

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
з дисципліни
МІКРОПРОЦЕСОРНІ ПРИСТРОЇ КЕРУВАННЯ
І ОБРОБКИ ІНФОРМАЦІЇ
для студентів спеціальності
6.050801 „Мікро- та наноелектроніка“
денної й заочної форм навчання

2010

Методичні вказівки до лабораторних робіт з дисципліни „Мікропроцесорні пристрої керування і обробки інформації“ для студентів спеціальності 6.050801 „Мікро- та наноелектроніка“ денної й заочної форм навчання / Укл.: Є.Л. Жавжаров, В.І.Грядун. – Запоріжжя: ЗНТУ, 2010. – 82 с.

Укладачі: Є. Л. Жавжаров, доц., канд.фіз.-матем.наук
В.І. Грядун, ст. викладач

Рецензент: В. В. Погосов, проф., д-р фіз.-матем. наук

Відповідальний за випуск: Г.В.Сніжной, доц., канд.фіз.-матем.наук

Затверджено
на засіданні кафедри
мікро- та наноелектроніки
Протокол №2
від 18 жовтня 2010 р.

ЗМІСТ

Підготовка до проведення лабораторної роботи та оформлення звіту.	4
1 Лабораторна робота №1 “Мікроконтролери AVR фірми Atmel. Засоби розробки програмного забезпечення”.	5
2 Лабораторна робота №2 “Система переривань. Таймери ATmega”.	11
3 Лабораторна робота №3 Засоби вводу-виводу інформації в МПС. LED-індикатори	21
4 Лабораторна робота №4 Робота з LCD-індикаторами на базі контролеру HD44780”.	31
5 Лабораторна робота №5 “Програмування внутрішнього АЦП ATmega”.	40
6 Лабораторна робота №6 “Робота з периферійними пристроями по I-wire – шині”	47
7 Лабораторна робота №7 “Послідовний ввід – вивід. UART”. ..	58
8 Лабораторна робота №8 “Керування кроковим двигуном за допомогою МПС”.	68
Додаток А Приклад оформлення титульної сторінки	75
Додаток Б Архітектура ATmega8.	76
Додаток В Текст програми на асемблері до лабораторної роботи №1.	77
Додаток Г Підпрограми для реалізації протоколу 1-Wire шини .	79

ПІДГОТОВКА ДО ПРОВЕДЕННЯ ЛАБОРАТОРНОЇ РОБОТИ ТА ОФОРМЛЕННЯ ЗВІТУ

Лабораторна робота включає самостійне опрацювання теоретичного матеріалу, вивчення документації з апаратних засобів, методик проведення розрахунків, програмування і моделювання, обробку й інтерпретацію результатів. При проведенні лабораторного практикуму необхідно:

- підготуватися до експрес-опитування з теоретичного матеріалу, необхідного для виконання роботи (щоб отримати допуск до лабораторної роботи);

- підготувати і оформити план виконання лабораторної роботи;

- після виконання лабораторної роботи провести необхідні розрахунки і оформити звіт, який повинен містити:

- 1) титульний аркуш (див. Додаток А);

- 2) мету роботи;

- 3) опис словами;

- 4) алгоритм роботи у вигляді блок схеми;

- 5) необхідні математичні розрахунки роботи пристрою;

- 6) текст розробленої програми;

- 7) основні результати моделювання пристрою;

- 8) стислі висновки по роботі, в яких:

- вказати, що досліджувалось;

- вказати методику (чи метод) дослідження;

- вказати особливості алгоритму роботи пристрою;

- визначити характер роботи пристрою при моделюванні;

- відзначити подібність і відмінність результатів моделювання від теоретичних даних;

- здати звіт з виконаної лабораторної роботи та відповісти на запитання викладача (отримати залік за виконану роботу).

1 ЛАБОРАТОРНА РОБОТА №1

„МІКРОКОНТРОЛЕРИ AVR ФІРМИ ATMEL. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ”

Мета роботи – ознайомитися з сімейством 8-розрядних мікроконтролерів фірми Atmel і отримати навички розробки програм для мікроконтролерів AVR.

1.1 Мікроконтролери фірми Atmel

Характерними особливостями сімейства мікроконтролерів AVR є:

- наявність великої кількості груп мікроконтролерів (МК), орієнтованих на виконання певних специфічних функцій, наприклад, робота з USB ("USB AVR"), CAN ("CAN AVR") і т. п.;
- конвеєр команд забезпечує їх виконання у більшості випадків за 1 машинний такт (що при тактовій частоті 20 МГц складає 50 нс);
- велика кількість регістрів загального призначення (32) дозволяє іноді взагалі не звертатися до оперативної пам'яті;
- загальна велика кількість команд на асемблері (90÷130) дозволяє складати короткі програми, які за структурою нагадують програми на мовах високого рівня;
- наявність безкоштовного програмного забезпечення для розробки програм як на асемблері, так і на СІ;
- можливість внутрішнього програмування контролерів.

Мікроконтролери AVR мають:

- удосконалену RISC-архітектуру;
- продуктивність на рівні 1 MIPS (millions of instructions per second);
- гарвардську архітектуру (на відміну від архітектури фон Неймана мають роздільні області пам'яті і даних, а, отже, і шини даних і команд);
- 32 регістри загального призначення;
- Flash-пам'ять програм з можливістю внутрішнього програмування в готовому пристрої (витримує 10 тис. циклів перезапису);
- окрему енергонезалежна пам'ять (EEPROM, 100 тис. циклів) для збереження даних з можливістю запису програмно або через SPI-інтерфейс;

- вбудований на кристалі 10-розрядний АЦП і аналоговий компаратор;
- сторожовий таймер, який дозволяє проводити автоматичне скидання контролера через певні проміжки часу;
- послідовні інтерфейси: SPI, TWI (I²C), UART (USART);
- таймери-лічильники 8- і 16-розрядні, з можливістю працювати в режимі багатоканальної 8-, 9-, 10- та 16-бітової ШІМ (PWM);
- тактові частоти від 0 до 20 МГц;
- діапазон робочих напруг $2,7 \div 5,5$ В (або $1,8 \div 6,0$ В);
- численні режими енергозбереження;
- вбудований монітор живлення (Brown out detection).

Загальну структуру МК наведено на рис.Б.1. Периферійними (зовнішніми стосовно ядра і пам'яті) у структурі AVR є таймери, порти, порт UART, аналоговий компаратор, сторожовий таймер і т.п. Особливістю цих пристроїв є те, що вони адресуються через регістри вводу-виводу (PBB). В AVR це виконується через проміжне занесення числа в РЗП і наступний запис з РЗП в PBB (команди *in*, *out*).

Для зменшення кількості виводів зовнішні виводи портів виконують ще й альтернативні функції (виводи АЦП тощо). Усі порти є двонаправленими, тобто можуть виконувати функції як вводу, так і виводу. Робота порту визначається установкою спеціальних бітів у регістр напрямку даних DDRx (data direction register), де x – літера порту. Щоб записати дані в порти, їх треба записати в регістри даних портів PORTx. У режимі виводу запис «0» чи «1» в PORTx визначає його стан: стан з високим імпедансом чи підключено підтягуючий резистор (pull-up) $35 \div 120$ кОм.

Для читання стану портів є спеціальний регістр PINx, з якого можна зчитувати інформацію.

1.2 Програмне забезпечення. AVR Studio

AVR Studio – це інструмент розробки для сімейства AVR мікроконтролерів, який дозволяє розробляти програми на вбудованому Atmel AVR Assembler.

AVR Studio дає можливість користувачу керувати виконанням програм на вбудованому AVR Instruction Set Simulator або на AVR In-

Circuit емуляторі. AVR Studio підтримує програми, написані на рівні Atmel Corporation's AVR Assembler й інших асемблерах і компіляторах, які підтримують формати UBROF і COFF об'єктного файлу.

1.2.1 Створення нового проекту

Перед створенням нового проекту необхідно створити папку з ім'ям "LAB_X" на вашому C: диску, у якій розмістити файл з ім'ям "m8def.inc" - це установлювальний файл для ATmega8, який містить установки МК, потрібні для компілятора. Цей файл буде відкритий в Application Note AVR000.

Крок 1 - запуск AVR Studio здійснюється так:

Пуск → Всі програми → Atmel AVR Tools → AVR Studio 4.

Крок 2 - вибір нового проекту. Створимо просту програму, що збільшує значення одного з РЗП на 1, тобто створимо лічильник.

Для створення нового проекту виберіть "Project" меню і виберіть "New". З'явиться діалогове вікно, у якому необхідно ввести ім'я проекту, наприклад test_LR1.

Далі оберемо розташування нашого проекту. Це місце, де AVR Studio зберігатиме всі файли, пов'язані з проектом. Ми використовуємо папку "C:\Atmel\RP". Якщо папка не існує, AVR Studio автоматично її створить. Необхідно щоб шлях розміщення робочих файлів не мав рос./укр. літер і був сумісний з іменами DOS.

Далі обираємо тип проекту.

Atmel AVR Assembler - тип проекту, який повідомляє AVR Studio що використовується вбудований Assembler коли компілюється проект. Ніяких додаткових налаштувань від користувача не вимагається. Цей тип проекту будемо використовувати і надалі.

AVR GCC (Generic 3rd party C compiler) - тип проекту, який вимагає від користувача додаткового налаштування AVR Studio і використання зовнішнього компілятора.

Уведіть ім'я проекту (група, ПІБ, номер ЛР). Зверніть увагу на розміщення файлів проекту (location). Галочка в полі "create initial file" і "create folder" дозволяє автоматично створити початковий файл і папку для проекту.

Крок 3 – вибір платформи дебагера і типу контролера. Обираємо дебагер AVR Simulator, мікроконтролер - ATmega8 і натискаємо Finish. Файл RPXXX_LR1.asm (XXX – номер групи) створено AVR Studio і розміщено у вказану папку. Програма відкриває вікно для вве-

дення коду на мові асемблера.

Крок 4 – написання коду програми на мові асемблер. Текст програми з усіма необхідними структурними елементами наведено у Додатку Б. У нескінченному циклі для організації лічильника треба дописати лише один рядок:

INC R23 ; збільшити значення РЗП на 1.

Крок 5 - асемблювання вихідного коду. Це можна зробити, вибравши "Build" в меню " Build " або натиснувши на клавіатурі клавішу "F7". Повинно з'явитися вікно Build, яке показує інформацію про результати асемблювання. Код програми, наведений у Додатку Б, займає близько 1 % флеш-пам'яті ATmega8.

Далі виконуємо програмний код.

1.2.2 Виконання програми (початкового коду)

Для запуску симулятора програми обираємо "Build and run" меню "Build". AVR Studio запустить режим симулятора.

Користувач може виконувати програму повністю в покроковому режимі, трасуючи блоки функцій, або виконуючи програму до місця, де знаходиться курсор. Для відповідних дій на панелі інструментів є відповідні клавіші: *Step into, Run to cursor*.

Установка точок призупинення здійснюється так: Debug→New Breakpoint. Можна визначати необмежену кількість таких точок, кожна з яких може бути включена або виключена. Точки призупинення зберігаються між сесіями роботи.

Запис проекту здійснюється після закінчення роботи в AVR Studio. AVR Studio запам'ятає всі налаштування проекту і відновить їх, коли знову буде запущено цей самий проект. Для запису проекту необхідно вибрати пункт 'Save' в меню 'Project'.

1.3 Програмне забезпечення для моделювання. Proteus 7 Pro

Програма Proteus Professional є системою моделювання схематехніки і базується на моделях електронних компонентів, прийнятих в PSpice. Відмінною рисою пакету Proteus Professional є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP і т.п. Додатково в пакет Proteus Professional входить система проектування печатних плат.

Порядок роботи з програмою включає такі етапи:

1 Запустити програму:

Пуск → Усі програми → Proteus 7 Professional → Isis 7 professional.

2 Додати / обрати з бібліотеки компоненти для роботи:

Бібліотека → Обрати пристрій

3 Розмістити обрані компоненти на робочому полі:

1 клік лівою клав'яшею миші на переліку компонентів → Ліва клав'яша миші на робочому полі.

4 Надати компонентам потрібні властивості:

2 кліка лівою клав'яшею миші, встановити потрібні параметри (вихідний hex-файл отриманий в AStudio).

5 Розмістити на робочому полі потрібні інструменти.

6 Включити симуляцію.

1.4 Завдання до роботи

1.4.1 Ознайомтеся з програмним забезпеченням.

1.4.2 Використовуючи текст програми, наведений у Додатку В і паспортні дані (datasheet) на мікроконтролер, напишіть програму на мові асемблера, яка забезпечує мерехтіння світлодіоду, підключеному за нижченаведеною схемою з частотою: $F = (N \times 2) \text{ Гц}$.

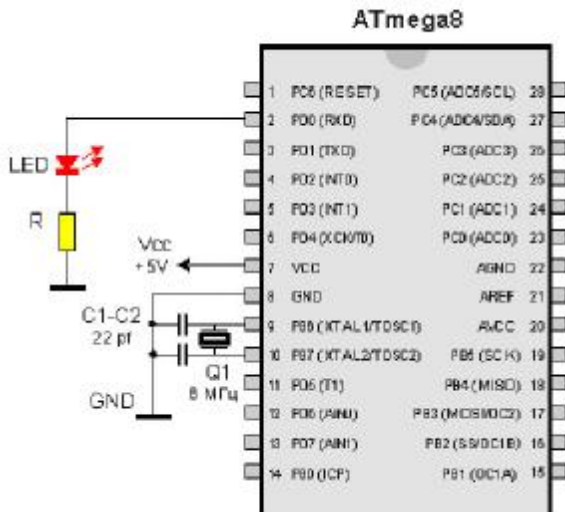


Рисунок 1.1 – Схема включення мікроконтролера

1.4.3 Оформити звіт, який має містити:

- опис розділів меню програми AVRStudio;
- основні характеристики мікроконтролера ATmega8;
- блок-схему алгоритму роботи програми;
- необхідні розрахунки;
- текст програми на асемблері з коментарями кожного рядка програми;
- результати моделювання у вигляді принскрінів (*print screens*);
- відповіді на контрольні запитання;
- висновки по роботі.

1.5 Контрольні запитання

- 1 Яку архітектуру мають МК сімейства AVR?
- 2 Які основні характеристики ATmega8?
- 3 Які види пам'яті є у ATmega8, чим вони відрізняються?
- 4 Що таке стек, яке його призначення?
- 5 Запишіть команди ініціалізації порту C для роботи на ввід із станом високого імпедансу.
- 6 Запишіть команди ініціалізації порту D для роботи на ввід при логічному «0» на виході.
- 7 Що таке таблиця переривань?
- 8 Які типи переривань підтримує ATmega8?
- 9 Як здійснюється затримка між виконанням двох команд?
- 10 Обчисліть час виконання наступних команд при тактовій частоті 8 МГц:

```

nop
ldi R25,$ff
ldi R24,0b00010001
Inc R24
ldi R18,R24
neg R18
out TCNT0,R18

```

2 ЛАБОРАТОРНА РОБОТА №2

„СИСТЕМА ПЕРЕРИВАНЬ. ТАЙМЕРИ АТМЕГА”

Мета роботи – ознайомитися із системою переривань мікроконтролерів сімейства ATmega; вивчити режими роботи таймерів і навчитися використовувати таймери для організації затримок.

2.1 Таймери

2.1.1 Таймери-лічильники

Будь-який мікроконтролер серії AVR має декілька вбудованих таймерів. За призначенням їх можна поділити на дві категорії: до *першої категорії* відносять таймери загального призначення, *друга категорія* - це сторожові таймери, призначені для автоматичного перезапуску мікроконтролера в разі «зависання» його програми.

Зависанням називають зациклення програми через помилку, допущену програмістом, або в результаті дії зовнішньої перешкоди.

Для кожної мікросхеми потрібен всього один сторожовий таймер. У будь-якому мікроконтролері AVR такий таймер є.

Таймери загального призначення використовують для формування різних інтервалів часу і прямокутних імпульсів заданої частоти. Також вони можуть працювати в режимі лічильника і підраховувати тактові імпульси заданої частоти, вимірюючи тривалість зовнішніх сигналів, та за необхідності підраховувати кількість будь-яких зовнішніх імпульсів, через що ці таймери називають *таймери/лічильники*.

У мікросхемах AVR застосовують як восьмирозрядні, так і шістнадцятирозрядні таймери/лічильники. Їх кількість для різних МК - від одного до чотирьох. Точну кількість таймерів/лічильників для кожної мікросхеми серії AVR можна визначити у оригінальній документації виробника. Всі таймери позначаються числами від 0 до 3.

Кожним восьмирозрядним таймером є один восьмирозрядний регістр, який для мікроконтролера є регістром вводу-виводу. Цей регістр зберігає поточне значення таймера і має назву *рахунковий регістр*. Шістнадцятирозрядні таймери мають шістнадцятирозрядний рахунковий регістр. Кожен рахунковий регістр має своє ім'я.

Рахунковий регістр восьмирозрядного таймера іменують TCNTx, де «x» - номер таймера. Для таймера T0 регістр називають TCNT0, для таймера T2 - TCNT2. Шістнадцятирозрядні регістри іменують подібно.

Кожен шістнадцятирозрядний рахунковий регістр для мікрокон-

тролера є двома регістрами вводу-виводу. Один призначений для зберігання старших бітів числа, другий - для зберігання молодших бітів. До імені регістра старших розрядів додають букву H, а до імені регістра молодших розрядів - букву L. Отже, рахунковий регістр таймера T1 - це два регістри вводу-виводу: TCNT1H і TCNT1L, рахунковий регістр таймера T3 - це регістри TCNT3H і TCNT3L.

Мікроконтролер може записати в будь-який рахунковий регістр будь-яке число у будь-який момент часу, а також у будь-який момент прочитати вміст будь-якого рахункового регістра. Коли таймер включається в режим рахунку, то на його вхід поступають рахункові імпульси. Після приходу кожного такого імпульсу вміст рахункового регістра збільшується на одиницю. Рахунковими імпульсами можуть служити як спеціальні тактові імпульси, які виробляються усередині самого мікроконтролера, так і зовнішні імпульси, що поступають на спеціальні входи мікросхеми. При переповненні рахункового регістра його вміст обнулюється, і рахунок починається спочатку.

Будь-який таймер жорстко зав'язаний з системою переривань. Викликати переривання може ціла купа подій, пов'язаних з таймером. Наприклад, існує переривання через переповнення таймера, через спрацювання спеціальної схеми збігу. Окремі переривання може викликати сторожовий таймер.

