ЗМІСТ

ВСТУП

Лабораторні роботи з дисциплін «Мікропроцесорна техніка», «Обчислювальна техніка та мікропроцесори» призначені для закріплення студентами знань, одержаних на лекційних і практичних заняттях. Метою цих методичних вказівок є отримання первинних навичок з програмування мікропроцесорних систем на рівні мікропроцесора (мовою Асемблер), а також ознайомлення з апаратними і програмними засобами налагодження таких систем і відповідного програмного забезпечення. Студент, який приступає до вивчення викладеного в методичних вказівках матеріалу, повинен володіти основами інформатики і цифрових електронних схем.

До виконання лабораторних робіт допускаються студенти, які пройшли інструктаж з техніки безпеки і успішно пройшли контрольне опитування. Звіт з лабораторної роботи складається кожним студентом окремо. Захист виконаної роботи відбувається під час наступного заняття. Під час перебування в лабораторії студенти повинні суворо дотримуватися вимог техніки безпеки щодо роботи з комп'ютерною технікою. Інструктаж з техніки безпеки проводить викладач на початку циклу лабораторних занять, про що кожен студент і викладач засвідчують у лабораторному журналі.

У кінці кожної роботи наведено контрольні питання та завдання, відповіді на які дозволяють визначити ступінь готовності студентів до виконання лабораторної роботи і рівень отриманих знань.

ЛАБОРАТОРНА РОБОТА 1
Моделювання мікропроцесорної системи в середовищі PROTEUS

1.1 Навчальні питання

1 Дослідження основних функцій і компонентів середовища PROTEUS.

2 Дослідження методів складання схем у середовищі PROTEUS.

1.2 Навчальна мета

Практичне ознайомлення з основними функціями та органами керування програми PROTEUS.

1.3 Теоретичні відомості

Емулятор електронних пристроїв PROTEUS підтримує мікроконтролери AVR, 8051, PIC10, PIC16, PIC18, ARM7, Motorola, HC11.

Для ознайомлення з основними функціями та органами керування програми розглянемо приклад Емуляції пристрою з МК AVR AT90S8515 та інтерфейсом 1-Wire, по якому підключені «таблетка» i-button DS1990 (для домофонів) та інші 1-Wire прилади.

Корисна порада на майбутнє – не створюйте проект із нуля, відкривайте будь-який простий приклад з папки SAMPLES і модифікуйте його під свої схему та потреби.

1.4 Порядок виконання роботи

1 Запустіть **PROTEUS** і зайдіть у меню **File -> Load Design.** Після цього перемістіться на один рівень вгору – у папку, де встановлений **PROTEUS**, і відкрийте подвійним натисканням папку **SAMPLES**.

Знайдіть у ній і відкрийте папку **VSM for AVR**, потім папку **One-Wire,** потім папку **NETWORK** і натисніть на файл проекту

1WIRE_NET.DSN – він виділиться, тепер натисніть «Відкрити». Проект відкриється. Розгорніть вікно **PROTEUS** на весь екран, натиснувши на квадратик у правому верхньому кутку головного вікна програми. **Результат попередніх дій зображено на рисунку 1.1.**

Інтерфейс PROTEUS не має лінійок прокручування. Це зроблено для максимізації робочої області.

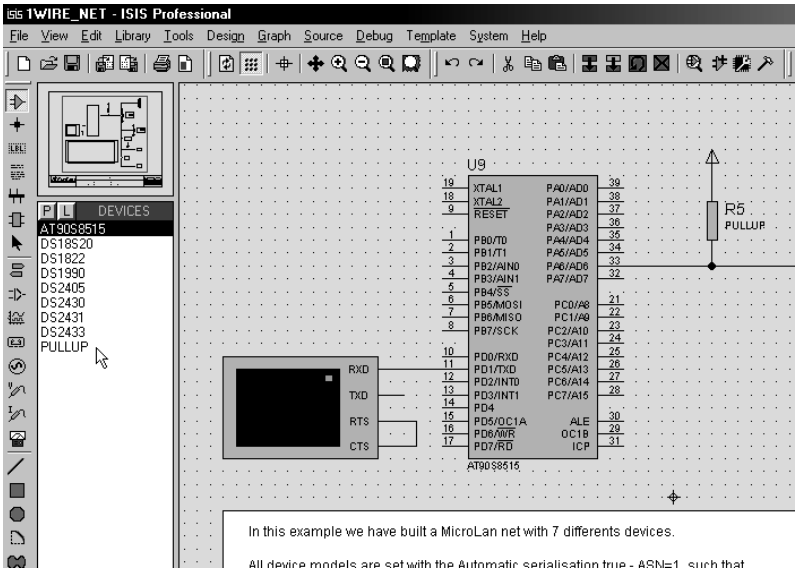

Рисунок 1.1 – Головне вікно програми

Масштаб зображення можна змінювати за допомогою колеса на мишці або за допомогою інструментів «лупа +» та «лупа –» у верхній панелі інструментів.

Переміщати зображення можна, вказуючи новий центр зображення інструментом «Хрест» ліворуч від «лупа +» та зміною масштабу.

Весь робочий аркуш можна побачити, кликнувши кнопку праворуч від «лупа –».

Показати певну область схеми можна, виділивши її за допомогою інструмента виділення області інструментом, що знаходиться ще правіше.

У лівій верхній області екрана ви бачите міні-макет сторінки, а трохи нижче панель **DEVICES** (компоненти проекту), що зображена на рисунку 1.2.

У цьому вікні відображуються **всі** елементи, які використовуються у схемі.

Кнопка з буквою **P** відкриває форму пошуку компонента в бібліотеках PROTEUS для розміщення на схемі.

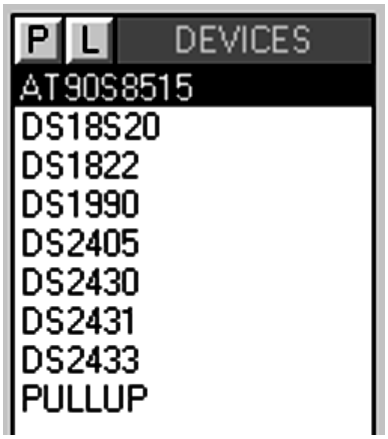

Рисунок 1.2 – Панель компонентів проекту

Кнопка з буквою «**L**» відкриває менеджер бібліотек – за його допомогою ви можете підключати нові бібліотеки компонентів.

У відкритому вікні проекту знаходиться мікроконтролер AVR **AT90s8515.** Тут також розміщені декілька компонентів компанії DALLAS-MAXIM – всі вони підключені за схемою однопровідного інтерфейсу 1-Wire.

Оскільки в даній програмі сигнал неможливо передати по одному провіднику, в цих компонентів повинні бути з'єднані і виводи GND.

У кінці списку знаходиться компонент PULLUP – це резистор, який підтягує напругу в точці, куди він підключений до + живлення МК і приладів DSxxxx – звичайно це +5 вольт.

PROTEUS – це інтерактивний довідник з електронних компонентів! Тут можна дізнатися, які компоненти розміщені на схемі та призначення їхніх виводів!

2 Натисніть кнопку з буквою **P –** відкриється меню пошуку й вибору компонентів **Pick Devices**.

У полі **Keywords** (ключові слова) уведіть **ds2** і виберіть, натиснувши мишку, верхній з чотирьох знайдених компонентів - DS2405. У полі **Description** (опис) бачимо «адресований перемикач». До цього приладу можна звернутися за його адресою і «наказати» йому видати на вихід PIO логічну 1 або 0 або перевести вихід у високоімпедансний Z-стан – вивід з дуже

великим опором, практично не проводить струму, тобто не впливає на те, що до нього підключено.

У правій частині форми можна побачити назву моделі компонента, його зображення на схемі (рисунок 1.3), а нижче – його **FootPrint** – це вигляд компонента на друкованій платі. Ще нижче – назва корпусу компонента - TO92 – це маленький трививідний пластиковий корпус у вигляді циліндра 5х5 мм.

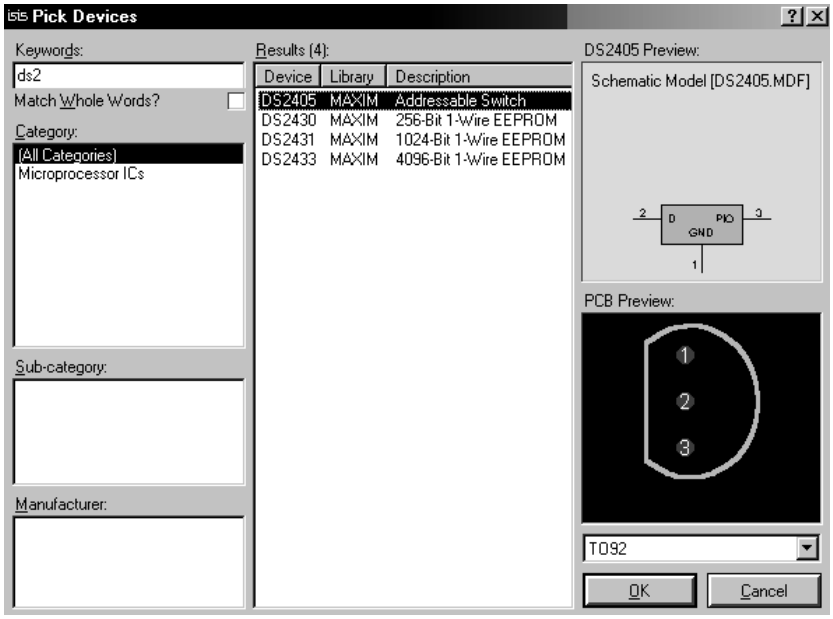

Рисунок 1.3 – Вікно пошуку компонентів

Інша частина 1-Wire приладів на схемі починається з **ds1** – введіть ці символи в поле ключових слів.

З Тепер знайдено 8 приладів. Причому вони розташовані у двох категоріях.

Виберіть мишкою DS18s20 – «Високоточний 1-Wire цифровий термометр». На схемному зображенні цього компонента (рисунок 1.4) видно деяке поле, що нагадує дисплей – у ньому будуть виводитися дані в процесі емуляції.

Помістити компонент на схему можна, натиснувши «ОК», у результаті чого компонент з’явиться у вікні DEVICES. Виділений компонент можна помістити на аркуші, розмістивши мишку в потрібне місце на аркуші схеми і натиснувши ліву кнопку. Компонент розташується на схемі (рисунок 1.5).

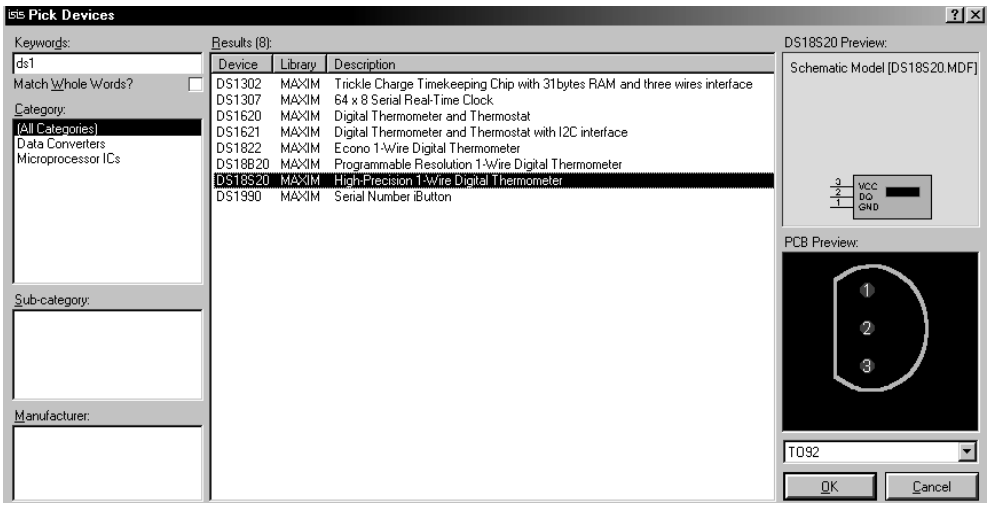

Рисунок 1.4 – Вікно з обраним компонентом

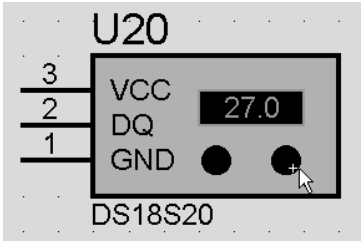

Рисунок 1.5 – Розміщення компонента на схемі

Увага! Компонент інтерактивний. Червоними стрілками ви зможете під час емуляції змінювати температуру корпусу датчика, тобто ту, яку він вимірює, натискаючи по них мишкою.

Температура відображується на дисплеї компонента. VCC – це + живлення датчика DS18s20, GND – «загальний провідник» або «земля», DQ – лінія даних.

Компонент, що присутній на схемі, можна додати і так: виділити, скопіювати і вставити, тільки в цьому випадку доведеться вручну призначити йому порядковий номер замість знаку питання – ?.

Виділити компонент або будь-який елемент схеми можна, натиснувши на нього правою кнопкою миші 2 рази. При першому натисканні він стане червоним і одночасно будуть виділені всі провідники, що підходять до компонента, – вони стануть червоними до першого з'єднання з будь-яким іншим компонентом.

Скасувати виділення ВСІХ виділених компонентів можна, натиснувши **праву** кнопку миші в порожньому місці схеми.

Видалити компонент або будь-який елемент зі схеми можна двома натисканнями правою кнопкою миші по ньому або натисканням **Del** після виділення лівою кнопкою миші.

4 Ознайомтесь: панель інструментів ліворуч – верхня частина:

*****відкрити панель DEVICES – компоненти проекту і пошук нових;

поставити точку з'єднання провідників вручну;

призначити назву провіднику – **однойменні провідники електрично З'ЄДНАНІ*!;**

додати текст у довільне місце схеми;

прокласти шину – на схемі жирна темно-синя лінія;

створити підсхему, тобто якийсь блок, що містить у собі свою схему і з'єднання;

натисканням на компонент відразу відкривається редагування його властивостей.

* – дуже корисно! Дозволяє не перетворювати схему в павутину з провідників, яку неможливо прочитати - використовуйте!

Середня частина – це інструментарій моделювання роботи електронного пристрою, тобто головне в PROTEUS!

***TERMINALS – виводи живлення, значки подачі сигналу, виведення даних, земля, з'єднання між блоками, просто виводи;**

додати вивід до створюваного компонента;

графічне відображення, збереження й **ПОТУЖНИЙ аналіз** результатів емуляції;

«магнітофон» для запису у файл і відтворення даних;

генератори будь-яких напруг, струмів, виведення їх з файлів даних;

вказати точку вимірювання напруги на провіднику;

вказати точку вимірювання струму на провіднику;

Virtual Instruments – вимірювальні прилади;

прокладання провідників на схемі.

Панель інструментів вгорі – Toggle wire autorouter ( ) – автоз'єднання провідником.

Панель керування емуляцією

1

1 – «Пуск» – запуск емуляції або продовження припиненої емуляції.

