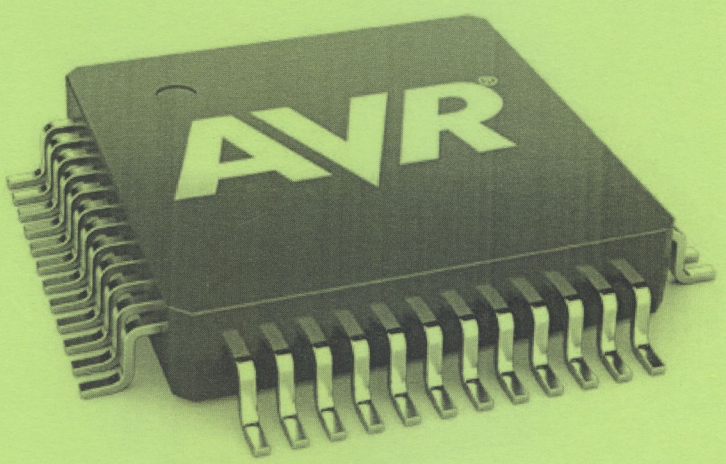
Ю. В. Шевчук, Д. П. Проценко, С. М. Бабій

ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ КУРСОВЕ ПРОЕКТУВАННЯ САМОСТІЙНА ТА ІНДИВІДУАЛЬНА РОБОТА СТУДЕНТІВ

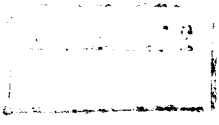


Міністерство освіти і науки України
Вінницький національний технічний університет

Ю. В. Шевчук, Д. П. Проценко, С. М. Бабій

ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ КУРСОВЕ ПРОЕКТУВАННЯ САМОСТІЙНА ТА ІНДИВІДУАЛЬНА РОБОТА СТУДЕНТІВ

Навчальний посібник



Вінниця
ВНТУ
2017

УДК 004(075)
ББК 32.973.2я73
Ш37

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 7 від 22 грудня 2016 р.)

Рецензенти:
П. К. Ніколюк, доктор фізико-математичних наук, професор
В. В. Кулик, доктор технічних наук, професор
Ю. В. Булига, кандидат технічних наук, доцент

Шевчук, Ю. В.

Ш37 Обчислювальна техніка та програмування. Курсове проектування. Самостійна та індивідуальна робота студентів : навчальний посібник / Ю. В. Шевчук, Д. П. Проценко, С. М. Бабій. – Вінниця : ВНТУ, 2017. – 63 с.

Посібник призначений для організації виконання курсової роботи з дисципліни «Обчислювальна техніка та програмування» і буде корисним студентам електротехнічних спеціальностей.

УДК 004(075)
ББК 31.973.2я73

ЗМІСТ

ПЕРЕДМОВА

Мікропроцесорні системи становлять основу більшості сучасних технологічних процесів, які використовуються в промисловості, сільському господарстві, на транспорті, в комунальному господарстві – у всіх сферах людської діяльності. Більшість виробничих механізмів функціонують за рахунок мікропроцесорних систем.

Мікропроцесорна система являє собою функціонально закінчений виріб, який складається із одного або декількох пристроїв, головним чином, мікропроцесорних.

Основні переваги мікропроцесорних систем порівняно з цифровими системами на «жорсткій логіці».

– багатофункціональність: більшу кількість функцій може бути реалізовано на одній елементній базі;

– гнучкість: можливість виправлення і зміни файлів мікропроцесора для реалізації різних режимів роботи системи;

– компактність: мініатюрні габарити мікросхем і зменшення їх кількості порівняно з реалізацією на «жорсткій логіці» дозволяють зменшити габарити пристроїв;

– підвищення завадостійкості: менша кількість з'єднувальних провідників сприяє підвищенню надійності пристроїв;

– продуктивність: можливість застосування великих робочих частот і більш складних алгоритмів обробки інформації;

– захист інформації: можливість захистити програму мікропроцесора від зчитування дозволяє захистити авторські права розробників.

Мікропроцесорний пристрій являє собою функціонально і конструктивно закінчений виріб, який складається із декількох мікросхем, до складу яких входить мікропроцесор або мікроконтролер; призначений він для виконання певного набору функцій.

Мікропроцесором називається програмно-апаратний пристрій, який здійснює процес обробки цифрової інформації. Головна особливість – можливість програмування логіки роботи.

Мікроконтролером називається мікросхема, призначена для керування електронними пристроями. Типовий мікроконтролер містить в собі функції мікропроцесора, периферійних пристроїв та пам'яті.

До складу структури мікроконтролера можуть входити такі додаткові периферійні пристрої:

– різні інтерфейси введення-виведення, такі як UART, I²C, SPI, CAN, USB, ETHERNET;

– аналого-цифрові і цифро-аналогові перетворювачі;

– компаратори;

– широтно-імпульсні модулятори;

– таймери-лічильники;

– генератор тактової частоти;

– контролери дисплеїв і клавіатур;

– масиви вбудованої флеш-пам'яті.

В навчальному посібнику розглянуто питання розробки та імплементації логіки роботи мікропроцесорного пристрою. Навчальний посібник призначений для організації самостійної роботи та допомоги студентам напряму підготовки «Електромеханіка» денної та заочної форм навчання при виконанні курсової роботи з дисципліни «Обчислювальна техніка та програмування» та буде корисним при виконанні частини мікропроцесорної реалізації в бакалаврських роботах, дипломних проектах та кваліфікаційних магістерських роботах. Навчальний посібник може бути використаний студентами електротехнічних спеціальностей при вивченні дисциплін «Автоматизований електропривод типових виробничих механізмів», «АСК технологічними процесами та комплексами», «Застосування інформаційних технологій в електромеханіці», «Діагностичні комплекси в електромеханічних системах».

Метою курсової роботи є:

– систематизація і закріплення знань з дисципліни «Обчислювальна техніка та програмування»;

– аналіз та обґрунтування вибраної програмно-апаратної реалізації;

– закріплення навичок побудови алгоритмів для мікропроцесорних пристроїв;

– розробка схем електричної структурної та принципової мікропроцесорного пристрою;

– засвоєння навичок реалізації заданих алгоритмів роботи мікропроцесорних пристроїв із використанням мов програмування Assembler та C;

– перевірка проектних рішень методами комп'ютерного моделювання або за допомогою фізичної реалізації.

Передбачені варіанти завдань охоплюють типові реалізації, що дозволяє набути практичних навичок розробки програм мікропроцесорних пристроїв із застосуванням мікроконтролерів AVR, основаних на сучасній RISC архітектурі, використовуючи мови програмування низького та високого рівнів.

В 1996 році корпорація Atmel представила своє сімейство чипів на новому прогресивному ядрі AVR. Більш продумана архітектура AVR, швидкодія, більш розвинена система команд, що налічує до 133 інструкцій, продуктивність, Flash ПЗУ програм з можливістю перепрограмування в схемі пристрою. Багато чипів мають функцію самопрограмування. AVR-архітектура оптимізована під мову високого рівня C. Величезну роль у широкому розповсюдженні мікроконролерів Atmel мала доступність програмного забезпечення і засобів підтримки розробки (в т. ч. безкоштовного).

1 ОРГАНІЗАЦІЯ РОБОТИ НАД КУРСОВОЮ РОБОТОЮ

Курсова робота є самостійною навчально-науковою роботою студента і має відображати рівень отриманих теоретичних та практичних навичок із загальних і спеціальних розділів навчальної дисципліни «Обчислювальна техніка та програмування» При написанні курсової роботи студент повинен показати навички роботи з спеціалізованою літературою, вміння критично аналізувати різні варіанти алгоритмічних та програмних рішень побудови мікропроцесорних пристроїв, робити обґрунтований вибір апаратних та програмних засобів, необхідних для розв'язання поставленої задачі.

Робота над обраною темою потребує від студента знань основ методології дослідження та пошуку інформації, творчого мислення, логіки аргументації, старанності і професіоналізму.

Завдання на курсову роботу (додаток А) видається студенту в терміни, визначені графіком навчального процесу. Дата видачі вноситься в індивідуальне завдання і вважається початком курсового проектування.

Виконання курсової роботи починається зі складання робочого плану, який дозволить чітко організувати роботу студента.

Робочий план складається в довільній формі і зазвичай його використовують напочатку написання курсової роботи. Такий план дозволяє ескізно подати задачу проектування та етапи її розв'язання. Рекомендується оформити робочий план у вигляді переліку основних етапів роботи та їхнього розширеного опису. Всі етапи роботи повинні бути пов'язані внутрішньою логікою відповідно до теми.

На більш пізніх етапах роботи доцільно скласти план-проспект, відповідно до якого в подальшому буде систематизуватися увесь зібраний матеріал. Це один з найбільш складних і трудомістких етапів в написанні курсової роботи. Студенту потрібно ознайомитися зі значним обсягом технічної документації, навчальної літератури, науково-технічних статей, в яких знайшли відображення різні підходи щодо реалізації запропонованих завдань курсової роботи.

Необхідність повного розкриття теми потребує від студента активного використання не тільки друкованих джерел інформації, а і пошуку та систематизації інформації з різноманітних електронних ресурсів мережі Internet, які стосуються теми курсової роботи.

Огляд джерел інформації слід починати з ознайомлення зі змістом тих глав і розділів підручників та навчальних посібників, в яких викладено матеріал про загальні принципи побудови мікропроцесорних пристроїв та основ програмування. Після ознайомлення з загальною інформацією варто сфокусуватися на прикладах, які відповідають темі курсової роботи. На цьому етапі слід виділити ті елементи програми, які виконують спільні функції, а також програмні алгоритми, які потрібно додатково реалізувати.

Необхідно пам'ятати, що підручники і навчальні посібники, як прави-

ло, відображають усталені, класичні методи розв'язання задач програмування та алгоритмізації, тому для пошуку нестандартних рішень потрібно скористатись інформацією з науково-технічних періодичних видань.

Враховуючи сучасний стан розвитку інформаційних технологій, обов'язковим є використання інформації, розміщеної в мережі Internet. При пошуку та систематизації інформації, знайденої в мережі Internet, слід точно фіксувати електронні адреси сайтів та дату звернення.

Індивідуальним завданням передбачено вирішення основних питань розробки мікропроцесорного пристрою. Важливо відзначити те, що розуміння завдання в подальшому дозволяє значно спростити складання алгоритмів та програми функціонування пристрою, тому після отримання завдання на курсову роботу потрібно детально його описати на вербальному рівні у вигляді послідовності операцій, які виконує мікроконтролер. Такий запис дозволить спростити розробку блок-схеми алгоритму та написання програми.

Пояснювальна записка курсової роботи оформляється відповідно до правил Єдиної системи конструкторської документації (ЄСКД). Всі технічні рішення повинні бути обґрунтовані. Кожен розділ пояснювальної записки повинен закінчуватись лаконічними висновками.

Закінчена курсова робота, після перевірки її викладачем-керівником і виправлення вказаних недоліків, захищається перед комісією, яка складається з викладачів кафедри. Захист робіт проводиться відповідно до попередньо оголошеного розкладу.

Під час захисту кожний студент протягом 3-5 хвилин доповідає про зміст роботи, звертаючи основну увагу на постановку задачі, основні положення роботи, результати досліджень і висновки. У доповіді необхідно підкреслити особистий внесок у розробку тих чи інших питань. Доповідь має супроводжуватися демонстрацією відповідних презентаційних матеріалів, що засвідчують факт виконання студентом поставлених задач і дають можливість присутнім ознайомитися з основними результатами роботи. Після доповіді відбувається обговорення, в рамках котрого кожен із присутніх може поставити студенту запитання з тематики його роботи, а студент має дати чітку й обґрунтовану відповідь

При оцінюванні студента комісією береться до уваги:

– обґрунтованість ухвалених в роботі рішень;

– глибина опрацювання основних питань;

– якість оформлення пояснювальної записки;

– якість, правильність і повнота відповідей на запитання, задані членами комісії в процесі захисту роботи.

Підсумкова оцінка є інтегрованим результатом викладених вище вимог.

2 ЗМІСТ І ОБСЯГ КУРСОВОЇ РОБОТИ

Структура курсової роботи:
– титульний аркуш;
– індивідуальне завдання;
– анотація;
– зміст;
– вступ;
– основна частина;
– висновки;
– перелік посилань;
– додатки.

До складу основної частини роботи входять такі розділи:

1. Аналіз алгоритмічних рішень поставленої задачі;
2. Розробка схем електричної структурної та електричної принципової;
3. Розробка алгоритму роботи пристрою;
4. Розробка програми мовою програмування Assembler;
5. Розробка програми мовою програмування С;
6. Моделювання роботи пристрою.

Кожен розділ курсової роботи повинен закінчуватися лаконічними висновками, у яких проаналізовано отримані результати. Описовий стиль викладення матеріалів висновків є недопустимим.

Обсяг пояснювальної записки курсової роботи становить 25...30 сторінок формату А4. Текст пояснювальної записки виконується кеглем №14.

2.1 Аналіз алгоритмічних рішень поставленої задачі

У розділі має бути проведений аналіз пристроїв, принцип роботи яких корелюється із функціоналом відповідного пристрою відповідно до завдання, при цьому матеріали рекомендовано подавати в такій формі:

– короткий опис роботи пристрою;
– електрична або функціональна схема мікропроцесорного пристрою, який було взято для аналізу;
– алгоритм роботи пристрою та його опис;
– аналіз особливостей програмних рішень, середовищ розробки, моделювання та налагоджування.