2.1.2 Режими роботи таймерів

Таймери мікроконтролерів сімейства AVR можуть працювати в декількох режимах. Різні мікроконтролери мають різні набори режимів для своїх таймерів. Для вибору режимів роботи є спеціальні регістри - регістри управління таймерами. Для простих таймерів використовується один регістр управління, для більш складних - два регістри. Регістри управління таймером називають TCCR_x (де «x» - номер таймера). Наприклад, для таймера T0 використовується один регістр з ім'ям TCCR0. Для управління таймером T1 використовується два регістри: TCCR1A і TCCR1B. За допомогою регістрів управління проводиться не лише вибір відповідного режиму, але і тонше налаштування таймера. Нижче наведені основні режими роботи таймера та їх опис.

2.1.2.1 Режим Normal. Це найпростіший режим. У цьому режимі таймер підраховує імпульси (від тактового генератора або зовнішнього пристрою), що приходять на його вхід, і викликає переривання

по переповнюванню. Цей режим є єдиним режимом роботи для восьмирозрядних таймерів більшості МК сімейства «Tіny» і для частки МК сімейства «Mega». Для решти восьмирозрядних і всіх шістнадцятирозрядних таймерів це лише один з можливих режимів.

2.1.2.2 Режим «Захвату». Суть цього режиму полягає в збереженні вмісту рахункового регістра таймера в певний момент часу. Запам'ятовування відбувається або за сигналом, що поступає через спеціальний вхід МК, або за сигналом з виходу вбудованого компаратора.

Цей режим зручний у випадку, коли треба виміряти тривалість якогось зовнішнього процесу. *Наприклад*, час, за який напруга на конденсаторі досягне певного значення. У цьому випадку напруга з конденсатора подається на один з входів компаратора, а на другий його вхід подається опорна напруга.

Мікроконтролер має одночасно запустити два процеси: подати напругу на конденсатор і запустити таймер в режимі Capture.

Конденсатор почне заряджатися, напруга на ньому плавно зростатиме. Одночасно лічильник таймера відлічуватиме тактові імпульси заданої частоти. У момент, коли напруга на конденсаторі порівняється з опорною напругою, логічний рівень на виході компаратора зміниться на протилежний. За цим сигналом поточне значення рахункового регістра запам'ятовується в спеціальному *регістрі захвату*. Ім'я цього регістра ICRx (для таймера T0 це буде ICR0, для T1 - ICR1 і т. д.). Одночасно виробляється запит на переривання.

Завдяки принципу вимірювання часу зарядки зручно створювати прості схеми, що працюють з різними аналоговими датчиками (температури, тиску тощо). Якщо принципом роботи датчика є зміна його внутрішнього опору, то такий датчик можна включити в ланцюг зарядки конденсатора. Ємнісні датчики можна підключати безпосередньо.

2.1.2.3 Режим «Скидання при збігу». Для роботи в цьому режимі використовують спеціальний регістр - *регістр збігу*. Якщо мікроконтролер містить декілька таймерів, то для кожного з них є свій окремий регістр збігу. Причому для восьмирозрядних таймерів регістр збігу - це один восьмирозрядний регістр, для шістнадцятирозрядних - це два восьмирозрядні регістри.

Регістри порівняння також мають свої імена. Наприклад, регістр

збігу таймера T1 складається з двох регістрів: OCR1L і OCR1H. У деяких мікроконтролерів існують два регістри збігу. Так, у всіх мікроконтролерах сімейства «Tiny» існує два регістри збігу для таймера T1. Це регістри OCR1A і OCR1B. Два регістри збігу для таймера T1 має і мікроконтролер ATmega8х. У другому випадку як таймер, так і його регістри збігу мають шістнадцять розрядів.

Якщо регістр збігу шістнадцатирозрядний, то фізично він складається з двох регістрів вводу-виводу. Наприклад, двома регістрами збігу таймера T1 мікросхеми ATmega8х є чотири регістри вводу-виводу з іменами OCR1AL, OCR1AH, OCR1BL, OCR1BH.

Як же використовуються регістри збігу? Ці регістри включаються в роботу тільки тоді, коли вибраний режим скидання при збігу. У цьому режимі, як і в попередньому, таймер підраховує вхідні імпульси. Поточне значення таймера з його рахункового регістра постійно порівнюється з вмістом регістрів збігу.

Якщо таймер має два регістри збігу, то для кожного з цих регістрів проводиться окреме порівняння. Коли вміст рахункового регістра збіжиться з вмістом одного з регістрів збігу, станеться виклик відповідного переривання. Окрім виклику переривання у момент збігу може відбуватися одна з наступних подій:

- скидання таймера (вірно тільки для регістрів збігу OCR1 і OCR1A);
- зміна стану одного з виводів мікроконтролера (вірно для всіх регістрів).

Станеться або не станеться одна або обидві події з вищенаведених, визначається налаштуваннями таймера.

2.1.2.4 Режим «Швидкодіючий ШІМ» (Швидкий PWM). ШІМ - широтно-імпульсна модуляція (англ. Puls Width Modulation, PWM). Сигнал з ШІМ часто використовується в пристроях управління.

Для формування сигналу ШІМ використовуються ті ж самі регістри збігу, які працюють і в режимі скидання при збігу. Формування сигналу ШІМ може здійснюватися декількома різними способами. Робота таймера в режимі Fast PWM проілюстрована на рис. 2.1.

Сигнал з ШІМ формується на спеціальному виході мікроконтролера. На вхід таймера подаються імпульси від системного генератора. Таймер перебуває в стані безперервного рахунку. При переповненні таймера його вміст скидається в нуль, і рахунок починається

спочатку. У режимі ШІМ переповнювання таймера не викликає переривань, на рис.2.1 це показано у вигляді пилоподібної кривої, позначеної **TCNTn**, яка є залежністю вмісту рахункового регістра від часу.

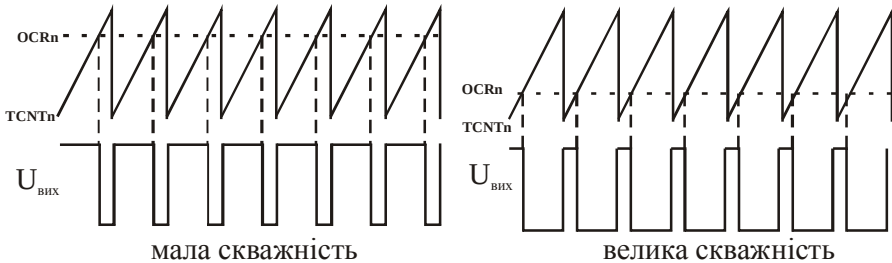


Рисунок 2.1 – Робота таймера в режимі FastPWM

Вміст рахункового регістра безперервно порівнюється з вмістом регістра збігу. Поки число в регістрі OCRn більше за число в рахунковому регістрі таймера (TCNTn), напруга на виході ШІМ дорівнює логічній одиниці. Коли ж в процесі рахунку вміст рахункового регістра TCNTn перевищить вміст OCRn, на виході ШІМ установиться нульовий потенціал.

Отже на виході ми отримаємо прямокутні імпульси. Скважність цих імпульсів залежатиме від вмісту регістра OCRn. Чим менше число в OCRn, тим вище скважність вихідних імпульсів. На рис. 2.1 показана скважність імпульсів для двох різних значень регістра OCRn.

Якщо вміст OCRn досягне свого максимального значення, то імпульси на виході ШІМ зникнуть, і там постійно буде присутня логічна одиниця. При зменшенні числа в OCRn з'являться імпульси малої скважності (тривалість майже дорівнює періоду). Якщо плавно зменшувати число в OCRn, то скважність плавно зменшуватиметься. Коли вміст OCRn досягне нуля, імпульси на виході ШІМ також зникнуть, і там встановиться логічний нуль.

2.1.2.5 Режим «ШІМ з точною фазою» (Phase Correct PWM).

Описаний вище режим ШІМ має один недолік: при зміні тривалості імпульсів міняється і їх фаза. Центр кожного імпульсу начебто зсувається в часі. При управлінні електродвигуном така поведінка фази небажана. Тому в мікроконтролерах AVR передбачений ще один режим

ШИМ - це ШИМ з точною фазою. Принцип роботи таймера в цьому режимі наведено на рис. 2.2.

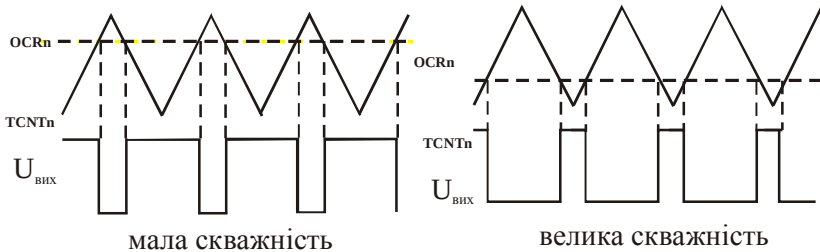


Рисунок 2.2 – Робота таймера в режимі *Phase Correct PWM*

Режими «Phase Correct PWM» і «Fast PWM» різняться режимом роботи лічильника, який спочатку працює як і в попередньому режимі (з кожним вхідним імпульсом його значення збільшується на одиницю). Досягнувши максимального значення лічильник не скидається в нуль, а перемикається в режим реверсивного рахунку.

Тепер уже з кожним вхідним імпульсом його вміст зменшується на одиницю. В результаті пилоподібна крива, що відображує вміст рахункового регістра TCNTn, стає симетричною, як показано на рис.2.2. Система збігу працює так само, як і у попередньому випадку.

Завдяки симетричності сигналу на таймері, фаза вихідних імпульсів в процесі регулювання скважності не змінюється. Середина кожного імпульсу строго прив'язана до точки зміни напрямку рахунку таймера.

Недоліком режиму «Phase Correct PWM» є в два рази менша частота вихідного сигналу, що зменшує динамічність регулювання. Також при використанні зовнішніх фільтрів для перетворення імпульсного сигналу ШИМ в аналоговий, схема з нижчою частотою потребує застосування комплектуючих з великими габаритами і масою.

2.1.2.6 Асинхронний режим. У деяких моделях мікроконтролерів таймер може працювати в асинхронному режимі. У цьому режимі на вхід таймера подається або частота від внутрішнього кварцового генератора, або від зовнішнього генератора. Лічильник не виробляє ніяких переривань і додаткових сигналів. У цьому режимі він працює годинником реального часу. Мікроконтролер може предвстановлюва-

ти вміст рахункового регістра, а потім у будь-який момент він може рахувати цей вміст, набувши, таким чином, поточного значення реального часу.

2.1.3 Переділільники таймерів/лічильників

Кожен таймер МК може працювати від двох різних джерел тактових імпульсів. Або це зовнішні імпульси, або імпульси, що виробляються внутрішньою схемою МК. Яке б джерело сигналів не було обране, перед тим, як потрапити на вхід таймера, цей сигнал проходить схему попереднього дільника. **Попередній дільник** призначений для розширення діапазону сформованих частот і тривалості таймера. Кожна мікросхема AVR має свою структуру попереднього дільника для таймерів/лічильників. Спрощену схему одного з варіантів попереднього дільника наведено на рис. 2.3.

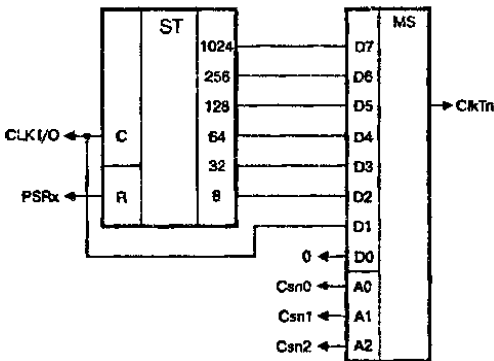


Рисунок 2.3 – Переділітник для таймера

регістра управління таймером (TCCRn).

Отже, записуючи в розряди CSn0, CSn1, CSn2 різні значення, можна обирати один з восьми режимів роботи переділітника. Залежно від обраного режиму, на вихід схеми можуть поступати сигнали з одного з виходів десятирозрядного дільника, прямий сигнал з тактового генератора або нульовий логічний рівень (входу D0). У останньому випадку сигналу на вході таймера немає, і його робота припиняється.

Схема, наведена на рис. 2.3, не є стандартом для всіх мікроконтролерів серії AVR. Вона відображає лише загальний принцип побудови переділільників. У різних моделях це зроблено трохи по-різному.

Частота внутрішнього тактового генератора CLK поступає на спеціальний десятирозрядний дільник. З виходів дільника знімаються сигнали CLK/8, CLK/32, CLK/64, CLK/128, CLK/256 і CLK/1024, які поступають на входи даних мультимплексора. На адресні входи мультимплексора поступають сигнали від трьох розрядів

На рис. 2.4 наведено ще одну схему переддільника. Ця схема, на відміну від попередньої, передбачає подачу на входи таймерів тактового сигналу від зовнішнього джерела. Для цього кількість сигналів, що знімаються з десятирозрядного дільника, зменшена до чотирьох. CLK/32 і CLK/128 виключені. Зате в схемі з'явилися ланцюги, через які на вхід таймера можуть поступати зовнішні імпульси.

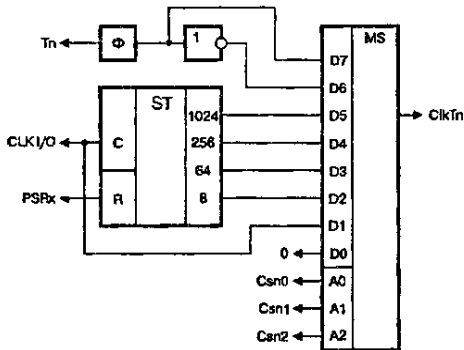


Рисунок 2.4 – Переддільник з входом для зовнішнього сигналу

Ці імпульси повинні подаватися на вхід Тп. З цього входу імпульси поступають на формуувач, який попередньо їх обробляє (наближає їх форму до прямокутної). Потім імпульси поступають на вхід D7 дешифратора. На вхід D6 поступають ті ж імпульси, але тільки в інвертованому вигляді. Отже, для схеми (рис.2.4) маємо

наступні вісім режимів роботи:

- **режим 0** - відсутність імпульсів;
- **режим 1** - прямий сигнал від внутрішнього генератора;
- **режими 2...S** - один з сигналів з дільника;
- **режим 6** - інверсний сигнал із зовнішнього входу;
- **режим 7** - прямий зовнішній сигнал.

2.2 Завдання до роботи

2.2.1 Вивчити документацію до мікроконтролера АТmega8, визначити наявні в цій моделі таймери/лічильники та їх можливості.

2.2.2 Скласти програму на асемблері, яка дозволяє сформувати послідовність прямокутних імпульсів заданої частоти (табл.2.1) на виході будь-якого порту мікроконтролера, наприклад, PD0, при натисканні на кнопку, що підключена до виводу PC0 (рис. 2.5).

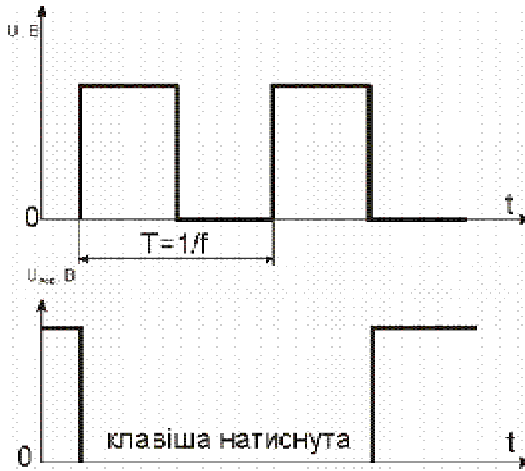


Рисунок 2.5 – Форма сигналу на виході порту

Таблиця 2.1 – Частота генератора прямокутних імпульсів

Варіант	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Частота, КГц	91	60	73	65	42	1	82	77	61	20	11	70	35	18
Варіант	15	16	17	18	19	20	21	22	23	24	25	26	27	
Частота, КГц	64	60	82	21	48	84	24	83	52	56	83	29	19	

Для програмування МК треба орієнтуватися на схему підключення рис.2.6

Щоб визначити стан кнопки - натиснута чи ні - опитують стан відповідного піну порту за допомогою команд роботи з бітами SBIS або SBIC, наприклад:

```
sbis pinC,0
```

```
  rjmp підпрограма_обробки.
```

Для реалізації програми використати переривання по переповненню таймера/лічильника $T0$. Для ініціалізації таймера слід записати у відповідні регістри коди. Наприклад, запис одиниці у молодший біт регістру $TIMSK$ дозволяє переривання по переповненню таймера $T0$

```
ldi tmp, 0b00000001
```

```
OUT TIMSK, tmp
```

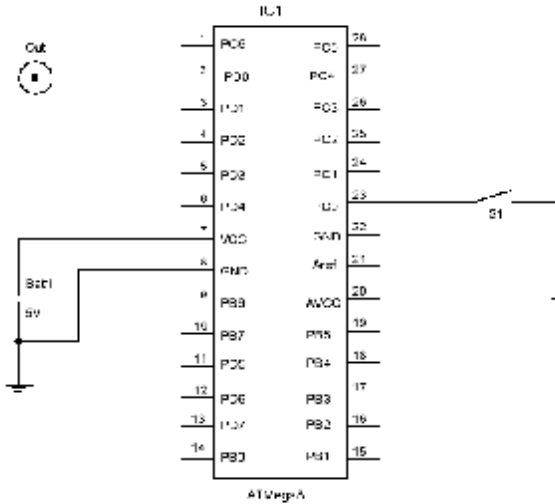


Рисунок 2.6 – Схема підключення мікро контролера

2.2.3 Скласти звіт з роботи, у якому слід розмістити:

- алгоритм роботи програми у вигляді блок схеми;
- схему електричну принципову;
- текст програми на асемблері;
- результати моделювання програми;
- висновки.

2.3 Контрольні запитання

- 1 Характеристика наявних у ATmega8 таймерів.
- 2 Як здійснити конфігурацію таймера?
- 3 Робота таймера у асинхронному режимі.
- 4 Як здійснити конфігурацію переддільника?
- 5 Які переривання доступні у ATmega8 при роботі з таймерами?

ЗЛАБОРАТОРНА РОБОТА №3

„ЗАСОБИ ВВОДУ-ВИВODУ ІНФОРМАЦІЇ В МПС. LED-ІНДИКАТОРИ”

Мета роботи – ознайомитися із загальними принципами організації системи вводу і виводу інформації в мікропроцесорних системах.

3.1 Принципи побудови системи вводу-виводу

3.1.1 Загальна інформація про індикатори у МПС

Відображення інформації є важливою частиною будь-якого інформаційного процесу вимірювання, управління і контролю. Інформація, яку сприймає людина, має подаватись в ефективній формі з мінімальними вимогами до перекодування і перерахунку інформації. Тому цифрове відтворення інформації у багатьох випадках краще аналогового.

У цифровій техніці застосовують індикатори, які базуються на різних принципах дії. *Індикатором* називають вихідний пристрій інформаційного приладу або системи, що забезпечує візуальне відображення інформації.