2 – «Крок» – виконати мінімальний крок за програмою МК, звичайно це одна інструкція мовою Асемблер. **Цією кнопкою теж можна почати емуляцію.**

3 – «Пауза» – пауза емуляції. Можна продовжити кнопками «Пуск» або «Крок».

4 – «Стоп» – зупинка емуляції. Після цього емуляція почнеться спочатку кнопками «Пуск» або «Крок».

Панель інструментів ліворуч, нижня частина. Ці кнопки дозволяють змінювати зображення **виділеного** компонента на схемі.

повернути на 90 градусів за годинниковою стрілкою;

повернути на 90 градусів проти годинникової стрілки;

повернути на довільний кут;

відобразити горизонтально;

відобразити вертикально.

5 Просимулюйте приклади з папки SAMPLES - \Simulation.

6 Відкрийте проект – 1WIRE_NET.DSN.

Виділіть мікроконтролер, натиснувши на нього правою кнопкою миші. МК і підключені до нього провідники стануть червоними.

Відкрийте панель редагування властивостей компонента (**Edit Component**), натиснувши на виділений МК **лівою** кнопкою миші або натиснути правою кнопкою миші, і виберіть пункт меню **Edit Properties,** у результаті чого відкриється також вікно (**Edit Component**). Натисніть на кнопку **Hidden Pins** (сховані виводи) – відкриється додаткове меню, у якому показано, як називаються вузли (провідники) схеми, до яких підключені живлення МК – **VCC** і його загальний провідник – **GND**. **Зміна цих назв може знадобитися вам при живленні МК або інших компонентів різними напругами або від різних джерел (рисунок 1.6).**

Натисніть «ОК», щоб закрити панель схованих виводів, і подивіться уважно на вміст панелі редагування компонента.

Головне для МК – це програма, за якою він буде працювати.

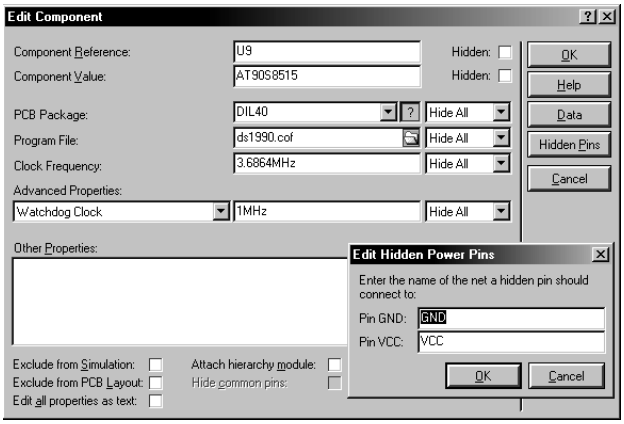

Рисунок 1.6 – Редагування властивостей компонента

У полі **Program File** потрібно вказати:

***.cof** – файл, якщо ви хочете проводити моделювання за програмою, текст якої написано мовою С;

***.hex** – файл прошивання, якщо у вас нема тексту програми ні мовою С, ні мовою Асемблер;

***.asm** – файл, якщо ви хочете проводити моделювання за програмою, текст якої написано мовою Асемблер.

Якщо у вас є текст програми ***.asm**, вам потрібно вказати назву ***.hex** файлу і ще через меню головної панелі інструментів **Source** (текст програми), далі **Add/Remove Source File...** (додати – видалити текст програми мовою Асемблер) – додати назву файлу з текстом програми мовою Асемблер, натиснувши **New** і вибравши потрібний компілятор: **t15demo.asm** (рисунок 1.7).

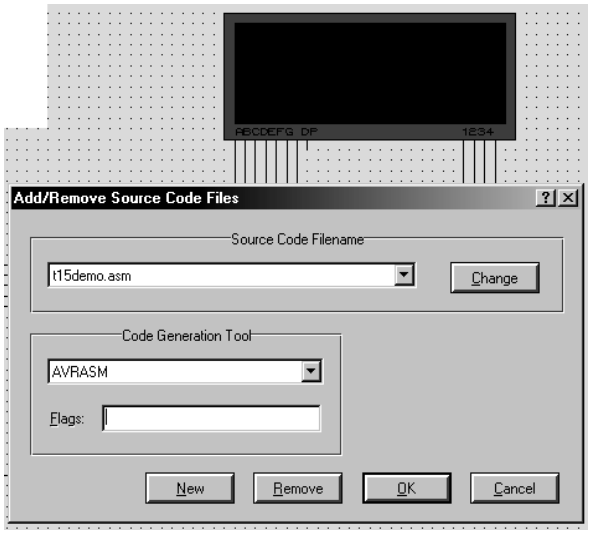

Рисунок 1.7 – Додавання файлу з текстом програми в проект

Тепер при запуску емуляції буде відбуватися компіляція файлу **t15demo.asm, і якщо нема помилок**, то МК почне працювати вже за оновленим файлом **T15DEMO.HEX.**

Увага! Це зображення проекту SAMPLES\AVR Tiny15 Demo! У нашому проекті ці поля ПОРОЖНІ! Тому що ми використовуємо файл *.cof.

Отже, щоб емулювати в PROTEUS роботу мікроконтролера, достатньо:

1) знайти його в бібліотеках і помістити на схему;

2) вказати, яку програму він повинен виконувати (як описано вище);

3) вказати частоту тактування МК.

Досить просто.

Можливо, PROTEUS попросить вас додати на схему вузли POWER та GROUND за допомогою лівої панелі інструментів.

7 Щоб побачити всі установки компонента в панелі редагування, поставте галочку **Edit All Prop. As Text (рисунок 1.8).**

CLOCK однозначно визначає частоту тактування МК при емуляції! Кварц і конденсатори не потрібні для емуляції, їх встановлюють на схему тільки для того, щоб урахувати при розведенні друкованої плати пристрою.

WDG_CLOCK показує частоту роботи генератора сторожового таймера. Хоча вона й позначена 1MHz – «собака» не вімкнений, поки ми не додамо у властивості МК рядок: {WDGON=1} – вам не потрібно його вмикати.

На рисунку 1.8 видно, що МК буде працювати за програмою **ds1990.cof** – його створив компілятор **CVAVR**.

Компілятор **WinAVR** створює файл із расширенням ***.elf** - приклад з папки **SAMPLES\AVR and SED1520.** У ньому вказана програма для МК **EW12A03GLY.hex** – замініть її на **EW12A03GLY.elf** і ви зможете бачити рух програми по тексту програми мовами С і Асемблер.

Значення інших параметрів можна довідатися в довідці – кнопка Help (рисунок 1.9).

На рисунку 1.9 зображена довідкова система PROTEUS і перелік мікроконтролерів AVR, які підтримуються цією програмою.

Зелені посилання пояснюють, як підключити файли для емуляції і налагодження програми по тексту програми мовою С.

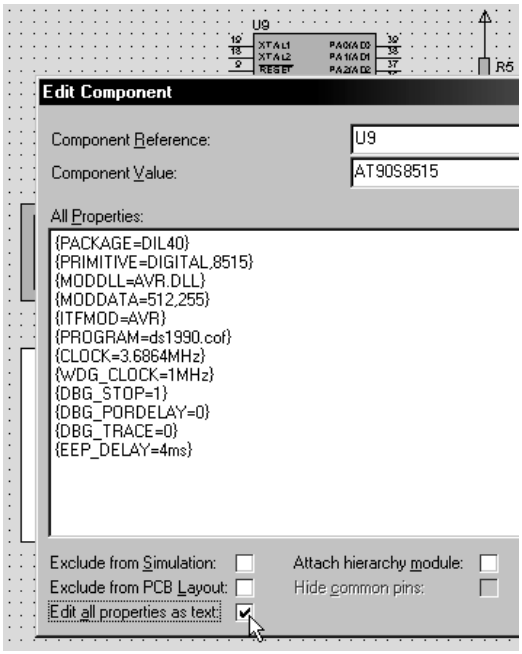

Рисунок 1.8 – Редагування властивостей компонента як тексту

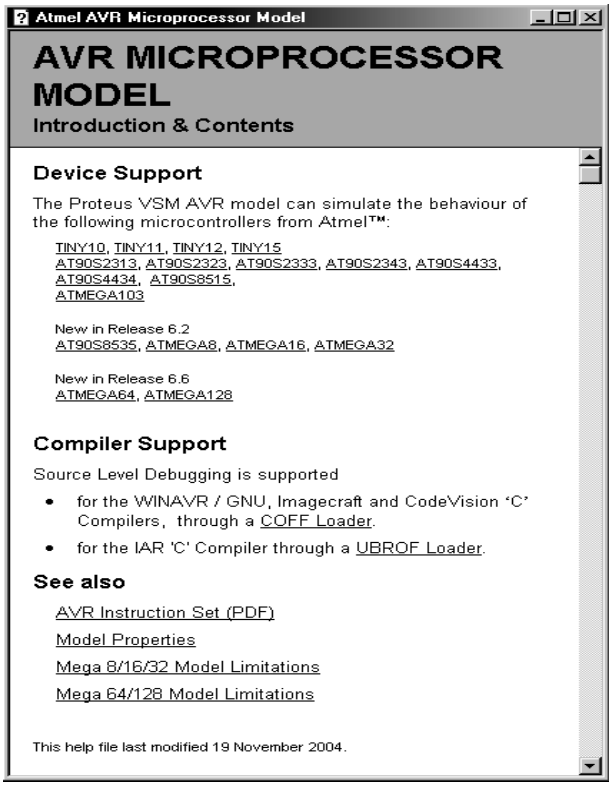

Рисунок 1.9 – Довідкова система PROTEUS

8 Дізнайтеся про обмеження моделей. Дуже важливо знати, що є в моделі МК – Model Limitations (рисунок 1.10):

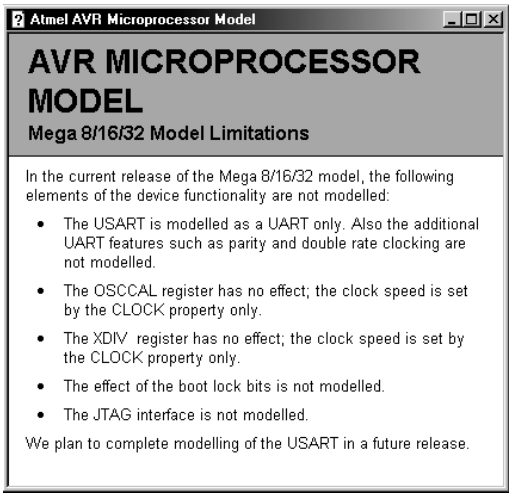

Рисунок 1.10 – Обмеження емуляції

1) симулюється звичайно використовуваний режим UART – асинхронне приймання і передача;

2) і 3) частота тактування при емуляції залежить тільки від параметра CLOCK;

4) біти **boot lock** не моделюються;

5) **JTAG** не моделюється.

Отже все, що потрібно моделюється!

Важливо! PROTEUS потрібно встановлювати для користувача з ім'ям з латинських символів або робити шлях до папки \temp без кирилиці!

Щоб налагодити функції PRINTF і SCANF в програмах, створених за допомогою компілятора CodeVision AVR, зайдіть у меню CVAVR – «проект» – «конфігурація» – «С компілятор» і в списках цих функцій виберіть «Long...» або «Float...», потім «ОК», звичайно...

9 Запустіть емуляцію! Кнопки керування емуляцією описано вище.

Крім того, керувати процесом емуляції програми мовою асемблер можна за допомогою клавіш F10, F11, F12 і їхніми комбінаціями з клавішами Alt та Ctrl та через меню DEBUG (рисунок 1.11).

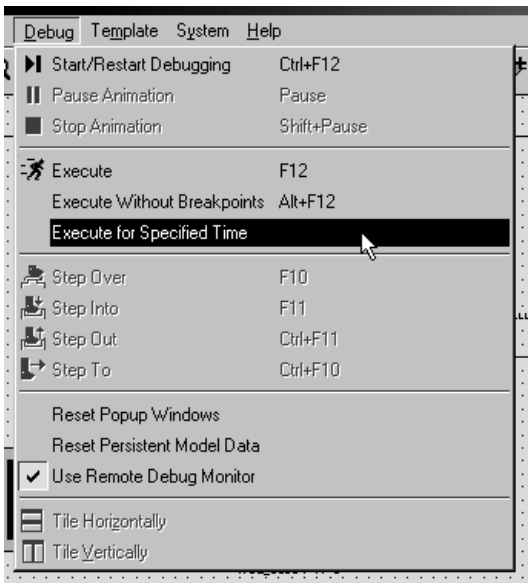

Рисунок 1.11 – Меню налагодження і емуляції

На рисунку 1.11 зображено меню налагодження і обрана команда, що дозволяє **виконати зазначений час емуляції.** Наприклад, 1 мікросекунду або 10.5 секунд.

При натисканні на неї виводиться поле введення для зазначення скільки часу повинна відпрацювати схема.

n – наносекунди;

u – мікросекунди;

m – мілісекунди:

Секунди можна не позначати.

Зазначений час буде просимульовано, якщо до його закінчення в програмі МК не зустрінеться **точка зупинки** (ВР – **BreakPoint**) – емуляція зупиниться на ній.

F12 – еквівалентна кнопці «Пуск» – запускає або продовжує емуляцію до ВР.

Alt+F12 – запускає або продовжує емуляцію, не зупиняється на ВР.

F10 – емулювати, не входячи в процедуру мовою Асемблер або функцію мовою С – вони будуть виконані як 1 крок.

F11 – емулювати крок за програмою з входженням у процедуру або функцію.

Ctrl+F11 – закінчити процедуру або функцію, у якій перебуваємо.

Ctrl+F10 – емулювати до виділеного мишею рядка **в програмі мовою Асемблер**. Ви повинні розуміти, що цей рядок може бути недосяжним по алгоритму програми.

Допомога з емуляції МК знаходиться в розділі – Proteus VSM Help – SOURCE LEVEL DEBUGGING WITHIN PROTEUS VSM.

При емуляції пристрою на екрані ми будемо бачити анімацію. Режим анімації можна встановити через меню **System -> Set Annimation Options (рисунок 1.12).**

На рисунку 1.12 зображено вікно настроювання параметрів анімації, що дозволяє відображувати напругу і струм на щупах; показувати логічні рівні на виводах; показувати кольорами напруги на провідниках; показувати стрілками напрямок струму в провідниках; звичайний режим – екран оновлюється 20 разів за секунду.