Особливу увагу варто приділити аналізу практичного застосування розглянутих алгоритмів в електромеханічних системах автоматизації. Можуть бути розглянуті особливості роботи окремих складових елементів, принципи побудови схем та алгоритмів.

2.2 Розробка схеми електричної структурної та принципової

Схема структурна – схема, що призначена для відображення загальної структури пристрою, тобто його основних блоків, вузлів, частин та головних зв'язків між ними. Із структурної схеми повинно бути зрозуміло, навіщо потрібний даний пристрій і як він працює в основних режимах роботи, як взаємодіють його частини. Позначення елементів на структурній схемі можуть обиратись довільно.

Схема принципова – схема, що визначає повний склад елементів виробу, спільна дія яких забезпечує вирішення задач керування, регулювання, захисту, вимірювання і сигналізації, а також зв'язків між ними і, як правило, дає детальне уявлення про принципи роботи виробу.

Розробка структурних та принципових електричних схем завжди містить елементи творчості і потребує умілого застосування елементарних електричних кіл і типових функціональних вузлів, оптимального компонування їх в єдину схему з урахуванням вимог, що висуваються до схем, а також можливого спрощення і мінімізації схем.

Розглянемо приклад розробки структурної та принципової схем для мікропроцесорного пристрою, що реалізує такий функціонал: *програмний формувач аналогових сигналів складної форми, що генерує сигнал в часі, показаний на рис. 2.1, де* $t_1 = 50$ *мс,* $t_2 = 30$ *мс,* $t_п = 20$*мс,* $U_{max} = 10$ *В. Аналоговий сингал потрібно отримати шляхом використання довільної мікросхеми DAC, використати мікроконтролер ATmega8.*

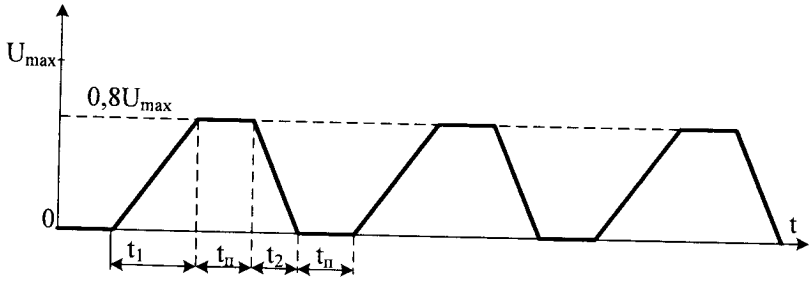


Рисунок 2.1 – Форма сигналу на виході пристрою

Згідно з поставленим завданням, для формування сигналу заданої форми потрібно використати мікросхему DAC, яка здійснює перетворення двійкового цифрового коду в аналоговий сигнал. Отже, завдання зводиться до реалізації алгоритму формування двійкового коду, пропорційного заданому сигналу в часі, та виведення його значення на порт. Оскільки пристрій буде містити декілька елементів, то необхідно розробити його структурну схему (рис. 2.2).

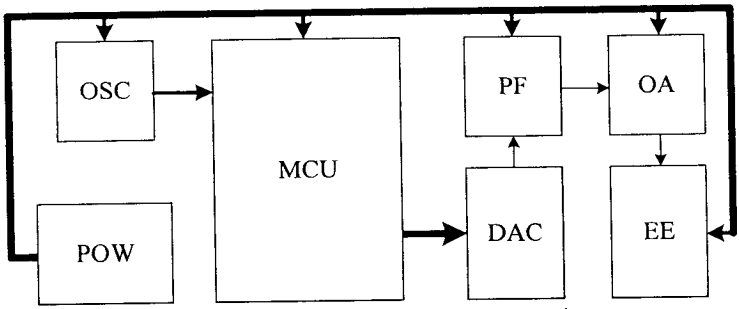


Рисунок 2.2 – Електрична структурна схема мікропроцесорного пристрою

На рис. 2.2: MCU – блок мікроконтролера; OSC – блок генератора тактового сигналу; POW – джерело напруги живлення; PF – фільтр; DAC – блок цифро-аналогового перетворення; OA – блок підсилювача напруги; EE – блок виконавчого органу.

Зазначена схема працює так. В MCU згідно з алгоритмом змінюється вміст восьмирозрядного робочого регістра. Характер зміни числа в згаданому регістрі повторює криву заданого сигналу в цифровому еквіваленті. Число із робочого регістра виводиться через порт введення-виведення на вхід цифро-аналогового перетворювача, далі аналоговий сигнал через фільтр PF та підсилювач OA подається на робочий орган EE.

На рис. 2.3 зображена відповідна схема електрична принципова мікропроцесорного пристрою. В мікроконтролері для виведення згенерованого восьмирозрядного коду задіяний порт PORTD. З виходу порту код подається на вхідну восьмирозрядну шину (A1-A8) цифро-аналогового перетворювача.

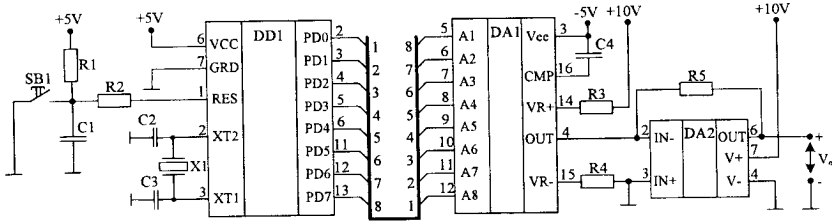


Рисунок 2.3 – Електрична принципова схема мікропроцесорного пристрою

На рис. 2.3: DD1 – мікроконтролер Atmega8; DA1 – цифро-аналоговий перетворювач DAC0808; DA2 – операційний підсилювач ma741C; C1-C4 – 0,1 мкФ; R1 – 10 кОм; R2 – 470 Ом; R3, R5 – 5 кОм; R4 – 1 кОм; X1 – кварцовий резонатор 1 МГц.

2.3 Розробка алгоритму роботи пристрою

Одним з важливих понять, на яких базується застосування мікроконтролерів для розв'язування різноманітних задач, є поняття алгоритму. Термін «алгоритм» звичайно використовується для позначення деякої послідовності дій, що приводять до досягнення потрібного результату.

Алгоритм – це набір інструкцій, що описує як деяке завдання може бути виконане. Іншими словами, алгоритм – система формальних правил, що визначає зміст і порядок дій над вхідними даними і проміжними результатами, необхідними для отримання кінцевого результату при розв'язуванні задачі.

2.3.1 Властивості алгоритму

При розв'язуванні будь-якої задачі та побудові алгоритму її розв'язку звичайно беруть до уваги наявність деяких вхідних даних і мають уявлення про результат, що необхідно отримати.

Будь-який алгоритм повинен мати такі основні властивості:

– детермінованість (визначеність) – через повну однозначність правил, встановлених в алгоритмі, застосування алгоритму до однакових вхідних даних повинно приводити до однакового результату;

– дискретність – процес, що визначається алгоритмом, можна розчленувати (розділити) на окремі елементарні етапи (кроки), кожен з яких називається кроком алгоритмічного процесу чи алгоритму;

– масовість – алгоритм повинен бути придатним для розв'язування всіх задач певного типу. Наприклад, алгоритм для розв'язування системи лінійних рівнянь повинен бути придатним для системи, що складається з довільної кількості рівнянь, причому для нього існує множина даних, що допускаються як вхідні, тобто початкова система величин може вибиратись із деякої потенційно нескінченної множини;

– результативність – вказує на наявність таких варіантів вхідних даних, для яких обчислювальний процес, що реалізується за наданим алгоритмом, повинен через скінченну кількість етапів (кроків) зупинитись і дати шуканий результат або сигнал про те, що наданий алгоритм непридатний для розв'язання поставленої задачі.

Як встановлено в теорії алгоритмів, існують і такі класи задач, для розв'язування яких нема і не може бути встановлено універсального прийому – алгоритму розв'язування (хоча при окремих обмеженнях на ці розв'язування алгоритм може бути знайдено). Такі задачі називають алгоритмічно нерозв'язуваними.

2.3.2 Основні структури алгоритмів

Основні структури алгоритмів – це обмежений набір блоків і стандартних способів їх з'єднання для виконання типових послідовностей дій. Використання кількох основних структур дає можливість будувати різнома-

нітні алгоритми.

До основних структур алгоритмів належать:

– лінійна або послідовна, без будь-яких розгалужень конфігурація алгоритму, що нагадує форму ланцюжка (рис. 2.4);

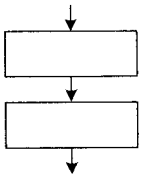


Рисунок 2.4 – Лінійна структура алгоритму

– розгалужена конфігурація алгоритму, що містить в собі як послідовності, так і розпаралелення послідовностей. Використовується, коли залежно від умови потрібно виконати ту чи іншу дію (рис. 2.5, а), здійснити обхід, якщо одна вітка не містить жодних дій (рис. 2.5, б), здійснити множинний вибір, коли умова має більше двох можливих варіантів (рис. 2.5, в, г);

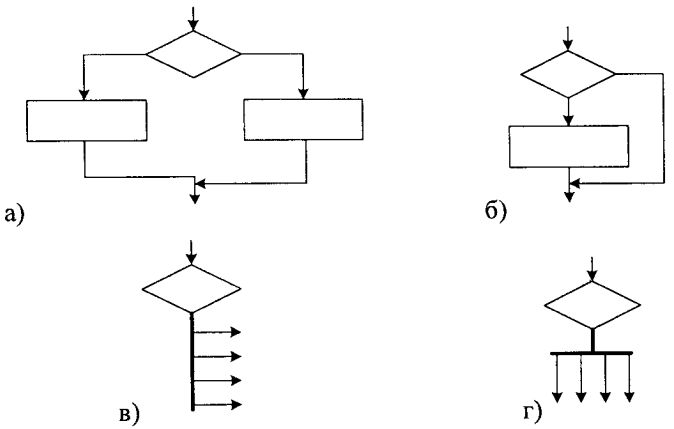


Рисунок 2.5 – Розгалужена структура алгоритму

– циклічна, що використовується за необхідності виконувати деякі дії кілька разів. Можливе виконання циклу «До», циклу «Поки», циклу за параметром (рис. 2.6).

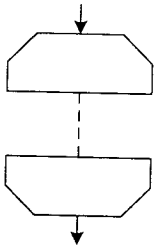


Рисунок 2.6 – Циклічна структура алгоритму

2.3.3 Опис алгоритму

На прикладі завдання, яке було розглянуте в підрозділі 2.2, розробимо структуру алгоритму та здійснимо його опис. Структура основного алгоритму програми для реалізації функціонала із згаданого завдання подана на рис. 2.7.

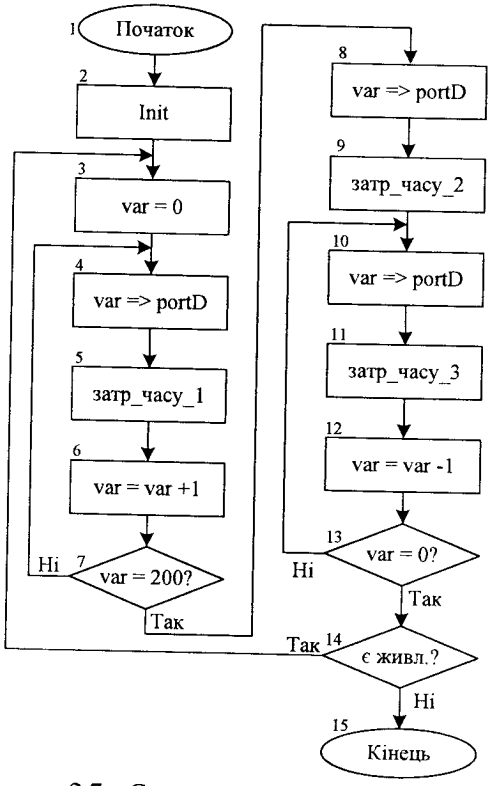


Рисунок 2.7 – Структура основного алгоритму

На рис. 2.7:

1. Блок «Початок» символізує місце, з якого буде починатися програма при подачі живлення на мікроконтролер;
2. В блоці «Ініціалізація» вказується область пам'яті мікроконтролера, присвоєння імен регістрам, визначення початкової адреси області STACK, налаштовуються порти введення/виведення;
3. Присвоєння значення 0 змінній var;
4. Виведення вмісту змінної var на порт D;
5. Виклик алгоритму першої затримки часу (50 мс). При цьому в пам'яті стека зберігається адреса програми (PC), з якої було викликано підпрограму затримки. Після виконання підпрограми затримки програма повертається в місце виклику PC+1;
6. Збільшення значення змінної var на одиницю;
7. Умова, в якій перевіряється, чи значення змінної var досягло 205. Константа 205 визначає число, яке при подачі на цифрову вхідну шину DAC сформує сигнал $0{,}8 \cdot U_{max}$ на його виході. У випадку, якщо не досягло – ідемо на наступну ітерацію, інакше – далі по алгоритму;
8. Виведення вмісту змінної var на порт D;
9. Виклик підпрограми затримки часу (30 мс);
10. Виведення вмісту змінної var на порт D;
11. Виклик підпрограми затримки часу (20 мс);
12. Зменшення значення змінної var на одиницю;
13. Умова, в здійснюється перевірка, чи значення змінної var досягло 0. У випадку якщо не досягло – ідемо на наступну ітерацію, інакше – далі по алгоритму;
14. Алгоритм припиняється, як тільки від мікроконтролера відключать живлення, блок описує відповідну властивість, якщо живлення відключене – перехід на блок «Кінець», інакше – на наступну ітерацію;
15. Блок «Кінець» – закінчення роботи програми.

