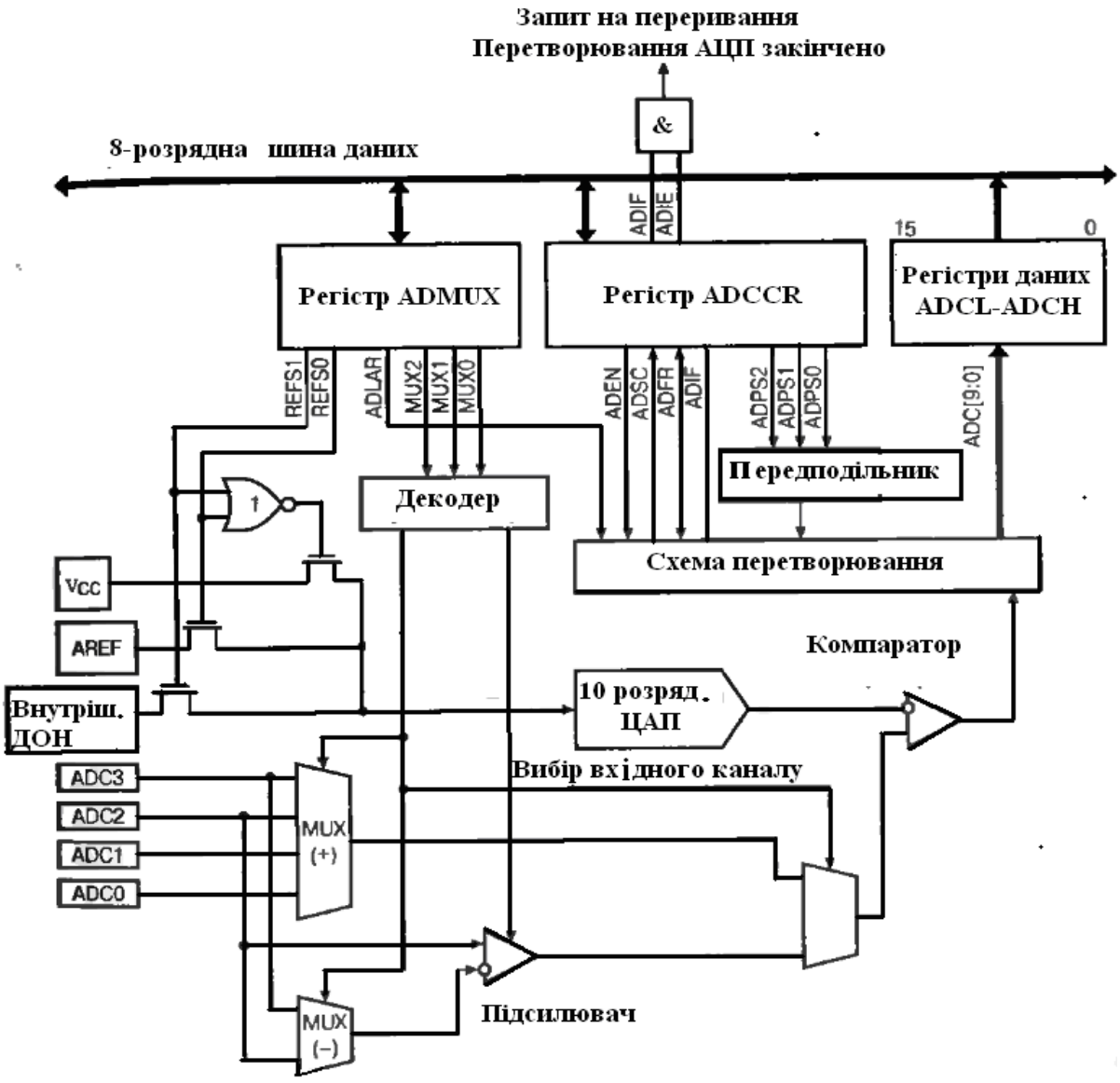

Рисунок 12.2 – Структурна схема АЦП

Таблиця 12.3 – Опис РВВ ADCSR

Розряд	Назва	Опис
7	ADEN	Вимкнення АЦП («0»)
6	ADSC	Почати перетворювання («1»)

5	ADFR	Вибір режиму АЦП («0» – поодиноке перетворювання)
4	ADIF	Прапорець переривання від АЦП
3	ADIE	Дозвіл переривання від АЦП
2…0	ADPS2…0	Вибір частоти перетворення

Якщо АЦП буде вимкнено (7-й розряд в «0»), то перетворення завершено не буде (у регістрі даних АЦП залишиться результат попереднього перетворення).

Якщо 5-й розряд встановлено в «1», то після закінчення першого перетворення автоматично буде зроблено наступне, те саме буде і далі. Якщо 5-й розряд скинений в «0», то АЦП працює в режимі поодинокого перетворення, запуск кожного перетворення здійснюється за командою програми.

І безперервне, і поодиноке перетворення починається встановленням в «1» 6-го розряду. При безперервному перетворенні це робиться один раз, а при поодинокому – кожного разу, коли необхідно провести перетворення. Тривалість циклу перетворення складає 13 циклів мікроконтролера (якщо встановлений режим поодинокого перетворення, то 6-й розряд скидається в «0»), після чого результат перетворення записується в регістр даних АЦП. Встановлюється прапорець переривання (4-й розряд) і генерується запит на переривання (див. таблицю 10.1). Як і у всіх інших перериваннях, прапорець скидається після переходу в підпрограму обробки переривання (прапорець може бути скинений програмно). Дозвіл переривання здійснюється установленням «1» в 2-й розряд при встановленому прапорці I регістра SREG.

Якщо АЦП працює в режимі безперервного перетворення, новий цикл почнеться відразу після запису результату. У режимі поодинокого перетворення – відразу після скидання 6-го розряду.

У ряді випадків АЦП можуть знадобитися 25 тактів на одне перетворення. Це відбувається при запуску першого перетворення після настання таких подій:

1) вмикання АЦП;
2) зміна ДОН;
3) вмикання каналу з диференційним входом.

Протягом цієї затримки відбувається ініціалізація АЦП.

Розряди 0-2 регістра ADCSR встановлює передподільник тривалості перетворення (таблиця 12.4).

Таблиця 12.4 – Встановлення передподільника тривалості перетворення

ADPS2	ADPS 1	ADPS 0	Коефіцієнт передподільника
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Щонайбільша точність перетворення досягається, якщо тактова частота модуля АЦП знаходиться в діапазоні 50-200 кГц. Для вибору вказаної частоти використовують передподільник.