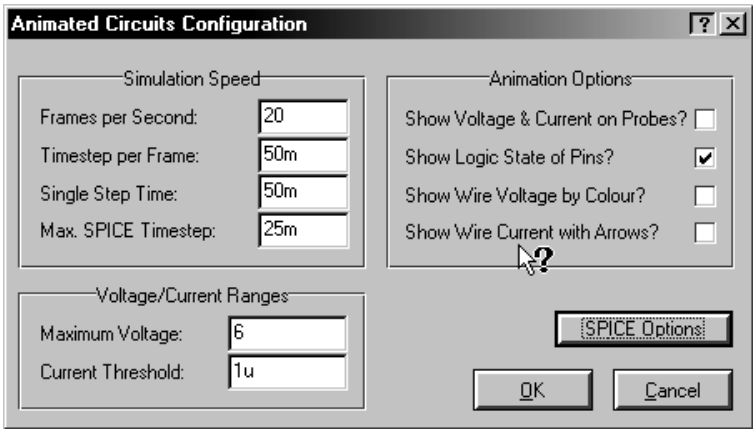

Рисунок 1.12 – Вікно настроювання параметрів анімації

Максимальна напруга не обмежує напругу у схемі, що симулюється! Вона вказується для того, щоб PROTEUS зміг вирішити, якими кольорами показувати логічні рівні. Крок оцифровування струму 1 мкА – ви можете зменшити цей параметр і поліпшити інші параметри для підвищення точності емуляції, але це потребує більшої обчислювальної потужності ПК.

SPICE Options – відкриває настроювання емуляції моделей.

Довідку по кожному рядку можна одержати за знаком питання.

СТАРТ...

10 Якщо у вас ще не відкритий проект **1WIRE_NET.DSN**, відкрийте його.

Запустіть емуляцію кнопкою «Пуск». З'явиться вікно віртуального термінала ПК, і приблизно за 1 секунду програма МК зробить усе, що від неї було потрібно.

Це виведення з UART МК на віртуальний COM порт ПК у вікно-термінал списку розпізнаних пристроїв на шині 1-Wire і їхні номери (рисунок 1.13).

При анімації логічні рівні на виводах вказуються кольоровими квадратиками.

Червоний – «Гарячий» – 1.

Синій – «Холодний» – 0.

Сірий – високоімпедансний вхід Z-стану. Для визначеності PROTEUS вважає, що на виводі половина напруги живлення МК.

Номер «таблетки» для домофонів DS1990, написаний на ній.

Серійні номери приладів DSxxxx задані на заводі і не змінювані.

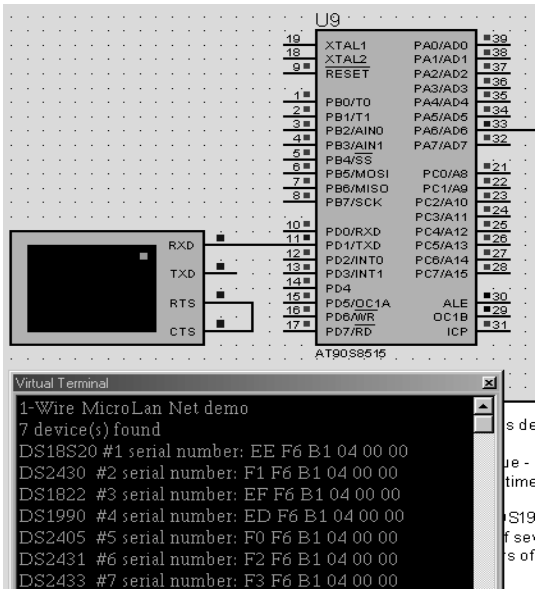

Рисунок 1.13 – Емуляція роботи МК

У PROTEUS ці номери задаються у вікні редагування властивостей компонента.

11 Натисніть кнопку «Стоп». Вікно термінала пропаде.

Давайте відключимо один 1-Wire прилад. Натисніть два рази праву кнопку миші на провіднику від мікросхеми **U5** – це видалить провідник зі схеми й відключить прилад від шини 1-Wire.

Натисніть кнопку «Пуск». **Знову з'явиться термінал,** але в списку розпізнаних приладів уже не з'явиться відключена **U5** і зменшиться кількість знайдених приладів до 6.

Відновіть вилучений провідник. Це можна зробити двома способами:

1) або **натиснути кнопку скасування дії** у верхній панелі інструментів;

2) або **прокласти провідник заново.** Виберіть інструмент – «провідник» на лівій панелі інструментів, проведіть провідник від компонента **U5** до шини 1-Wire, утримуючи натиснутою кнопку миші.

12 Ознайомтесь з віртуальним терміналом.

Зверніть увагу на підключення одного з найважливіших інструментів настроювання РЕАЛЬНИХ пристроїв до МК – термінал ПК, що обмінюється даними з UART (USART) МК.

Термінал (Virtual Terminal) знаходиться під кнопкою «віртуальні інструменти».

У PROTEUS ви можете не використовувати узгоджувач рівнів RS232 з UART типу MAX232, а підключати термінал прямо до МК.

Термінал підтримує службові символи ASCII – керування виведенням тексту: CR(0Dh), BS(0x08h) і BEL(0x07h).

LF(0x0A) та інші службові коди не підтримуються!

Підтримуються швидкості від 300 до 57600 бод, Формат данных 8N1 та інші.

У режимі паузи емуляції ви можете, натиснувши праву кнопку на екран термінала, відкрити меню, у якому можна скопіювати – вставити інформацію, можна змінити вид виведення числа – у вигляді символів або в HEX вигляді, можна очистити екран від даних, можна встановити режим відгуку – прийняті на **RXD** термінала символи будуть виводитися на його ніжку **TXD**.

Виводи **RTS** і **CTS** можна не з'єднувати і не підключати взагалі, хоча вони працюють як у дійсному COM порту ПК.

Термінал може передавати щось тільки за відсутності 0 на виводі **CTS.**

Можна використати кілька незалежних терміналів.

Якщо натиснути мишку у вікні термінала, то він почне передавати символи, що набирають на **клавіатурі** ПК, на ніжку TXD. Можна вставити з буфера обміну інформацію – термінал так само буде виводити її на ніжку **TXD**.

Інформація, що передається, не відображується у вікні термінала! Курсор залишається на місці. У властивостях термінала в полі **додаткові властивості** (Other Properties) можна ввести текст, який буде посилати з термінала при натисканні кнопки «Пуск» із затримкою на час передачі одного символу.

Наприклад, ви ввели: TEXT="Hello World". Термінал передасть: Hello World.

Докладніше про термінал читайте по кнопці Help у його властивостях.

Не забувайте про компонент COMPIM, який дозволяє вашому віртуальному пристрою підключитися до реального COM порту вашого ПК!

13 Натисніть «Стоп» потім «Старт» і ... «Пауза».

На екрані з'являться pop-up вікна з інформацією.

Перелік цих вікон задається внизу меню DEBUG (рисунок 1.14).

Відзначені для відображення 5 – вихідний код, 6 – вікно змінних.

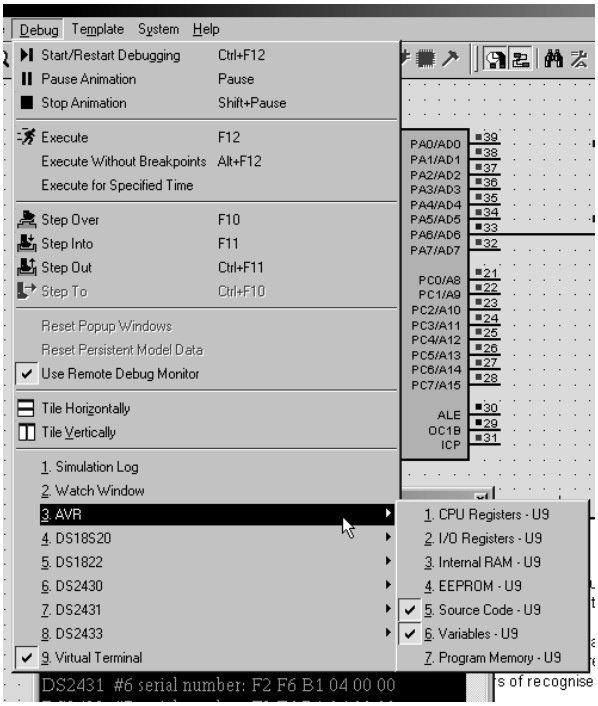

Рисунок 1.14 – Вибір інформаційних вікон компонентів

Ви можете відзначити стільки вікон, скільки вам потрібно.

Якщо у схемі кілька МК, то для кожного буде своє підменю.

Ви бачите, що й інші елементи схеми, у яких є пам'ять, мають свої підменю з можливістю виведення інформаційних вікон. Наприклад цифровий термометр ds18s20 має Scratch RAM і EEPROM.

Watch Window – вікно спостереження.

Після настроювання воно не зникає з екрана і при емуляції, і знаходженні в паузі. У цьому вікні ви можете розміщувати регістри МК і не тільки відслідковувати їхній вміст по ходу програми, але й задавати деякі умови і дії при досягненні цих умов, наприклад зупинити емуляцію. Після натискання на **вкладку 2** (рисунок 1.14) з'являється панель спостереження. Натиснувши на неї правою кнопкою миші, випадає меню для додавання спостережуваних регістрів за ім’ям або адресою.

Подвійним натисканням на назву регістра ви додаєте його у вікно спостереження (рисунок 1.15).

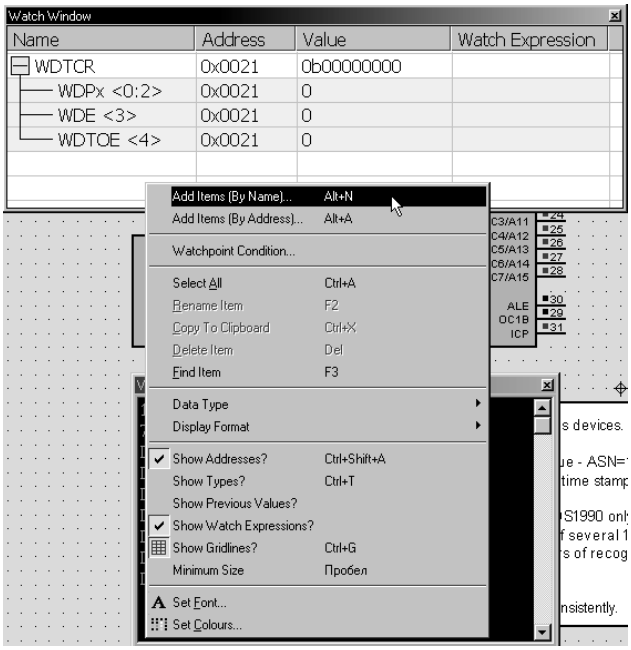

Рисунок 1.15 – Додавання регістрів у вікно спостереження

Якщо перед назвою регістра в **Watch Window** є квадратик із плюсом, ви можете розгорнути регістр на групи функціонально зв'язаних бітів.

Перевірте настроювання ліцензії PROTEUS!!!

Якщо ви задасте умову в **Watch Expression,** дії, які необхідно робити емулятору при її виникненні, визначаються у вікні, зображеному на рисунку 1.16.

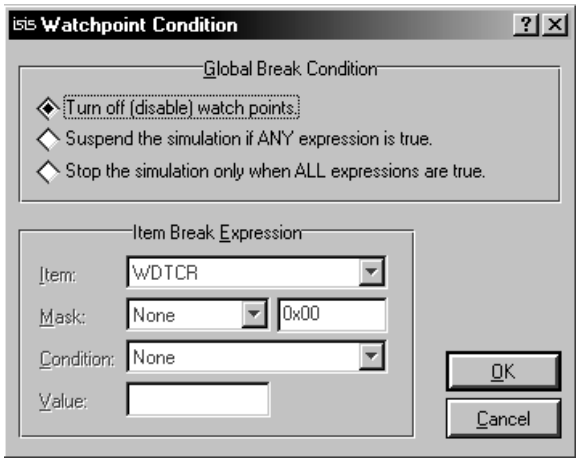

Рисунок 1.16 – Умови виникнення зупинок програми

Подивіться уважно і інші пункти меню DEBUG.

14 Практичне завдання. Складіть схему, як на рисунку 1.17, і приведіть її до робочого стану.

У даній схемі світлодіоди підключено до анодних виводів контролера, а катоди через резистори гасіння з'єднані з землею. Це означає, що світлодіод засвічується подачею 1 на відповідний вивід порту. Принципова схема наведена на рисунку 1.17.

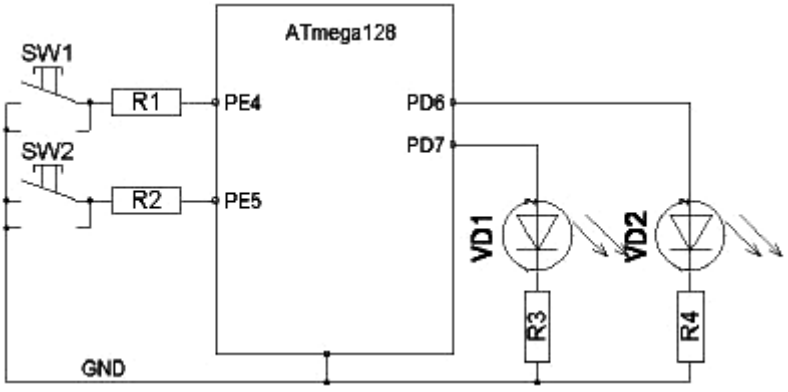

Рисунок 1.17 – Принципова схема

Схему складати, відкривши Proteus ISIS.

Необхідні компоненти розміщені в бібліотеці **Pick Divices**:

- світлодіоди, лампочки – Category – Optoelectronics, Sub-category – LEDs, компоненти LED, LAMP;

- резистори – Modelling primitives – pullup, pulldown;

- ключі (SW) – Button (кнопка), SW-SPST (ключ);

- LCD – дисплеї, наприклад таймер-лічильник Milford-2X16-BKP (вхід RXD).

Готові приклади знаходяться в **Program files** – Labcenter Electronics – Proteus 7 Professional – **Samples** – VSM for AVR.

1.5 Зміст звіту

Звіт повинен містити:

1 УГЗ та призначення виводів МК Atmega 128.

2 Складену схему за завданням.

3 Висновки про обсяг виконаної роботи, досягнену мету роботи, співпадіння чи неспівпадіння практичних результів з теоретичними.

Контрольні тестові питання

1 Які мікроконтролери підтримує емулятор електронних пристроїв PROTEUS?

2 Що відображує панель **DEVICES?**

3 Як виконати подачу живлення або певного сигналу в довільну точку схеми?

4 Який вигляд мають файли прошивання виконані мовами С, Асемблер та в машинних кодах?

5 Які дії необхідно провести, щоб проемулювати в PROTEUS роботу мікроконтролера?

6 Як задати частоту тактування МК при емуляції?

7 Яке призначення має віртуальний термінал?

8 Як задати необхідний час емуляції?

ЛАБОРАТОРНА РОБОТА 2
Програмування мікроконтролерів у програмному середовищі AVR Studio

2.1 Навчальні питання

1 Дослідження основних функцій і компонентів середовища AVR Studio.

2 Дослідження методики створення та налагодження програм прошивки мікроконтролерів за допомогою програми AVR Studio.

2.2 Навчальна мета

Практичне дослідження методики створення та налагодження програм прошивання мікроконтролерів за допомогою програми AVR Studio.

2.3 Теоретичні відомості

Програмування в середовищі AVR Studio. Для програмування AVR-мікроконтролерів існує немало засобів розроблення, проте найбільш популярним, поза сумнівом, слід визнати пакет AVR Studio. Однією з причин такої популярності є безкоштовний пакет, розроблений фірмою ATMEL, він об'єднує в собі текстовий редактор, Асемблер і симулятор. Пакет AVR Studio також використовується спільно з апаратними засобами налагодження.