Для реалізації затримки часу мовою програмування AVR Assembler, на даному етапі, будемо використовувати підпрограму, основану на виконанні так званих «пустих» команд. Тобто згадана підпрограма не виконує нічого «корисного», крім того, що займає деяку кількість тактів, які в сукупності формують затримку часу.

Алгоритми всіх трьох затримок часу типові, тому розглянемо їх на прикладі одного із них. На рис. 2.8 подано алгоритм реалізації затримки часу.

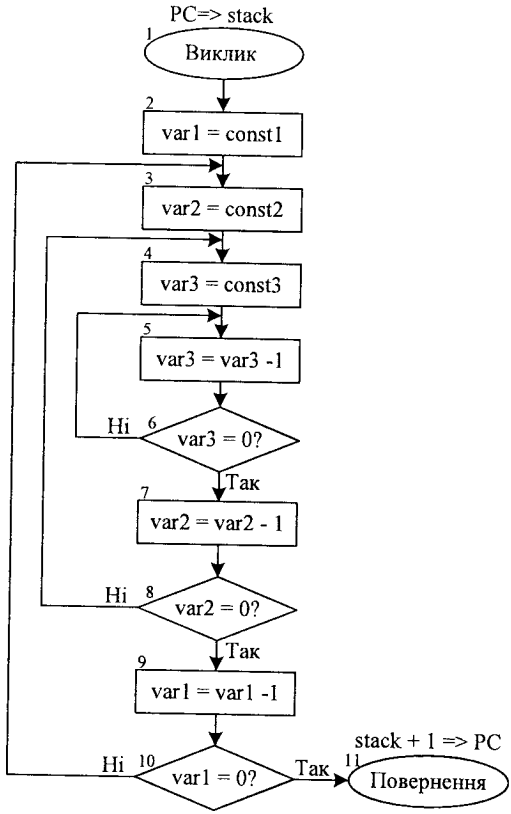


Рисунок 2.8 – Структура основного алгоритму

На рис. 2.8:

1. Блок «Виклик» відображає адресу пам'яті, за якою знаходиться підпрограма затримки часу. Після виклику підпрограми адреса місця виклику записується в STACK;
2. Запис константи const1 в змінну var1;
3. Запис константи const2 в змінну var2;
4. Запис константи const3 в змінну var3;
5. Зменшення вмісту змінної var3 на одиницю;
6. Перевірка, чи вміст змінної var3 дорівнює нулю, у випадку якщо ні – на наступну ітерацію, інакше – далі по алгоритму;
7. Зменшення вмісту змінної var2 на одиницю;

8. Перевірка, чи вміст змінної var2 дорівнює нулю, у випадку якщо ні – на наступну ітерацію, інакше – далі по алгоритму;
9. Зменшення вмісту змінної var1 на одиницю;
10. Перевірка, чи вміст змінної var1 дорівнює нулю, у випадку якщо ні – на наступну ітерацію, інакше – далі по алгоритму;
11. Блок відповідає за повернення із підпрограми у місце її виклику в основному алгоритмі. Для цього адреса місця виклику, збільшена на одиницю, записується в PC.

Розглянемо аспект реалізації згаданого алгоритму затримки часу та його особливості мовою програмування AVR Assembler. В наступному прикладі наведено підпрограму затримки часу 1 секунда при роботі мікроконтролера від джерела тактової частоти 1 МГц (текст за символом « ; » є однорядковим коментарем):

```
Delay_1sec:       ; For CLK(CPU) = 1 MHz
   LDI R20, 8     ; One clock cycle;
Delay1:
   LDI R21, 125   ; One clock cycle
Delay2:
   LDI R22, 250   ; One clock cycle
Delay3:
   DEC R22        ; One clock cycle
   NOP            ;One clock cycle
   BRNE Delay3    ; Two clock cycles when jumping to Delay3, 1 clock when continuing to DEC

   DEC R21        ; One clock cycle
   BRNE Delay2    ; Two clock cycles when jumping to Delay2, 1 clock when continuing to DEC

   DEC R20        ; One clock Cycle
   BRNE Delay1    ; Two clock cycles when jumping to Delay1, 1 clock when continuing to RET
RET
```

Одна ітерація внутрішнього циклу Delay3 виконується за 4 такти, тому що DEC = 1, NOP = 1 та BRNE = 2, коли відбувається перехід на Delay3. Цей цикл виконується 250 разів (1000 тактів або 1 мс), оскільки R22 містить число 250 до початку виконання циклу.

Цикл Delay2 повторюється 125 разів (R21 = 125), отже накопичена затримка буде становити 125 мс.

Зовнішній цикл Delay1 повторюється 8 разів (R20 = 8) – тримаємо затримку часу в 1000 мс або 1 секунду.

Слід зауважити, що реалізація алгоритму таким чином не дозволить виконувати інші операції впродовж затримки часу, але для виконання поставленої задачі підхід цілком прийнятний.

2.4 Розробка програми мовою програмування Assembler

2.4.1 Створення проекту в AVR Studio

Розглянемо послідовність дій при створенні проекту мовою програмування Assembler.

AVR Studio – це інтегроване середовище розробки (IDE) для мікроконтролерів сімейства AVR фірми Atmel.

IDE AVR Studio містить:

– транслятор мови асемблер (Atmel AVR macroassembler);

– налагоджувач (Debugger);

– програмне забезпечення верхнього рівня для підтримки внутрішньосхемного програмування (In-system Programming, ISP).

Для створення проекту необхідно:

1. Запустити AVR Studio, як показано на рис. 2.9;

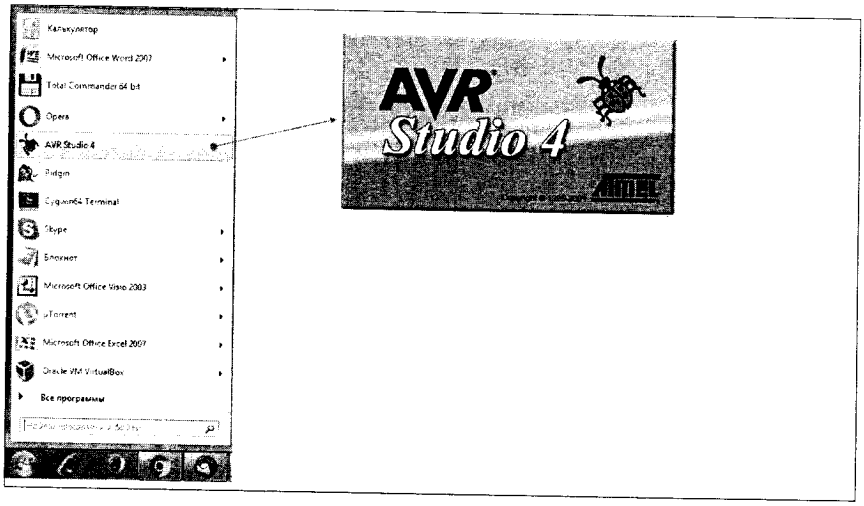


Рисунок 2.9 – Запуск AVR Studio IDE

2. В меню Project обрати команду New Project (рис. 2.10);

3. У вікні створення нового проекту необхідно вказати назву проекту (Project name) і його розміщення (Location). Новий проект зручно створювати в новій папці. Зверніть увагу, що назва проекту та його розміщення не повинні містити кириличні символи;

4. Далі обрати тип проекту – AVR Assembler, що використовує вбудований макроасемблер AVR Studio. Натиснути кнопку Next;

5. У вікні (рис. 2.11) в розділі Debug Platform вибрати AVR Simulator, у розділі Device – ATmega8 (тип контролера згідно з завданням). Натиснути кнопку Finish;

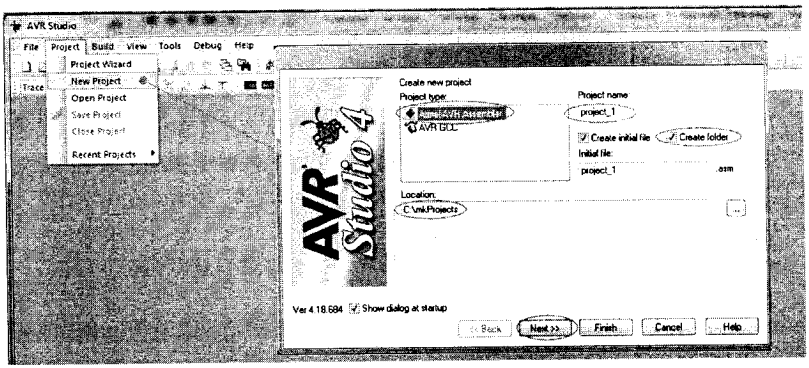


Рисунок 2.10 – Запуск AVR Studio IDE

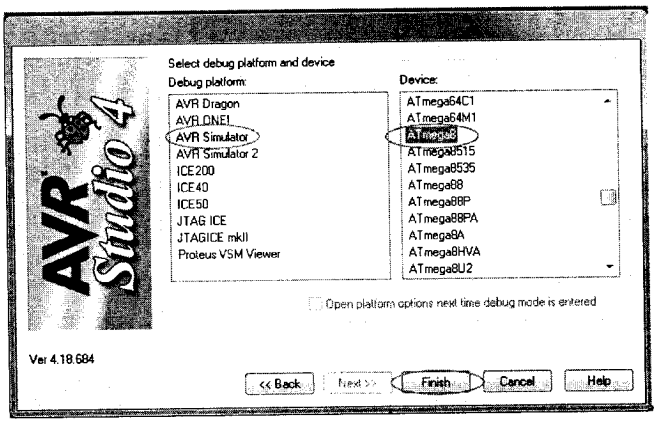


Рисунок 2.11 – Запуск AVR Studio IDE

6. У вікні редактора набрати програму;

7. Далі необхідно відтранслювати програму, тобто перевірити її на правильність написання. Для цього потрібно вибрати пункт Build в меню Build. У нижньому вікні, з'явиться інформація про кількість слів коду та даних, про наявність помилок і т. д.

2.4.2 Реалізація та опис програми на Assembler

Розглянемо програму мовою програмування Assembler та її опис на прикладі поставленого завдання в підрозділі 2.2.

Для забезпечення можливості використання в програмі констант, визначених для мікроконтролера ATmeg8, необхідно завантажити відповідний файл, як показано в такому рядку програми:

```
.include "m8def.inc"
```

Перша із наступних двох директив вказує на початок програмного сегмента, друга – встановлює лічильник програмного сегмента в 0:

```
.cseg
.org 0
```

В наступному рядку коду відбувається присвоєння імені робочому регістру:

```
.def temp = r16
```

Блок коду нижче реалізовує ініціалізацію області STACK в кінці області пам'яті RAM:

```
LDI r31,low(ramend)
OUT spl,r31
LDI r31,high(ramend)
OUT sph,r31
```

В наступному фрагменті програми здійснюється налаштування всіх ліній порту D, як вихідних ліній (виходів) та обнулення робочого регістра:

```
LDI temp, 0xff
OUT DDRD, temp
LDI temp, 0
```

Головний блок програми починається з такої позначки:

```
main:
```

Блок коду, який відповідає за формування ділянки наростання сигналу (див. рис. 2.1) тривалістю t_1:

```
t1:
   OUT PORTD, temp
   RCALL time_1
   INC temp
   CPI temp, 200
   BRNE t1
```

Ділянка усталеного сигналу в області максимуму (див. рис. 2.1) тривалістю $t_п$ реалізована таким чином:

```
   out PORTD, temp
   rcall time_p
```

Ділянка сигналу, на якій відбувається спад сигналу на періоді (див. рис. 2.1) тривалістю t_2 інтерпретована такими інструкціями програми:

```
t2:
   OUT PORTD, temp
   RCALL time_2
   DEC temp
   BRNE t2
```

Ділянка усталеного сигналу в області мінімуму (див. рис. 2.1) тривалістю $t_п$ реалізована таким чином:

```
   ldi temp, 0
   out PORTD, temp
   rcall time_p
```

Закінчення головного блока та перехід на його початок здійснюється командою, поданою нижче:

```
RJMP main
```

Блок коду, який реалізовує підпрограму затримки часу, що використовується в формуванні ділянки наростання сигналу (див. рис. 2.1) тривалістю t_1:

```
time_1:
LDI r17, 1
time_1_1:
   LDI r18, 1
time_1_2:
   LDI r19, 60
time_1_3:
    DEC r19
    NOP
    BRNE time_1_3
   DEC r18
   BRNE time_1_2
    DEC r17
    BRNE time_1_1
RET
```

Підпрограма, яка формує затримку часу, що використовується в реалізації ділянки спаду сигналу (див. рис. 2.1) тривалістю t_2:

```
time_2:
   LDI r17, 1
time_2_1:
   LDI r18, 1
time_2_2:
   LDI r19, 15
time_2_3:
   DEC r19
   NOP
   BRNE time_2_3
   DEC r18
   BRNE time_2_2
   DEC r17
   BRNE time_2_1
RET
```

Набір інструкцій підпрограми, яка створює затримку часу для формування паузи тривалістю $t_п$ в ділянках максимуму та мінімуму заданого сигналу (див. рис. 2.1):

```
time_p:
   LDI r17, 1
time_p_1:
   LDI r18, 80
time_p_2:
   LDI r19, 100
time_p_3:
   DEC r19
   NOP
   BRNE time_p_3
   DEC r18
   BRNE time_p_2
   DEC r17
   BRNE time_p_1
RET
```

2.5 Розробка програми мовою програмування С

2.5.1 Створення проекту CV AVR

Розглянемо послідовність дій при створенні проекту мовою програмування С.