Індикатори класифікують за принципом формування зображення на знакомодулюючі (ЗМІ) і знаковинтезуючі (ЗСІ).

Знакомодулюючі індикатори містять певний набір знаків і відображають лише один з них, наприклад, газорозрядні індикатори.

Знаковинтезуючі індикатори містять набір сегментів, за допомогою якого проводиться синтез великої кількості різних знаків, наприклад, вакуумні люмінесцентні, напівпровідникові, рідкокристалічні. Серед знаковинтезуючих індикаторів розрізняють сегментні і матричні індикатори.

Сегментні індикатори – це індикатори, елементи відображення яких є сегменти і згруповані в одне (однорозрядний) або декілька (багаторозрядний) знакомісць.

Матричні індикатори (або інакше - графічні) складаються з набору окремих точок. Призначені для відображення будь-якої інформації, включаючи символи і графіку. Їх використовують для побудови високоергономічних відображаючих пристроїв.

У сучасній електронній техніці найбільш поширені такі типи індикаторів: світлодіодні (LED – light emission diode) і рідкокристалічні

індикатори (LCD – liquid crystal display). Поширеність LCD обумовлена в порівнянні з іншими типами індикаторів майже ідеальними електричними характеристиками. Світлодіодні індикатори мають низьку напругу живлення (1,5...3,5 В), що зручно, проте їх споживаний струм досить великий (2 - 20 мА), і це практично "ставить хрест" на їх використанні в сучасній мікропотужній радіоелектронній апаратурі. LCD при робочій напрузі 3-5 В споживають малі струми - доли мікроампера. Керування LCD здійснюється змінною напругою, але для сучасної техніки це - не проблема. На відміну від решти індикаторів вони практично нечутливі до електричних перевантажень. І ще одна особливість: LED-індикатори випромінюють світло, а LCD – поглинають.

Використання того чи іншого індикатора залежить від багатьох факторів: умов роботи, вартості, вимог споживачів готової продукції.

LED-індикатори набули поширення завдяки зручності, наочності і невисокій вартості. Цифрові і знакові LED-індикатори є впорядкованим набором окремих світлодіодів (сегментів індикатора), розміщених в одному корпусі і, зазвичай, сполучених між собою в певну електричну схему (рис. 3.1а).

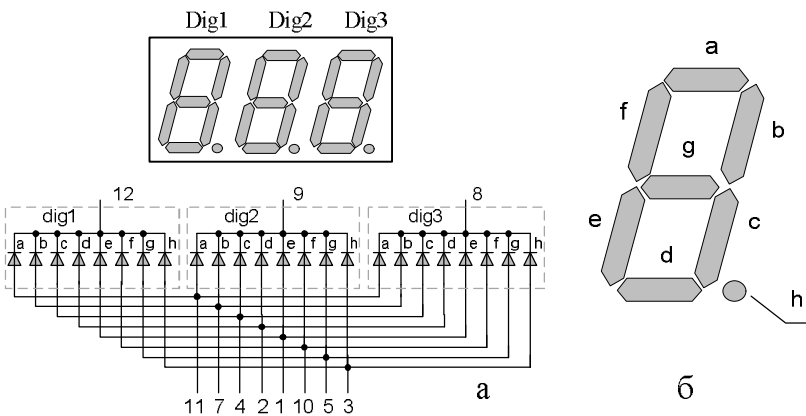


Рисунок 3.1 - Будова семисегментних LED-індикаторів

Цифрові і знакові індикатори бувають однорозрядними і багаторозрядними, їх знаки складаються з семи і більш сегментів. Окрім сегментів індикатори мають «крапку» або «кому». На рис. 3.1,б наведено стандартне позначення сегментів індикатора. Висвітлення обраного сегменту або групи сегментів при отриманні зображення знаку забез-

печуться включенням їх в ланцюг проходження струму.

Залежно від схеми включення світлодіодів в індикаторі розрізняють індикатори із загальним катодом (рис. 3.1,а) і загальним анодом. За способом включення індикаторів в електричну схему можна виділити два режими: статична і динамічна індикація.

LCD-індикатори конструктивно виконуються як сегментні, наприклад (ИЖКЦ1-4/14), або матричні. Останні внаслідок необхідності роботи на змінному струмі, зазвичай виконуються у вигляді готового модуля на друкованій платі із спеціальним контролером (наприклад HD44780 для знакосинтезуючих і S6B0108 для матричних).

3.1.2 Види індикації LED-індикаторів

Спосіб *статичної індикації* полягає в постійному підсвічуванні індикатора від одного джерела інформації (рис. 3.2, 3.3), тобто кожен з цифрових індикаторів блоку індикації постійно підключений через власний перетворювач коду (наприклад, дешифратор) до власного джерела інформації (лічильник або МК).

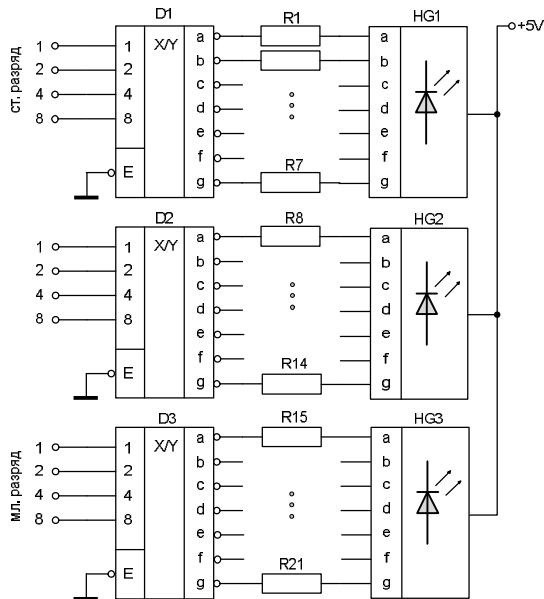


Рисунок 3.2 – Схема пристрою з блоком статичної індикації на індикаторах із загальним анодом

Коли мікроконтролер є джерелом інформації і перетворювачем інформації, індикатори приєднують безпосередньо до нього через струмообмежувальні резистори (рис. 3.3).

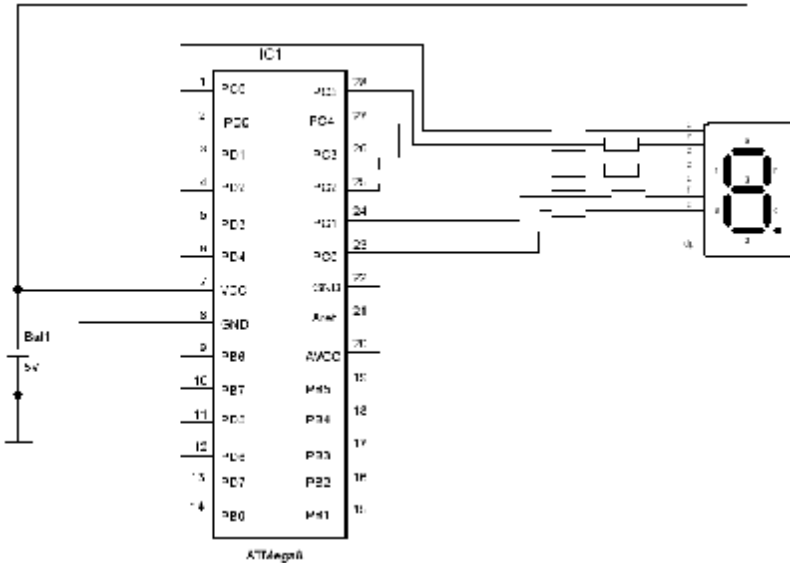


Рисунок 3.3 – Схема МПС із статичною індикацією: один індикатор із загальним анодом

При кількості індикаторів більше 4 технічна реалізація режиму статичної індикації стає недоцільною. Це пов'язано із збільшенням необхідної кількості перетворювачів коду, резисторів і дрітків для сполучення, а також значним збільшенням споживаного струму (переважно, індикаторами).

Крім того, деякі види індикаторів, наприклад матричні світлодіодні або рідкокристалічні індикатори, зображення графічних символів в яких отримують при подачі імпульсної напруги на впорядкованих в стовпці і рядки певні точки матриці, не розраховані на застосування в системах статичної індикації.

Динамічна індикація полягає в наступному. Для зменшення загальної кількості провідників, які зв'язують між собою мікросхеми індикатора, цифрові індикатори частіше почали виготовляти багаторозрядними блоками, біля яких в єдиному корпусі може бути від 2 до 16

цифрових знаків. Для скорочення кількості виводів у багаторозрядному індикаторі всі однойменні сегменти з'єднані разом і мають один загальний вивід (рис. 3.1, а).

Для забезпечення роботи багаторозрядних індикаторів або декількох однорозрядних індикаторів використовують динамічну індикацію, суть якої полягає в тому, що розряди індикатора працюють не одночасно, а по черзі. Якщо перемикання індикаторів здійснюється з великою швидкістю, то людина не помічає мерехтіння, і картинка виглядає як статична. Індикатори підключають з частотою $50\div 70$ Гц, що достатньо для того, щоб не помічати мерехтіння індикаторів. Загальну структурну схему динамічної індикації наведено на рис. 3.4.

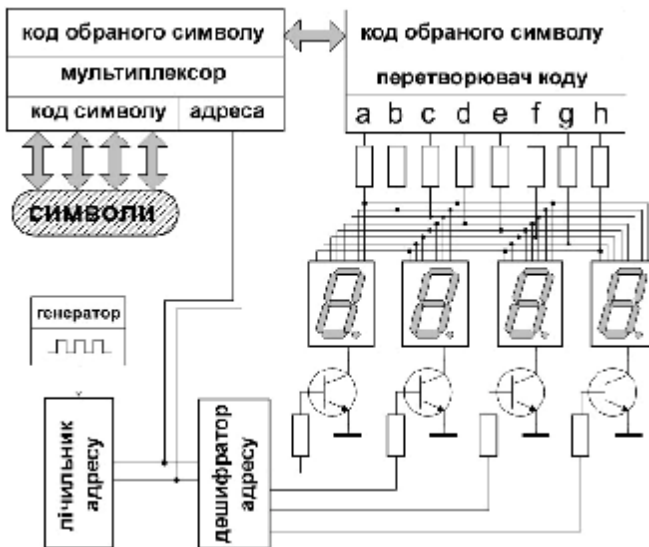


Рисунок 3.4 - Структурна схема блоку динамічної індикації на 4 знаки

Інформація поступає не на декілька перетворювачів, а на один, загальний для всіх, але порціями. Цей перетворювач коду своїми виходами підключений до всіх елементів відразу. У системі динамічної індикації мов би працює швидкодіючий безконтактний перемикач на багато положень. У одному з його положень всі виводи (сегменти) одного з розрядів підключені до відповідних виходів дешифратора. І в

цей же момент через транзисторні ключі поступає сигнал управління засвіченням елементів того знакомісця, яке відповідає цьому розряду.

Коли МПС виконує всі завдання з перетворювання коду і вибору адреси знакомісця, схема підключення значно спрощується (рис.3.5).

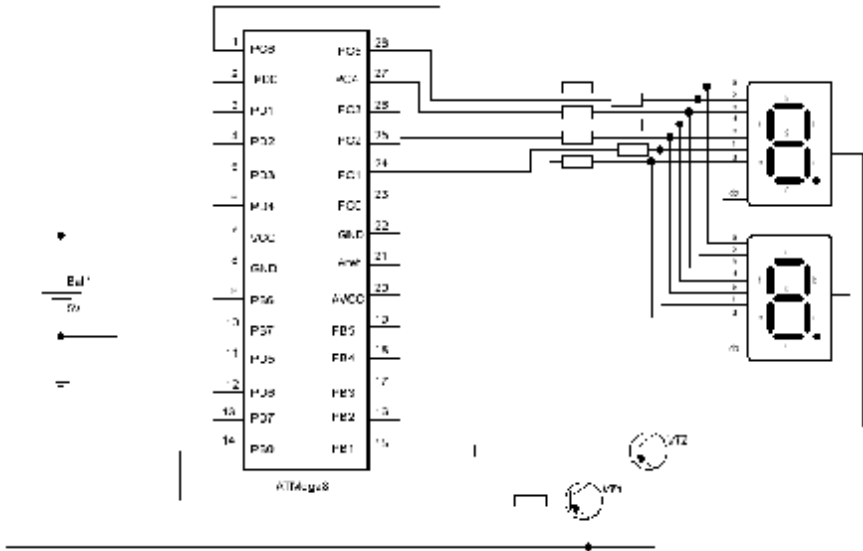


Рисунок 3.5 – Організація динамічної індикації дворозрядного індикатора

Перевага способу динамічної індикації відчутна при кількості розрядів більше 4. Економія перетворювачів коду і дротів сполучення є значною, якщо схема динамічної цифрової індикації віддалена від джерела інформації. Схема з динамічною індикацією споживає менший струм, має менші габарити і вартість. Однак при організації великої кількості розрядів виникає проблема яскравості індикаторів у зв'язку з малим часом світіння кожного розряду індикатора.

3.1.3 Принципи побудови системи вводу інформації у МПС

Найпростішим засобом вводу інформації в мікропроцесорну систему є ручне введення інформації з клавіатури. У найпростішому випадку клавіатура може складатися лише з однієї кнопки (рис. 3.6).

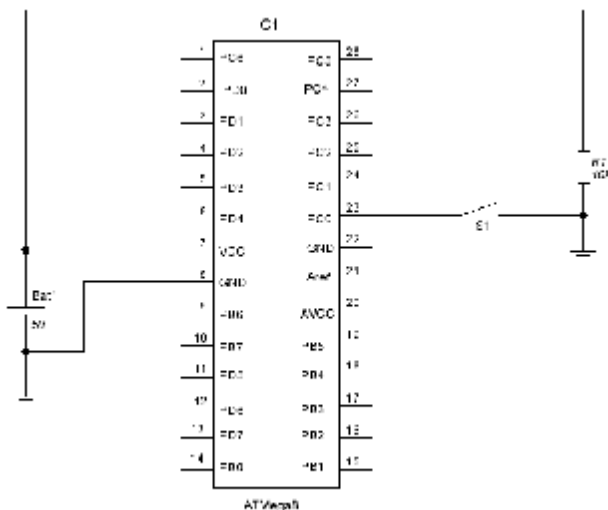


Рисунок 3.6 – Схема підключення кнопки до МПС

Для введення інформації потрібно опитати стан порту, для чого використовуються наступні команди роботи з бітами:

```
SBIC PinC,0      ;пропустити наступний рядок (команду),
                  ;якщо біт 0 порту C очищено
rcall Button1    ;виклик підпрограми Button1
.....
SBIS PinC,0      ;пропустити наступний рядок (команду),
                  ;якщо біт 0 порту C встановлено
rcall Button2    ;виклик підпрограми Button2
```

Порти, до яких буде приєднана клавіша (клавіатура), потрібно відповідно сконфігурувати (як входи).

У випадку застосування більшої кількості кнопок, які утворюють клавіатуру, розробнику може просто не вистачити вільних портів. З урахуванням того, що контролер зазвичай ще виконує функції виводу якоїсь інформації чи керування певною кількістю виконуючих пристроїв, це питання стає проблемою. Рішенням цієї проблеми є складання кнопок у матриці, рис. 3.7.

Роботу мікроконтролера при такому підключенні умовно можна представити так. Порт PB (виводи 0-1-2) налаштовані на вивід, у вихі-

дному стані на виході «1». Виводи порту PD налаштовані на ввід. Через струмообмежуючі резистори на входах портів присутні логічні одиниці. Читання порту PD дає значення \$ff. При роботі МК сканує клавіатуру: по черзі подає на виводи порту PB0, PB1, PB2 логічні нулі і одночасно здійснює читання стану порту PD. У випадку натискування однієї з клавіш результат читання порту PD буде відрізнятися від значення \$ff.

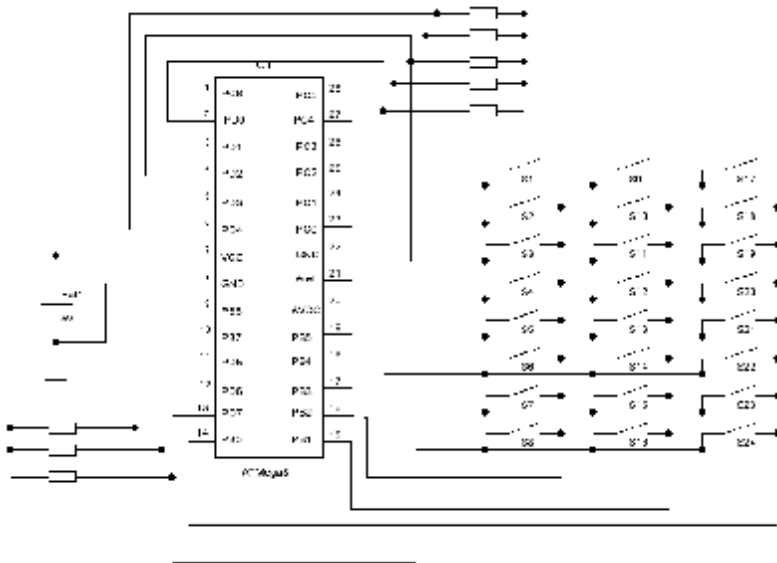


Рисунок 3.7 – Схема підключення матричної клавіатури 3x8 (24 кнопки)

3.2 Завдання до лабораторної роботи

3.2.1 Розробити принципову схему і алгоритм, та написати програму для пристрою на мікроконтролері ATmega8, яка б дозволяла:

- генерувати задану послідовність імпульсів заданої форми (рис. 3.8) і частоти (табл. 3.1);
- змінювати у заданих межах період/скважність при натисненні на клавішу (див. табл. 3.1);
- відображати змінюваний параметр на семисегментному дисплеї.