Дистрибутив пакета і Service Pack можна завантажити з сайта www.atmel.com або одержати компакт-диск з цим дистрибутивом у російського дистриб'ютора фірми ATMEL.

Роботу пакета AVR Studio зручно розглядати на якій-небудь конкретній програмі. Для ілюстрації ми розглянемо створення проекту для простої програми, яка по черзі запалюватиме два світлодіоди. Для визначеності візьмемо мікросхему Atmega128 і підключимо два світлодіоди до виводів 31 і 32 (це біти 6 і 7 порту D мікросхеми Atmega128).

AVR-контролери мають потужні вихідні каскади, типовий струм кожного виводу складає 20 мА, максимальний струм виводу 40 мА, причому це стосується як вхідного, так і витікаючого струму. У нашому прикладі світлодіоди підключені анодами до виводів контролера, а катоди через гасильні резистори з'єднані із землею. Це означає, що світлодіод засвічується подачею 1 на відповідний вивід порту.

Принципова схема наведена на рисунку 2.1. На схемі також показано дві кнопки, які будуть використані в одній з програм.

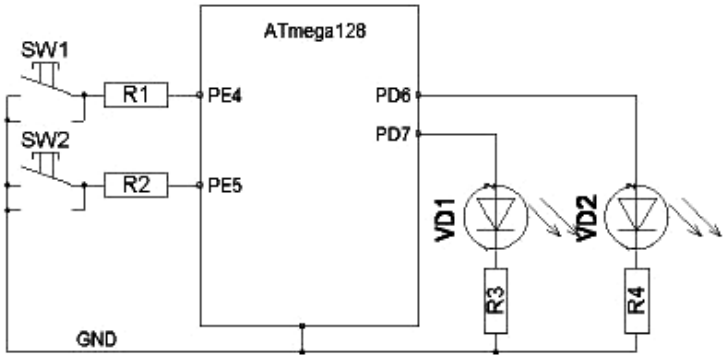

Рисунок 2.1 – Принципова схема

Тут доречно зробити невеликий відступ про вибір типу мікросхеми для простого прикладу. Дійсно, з першого погляду може здатися дивним, навіщо потрібен такий потужний кристал у 64-вивідному корпусі там, де вистачить і 8-вивідної мікросхеми ATtiny12? Проте в такому підході є логіка. Відомо, що в основі практично будь-якого AVR-контролера лежить однакове ядро. За великим рахунком, контролери розрізняються об'ємом пам'яті, кількістю портів введення/виведення і набором периферійних модулів. Особливості кожного конкретного контролера – прив'язка логічних імен регістрів введення/виведення до фізичних адрес, адреси векторів переривань, визначення бітів портів і т. д., які описані у файлах з розширенням ***.inc***, що входять до складу пакета AVR Studio. Отже, використовуючи конкретний тип кристала, можна настроювати програму як власне для нього, так і для будь-якого молодшого кристала. Далі, якщо використовувати як налагоджувальний найстарший кристал, можна настроювати програму практично для будь-якого AVR-контролера, треба просто не використовувати апаратні ресурси,

відсутні в цільового мікроконтролера. Таким чином, наприклад, можна налагоджувати на Atmega128 програму, яка виконуватиметься на ATtiny13. При цьому початковий код залишиться практично тим самим, зміниться лише ім'я файлу, що підключається, з ***128def.inc*** на ***tn13def.inc***.

У такого підходу також є свої переваги. Наприклад, «зайві» порти введення/виведення можна використовувати для підключення РК-індикатора, на який можна виводити налагоджувальну інформацію або скористатися внутрішньосхемним емулятором, який підключається до JTAG-порту мікросхеми Atmega128 (контролер ATtiny13 такий порт не має). Таким чином, можна використовувати єдину налагоджувальну плату, на якій встановлений «старший» AVR-контролер, для налагодження будь-яких систем, що знов розробляються, природно, що базуються також на AVR-мікроконтролерах. Одна з таких плат називається AS-megaM. Саме вона використовувалася для створення прикладів програм, що наводяться в даній роботі. Це універсальний одноплатний контролер на базі мікросхеми Atmega128, який містить зовнішнє ОЗУ, два порти RS-232, порт для підключення РК-індикатора, внутрішньосхемного програматора і емулятора AT JTAG ICE. На платі також є місце для розпаювання мікросхеми FLASH-ПЗП серії AT45 в корпусах TSOP32/40/48 і двоканального ЦАП серії AD5302/ AD5312/ AD5322. Тепер, після пояснення причин використання AVR-монстра для запалювання пари світлодіодів, можна йти далі.

При програмуванні в середовищі AVR Studio треба виконати стандартну послідовність дій:

- створення проекту;

- завантаження файлу;

- компіляція;

- симуляція;

- завантаження hex-коду в мікроконтролер.

Створення проекту починається з вибору рядка меню Project\New Project. У вікні Create new, що відкриє Project треба вказати ім'я проекту (у нашому випадку – sample1) і ім'я файлу ініціалізації. Після натискання кнопки Next відкривається вікно Select debug platform and device, де вибирається налагоджувальна платформа ***(симулятор або емулятор)*** і тип мікроконтролера.

Можна вибрати один із запропонованих внутрішньосхемних ***емуляторів***. Зазначимо, що в кожного емулятора свій список підтримуваних мікросхем. Для даного прикладу ми вибираємо як налагоджувальну платформу ***AVR Simulator*** і мікросхему **Atmega128**. Після натискання кнопки Finish ми побачимо власне робочі вікна пакета AVR Studio, поки порожні.

Слід у центральне вікно помістити початковий текст програми. Це можна зробити двома способами: або набрати весь текст безпосередньо у вікні редактора, або завантажити вже існуючий файл. Нижче наведено повний текст простої програми з коментарями.

Приклад 1

```
; Приклад «Керування світлодіодами»
; написаний для налагоджувальної плати AS-MegaM
; Частота задавального генератора 7,37 Мгц
; світлодіоди підключені до виводів PD6 і PD7 і через резистори – на загальний дріт
; підключення файлу опису введення-виведення мікросхеми Atmega128
.include "m128def.inc"
; початок програми
begin:
; перша операція – ініціалізація стеку
; якщо цього не зробити, то виклик підпрограми або переривання
; не поверне керування назад
; покажчик на кінець стеку встановлюється на останню адресу внутрішнього ОЗУ – RAMEND
ldi r16,low(RAMEND)
out spl,r16
ldi r16,high(RAMEND)
out sph,r16
; для того щоб керувати світлодіодами, підключеними до виводів PD6 і PD7,
; необхідно оголосити ці виводи вихідними.
; для цього потрібно записати 1 у відповідні біти регістра DDRD (DataDiRection)
ldi r16,(1<<6) | (1<<7)
```

```
out DDRD,r16
; основний цикл програми
loop:
ldi r16,(1<<6) ; світиться один світлодіод
out PORTD,r16
rcall delay ; затримка
ldi r16,(1<<7) ; світиться другий світлодіод
out PORTD,r16
rcall delay ; затримка
rjmp loop ; повторення циклу
; процедура затримки приблизно півсекунди при частоті 7,37 МГц
; три порожні вкладені цикли відповідно
delay:
ldi r16,30 ; 30
delay1:
ldi r17,200 ; 200
delay2:
ldi r18,200 ; і ще 200 ітерацій
delay3:
dec r18
brne delay3
dec r17
brne delay2
dec r16
brne delay1
ret ; повернення в головну програму
```

Проект може складатися з декількох файлів, при цьому один файл призначається основним. Всі операції зручно проводити, використовуючи контекстну кнопку миші.

Компіляція проекту проводиться командою **Build** або натисканням кнопки **F7**. Процес компіляції відображується в нижньому вікні Build.

У AVR Studio для виведення повідомлень про проходження асемблювання існує вікно Build (рисунок 2.2).

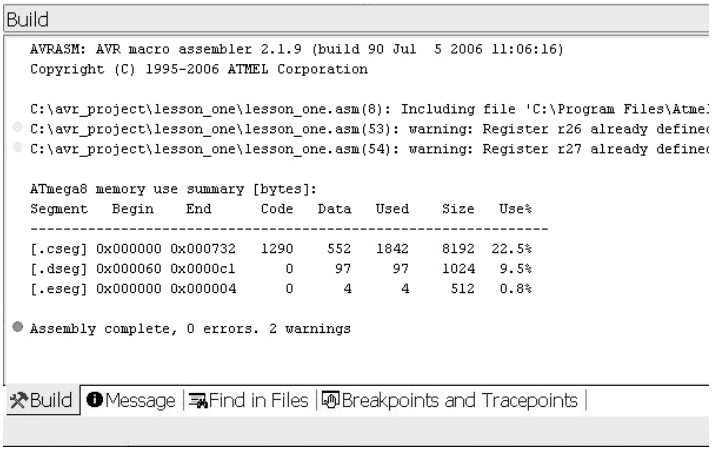

Рисунок 2.2

Після асемблювання у вікні Build з'являється повідомлення: Assembly complete, xx errors. xx warnings. Якщо асемблювання закінчилося з помилками, у вікні Build з'явиться список цих помилок, перейти на рядок, що містить помилку, можна натиснувши на рядок з цим повідомленням у вікні.

Після того як програма скомпілювалася без помилок, можна просимулювати її роботу.

Для симуляції роботи програми необхідно обрати в меню **Debug – Start Debugging** або відразу натискати **Build and Run** чи **CTRL+F7**.

З'явиться жовта стрілка, що вказує на поточний крок симуляції.

Команди налагодження:

1 Run - Пуск (F5) – запуск виконання програми.

2 Reset – Скидання (SHIFT+F5) – зупинка процесу трасування, курсор переміщується на початок виконуваної програми

3 Break – Перервати (CTRL+F5) – перервати виконання програми

4 Step Into – Покрокове налагодження (F11) – найкорисніша команда, результат роботи мікроконтролера видно в лівій панелі периферії, а це порти введення/виведення, таймери/лічильники, прапори системних регістрів тощо.

5 Run to Cursor – Запустити з позиції курсора (CTRL+F10) – перехід на позицію курсора, симуляція продовжується з цього місця

Після налагодження натиснути Save.

Ми вже одержали вихідний файл у форматі .hex, який вже можна завантажувати в мікросхему і спостерігати миготіння світлодіодів.

Пакет AVR Studio містить потужні засоби для перегляду і редагування стану внутрішніх регістрів і портів введення/виведення відлагоджуваного мікроконтролера, а також час виконання програми. Доступ до них здійснюється через вікно I/O.

Для полегшення налагодження існують вікна:

1 **View-Watch window** – вікно показує значення і адреси заданих змінних (рисунок 2.3).

Змінні можна додавати вручну або за допомогою меню правої кнопки миші

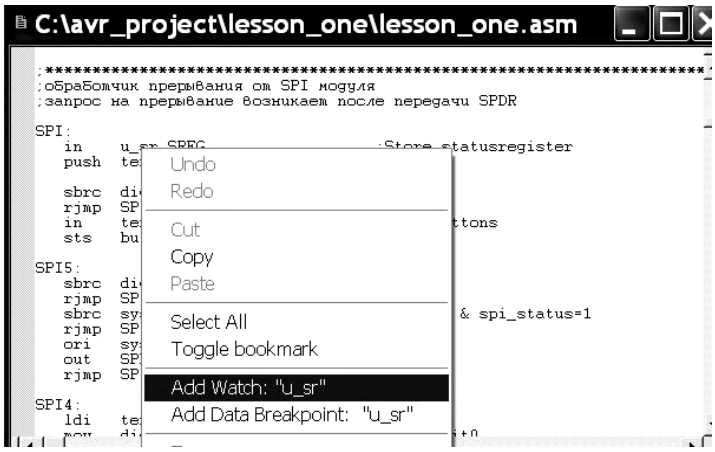

Рисунок 2.3

Значення змінних можна додавати і змінювати під час зупинки роботи програми.

2 **Register window** – вікно показує вміст регістрів. Регістри можна змінювати під час зупинки програми (рисунок 2.4).

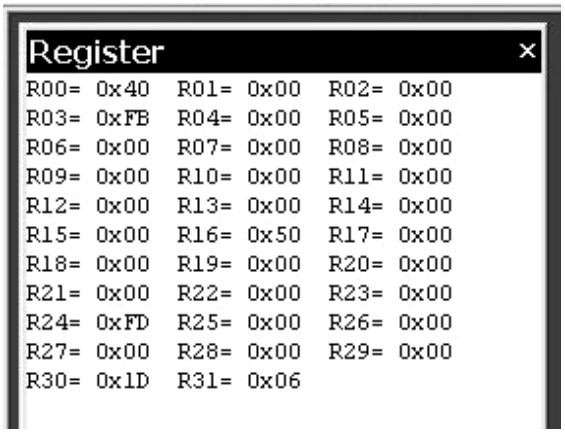

Рисунок 2.4

3 **Memory windows** – вікна показують вміст пам'яті програм, даних, портів введення/виведення і незалежного ПЗП (рисунок 2.5).

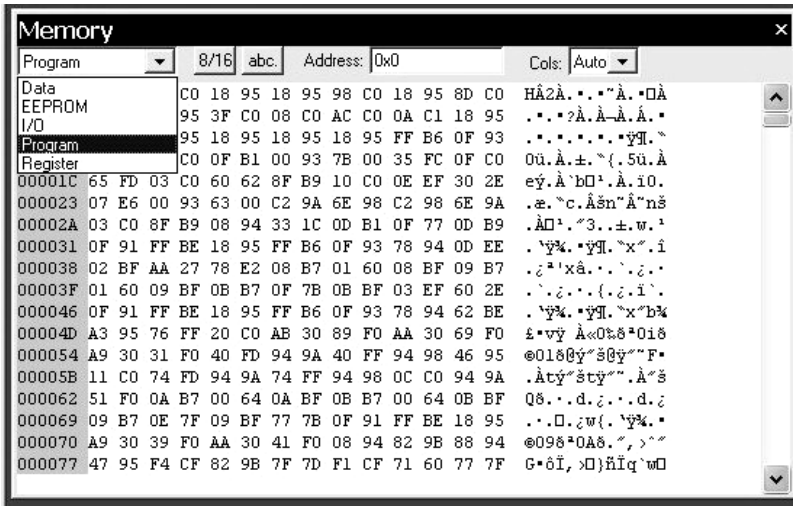

Рисунок 2.5

Пам'ять можна проглядати в HEX, двійковому або десятковому форматах. Вміст пам'яті можна змінювати під час зупинки програми.

4 **I/O View** – показує вміст різних регістрів введення/виведення: EEPROM, USART, таймери та ін.

5 **Project window** – вікно показує назву тих файлів, що входять в проект.

6 **Message window** – вікно показує повідомлення від AVR Studio.

При першому запуску потрібно налаштувати вікна для керування і виведення необхідної інформації. Під час наступної сесії роботи настроювання автоматично відновлюються.