CodeVisionAVR – IDE для розробки програмного забезпечення для мікроконтролерів сімейства AVR фірми Atmel.

До складу CodeVisionAVR входять такі компоненти.

1. Компілятор мови програмування С для AVR.

2. Генератор початкового коду програми, що дозволяє зробити ініціалізацію периферійних пристроїв.

3. Модуль взаємодії з відлагоджувальною платою STK-500.
4. Модуль взаємодії з програматором.
5. Редактор вихідного коду з підсвічуванням синтаксису.
6. Термінал.

Для створення проекту в CodeVisionAVR виконуємо пункти в такій послідовності:

1. Обираємо пункт New в меню File, в вікні Create New File ставимо відмітку напроти пункту Project, підтверджуємо намір створити проект у вікні Confirm, в вікні CodeWizardAVR вибираємо групу мікроконтролерів, до якої входить необхідний чип (рис. 2.12).

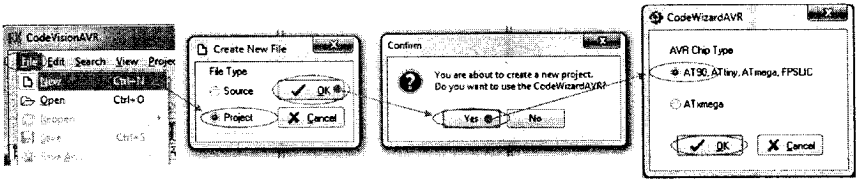


Рисунок 2.12 – Запуск AVR Studio IDE

2. У вікні CodeWizardAVR здійснюємо необхідні налаштування для генерації проекту та натискаємо кнопку генерації, у вікні Save необхідно вказати ім'я файлів, які створяться після генерації, три рази (рис. 2.13). Задаємо однакове ім'я файлів для всіх трьох випадків при збереженні. Імена проекту, файлів та шлях до розміщення проекту не повинні містити кириличних символів.

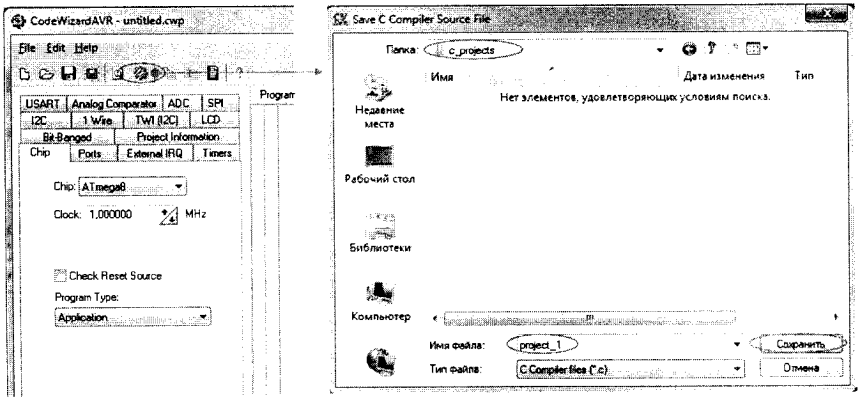


Рисунок 2.13 – Запуск AVR Studio IDE

2.5.2 Реалізація та опис програми на С

Розглянемо програму мовою програмування С та її опис на прикладі завдання в підрозділі 2.2.

Для забезпечення можливості використання в програмі констант, визначених для мікроконтролера ATmeg8, та бібліотечних функцій необхідно завантажити відповідні файли, як показано в такій частині програми:

```
#include <mega8.h>
#include <delay.h>
```

Основний алгоритм програми мовою програмування С повинен бути написаний в головній функції, яка викликається автоматично після подачі живлення на мікроконтролер. Визначення головної функції та її тіла подано нижче:

```
void main(void)
{
  int output_byte = 0;
```

В наведеній нижче групі інструкцій здійснено налаштування портів введення/виведення мікроконтролера. Зокрема всі виводи PORTD налаштовані як виходи, всі інші виводи (інших портів) залишаються входами за замовчуванням. Лінії всіх портів відключені від внутрішніх підтягувальних резисторів:

```
PORTB=0x00;
DDRB=0x00;
PORTC=0x00;
DDRC=0x00;
PORTD=0x00;
DDRD=0xff;
```

Для коректної роботи мікроконтролера потрібно в тілі головної функції визначити нескінченний цикл, який буде перериватися тільки у випадку відключення мікроконтролера від живлення. В тілі нескінченного циклу реалізовується основний алгоритм мікроконтролера:

```
while (1)
{
```

Затримка часу тривалістю $t_п$ – пауза в періоді між імпульсами заданого сигналу (див. рис. 2.1):

```
delay_ms(20);
```

Реалізація ділянки наростаючого сигналу тривалістю t_1 (див. рис. 2.1) здійснена за допомогою групи інструкцій в тілі такого циклу:

```
while(output_byte<205)
{
        PORTD = output_byte++;
        delay_us(244);
}
```

Ділянка усталеного сигналу в області максимуму тривалістю $t_п$ (див. рис. 2.1) організована викликом функції затримки із відповідним параметром у вказаному місці програми:

```
delay_ms(20);
```

Реалізація ділянки спадаючого сигналу тривалістю t_2 (див. рис. 2.1) здійснена за допомогою групи інструкцій в тілі циклу, поданого нижче:

```
while(output_byte>0)
{
        PORTD = output_byte--;
        delay_us(146);
}
  }
}
```

2.6 Моделювання роботи пристрою

2.6.1 Особливості програмного пакета Proteus VSM

Proteus VSM складається з двох самостійних програм ISIS і ARES. ARES використовується для розробки друкованих плат з можливістю створення своїх бібліотек корпусів і т. д.

Основною програмою пакета є ISIS, в ній передбачений зв'язок з ARES для автоматичного розведення плат згідно зі схемою.

При запуску програми з'являється головне вікно (рис. 2.14).

Найбільшу частину вікна відведено під робочу область Edit Window (на рис. 2.14 – 1). Саме ній відбуваються створення, редагування та налагодження схеми пристрою.

Зліва вгорі маленьке вікно попереднього перегляду Overview Window (на рис. 2.14 – 2), з його допомогою можна переміщатися по вікну редагування (клацаючи лівою кнопкою миші по вікну попереднього перегляду, ми переміщаємо вікно редагування по схемі, якщо звичайно схема не вміщається у вікно). Наближати і віддаляти схему у вікні можна відповідно клавішами F6 і F7 або ж скролом миші, F5 центрує схему у вікні, а натискання F8 підганяє розмір схеми під вікно редагування.

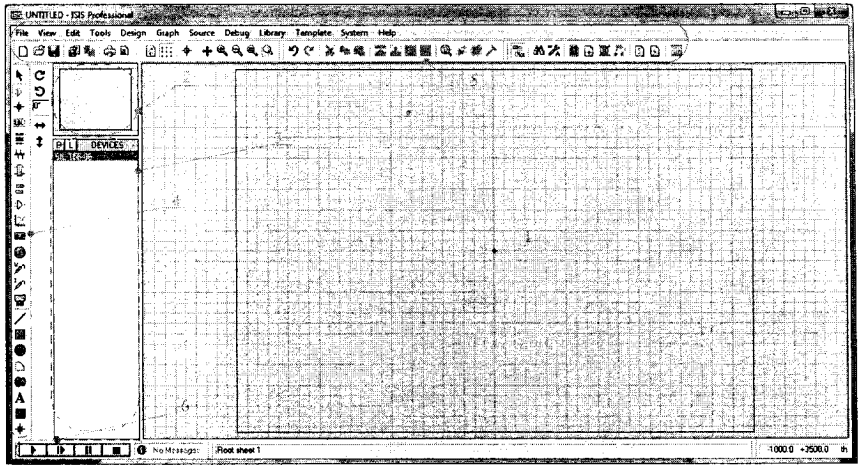


Рисунок 2.14 – Вигляд головного вікна Proteus VSM

Під вікном попереднього перегляду знаходиться Object Selector (на рис. 2.14 – 3) список вибраних в даний момент компонентів, символів та інших елементів. Виділений в списку об'єкт відображається у вікні попереднього перегляду.

Всі можливі функції та інструменти Proteus VSM доступні через меню, розташовані на верхній та лівій боковій панелях (на рис. 2.14 – 4, 5), як відповідні піктограми.

Внизу основного вікна розташовані елементи керування (на рис. 2.14 – 6): кнопки пуск, покроковий режим виконання (використовується для відлагоджування), пауза, стоп та рядок статусу, в якому відображаються помилки, підказки, поточний стан процесу симуляції і т. д.).

Для побудови схеми моделювання необхідно користуватися бібліотекою компонентів, для цього потрібно перейти в меню Component (компоненти), натиснувши відповідну піктограму (рис. 2.15, а).

Клацнувши на піктограмі P (Pick devices) або двічі клацнувши лівою кнопкою в полі вибору компонентів Object Selector, відкриваємо вікно бібліотеки компонентів. Компоненти можна вибирати за категоріями – Category, підкатегоріями – Sub category, виробником – Manufacturer або ж шукати за ключовими словами Keywords.

2.6.1 Схема моделі та результати моделювання

Відповідно до розроблених структурної та принципової схем (див. рис. 2.2 та 2.3), розробимо схему моделювання в середовищі Proteus VSM. Для цього створимо проект та виберемо перелік необхідних елементів із бібліотеки компонентів, як показано на рис. 2.15, а). Схема моделювання мікропроцесорного пристрою зображена на рис. 2.16.

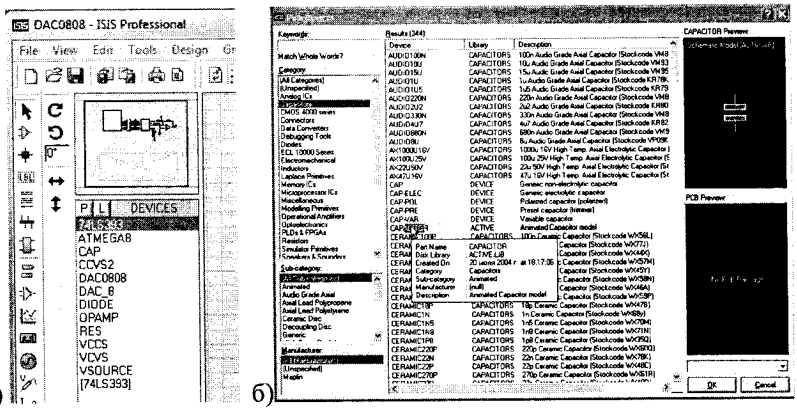


Рисунок 2.15 – Меню компонентів

Основні блоки, які були вибрані для побудови схеми:
– блок моделювання мікроконтролера - ATMEGA8;
– блок моделювання DAC – DAC0808;
– блок операційного підсилювача.

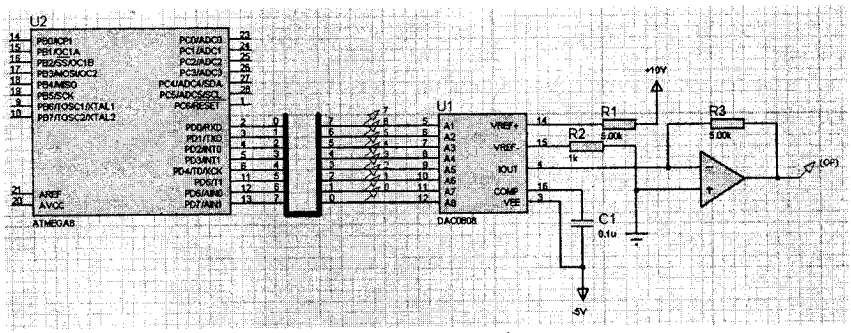


Рисунок 2.16 – Схема моделювання мікропроцесорного пристрою

Вікно налаштувань блока мікроконтролера (Edit Component) подано на рис. 2.17. Вікно Edit Component містить ряд опцій, зокрема, поле Program File використовується для вибору файлу з програмним кодом, поле CKSEL Fuses – для встановлення тактової частоти роботи мікроконтролера тощо.

Якщо в тексті є посилання на складові частини зображеного засобу, то на відповідній ілюстрації вказують їх порядкові номери в межах ілюстрації.

Якщо ілюстрація є фрагментом повної розробленої схеми, то для всіх компонентів вказують ті позиційні позначення, які вказані на схемі.

Якщо ілюстраціями є фотографії, то останні повинні бути наклеєні на стандартні аркуші білого паперу і позначені як рисунки.

3.5 Оформлення таблиць

Таблицю розміщують симетрично до тексту після першого посилання на даній сторінці або на наступній, якщо на даній вона не уміщується, і таким чином, щоб зручно було її розглядати без повороту або з поворотом на кут 90° проти годинникової стрілки.