Увага! Результат перетворення зберігається в регістрі даних АЦП. АЦП 10-розрядне, цей регістр розміщений фізично у двох РВВ, доступних тільки для читання, – ADCL і ADCH. Ці регістри при вмиканні мікроконтролера містять «$0000» і розташовані за адресами $05 і $04. Звернення до цих регістрів повинне проходити в строгій послідовності: ADCL – ADCH. Після звернення до регістра ADCL мікроконтролер блокує доступ до регістрів даних до тих пір, поки не буде прочитаний другий регістр ADCH. Відповідно якщо прочитаний ADCL, але поки не буде прочитаний регістр ADCH, то результати перетворень будуть втрачатися.

Для керування вирівнювання результату перетворення

служить розряд ADLAR регістра ADMUX (таблиця 12.5), розташованого за адресою $07.

Таблиця 12.5 – Опис РВВ ADMUX

Розряд	Назва	Опис
7,6	REFS1,0	Вибір джерела опорного живлення
5	ADLAR	Вирівнювання результату перетворення
4,3	-	Зарезервовано
2…0	MUX2…0	Вибір вхідного каналу

Якщо ADLAR встановлено в «1», результат перетворення вирівнюється за лівою межею 16-розрядного слова (регістри ADCH:ADCL), якщо скинений в «0», то за правою межею. Також регістром ADMUX керуються мультиплексор і вибір ДОН.

У розрядах 0-2 визначають номер активного каналу (таблиця 12.6) при несиметричному використанні АЦП і ці розряди визначають коефіцієнт попереднього посилення аналогового сигналу при диференціальному використанні АЦП.

Таблиця 12.6 – Опис 0, 1 та 2 розрядів РВВ ADMUX

MUX2	MUX 1	MUX 0	Несиметрич-ний вхід АЦП	Диферен-ційний вхід («+»)	Диферен-ційний вхід («-»)	Підси-лювач вхід-ного сигналу
0	0	0	ADC0(PB5)	Не використовують		
0	0	1	ADC1(PB2)			
0	1	0	ADC2(PB3)			
0	1	1	ADC3(PB4)			
1	0	0	Не використо-	ADC2(PB3)	ADC2(PB3)	1*
1	0	1		ADC2(PB3)	ADC2(PB3)	20*
1	1	0		ADC2(PB3)	ADC3(PB4)	1*

1	1	1	вують	ADC2(PB3)	ADC3(PB4)	20*

При виборі каналу з диференційним входом обидва входи можна підключити до одного виводу мікроконтролера. Результат перетворення в цьому випадку буде дорівнювати величині сумарного зсуву вхідного підсилювача і схеми перетворювача. Це значення згодом можна відняти від результату подальших перетворень (при тому самому коефіцієнті посилення) для зниження помилки зсуву до величини, меншої від молодшого значення розряду (МЗР).

Стан розрядів 0-2 можна змінити у будь-який момент, але якщо це буде зроблено під час циклу перетворення, зміна каналу відбудеться тільки після завершення перетворення. Це дозволяє легко сканувати канали в режимі безперервного перетворення. АЦП може використовувати будь-які ДОН (таблиця 12.7).

Таблиця 12.7 – Опис 6 та 7 розрядів PBB ADMUX

REFS1	REFS0	Джерело опорного живлення АЦП
0	0	Напруга живлення мікроконтролера тільки для каналів з несиметричними входами
0	1	Зовнішнє ДОН, підключене до виводу PB0 (AREF)
1	0	Внутрішнє ДОН, відключене від виводу PB0 (AREF), 2,56 В
1	1	Внутрішнє ДОН, підключене до виводу PB0 (AREF), 2,56 В

В останньому випадку до виводу PB0 можна підключити зовнішній фільтруючий конденсатор для підвищення якості вхідного сигналу.

Приклад програми.

```
.include «tn15def.inc»
.CSEG
.ORG $000
```

```
        Rjmp    Main
.ORG $008
        Rjmp    ObrobkaADC
.ORG $009
Main:
        Ldi     r21,17
        Ldi     r20,30
        Out     DDRB,r20
        Ldi     r20,32
        Out     ADMUX,r20
M4:
        Ldi     r20,200
        Sei
        Out     ADCSR,r20
        Ldi     r20,5
M1:
        Cpse        r23,r24
        Rjmp        M2
        Rjmp        M1
M2:
        Rcall       Leds
        Ldi     r23,0
        Rjmp        M4
Ret
        Leds:
        Rcall       Div
        Lsl     r25
        Out     PORTB,r25
        Ret
Div:
        Ldi     r25,0
M3:
        Cp      r22,r21
        Brsh    Add1
        Rjmp        Exit
Add1:
        Sub     r22,r21
```

```
        Inc         r25
        Rjmp            M3
Exit:
        Ret
ObrobkaADC:
        In          r22,ADCH
        Ldi         r23,1
        Reti
```

Підвищити точність АЦП можна різними способами:

– по-перше, правильно вибрати частоту перетворення АЦП. Це питання розглядалося раніше;

– по-друге, працюючий мікроконтролер – джерело електромагнітних перешкод. Щоб звести до мінімуму перешкоди в мікроконтролері, передбачений спеціальний сплячий режим. Цей режим був розглянутий раніше. Головна ідея цього сплячого режиму полягає в тому, щоб зупинити процесор, почати перетворення та згенерувати переривання від АЦП для переведення мікроконтролера в робочий режим після закінчення перетворення.

Також перешкоди можуть наводитися не тільки внутрішніми пристроями мікроконтролера, але і зовнішніми. Пропонується використовувати правила монтажу мікроконтролерів на платі і не нехтувати цим.

СПИСОК ЛІТЕРАТУРИ

1 Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL». – 2-е изд., стер. –- М.: Издательский дом «Додэка-XXI», 2005. – 560 с.

2 Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательство дом «Додэка-XXI», 2004. – 288 с.

3 Голубцов М.С. Микроконтроллеры AVR: от простого к сложному. – М.: СОЛОН-Пресс, 2003. – 288 с.

4 Трамперт В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров /Пер. с нем. – К.: «МК-Пресс», 2006. – 208 с.

5 Мортон Дж. Микроконтроллеры AVR: Вводный курс / Пер. с англ. – М.: Издательский дом «Додэка-XXI», 2006. – 272 с.