Основні асемблерні команди мікроконтролерів AVR

Для вивчення азів програмування мікроконтролерів AVR мовою Асемблер необхідно розуміти значення асемблерних мнемонік. У новітніх мікроконтролерах AVR сімейства MEGA доступно близько двох сотень операцій, і майже кожна з команд виконується мікроконтролером за один такт, за виключення команд галуження і апаратного множення.

Найчастіше вживані асемблерні команди необхідно знати для розуміння чужого коду і тим більше для написання свого.

Регістри введення/виведення:

CBI – скидання біта порту;

SBI – установлення мітки. 1 біта порту;

IN – вантаження значення з порту в регістр;

OUT – завантаження значення в регістр;

SBIC – пропускання наступної команди, якщо біт порту скинутий;

SBIS – пропускання наступної команди, якщо біт порту встановлений.

Галуження:

CALL – абсолютний виклик;

RCALL – відносний виклик;

RET – повернення з підпрограми;

RETI – повернення з переривання, прапорець дозволу переривань I «жорстко» встановлюється;

JMP – абсолютний перехід;

RJMP – відносний перехід;

BRBC – перехід, якщо біт регістра SREG скинутий;

BRBS – перехід, якщо біт регістра SREG встановлений;

SBRC – пропустити наступну команду, якщо біт регістра скинутий;

SBRS – пропустити наступну команду, якщо біт регістра встановлений.

Робота зі стеком:

PUSH – зберегти регістр у стеку;

POP – добування регістра зі стеку.

Важливі команди:

NOP – команда, яка нічого не робить;
CLI – заборонити переривання;
SEI – дозволити переривання;
CP – порівняти значення двох регістрів загального призначення;
CPI – порівняти значення регістра з константою;
LDI – завантажити константу в регістр загального призначення;
CLR – очистити регістр загального призначення.

Приклади застосування даних команд:

```
ldi        R16, 0b00001001 ;   завантаження константи в регістр загального призначення (R16 - R32)
out    PORTD R16        ;   запис значення регістра в порт D

in      R25, PORTB        ; зчитувати значення Port B в регістр R25
cpi     R25, 4            ; порівняти лічене значення з константою =4
breq   exit               ; перехід на мітку, якщо вони були рівними

exit:                     ; мітка
nop                       ; порожній такт

sbi       PORTD PD4       ; записати в 4-й біт порту D мітку 1
rcall     my_delay        ; виклик підпрограми затримки
cbi       PORTD PD4       ; скидання 4-го біта порту D

my_delay:                 ; підпрограма затримки (4 такти)

  nop                     ; холостий такт
  nop
  nop
  nop
  nop

ret                       ; повернення з підпрограми (3 такти)
; виклик і виконання підпрограми my_delay займе 4+ 5+3 = 12 тактів
```

2.4 Порядок виконання роботи

1 Створити новий проект в AVR Studio, при цьому вибрати як налагоджувальну платформу ***AVR Simulator*** і мікросхему **Atmega128**.

2 Для мікроконтролера Atmega128 на схемі рисунка 2.1 написати програму послідовного вмикання лампочок (світлодіодів) з періодом перемикання 0,5 секунди (див. приклад 1).

3 Виконати налагодження програми.

Встановити у вікні **Debug**\Simulator Options частоту кварцу 7,3728 МГц для точного вимірювання часу виконання програми.

Решту опцій слід залишити без зміни. Тепер можна виконувати програму в покроковому режимі за допомогою миші або кнопки **F11.**

Для налагодження нашого прикладу, щоб отримати доступ до бітів порту D, треба розкрити рядок I/O ATMEGA128, а потім рядок PORTD. Тепер видно всі три регістри цього порту: PORTD, DDRD і PIND.

Після цього 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 записуванням нового коду в полі Value, або безпосередньо натиснувши мишею на потрібному біті регістра.

4 Відкрити створений проект (рисунок 2.1) у PROTEUS і підключити написану програму **Atmega128.** Запустити симуляцію, перевірити працездатність схеми.

5 Написати програму, яка відрізняється від попередньої тим, що запалюється три рази спочатку перший діод з періодом 3 секунди, потім три рази другий діод з періодом 5 секунд. Цикл повинен повторювати 50 разів.

6 Написати програму, яка відрізняється від попередньої тим, що запаленням світлодіодів керують дві кнопки.

7 Виконати налагодження програми.

8 Відкрити створений в лр1 проект у PROTEUS і підключити написану програму до **Atmega128.** Запустити симуляцію, перевірити працездатність схеми.

Таким чином, на прикладі простих програм показано деякі можливості пакета AVR Studio. Треба розуміти, що це лише перше знайомство, що дозволяє швидше освоїтися з базовими командами пакета. Тим часом можливості даного пакета набагато ширше. Наприклад, тут можна налагоджувати програми, написані мовами високого рівня. Зокрема Сі-компілятор фірми ImageCraft користується налагоджувачем AVR Studio «як рідним». Для цього при компіляції початкового коду треба встановити опцію генерації вихідного файлу у форматі, сумісному з AVR Studio. При цьому з'являється можливість проводити налагодження в початкових кодах.

2.5 Зміст звіту

Звіт повинен містити:

1 Тексти написаних програм.

2 Перелік використаних команд з поясненням їх призначення.

3 Висновки про обсяг виконаної роботи, досягнену мету роботи, співпадіння чи неспівпадіння практичних результатів з теоретичними.

Контрольні тестові питання

1 Які мікроконтролери підтримує програмне середовище AVR Studio?

2 Чим відрізняються Atmega8 і Atmega128?

3 Як жорстко задати точний час виконання програми?

4 Який вигляд мають файли прошивки виконані, мовами С, Асемблер і в машинних кодах?

5 Які дії необхідно провести, щоб проемулювати в PROTEUS роботу мікроконтролера?

6 Як задати частоту тактування МК при емуляції?

7 Яке проглянути виконання програми покроково?

8 Як задати необхідний час емуляції?

ЛАБОРАТОРНА РОБОТА 3
Дослідження способів програмування портів мікроконтролерів AVR

3.1 Навчальні питання

1. Дослідження методики створення та налагодження програм прошивання мікроконтролерів за допомогою програми AVR Studio.
2. Запрограмувати пристрій керування світлодіодними індикаторами за допомогою кнопок.

3.2 Навчальна мета

Практичне дослідження способів і засобів програмування портів мікроконтролерів AVR.

3.3 Теоретичні відомості

Загальні відомості

Мікроконтролери AVR фірми Atmel сімейства Mega є 8-бітними мікроконтролерами, призначеними для використання у вбудованих системах. Вони виготовляються за КМОП-технологією малого споживання, мають удосконалену RISC-архітектуру, що дозволяє досягти найкращого співвідношення вартості, швидкодії та енергоспоживання. Мікроконтролери цього сімейства є найбільш розвиненими представниками мікроконтролерів AVR загального застосування.

Відмінні риси

До особливостей мікроконтролерів AVR сімейства Mega можна віднести:

– FLASH-пам'ять програм об’ємом від 8 до 256 кбайтів (кількість циклів стирання/запису не менше 10 000);

– оперативна пам'ять (статичне ОЗП) об’ємом від 512 байтів до 8 кбайтів;

– пам'ять даних на основі ЕСППЗП (EEPROM) об’ємом від 256 байтів до 4 кбайтів (кількість циклів стирання/запису не менше 100 000);

– можливість захисту від читання й модифікації пам'яті програм і даних;

– можливість програмування безпосередньо в системі через послідовні інтерфейси SPI та JTAG;

– можливість самопрограмування;

– можливість внутрішньосхемного настроювання у відповідності зі стандартом IEEE 1149.1 (JTAG), а також наявність власного однопровідного інтерфейсу внутрішньосхемного настроювання debugWire;

– різноманітні способи синхронізації: вбудований RC-генератор із внутрішньою або зовнішньою RC ланкою, вбудований генератор із зовнішнім кварцовим резонатором, зовнішній сигнал синхронізації;

– наявність декількох режимів зниженого енергоспоживання;

– наявність детектора зниженої напруги живлення (Brown-Out Detector-BOD);

– можливість програмного зниження частоти тактового генератора.

Характеристики процесора

Основними характеристиками процесора мікроконтролерів AVR сімейства Mega є:

– повністю статична архітектура, мінімальна тактова частота дорівнює нулю;

– арифметико-логічний пристрій (АЛП) підключено безпосередньо до регістрів загального призначення (32 регістри);

– більшість команд виконуються за один період тактового сигналу;

– векторна система переривань, підтримка черги переривань;

– велика кількість джерел переривань (16 внутрішніх і 2 зовнішніх);

– наявність апаратного перемножувача.

Характеристики підсистеми введення/виведення

Підсистема введення/виведення мікроконтролерів AVR сімейства Mega має такі особливості:

– програмне конфігурування і вибір портів введення/виведення;

– виводи можуть бути запрограмовані як вхідні або як вихідні незалежно один від одного;

– вхідні буфери з тригером Шмідта на всіх виводах;

– є можливість повного відключення цифрового порту введення/виведення від фізичного виводу мікросхеми;

– на всіх входах є внутрішні індивідуальні підтягувальні резистори опором 20...50 кОм.

Периферійні пристрої

Мікроконтролери сімейства Mega мають багатий набір периферійних пристроїв (ПП):

– один або два 8-бітних таймери/лічильники. У всіх моделях із двома 8-бітними таймерами/лічильниками один з них може працювати як годинник реального часу (в асинхронному режимі);

– від одного до чотирьох 16-бітних таймерів/лічильників;

– сторожовий таймер;

– одно- і двоканальні генератори 8-бітного ШІМ-сигналу (один з режимів роботи 8-бітних таймерів/лічильників);

– дво- і триканальні генератори ШІМ-сигналу регульованої розрядності (один з режимів роботи 16-бітних таймерів/лічильників). Розрядність сформованого сигналу може становити від 1 до 16 бітів;

– аналоговий компаратор;

– багатоканальний 10-бітний АЦП послідовного наближення, що має як несиметричні, так і диференціальні входи;

– послідовний синхронний інтерфейс SPI;

– послідовний двопровідний інтерфейс TWI (повний аналог інтерфейсу I2C);

– від одного до чотирьох універсальних синхронних/асинхронних прийомо-передавачів (USART);

– універсальний послідовний інтерфейс USI, що може використовуватися як інтерфейс SPI або I2C.

Архітектура ядра

Ядро мікроконтролерів AVR сімейства Mega виконано за вдосконаленою RISC-архітектурою (enhanced RISC) (рисунок 3.1), у якій використовується ряд рішень, спрямованих на підвищення швидкодії мікроконтролерів.

Арифметико-логічний пристрій (АЛП), що виконує всі обчислення, підключено безпосередньо до 32 робочих регістрів, об'єднаних у регістровий файл. Завдяки цьому АЛП може виконувати одну операцію (читання вмісту регістрів, виконання операції і запис результату назад у регістровий файл) за такт. Крім того, практично кожна з команд (за винятком команд, у яких одним з операндів є 16-бітна адреса) займає одну комірку пам'яті програм.

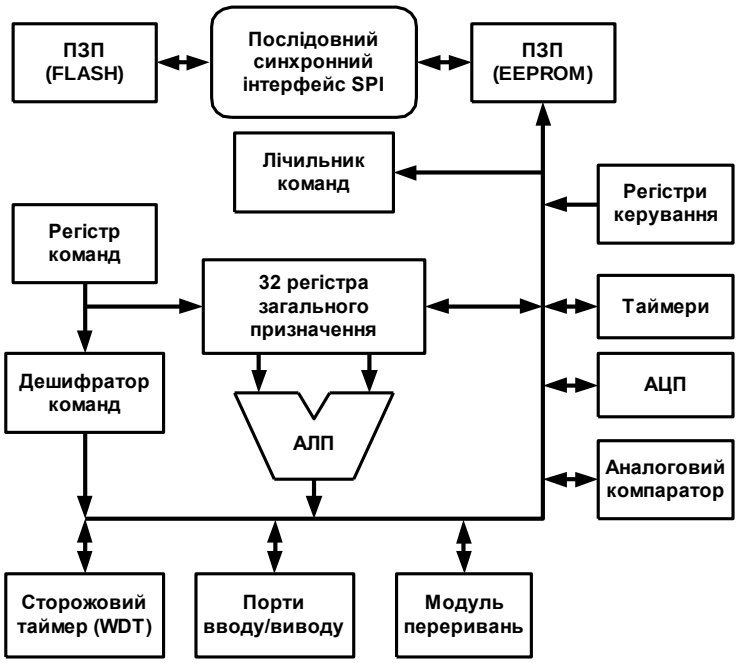

Рисунок 3.1 – Архітектура ядра мікроконтролерів AVR

У мікроконтролерах AVR реалізована гарвардська архітектура, що характеризується роздільною пам'яттю програм і даних, кожна з яких має власні шини доступу. Така організація дозволяє одночасно працювати як з пам'яттю програм, так і пам'яттю даних. Поділ інформаційних шин дозволяє використати для кожного типу пам'яті шини різної розрядності, причому способи адресації й доступу до кожного типу пам'яті також розрізняються. У сполученні з дворівневим конвеєром команд така архітектура дозволяє досягти продуктивності в 1 MIPS на кожен мегагерц тактової частоти.

Розміщення та опис виводів (таблиця 3.1)

Мікроконтролер Atmega8 конструктивно виконаний у 32-вивідному корпусі типу TQFP і MLF (також випускається у 28-

вивідному корпусі типу DIP) з максимальною кількістю контактів введення/виведення, рівною 23: Atmega8, Atmega8L (рисунок 3.2) мають FLASH-пам'ять програм об’ємом 8 кбайтів, ОЗП об’ємом 1 кбайт та EEPROM-пам'ять даних об’ємом 512 байтів.

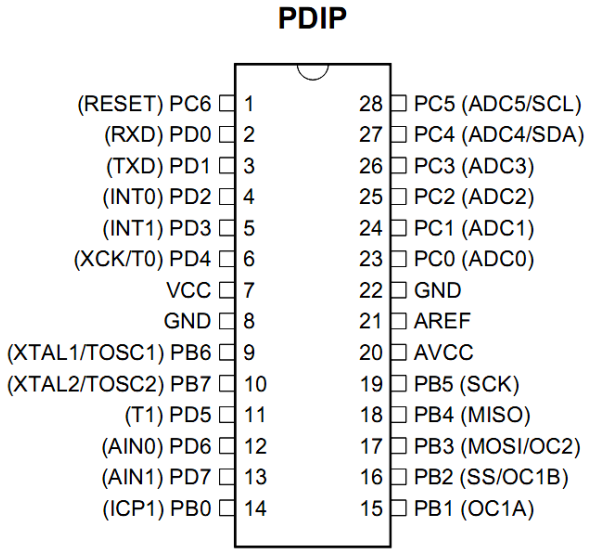

Рисунок 3.2 - Розміщення виводів мікроконтролера Atmega8.