Таблиця 3.1- Параметри імпульсів для індивідуального завдання

№ варі- анту	t ₁ , мкс	t ₂ , мкс	t ₃ , мкс	T, мс	При натисненні кнопок змі- нювати	
					параметр	величина, %
1	10	5	10	1	t ₁	+ / - 20
2	12	6	12	2	t ₁	+ / - 30
3	14	7	14	3	t ₁	+ / - 40
4	16	8	16	4	t ₁	+ 50
5	18	9	18	5	t ₁	- 50
6	20	10	20	6	t ₂	+ / - 30
7	22	11	22	7	t ₂	+ / - 40
8	24	12	24	8	t ₂	+ / - 50
9	26	13	26	9	t ₂	+ / - 20
10	28	14	28	10	t ₂	- 30
11	30	15	30	1	t ₃	+ 40
12	32	16	32	2	t ₃	+ / - 50
13	34	17	34	3	t ₃	+ / - 20
14	36	18	36	4	t ₃	+ 50
15	38	19	38	5	t ₃	+ / - 40
16	38	20	40	6	t ₃	- 50
17	36	21	10	7	t ₃	+ / - 20
18	34	22	12	8	t ₃	+ / - 30
19	32	23	14	9	t ₃	+ / - 40
20	30	24	16	10	t ₃	+ / - 50
21	28	25	18	1	t ₁	+ / - 20
22	26	26	20	2	t ₂	+ / - 30
23	24	27	22	3	t ₃	+ 40
24	22	28	24	4	t ₁	- 50
25	20	29	26	5	t ₂	+ / - 20
26	18	30	28	6	t ₃	+ / - 30
27	16	31	30	7	t ₁	+ / - 40
28	14	32	32	8	t ₂	+ / - 50
29	12	33	34	9	t ₃	+ 40
30	10	34	36	10	t ₁	- 50

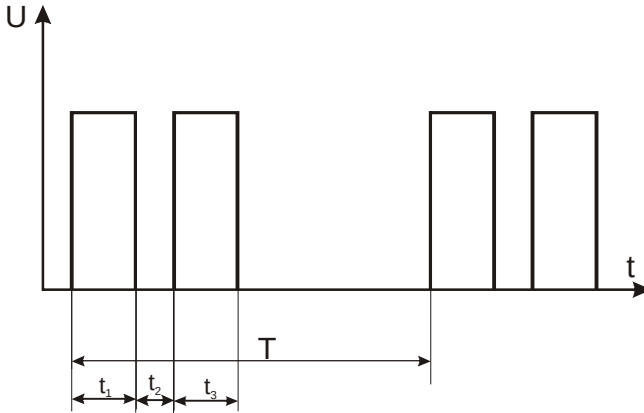


Рисунок 3.8 – Форма імпульсів на виході пристрою

3.2.2 Провести моделювання роботи приладу засобами AStudio та Proteus 7.6.

3.2.3 Оформити звіт за вимогами. Звіт повинен містити:

- титульну сторінку, тему і мету роботи;
- необхідний теоретичний матеріал;
- розрахунки і пояснення;
- електричні принципові схеми включення МПС;
- блок-схему алгоритму роботи програми;
- лістинг програми на асемблері із поясненнями;
- результати моделювання: *print screen* екрану у потрібні моменти часу;
- висновки по роботі;
- відповіді на контрольні запитання.

3.3 Контрольні запитання

- 1 Структура статичної індикації, особливості, переваги і недоліки.
- 2 Структура динамічної індикації, особливості, переваги і недоліки.
- 3 Якою має бути частота оновлення для динамічної індикації?
- 4 Ефект брязкоту контактів, способи його усунення.
- 5 Організація індикації восьмирозрядного індикатора.

4 ЛАБОРАТОРНА РОБОТА №4 РОБОТА З LCD-ІНДИКАТОРАМИ НА БАЗІ КОНТРОЛЕРУ HD44780

Мета роботи – ознайомитися із загальними принципами організації системи виводу інформації в мікропроцесорних системах за допомогою сучасних знакосинтезуючих дисплеїв на базі контролеру HD44780.

4.1 Коротка інформація про LCD-індикатори

Контролер HD44780 виробництва Hitachi є найбільш поширеним контролером управління алфавітно-цифровим модулем. Майже всі провідні виробники - Epson, Sanyo, Toshiba, Samsung, Philips випускають аналоги цього контролера або сумісні з ним по інтерфейсу і командній мові мікросхеми або PKI на базі цих контролерів. Практично HD44780 є промисловим стандартом. Модулі з цим контролером застосовують у вимірювальних приладах, промисловому, технологічному і медичному устаткуванні, офісній техніці.

Контролер підтримує розмір символу 5x7 точок і 5x10 точок. HD44780 може управляти двома рядками по 40 символів.

Підключення до МК здійснюється так. Контролер з'єднується з PKI через паралельну синхронну шину (8 або 4 ліній даних - вибирається програмно), через лінію вибору операцій (R/W), лінію вибору регістра (RS), лінію стробування і синхронізації (E) (рис. 4.1).

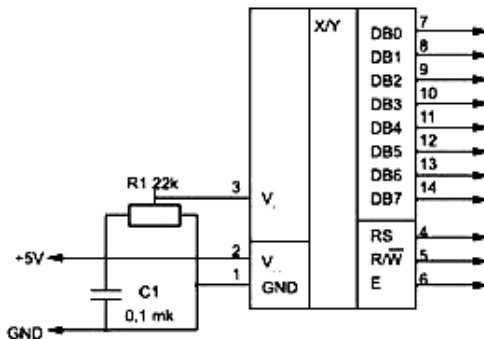
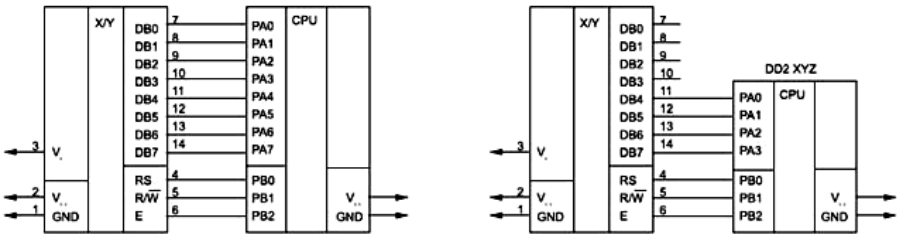


Рисунок 4.1 – Інтерфейс підключення контролера HD44780

Підстроювальний резистор R1 плавно змінює напругу живлення РКІ, що дозволяє виставляти необхідну контрастність індикатора при необхідному куті огляду. Модуль може під'єднуватися за допомогою 4- (рис. 4.2, б) або 8-мирозрядної шини (рис. 4.2, а). При цьому обмін можна організувати або з системною шиною, або через порти вводу-виводу програмними засобами.



а – восьмирозрядна шина;

б – чотирирозрядна шина

Рисунок 4.2 – Схема підключення РКІ до МК

На часовій діаграмі на рис. 4.3 показані стани керуючих сигналів і шини даних під час читання і запису.

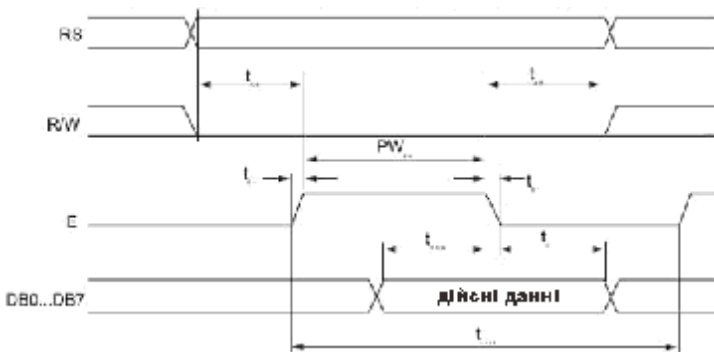


Рисунок 4.3 – Часова діаграма запису даних у РКІ

У вихідному стані $E=0$, $R/W=0$ значення сигналу RS довільне, шина даних DB0...DB7 перебуває в стані високого імпедансу. У проміжках між операціями обміну E і R/W також мають дорівнювати 0, у цей момент шина даних вільна і може використовуватися в мультиплексному режимі для інших цілей. У табл.4.1 наведено послідовність

дій при виконанні операцій читання-запису. Час виконання кожного кроку не менше 250 нс.

Таблиця 4.1 – Послідовність дій при читання та запису

Операції запису для 8-мі розрядної шини	
1	Установка значення лінії RS
2	Виведення значення байта даних на шину DB0...DB7
3	Установка лінії E=1
4	Установка лінії E=0
5	Установка шини DB0...DB7 у стан HI
Операції читання для 8-мі розрядної шини	
1	Установка значення лінії RS
2	Установка лінії R/W=1
3	Установка лінії E=1
4	Зчитування байта даних з шини DB0...DB7
5	Установка лінії E=0
6	Установка лінії R/W=0
Операції запису для 4-х розрядної шини	
1	Установка значення лінії RS
2	Виведення значення старшої тетради байта даних DB4...DB7
3	Установка лінії E=1
4	Установка лінії E=0
5	Виведення значення молодшої тетради байта даних DB4...DB7
6	Установка лінії E=1
7	Установка лінії E =0
8	Установка шини DB4...DB7 в стан HI
Операції читання для 4-х розрядної шини	
1	Установка значення лінії RS
2	Установка лінії R/W=1
3	Установка лінії E=1
4	Зчитування значення старшої тетради байта даних DB4...DB7
5	Установка лінії E=0
6	Установка лінії E=1
7	Зчитування значення молодшої тетради байта даних DB4...DB7
8	Установка лінії E=0
9	Установка лінії R/W=0

Описані операції читання-запису байта є базовими для здійснення обміну даними з РКІ. Процес обміну по 4-х і 8-мі розрядною шиною розрізняється лише реалізацією цих операцій. Ці дві операції можуть бути реалізовані апаратно, коли модуль підключений до системної шини, або програмно, коли він взаємодіє з портами МК.

У таблиці 4.2 наведені значення часових інтервалів сигналів.

Таблиця 4.2 – Часові характеристики

Параметр	Позн.	Мін., нс	Макс., нс
Операція читання			
Період сигналу E	t_{cycE}	500	-
Позитивний напівперіод сигналу E	PW_{EH}	230	-
Фронт/спад сигналу E	t_{Ef}, t_{Er}	-	20
Встановлення адреси	t_{AS}	40	-
Утримання адреси	t_{AH}	10	-
Встановлення даних	t_{DSW}	80	-
Утримання даних	t_{DSH}	10	-
Операція запису			
Період сигналу E	t_{cycE}	500	-
Позитивний напівперіод сигналу E	PW_{EH}	230	-
Фронт/спад сигналу E	t_{Ef}, t_{Er}	-	20
Встановлення адреси	t_{AS}	40	-
Утримання адреси	t_{AH}	10	-
Встановлення даних	t_{DSW}	-	160
Утримання даних	t_{DHW}	5	-

4.1.3 Програмування і управління LCD на базі HD4470

На рис. 4.4 наведено основні елементи контролера HD4470, які безпосередньо взаємодіють з керуючою програмою - регістр даних (DR), регістр команд (IR), відеопам'ять (DDRAM), ОЗП знакогенератора (CGRAM), лічильник адреси пам'яті (AC), прапор зайнятості контролера.

Управляється контролер через керуючий інтерфейс системи. Основні об'єкти взаємодії - регістри DR і IR. Вибір регістра, що адресується, здійснюється лінією RS, якщо RS=0 - адресується регістр команд (IR), якщо RS=1 - регістр даних (DR). Дані через регістр DR мо-

жуть розміщатися або зчитуватися у відеопам'ять (DDRAM), чи в ОЗП знакогенератора (CGRAM) за поточною адресою, на яку вказує лічильник адреси (AC). Інформація в регістрі IR інтерпретується пристроєм виконання команд як керуюча послідовність. Зчитування регістра IR повертає в 7-мі молодших розрядах поточне значення лічильника AC, а в старшому розряді прапор зайнятості (BF). Відеопам'ять має загальний об'єм 80 байтів і призначена для зберігання коду символів.

У таблиці 4.3 показаний набір різних прапорів для HD44780. Прапори визначають режими роботи різних елементів контролера. У таблиці 4.4 наведені значення управляючих прапорів, безпосередньо після подачі на РКІ-модуль напруги живлення. Перевизначення значень прапорів виробляється спеціальними командами, записуваними в регістр IR, при цьому комбінації старших бітів визначають групу прапорів або команду, а молодші містять власне прапори.

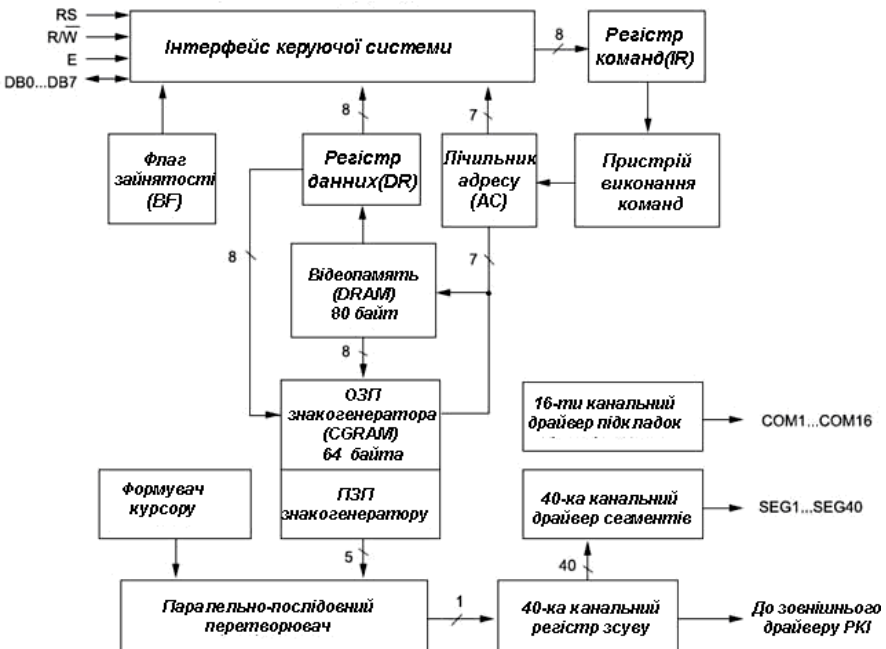


Рисунок 4.4 –Спрощена структурна схема контролера HD4470

Таблиця 4.3 – Управляючі прапори контролера HD44780

I/D	Режим зсуву лічильника адреси АС, 0 - зменшення, 1 - збільшення
S	Прапор режиму зсуву вмісту екрану. 0 – зсуву немає, 1 - після запису в DDRAM чергового коду екран зсувається в напрямі, який визначається прапором I/D: 0 - вправо, 1 - вліво. При зсуві не змінюється вміст DDRAM. змінюються лише внутрішні показники розташування видимого початку рядка в DDRAM
S/C	Прапор-команда, яка виробляє разом з прапором R/L операцію зсуву вмісту екрану (так само, як і у попередньому випадку, без змін в DDRAM) або курсору. Визначає об'єкт зсуву: 0 - зсувається курсор, 1 - зсувається екран
R/L	Прапор-команда, яка виробляє разом з прапором S/C операцію зсуву екрану або курсору. Уточнює напрям зсуву: 0 - вліво, 1 - вправо
D/L	Прапор, що визначає ширину шини даних: 0 - 4 розряди, 1 - 8 розрядів
N	Режим розгортки зображення на ЖКИ: 0 - один рядок, 1 - два рядки
F	Розмір матриці символів: 0 - 5 x 8 точок, 1 - 5 x 10 точок
D	Наявність зображення: 0 - вимкнено, 1 - включено
C	Курсор у вигляді підкреслення: 0 - вимкнений, 1 - включений
B	Курсор у вигляді мерехтливого знакомісця: 0 - вимкнений, 1 - включений

Таблиця 4.4 – Значення прапорів після подачі живлення

I/D=1	Режим збільшення лічильника на 1
S=0	Без зсуву зображення
D/L=1	8-мирозрядна шина даних
N=0	Режим розгортки одного рядка
F=0	Символи з матрицею 5 x 8 точок
D=0	Відображення вимкнене
C = 0	Курсор у вигляді підкреслення вимкнений
B = 0	Курсор у вигляді мерехтливого знакомісця вимкнений

Список комбінацій бітів регістра IR і виконуваних ними управляючі команди наведені в табл.4.5. Оскільки на момент включення РКІ-модуль нічого не відображує (прапор D=0), то для того, щоб вивести будь-який текст, необхідно, як мінімум, включити відображення, встановивши прапор D=1. Наприклад, широко поширена послідов-

ність для ініціалізації модуля: \$OC, 6 (знак \$ перед числом вказує на шістнадцяткове представлення) встановлює режим відображення 2-х рядків з матрицею 5 x 8 точок і робота з 8-мирозрядною шиною даних; \$OC включає відображення на екрані модуля без відображення курсорів; 6 встановлює режим автоматичного переміщення курсору зліва-направо після виведення кожного символу.

Таблиця 4.5 - Комбінації бітів регістра IR

Призначення	D7	D6	D5	D4	D3	D2	D1	D0
Очищення екрану, AC = 0, адресація AC на DDRAM	0	0	0	0	0	0	0	1
AC=0, адресація на DDRAM, скинуті зсуви, початок рядка адресується на початку DDRAM	0	0	0	0	0	0	1	-
Вибирається напрям зсуву курсору чи екрану	0	0	0	0	0	1	I/D	S
Вибирається режим відображення	0	0	0	0	1	D	C	B
Команда зсуву курсору/екрану	0	0	0	1	S/C	R/L	-	-
Визначення параметрів розгортки і ширини шини даних	0	0	1	DL	N	F	-	-
Присвоєння лічильнику AC адреси в області CGRAM	0	1	AG	AG	AG	AG	AG	AG
Присвоєння лічильнику AC адреси в області DDRAM	1	AD	AD	AD	AD	AD	AD	AD

Контролер HD44780 підтримує як операції запису, так і операції читання. Читання регістра DR призводить до завантаження вмісту DDRAM або CGRAM залежно від поточного режиму, при цьому курсор зміщується на одну позицію, як і при записі. Читання регістра IR повертає 8 значущих розрядів, причому в 7-ми молодших міститься поточне значення лічильника AC (7 розрядів, якщо адресується DDRAM, і 6 - якщо CGRAM), а в старшому - прапор зайнятості BF. Цей прапор має значення 1 коли контролер зайнятий і 0 - коли вільний. Необхідно враховувати, що більшість операцій, які виконуються контролером, займають значний час, близько 40 мкс, а час виконання деяких доходить до одиниць мілісекунд, тому цикл чекання зняття прапора BF має бути обов'язково присутнім в програмах драйвера РКІ-модуля і передувати здійсненню будь-якої операції (природно, окрім операції перевірки прапора BF).

Після здійснення операції запису або читання DDRAM і появи після неї ознаки готовності (BF=0), прочитане в цьому ж циклі (разом з прапором BF) значення АС швидше за все не буде достовірним, оскільки між появою ознаки готовності і обчисленням контролером нового значення АСА існує деякий часовий інтервал близько 4 мкс при тактовій частоті контролера 270 кГц. Тому якщо необхідно набути дійсного значення АСА, потрібно зробити повторну операцію прочитання IR після не менше ніж 4 мкс (якщо контролер працює на частоті 270 кГц час чекання необхідно пропорційно збільшити).

Виведення на екран символу здійснюється шляхом запису його коду (табл. 4.6) в регістр DR. Наприклад, код символу Q за таблицею є \$30. При цьому символ розміщується в DDRAM за поточною адресою, що вказується в АС, а значення АС збільшується чи зменшується на 1.