ДОДАТОК А

РВВ мікроконтролера ATtiny15L

Таблиця А.1

Назва	Функція	Адреса
SREG	Регістр стану	$3F
GIMSK	Загальний регістр маски переривань	$3B
GIFR	Загальний регістр прапорцяв переривань	$3A
TIMSK	Регістр маски переривань від таймерів	$39
TIFR	Регістр прапорців переривань від таймерів	$38
MCUCR	Загальний регістр керування мікроконтролера	$35
TCCR0	Регістр керування таймером Т0	$34
TCNT0	Регістр-лічильник таймера Т0	$33
MCUSR	Загальний регістр стану мікроконтролера	$32
TCCR1	Регістр керування таймером Т1	$31
TCNT1	Регістр-лічильник таймера Т1	$30
OSCCAL	Регістр настроювання генератора	$2F
OCR1A	Регістр збігу А таймера Т1	$2E
OCR1B	Регістр збігу В таймера Т1	$2D
SFIOR	Регістр спеціальних функцій	$2C
WDTSR	Регістр керування сторожового таймера	$21
EEAR	Регістр адреси EEPROM	$1E
EEDR	Регістр даних EEPROM	$1D
EECR	Регістр керування EEPROM	$1C
PORTB	Регістр даних порту В	$18
DDRB	Регістр напрямку порту В	$17
PINB	Виводи порту В	$16
ACSR	Регістр керування компаратором	$08

ADMUX	Регістр керування мультиплексора АЦП	$07
ADCSR	Регістр керування АЦП	$06
ADCH	Старший байт даних АЦП	$05
ADCL	Молодший байт даних АЦП	$04

ДОДАТОК Б

Система команд

Таблиця Б.1 – Зведені команди логічних операцій

Мнемоніка	Опис
AND Rd,Rr	Логічне множення двох РЗП
ANDI Rd,K	Логічне множення РЗП і константи
EOR Rd,Rr	Виключне додавання двох РЗП
OR Rd,Rr	Логічне додавання двох РЗП
ORI Rd,K	Логічне додавання РЗП і константи
COM Rd	Переведення в зворотний код
NEG Rd	Переведення в додатковий код
CLR Rd	Скидання всіх розрядів РЗП
SER Rd	Установлення всіх розрядів РЗП
SWAP Rd	Обмін місцями тетрад у РЗП

Таблиця Б.2 – Зведені арифметичні команди

Мнемоніка	Опис
ADD Rd,Rr	Складання двох РЗП
ADC Rd,Rr	Складання двох РЗП з перенесенням
SUB Rd,Rr	Віднімання двох РЗП
SUBI Rd,Rr	Віднімання двох РЗП з перенесенням
SBC Rd,Rr	Віднімання двох РЗП з позикою
SBCI Rd,K	Віднімання константи з РЗП з позикою
DEC Rd	Дикримент РЗП
INC Rd	Інкримент РЗП
ASR Rd	Арифметичний зсув вправо
LSL Rd	Логічний зсув вліво

LSR Rd	Логічний зсув вправо
ROL Rd	Зсув вліво через перенесення
ROR Rd	Зсув вправо через перенесення

Таблиця Б.3 – Зведені команди операцій з бітами

Мнемоніка	Опис
CBR Rd,K	Скидання розрядів РЗП
SBR Rd,K	Установлення розрядів РЗП
CBI A,b	Скидання розряду РВВ
SBI A,b	Установлення розряду РВВ
BCLR s	Скидання прапорця
BSET s	Установлення прапорця
BLD Rd,b	Завантаження розрядів РЗП з прапорця T(SREG)
BST Rd,b	Запис розрядів РЗП з прапорця T(SREG)
CLC	Скидання прапорця перенесення
SEC	Установлення прапорця перенесення
CLN	Скидання прапорця від’ємного числа
SEN	Установлення прапорця від’ємного числа
CLZ	Скидання прапорця нуля
SEZ	Установлення прапорця нуля
CLI	Загальна заборона переривань
SEI	Загальний дозвіл переривань
CLS	Скидання прапорця знака
SES	Установлення прапорця знака
CLV	Скидання прапорця переповнення додаткового коду
SEV	Установлення прапорця переповнення додаткового коду
CLT	Скидання прапорця T
SET	Установлення прапорця T
CLH	Скидання прапорця половинного перенесення
SEH	Установлення прапорця половинного перенесення

Таблиця Б.4 – Зведені команди пересилання даних

Мнемоніка	Опис
MOV Rd,Rr	Пересилання між РЗП
LDI Rd,K	Завантаження константи в РЗП
LD Rd,Z	Непряме читання
ST Z,Rr	Непрямий запис
LPM	Завантаження даних з пам’яті програм
IN Rd,A	Пересилання з РВВ у РЗП
OUT A,Rd	Пересилання з РЗП у РВВ

Таблиця Б.5 – Зведені команди передачі керування

Мнемоніка	Опис
1	2
RJMP k	Відносний безумовний перехід
RCALL k	Абсолютний перехід
RET	Повернення з підпрограми
RETI	Повернення з підпрограми обробки переривань
CP Rd,Rr	Порівняння РЗП
CPC Rd,Rr	Порівняння РЗП з врахуванням перенесення
CPI Rd,K	Порівняння РЗП з константою
CPSE Rd,Rr	Порівняння РЗП і пропусканняання наступної команди при рівності
SBRC Rr,b	Пропусканняання наступної команди, якщо розряд РЗП скинуто
SBRS Rr,b	Пропускання наступної команди, якщо розряд РЗП встановлено
SBIC A,b	Пропускання наступної команди, якщо розряд РВВ скинуто

SBIS A,b	Пропускання наступної команди, якщо розряд PBB встановлено
BRBC s,k	Перехід, якщо прапорець s регістра SREG скинуто
BRBS s,k	Перехід, якщо прапорець s регістра SREG встановлено

Продовження таблиці Б.5

1	2
BRCS k	Перехід, якщо є перенесення
BRCC k	Перехід, якщо немає перенесення
BREQ k	Перехід, якщо дорівнює
BRNE k	Перехід, якщо не дорівнює
BRSH k	Перехід, якщо більше або дорівнює
BRLO k	Перехід, якщо менше
BRMI k	Перехід, якщо від’ємне значення
BRPL k	Перехід, якщо додатне значення
BRGE k	Перехід, якщо більше або дорівнює (числа зі знаком)
BRLT k	Перехід, якщо менше (числа зі знаком)
BRHS k	Перехід, якщо є половинне перенесення
BRHC k	Перехід, якщо немає половинного перенесення
BRTS k	Перехід, якщо прапорець T встановлено
BRTC k	Перехід, якщо прапорець T скинуто
BRVS k	Перехід, якщо є переповнення додаткового коду
BRVC k	Перехід, якщо немає переповнення додаткового коду
BRID k	Перехід, якщо переривання заборонені
BRIE k	Перехід, якщо переривання дозволені

Таблиця Б.6 – Зведені команди керування системою

NOP	Немає операцій (пуста операція)
SLEEP	Перехід у сплячий режим
WDR	Скидання сторожового таймера

ДОДАТОК В

Розширений опис команд

Таблиця В.1 – Прийняті значення

Значення	Опис
C	Прапорець перенесення регістра SREG
Z	Прапорець нуля регістра SREG
N	Прапорець від’ємного значення регістра SREG
V	Прапорець переповнення додаткового коду регістра SREG
S	Прапорець знаку регістра SREG
H	Прапорець половинного перенесення регістра SREG
T	Прапорець користувача регістра SREG
I	Прапорець переривань регістра SREG
Rd	Регістр загального призначення (РЗП)
Rr	Регістр загального призначення (РЗП)
K	Константа
К	Адреса константи
B	Номер біта (розряду) в регістрі
S	Номер біта (розряду) в регістрі SREG
Z	Регістр-вказівник (R31:R30) адреси при непрямій адресації
A	Адреса у просторі введення-виведення
∩	Диз’юнкція
U	Кон’юнкція
PC	Лічильник команд
STACK	Рівень стеку

SP	Вказівник стеку
МЦ	Кількість машинних циклів для виконання команди
↔	Команда може змінити прапорець

Повний опис команд наведено нижче.