Пропонується такий запис таблиці:

Таблиця _____ – _____
(номер) *(назва таблиці)*

На всі таблиці мають бути посилання за формою: *«наведено в таблиці 3.1»*; *« ... в таблицях 3.1 – 3.5»* або в дужках по тексту (*таблиця 3.6*). Посилання на раніше наведену таблицю дають з скороченим словом *«дивись»* (*див. таблицю 2.4*) за ходом чи в кінці речення.

Таблицю розділяють на графи (колонки) і рядки. В верхній частині розміщують головку таблиці, в якій вказують найменування граф. Діагональне ділення головки таблиці не допускається. Ліву графу (боковик) часто використовують для найменування рядків. Допускається не розділяти рядки горизонтальними лініями. Мінімальний розмір між основами рядків – 8 мм. Розміри таблиці визначаються обсягом матеріалу. Графу «№ з/п» до таблиці не включають. За необхідності нумерації, номери вказують в боковику таблиці перед найменуванням рядка.

Найменування граф може складатися з заголовків і підзаголовків, які записують в однині, симетрично до тексту графи малими буквами, починаючи з великої. Якщо підзаголовок становить одне речення з заголовком, то в цьому випадку його починають з малої букви. В кінці заголовків і підзаголовків граф таблиці крапку не ставлять. Дозволяється заголовки і підзаголовки граф таблиці виконувати через один інтервал.

Якщо всі параметри величин, які наведені в таблиці, мають одну й ту саму одиницю фізичної величини, то над таблицею розміщують її скорочене позначення (*мм*). Якщо ж параметри мають різні одиниці фізичних величин, то позначення одиниць записують в заголовках граф після коми (*Довжина, мм*).

Текст заголовків і підзаголовків граф може бути замінений буквеними позначеннями, якщо тільки вони пояснені в попередньому тексті чи на

Побудуємо схему пристрою в програмному середовищі Proteus VSM. Схема моделювання подана на рис. 2.19.

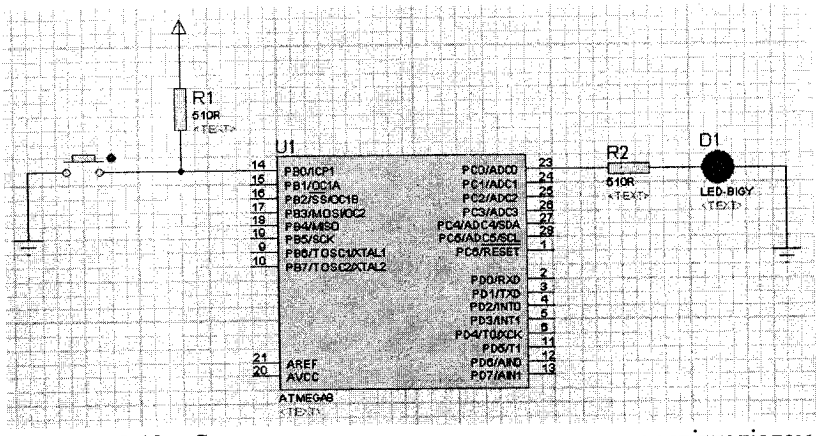


Рисунок 2.19 – Схема моделювання пристрою керування світлодіодом

2.7.1.1 Програмування портів введення/виведення мовою програмування Assembler

1. Текст програми, яка реалізовує алгоритм опитування однієї кнопки та перемикання стану світлодіода (ввімкнений/вимкнений) подано нижче.

Блок ініціалізації та налаштувань мікроконтролера:

```
.include "m8def.inc"
LDI r16, 0b00000000
OUT ddrb, r16
LDI r16, 0b00000001
OUT ddrc, r16
```

Блок основної програми, який розглянемо детальніше, починається з позначки:

```
main:
```

Записуємо нуль в робочий регістр:

```
LDI r16, 0b00000000
```

Перевіряємо нульовий біт службового регістра PINB, який відображає стан на фізичному виводі (у даному випадку вході) PORTB.0. Якщо біт очищений (кнопка натиснута), то наступна команда пропускається:

```
SBIC pinb, 0
```

Якщо біт PINB.0 не був очищений (кнопка не натиснута), то записуємо в робочий регістр одиницю:

```
LDI r16, 0b00000001
```

Таким чином, наступна команда виводить на PORTC вміст робочого регістра. Залежно від вмісту робочого регістра (від стану кнопки), на порт виведеться або одиниця, або нуль:

```
OUT portc, r16
```

Перехід на позначку початку основної програми:

```
RJMP main
```

2. Опишемо програму, яка реалізовує алгоритм опитування однієї кнопки та реалізації ефекту «мигання» світлодіода. Після натискання кнопки світлодіод переходить у режим «мигання», повторне натискання кнопки вимикає режим і т. д.

Слід виокремити такий аспект. Для того, щоб система реагувала на натискання кнопки в режимі «мигання», коли алгоритм відпрацьовує затримку часу після засвічування або погашення світлодіода, в поточній реалізації запропоновано опитувати стан кнопки у вкладеному циклі алгоритму затримки. Таким чином частота опитування кнопки буде значно більшою, ніж частота мигання світлодіода у однойменному режимі.

Блок ініціалізації відрізняється від попереднього наявністю частини, яка відповідає за визначення початкової адреси для STACK, що забезпечує можливість використання підпрограм:

```
.include "m8def.inc"
```

Ініціалізація вершини STACK в кінці оперативної пам'яті мікроконтролера SRAM:

```
LDI     r16, low(RAMEND)
OUT     spl, r16
LDI     r16, high(RAMEND)
OUT     sph, r16

LDI r16, 0b00000000
OUT ddrb, r16
LDI r16, 0b00000001
OUT ddrc, r16
```

Блок основної програми складається із двох підблоків, які реалізовують два стійких режими: режим «спокою», коли світлодіод вимкнений та режим «мигання». В кожному із перерахованих підблоків реалізовано безпосередньо відповідний режим, процес опитування кнопки та забезпечення переходу в інший режим. При натисненні кнопки в тілі одного із підблоків відбувається перехід до виконання алгоритму іншого:

```
main:
```

Підблок, що реалізовує режим «спокою» світлодіода та опитування кнопки у вказаному режимі:

```
do_nothing:
        while_pressed:
                SBIS pinb, 0
        RJMP while_pressed
        RCALL drebezg
        LDI r16, 0b00000000
        OUT portc, r16
        RCALL delay
        SBIS pinb, 0
        RJMP do_blinking
RJMP do_nothing
```

Підблок, що реалізовує режим «мигання» світлодіода та опитування кнопки у вказаному режимі:

```
do_blinking:
        while_pressed_:
                SBIS pinb, 0
        RJMP while_pressed_
        RCALL drebezg
        LDI r16, 0b00000001
        OUT portc, r16
        RCALL delay
        SBIS pinb, 0
        RJMP do_nothing
        LDI r16, 0b00000000
        OUT portc, r16
        RCALL delay
        SBIS pinb, 0
        RJMP do_nothing
  RJMP do_blinking
RJMP main
```

Підпрограма затримки часу для фіксації стану світлодіода в режимі «мигання»:

```
delay:
LDI r17, 2
time_1_1:
   LDI r18,200
time_1_2:
   LDI r19, 200
time_1_3:
```

Опитування кнопки у вкладеному циклі алгоритму затримки часу. В разі натискання кнопки відбувається вихід із підпрограми та опрацювання зміни режиму на протилежний:

```
    SBIS pinb, 0
    RET

    DEC r19
    NOP
    BRNE time_1_3

    DEC r18
    BRNE time_1_2
    DEC r17
    BRNE time_1_1
RET
```

Затримка часу для компенсації ефекту брязкоту контактів кнопки:

```
drebezg:
LDI r17, 1
time_2_1:
   LDI r18, 1
time_2_2:
   LDI r19, 5
time_2_3:
    DEC r19
    NOP
    BRNE time_2_3
    DEC r18
    BRNE time_2_2
    DEC r17
    BRNE time_2_1
 RET
```

2.7.1.2 Програмування портів введення/виведення мовою програмування С

1. Опишемо програму, яка реалізовує алгоритм опитування однієї кнопки та перемикання стану світлодіода (ввімкнений/вимкнений):

```
#include <mega8.h>
int output = 0;
void main(void)
{
   PORTB=0x00;
   DDRB=0x00;
   PORTC=0x00;
   DDRC=0x01;
```

Основний цикл програми:

```
   while (1)
   {
```

Присвоюємо проміжній змінній значення нуля:

```
         output = 0;
```

Якщо кнопка натиснута, то проміжній змінній присвоюється одиниця:

```
         if(PINB.0==0) output = 1;
```

Виводимо на PORTC вміст проміжної змінної:

```
         PORTC = output;
   }
}
```

2. Розглянемо приклад програми на С, яка реалізовує алгоритм опитування кнопки та перемикання режимів індикації світодіода:

```
#include <mega8.h>
#include <delay.h>
int counter = 0;
```

В поточній реалізації алгоритму визначено користувацьку функцію затримки часу, яка приймає параметр – ціле число time_ms. В тілі функції реалізовано цикл, який time_ms разів виконує команду затримки в 1 мс та перевірку стану кнопки. У випадку, якщо кнопка була натиснута, функція переривається і повертає «1», інакше згаданий цикл виконується передбачену кількість разів і після закінчення виконання функції повертається «0»:

```
int my_custom_delay(int time_ms)
{
  for(counter = 0; counter<time_ms; counter++)
  {
        delay_ms(1);
        if (PINB.0 == 0) return 1;
  }
  return 0;
}
```

Головна функція програми:

```
void main(void)
{
      PORTB=0x00;
      DDRB=0x00;
      PORTC=0x00;
      DDRC=0xFF;
      PORTD=0x00;
      DDRD=0x00;
```

Головний цикл програми:

```
while (1)
{
```

Поки кнопка не натиснута – режим «спокою»:

```
      while(PINB.0 == 1)
      {
            //do nothing
            PORTC.0 = 0;
      }
```

Після натискання кнопки виконується затримка часу для компенсації брязкоту контактів:

```
      while(PINB.0 == 0)
      delay_ms(10);
```

Цикл відпрацювання режиму «мигання» світлодіода:

```
      while(PINB.0 == 1)
      {
            PORTC.0 = 1;
```

Якщо в функції затримки часу, яка фіксує «засвічений» стан світлодіода була натиснута кнопка, то виходимо із циклу:

```
if(my_custom_delay(600)) break;
PORTC.0 = 0;
```

Аналогічно – залишаємо цикл, якщо кнопка була натиснута, у функції затримки часу, яка фіксує «погашений» стан світлодіода:

```
        if(my_custom_delay(600)) break;
    }
```

Затримка часу після натискання кнопки для врахування брязкоту контатків:

```
    while(PINB.0 == 0)
    delay_ms(10);
  }
}
```

3 ПРАВИЛА ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

При оформленні пояснювальної записки курсової роботи необхідно дотримуватись вимог ДСТУ 3008-95, а також стандарту на виконання документів з використанням друкувальних і графічних пристроїв виведення ЕОМ.

Пояснювальна записка курсової роботи з врахуванням вимог до нормативно-технічних документів має подаватись на аркушах паперу формату А4 без рамок.

Текст пояснювальної записки виконується машинописним (друкарським) або машинним (за допомогою комп'ютерної техніки) способом на одному боці аркуша білого паперу.

За машинного способу текст пояснювальної записки друкують з розрахунку не більше 40 рядків на сторінці за умови її рівномірного заповнення та висотою літер і цифр не менше, ніж 1,8 мм.

За машинописного способу виконання текст друкують через 1,5 інтервали.

Відступи тексту від країв аркуша: зверху, знизу і зліва – не менше 20 мм; справа – не менше 10 мм. Абзацний відступ – 5 знаків.

Сторінки пояснювальної записки слід нумерувати арабськими цифрами, додержуючись наскрізної нумерації впродовж усього тексту звіту. Номер сторінки проставляють у правому верхньому куті сторінки без крапки в кінці.

3.1 Вимоги до оформлення розділів та підрозділів

Текст основної частини пояснювальної записки поділяють на розділи, підрозділи, пункти та підпункти.

Розділи і підрозділи повинні мати заголовки. Пункти і підпункти можуть мати заголовки.

Заголовки розділів, а також таких структурних елементів курсової роботи як «ЗМІСТ», «ВСТУП», «ВИСНОВКИ» та «ПЕРЕЛІК ПОСИЛАНЬ» слід розташовувати посередині рядка і друкувати великими літерами без крапки в кінці, не підкреслюючи.

Заголовки підрозділів, пунктів і підпунктів слід починати з абзацного відступу і друкувати маленькими літерами, крім першої великої, не підкреслюючи, без крапки в кінці.

Якщо заголовок складається з двох і більше речень, їх розділяють крапкою. Перенесення слів у заголовку розділу не допускається.

Заголовки розділів, підрозділів, пунктів і підпунктів слід нумерувати арабськими цифрами. Після номера крапку не ставлять, а пропускають один знак.

Розділи нумерують порядковими номерами в межах всього документа

(1, 2 і т. д.).

Підрозділи нумерують в межах кожного розділу, пункти в межах підрозділу і т. д. за формою (3.1, 3.2, 3.2.1, 3.2.2, 3.2.2.1 і т. д.).

Між заголовком розділу та підрозділу, а також заголовком розділу або підрозділу та подальшим чи попереднім текстом повинні бути забезпечені відступи.

Допускається розміщувати текст між заголовками розділу і підрозділу, між заголовками підрозділу і пункту.

Кожен розділ рекомендується починати з нової сторінки.