Таблиця 4.6 – Коды символів для відображення на екрані

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	1	2	3	4	5	6	7	8	9	A	B	C
1			!	1	A	Q	a	q						Г	Я	ш
2			"	2	B	R	b	r						Ё	б	ъ
3			#	3	C	S	c	s						Ж	в	ы
4			\$	4	D	T	d	t						Э	г	ь
5			%	5	E	U	e	u						И	ё	э
6			&	6	F	V	f	v						Й	ж	ю
7			'	7	G	W	g	w						Л	з	я
8			(	8	H	X	h	x						П	и	«
9			)	9	I	Y	i	y						У	й	»
A			*	:	J	Z	j	z						Ф	к	„
B			+	;	K	[	k	]						Ч	л	”
C			,	<	L	φ	l	φ						Ш	м	№
D			-	=	M	]	m	]						Ъ	н	с
E			.	>	N	^	n	^						Ы	п	ф
F			/	?	O	_	o	_						Э	т	ё

Виробник контролера рекомендує виконувати наступну послідовність дій для ініціалізації. Витримати паузу не менше 15 мс між встановленням робочої напруги живлення (більше 4,5 В) і виконанням будь-яких операцій з контролером. Першою операцією виконати команду, яка вибирає розрядність шини (це має бути команда незалежно від того, якої розрядності інтерфейс ви збираєтеся використовувати надалі), причому перед виконанням цієї операції не перевіряти значення прапора BF. Далі знову витримати паузу не менше 4,1 мс і повторити команду вибору розрядності шини, причому перед поданням команди знову не перевіряти прапор BF. Наступний крок - знов витримати паузу, цього разу 100 мкс, і втретє повторити команду встановлення розрядності шини, знов без перевірки BF. Ці три операції призначені для ініціалізації і покликані вивести контролер у вихідний режим роботи (тобто перевести в режим роботи з 8-мирозрядною шиною) з будь-якого стану. Слідом за ними нормальним порядком (без витримки пауз, але з перевіркою прапора BF) виконується ініціалізація режимів роботи з видачею послідовності для ініціалізації, аналогічний вказаній раніше (що містить у тому числі команду вибору необхідної розрядності шини).

Слід пам'ятати, що при виборі режиму роботи з 4-х розрядною шиною, тобто видачі команд, це звичайно відбувається з 8-ми розрядного режиму, який встановлюється автоматично після подачі напруги живлення, а отже ви не зможете адекватно оголосити необхідне значення прапорів N і F, розташованих в молодшій тетраді команди установки розрядності шини. Тому команду необхідно повторити у вже сталому 4-х розрядному режимі послідовною передачею двох тетрад.

4.2 Завдання до лабораторної роботи

4.2.1 Ознайомитися з технічною документацією на запропоновану викладачем модель LCD – дисплею.

4.2.2 Нарисувати блок-схему алгоритму ініціалізації LCD - дисплею по 8-бітному інтерфейсу.

4.2.3 Нарисувати блок-схему алгоритму ініціалізації LCD - дисплею по 4-бітному інтерфейсу.

4.2.4 Нарисувати блок схему і написати програму для виводу на LCD-дисплей (HD44780) свого прізвища при підключенні його до МК по 4-бітному інтерфейсу.

4.2.5 Здійснити симуляцію роботи МПС. За наявності помилок знайти і виправити помилку алгоритму чи самої програми.

4.2.6 Оформити звіт, занести до нього результати моделювання. Зробити висновки.

4.3 Контрольні запитання

- 1 Принцип побудови РКІ-модуля з контролером HD44780.
- 2 Алгоритм роботи по 4-х провідному інтерфейсу.
- 3 Алгоритм роботи по 8-ми провідному інтерфейсу.
- 4 Переваги і недоліки модулів з контролером HD44780.
- 5 Схема підключення РКІ-модуля до МК.

5 ЛАБОРАТОРНА РОБОТА №5 „ПРОГРАМУВАННЯ ВНУТРІШНЬОГО АЦП АТМЕГА”

Мета роботи – вивчити принцип дії і функціонування внутрішнього АЦП АТМега; отримати навички його програмування.

5.1 Принцип дії

АТmega8 містить 10-розр. АЦП послідовного наближення. АЦП пов'язаний з 8-канальним аналоговим мультиплексором, вісім однополярних входів якого з'єднані з лініями порту С. АЦП перетворює вхідну аналогову напругу в 10-розр. код методом послідовних наближень. Мінімальне значення відповідає рівню GND, а максимальне рівню AREF рівне 2,56 В. Канал аналогового введення вибирається шляхом запису бітів MUX в регістрі ADMUX. У якості входу АЦП може бути обрано один з входів ADC0..ADC7, а також GND і вихід фіксованого джерела опорної напруги 1,22 В. Робота АЦП дозволяється шляхом установки біта ADEN в ADCSRA. Вибір опорного джерела і каналу перетворення не можливо виконати до установки ADEN. АЦП генерує 10-розрядний результат, який поміщається в пару регістрів даних АЦП ADCH і ADCL. За умовчанням результат перетворення розміщується в молодших 10-ти

розрядах 16-разр. слова (вирівнювання справа), але може бути опційно розміщений в старших 10-ти розрядах (вирівнювання зліва) шляхом установки біта ADLAR в регістрі ADMUX. Практична корисність представлення результату з вирівнюванням зліва існує, коли досить 8-розрядів, оскільки в цьому випадку необхідно зчитувати лише регістр ADCH. У іншому ж випадку необхідно першим зчитувати вміст регістра ADCL, а потім ADCH, чим гарантується, що обидва байти є результатом одного і того ж перетворення. Як тільки виконано читання ADCL блокується доступ до регістрів даних з боку АЦП. Це означає, що якщо зчитується регістр ADCL, а перетворення завершується перед читанням регістра ADCH, то жоден з регістрів не може модифікуватися і результат перетворення втрачається. Після читання ADCH доступ до регістрів ADCH і ADCL з боку АЦП знову дозволяється. Одиночне перетворення запускається шляхом запису «1» в біт запуску перетворення АЦП ADSC. Даний біт залишається у високому стані в процесі перетворення і скидається після закінчення перетворення. Якщо в процесі перетворення перемикається канал аналогового введення, то АЦП автоматично завершить поточне перетворення перш, ніж перемкне канал. У режимі автоматичного перезавпуску АЦП безперервно оцифровує аналоговий сигнал і оновлює регістр даних АЦП. Даний режим задається записом «1» у біт ADFR регістра ADCSRA. Перше перетворення ініціюється шляхом запису «1» в біт ADSC регістра ADCSRA. У даному режимі АЦП виконує послідовні перетворення, незалежно від того скидається прапор переривання АЦП ADIF.

5.2 Регістри управління АЦП

Управління роботою АЦП здійснюється за допомогою регістра ADCSRA (табл.5.1)

Таблиця 5.1 - Регістр управління і статусу АЦП – **ADCSRA**

Розряд	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Читання / запис	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.
Вихідне значення	0	0	0	0	0	0	0	0

Розряд 7 - **ADEN**: Дозвіл роботи АЦП. Запис в даний біт «1» дозволяє роботу АЦП. Якщо в даний біт записати «0», то АЦП відключається, навіть якщо він знаходився в процесі перетворення.

Розряд 6 - **ADSC**: Запуск перетворення АЦП. У режимі одиночного перетворення установка даного біта ініціює старт кожного перетворення. У режимі автоматичного перезапуску установкою цього біта ініціюється лише перше перетворення, а всі інші виконуються автоматично. Перше перетворення після дозволу роботи АЦП, ініційоване бітом ADSC, виконується по розширеному алгоритму і триває 25 тактів синхронізації АЦП, замість звичайних 13 тактів. Це пов'язано з необхідністю ініціалізації АЦП.

Розряд 5 - **ADFR**: Вибір режиму автоматичного перезапуску АЦП. Якщо в даний біт записати «1», то АЦП перейде в режим автоматичного перезапуску. У цьому режимі АЦП автоматично виконує перетворення і модифікує регістри результату перетворення через фіксовані проміжки часу. Запис «0» в цей біт припиняє роботу в даному режимі.

Розряд 4 - **ADIF**: прапор переривання АЦП. Даний прапор встановлюється після завершення перетворення АЦП і оновлення регістрів даних. Якщо встановлені біти ADIE і I (регістр SREG), то відбувається переривання після закінчення перетворення. Прапор ADIF скидається апаратно при переході на відповідний вектор переривання. Альтернативно прапор ADIF скидається шляхом запису в нього «1». Треба звернути увагу, що при виконанні команди "читання-модифікації-запису" з регістром ADCSRA очікуване переривання може бути відключене. Це також поширюється на використання інструкцій SBI і CBI.

Розряд 3 - **ADIE**: Дозвіл переривання АЦП. Після запису «1» в цей біт, за умови, що встановлений біт I в регістрі SREG, дозволяється переривання після закінчення перетворення АЦП.

Розряди 2:0 - **ADPS2:0**: Біти управління переддільником АЦП. Дані біти визначають на яке значення буде поділена тактова частота CPU перед подачею на вхід синхронізації АЦП. Значення коефіцієнтів ділення вказані в таблиці 5.2. Якщо потрібна максимальна роздільна здатність (10 розрядів), то частота синхронізації має бути в діапазоні 50 кГц -200 кГц.

Склад регістру управління мультіплексором АЦП – **ADMUX** наведено у таблиці 5.3.

Розряд 7:6 - **REFS1:0**. Біти вибору джерела опорної напруги - визначають, яка напруга використовуватиметься як опорна для АЦП.

Таблиця 5.2 – Управління переддільником АЦП

ADPS2	ADPS1	ADPS0	Коефіцієнт ділення
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Таблиця 5.3 - Регістр управління мультіплексором АЦП-ADMUX

Розряд	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0
Читання /запис	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит.	Чит./Зп.	Чит./Зп.	Чит./Зп.	Чит./Зп.
Вихідне значення	0	0	0	0	0	0	0	0

Розряд 5 – **ADLAR** - біт управління представленням результату перетворення. Біт ADLAR впливає на представлення результату перетворення в парі регістрів результату перетворення АЦП.

Таблиця 5.4 – Вибір опорного джерела АЦП

REFS1	REFS0	Опорне джерело
0	0	AREF, внутрішній ІОН відключений
0	1	AVCC із зовнішнім конденсатором на виведенні AREF
1	0	Зарезервовано
1	1	Внутрішнє джерело опорної напруги 2.56В із зовнішнім конденсатором на виводі AREF

Якщо ADLAR=1, то результат перетворення матиме лівобічний формат, інакше - правосторонній. Дія біта ADLAR набирає чинності відразу після зміни, незалежно від того, що виконується паралельно перетворення.

Розряд 3:0 - **MUX3:0** - біти вибору аналогового каналу. Дані біти визначають, які з наявних аналогових входів підключаються до АЦП. Якщо значення бітів змінити в процесі перетворення, то механізм їх дії набере чинності лише після завершення поточного перетворення (після установки біта ADIF в регістрі ADCSRA).

Таблиця 5.5 – Вибір вхідного каналу

MUX3..0	Однополярний вхід
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1110	1.23B
1111	0B(GND)

Таблиця 5.6 - Регістри результату перетворення - ADCL і ADCH при ADLAR=0

Розряд	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Читання /запис	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
Вих. значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Таблиця 5.7 - Регістри результату перетворення - ADCL і ADCH при ADLAR = 1

Розряд	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	
Читання	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
/запис	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
Зих. зна-	0	0	0	0	0	0	0	0	
чення									

Після закінчення перетворення результат розміщується в цих двох регістрах. Лівобічний формат ADLAR=1 представлення результату зручно використовувати, якщо достатньо 8 розрядів. У цьому випадку 8-розрядний результат зберігається в регістрі ADCH і, отже, читання регістра ADCL можна не виконувати. При правосторонньому форматі *необхідно спочатку рахувати ADCL, а потім ADCH.*

5.2 Завдання до лабораторної роботи

5.2.1 Розробити схему електричну принципову для зчитування аналогового сигналу величиною до 1 В з каналів 0 і 1. Передбачити вивід цифрового коду на 8 світлодіодів, підключених до порту МК. АЦП повинен працювати з частотою (№варіанту*10) Гц. Розробити блок-схему алгоритму роботи, написати програму на асемблері.

5.2.2 Розробити схему електричну принципову для зчитування аналогового сигналу величиною до 4 В з каналів 2 і 3. Передбачити вивід цифрового коду на LCD-індикатор. АЦП повинен працювати з частотою 200 Гц. Розробити блок-схему алгоритму роботи, написати програму на асемблері.

5.2.3 Розробити схему електричну принципову для зчитування аналогового сигналу величиною до 5 В з 0 каналу. Передбачити вивід цифрового коду та числового значення на LCD-індикатор. Передбачити запуск АЦП користувачем (натискання на кнопку), зчитування 10

значень та їх усереднення. Розробити блок-схему алгоритму роботи, написати програму на асемблері.

5.2.4 Напишіть програму, яка, отримавши по USART0 задане число, запускає АЦП в режимі одиночного перетворення з частотою 1000 Гц, використовує канал 0 і після перетворення посилає результат.

5.3 Порядок виконання роботи

5.3.1 Написати програму індивідуального завдання на ПК.

5.3.2 За допомогою програми для моделювання проаналізувати виконання індивідуальної програми.

5.3.3 Завантажити програму в стенд. Переконалися в правильному виконанні індивідуального завдання, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд.

5.3.4 Зберегти лістинг правильно працюючої програми.

5.3.5 Показати результат виконання програми викладачеві і відповісти на контрольні запитання

5.4 Контрольні запитання

- 1 Принцип роботи АЦП, його основні характеристики.
- 2 Технічні характеристики вбудованого АЦП ATmega8.
- 3 В яких режимах може працювати АЦП ATmega8?
- 4 Яка точність оцифрування сигналу за допомогою АЦП ATmega8? Які засоби передбачені для підвищення точності роботи АЦП?
- 5 Яка максимальна амплітуда аналогового сигналу може подаватися на вхід АЦП, від чого це залежить?

6 ЛАБОРАТОРНА РОБОТА №6

„РОБОТА З ПЕРИФЕРІЙНИМИ ПРИСТРОЯМИ ПО 1-WIRE – ШИНІ”

Мета роботи – ознайомитися із принципами організації мереж MicroLAN; отримати навички програмування пристроїв за протоколом шини 1-Wire.

6.1 Організація роботи 1-Wire шини

6.1.1 Загальні відомості

Шина 1-Wire є основою мереж MicroLAN і розроблена наприкінці 90-х років фірмою Dallas Semiconductor. В даний час фірма Dallas Semiconductor є дочірнім підприємством фірми MAXIM.

Шина 1-Wire і засновані на ній мережі MicroLAN давно з успіхом застосовуються в електронній техніці. Нижче наведені основні характеристики цієї шини:

- максимальна протяжність шини до 300 м;
- швидкість передачі інформації 16,3 Кбіт/с;
- максимальна кількість елементів, що адресуються, на шині 256;
- рівні напруги на шині відповідають стандартним рівням КМОП/ТТЛ;
- напруга живлення компонентів мережі 2,8...6 В;
- для з'єднання елементів мережі може застосовуватися звичайний телефонний кабель або вита пара.

Існують і модифікації шини 1-Wire. Наприклад, окремі види мікросхем підтримують швидкісний режим роботи шини (Overdrive, 142 Кбіт/с). Проте такі мікросхеми можуть працювати лише на шині малої протяжності і за умови, коли рівень зовнішніх електричних перешкод зведений до мінімуму.

Шина MicroLAN побудована за технологією Master/Slave. На шині має бути хоча б один провідний пристрій (Master). Всі інші пристрої мають бути веденими (Slave) Провідний пристрій ініціює всі процеси передачі інформації в межах шини. Master може прочитати дані з будь-якого Slave-пристрою або записати їх туди. Передача інформації від одного Slave до іншого безпосередньо неможлива. Для

того, щоб Master міг звертатися до будь-якого з ведених пристроїв по шині, кожен ведений пристрій містить в собі індивідуальний код (ID-код). Цей код заноситься в спеціальну область мікросхеми за допомогою лазера. Фірма-виробник гарантує, що цей код ніколи не повториться і всі будь-коли виготовлені мікросхеми, що містять 1-Wire інтерфейс, завжди матимуть різні коди.

Протокол 1-Wire включає спеціальну команду пошуку, за допомогою якої провідний пристрій (Master) може здійснювати автоматичний пошук ведених пристроїв. Середня швидкість пошуку елементів в мережі MICROLAN складає приблизно 75 вузлів в секунду.

Приклад топології мережі MicroLAN наведено на рис. 6.1, однак використання топології лише з однією 1-Wire шиною виявляється недостатньо. Обмеження можуть бути викликані значною ємністю або значним сумарним струмом споживання всіх підключених до шини пристроїв. Спеціально для таких випадків в стандарті мереж MicroLAN передбачена можливість розгалуження. Для галуження шини 1-Wire застосовують спеціальні мікросхеми - керовані розгалужувачі (напр. DS2409).

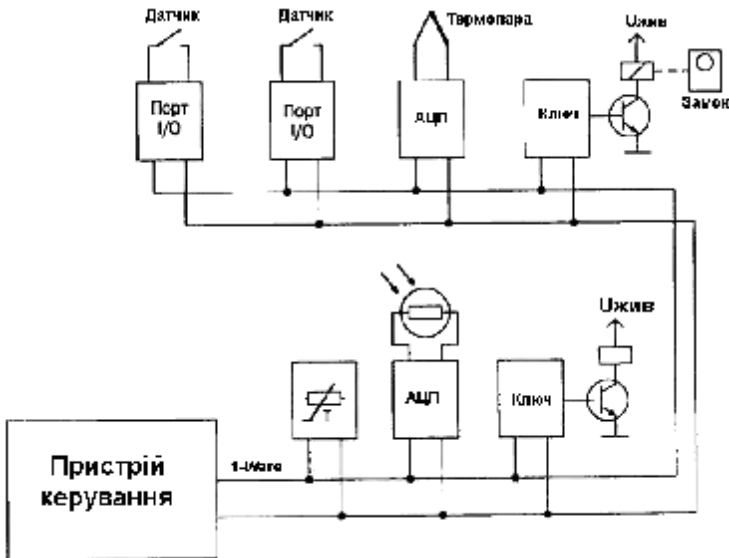


Рисунок 6.1 – Топологія мережі MicroLAN

6.1.2 Схемна реалізація 1-Wire інтерфейсу

Особливості схемної реалізації вихідних каскадів ведучого і веденого пристроїв, які реалізують 1-Wire інтерфейс, наведені на рис. 6.2.