Команда	ADDR Rd,Rr							
Операція	Rd=Rd+Rr							
Код операції	0000 11rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Складання Rd та Rr. Результат у Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	ADDR R1,R2 ; додати R2 до R1 (R1=R1+R2)							

Команда	AND Rd,Rr							
Операція	Rd=Rd∩ Rr (Кон'юнкція)							
Код операції	0010 00rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31,							
Опис	Виконує операцію «логічне множення» між Rd та Rr. Результат у Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	LDI R16,1 ; присвоїти маску 0000 0001 в R 16 AND R2,R16 ; виділити 0-й розряд у R2							

Команда	ANDI Rd,K							
Операція	Rd=Rd∩ Rr (Кон'юнкція)							
Код операції	0111 KKKK dddd KKKK					1 слово (2 байти)		
Операнди	16≤ d ≤ 31, 0≤ K ≤ 255							
Опис	Виконує операцію «логічне множення» між Rd та K. Результат у Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	ANDI R17,0F ; обнулити старший півбайт R17							

Команда	ASR Rd							
Операція	До 1100 1101. Після **1**110 0110, **C=1**							
Код операції	1001 010d dddd 0101					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Зсування вмісту Rd вправо. Старший біт не змінюється. Молодший біт йде у прапорець C							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	↔	↔	↔	↔
МЦ	1							
Приклад	LDI R17,$FC ; R 17=-4 ASR R17 ; R17=R17/2							

Команда	BCLR s							
Операція	SREG.s=0							
Код операції	1001 0100 1sss 1000					1 слово (2 байти)		
Операнди	0≤ s ≤ 7							
Опис	Скидає в «0» вказаний (s) біт регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	↔	↔	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	BCLR 7 ; скидання прапорця переривань I							

Команда	BLD Rd,b							
Операція	Rd.b=T							
Код операції	1111 100d dddd 0bbb					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ b ≤ 7							
Опис	Копіює прапорець T у розряд b регістра Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	BLD R1,2 ; запис стану прапорця T у 2 розряд R1							

Команда	BRBC s,k							
Операція	Перехід на мітку, якщо біт s регістра SREG - «0»							
Код операції	1111 01kk kkkk ksss					1 слово (2 байти)		
Операнди	0≤ s ≤ 7, -64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє вказаний біт регістра SREG: якщо «0», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	CPI R20,5 ; порівняти R20 з числом 5 BRBC 1, Mit ; перехід, якщо не дорівнює ….. Mit:							

Команда	BRBS s,k							
Операція	Перехід на мітку, якщо біт s регістра SREG - «1»							
Код операції	1111 00kk kkkk ksss					1 слово (2 байти)		
Операнди	0≤ s ≤ 7, -64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє вказаний біт регістра SREG: якщо «1», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							

Приклад	CPI R20,5 ; порівняти R20 з числом 5 BRBS 1, Mit ; перехід, якщо дорівнює ….. Mit:

Команда	BRCC k							
Операція	Перехід на мітку, якщо прапорець C регістра SREG - «0»							
Код операції	1111 01kk kkkk k0					1 слово (2 байти)		
Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець C регістра SREG: якщо «0», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо C=1; 2, якщо C=0							
Приклад	ADD R22,R23 ; додати R23 до R22 BRCC Mit ; перехід, якщо R22 не переповнено ….. Mit:							

Команда	BRCS k							
Операція	Перехід на мітку, якщо прапорець C регістра SREG – «1»							
Код операції	1111 00kk kkkk k000					1 слово (2 байти)		
Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець C регістра SREG: якщо «1», то переходить на мітку у межі k							
Регістр	I	T	H	S	V	N	Z	C

SREG	-	-	-	-	-	-	-	-
МЦ	1, якщо C=0; 2, якщо C=1							
Приклад	ADD R22,R23 ; додати R23 до R22 BRCS Mit ; перехід, якщо R22 переповнено ….. Mit:							

Команда	BREQ k							
Операція	Перехід на мітку, якщо дорівнює (прапорець Z)							
Код операції	1111 00kk kkkk k001						1 слово (2 байти)	
Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець Z регістра SREG: якщо «1», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо Z=0; 2, якщо Z=1							
Приклад	CP R22,R23 ; порівняти R23 та R22 BREQ Mit ; перехід, якщо R23=R22 ….. Mit:							

Команда	BRGE k							
Операція	Перехід на мітку, якщо більше або дорівнює (прапорець S)							
Код операції	1111 01kk kkkk k100						1 слово (2 байти)	
Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець S регістра SREG: якщо «0», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-

МЦ	1, якщо S=1; 2, якщо S=0
Приклад	CP R22,R23 ; порівняти R23 та R22 BRGE Mit ; перехід, якщо R23≤R22 ….. Mit:

Команда	BRHC k							
Операція	Перехід на мітку, якщо не було половинного перенесення (прапорець H)							
Код операції	1111 01kk kkkk k101				1 слово (2 байти)			
Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець H регістра SREG: якщо «0», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо H=1; 2, якщо H=0							
Приклад	BRHC Mit ; перехід, якщо H=0 ….. Mit:							

Команда	BRHS k							
Операція	Перехід на мітку, якщо було половинне перенесення (прапорець H)							
Код операції	1111 00kk kkkk k101				1 слово (2 байти)			
Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець H регістра SREG: якщо «1», то переходить на мітку у межі k							
Регістр	I	T	H	S	V	N	Z	C

SREG	-	-	-	-	-	-	-	-
МЦ	1, якщо H=0; 2, якщо H=1							
Приклад	BRHS Mit ; перехід, якщо H=1 ….. Mit:							