Таблиця 3.1 – Опис виводів мікроконтролера Atmega8

Позначення	Номер		Тип	Опис
	DIP	TQFP MLF		
1	2	3	4	5
Порт B. 8-бітний двонаправлений порт введення/виведення з внутрішніми підтягувальними резисторами				
PB0 (ICP)	14	12	I/O	0-й біт порту B. Вхід захвата таймера/лічильника T1
PB1 (OC1A)	15	13	I/O	1-й біт порту B. Вихід A таймера/лічильника T1
PB2 (SS/OC1B)	16	14	I/O	2-й біт порту B. Вибір Slave-пристрою на шині SPI Вихід B таймера/лічильника T1

Продовження таблиці 3.1

1	2	3	4	5
PB3 (MOSI/OC2)	17	15	I/O	3-й біт порту B. Вихід (Master) або вхід (Slave) даних модуля SPI Вихід таймера/лічильника T2
PB4 (MISO)	18	16	I/O	4-й біт порту B. Вхід (Master) або вихід (Slave) даних модуля SPI
PB5 (SCK)	19	17	I/O	5-й біт порту B. Вихід (Master) або вхід (Slave) тактового сигналу модуля SPI
PB6 (XTAL1/TOSC1)	9	7	I/O	6-й біт порту B. Вхід тактового генератора Вивід для підключення резонатора до таймера/лічильника T2
PB7 (XTAL2/TOSC2)	10	8	I/O	7-й біт порту B. Вихід тактового генератора Вивід для підключення резонатора до таймера/лічильника T2
Порт C. 7-бітний двонаправлений порт введення/виведення з внутрішніми підтягувальними резисторами				
PC0 (ADC0)	23	23	I/O	0-й біт порту C. Вхід АЦП
PC1 (ADC1)	24	24	I/O	1-й біт порту C. Вхід АЦП
PC2 (ADC2)	25	25	I/O	2-й біт порту C. Вхід АЦП
PC3 (ADC3)	26	26	I/O	3-й біт порту C. Вхід АЦП
PC4 (ADC4/SDA)	27	27	I/O	4-й біт порту C. Вхід АЦП Вхід/вихід даних модуля TWI

Продовження таблиці 3.1

1	2	3	4	5
PC5 (ADC5/SCL)	28	28	I/O	5-й біт порту C. Вхід АЦП Вхід/вихід тактового сигналу модуля TWI
PC6 (RESET)	1	29	I/O	6-й біт порту C. Вхід скидання
ADC6	—	19	I	Вхід АЦП
ADC7	—	22	I	Вхід АЦП
Порт D. 8-бітний двонаправлений порт введення/виведення з внутрішніми підтягувальними резисторами				
PDO (RXD)	2	30	I/O	0-й біт порту D. Вхід USART
PD1 (TXD)	3	31	I/O	1-й біт порту D. Вихід USART
PD2 (INTO)	4	32	I/O	2-й біт порту D. Вхід зовнішнього переривання
PD3 (INT1)	5	1	I/O	3-й біт порту D. Вхід зовнішнього переривання
PD4 (TE/XCK)	6	2	I/O	4-й біт порту D. Вхід зовнішнього тактового сигналу таймера/лічильника TE Вхід/вихід зовнішнього тактового сигналу USART
PD5 (T1)	11	9	I/O	5-й біт порту D. Вхід зовнішнього тактового сигналу таймера/лічильника T1
PD6 (AINO)	12	10	I/O	6-й біт порту D. Прямий вхід аналогового компаратора
PD7 (AIN1)	13	11	I/O	7-й біт порта D Інверсний вхід аналогового компаратора

Продовження таблиці 3.1

1	2	3	4	5
AREF	21	20	р	Вхід опорної напруги для АЦП
AVCC	20	18	р	Вивід джерела живлення АЦП
VCC	7	4,6	р	Вивід джерела живлення
GND	8,22	3,5,21	р	Загальний вивід

Архітектура мікроконтролерів Atmega8

Загальні відомості

Мікроконтролери AVR сімейства Mega є 8-бітними мікроконтролерами з RISC-архітектурою. Вони містять у своєму складі пам'ять програм (FLASH) і даних (EEPROM), а також різноманітні периферійні пристрої.

На рисунку 3.3 наведено структурну схему мікроконтролера Atmega8. Особливостями даної моделі є:

– 3 порти введення/виведення (порти B, D – 8-бітні, порт C – 7-бітний);

– вхід апаратного скидання й виводи для підключення резонатора сполучені з лініями введення/виведення;

– два 8-бітних (T0, T2) і один 16-бітний (T1) таймер/лічильник;

– 3 канали ШІМ;

– по одному інтерфейсному модулю USART, SPI та TWI;

– 6- або 8-канальний (залежно від корпусу) 10-бітний АЦП.

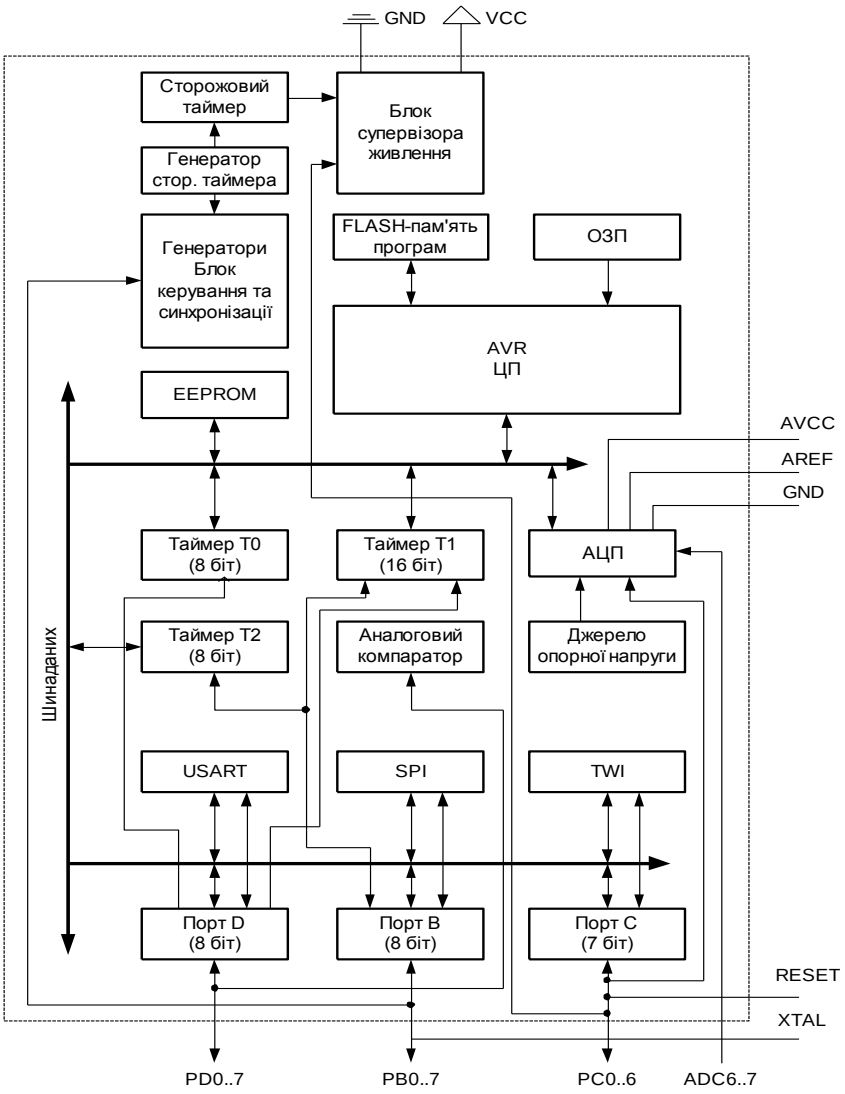

Рисунок 3.3 – Структурна схема мікроконтролерів ATmega8

Організація пам'яті

У мікроконтролерах AVR сімейства Mega реалізована гарвардська архітектура, відповідно до якої розділені не тільки адресні простори пам'яті програм і пам'яті даних, але також і шини доступу до них. Способи адресації й доступу до цих областей пам'яті також різні. Така структура дозволяє центральному процесору працювати одночасно як з пам'яттю програм, так і пам'яттю даних, що істотно збільшує продуктивність. Кожна з областей пам'яті даних (ОЗП і EEPROM) також розташована у своєму адресному просторі.

Узагальнена карта пам'яті мікроконтролерів AVR сімейства Mega наведена на рисунку 3.4.

Зверніть увагу на таке. Оскільки мікроконтролери AVR мають 16-бітну систему команд, об’єм пам'яті програм на

рисунку зазначений не в байтах, а в 16-бітних словах. Символ «$» перед числом означає, що це число записано в шістнадцятковій системі числення.

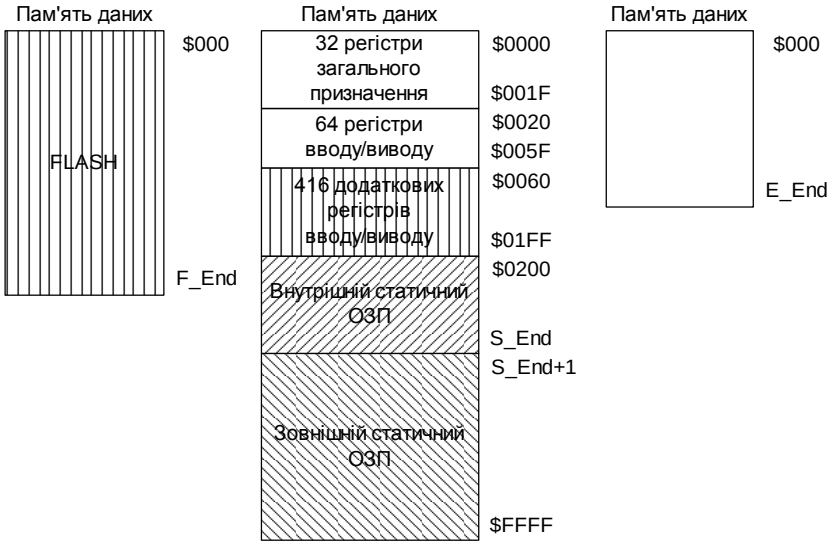

Рисунок 3.4 – Карта пам'яті мікроконтролерів сімейства Mega

Регістри введення/виведення

Всі регістри введення/виведення умовно можна розділити на дві групи: службові регістри мікроконтролера й регістри, що належать до конкретних периферійних пристроїв (у тому числі регістри портів введення/виведення).

У всіх мікроконтролерах AVR регістри введення/виведення (РВВ) розташовуються в просторі введення/виведення розміром 64 байти. У більшості моделей сімейства Mega є також простір додаткових регістрів вводу/виводу розміром 160 або 416 байтів. Введення додаткових РВВ пов'язане з тим, що для підтримки всіх периферійних пристроїв, наявних у цих моделях, звичайних 64-байтових РВВ недостатньо.

Розподіл адрес простору введення/виведення (як основного, так і додаткового) залежить від конкретної моделі мікроконтролера або, якщо точніше, від складу і можливостей периферійних пристроїв даної моделі. Розміщення РВВ у просторі введення/виведення для всіх моделей сімейства наведено в таблиці 3.2.

У таблиці біля адрес РВВ у дужках вказуються відповідні їм адреси комірок ОЗП. Відповідно, якщо адреса регістра вказується

тільки в дужках, цей регістр розташований у просторі додаткових РВВ. Якщо адреса в таблиці не зазначена, це означає, що для даної моделі він зарезервований, і запис за цією адресою заборонений (для сумісності з майбутніми моделями).

Таблиця 3.2 – Регістри введення/виведення моделей Atmega8

Назва	Адреса	Функція
1	2	3
SREG	$3F($5F)	Регістр стану
SPH	$3E($5E)	Покажчик стеку, старший байт
SPL	$3D($5D)	Покажчик стеку, молодший байт
GICR	$3B($5B)	Загальний регістр керування перериваннями
GIFR	$3A($5A)	Загальний регістр прапорців переривань
TIMSK	$39 ($59)	Регістр маски переривань від таймерів/лічильників
TIFR	$38 ($58)	Регістр прапорців переривань від таймерів/лічильників
SPMCR	$37 ($57)	Регістр керування й стану операцій запису в пам’ять програм
TWCR	$36 ($56)	Регістр керування TWI
MCUCR	$35 ($55)	Регістр керування мікроконтролера
MCUCSR	$34 ($54)	Регістр керування й стану мікроконтролера
TCCR0	$33 ($53)	Регістр керування таймера/лічильника ТЕ
TCNT0	$32($52)	Рахунковий регістр таймера/лічильника ТЕ
OSCCAL	$31($51)	Регістр калібрування тактового генератора
SFIOR	$30 ($50)	Регістр спеціальних функцій
TCCR1A	$2F($4F)	Регістр А керування таймера/лічильника Т1
TCCR1B	$2E ($4E)	Регістр В керування таймера/лічильника Т1
TCNT1H	$2D ($4D)	Рахунковий регістр таймера/лічильника Т1, старший байт

Продовження таблиці 3.2

1	2	3
TCNT1L	$2C ($4C)	Рахунковий регістр таймера/лічильника T1, молодший байт
OCR1AH	$2B ($4B)	Регістр A збігу таймера/лічильника T1, старший байт
OCR1AL	$2A ($4A)	Регістр A збігу таймера/лічильника T1, молодший байт
OCR1BH	$29 ($49)	Регістр B збігу таймера/лічильника T1, старший байт
OCR1BL	$28($48)	Регістр B збігу таймера/лічильника T1, молодший байт
ICR1H	$27 ($47)	Регістр захоплювача таймера/лічильника T1, старший байт
ICR1L	$26 ($46)	Регістр захоплювача таймера/лічильника T1, молодший байт
TCCR2	$25 ($45)	Регістр керування таймера/лічильника T2
TCNT2	$24 ($44)	Рахунковий регістр таймера/лічильника T2
OCR2	$23 ($43)	Регістр збігу таймера/лічильника T2
ASSR	$22 ($42)	Регістр стану асинхронного режиму
WDTCR	$21 ($41)	Регістр керування сторожового таймера
UBRRH	$20 ($40)	Регістр швидкості передачі USART, старший байт
UCSRC		Регістр керування й стану USART
EEARH	$1F($3F)	Регістр адреси EEPROM, старший байт
EEARL	$1E($3E)	Регістр адреси EEPROM, молодший байт
EEDR	$1D($3D)	Регістр даних EEPROM
EECR	$1C($3C)	Регістр керування EEPROM
PORTB	$18($38)	Регістр даних порту B
DDRB	$17($37)	Регістр напрямку даних порту B
PINB	$16($36)	Виводи порту B
PORTC	$15($35)	Регістр даних порту C
DDRC	$14 ($34)	Регістр напрямку даних порту C
PINC	$13($33)	Виводи порту C
PORTD	$12($32)	Регістр даних порту D

Продовження таблиці 3.2

1	2	3
DDRD	$11 ($31)	Регістр напрямку даних порту D
PIND	$10($30)	Виводи порту D
SPDR	$0F($2F)	Регістр даних SPI
SPSR	$0E($2E)	Регістр стану SPI
SPCR	$0D($2D)	Регістр керування SPI
UDR	$0C	Регістр даних USART
UCSRA	$0B ($2B)	Регістр А керування і стану USART
UCSRB	$0A ($2A)	Регістр В керування і стану USART
UBRRL	$09 ($29)	Регістр швидкості передачі USART, молодший байт
ACSR	$08 ($28)	Регістр керування і стану аналогового компаратора
ADMUX	$07 ($27)	Регістр керування мультиплексором АЦП
ADCSRA	$06($26)	Регістр А керування і стану АЦП
ADCH	$05 ($25)	Регістр даних АЦП, старший байт
ADCL	$04 ($24)	Регістр даних АЦП, молодший байт
TWDR	$03 ($23)	Регістр даних TWI
TWAR	$02($22)	Регістр адреси TWI
TWSR	$01 ($21)	Регістр стану TWI
TWBR	$00($20)	Регістр швидкості передачі TWI

До регістрів введення/виведення, розташованих в основному просторі введення/виведення, можна прямо звернутися за допомогою команд IN та OUT, що виконують пересилання даних між одним з 32-х регістрів загального призначення (РЗП) і простором введення/виведення. У системі команд є також чотири команди побітового доступу, що використовують у якості операндів регістри введення/виведення: команди установлення/скидання окремого біта (SBI і CBI) і команди перевірки стану окремого біта (SBIS і SBIC). На жаль, ці команди можуть звертатися тільки до першої половини основного простору введення/виведення (адреси $00...$1F).