Не допускається розміщувати назву підрозділу, пункту й підпункту в нижній частині сторінки, якщо після неї розміщено тільки один рядок тексту.

Посилання в тексті на розділи виконується за формою: *«...наведено в розділі 3»*.

3.2 Правила написання тексту

При написанні тексту слід дотримуватися таких правил:

а) текст необхідно викладати обґрунтовано в лаконічному технічному стилі;

б) умовні буквені позначення фізичних величин і умовні графічні позначення компонентів повинні відповідати установленим стандартам. Перед буквеним позначенням фізичної величини повинно бути її пояснення (*резистор R, конденсатор C*);

в) числа з розмірністю слід записувати цифрами, а без розмірності – словами (*відстань – 2 мм, відміряти три рази*);

г) позначення одиниць слід писати в рядок з числовим значенням без перенесення в наступний рядок. Між останньою цифрою числа і позначенням одиниці слід робити пропуск (*100 Вт, 2 А*);

д) якщо наводиться ряд числових значень однієї і тієї ж фізичної величини, то одиницю фізичної величини вказують тільки після останнього числового значення (*1,5; 1,75; 2 мм*);

е) позначення величин з граничними відхиленнями слід записувати так: 100 ± 5 *мм*;

ж) буквені позначення одиниць, які входять в добуток, розділяють крапкою на середній лінії (·); знак ділення замінюють косою рискою (/);

и) порядкові числівники слід записувати цифрами з відмінковими закінченнями (*9-й день, 4-а лінія*); при кількох порядкових числівниках відмінкове закінчення записують після останнього (*3, 4, 5-й графіки*); кількісні числівники записують без відмінкових закінчень (*на 20 аркушах*); не пишуть закінчення в датах (*21 жовтня*) та при римських числах (*XXI століття*);

к) скорочення слів в тексті не допускаються, крім загальноприйнятих в українській мові, а також скорочень, які прийняті для надписів на виробі (в

тексті вони повинні бути виділені великими літерами: *ON, OFF*), а якщо надпис складається з цифр або знаків, то в лапках. Лапками також виділяють найменування команд, режимів, сигналів (*«Запуск»*);

л) не дозволяється:

– допускати професійних або місцевих слів і виразів (техніцизмів);

– після назви місяця писати слово *«місяць»* (не *«в травні місяці»*, а *«в травні»*);

– використовувати вирази: *«цього року»*, *«минулого року»*, слід писати конкретну дату *«в червні 2001 року»*;

– використовувати позначення одиниць фізичних величин без цифр, необхідно писати повністю: *«кілька кілограмів»* (за винятком оформлення таблиць і формул);

– з'єднувати текст з умовним позначенням фізичних величин за допомогою математичних знаків (не *«швидкість = 5 км/год»*, а *«швидкість дорівнює 5 км/год»*, не *«температура дорівнює – 5 °С»*, а *«температура дорівнює мінус 5 °С»*);

– використовувати математичні знаки <, >, *0*, *№*, *%*, *sin*, *cos*, *tg*, *log* та ін. без цифрових або буквених позначень. В тексті слід писати словами *«нуль»*, *«номер»*, *«логарифм»* і т. д.;

– використовувати індекси стандартів (*ДСТУ, СНіП, СТП*) без реєстраційного номера.

В тексті документа може наводитись перелік, який рекомендується нумерувати малими літерами української абетки з дужкою або дефісом перед текстом (перший рівень деталізації). Для подальшої деталізації переліку використовують арабські цифри з дужкою (другий рівень деталізації).

Переліки першого рівня деталізації друкують малими літерами з абзацного відступу, другого рівня – з відступом відносно місця розташування переліків першого рівня.

Приклад:

а) форма і розмір клітин;

б) живий склад клітин:

1) частини клітин;

2) неживі включення протопластів;

в) утворення тканини.

3.3 Оформлення формул

Формули та рівняння розташовують безпосередньо після тексту, в якому вони згадуються, посередині сторінки. Вище і нижче кожної формули або рівняння повинно бути залишено не менше одного вільного рядка.

Умовні буквені позначення (символи) в формулі повинні відповідати установленим в стандартах. Їх пояснення наводять в тексті або зразу ж під формулою. Для цього після формули ставлять кому і записують пояснення до кожного символа з нового рядка в тій послідовності, в якій вони наве-

дені у формулі, розділяючи крапкою з комою. Перший рядок повинен починатися з абзацу з слова *«де»* і без будь-якого знака після нього.

Всі формули нумерують в межах розділу арабськими цифрами. Номер вказують в круглих дужках з правої сторони, в кінці рядка, на рівні закінчення формули. Номер формули складається з номера розділу і порядкового номера формули в розділі, розділених крапкою. Дозволяється виконувати нумерацію в межах всього документа.

Струм короткого замикання

$$I_{кз} = \frac{U}{R_я}, \tag{3.1}$$

де U – напруга прикладена до якоря, В;

$R_я$ *– опір кола якоря, Ом.*

Якщо виникає необхідність зазначити значення параметрів, які входять до складу формули, то це здійснюють так:

$$I_{кз} = \frac{U}{R_я}, \tag{3.2}$$

де U – напруга прикладена до якоря (U =220 В);

$R_я$ *– опір кола якоря (*$R_я$ *=3,3 Ом).*

Одиницю вимірювання, за необхідності, беруть в квадратні дужки

$$I_{кз} = \frac{U}{R_я}\ [\text{А}]. \tag{3.3}$$

Числову підстановку і розрахунок виконують з нового рядка не нумеруючи. Одиницю вимірювання беруть в круглі дужки. Наприклад,

$$I_{кз} = \frac{220}{100} = 2{,}2\ (\text{А}).$$

Розмірність одного й того ж параметра в межах документа повинна бути однаковою.

Якщо формула велика, то її можна переносити в наступні рядки. Перенесення виконують тільки математичними знаками, повторюючи знак на початку наступного рядка. При цьому знак множення «·» замінюють знаком «×».

Формула є частиною речення, тому до неї застосовують такі ж правила граматики, як і до інших членів речення. Якщо формула знаходиться в кінці речення, то після неї ставлять крапку. Формули, які йдуть одна за одною і не розділені текстом, відокремлюють комою.

Посилання на формули в тексті дають в круглих дужках за формою: *«...в формулі (5.2)»*; *«... в формулах (5.7, ..., 5.10)»*.

3.4 Оформлення ілюстрацій

Для пояснення викладеного тексту рекомендується його ілюструвати графіками, кресленнями, фрагментами схем та ін., які можна виконувати чорною тушшю, простим олівцем середньої твердості та за допомогою комп'ютерної графіки.

Розміщують ілюстрації в тексті пояснювальної записки або в додатках.

В тексті ілюстрацію розміщують симетрично до тексту після першого посилання на неї або на наступній сторінці, якщо на даній вона не уміщується без повороту.

Ілюстрації виділяють з тексту відступами.

Всі ілюстрації в пояснювальній записці називають рисунками і позначають під ілюстрацією симетрично до неї за такою формою: *«Рисунок 3.5 – Найменування рисунка»*. Крапку в кінці не ставлять, знак переносу не використовують. Якщо найменування рисунка довге, то його продовжують у наступному рядку починаючи від найменування.

На всі ілюстрації в тексті пояснювальної записки мають бути посилання. Посилання виконують за формою: *«...показано на рисунку 3.1»* або в дужках за текстом (*рисунок 3.1*), на частину ілюстрації: *«... показані на рисунку 3.2, б»*. Посилання на раніше наведені ілюстрації дають зі скороченим словом *«дивись»* відповідно в дужках (*див. рисунок 1.3*).

Нумерують ілюстрації в межах розділів, вказуючи номер розділу і порядковий номер ілюстрації в розділі, розділяючи їх крапкою.

Пояснювальні дані розміщують під ілюстрацією над її позначенням.

У випадку, коли ілюстрація складається з частин, їх позначають малими буквами українського алфавіту з дужкою *а), б)* і т. д. під відповідною частиною. В такому випадку після найменування ілюстрації ставлять двокрапку і дають найменування кожної частини за формою:

Рисунок 3.2 – Фазометр: а) – структурна схема; б) – часові діаграми роботи

або за ходом найменування ілюстрації, беручи букви в дужки:

Рисунок 3.2 – Структурна схема (а) і часові діаграми (б) роботи фазометра

Якщо ілюстрація не вміщується та одній сторінці, можна переносити її на інші сторінки, вміщуючи назву ілюстрації на першій сторінці, пояснювальні дані – на кожній сторінці, і під ними позначають: "Рисунок __, аркуш __".

ілюстраціях (*D – діаметр*, *H – висота* і т. д.). Однакові буквені позначення групують послідовно в порядку зростання їх індексів, наприклад: (*L1*, *L2*, ...).

Найменування рядків записують в боковику таблиці у вигляді заголовків в називному відмінку однини, малими буквами, починаючи з великої і з однієї позиції. В кінці заголовків крапку не ставлять. Позначення одиниць фізичних величин вказують в заголовках після коми. Для опису певного інтервалу значень в найменуваннях граф і рядків таблиці можна використовувати слова: *«більше»*, *«менше»*, *«не більше»*, *«не менше»*, *«в межах»*. Ці слова розміщують після одиниці фізичної величини:

(*Напруга, В, не більше*),

а також використовують слова *«від»*, *«більше»*, *«до»*:

(*Від 10 до 15; більше 15; до 20*)

Дані, що наводяться в таблиці, можуть бути словесними і числовими. Слова записують в графах з однієї позиції. Текст, що повторюється в рядках однієї і тієї ж графи та складається з окремих слів, що чергуються з цифрами, замінюють лапками (″). Якщо текст складається з двох і більше слів, то при першому повторенні його замінюють словами *«Те ж»*, а далі лапками. При розділенні таблиці горизонтальними лініями ніякої заміни не виконують.

Числа записують посередині графи так, щоб їх однакові розряди по всій графі були точно один під одним, за винятком випадку, коли вказують інтервал. Інтервал вказують від меншого числа до більшого з тире між ними:

12–35,
122–450.

Дробові числа наводять у вигляді десяткових дробів, з однаковою кількістю знаків після коми в одній графі. Розміри в дюймах можна записувати у вигляді: *1/2″, 1/4″, 1/8″.*

Ставити лапки замість цифр чи математичних символів, які повторюються, не можна. Якщо цифрові чи інші дані в таблиці не наводяться, то ставиться прочерк.

Таблиці нумерують в межах розділів і позначають зліва над таблицею за формою: *«Таблиця 4.2 – Найменування таблиці»*. Крапку в кінці не ставлять. Якщо найменування таблиці довге, то продовжують у наступному рядку, починаючи від слова *«Таблиця»*. Номер таблиці складається з номера розділу і порядкового номера таблиці в розділі, розділених крапкою. Дозволяється нумерувати в межах всього документа.

Таблиця може бути великою як в горизонтальному, так і в вертикальному напрямках, або, іншими словами, може мати велику кількість граф і рядків. В таких випадках таблицю розділяють на частини і переносять на інші сторінки або розміщують одну частину під іншою чи поряд.

Якщо частини таблиці розміщують поряд, то в кожній частині повторюють головку таблиці, а при розміщенні однієї частини під іншою –

повторюють боковик.

Якщо в кінці сторінки таблиця переривається і її продовження буде на наступній сторінці, в першій частині таблиці нижню горизонтальну лінію, що обмежує таблицю, не проводять.

При перенесенні частин таблиці на інші сторінки повторюють або продовжують найменування граф. Допускається виконувати нумерацію граф на початку таблиці і при перенесенні частин таблиці на наступні сторінки повторювати тільки нумерацію граф.

У всіх випадках найменування (за його наявності) таблиці розміщують тільки над першою частиною, а над іншими частинами зліва пишуть *«Продовження таблиці 4.2»* без крапки в кінці.

Інші вимоги до виконання таблиць – відповідно до чинних стандартів на технічну документацію.

3.6 Зміст

Зміст розташовують безпосередньо після анотації, починаючи з нової сторінки. До змісту відносять: перелік умовних позначень, символів, одиниць, скорочень і термінів; вступ; послідовно перелічені назви всіх розділів, підрозділів, пунктів і підпунктів (якщо вони мають заголовки); висновки; рекомендації; літературу; назви додатків і номери сторінок, які містять початок матеріалу.

Зміст за нумерацією пояснювальної записки є третьою сторінкою.

Назви заголовків змісту повинні однозначно відповідати назвам заголовків пояснювальної записки за текстом. Нумерація сторінок повинна бути наскрізною. Форма подання розділів та підрозділів в змісті для курсової роботи показана нижче.

1 РОЗРОБКА ...
1.1 Варіанти ...
1.1.1 ...
2 ЗАГОЛОВОК ДРУГОГО РОЗДІЛУ
2.1 Заголовки підрозділів
2.1.1 ...
3 ЗАГОЛОВОК ТРЕТЬОГО РОЗДІЛУ
3.1 ...

При виконанні курсової роботи обсяг пояснювальної записки враховується до додатків. Якщо додатки підтверджують цінність результату проектування, то обсяг пояснювальної записки з додатками повинен мати наскрізну нумерацію.

3.7 Перелік літературних джерел

«ПЕРЕЛІК ПОСИЛАНЬ» містить перелік літературних джерел, на які повинні бути обов'язкові посилання в тексті пояснювальної записки. Література (книги, статті, патенти, журнали, ..) в загальний список записується в порядку посилання на неї в тексті. Посилання на літературу наводять в квадратних дужках [...], вказуючи порядковий номер за списком.