Команда	BRID k	
Операція	Перехід на мітку, якщо було заборонено переривання (прапорець I)	
Код операції	1111 01kk kkkk k111	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець I регістра SREG: якщо «0», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо I=1; 2, якщо I=0							
Приклад	BRID Mit ; перехід, якщо I=0 ….. Mit:							

Команда	BRIE k	
Операція	Перехід на мітку, якщо було дозволено переривання (прапорець I)	
Код операції	1111 00kk kkkk k111	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець I регістра SREG: якщо «1», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо I=1; 2, якщо I=0							
Приклад	BRIE Mit ; перехід, якщо I=1 ….. Mit:							

Команда	BRLO k	
Операція	Перехід на мітку, якщо менше, для незнакових чисел (прапорець C)	
Код операції	1111 00kk kkkk k000	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець C регістра SREG: якщо «1», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо C=0; 2, якщо C=1							
Приклад	CPI R19,$10 ; порівняти R19 та число 10 BRIE Mit ; перехід, якщо R19 менше 10 ….. Mit:							

Команда	BRLT k	
Операція	Перехід на мітку, якщо менше, для знакових чисел (прапорець S)	
Код операції	1111 00kk kkkk k100	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє	

	прапорець S регістра SREG: якщо «1», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо S=0; 2, якщо S=1							
Приклад	CP R19,R20 ; порівняти R19 та R20 BRLT Mit ; перехід, якщо R19 менше R20 ….. Mit:							

Команда	BRMI k	
Операція	Перехід на мітку, якщо від’ємне значення (прапорець N)	
Код операції	1111 00kk kkkk k100	1 слово (2 байти)
Операнди	-64 ≤k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець N регістра SREG: якщо «1», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо N=0; 2, якщо N=1							
Приклад	SUBI R18,4 ; R18=R18-4 BRMI Mit ; перехід, якщо R19 <0 ….. Mit:							

Команда	BRNE k	
Операція	Перехід на мітку, якщо не дорівнює (прапорець Z)	
Код операції	1111 01kk kkkk k001	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	

Опис	Умовний відносний перехід. Перевіряє прапорець Z регістра SREG: якщо «0», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо Z=1; 2, якщо Z=0							
Приклад	CPI R27,4 ; порівняти R27 та число 4 BRNE Mit ; перехід, якщо R19 не дорівнює 4 ….. Mit:							

Команда	BRPL k	
Операція	Перехід на мітку, якщо додатне значення (прапорець N)	
Код операції	1111 01kk kkkk k010	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець N регістра SREG: якщо «0», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-

МЦ	1, якщо N=1; 2, якщо N=0
Приклад	SUBI R18,4 ; R18=R18-4 BRPL Mit ; перехід, якщо R19 >0 ….. Mit:

Команда	BRSH k	
Операція	Перехід на мітку, якщо більше або дорівнює (прапорець C)	
Код операції	1111 01kk kkkk k000	1 слово (2 байти)

Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець C регістра SREG: якщо «0», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо C=1; 2, якщо C=0							
Приклад	SUBI R18,4 ; R18=R18-4 BRSH Mit ; перехід, якщо R18 ≥4 ….. Mit:							

Команда	BRTC k	
Операція	Перехід на мітку, якщо T=0 (прапорець T)	
Код операції	1111 01kk kkkk k110	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець T регістра SREG: якщо «0», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо T=1; 2, якщо T=0							
Приклад	BST R27,4 ; записати 4-й розряд R27 у прапорці T BRTC Mit ; перехід, якщо T=0 ….. Mit:							

Команда	BRTS k	
Операція	Перехід на мітку, якщо T=1 (прапорець T)	
Код операції	1111 00kk kkkk k110	1 слово (2 байти)

Операнди	-64≤ k ≤ +63							
Опис	Умовний відносний перехід. Перевіряє прапорець T регістра SREG: якщо «1», то переходить на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо T=0; 2, якщо T=1							
Приклад	BST R27,4 ; записати 4-й розряд R27 у прапорці T BRTS Mit ; перехід, якщо T=1 ….. Mit:							

Команда	BRVC k	
Операція	Перехід на мітку, якщо немає переповнення додаткового коду (прапорець V)	
Код операції	1111 01kk kkkk k011	1 слово (2 байти)
Операнди	-64≤ k ≤ +63	
Опис	Умовний відносний перехід. Перевіряє прапорець V регістра SREG: якщо «0», то переходить на мітку у межі k	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо V=1; 2, якщо V=0							
Приклад	ADD R20,R21 ; R20=R20+R21 BRVC 1, Mit ; перехід, якщо V=0 ….. Mit:							

Команда	BRVS k
Операція	Перехід на мітку, якщо є переповнення додаткового коду (прапорець V)

Код операції	1111 00kk kkkk k011						1 слово (2 байти)
Операнди	-64≤ k ≤ +63						
Опис	Умовний відносний перехід. Перевіряє прапорець V регістра SREG: якщо «1», то переходить на мітку у межі k						
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо V=0; 2, якщо V=1						
Приклад	ADD R20,R21 ; R20=R20+R21 BRVS 1, Mit ; перехід, якщо V=1 ….. Mit:						

Команда	BSET s							
Операція	Встановлення розряду регістра SREG							
Код операції	1001 0100 0sss 1000						1 слово (2 байти)	
Операнди	0≤ s ≤ 7							
Опис	Встановлює розряд s регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	↔	↔	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	BSET 6 ; встановити прапорець T							

Команда	BST Rd,b							
Операція	T=Rd,b							
Код операції	1111 101d dddd 0bbb						1 слово (2 байти)	
Операнди	0≤ d ≤ 31, 0≤ b ≤ 7							
Опис	Запис розряду РЗП у прапорець T регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	↔	-	-	-	-	-	-

МЦ	1
Приклад	BST R6,6 ; запис 6 розряду регістра R6 у T

Команда	CBI A,b							
Операція	Скинути розряд PBB							
Код операції	1001 1000 AAAA Abbb					1 слово (2 байти)		
Операнди	0≤ A ≤ 31, 0≤ b ≤ 7							
Опис	Скидає розряд b регістра PBB (адреса PBB від 0 до 31)							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	2							
Приклад	CBI PORTB,4 ; скинути 4 розряд регістра PORTB							
Команда	CBR Rd,K							
Операція	Rd=Rd∩($FF-K)							
Код операції	0111 $\overline{KKKK}$ dddd $\overline{KKKK}$					1 слово (2 байти)		
Операнди	16≤ d ≤ 31, 0≤ K ≤ 255							
Опис	Скидає розряди РЗП (R16…R31) у відповідності до маски K через диз'юнкцію							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	CBR R16,$F0 ; обнулити старший півбайт R16							