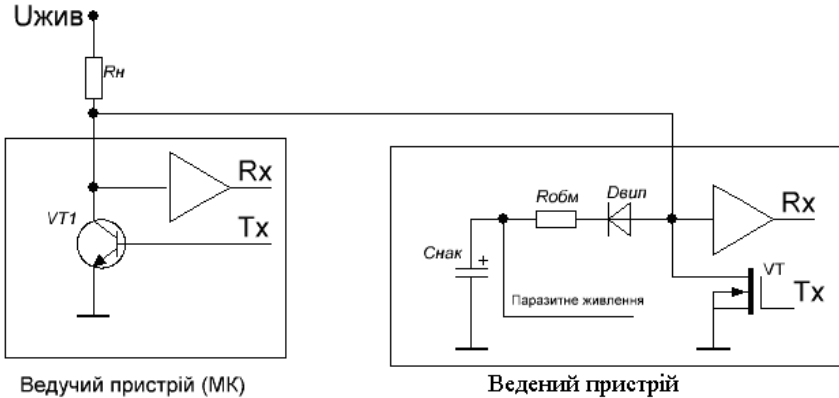


Рисунок 6.2 – Електрична схема мережі MicroLAN

У схемі 1-Wire інтерфейсу використовуються вихідні каскади з відкритим колектором (стоком) і загальним навантаженням R_H для всіх елементів мережі, причому R_H має перебувати в безпосередній близькості від провідного пристрою. У режимі чекання всі вихідні транзистори закриті. На шині присутня напруга логічної одиниці. Інформація по шині передається за допомогою негативних імпульсів. Будь-який пристрій, що підключений до шини, як ведучий так і ведений, може створювати негативні імпульси і тим самим передавати інформацію. Проте ведений пристрій починає процес передачі лише під управлінням ведучого (Master).

У складі веденого пристрою (рис. 6.2) є:

- вбудоване джерело струму (позначене двома пересіченими колами), яке створює внутрішній витік на вході 1-Wire інтерфейсу. Сенс цього витіку - створити нульовий рівень сигналу на внутрішніх елементах веденого пристрою при його відключенні від шини 1-Wire, аби відразу після повторного підключення до мережі ведена мікросхема видала сигнал присутності;

- ланцюг паразитного живлення ($C_{НАК}$, $R_{ОБМ}$, $D_{ВИП}$).

6.1.3 Синхронізація і побітна передача інформації

Протокол 1-Wire має декілька різних рівнів. Найнижчий рівень описує як передаються окремі біти. Всі операції на шині здійснюються тільки під управлінням Master-пристрою, який може виконувати операції двох видів: записувати інформацію в Slave пристрій або зчитувати інформацію з нього. Інформація передається побайтно в послідовному вигляді, біт за бітом, починаючи з молодшого біта. У будь-якому з цих двох випадків для передачі інформації Master-пристрій виробляє на шині тактові імпульси, для чого він періодично «підсаджує» шину за допомогою вихідного транзистора. Корисна інформація передається завдяки зміні тривалості цих імпульсів. Причому при записі інформації тривалістю імпульсів управляє виключно Master-пристрій.

У режимі читання Master пристрій починає формування імпульсу, але Slave пристрій може подовжувати тривалість будь-якого імпульсу, «підсаджуючи» у свою чергу сигнал на лінії в потрібний момент.

Діаграма на рис. 6.4 ілюструє режим запису двох різних бітів інформації, діаграма на рис. 6.4 - режим читання.

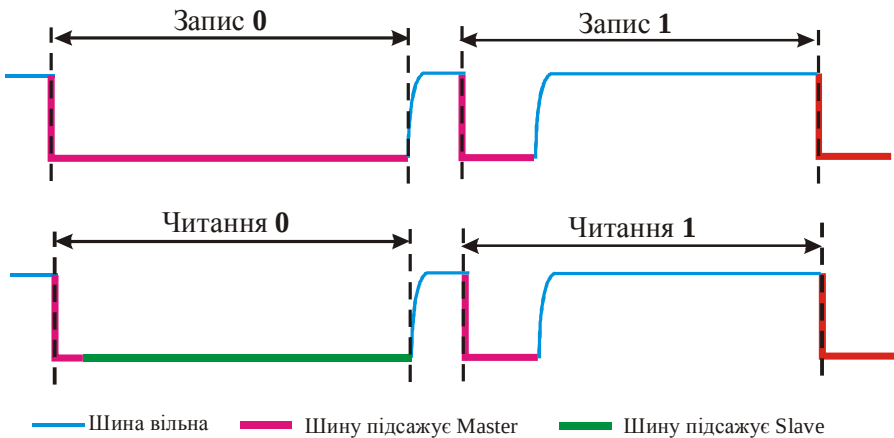


Рисунок 6.4 – Спрощена діаграма процесу запису і читання одного біта

Запис біта здійснюється так. У вихідному стані всі Slave пристрої, підключені до шини, перебувають у режимі чекання. Лінія «відпущена», напруга на шині визначається резистором навантаження R_H .

Для запису даних Master починає формувати негативні синхронізовані імпульси (рис. 6.4). На кожен передаваний біт формується один ім-

пульс. Імпульси передаються «підсаджуванням» лінії до нуля. Для передачі кожного біта виділяється проміжок часу стандартної тривалості, який отримав назву «*слот*» (Slot). Якщо значення передаваного біта дорівнює 0, то Master виробляє «довгий» імпульс, що дорівнює тривалості слота. Для передачі одиничного біта Master виробляє «короткий» імпульс, який, по суті, є чистим синхроімпульсом. Частина слота, що залишилася, має бути заповнена одиничним сигналом.

Виявивши початок синхроімпульса, Slave-пристрій починає процес прийому біта інформації. Передній фронт цього імпульсу служить Slave-пристрою початком відліку. Витримавши паузу, рівну тривалості синхроімпульса, Slave-пристрій читає рівень сигналу на лінії.

Протокол шини I-Wire жорстко визначає лише тривалість слоту. Інтервал між слотами має обмеження лише на мінімальне своє значення. Максимальне значення інтервалу між слотами необмежене. Так може легко регулюватися швидкість передачі даних від максимального значення (16.3 Кбіт/с) практично до нуля.

При **читанні біта** (рис. 6.4) Master виробляє лише синхроімпульси. Виявивши синхроімпульс, Slave-пристрій подовжує або не подовжує цей синхроімпульс у межах слоту. Якщо черговий прочитуваний біт дорівнює нулю, то синхроімпульс подовжується, якщо одиниці - подовження немає. Master-пристрій прочитує цю інформацію і контролює рівень сигналу усередині слоту після синхроімпульсу.

Для надійної роботи однопровідного інтерфейсу необхідно щоб у процесі передачі інформації всіма елементами мережі чітко дотримувалися **часових параметрів**. Кожна мікросхема, підключена до мережі, самостійно виробляє всі необхідні для її роботи інтервали часу.

На рис. 6.5 наведено часові параметри протоколу I-Wire в різних режимах роботи.

Величина слоту для передачі одного біта інформації (Tx) має перебувати в межах від 60 до 120 мкс. Тривалість синхроімпульсу – 1 мкс. Ведений пристрій, виявивши на шині передній фронт синхроімпульсу, має сформувати затримку мінімум у 15 мкс, і потім здійснити перевірку сигналу на шині. Допустимий розкид часу затримки для різних екземплярів мікросхем лежить в межах від 15 до 60 мкс. Цей діапазон показаний на рис. 6.5 у вигляді області, позначеної як «Зона перевірки рівня Slave». Мінімальна величина інтервалу між слотами (T_{REC}) рівна 1 мкс, максимальна - необмежена. У режимі читання біта Master виробляє лише синхроімпульси тривалістю 1 мкс. Якщо біт до-

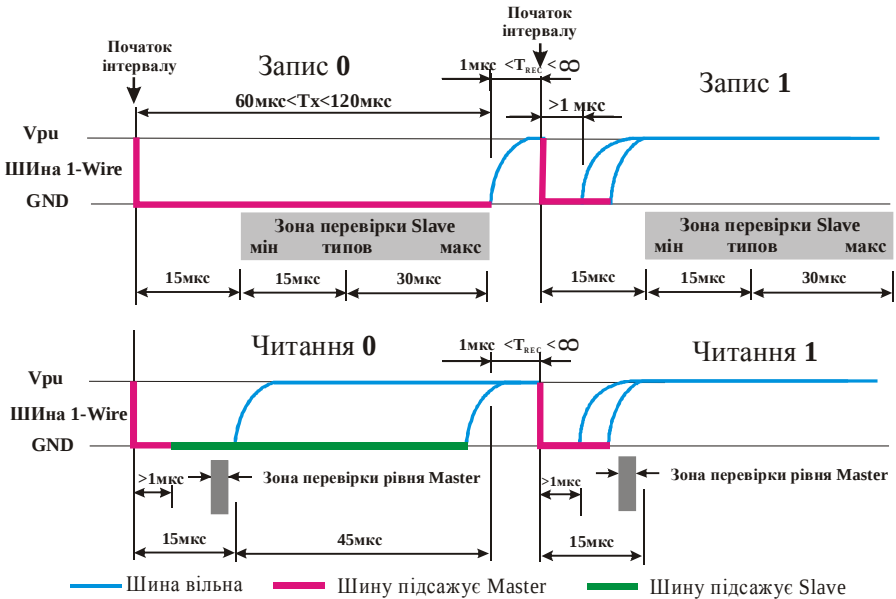


Рисунок 6.5 – Часові характеристики протоколу 1-Wire

рівнює нулю, Slave-пристрій подовжує тривалість синхроімпульсу. Мінімальна тривалість продовженого імпульсу складає 15 мкс. Для цього тимчасового інтервалу теж допускається досить значний розкид (ще на 45 мкс). Якщо біт дорівнює одиниці, подовження синхроімпульсу не відбувається. Аби правильно оцінити значення читаного байта, Master-пристрій повинен прочитати рівень сигналу на шині відразу після закінчення синхроімпульсу, але не пізніше, ніж через 15 мкс. Зона перевірки рівня для Master-пристрою в режимі читання значно вужче аналогічної зони для Slave-пристрою в режимі запису.

6.1.4 Скидання і виявлення присутності на лінії

Байти формуються шляхом підрахунку бітів. Байти передаються молодшим бітом вперед. Перші вісім бітів - це перший байт, наступні вісім бітів - другий байт, і так далі. Початок же цього ланцюжка визначається сигналом скидання. Будь-який цикл обміну даними в мережі MicroLAN починається з імпульсу скидання - наддовгого негативного імпульсу на шині 1-Wire, який виробляється провідним пристроєм (рис. 6.6).

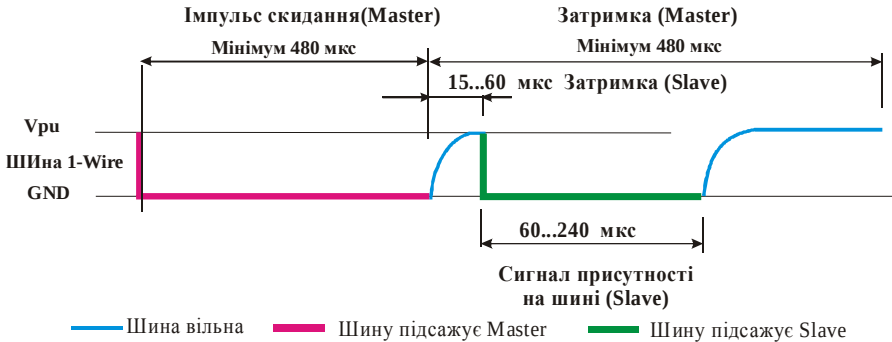


Рисунок 6.6 – Часова діаграма процесу початкового скидання

З імпульсом скидання тісно пов'язаний ще один службовий сигнал - сигнал присутності на шині. Сигнал присутності виробляє кожний Slave-пристрій відразу після закінчення дії імпульсу скидання. Master-пристрій повинен проконтролювати наявність цього імпульсу. Якщо імпульсу присутності немає, то це означає, що на лінії немає жодного Slave-пристрою. Окрім ініціації імпульсів присутності імпульс скидання переводить у вихідний стан всю систему.

Тривалість імпульсу скидання має бути не менше 480 мкс. Процес передачі інформації по лінії починатиметься не раніше, ніж через 480 мкс після закінчення дії імпульсу скидання. У цьому часовому інтервалі й очікується поява сигналу присутності. Для цього після закінчення імпульсу скидання Master «відпускає» лінію і чекає сигналу від Slave-пристроїв.

Кожен Slave-пристрій, виявивши імпульс скидання, витримує паузу в 15÷60 мкс, а потім «підсажує» лінію. Тривалість імпульсу присутності складає 60÷240 мкс. Сигнали присутності від всіх Slave-пристроїв зливаються в один загальний імпульс. Провідний пристрій перевіряє наявність нульового рівня на лінії в середині цього інтервалу. Якщо сигнал виявлено, то це означає, що на лінії є хоча б один нормально працюючий пристрій і Майстер Slave може продовжувати роботу в мережі.

Будь-яка операція в мережі MicroLAN починається з команди, яка є одним байтом інформації. Кожна команда має свій власний код. Виконання команди починається з імпульсу скидання. Потім Slave-пристрій виробляє, а Master перевіряє сигнал присутності на лінії.

Якщо сигнал на місці, Master видає на лінію код команди. Код видається в режимі запису.

6.1.5 Система команд протоколу 1 –Wire

Розглянута вище робота мережі MicroLAN базується на декількох різних рівнях, кожен з них має свою назву:

- *фізичний рівень* - це схемне вирішення шини, рівні напруги, взаємодія елементів мережі на рівні електричних сигналів;

- *рівень зв'язку* - це правила формування сигналу скидання, синхронізація і спосіб передачі бітів інформації в обох напрямках.

Ці два рівні визначаються фізичними особливостями мережі MicroLAN. Проте при описі протоколу прийнято виділяти ще два рівні, які, навпаки, пов'язані з логікою роботи протоколу. Це так звані *мережевий і транспортний рівні*. Будь-яка шина I-Wire завжди перебуває на одному з цих рівнів. Причому відразу після сигналу скидання шина переходить на мережевий рівень, і лише відпрацювавши команду мережевого рівня, шина переходить на транспортний. Команди мережевого рівня призначені для адресації Slave-мікросхем. Це команди типу «Вибрати таку-то мікросхему». Команди транспортного рівня призначені для безпосередньої передачі інформації. Це команди типу «записати», «прочитати». Всі команди мережевого рівня наведені в табл. 6.1, транспортного – в табл. 6.2.

Таблиця 6.1 – Команди мережевого рівня

Код	Команда	Опис
33H	Читання ПЗУ	Читання індивідуального коду веденої мікросхеми із спеціального ПЗП
0FH	Читання ПЗУ	Те ж саме, але для мікросхеми DS1990A (у цій мікросхемі використовується небагато інша система команд)
55H	Збіг ПЗУ	Пошук і активізація елемента із заданим кодом. Інші елементи переходять у пасивний режим
0CCH	Пропуск ПЗУ	Пропуск команди вибору пристрою. Застосовується, якщо потрібно працювати відразу зі всіма пристроями, або він лише один
0F0H	Пошук ПЗУ	Команда, що дозволяє здійснювати пошук елементів, підключених до шини. Одночасно з пошуком визначаються їх індивідуальні номери

Таблиця 6.2 –Команди транспортного рівня

Мікросхема	Код	Команда
DS1994 (4096-бітна перезаписувана енерго-незалежна пам'ять)	0FH	Запис блокнотної пам'яті
	0AAH	Читання блокнотної пам'яті
	55H	Копіювання блокнотної пам'яті
DS2417	66H	Читання часу
	99H	Запис часу
DS2450 4- канальний АЦП	0AAH	Читання з пам'яті результатів перетворення напруги
	55H	Запис в пам'ять команд, що управляють
DS1BS20/DS18B20/DS1822 Електронні датчики температури	44H	Запуск процесу конвертації температури в код
	4EH	Запис в блокнотну пам'ять(меж регулювання температури)
	0BEH	Читання блокнотної пам'яті
	48H	Копіювання блокнотної пам'яті в спеціальні вічка внутрішньою EEPROM
	0B8H	Читання інформації з EEPROM в блокнотну пам'ять
	0B4H	Читання інформації про режим живлення

6.1.6 Інтегральні датчики температури

Термодатчики DS1820 і DS18S20 мають робочий діапазон температур від -55°C до $+125^{\circ}\text{C}$. Максимальний час перетворення температури в код 750 мс. Результати вимірювання прочитуються по 1-Wire інтерфейсу із спеціальних внутрішніх регістрів мікросхеми у вигляді дев'ятирозрядного двійкового коду. Дев'ять двійкових розрядів є значенням температури в додатковому коді, вимірюного з кроком у $0,5^{\circ}\text{C}$.

Точність вимірювання температури різна в різних точках робочого діапазону. На відрізку $-10...+85\text{ }^{\circ}\text{C}$ точність вимірювання дорівнює $\pm 0,5\text{ }^{\circ}\text{C}$. Чим ближче до краю діапазону, тим точність вимірювання менша. На самих краях вона складає всього $\pm 2\text{ }^{\circ}\text{C}$.

Більш досконала мікросхема DS18B20 не вимагає жодних додаткових обчислень. Висока дискретність досягається збільшенням кількості розрядів результуючого коду. Причому в мікросхемі є можливість зміни кількості розрядів вихідного регістра. За умовчанням вихідний регістр має 9 розрядів. Змінюючи вміст регістра конфігурації, мікроконтролер може збільшити кількість розрядів до 12. Точність вимірювання температури в діапазоні $-10...+85\text{ }^{\circ}\text{C}$ складає $\pm 0,5\text{ }^{\circ}\text{C}$. На виході мікросхеми DS18B20 отримуємо прямий код, значення якого дорівнює величині вимірюваної температури. У 9-розрядному режимі значення вимірюваної температури видається з дискретністю в $0,5\text{ }^{\circ}\text{C}$. У дванадцятирозрядному режимі кількість відліків підвищується у вісім разів. Максимальний час перетворення для мікросхеми DS18B20 також залежить від обраної кількості розрядів. Для 12-розрядного режиму роботи воно дорівнює 750 мс. На сьогодні мікросхема DS18B20 є найпоширенішим видом мікросхем подібного призначення.

6.2 Завдання до лабораторної роботи

6.2.1 Ознайомитися з технічними характеристиками і архітектурою температурних датчиків DS18B20.

6.2.2 Ознайомитися з підпрограмами, які реалізують найнижчий рівень протоколу 1-Wire шини (Додаток Г).

6.2.3 Нарисувати блок-схему алгоритму роботи МК і написати програму на асемблері для здійснення операції читання ID-коду мікросхеми, яка підключена і працює за протоколом шини 1-Wire.

6.2.4 Нарисувати блок-схему алгоритму і написати програму на асемблері для опитування мікросхеми DS18B20. Передбачити зчитування 8 бітного коду і його відображення на світлодіодах, що підключені до порту МК.