Літературу записують мовою оригіналу. В списку кожну літературу записують з абзацу, нумерують арабськими цифрами, починаючи з одиниці.

Таблиця 3.1 – Приклади бібліографічного запису

Вид	Приклад запису
Статті	Марченко Б. Г. Преспективные подходы к созданию систем диагностики электротехнического оборудования / Б. Г. Марченко, М. В. Мыслович // Техническая электродинамика. – 1997. – № 2. – С. 49–52.
Стандарти	Технічне діагностування та контроль технічного стану. Терміни та визначення: ДСТУ 2389-94. – [Чинний від 01.01.95]. – К. : Держстандарт України, 1994. – 24 с. – (Національний стандарт України).
Монографії	Грабко В. В. Моделі та системи технічної діагностики високовольтних вимикачів : монографія / В. В. Грабко, Б. І. Мокін. – Вінниця : УНІВЕРСУМ-Вінниця, 1999. – 74 с.
Підручники	Зайцев Г. Ф. Теория автоматического управления и регулирования / Зайцев Г. Ф. – [2-е изд.]. – К. : Выща школа, 1989. – 431 с. Чиликин М. Г. Общий курс електропривода : учебник [для вузов] / М. Г Чиликин, А. С. Сандлер. – [6-е изд.]. – М. : Энергоиздат, 1981. – 576 с. Анхимюк В. Л. Теория автоматичаского управления / Анхимюк В. Л., Опейко О. Ф., Михеев Н. Н. – Мн. : Дизайн ПРО, 2000. – 352 с. Основы технической диагностики : в 2-х книгах / под ред. Пархоменко П. П. – М. : Энергия, 1976. Книга 1 : Модели объектов, методы и алгоритмы диагноза. – 1976. – 464 с.
Патенти	Пат. № 24523 Україна. МПК G 06 F 15/00. Пристрій для параметричного діагностування інформаційних систем / Кулік А. С.; Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут». – № u200612145 ; заявл. 20.11.2006 ; опубл. 10.07.2007 , Бюл. № 10.

Продовження таблиці 3.1

Вид	Приклад запису
Електронні ресурси	Логические модули LOGO! с модулями расширения. [Електронний ресурс]. – Режим доступу : http://automation-drives.ru/as/products/microsystems/logo/

3.8 Додатки

Матеріал, що доповнює текст пояснювальної записки, допускається поміщати в додатках. Додатками можуть бути, наприклад, графічний матеріал, таблиці великого формату, розрахунки, описи апаратури і приладів, описи алгоритмів і програм задач, що вирішуються на ЕОМ тощо.

У тексті пояснювальної записки на всі додатки повинні бути дані посилання. Додатки розташовують у порядку посилань на них у тексті пояснювальної записки, за винятком технічного завдання, яке є першим додатком курсової роботи.

Посилання на додатки в тексті пояснювальної записки дають за формою: *«... наведено в додатку А»*, *«... наведено в таблиці В.5»* або (*додаток Б*); (*додатки К, Л*).

Кожен додаток необхідно починати з нової сторінки, вказуючи зверху посередині рядка слово *«Додаток»* (малими літерами з першої великої) і через пропуск його позначення. Додатки позначають послідовно великими літерами українського алфавіту, за винятком літер Ґ, Є, З, І, Ї, Й, О, Ч, Ь, наприклад, *Додаток А*, *Додаток Б* і т. д. Якщо додатків більше ніж літер, то продовжують позначати арабськими цифрами. Дозволяється позначати додатки латинськими літерами, за винятком літер I та O.

Під позначенням для обов'язкового додатку пишуть в дужках слово *(обов'язковий)*, а для інформативного – *(довідковий)*.

Кожен додаток повинен мати тематичний (змістовний) заголовок, який записують посередині рядка малими літерами, починаючи з великої.

Ілюстрації, таблиці, формули нумерують в межах кожного додатка, вказуючи його позначення: *«Рисунок Б.3 – Найменування»*; *«Таблиця В.5 – Найменування»* і т. п.

Нумерація аркушів документа і додатків, які входять до його складу, повинна бути наскрізною.

Всі додатки вносять до змісту, вказуючи номер, заголовок і сторінки, з яких вони починаються.

В окремих дисциплінах допускається принципові електричні, структурні, функціональні, монтажні схеми підшивати в записку як обов'язкові додатки. В цьому випадку перед схемою в записці розміщується окремий аркуш формату А4 з надписом в верхній частині посередині поля *«Додаток Б (обов'язковий)»*, а в середній частині аркуша пишеться назва схеми.

ЛІТЕРАТУРА

1. 8-bit Atmel with 8KBytes InSystem Programmable Flash [Електронний ресурс]. – Режим доступу : http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf
2. Програмные средства для микроконтроллеров AVR фирмы Atmel. [Електронний ресурс]. – Режим доступу до документа: http://www.gaw.ru/html.cgi/txt/soft/avr/start.htm
3. Колдаев В. Д. Основы алгоритмизации и программирования : учеб. пособие / Колдаев В. Д. ; под ред. проф. Л. Г. Гагариной. – М. : ИД «Форум»; ИНФРА-М, 2006. – 416 с.
4. Ревич Ю. В. Практическое програмирование микроконтроллеров Atmel AVR / Ревич Ю. В. – [2-е изд. испр.]. – СПб. : БХВ–Петербург, 2011. – 352 с.
5. Баранов В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, програмы / Баранов В. Н. – М. : Издательский дом «Додэка XXI», 2004. – 288 с.
6. Евстифеев А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / Евстифеев А. В. – М. : Издательский дом «Додэка-XXI», 2007. — 592 с.
7. Мортон Дж. Микроконтроллеры AVR. Вводный курс / Мортон Дж. ; пер. с англ. – М. : Издательский дом «Додэка-XXI», 2006. – 272 с.
8. Гребнев В. В. Микроконтроллеры семейства AVR фирмы Atmel / Гребнев В. В. – М. : ИП Радиософт, 2002 – 176 с.
9. Белов А. В. Самоучитель разработчика устройств на микроконтроллерах AVR / Белов А. В. – СПб. : Наука и Техника, 2008. – 544 с.
10. Хартов В. Я. Микроконтроллеры AVR. Практикум для начинающих / Хартов В. Я. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2007. – 240 с.
11. Шпак Ю. А. Програмирование на языке С для микроконтроллеров AVR и PIC. Практикум для начинающих / Шпак Ю. А. – К. : «МК-Пресс», 2006. – 400 с.

ГЛОСАРІЙ

Мікропроцесор	microprocessor
Мікроконтролер	microcontroller
Мікропроцесорний пристрій	microprocessor appliance
Мікропроцесорна система	microprocessor system
Схема електрична структурна	structural electrical scheme
Схема електрична принципова	principle electrical scheme
Алгоритм	algorithm
Програма	program
Перетворювач	converter
Підпрограма	subprogram
Проект	project
Налагодження	debug
Середовище розробки	development environment
Компілятор	compilator
Генератор коду	code generator
Порт	port
Сигнал	signal
Цикл	loop
Умова	condition
Моделювання	modeling
Ініціалізація	initialization
Регістр	register
Затримка часу	time delay
Стек	stack
Вхід	input
Вихід	output
Функція	function
Брязкіт контактів	contact bounce
Струм	current

Додаток А
Завдання на курсову роботу

А.1 Програмний підсумовувальний лічильник

Початковий стан програмного лічильника дорівнює нулю. При натисканні кнопки, підключеної до вивода порту мікроконтролера стан лічильника збільшується на одиницю, після чого здійснюється блокування кнопки на час затримки. Після закінчення часу затримки процес може повторюватися. При досягненні лічильником значення N відбувається його скидання в нуль. Кількість натискань, підрахована лічильником, відображається на дворозрядному семисегментному індикаторі.

Варіанти завдань наведені в табл. А.1.

Таблиця А.1 – Варіанти завдань

Варіант	Тип мікроконтролера	Опорне значення, N	Час заримки, t_3, мс
1	ATmega8	20	50
2	ATmega8	25	100
3	ATmega8	15	80
4	ATmega8	12	90
5	ATmega8	14	40
6	ATmega8	22	30
7	ATmega8	24	50
8	ATmega8	26	100
9	ATmega8	10	80
10	ATmega8	11	90
11	ATmega8	23	40
12	ATmega8	26	30
13	ATmega8	21	90
14	ATmega8	27	40
15	ATmega8	13	30

А.2 Програмний реверсивний лічильник

Початковий стан програмного лічильника дорівнює N. При натисканні кнопки, підключеної до порту мікроконтролера, стан лічильника зменшується на одиницю, після чого на час затримки здійснюється блокування кнопки. Після закінчення часу затримки процес повторюється. При досягненні лічильником значення 0 в нього завантажується число N. Вміст лічильника відображається на дворозрядному семисегментному індикаторі.

Варіанти завдань подані в табл. А2.

Таблиця А.2 – Варіанти завдань

Варіант	Тип мікроконтролера	Початкове число, N	Час заримки, $t_з$, мс
1	ATmega8535	23	6
2	ATmega8535	26	11
3	ATmega8535	21	9
4	ATmega8535	27	7
5	ATmega8535	13	8
6	ATmega8535	20	10
7	ATmega8535	25	2,5
8	ATmega8535	15	3,5
9	ATmega8535	12	4,5
10	ATmega8535	14	5,5
11	ATmega8535	22	2
12	ATmega8535	24	4
13	ATmega8535	26	5
14	ATmega8535	10	1
15	ATmega8535	11	3

А.3 Програмний формувач аналогових сигналів

Програмний формувач аналогових сигналів складної форми являє собою пристрій, що генерує в часі сигнал заданої форми. Вихідний аналоговий сигнал може керувати виконавчим пристроєм, що, в свою чергу, може регулювати деякий фізичний параметр (температуру, тиск тощо). На рис. А.1–А.4 зображені різні форми напруг, які потрібно сформувати системою на основі мікроконтролера. Для виконання завдання потрібно використати ЦАП. Варіанти завдань подані в табл. А.3.

Таблиця А.3 – Варіанти завдань

Варіант	Тип мікроконтролера	Рис.	t_1, с	t_2, с	$t_П$, с	U_{max}, В	U_{min}, В
1	ATmega8535	А.1	6	0	2	5	1
2	ATmega8535	А.1	7	0	3	5	2
3	ATmega8535	А.1	1	0	1	4	1
4	ATmega8535	А.1	2	0	1	4	2
5	ATmega8535	А.2	8	0	2	5	1
6	ATmega8535	А.2	9	0	3	5	2
7	ATmega8535	А.2	6	2	3	4	1
8	ATmega8535	А.3	10	8	2	5	1
9	ATmega8535	А.3	2	5	3	5	2
10	ATmega8535	А.3	5	3	4	4	1
11	ATmega8535	А.4	3	12	2	5	1
12	ATmega8535	А.4	4	13	3	5	2
13	ATmega8535	А.4	5	4	2	5	1
14	ATmega8535	А.4	6	3	3	5	2
15	ATmega8535	А.4	4	5	4	4	2

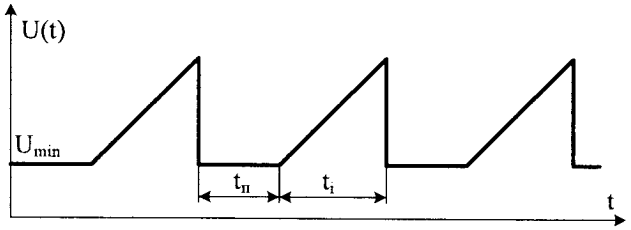


Рисунок А.1 – Форма сигналу

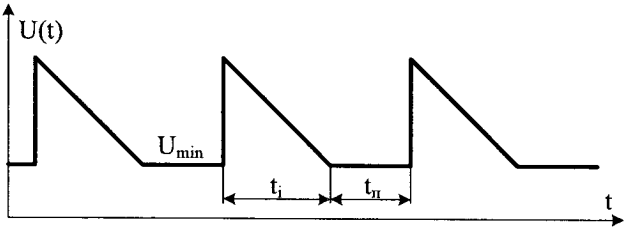


Рисунок А.2 – Форма сигналу

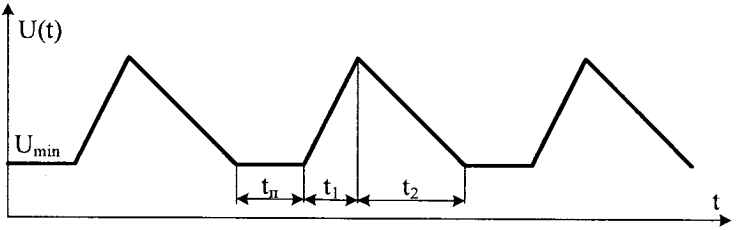


Рисунок А.3 – Форма сигналу

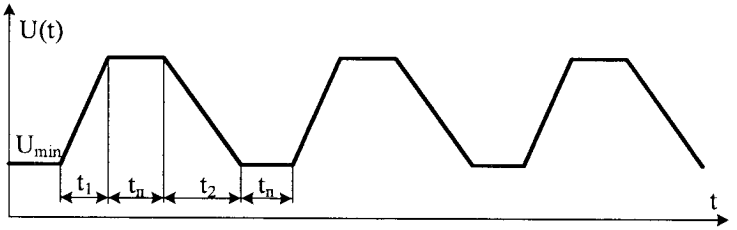


Рисунок А.4 – Форма сигналу

А.4 Кодовий замок

Реалізувати алгоритм пристрою «кодовий замок» із кодом відповідно до варіанта у табл. А.4. Код формується за допомогою першої кнопки «кількістю натискань», вводиться за допомогою другої кнопки. Кількість натискань першої кнопки повинна відображатися на семисегментному індикаторі. Введення неправильної кодової комбінації має відображатися виведенням помилки на дисплеї.