Команда	CLC							
Операція	C=0							
Код операції	1001 0100 1000 1000					1 слово (2 байти)		
Операнди	-							
Опис	Скидає прапорець C регістра SREG							
Регістр	I	T	H	S	V	N	Z	C

SREG	-	-	-	-	-	-	-	0
МЦ	1							
Приклад	CLC ; C=0							

Команда	CLH							
Операція	H=0							
Код операції	1001 0100 1101 1000						1 слово (2 байти)	
Операнди	-							
Опис	Скидає прапорець H регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	0	-	-	-	-	-
МЦ	1							
Приклад	CLH ; H=0							
Команда	CLI							
Операція	I=0							
Код операції	1001 0100 1111 1000						1 слово (2 байти)	
Операнди	-							
Опис	Скидає прапорець I регістра SREG (заборона переривань)							
Регістр SREG	I	T	H	S	V	N	Z	C
	0	-	-	-	-	-	-	-
МЦ	1							
Приклад	CLI ; I=0							

Команда	CLN	
Операція	N=0	
Код операції	1001 0100 1010 1000	1 слово (2 байти)
Операнди	-	
Опис	Скидає прапорець N регістра SREG	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	0	-	-
МЦ	1							
Приклад	CLN ; N=0							

Команда	CLR Rd	
Операція	Rd=0	
Код операції	0010 01dd dddd dddd	1 слово (2 байти)
Операнди	0≤ d ≤ 31	
Опис	Скидає РЗП в нуль	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	0	0	0	1	-
МЦ	1							
Приклад	CLR R12 ; R12=0000 0000							

Команда	CLS	
Операція	S=0	
Код операції	1001 0100 1100 1000	1 слово (2 байти)
Операнди	-	
Опис	Скидає прапорець S регістра SREG	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	0	-	-	-	-
МЦ	1							
Приклад	CLS ; S=0							

Команда	CLT	
Операція	T=0	
Код операції	1001 0100 1110 1000	1 слово (2 байти)
Операнди	-	
Опис	Скидає прапорець T регістра SREG	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	0	-	-	-	-	-	-
МЦ	1							
Приклад	CLT ; T=0							

Команда	CLV	
Операція	V=0	
Код операції	1001 0100 1011 1000	1 слово (2 байти)
Операнди	-	
Опис	Скидає прапорець V регістра SREG	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	0	-	-	-
МЦ	1							
Приклад	CLV ; V=0							

Команда	CLZ	
Операція	Z=0	
Код операції	1001 0100 1001 1000	1 слово (2 байти)
Операнди	-	
Опис	Скидає прапорець Z регістра SREG	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	0	-
МЦ	1							
Приклад	CLZ ; Z=0							

Команда	COM Rd	
Операція	Rd=$FF-Rd	
Код операції	1001 010d dddd dddd	1 слово (2 байти)
Операнди	0≤ d ≤ 31	

Опис	Розраховує непрямий код числа РЗП (R0… R31) та встановлює його у РЗП							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	1
МЦ	1							
Приклад	COM R16 ; R16=$0F, то після R16=$F0							

Команда	CP Rd,Rr							
Операція	Rd-Rr							
Код операції	0001 01rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Порівнює два РЗП (R0…R31) та встановлює або скидає прапорці регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	CP R1,R2 ; порівняти R1 та R2 BRNE Mit ; перехід на мітку, якщо R1< >R2 Mit:							

Команда	CPC Rd,Rr
Операція	Rd-Rr-C

Код операції	0000 01rd dddd rrrr						1 слово (2 байти)
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31						
Опис	Порівнює два РЗП (R0…R31) та встановлює або скидає прапорці регістра SREG з урахуванням прапорця C регістра SREG						
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	CPC R1,R2 ; порівняти R1 та R2 BRNE Mit ; перехід не мітку, якщо R1< >R2-C … Mit:							

Команда	CPI Rd,K						
Операція	Rd-K						
Код операції	0011 KKKK dddd KKKK						1 слово (2 байти)
Операнди	0≤ d ≤ 31, 0≤ K ≤ 255						
Опис	Порівнює РЗП (R0…R31) та константу						
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	CPI R1,$4 ; порівняти R1 та число 4 BRNE Mit ; перехід не мітку, якщо R1< >4 … Mit:							

Команда	CPSE Rd,Rr

Операція	Якщо Rd=Rr, то PC=PC+2, інакше PC=PC+1							
Код операції	0001 00rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Порівнює два РЗП (R0…R31) та пропускає наступну команду, якщо значення РЗП однакові							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо R1< >R2; 2, якщо R1=R2;							
Приклад	INC R1 ; R1=R1+1 CPSE R1,R2 ; пропускання INC R2, якщо R1=R2 INC R2							

Команда	DEC Rd							
Операція	Rd=Rd-1							
Код операції	1001 010d dddd 1010					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Зменшує на одиницю РЗП (R0…R31)							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	↔	↔	↔	-
МЦ	1							
Приклад	DEC R1 ; R1=R1-1							

Команда	EOR Rd,Rr	
Операція	Виключна кон’юнкція (U) РЗП	
Код операції	0010 01rd dddd rrrr	1 слово (2 байти)

Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Проводить виключну кон’юнкцію двох РЗП Результат у Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	EOR R1,R1 ; R1=0							

Команда	IN Rd,A							
Операція	Rd=A							
Код операції	1011 0AAd dddd AAAA						1 слово (2 байти)	
Операнди	0≤ d ≤ 31, 0≤ A ≤ 63							
Опис	Пересилання вмісту РВВ А в РЗП							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	IN R1,PORTB ; занести вміст PORTB в R1							
Команда	INC Rd							
Операція	Rd=Rd+1							
Код операції	1001 010d dddd 0011						1 слово (2 байти)	
Операнди	0≤ d ≤ 31							
Опис	Збільшення на одиницю РЗП (R0…R31)							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	↔	↔	↔	-
МЦ	1							
Приклад	INC R1 ; R1=R1+1							

Команда	LD Rd,Z	
Операція	Rd=[Z]	
Код	1000 000d dddd 0000	1 слово (2 байти)

операції								
Операнди	0≤ d ≤ 31							
Опис	Пересилання вмісту байта, що знаходиться за адресою, яка знаходиться в індексному регістрі Z (R31:R30) у РЗП							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	2							
Приклад	LD R1,Z ; занести вміст з адреси Z в R1							

Команда	LDI Rd,K							
Операція	Rd=K							
Код операції	1110 KKKK dddd KKKK					1 слово (2 байти)		
Операнди	16≤ d ≤ 31, 0≤ K ≤ 255							
Опис	Записування константи в РЗП							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	LDI R16,$0F ; R16=0000 1111							
Команда	LPM							
Операція	R0=[Z]							
Код операції	1001 0101 1100 1000					1 слово (2 байти)		
Операнди	-							
Опис	Пересилання в R0 вмісту байта, що знаходиться за адресою, яка знаходиться в індексному регістрі Z (R31:R30)							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	3							
Приклад	LPM R1,Z ; занести вміст з адреси Z в R0							