6.2.5 Нарисувати блок-схему алгоритму і написати програму на асемблері для опитування мікросхеми DS18B20. Передбачити зчитування 12 бітного коду і відображення значення температури на РКІ-індикатор.

6.3 Контрольні запитання

- 1 Особливості та організація 1-Wire шини.
- 2 Як здійснюється читання інформації?
- 3 Як здійснюється запис байта інформації?
- 4 Як здійснити передачу інформації від одного Slave-пристрою (IC) до іншого Slave-пристрою?
- 5 Команди мережевого рівня.
- 6 Команди транспортного рівня.
- 7 Характеристика IC DS18B20.
- 8 Архітектура IC DS18B20.
- 9 Загальна схема портів Slave- і Master-пристроїв.
- 10 Алгоритм пошуку усіх пристроїв, підключених до 1-Wire шини.
- 11 ID-код.

7 ЛАБОРАТОРНА РОБОТА №7 „ПОСЛІДОВНИЙ ВВІД – ВИВІД. UART”

Мета роботи – ознайомитися із принципами організації послідовного вводу-виводу інформації через UART. Вивчити функціонування і програмування універсального приймача.

7.1 Загальна інформація про UART

Мікроконтролер ATmega8 має послідовний порт USART - *універсальний синхронний і асинхронний приймач*, призначений для організації послідовного зв'язку. Особливості:

- повнодуплексна робота (роздільні регістри послідовного прийому і передачі);
- асинхронна або синхронна робота;
- провідне або підлегле тактування зв'язку в синхронному режимі роботи;
- висока роздільна здатність генератора швидкості зв'язку;
- підтримка формату даних, які передаються, з 5, 6, 7, 8 або 9 бітами даних і 1 або 2 стоп-бітами;
- апаратна генерація і перевірка біта паритету (парність/непарність);
- визначення переповнювання даних;
- визначення помилки в структурі посилки;
- фільтрація шуму з детектуванням помилкового старт-біта і цифровим ФНЧ;
- три роздільні переривання після закінчення передачі, звільнення регістра передаваних даних і завершення прийому;
- режим багатопроцесорного зв'язку;
- режим подвоєння швидкості зв'язку в асинхронному режимі.

7.1.1 Ініціалізація USART

Перед початком сеансу зв'язку необхідно виконати ініціалізацію USART, яка зазвичай складається з установки швидкості зв'язку, завдання формату посилки і дозволу роботи передавача і приймача. Якщо використовується управління зв'язком по перериваннях, то під час ініціалізації необхідно, щоб був скинутий прапор загального дозволу переривань (необхідно заборонити всі переривання).

Якщо необхідно виконати повторну ініціалізацію USART, наприклад, для зміни швидкості зв'язку або формату посилки, то необхідно переконатися, аби під час ініціалізації передавача була припинена. Прапор TXC може використовуватися для перевірки завершення роботи передавача, а прапор RXC - для перевірки відсутності в приймальному буфері нелічених даних. Зауважимо, що при використанні прапора TXC він повинен скидатися програмно перед початком кожної передачі (перед записом в UDR).

7.1.2 Передача даних - Передавач USART

Робота передавача USART дозволяється установкою біта дозволу передачі (TXEN) в регістрі UCSRB. Після дозволу функція виведення TXD як звичайного порту замінюється на функцію виходу послідовної передачі даних. Швидкість зв'язку, режим роботи і формат посилки мають встановлюватися одноразово перед початком передачі.

Початок передачі ініціюється записом передаваних даних в буфер передавача. Буферизовані дані в буфері передавача будуть переміщені в регістр зсуву після того як він буде готовий до відправки нової посилки. Запис в регістр зсуву нових даних відбувається в стані чекання (коли передача завершена) або відразу після завершення передачі останнього стоп-біта попередньої посилки. Якщо в регістр зсуву записані нові дані, то починається передача однієї посилки на швидкості, визначеній в регістрі швидкості зв'язку.

7.1.3 Прапори і переривання передавача

Передавач USART має два прапори, які відображають його стан: прапор звільнення регістра даних UDRE і прапор завершення передачі TXC в регістрі UCSRA. Обидва прапори можуть використовуватися для генерації переривань.

Прапор звільнення регістра даних UDRE відображає готовність буфера передавача прийняти нові дані. Даний біт встановлюється при звільненні передавального буфера, і скидається коли буфер передавача містить дані для передачі.

Прапор завершення передачі TXC набуває одиничного значення, якщо вся посилка в регістрі зсуву передавача була повністю зсунута (передана) і в буфері передавача відсутні нові дані для передачі. Прапор TXC автоматично скидається при переході на вектор

обробки переривання по завершенню передачі або програмно скидається шляхом запису в нього логічної 1.

7.1.4 Прийом даних - Приймач USART

Робота приймача USART дозволяється, якщо записати логічну 1 в біт дозволу роботи приймача RXEN в регістрі UCSRB. Після дозволу роботи приймача звичайне призначення виведення RXD замінюється на вхід послідовного введення даних приймача USART. Швидкість зв'язку, режим роботи і формат посилки мають бути встановлені одноразово перед початком виконання прийому даних.

Приймач починає прийом даних лише після визначення дійсного старт-біта. Вибірка наступних за старт-бітом біт даних відбувається з частотою, рівною швидкості зв'язку. Після отримання першого стоп-біта, тобто коли послідовна посилка повністю прийнята і знаходиться в регістрі зсуву приймача, вміст регістра зсуву переміщається в приймальний буфер. Приймальний буфер зчитується при читанні регістра вводу-виводу UDR.

7.1.5 Прапор і переривання по завершенню прийому

Прапор завершення прийому RXC у регістрі UCSRA сигналізує про наявність нелічених даних у приймальному буфері. Даний прапор дорівнює 1, якщо є нелічені дані, і рівний 0, якщо буфер приймача вільний (тобто не містить яких-небудь нелічених даних). Якщо приймач відключається (RXEN=0), то приймальний буфер буде скинутий і прапор RXC набуде нульового значення.

Якщо встановлений біт дозволу переривання по завершенню прийому RXCIE в регістрі UCSRB, то при установці прапора RXC програма переходить на вектор обробки даного переривання.

Буферні регістри даних передавача і приймача USART розташовані поодиночі і в тій же адресі в області вводу-виводу, позначеною як регістр даних UDR. Якщо виконувати запис за адресою регістра UDR, то записувані дані поміщаються в буферний регістр даних передавача. Аналогічно при читанні регістра UDR витягується вміст буферного регістра даних приймача.

Таблиця 7.1 - Адреси використовуваних в приймачі UART0 регістрів

	UDR	UCSRA	UCSRB	UCSRC	UBRRH	UBRRL
USART0 (x=0)	\$0C (\$2C)	\$0B (\$2B)	\$0A (\$2A)	(\$95)	(\$90)	\$09 (\$29)
USART1 (x=1)	(\$9C)	(\$9B)	(\$9A)	(\$9D)	(\$98)	(\$99)

Запис у буфер передавача можна виконувати, якщо встановлений прапор UDRE в регістрі UCSRA (табл. 7.2). Дані, записані в UDR при скинутому прапорі UDRE, будуть ігноровані передавачем.

Таблиця 7.2 - Регістр А управління і статусу USART – UCSRA

Розряд	7	6	5	4	3	2	1	0
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
Читання /запис	Чит.	Чит./Зап.	Чит.	Чит.	Чит.	Чит.	Чит./Зап.	Чит./Зап.
Вих. значення	0	0	0	0	0	0	0	0

Розряд 7 – **RXC** - прапор завершення прийому. Даний прапор встановлюється, якщо в приймальному буфері містяться нелічені дані і скидається, коли приймальний буфер вільний (тобто, не містить нелічених даних).

Розряд 6 – **TXC** - прапор завершення передачі. Даний прапор встановлюється, якщо вся послідовність з регістру зсуву передавача повністю передана, і в передавальному буфері UDR немає нових даних для передачі.

Розряд 5 – **UDRE** - прапор звільнення регістра даних USART. Прапор UDRE відображає готовність приймального буфера UDR до прийому нових даних.

Розряд 4 – **FE** – помилка послідовності. Даний біт встановлюється, якщо при прийомі послідовності була визначена помилка в структурі послідовності. Помилкою структури в даному випадку є нульове значення першого стоп-біта в цій послідовності. Прапор FE набуває нульового значення, якщо прийнятий стоп-біт мав правильне одиничне значення.

Розряд 3 – **DOR** - прапор переповнювання даних. Даний біт встановлюється, якщо виявлена умова переповнювання. Переповнювання даних виникає, якщо заповнений приймальний буфер.

Розряд 2 – **UPE** – помилка паритету. Даний біт встановлюється, якщо прийняті дані характеризуються помилкою паритету за умови дозволу контролю паритету ($UPM1 = 1$).

Розряд 1 - **U2X** - подвоєння швидкості зв'язку. Цей біт використовують лише в асинхронному режимі зв'язку. Запис у даний біт логічної 1 зменшує в два рази значення коефіцієнта ділення швидкості зв'язку з 16 до 8, тим самим подвоюючи швидкість передачі даних в асинхронному режимі.

Розряд 0 – **MPCM** - режим багатопроцесорного зв'язку. Даний біт визначає режим багатопроцесорного зв'язку. Якщо в біт MPCM записати логічну 1, то всі вхідні дані, що приймаються приймачем USART, ігноруватимуться, якщо вони не містять адресної інформації. Стан біта MPCM не впливає на роботу передавача.

Таблиця 7.3 - Регістр В управління і статусу - UCSRB

Розряд	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Читання /запис	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит.	Чит./ зап.
Вих. значення	0	0	0	0	0	0	0	0

Розряд 7 - **RXCIE** - дозвіл переривання по завершенню прийому. Запис у даний біт логічної 1 дозволяє переривання по прапору RXC.

Розряд 6 - **TXCIE** - дозвіл переривання по завершенню передачі. Запис в даний біт логічної 1 дозволяє переривання по прапору TXC.

Розряд 5 - **UDRIE** - дозвіл переривання по звільненню регістра даних USART. Установка даного прапора дозволяє переривання по прапору UDRE.

Розряд 4 - **RXEN** - дозвіл роботи приймача. Запис в даний біт логічної 1 приводить до дозволу роботи приймача USART.

Розряд 3 - **TXEN** - дозвіл роботи передавача. Запис у даний біт логічної 1 дозволяє роботу передавача USART.

Розряд 2 - **UCSZ2** - формат даних. Біт UCSZ2 разом з бітами UCSZ1:0 в регістрі UCSRC задають кількість біт даних в посилці, як для приймача, так і для передавача.

Розряд 1 - **RXB8** - значення 8-ого розряду прийнятих даних. RXB8 містить значення 9-го біта прийнятої посилки з 9-бітовим форматом. Даний біт необхідно рахувати перш, ніж будуть лічені молодші 8 біт з регістра UDR.

Розряд 0 - **TXB8** - 8-ой розряд передаваних даних. TXB8 містить значення 9-ого біта даних для передачі посилки з 9-бітовим форматом. Даний біт необхідно записати перед тим, як будуть записані молодші розряди даних в UDR.

Таблиця 7.4 – Регістр С керування і статусу USART –UCSRC

Розряд	7	6	5	4	3	2	1	0
	-	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Читання /запис	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.
Вих. значення	0	0	0	0	0	0	0	0

Розряд 7 - зарезервовані біти.

Розряд 6 - **UMSEL** - вибір режиму USART. Даний біт дозволяє перемикається між асинхронним (UMSEL=0) і синхронним (UMSEL=1) режимами послідовного зв'язку.

Розряди 5:4 - **UPM1:0** - режим паритету. Дані біта дозволяють і встановлюють типи генерованого і контрольованого паритету (табл.7.5). Після дозволу паритету передавач автоматично генерує і передає біт паритету в кожній посилці. Приймач генерує біт паритету для прийнятих даних і порівнює його із значенням прийнятого в цій

Таблиця 7.5 – Установка бітів UPM

UPM1	UPM0	Режим паритету
0	0	Відключений
0	1	(Резерв)
1	0	Парність
1	1	Непарність

посилці біта паритету, а за результатом порівняння встановлює прапор помилки паритету UPE в регістрі UCSRA.

Розряд 3 - **USBS** - вибір числа стоп-біта. Даний біт визначає, скільки стопових бітів виставляє передавач при генерації послідовності (табл. 7.6). Приймач ігнорує це налаштування.

Таблиця 7.6 – Установки біта USBS

USBSn	Число стоп-біт
0	1 біт
1	2 біта

Розряди 2:1 - **UCSZ1:0** – формат даних. Біти UCSZ1:0 разом з UCSZ2 у регістрі UCSRB задають кількість біт даних в послідовності, як для приймача, так і для передавача (табл. 7.7).

Таблиця 7.7 – Установка бітів UCSZ

UCSZ2	UCSZ1	UCSZ0	Формат даних
0	0	0	5 біт
0	0	1	6 біт
0	1	0	7 біт
0	1	1	8 біт
1	0	0	Резерв
1	0	1	Резерв
1	1	0	Резерв
1	1	1	9 біт

Розряд 0 – **UCPOL** – полярність синхронізації. Даний біт використовується лише в синхронному режимі. Якщо використовується асинхронний режим, то в даний біт необхідно записати лог. 0. У синхронному режимі біт UCPOL визначає співвідношення між вибіркою вхідних даних і оновленням передаваних даних і тактуючим сигналом синхронного зв'язку (ХСК) (табл. 7.8).

Розряди 15:12 - зарезервовані розряди.

Розряди 11:0 - **UBRR11:0** - регістр швидкості зв'язку USART.

UBRR - 12-розр. регістр, який задає значення швидкості зв'язку USART. Регістр UBRRH містить 4 старших розряди, а UBRL - 8

молодших розрядів значення швидкості USART. Якщо під час передачі або прийому змінити швидкість зв'язку, то сеанс зв'язку буде порушений. Запис у регістр UBRR1 ініціює оновлення швидкості зв'язку (табл. 7.10).

Таблиця 7.8 – Установки біт UCPOL

UCPOL	Зміна передаваних даних на виході TxDx	Вибірка даних, що приймаються, на вході RxDx
0	Наростаючий фронт ХСК	Спадаючий фронт ХСК
1	Спадаючий фронт ХСК	Наростаючий фронт ХСК

Таблиця 7.9 – Регістри швидкості зв'язку USART - UBRR1 і UBRR0

Розряд	7	6	5	4	3	2	1	0	
	-	-	-	-	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
Читання /запис	Чит.	Чит.	Чит.	Чит.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	
	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	Чит./ зап.	
Вих. значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Таблиця 7.10 – Вираз для обчислення установок регістра швидкості зв'язку

Режим роботи	Вираз для розрахунку швидкості зв'язку	Вираз для розрахунку значення UBRR
Нормальний асинхронний режим (U2=0)	$BAUD = \frac{f_{osc}}{16(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{16BAUD} - 1$
Асинхронний режим з подвоєнням швидкості (U2=1)	$BAUD = \frac{f_{osc}}{8(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{8BAUD} - 1$
Синхронний відомим режим	$BAUD = \frac{f_{osc}}{2(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{2BAUD} - 1$

У таблиці прийняті такі позначення:

- BAUD - швидкість зв'язку (у бітах в секунду, бод);
- f_{OSC} - тактова частота мікроконтролера;
- UBRR - вміст регістрів UBRRH і UBRL (0- 4095).

7.2 Завдання до лабораторної роботи

7.2.1 Написати програму по прийому даних, в якій кількість байтів, що приймаються, визначається першим прийнятим байтом, а швидкість роботи приймача складає 57600 біт/с.

7.2.2 Написати програму по прийому даних, в якій прийом даних здійснюється без використання переривання по прийому даних, а за допомогою прапора прийнятих даних. Встановити формат послідовності: 8 біт даних, 2 стоп-біта.

7.2.3 Написати програму по прийому даних, в якій перші 4 байти прийнятих даних порівнюються з подальшими 4-ма байтами.

7.2.4 Написати програму по прийому даних, в якій кожен прийнятий байт відсилається назад відразу після отримання. Встановити формат послідовності: 8 біт даних, 2 стоп-біта, швидкість 14400.

7.2.5 Написати програму по прийому даних, в якій кожен байт прийнятих байт відсилається назад. Встановити формат послідовності: 8 біт даних, 1 стоп-біт, швидкість 28800.

7.2.6 Написати програму по прийому даних, в якій перші 4 байти прийнятих даних порівнюються з подальшими 4-ма байтами, і в разі збігу у відповідь посилаються байти з 3-го по 5-й. Встановити формат послідовності: 8 біт даних, 2 стоп-біта, швидкість 57600.

7.3 Порядок виконання лабораторної роботи

7.3.1 Розробити алгоритм для індивідуального завдання.

7.3.2 Розробити програму для індивідуального завдання.

7.3.3 Написати програму індивідуального завдання на ПК.

7.3.4 За допомогою програм моделювання проаналізувати виконання індивідуальної програми.

7.3.5 Завантажити програму в стенд. Переконатися в правильному виконанні індивідуального завдання. При негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд.

7.3.6 Зберегти лістинг правильно працюючої програми.

7.3.7 Показати результат виконання програми викладачеві і відповісти на контрольні запитання.

7.4 Контрольні запитання

- 1 Структура універсального асинхронного приймача.
- 2 Особливості програмування універсального асинхронного приймача.
- 3 Як розрахувати швидкість передачі?
- 4 Призначення регістрів UBRR.
- 5 Призначення регістрів UCSRC.
- 6 Призначення регістрів UCSRB.

8 ЛАБОРАТОРНА РОБОТА №8 „КЕРУВАННЯ КРОКОВИМ ДВИГУНОМ ЗА ДОПОМОГОЮ МПС”

Мета роботи – ознайомитися із будовою і принципом роботи крокових двигунів, отримати навички керування кроковим двигуном (КД) за допомогою мікроконтролера.

8.1 Теоретична інформація про крокові двигуни

8.1.1 Визначення КД, переваги і недоліки

Крокові двигуни – це пристрої, які перетворюють електричні імпульси напруги керування в дискретні (стрибокоподібні) кутові або лінійні переміщення ротора з можливою його фіксацією в потрібних положеннях.

Переваги КД:

- кут повороту ротора двигуна пропорційний кількості вхідних імпульсів;
- двигун працює з повним моментом у стані спокою (якщо обмотки підключені до живлення);
- прецизійне позиціонування і повторюваність кроку (хороші крокові двигуни мають точність близько $3\div 5\%$ кроку, і ці помилки не накопичуються від кроку до кроку);
- можливість швидкого розгону, гальмування і зміни напрямку руху;
- безвідмовність - через відсутність щіток; довговічність двигуна залежить тільки від довговічності підшипників;
- залежність обертів двигуна від дискретних імпульсів дозволяє керувати двигуном без зворотного зв'язку, завдяки чому кроковий двигун простіший і дешевший у керуванні;
- можливість досягнення дуже низьких швидкостей обертання з навантаженням, закріпленим безпосередньо на осі двигуна;
- широкий діапазон швидкостей обертання завдяки тому, що швидкість пропорційна частоті вхідних імпульсів.