Таблиця А.4 – Варіанти завдань

Варіант	Тип мікроконтролера	Кодова комбінація
1	ATmega8535	1357
2	ATmega8535	2468
3	ATmega8535	1345
4	ATmega8535	1423
5	ATmega8535	5623
6	ATmega8535	6734
7	ATmega8535	4387
8	ATmega8535	4365
9	ATmega8535	1322
10	ATmega8535	9867
11	ATmega8535	3424
12	ATmega8535	1314
13	ATmega8535	2325
14	ATmega8535	2526
15	ATmega8535	2627

А.5 Пристрій індикації

Пристрій складається з мікроконтролера, семисегментного індикатора, матричної клавіатури. При утримуванні деякої кнопки клавіатури відповідна цифра має відображатися на індикаторі миганням із частотою, вказаною в табл. А.5.

Таблиця А.5 – Варіанти завдань

Варіант	Тип мікроконтролера	Частота мигання, Гц
1	ATmega8535	1
2	ATmega8535	2
3	ATmega8535	3
4	ATmega8535	4
5	ATmega8535	1,5
6	ATmega8535	2,5
7	ATmega8535	3,5
8	ATmega8535	4,5
9	ATmega8535	0,5
10	ATmega8535	6
11	ATmega8535	5
12	ATmega8535	6,5
13	ATmega8535	5,5
14	ATmega8535	0,25
15	ATmega8535	0,75

А.6 Програмний підсумовувальний лічильник з завантаженням граничного значення

Граничне значення програмного лічильника завантажується за допомогою кнопок «Більше», «Менше» та «Ввести». При натисканні кнопки «Збільшити» стан лічильника збільшується на одиницю, після чого здійснюється блокування кнопки на час затримки. Після закінчення часу затримки процес може повторюватися. При досягненні лічильником введеного значення відбувається його скидання в нуль. Кількість натискань, підрахована лічильником, відображається на дворозрядному семисегментному індикаторі.

Варіанти завдань подані в табл. А.6.

Таблиця А.6 – Варіанти завдань

Варіант	Тип мікроконтролера	Час заримки, $t_з$, мс
1	ATmega8	50
2	ATmega8	100
3	ATmega8	80
4	ATmega8	90
5	ATmega8	40
6	ATmega8	30
7	ATmega8	50
8	ATmega8	100
9	ATmega8	80
10	ATmega8	90
11	ATmega8	40
12	ATmega8	30
13	ATmega8	90
14	ATmega8	40
15	ATmega8	30

А.7 Програмний реверсивний лічильник з завантаженням початкового значення

Початкове значення програмного лічильника завантажується за допомогою кнопок «Більше», «Менше» та «Ввести». При натисканні кнопки «Зменшити» стан лічильника зменшується на одиницю, після чого здійснюється блокування кнопки на час затримки. Після закінчення часу затримки процес може повторюватися. При досягненні лічильником нульового значення відбувається його скид (завантаження початого значення). Кількість натискань, підрахована лічильником, відображається на дворозрядному семисегментному індикаторі.

Варіанти завдань подані в табл. А.7.

Таблиця А.7 – Варіанти завдань

Варіант	Тип мікроконтролера	Час затримки, $t_з$, мс
1	ATmega8	50
2	ATmega8	100
3	ATmega8	80
4	ATmega8	90
5	ATmega8	40
6	ATmega8	30
7	ATmega8	50
8	ATmega8	100
9	ATmega8	80
10	ATmega8	90
11	ATmega8	40
12	ATmega8	30
13	ATmega8	90
14	ATmega8	40
15	ATmega8	30

А.8 Генератор імпульсів

Розробити генератор імпульсів на базі мікроконтролера, на виході якого формується сигнал прямокутної форми із заданим коефіцієнтом заповнення (D). Частота на виході генератора може набувати трьох значень (f1, f2, f3), які перемикаються кнопкою і сигналізуються відповідним світлодіодом. Вихідні дані наведені в табл. А.8.

Таблиця А.8 – Варіанти завдань

Варіант	Тип мікроконтролера	f1, Гц	f2, Гц	f3, Гц	D, %
1	ATmega8535	10	20	30	50
2	ATmega8535	50	100	150	40
3	ATmega8535	20	10	5	30
4	ATmega8535	12	20	35	20
5	ATmega8535	75	45	20	10
6	ATmega8535	35	50	87	60
7	ATmega8535	75	25	10	70
8	ATmega8535	15	20	35	80
9	ATmega8535	100	200	300	90
10	ATmega8535	15	30	60	15
11	ATmega8535	24	12	6	25
12	ATmega8535	40	18	45	35
13	ATmega8535	150	250	350	45
14	ATmega8535	80	50	20	55
15	ATmega8535	2	4	6	65

А.9 Подільник частоти

Розробити керований подільний частоти на базі мікроконтролера, на вхід якого подається сигнал прямокутної форми із фіксованою частотою. Коефіцієнт ділення частоти задається двійковим n-розрядним кодом, який формується DIP-перемикачем. Передбачити кнопку, при утримуванні якої встановлюється заданий коефіцієнт ділення (k). Вихідні дані наведені в табл. А.9.

Таблиця А.9 – Варіанти завдань

Варіант	Тип мікроконтролера	Розрядність коду керування n	Коефіцієнт ділення k
1	ATmega8535	2	2
2	ATmega8535	3	3
3	ATmega8535	4	4
4	ATmega8535	5	5
5	ATmega8535	6	6
6	ATmega8535	7	7
7	ATmega8535	8	8
8	ATmega8535	2	9
9	ATmega8535	3	10
10	ATmega8535	4	11
11	ATmega8535	5	12
12	ATmega8535	6	13
13	ATmega8535	7	14
14	ATmega8535	8	15
15	ATmega8535	2	16

А.10 Таймер

Розробити керований таймер на базі мікроконтролера, який формує витримку часу на виході за відповідною командою на вході його запуску. Можливі три варіанти функціонування таймера:

1) таймер із затримкою на ввімкнення (TON), такий таймер формує затримку часу (t) появи логічної одиниці на виході після її появи на вході;

2) таймер із затримкою на вимкнення (TOFF), такий таймер формує затримку часу (t) утримування сигналу логічної одиниці на виході після того, як на вході сигнал логічної одиниці змінився на сигнал логічного нуля;

3) таймер ввімкнення (TT), такий таймер формує затримку часу (t) утримування сигналу логічної одиниці на виході після того, як на вході сигнал логічного нуля змінився на сигнал логічної одиниці.

Задавання затримки часу t здійснюється за допомогою двох кнопок (збільшення та зменшення), які ступінчасто змінюють затримку часу t через рівні проміжки t1. Кількість ступенів m, передбачити світлодіодну індикацію поточного значення ступеня затримки. Вихідні дані наведені в табл. А.10.

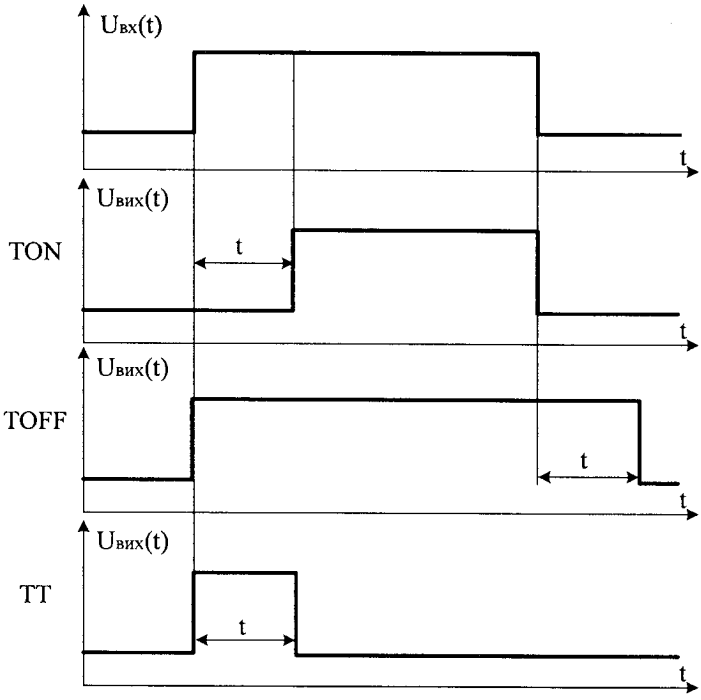


Рисунок А.5 – Діаграми роботи таймерів

Таблиця А.10 – Варіанти завдань

Варіант	Тип мікроконтролера	Тип таймера	t1, с	m
1	ATmega8535	TON	0,1	3
2	ATmega8535	TOFF	0,2	4
3	ATmega8535	TT	0,3	5
4	ATmega8535	TON	0,5	6
5	ATmega8535	TOFF	0,8	8
6	ATmega8535	TT	1	3
7	ATmega8535	TON	2	4
8	ATmega8535	TOFF	3	5
9	ATmega8535	TT	4	6
10	ATmega8535	TON	5	8
11	ATmega8535	TOFF	6	3
12	ATmega8535	TT	7	4
13	ATmega8535	TON	8	5
14	ATmega8535	TOFF	9	6
15	ATmega8535	TT	10	8

Додаток Б
Зразки оформлення ключових сторінок

Вінницький національний технічний університет
Кафедра електромеханічних систем автоматизації в промисловості і на транспорті

КУРСОВА РОБОТА

з дисципліни «Обчислювальна техніка і програмування»
на тему: «Розробка програми для мікропроцесорного пристрою»

Студента _____ курсу _____ групи
спеціальності _____
_____
_____

_____
(прізвище та ініціали)

Керівник _____
(посада, вчене звання)
_____
(науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____ Оцінка: ECTS ___

Члени комісії

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Вінниця – 20__ рік

Рисунок Б.1 – Зразок титульного аркуша

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет електроенергетики та електромеханіки

ЗАТВЕРДЖУЮ

Завідувач кафедри ЕМСАПТ

_____

"___" _____ 20__ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу з дисципліни «Обчислювальна техніка і програмування»

студенту _____ групи _____

Розробити програму для мікропроцесорного пристрою за такими вхідними даними:

– тип мікроконтролера ATMEGA8;

– семисегментний індикатор;

– дві кнопки;

– реалізувати «кодовий замок» із кодом 1, 2, 3, 4 (код формується за допомогою першої кнопки «кількістю натискань», вводиться за допомогою другої кнопки).

ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

1. Огляд алгоритмів, схемних та програмних рішень поставленої задачі.
2. Розробка схем структурної, принципової та алгоритму програми.
3. Розробка програми мовою програмування Assembler.
4. Розробка програми мовою програмування С та моделювання в Proteus VSM.

Рисунок Б.2 – Зразок індивідуального завдання (лицьова сторона)

КАЛЕНДАРНИЙ ПЛАН

№ етапу	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання		
2	Огляд алгоритмів, схемних та програмних рішень поставленої задачі		
3	Розробка структурної схеми		
4	Розробка принципової схеми		
5	Розробка програми мовою програмування Assembler		
6	Розробка програми мовою програмування С		
7	Моделювання роботи пристрою в Proteus VSM		
8	Здача готової роботи		
9	Захист курсової роботи	за графіком	

Дата видачі «____» _____ 20__ р.

Керівник роботи _____ _____

(підпис) (ПІБ)

Завдання отримав _____ _____

(підпис) (ПІБ)

Рисунок Б.3 – Зразок індивідуального завдання (зворотна сторона)

АНОТАЦІЯ

_____ «Розробка програми для мікропроцесорного пристрою». Курсова робота. – Вінниця : ВНТУ. 20__.– __ с. Бібліогр. : __. Іл. : __. Табл. : __.

Розглянуто структуру мікроконтролера ATMEGA8. На основі індивідуального завдання запропоновано структурну та принципову схеми мікропроцесорного пристрою, розроблено алгоритм роботи пристрою, реалізовано алгоритм мовами програмування Assembler та C, доведено відповідність роботи реалізованого алгоритму вимогам індивідуального завдання шляхом моделювання. Для перевірки моделювання використовувалася програма Proteus VSM.

Ключові слова: мікроконтролер, алгоритм роботи, програма роботи.

Рисунок Б.4 – Зразок анотації

Навчальне видання

Шевчук Юрій Володимирович
Проценко Дмитро Петрович
Бабій Сергій Миколайович

ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ КУРСОВЕ ПРОЕКТУВАННЯ САМОСТІНА ТА ІНДИВІДУАЛЬНА РОБОТА СТУДЕНТІВ

Навчальний посібник

Редактор Т. Старічек

Оригінал-макет підготовлено Ю. Шевчуком

Підписано до друку 25.04.2017 р.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Ум. друк. арк. 3,86.
Наклад 50 пр. Зам. № 2017-067.

Видавець та виготовлювач
Вінницький національний технічний університет,
інформаційний редакційно-видавничий центр.

ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 59-85-32, 59-87-38,
press.vntu.edu.ua,
e-mail: kivc.vntu@gmail.com.

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