Команда	LSL Rd

Операція	До Rd=1100 1101. Після C=1, Rd=1001 1010							
Код операції	1001 010d dddd 0110					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Зсування вмісту розрядів РЗП Rd вліво. У молод-ший біт йде «0». Старший біт йде у прапорець C							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	↔	0	↔	↔
МЦ	1							
Приклад	LDI R17,$4 ; R 17=4 LSL R17 ; R17=R17*2=8							

Команда	LSR Rd							
Операція	До Rd=1100 1101. Після Rd=0110 0110 C=1							
Код операції	1001 010d dddd 0110					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Зсування вмісту розрядів РЗП Rd вправо. У стар-ший біт йде «0». Молодший біт йде у прапорець C							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	↔	0	↔	↔
МЦ	1							
Приклад	LDI R17,$4 ; R 17=4 LSR R17 ; R17=R17/2=2							

Команда	MOV Rd,K

Операція	Rd=Rr							
Код операції	0010 11rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Копірування Rr в Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	MOV R16,R17 ; R16=R17							

Команда	NEG Rd							
Операція	Rd=$00-Rd							
Код операції	1001 010d dddd 0001					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Розрахування додаткового коду числа							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	NEG R1							
Команда	NOP							
Операція	Немає							
Код операції	0000 0000 0000 0000					1 слово (2 байти)		
Операнди	-							
Опис	Розрахування додаткового коду числа							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	NOP ; один машинний цикл							

Команда	OR Rd,Rr	
Операція	Кон’юнкція (U) РЗП	
Код	0010 10rd dddd rrrr	1 слово (2 байти)

операції								
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Кон'юнкція двох РЗП, результат у Rd R1=0000 0001, R2=0000 0010, R1=0000 0011							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	OR R1,R2 ; R1 U R2							

Команда	ORI Rd,K	
Операція	Кон'юнкція (U) РЗП з константою	
Код операції	0110 КККК dddd КККК	1 слово (2 байти)
Операнди	16≤d ≤31, 0≤К ≤255	
Опис	Кон'юнкція РЗП з константою, результат у Rd (R1=0000 0001, К=0000 0010, R1=0000 0011)	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	OR R1,$02 ; R1 U $02							

Команда	OUT A,Rd	
Операція	A=Rd	
Код операції	1011 1AAd dddd AAAA	1 слово (2 байти)
Операнди	0≤ d ≤ 31, 0≤ A ≤ 63	
Опис	Пересилання вмісту РЗП в РВВ А	

Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	OUT PORTB, R1 ; занести вміст PORTB в R1							

Команда	RCALL k

Операція	STACK=PC+1; PC=PC+k+1; SP=SP-2							
Код операції	1101 kkkk kkkk kkkk					1 слово (2 байти)		
Операнди	-2047≤ k ≤ 2047							
Опис	Відносний виклик підпрограми. Перехід на мітку у межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	3							
Приклад	RCALL mit ; виклик підпрограми … Mit: …. RET ; повернення з підпрограми							

Команда	RET							
Операція	PC= STACK; SP=SP+2							
Код операції	1001 0101 0000 1000					1 слово (2 байти)		
Операнди	-							
Опис	Повернення з підпрограми							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	4							
Приклад	RCALL mit ; виклик підпрограми … Mit: ….							

	RET ; повернення з підпрограми

Команда	RETI							
Операція	PC= STACK; SP=SP+2							
Код операції	1001 0101 0001 1000					1 слово (2 байти)		
Операнди	-							
Опис	Повернення з підпрограми переривання							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	4							
Приклад	RCALL mit ; виклик підпрограми переривання … Mit: …. RET ; повернення з підпрограми переривання							

Команда	RJMP k							
Операція	PC=PC+k+1							
Код операції	1100 kkkk kkkk kkkk					1 слово (2 байти)		
Операнди	-2047≤ k ≤ 2047							
Опис	Відносний безумовний перехід на мітку в межі k							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	2							
Приклад	RJMP Mit ; виклик підпрограми … Mit:							

Команда	ROL Rd							
Операція	До C=**0**, Rd=**1**100 1101. Після Rd=0100 110**0**, C=**1**							
Код операції	0001 110d dddd 0101					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Зсування вмісту Rd вліво. Старший біт іде у прапорець C. Прапорець C йде в молодший біт							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	ROL R17,$FC							

Команда	SBC Rd,Rr							
Операція	Rd=Rd-Rr-C							
Код операції	0000 10rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ r ≤ 31							
Опис	Віднімання з РЗП іншого РЗП з урахуванням прапорця C регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	SBC R16,R17							

Команда	SBCI Rd,K							
Операція	Rd=Rd-K-C							
Код операції	0100 KKKK dddd KKKK				1 слово (2 байти)			
Операнди	16≤ d ≤ 31, 0≤ K ≤ 255							
Опис	Віднімання з РЗП константи з урахуванням прапорця C регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	SBCI R16,$0F							

Команда	SBI A,b							
Операція	Встановити розряд РВВ							
Код операції	1001 1010 AAAA Abbb				1 слово (2 байти)			
Операнди	0≤ A ≤ 31, 0≤ b ≤ 7							
Опис	Встановлення розряду b регістра РВВ (адреса РВВ від 0 до 31)							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	2							
Приклад	SBI PORTB,4 ; скинути 4 розряд регістра PORTB							
Команда	SBIC A,b							
Операція	Якщо РВВ A,b=0, то PC=PC+2, інакше PC=PC+1							
Код операції	1001 1001 AAAA Abbb				1 слово (2 байти)			
Операнди	0≤ A ≤ 31, 0≤ b ≤ 7							
Опис	Пропускання наступної команди, якщо розряд РВВ «0»							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо A,b=1; 2, якщо A,b=0							
Приклад	SBIC PORTB,4							

Команда	SBIS A,b							
Операція	Якщо PBB A,b=1, то PC=PC+2, інакше PC=PC+1							
Код операції	1001 1011 AAAA Abbb				1 слово (2 байти)			
Операнди	0≤ A ≤ 31, 0≤ b ≤ 7							
Опис	Пропускання наступної команди, якщо розряд PBB «1»							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо A,b=0; 2, якщо A,b=1							
Приклад	SBIS PORTB,4							