Крім безпосередньої адресації (за допомогою команд IN і OUT), до РВВ можна звертатися і як до комірок ОЗП за допомогою відповідних команд ST/SD/SDD і LD/LDS/LDD (для додаткових РВВ цей спосіб є єдино можливим). У першому випадку використовуються адреси РВВ, що належать до основного простору введення/виведення ($00...$3F). У другому випадку адресу РВВ необхідно збільшити на $20.

Серед РВВ є один регістр, який використовується найчастіше в процесі виконання програм. Це регістр стану SREG. Він розташований за адресою $3F ($5F) і містить набір прапорців, що показують поточний стан мікроконтролера. Більшість прапорців автоматично встановлюються в 1 або скидаються в 0 при настанні певних подій (відповідно до результату виконання команд). Всі біти цього регістра доступні як для читання, так і для запису; після скидання мікроконтролера всі біти регістра скидаються в 0. Опис прапорців цього регістра наведений у таблиці 3.3.

Таблиця 3.3 – Біти регістра стану SREG

Біт	Назва	Опис
1	2	3
7	I	Загальний дозвіл переривань. Для дозволу переривань цей прапорець повинен бути встановлений в 1. Дозвіл/заборона окремих переривань виробляється установленням або скиданням відповідних бітів регістрів масок переривань (регістрів керування перериваннями). Якщо прапорець скинутий, то переривання заборонені незалежно від стану бітів цих регістрів. Прапорець скидається апаратно після входу в переривання й відновлюється командою RETI для дозволу обробки наступних переривань

Продовження таблиці 3.3

1	2	3
6	T	Зберігання біта. Цей біт регістра використовується як джерело або приймач командами копіювання бітів BLD (Bit Load) і BST (Bit STore). Заданий біт будь-якого РЗП може бути скопійований у цей біт командою BST або встановлений відповідно до вмісту даного біта командою BLD
5	H	Прапорець половинного перенесення. Цей прапорець встановлюється в 1, якщо відбулося перенесення із молодшої половини байта (з 3-го біта в 4-й) або запозичення зі старшої половини байта при виконанні деяких арифметичних операцій
4	S	Прапорець знака. Цей прапорець дорівнює результату операції «виключаюче АБО» (XOR) між прапорцями N (негативний результат) і V (переповнення числа в додатковому коді). Відповідно цей прапорець встановлюється в 1, якщо результат виконання арифметичної операції менше нуля
3	V	Прапорець переповнення додаткового коду. Цей прапорець установлюється в 1 при переповненні розрядної сітки знакового результату. Використовується при роботі зі знаковими числами (представленими в додатковому коді). Докладніше — див. опис системи команд
2	N	Прапорець негативного значення. Цей прапорець установлюється в 1, якщо старший біт (7-й) результату операції дорівнює 1. В іншому випадку прапорець дорівнює 0
1	Z	Прапорець нуля. Цей прапорець встановлюється в 1, якщо результат виконання операції дорівнює нулю
0	C	Прапорець перенесення. Цей прапорець встановлюється в 1, якщо в результаті виконання операції відбувся вихід за межі байта

Переривання

Загальні відомості

Переривання припиняє нормальний хід програми для виконання пріоритетного завдання, обумовленого внутрішньою або зовнішньою подією мікроконтролера. При виникненні переривання мікроконтролер зберігає в стеку вміст лічильника команд PC і завантажує в нього адресу відповідного вектора переривання. За цією адресою, як правило, перебуває команда безумовного переходу до підпрограми обробки переривання. Останньою командою підпрограми обробки переривання повинна бути команда RETI, що здійснює повернення в основну програму й відновлення попередньо збереженого лічильника команд.

Оскільки основними джерелами переривань є різні периферійні пристрої мікроконтролерів, кількість переривань залежить від конкретної моделі.

Таблиця векторів переривань

Мікроконтролери AVR сімейства Mega мають багаторівневу систему пріоритетних переривань. Молодші адреси пам'яті програм, починаючи з адреси $0001, відведені під таблицю векторів переривання. Кожному перериванню відповідає адреса в цій таблиці, що завантажується в лічильник команд при виникненні переривання (таблиця 3.4). Положення вектора в таблиці також визначає й пріоритет відповідного переривання: чим менше адреса, тим вище пріоритет переривання. Розмір вектора переривання залежить від об’єму пам'яті програм мікроконтролера й становить 2 байти. Відповідно для переходу до підпрограм обробки переривань використовуються команди RJMP.

Практично у всіх мікроконтролерах сімейства Mega, за винятком моделі Atmega48x, положення таблиці векторів переривань може бути змінено. Таблиця може розташовуватися не тільки на початку пам'яті програм, причому переміщення таблиці може бути здійснене безпосередньо в ході виконання програми.

Таблиця 3.4 – Таблиця векторів переривань моделі ATmega8

Джерело	Опис	Номер	Адреса
INT0	Зовнішнє переривання 0	1	$0001
INT1	Зовнішнє переривання 1	2	$0002
TIMER2COMP	Збіг таймера/лічильника T2	3	$0003
TIMER2OVF	Переповнення таймера/лічильника T2	4	$0004
TIMER1 CAPT	Захоплювач таймера/лічильника T1	5	$0005
TIMER1 COMPA	Збіг A таймера/лічильника T1	6	$0006
TIMER1 COMPB	Збіг B таймера/лічильника T1	7	$0007
TIMER1OVF	Переповнення таймера/лічильника T1	8	$0008
TIMER0OVF	Переповнення таймера/лічильника T0	9	$0009
SPI, STC	Передача по SPI завершена	10	$000A
USART, RXC	USART, приймання завершено	11	$000У
USART, UDRE	Регістр даних USART порожній	12	$000C
USART, TXC	USART, передача завершена	13	$000D
ADC	Перетворення АЦП завершено	14	$000E
EE_RDY	EEPROM готово	15	$000F
ANA_COMP	Аналоговий компаратор	16	$0010
TWI	Переривання від модуля TWI	17	$0011
SPM_RDY	Готовність SPM	18	$0012

Обробка переривань

Для глобального дозволу/заборони переривань призначений прапорець I регістра SREG. Для дозволу переривань він повинен бути встановлений в 1, а для заборони – скинутий у 0. Індивідуальний дозвіл або заборона переривань виробляється

установленням/скиданням відповідних бітів регістрів масок переривань, розглянутих нижче.

При виникненні переривання прапорець I регістра SREG апаратно скидається, забороняючи тим самим обробку наступних переривань. Однак у підпрограмі обробки переривання цей прапорець можна знову встановити в 1 для дозволу вкладених переривань. При поверненні з підпрограми обробки переривання (при виконанні команди RETI) прапорець I установлюється апаратно.

Всі наявні переривання можна поділити на два типи. Переривання першого типу генеруються при настанні деякої події, у результаті якої встановлюється прапорець переривання. Потім, якщо переривання дозволено, у лічильник команд завантажується адреса вектора відповідного переривання. При цьому прапорець переривання апаратно скидається. Він також може бути скинутий програмно, записом 1 у біт регістра, що відповідає прапорцю.

Переривання другого типу не мають прапорців переривань і генеруються протягом усього часу, поки присутні умови, необхідні для генерації переривання. Відповідно, якщо умови, що викликають переривання, зникнуть до дозволу переривання, генерації переривання не відбудеться.

Варто пам'ятати, що при виклику підпрограм обробки переривань регістр стану SREG не зберігається. Тому користувач повинен самостійно запам'ятовувати вміст цього регістра при вході в підпрограму обробки переривання (якщо це необхідно) і відновлювати його значення перед викликом команди RETI.

Мікроконтролери сімейства Mega підтримують чергу переривань, що працює в такий спосіб: якщо умови генерації одного або більше переривань виникають у той час, коли прапорець загального дозволу переривань скинутий (всі переривання заборонені), відповідні прапорці переривань встановлюються в 1 і залишаються в цьому стані до встановлення прапорця загального дозволу переривань. Після дозволу переривань виконується їхня обробка в порядку пріоритету.

Найменший час відгуку для будь-якого переривання Atmega8 становить 4 такти. Протягом цього часу відбувається збереження лічильника команд у стеку. Протягом наступних двох

тактів виконується команда переходу до підпрограми обробки переривання. Якщо переривання відбудеться під час виконання команди, що триває кілька циклів, то генерація переривання відбудеться тільки після виконання цієї команди. Якщо ж переривання відбудеться під час знаходження мікроконтролера в «сплячому» режимі, то час відгуку збільшується ще на 4 або 5 тактів.

Повернення в основну програму займає 4 такти, протягом яких відбувається відновлення лічильника команд зі стеку. Після виходу з переривання процесор завжди виконує одну команду основної програми, перш ніж обслужити будь-яке відкладене переривання.

Регістри портів введення/виведення

Звертання до портів проводиться через регістри введення/виведення. Під кожен порт в адресному просторі введення/виведення зарезервовано по 3 адреси, за якими розміщені такі регістри: регістр даних порту PORTx, регістр напрямку даних DDRx і регістр виводів порту PINx. Дійсні назви регістрів отримуються підстановкою назви порту замість символу x. Відповідно, регістри порту A називаються PORTA, DDRA, PINA, порту B – PORTB, DDRB, PINB і т. д. Оскільки за допомогою регістрів PINx здійснюється доступ до фізичних значень сигналів на виводах порту, вони доступні тільки для читання.

8-бітні таймери/лічильники

8-бітний таймер/лічильник T0 присутній у всіх моделях мікроконтролерів сімейства Mega, а таймер/лічильник T2 – в усіх, крім ATmega8515x. Усього в мікроконтролерах сімейства реалізовано п'ять виконань 8-бітних таймерів/лічильників, що відрізняються набором виконуваних функцій. Найпростішим є таймер/лічильник T0 у моделі Atmega8. Він може використовуватися тільки для відліку часових інтервалів або як лічильник зовнішніх подій. Таймер/лічильник T2 відрізняється тим, що може працювати в асинхронному режимі (звичайно цей

режим використовується для реалізації годинника реального часу).

Рахунковий регістр таймера/лічильника TCNTn входить до складу основного блока модуля – блока реверсивного лічильника. Залежно від режиму роботи модуля вміст рахункового регістра скидається, збільшується або зменшується по кожному імпульсу тактового сигналу таймера/лічильника clk0 (clk2). Незалежно від того, присутній тактовий сигнал чи ні, регістр доступний у будь-який момент часу як для читання, так і для запису. Однак варто пам'ятати, що будь-яка операція запису в рахунковий регістр блокує роботу блока порівняння на час одного періоду тактового сигналу таймера/лічильника. Після подачі напруги живлення в регістрі TCNTn перебуває нульове значення. При досягненні таймером/лічильником максимального або мінімального значення (конкретний варіант залежить від його режиму роботи) встановлюється прапорець TOVn у регістрі прапорців TIFR (TIFRn). Дозвіл переривання здійснюється установленням в 1 біта TOIEn регістра маски TIMSK (TIMSKn). Зрозуміло, прапорець I регістра SREG також повинен бути встановлений в 1.

Регістри порівняння OCRn (OCRn/OCRn) входять до складу блоків порівняння модуля. Під час роботи таймера/лічильника проводится безперервне (у кожному такті) порівняння цих регістрів з регістром TCNTn. У випадку рівності вмісту цих регістрів у наступному такті встановлюється прапорець OCFn (OCFn/OCFn) у відповідному регістрі прапорців і генерується переривання (якщо воно дозволено). Крім того, при настанні цієї події може змінюватися стан виводу OCn (OCnA/OCnB) мікроконтролера. Щоб таймер/лічильник мав можливість керувати станом цих виводів, вони повинні бути сконфігуровані як виходи (відповідний біт регістра DDRx повинен бути встановлений в 1).

Будь-яка операція запису в рахунковий регістр блокує формування сигналу про збіг, якщо воно відбудеться в наступному такті.

3.4 Порядок виконання роботи

У прикладі 2 наведено фрагмент програми керування двома світлодіодами за допомогою кнопок з програмуванням окремих бітів портів D та E мікроконтролера **Atmega128,** що використаний за схемою на рисунку 3.5.

Приклад 2

```
; Приклад «Керування світлодіодами від кнопок»
; написаний для налагоджувальної плати AS-MegaM
; світлодіоди підключені до виводів PD6 і PD7 через
резистори – на загальний
; провід. Кнопки – на PE4 і PE5
.include "m128def.inc"
; основна програма
begin:
; ініціалізація стеку
ldi r16,low(RAMEND)
out spl,r16
ldi r16,high(RAMEND)
out sph,r16
; ініціалізація світлодіодів
ldi r16,(1<<6) | (1<<7)
out DDRD,r16
; ініціалізація виводів, до яких підключені кнопки (на вхід)
; внутрішні підтягувальні резистори підключені
; для цього в PORTE потрібно встановити відповідні біти в
одиниці
ldi r16,(1<<4) | (1<<5)
out PORTE,r16
; а в DDRE – у нулі, щоб світлодіоди в початковому положенні
були вимкнені
ldi r16,0
out DDRE,r16
; нескінченний цикл
forever:
in r16,PINE ; тепер в r16 знаходиться поточний "стан" кнопок
com r16 ; кнопка "натискається" нулем, тому інвертуємо регістр
```

```
lsl r16
lsl r16
; переносимо біти 4, 5 у позиції 6, 7 шляхом подвійного зсуву вліво по
; розрядах регістра
andi r16,(1<<6) | (1<<7) ; і оновлюємо «світіння» світлодіодів при відпусканні кнопки
out PORTD,r16
rjmp forever ; цикл виконується нескінченно
```

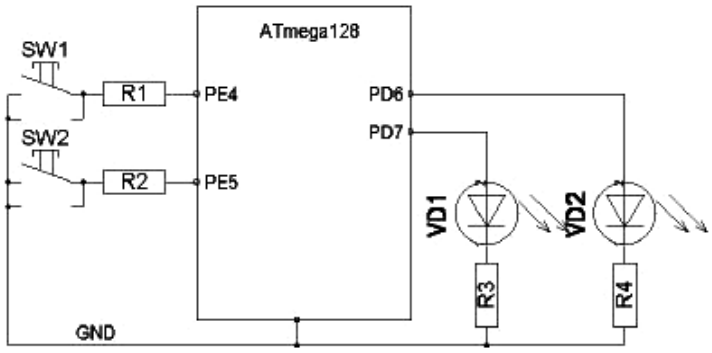

Рисунок 3.5 – Принципова схема

На рисунку 3.5 взято мікросхему Atmega128 і підключено два світлодіоди до виводів 31 і 32 (це біти 6 і 7 порту D).