Недоліки КД:

- механічний резонанс з'являється при неправильному керуванні;
- складнощі при роботі з надвисокими швидкостями.

Обмотки крокового двигуна є частиною статора. На роторі розташований постійний магніт або зубчастий блок з магнітом'якого матеріалу в разі двигуна із змінним магнітним опором. Комутації обмоток здійснюються зовнішніми схемами.

8.1.2 Принцип роботи і конструкція КД

Двигуни постійного струму (ДПС) Lenze починають працювати відразу після прикладення постійної напруги. Перемикання напруги струму через обмотки ротора здійснюється механічним комутатором - колектором. Постійні магніти при цьому розташовані на статорі. Кроковий двигун можна розглядати як ДПС без комутатора. Обмотки його є частиною статора. На роторі розташований постійний магніт або, для випадків КД із змінним магнітним опором, зубчастий блок з магнітом'якого матеріалу. Всі комутації здійснюються зовнішніми схемами. Зазвичай система мотор - контролер розробляють так, щоб була можливість виведення ротора в будь-яку фіксовану позицію, тобто коли система керується по положенню. Циклічність позиціонування ротора залежить від його геометрії.

Прийнято розрізняти крокові двигуни (Autonics, Motionking, Fulling motor) і серводвигуни (Lenze). Принцип їх дії багато в чому схожий, і багато контролерів можуть працювати з обома типами. Основна відмінність полягає в кількості кроків на цикл (один поворот ротора). Серводвигуни вимагають наявності в системі управління аналогового зворотного зв'язку (наприклад, потенціометра). Крокові двигуни переважно використовуються в системах без зворотних зв'язків, які потребують невеликих прискорень під час руху.

Крокові двигуни поділяють на два різновиди: двигуни з постійними магнітами і двигуни зі змінним магнітним опором (гібридні двигуни). З точки зору контролера відмінності між ними немає. Двигуни з постійними магнітами зазвичай мають дві незалежні обмотки, в яких є або немає серединного відведення (рис. 8.1).

Біполярні крокові двигуни з постійними магнітами і гібридні двигуни сконструйовані простіше, ніж уніполярні двигуни, обмотки в них не мають центрального відведення (рис. 8.2).

За це спрощення доводиться платити складнішим реверсуваням полярності кожної пари полюсів двигуна.

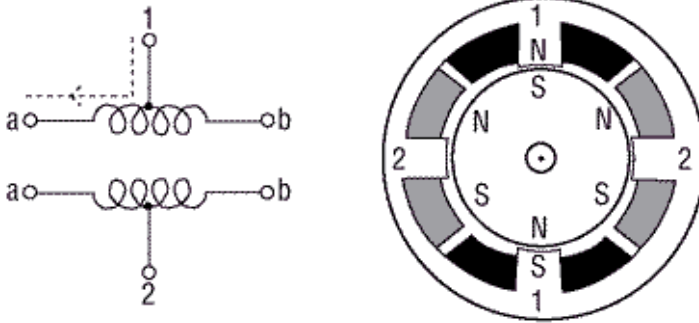


Рисунок 8.1 – Уніполярний КД з постійними магнітами

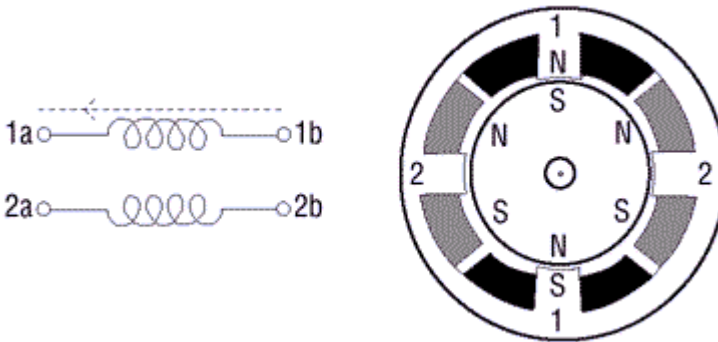


Рисунок 8.2 – Біполярний і гібридний КД

Крокові двигуни мають широкий діапазон кутових дозволів. Грубіші мотори зазвичай обертаються на 90° за крок, тоді як прецизійні двигуни можуть мати дозвіл $1,8^\circ$ або $0,72^\circ$ на крок. Якщо контролер дозволяє, то можливе використання напівкрокового режиму або режиму з більш дрібним дробленням кроку (мікрокроковий режим), при цьому на обмотки подаються дробові значення напруги (часто формуються за допомогою ШІМ).

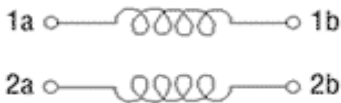
Якщо в процесі управління використовується збудження лише однієї обмотки у будь-який момент часу, то ротор буде обертатися на фіксований кут, який стримуватиметься, доки зовнішній момент не перевищить моменту утримання двигуна в точці рівноваги.

8.1.3 Керування КД

Для правильного керування біполярним кроковим двигуном необхідна електрична схема, яка має виконувати функції старту, стопа, реверсу і зміни швидкості. Кроковий двигун транслює послідовність цифрових перемикачів в рух. Магнітне поле, яке "обертається", забезпечується відповідними перемиканнями напруги на обмотках. Услід за цим полем обертатиметься ротор, сполучений за допомогою редуктора з вихідним валом двигуна.

Схема управління для біполярного крокового двигуна вимагає наявності мостової схеми для кожної обмотки. Ця схема дозволить незалежно міняти полярність напруги в кожній обмотці.

На рис. 8.3 показана послідовність управління для режиму з одиничним кроком.



Індекс	1a	1b	2a	2b
1	+	-	-	-
2	-	+	-	-
3	-	-	+	-
4	-	-	-	+
5	+	-	-	-
6	-	+	-	-
7	-	-	+	-
8	-	-	-	+

Рисунок 8.3 – Керуюча послідовність для режиму з одиничним кроком

Максимальна швидкість руху визначається з фізичних можливостей крокового двигуна. При цьому швидкість регулюється зміною розміру кроку: більші кроки відповідають більшій швидкості руху.

У табл.8.1 наведено послідовність для напівкрокового керування.

Таблиця 8.2 – Керуюча послідовність для режиму з половинним кроком

індекс	1a	1b	2a	2b
1	+	-	-	-
2	+	+	-	-
3	-	+	-	-
4	-	+	+	-
5	-	-	+	-
6	-	-	+	+
7	-	-	-	+
8	+	-	-	+
9	+	-	-	-
10	+	+	-	-
11	-	+	-	-
12	-	+	+	-
13	-	-	+	-
14	-	-	+	+
15	-	-	-	+
16	-	-	-	+

8.2 Завдання до лабораторної роботи

8.2.1 Ознайомитися із схемою електричною принциповою стенду (рис. 8.5). Вивчити документацію. Розробити алгоритм роботи і програмне забезпечення для керування роботою крокового двигуна: забезпечити обертання з частотою (№варіанту/5) обертів за хвилину.

8.2.2 Розробити алгоритм і програмне забезпечення для керування роботою крокового двигуна: забезпечити можливість реверсивного обертання крокового двигуна з частотою 5 обертів за хвилину.

8.2.3 Розробити алгоритм і програмне забезпечення для керування роботою крокового двигуна: забезпечити можливість реверсивного обертання крокового двигуна із змінною частотою обертання. Зміну частоти обертання здійснити з клавіатури, поточне значення потрібно виводити на індикатор.

8.3 Контрольні запитання

- 1 Класифікація крокових двигунів.
- 2 Переваги та недоліки КД у порівнянні з іншими видами двигунів.
- 3 Принцип роботи крокового двигуна.
- 4 Апаратні засоби для побудови схем на крокових двигунах.
- 5 У чому полягає відмінність КД від сервоприводу?
- 6 Алгоритм керування уніполярним КД з одиночним кроком.
- 7 Алгоритм керування біполярним КД з половинним кроком.

Додаток А
Приклад оформлення титульної сторінки
Міністерство освіти та науки України

Запорізький національний технічний університет

Кафедра МiHE

ЗВІТ
З ЛАБОРАТОРНОЇ РОБОТИ №__

(тема роботи)
з дисципліни „МПКОІ“

Виконав:
студент гр. РП__

(Ініціали, Прізвище)

Прийняв:
(посада)

(Ініціали, Прізвище)

201__ р.

Додаток Б Архітектура ATmega8

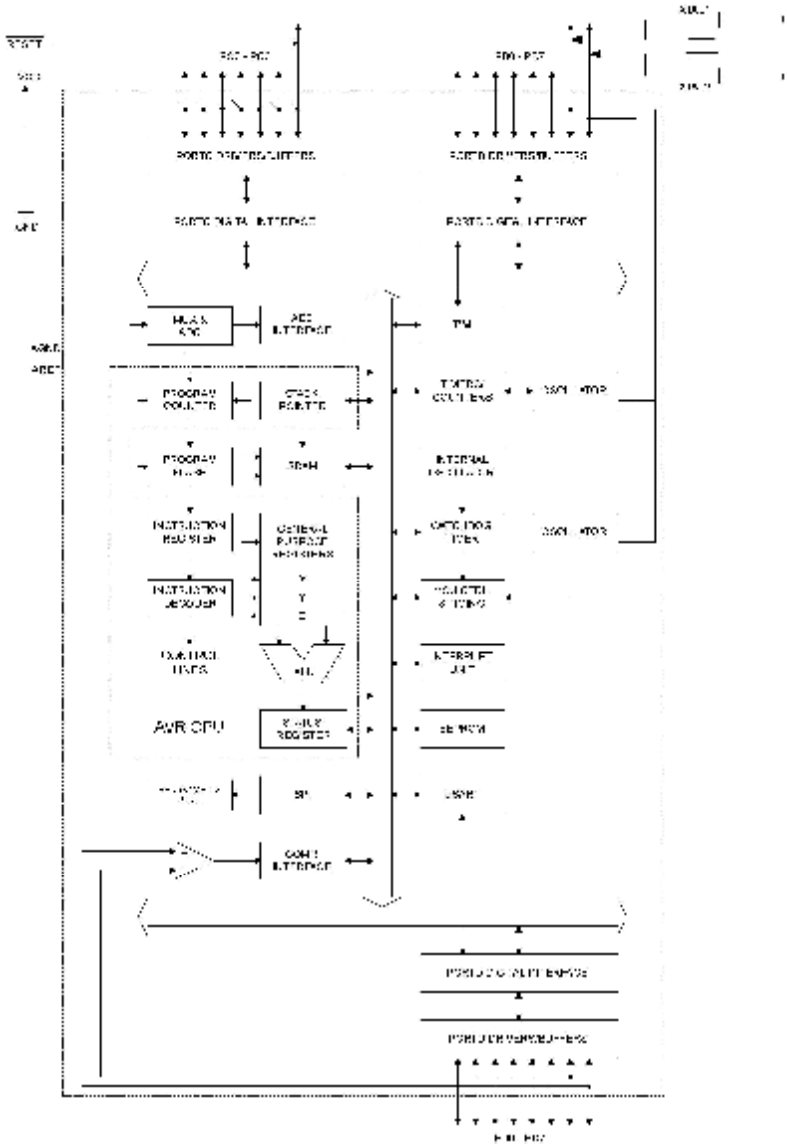


Рисунок Б.1 – Архітектура ATmega8

Додаток В

Текст програми на асемблері до лабораторної роботи №1

```

.include "m8def.inc"
; Перша лабораторна робота
; Виконав студент ПІБ
; Група ПП-XX

.def tmp = r16
.def tmp1 = r17
.equ db4 = PD4

.cseg
.org 0
rjmp RESET ;Reset Handler
    nop ;rjmp EXT_INT0 ;IRQ0 Handler
    nop ;rjmp EXT_INT1 ;IRQ1 Handler
    nop ;rjmp TIM2_COMP ;Timer2 Compare Handler
    nop ;rjmp TIM2_OVF ;Timer2 Overflow Handler
    nop ;rjmp TIM1_CAPT ;Timer1 Capture Handler
    nop ;rjmp TIM1_COMPA ;Timer1 CompareA Handler
    nop ;rjmp TIM1_COMPB ;Timer1 CompareB Handler
    nop ;rjmp TIM1_OVF ;Timer1 Overflow Handler
    nop ;rjmp TIM0_OVF ;Timer0 Overflow Handler
    nop ;rjmp SPI_STC ;SPI Transfer Complete Handler
    nop ;rjmp USART_RXC ;USART RX Complete Handler
    nop ;rjmp USART_UDRE ;UDR Empty Handler
    nop ;rjmp USART_TXC ;USART TX Complete Handler
    nop ;rjmp ADC ;ADC Conversion Complete Handler
    nop ;rjmp EE_RDY ;EEPROM Ready Handler
    nop ;rjmp ANA_COMP ;Analog Comparator Handler
    nop ;rjmp TWSI ;Two-wire Serial Interface Handler
    nop ;rjmp SPM_RDY ;Store Program Memory Ready Handler

RESET:
    ldi r16,high(RAMEND) ; Main program start
    out SPH,r16 ;Set Stack Pointer to top of RAM
    ldi r16,low(RAMEND)

```

```

        out SPL,r16
; Початок коду на асемблері
rcall port_init          ;виклик підпрограми ініціалізація портів
rcall del                ; виклик підпрограми затримки

; нескінчений цикл
loop:
.....
rjmp loop

; Підпрограма ініціалізації портів
port_init:
    ldi tmp,$ff
    out ddrc,tmp
    out ddrd, tmp
    ldi tmp, 0b00000000
    out portc,tmp
    out portd,tmp
    out ddrb,tmp
    ldi tmp,$ff
    out portb,tmp
ret

;Підпрограма затримки
del:
    push tmp
    ldi tmp,45
    m4: dec tmp
    brne m4
    pop tmp
ret

```

Додаток Г

Підпрограми для реалізації протоколу 1-Wire шини

Г.1 Процедура формування сигналу початкового скидання

resLAN:

```

    cbi      portb,wirelan      ; 'закорачиваем' линию pb1
    ldi pin_s,$ff                ; перевод линии в режим «Выхода»
                                out ddrb,pin_s      ;

                                ldi tmp,22 ; задержка 700 мкс
                                m10:
                                    ldi tmp1,41
                                    nop
                                m11: dec tmp1
                                    brne m11
                                    dec tmp
                                    nop
                                    brne m10

                                sbi      portb,wirelan      ; 'отпускаем'линию pb1

                                ldi pin_s,$00                ;перевод линии в режим Входа
                                out ddrb,pin_s                ;
                                ldi tmp,86                    ;Задержка в 65 мкс
                                m12: dec tmp                  ;
                                brne m12

                                SBIC     PINB,wirelan          ; ПРОВЕРКА НУЛЯ НА ЛИНИИ
                                rjmp  rst1                      ; Проверка сигнала присутствия

                                ldi tmp,15                    ; Задержка в 500 мкс
                                m13: ldi tmp1,43
                                m14: dec tmp1
                                brne m14
                                dec tmp
                                nop
                                brne m13
                                nop
                                nop
                                nop

                                SBIC     PINB,wirelan          ; Если на линии 1 переход

```

```

rjmp rst2
ldi prisut,2 ; Код ошибки "КЗ в линии"
ret
rst2: ldi prisut,0 ; Код нормального завершения
ret
rst1: ldi prisut,1 ; Код ошибки "Нет датчика"
RET

```

Г.2 Підпрограма запису байта

wr8LAN:

```

ldi tmp,8
ldi count,0 ; Счетчик к-ва разрядов
wb1: ror bait_w ; Младший бит в 'с'
inc count
brcs wb2 ; Если текущий С бит = 1

rcall mwlw ; запись 0 в слот
rjmp wb3

wb2: rcall mwhi ; запись 1 в слот
wb3: CPSE count,tmp
rjmp wb1 ; Продолжать, если не последний
ret

```

Г.3 Підпрограма запису біта

wr1LAN:

```

brcs wb4 ; Если бит = 0
rcall mwlw ; запись 0 в слот
rjmp vb5

wb4: rcall mwhi ; запись 1 в слот
vb5: ret

```

Г.4 Читання байта

rd8LAN:

```

push count
        ldi count,8      ; Счетчик количества битов
rb1:    rcall mrslot      ; Чтение очередного бита
        ROR bait_r       ; сдвиг бита С в регистр
        dec count        ; младшим разрядом вперед
        brne rb1         ; Продолжать, если не последний
        pop count
        ret

```

Г.5 Читання біта

rd1LAN:

```

rcall mrslot
ret

```

Г.6 Читання слота

; результат чтения (1 бит) возвращается через признак переноса

mrslot:

```

        ldi pin_s,$ff    ;перевод линии в режим Выхода
        out ddrb,pin_s   ;
        cbi portb, wirelan ;Закоротить линию 1 mksec

nop
nop

        nop
        nop

        ldi pin_s,$00    ; перевод линии в режим Входа
        out ddrb,pin_s   ;

        sbi portb, wirelan ; Отпустить линию получилось xx мкс

        ldi LoopCnt,2    ; Задержка 13 мкс
msl2:    rcall del4_5

        nop
        dec LoopCnt      ;
        brne msl2

        SBIC PINB, wirelan ; пропустить команду если бит 0
        rjmp msl3
        CLC              ; очистить бит переноса C
        rjmp msl1

```

```

msl3: SEC ; установить бит C в регистре SREG
msl1: ldi LoopCnt,14 ; Задержка в 76 мкс
msl0: rcall del4_5 ;
      nop
      dec LoopCnt
      brne msl0
RET

```

Г.7 Запис '0' у слот

```

mwl0: ldi pin_s,$ff ; перевод линии в режим Выхода
      out ddrb,pin_s ;
      CBI portb, wirelan ; Закоротить линию
      ldi LoopCnt,106 ; Задержка 80 мкс
mwl1: dec LoopCnt
      brne mwl1
      nop
      nop
      SBI portb, wirelan ; Отпустить линию
      RET

```

Г.8 Запис '1' у слот

```

mwhi: push count
      ldi pin_s,$ff ; перевод линии в режим Выхода
      out ddrb,pin_s ;
      cbi portb,wirelan ; Закоротить линию
      nop
      nop
      nop
      ;rcall del4_5 ; Задержка 4.5 мкс
      sbi portb, wirelan ; Отпустить линию

      ldi LoopCnt,100 ;Задержка в 75 мкс
      mwh1: dec LoopCnt
      brne mwh1
      pop count
      RET

```