Команда	SBR Rd,K							
Операція	Rd=Rd U K							
Код операції	0110 KKKK dddd KKKK				1 слово (2 байти)			
Операнди	16≤ d ≤ 31, 0≤ K ≤ 255							
Опис	Встановлення розрядів РЗП (R16…R31) у відповідності до маски К через кон’юнкцію							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	↔	0	↔	↔	-
МЦ	1							
Приклад	SBR R16,$F0 ; всі одиниці у старшому полубайті R16							

Команда	SBRC Rd,b							
Операція	Якщо РЗП Rd=0, то PC=PC+2, інакше PC=PC+1							
Код операції	1111 110d dddd 0bbb					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ b ≤ 7							
Опис	Пропускання наступної команди, якщо розряд РЗП «0»							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо Rd,b=1; 2, якщо Rd,b=0							
Приклад	SBRC R16,4							

Команда	SBRS Rd,b							
Операція	Якщо РЗП Rd=1, то PC=PC+2, інакше PC=PC+1							
Код операції	1111 111d dddd 0bbb					1 слово (2 байти)		
Операнди	0≤ d ≤ 31, 0≤ b ≤ 7							
Опис	Пропускання наступної команди, якщо розряд РЗП «1»							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1, якщо Rd,b=0; 2, якщо Rd,b=1							
Приклад	SBRS R16,4							
Команда	SEC							
Операція	C=1							
Код операції	1001 0100 0000 1000					1 слово (2 байти)		
Операнди	-							
Опис	Встановлення прапорця C регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	1
МЦ	1							
Приклад	SEC ; C=0							

Команда	SEH							
Операція	H=1							
Код операції	1001 0100 0101 1000						1 слово (2 байти)	
Операнди	-							
Опис	Встановлення прапорця H регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	1	-	-	-	-	-
МЦ	1							
Приклад	SEH ; H=1							

Команда	SEI							
Операція	I=1							
Код операції	1001 0100 0111 1000						1 слово (2 байти)	
Операнди	-							
Опис	Встановлення прапорця I регістра SREG (заборона переривань)							
Регістр SREG	I	T	H	S	V	N	Z	C
	1	-	-	-	-	-	-	-
МЦ	1							
Приклад	SEI ; I=1							
Команда	SEN							
Операція	N=1							
Код операції	1001 0100 0010 1000						1 слово (2 байти)	
Операнди	-							
Опис	Встановлення прапорця N регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	1	-	-
МЦ	1							
Приклад	SEN ; N=1							

Команда	SER Rd							
Операція	Rd=$FF (1111 1111)							
Код операції	1110 1111 dddd 1111					1 слово (2 байти)		
Операнди	16≤ d ≤ 31							
Опис	Встановлення РЗП в одиниці							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	SER R12 ; R12=1111 1111							

Команда	SES							
Операція	S=1							
Код операції	1001 0100 0100 1000					1 слово (2 байти)		
Операнди	-							
Опис	Встановлення прапорця S регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	1	-	-	-	-
МЦ	1							
Приклад	SES ; S=1							

Команда	SET							
Операція	T=1							
Код операції	1001 0100 0110 1000					1 слово (2 байти)		
Операнди	-							
Опис	Встановлення прапорця T регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	1	-	-	-	-	-	-
МЦ	1							
Приклад	SET ; T=1							

Команда	SEV							
Операція	V=1							
Код операції	1001 0100 0011 1000					1 слово (2 байти)		
Операнди	-							
Опис	Встановлення прапорця V регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	1	-	-	-
МЦ	1							
Приклад	SEV ; V=1							

Команда	SEZ							
Операція	Z=1							
Код операції	1001 0100 0001 1000					1 слово (2 байти)		
Операнди	-							
Опис	Встановлення прапорця Z регістра SREG							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	1	-
МЦ	1							
Приклад	SEZ ; Z=1							

Команда	SLEEP							
Операція	Перехід у режим зниженого енергоспоживання							
Код операції	1001 0101 1000 1000					1 слово (2 байти)		
Операнди	-							
Опис	Дивись розділ «режим зниженого енергоспоживання»							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	1	-
МЦ	1							

Приклад	SLEEP

Команда	ST Z,Rd							
Операція	[Z]=Rd							
Код операції	1000 001d dddd 0000					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Пересилання вмісту РЗП за адресою, яка знаходиться в індексному регістрі Z (R31:R30)							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	2							
Приклад	ST Z,R1 ; занести вміст R1 за адресою Z							

Команда	SUB Rd,Rr							
Операція	Rd=Rd-Rr							
Код операції	0001 10rd dddd rrrr					1 слово (2 байти)		
Операнди	0≤ d≤ 31, 0≤ r≤ 31							
Опис	Віднімання Rr від Rd. Результат у Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	SUB R1,R2 ; R1=R1-R2							
Команда	SUBI Rd,K							
Операція	Rd=Rd-K							
Код операції	1010 KKKK dddd KKKK					1 слово (2 байти)		
Операнди	16≤ d≤ 31, 0≤ K ≤ 255							
Опис	Віднімання константи від Rd. Результат у Rd							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	↔	↔	↔	↔	↔	↔
МЦ	1							
Приклад	SUBI R21,$05 ; R21=R21-5							

Команда	SWAP Rd							
Операція	Rd(7..4)=Rd(3..0), Rd(3..0)=Rd(7..4)							
Код операції	1001 010d dddd 0010					1 слово (2 байти)		
Операнди	0≤ d ≤ 31							
Опис	Старший півбайт стає молодшим і навпаки							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	SWAP R21 ; R21=1111 0000 стає R21=0000 1111							

Команда	WDR							
Операція	Скидання сторожового таймера							
Код операції	1001 0101 1010 1000					1 слово (2 байти)		
Операнди	-							
Опис	Скидання сторожового таймера, не допускаючи скидання мікроконтролера							
Регістр SREG	I	T	H	S	V	N	Z	C
	-	-	-	-	-	-	-	-
МЦ	1							
Приклад	WDR							

ДОДАТОК Г

Директиви Assembler

Таблиця Г.1

Директива	Опис
.BYTE	Зарезервувати байт в ПЗП
.CSEG	Сегмент программ
.DB	Встановити байт у флеш або EEPROM
.DEF	Призначити регістру ім’я

.DEVICE	Встановити пристрій, для якого компілюється программа
.DSEG	Сегмент даних
.DW	Встановити слово у флеш або EEPROM
.ENDM	Кінець макросу
.EQU	Встановити константу
.ESEG	Сегмент EEPROM
.EXIT	Вихід з файлу
.INCLUDE	Вложити інший файл
.LIST	Підключити генерацію лістингу
.LISTMAC	Підключити генерацію лістингу в макросі
.MACRO	Початок макросу
.NOLIST	Вимкнути генерацію лістингу
.ORG	Встановити адресу положення у сегменті
.SET	Встановити змінний символічний еквівалент виразу