AVR-контролери мають потужні вихідні каскади, типовий струм кожного виводу складає 20 мА, максимальний струм виводу – 40 мА, причому це стосується як вхідного, так і вихідного струму.

У нашому прикладі світлодіоди підключені анодами до виводів мікроконтролера, а катоди через гасильні резистори сполучені з землею. Це означає, що світлодіод запалюється подачею **1** на відповідний вивід порту.

1 Створити новий проект в AVR Studio, при цьому вибрати як налагоджувальну платформу ***AVR Simulator*** і мікросхему **ATtiny2313**.

2 Для мікроконтролера **ATtiny2313** для схеми на рисунку 3.5 написати програму вмикання декількох світлодіодів за допомогою декількох кнопок згідно з варіантом, заданим викладачем. Дивись приклад 2.

3 Виконати налагодження програми та перевірку її роботи разом із зібраною схемою в середовищі **PROTEUS**.

4 Створити другий проект в AVR Studio, при цьому вибрати як налагоджувальну платформу ***AVR Simulator*** і мікросхему **ATtiny2313.**

5 Для мікроконтролера **ATtiny2313** написати програму аналогічну попередній згідно з варіантом, заданим викладачем.

6 Виконати налагодження програми та перевірку її роботи разом із зібраною схемою в середовищі **PROTEUS**.

7 Створити новий проект в AVR Studio, при цьому вибрати як налагоджувальну платформу ***AVR Simulator*** і мікросхему **Atmega128**.

8 Для мікроконтролера **Atmega128** для схеми на рисунку 3.3 написати програму вмикання декількох світлодіодів за допомогою декількох кнопок згідно з варіантом, заданим викладачем. Дивись приклад 2.

9 Виконати налагодження програми та перевірку її роботи разом із зібраною схемою в середовищі **PROTEUS**.

Таким чином, на прикладі простих програм показано 3 варіанти програмування портів і керування світлодіодами за допомогою кнопок.

3.6 Зміст звіту

Звіт повинен містити:

1 Тексти трьох написаних програм.

2 Зібрані в PROTEUS 3 схеми.

3 До кожної програми блок-схему (flow chart) алгоритму.

4 Перелік використаних команд з поясненням їхнього призначення.

5 Висновки про обсяг виконаної роботи, досягнену мету роботи, співпадіння чи неспівпадіння практичних результатів з теоретичними.

Контрольні тестові питання

1 Як ініціалізувати порт вхідним або вихідним?

2 На які сегменти поділяється пам’ять мікроконтролерів AVR?

3 За допомогою яких команд обирається той чи інший сегмент памяті?

4 Як називається адреса вершини стеку памяті програм?

5 Як записати в 3-й біт порту D одиницю з використанням шістнадцяткової та двійкової системи числення?

6 Як задати частоту тактування МК при емуляції?

7 За допомогою яких команд провести запис/зчитування даних у порт E?

8 Як визначити назву файлу опису мікроконтролера?

9 Що, 1 чи 0, потрібно подати на вихід порту PB.0 для того, щоб засвітився приєднаний світлодіод для схем на рисунках 3.1, 3.2 та 3.3.

10 Пояснити принцип роботи схем на рисунках 3.1, 3.2 та 3.3.

11 Пояснити алгоритм роботи написаних програм до рисунків 3.1, 3.2 та 3.3.

12 Пояснити суть і призначення використаних при виконанні лабораторної роботи команд.

ЛАБОРАТОРНА РОБОТА 4
Дослідження методики використання програмної затримки

4.1 Навчальні питання

1 Дослідження способів використання програмної затримки.

2 Дослідження методики написання програми «миготіння світлодіодів» із заданою черговістю і тривалістю світіння з використанням програмної затримки.

4.2 Навчальна мета

Практичне дослідження методики створення та налагодження програм з використанням програмної затримки за допомогою програми AVR Studio.

4.3 Теоретичні відомості

Для прикладу візьмемо мікросхему Atmega128 і підключимо два світлодіоди до виводів 31 і 32 (це біти 6 і 7 порту D мікросхеми Atmega128).

AVR-контролери мають потужні вихідні каскади, типовий струм кожного виводу складає 20 мА, максимальний струм виводу – 40 мА, причому це стосується як вхідного, так і вихідного струму. У нашому прикладі світлодіоди підключені анодами до виводів контролера, а катоди через гасильні резистори сполучені з землею. Це означає, що світлодіод запалюється подачею 1 на відповідне виведення порту.

Принципова схема наведена на рисунку 4.1. На схемі також показано дві кнопки, які будуть використані в одній з програм.

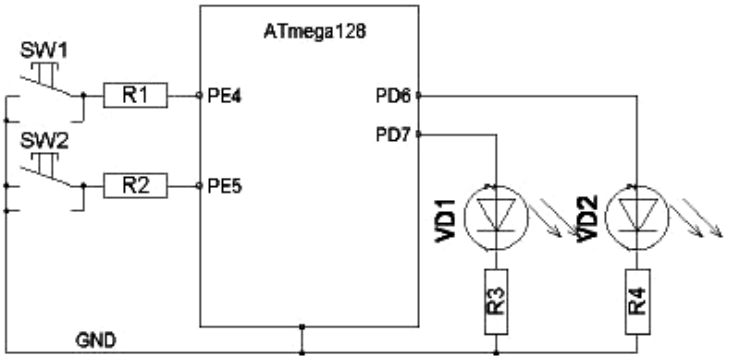

Рисунок 4.1 – Принципова схема

Тут доречно зробити невеликий відступ про вибір типу мікросхеми для простого прикладу. Дійсно, з першого погляду може здатися дивним, навіщо потрібен такий потужний кристал у 64-вивідному корпусі там, де вистачить і 8-вивідної мікросхеми ATtiny12? Проте в такому підході є логіка. Відомо, що в основі практично будь-якого AVR-контролера лежить однакове ядро. За великим рахунком, контролери розрізняються об'ємом пам'яті, кількістю портів введення/виведення і набором периферійних модулів. Особливості кожного конкретного контролера – прив'язка логічних імен регістрів введення/виведення до фізичних адрес, адреси векторів переривань, визначення бітів портів і т. д., які описані у файлах з розширенням ***.inc***, що входять до складу пакета AVR Studio. Отже, використовуючи конкретний

тип кристала, можна налагоджувати програму як власне для нього, так і для будь-якого молодшого кристала. Далі, якщо використовувати як налагоджувальний найстарший кристал, можна налагоджувати програму практично для будь-якого AVR-контролера, треба просто не використовувати апаратні ресурси, відсутні в цільового мікроконтролера. Таким чином, наприклад, можна настроювати на Atmega128 програму, яка виконуватиметься на ATtiny13. При цьому початковий код залишиться практично тим самим, зміниться лише ім'я файлу, що підключається, з ***128def.inc*** на ***tn13def.inc***.

У такого підходу також є свої переваги. Наприклад, «зайві» порти введення/виведення можна використовувати для підключення РК-індикатора, на який можна виводити налагоджувальну інформацію, або скористатися внутрішньосхемним емулятором, який підключається до JTAG-порту мікросхеми Atmega128 (контролер ATtiny13 такий порт не має). Таким чином, можна використовувати єдину налагоджувальну плату, на якій встановлений «старший» AVR-контролер, для налагодження будь-яких розроблюваних систем, що базуються також на AVR-мікроконтролерах. Одна з таких плат називається AS-megaM. Саме вона використовувалася для створення прикладів програм, що наводяться в даній роботі. Це універсальний одноплатний контролер на базі мікросхеми Atmega128, який містить зовнішнє ОЗУ, два порти RS-232, порт для підключення РК-індикатора, внутрішньосхемного програматора і емулятора AT JTAG ICE. На платі також є місце для розпаювання мікросхеми FLASH-ПЗП серії AT45 в корпусах TSOP32/40/48 і двоканального ЦАП серії AD5302/ AD5312/ AD5322. Тепер після пояснення причин використання AVR-мікроконтролера для запалення пари світлодіодів (приклад 3).

Приклад 3

```
; Приклад «Керування світлодіодами»
; написаний для налагоджувальної плати AS-MegaM
; Частота задавального генератора 7,37 Мгц
; світлодіоди підключені до виводів PD6 і PD7 і через резистори
- на загальний дріт.
```

```
; підключення файлу опису введення-виведення мікросхеми Atmega128
.include "m128def.inc"
; почало програми
begin:
; перша операція - ініціалізація стеку
; якщо цього не зробити, то виклик підпрограми або переривання
; не поверне керування назад
; покажчик на кінець стеку встановлюється на останню адресу внутрішнього ОЗУ - RAMEND
ldi r16,low(RAMEND)
out spl,r16
ldi r16,high(RAMEND)
out sph,r16
; для того щоб керувати світлодіодами, підключеними до виводів PD6 і PD7,
; необхідно оголосити ці виводи вихідними.
; для цього потрібно записати "1" у відповідні біти регістра DDRD (DataDiRection)
ldi r16,(1<<6) | (1<<7)
out DDRD,r16
; основний цикл програми
loop:
ldi r16,(1<<6) ; світиться один світлодіод
out PORTD,r16
rcall delay ; затримка
ldi r16,(1<<7) ; світиться другий світлодіод
out PORTD,r16
rcall delay ; затримка
rjmp loop ; повторення циклу
; процедура затримки, приблизно півсекунди при частоті 7,37 МГц
; три порожні вкладені цикли відповідно
delay:
ldi r16,30 ; 30
delay1:
ldi r17,200 ; 200
```

```
delay2:
ldi r18,200 ; і ще 200 ітерацій
delay3:
dec r18
brne delay3
dec r17
brne delay2
dec r16
brne delay1
ret ; повернення в головну програму
```

4.4 Порядок виконання роботи

1 Створити новий проект в AVR Studio, при цьому вибрати як налагоджувальну платформу ***AVR Simulator*** і мікросхему **Atmega128**.

2 Для мікроконтролера Atmega128 на схемі рисунка 4.1 написати програму послідовного вмикання лампочок (світлодіодів) із заданою тривалістю і черговістю світіння (дивися приклад 1).

3 Виконати налагодження програму.

4 Створити проект із заданою схемою в середовищі **PROTEUS** і підключити написану програму **Atmega128.** Запустити симуляцію, перевірити працездатність схеми.

5 Написати програму, яка відрізняється від попередньої тим, що цикл миготіння повинен повторювати задану згідно з варіантом кількість разів і з використанням шістнадцяткової системи зчислення для програмування бітів портів.

6 Виконати налагодження програми.

4.5 Зміст звіту

Звіт повинен містити:

1 Тексти написаних програм.

2 Зібрані в PROTEUS схеми.

3 До кожної програми блок-схему (flow chart) алгоритмів.

4 Перелік використаних команд з поясненням їхнього призначення.

5 Висновки про обсяг виконаної роботи, досягнену мету роботи, співпадіння чи неспівпадіння практичних результів з теоретичними.

Контрольні тестові питання

1 Які мікроконтролери підтримує емулятор електронних пристроїв PROTEUS?

2 Що відображує панель **DEVICES?**

3 Як виконати подачу живлення або певного сигналу в довільну точку схеми?

4 Який вигляд мають файли прошивки, виконані мовами С, Асемблер та в машинних кодах?

5 Які дії необхідно провести, щоб проемулювати в PROTEUS роботу мікроконтролера?

6 Як задати частоту тактування МК при емуляції?

7 Яке призначення має віртуальний термінал?

8 Як задати необхідний час емуляції?

СПИСОК ЛІТЕРАТУРИ

1 Цифровая и вычислительная техника [Текст] : учеб. для вузов / Э. В. Евреинов, Ю. Т. Бутыльский, И. А. Мамзелев [и др.]; под ред. Э. В. Евреинова. – М. : Радио и связь, 1991.

2 Угрюмов, Е. П. Проектирование элементов и узлов ЭВМ [Текст] : учеб. пособие для спец. ЭВМ вузов / Е. П. Угрюмов. – М. : Высш. шк., 1987.

3 Мікропроцесорна техніка [Текст] : підручник / Ю. І. Якименко, Т. О. Терещенко, Є. І. Сокол [та ін.]; за ред. Т. О. Терещенко. – 2-ге вид., перероб. та доп. – К. : ІВЦ «Політехніка»: «Кондор», 2008. – 594 с.

4 Схемотехніка електронних систем [Текст] : підручник; у 3 кн. Кн. 2. Цифрова схемотехніка / В. І. Бойко, А. М. Гурій, В. Я. Жуйков [та ін.]. – 2-ге вид., доп. і перероб. – К. : Вища шк., 2004. – 423 с.

5 Схемотехніка електронних систем [Текст] : підручник; у 3 кн. Кн. 3. Мікропроцесори та мікроконтролери / В. І. Бойко,

А. М. Гурій, В. Я. Жуйков [та ін.]. – 2-ге вид., доп. і перероб. – К. : Вища шк., 2004. – 399 с.

6 Материалы технического семинара AVR Technical Training. Atmel. Norway. December 2003.

7 Королев, Н. AVR-микроконтроллеры второго поколения: средства разработчика [Текст] / Н. Королев, Д. Королев // Компоненты и технологии. – 2003. – № 7.

8 Королев, Н. AVR-микроконтроллеры второго поколения: новые аппаратные возможности [Текст] / Н. Королев, Д. Королев // Компоненты и технологии. – 2003. – № 4.

9 Королев, Н. AVR-микроконтроллеры: большое в малом [Текст] / Н. Королев, Д. Королев // Схемотехника. – 2001. – № 5.

10 Королев, Н. AVR-микроконтроллеры: программные средства [Текст] / Н. Королев, Д. Королев // Компоненты и технологии. – 2000. – № 4.

11 Королев, Н. HAVR: аппаратные средства разработчика [Текст] / Н. Королев // Компоненты и технологии. – 1999. – № 1.

12 Королев, Н RISC-микроконтроллеры фирмы ATMEL [Текст] / Н. Королев // Chip-News. – 1998. – № 2.

13 Королев, Н. AVR: новые 8-разрядные RISC-микроконтроллеры фирмы ATMEL [Текст] / Н. Королев, Д. Королев // Микропроцессор Ревю. – 1998. – № 1.

14 ATmega8 – оригінальна документація на мікроконтролер від виробника.

15 Evstifeev. MC_AVR_Mega_2007 – загальний опис мікроконтролерів.

16 Hartov. MC_AVR_Praktikum_dlia_nachinaiushih_2007.