МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

С.М. Цирульник, В.О. Денисюк

МІКРОПРОЦЕСОРНІ СИСТЕМИ УПРАВЛІННЯ

Навчальний посібник

Вінниця - 2021

ЗМІСТ

Лекція 1. Основні поняття мікропроцесорних систем. Кодування інформації

Мікропроцесор (МП), мікроконтролер (МК), мікрокомп'ютер (МКП) слова схожі, але за змістом різні. Спрощена структурна схема типового мікропроцесора показана на рис 1.1. У його основі – центральний процесорний пристрій (ЦПП), який містить арифметичний обчислювач, логічне ядро і регістри загального призначення. Із зовнішнім світом ЦПП спілкується за допомогою трьох шин: адреси, даних і управління. По цих же шинах в нього поступають коди програми керування, яка зберігається на зовнішньому носії. Початкова установка регістрів ЦПП виконується по сигналу скидання RESET, а синхронізація роботи здійснюється від тактових імпульсів SYN.

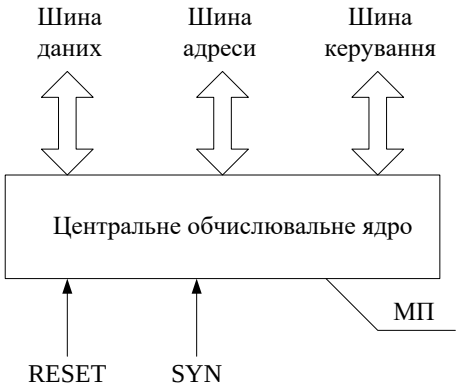

Рисунок 1.1 — Функціональна схема мікропроцесора (МП)

Якщо до ЦПП на кристал додати оперативний і постійний запам’ятовуючий пристрій (ОЗП, ПЗП), таймери, лічильники, аналого-цифрові і цифро-аналогові перетворювачі (АЦП, ЦАП), інтерфейсні вузли і порти введення/виведення, то мікропроцесор перетвориться на МК **(**рис. 1.2). Тактові імпульси виробляє вбудований синхрогенератор, частота якого стабілізується кварцовим резонатором. Для програмування ПЗП використовується окремий вхід PROG або шина з декількох сигналів.

Раніше МК називали однокристальними мікро ЕОМ, віддаючи належне вітчизняним мікросхемам К1816ВЕхх, КР1830ВЕхх. Однак ця назва не закріпилася, оскільки зараз переважні позиції на ринку займають МК

(microcontroller) закордонного виробництва сімейств AVR, MCS-51, PIC, Scenix, Z8.

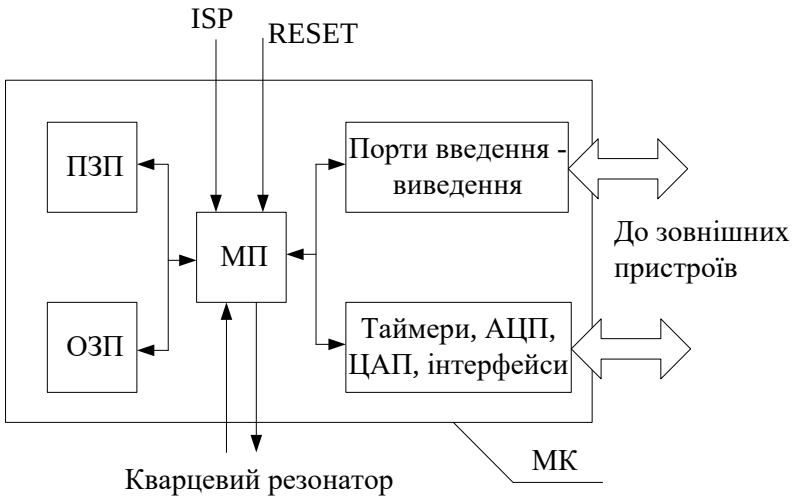

Рисунок 1.2 — Функціональна схема мікроконтролера (МК)

Мікроконвертор — це вдалий рекламний винахід фірми Analog Devices. Першим мікроконвертором був ADUC812, випущений в 1998 р. Ключове слово «MicroConverter» є офіційною торгівельною маркою і захищено юридичними правами фірми Analog Devices. Відноситься воно до лінійок мікросхем ADuC7xxx, ADuC8xx, що виконують функцію центрального ядра інтелектуальних систем збору інформації.

«Родзинкою» мікроконверторів є швидкодіючий прецизійний АЦП, доповнений універсальним блоком логічної обробки даних і багато розрядним ЦАП. Якщо врахувати наднизьке споживання струму і малі габарити мікроконверторів, то стає ясно, що спеціалізовані ІМС по праву займають свою нішу на ринку.

Проте, структурні схеми в мікроконверторів і МК повністю збігаються. Проте принципова різниця все ж є. Для звичайного МК спочатку вибирається цифрове обчислювальне ядро, а потім до нього додається АЦП і ЦАП. В протилежність цьому, ядром мікроконвертора спочатку служить зв'язка прецизійних АЦП і ЦАП, до яких додається процесор, що управляє.

Цифрові сигнальні процесори (англ. DSP - Digital Signal Processor) теж відносяться до мікроконтролерних пристроїв (рис. 1.3). Їх особливістю є обробка широкосмугових сигналів в режимі реального часу. Це характерно як для аудіо/відео техніки, так і для систем гнучкого управління роботизованими комплексами. Досягненню мети сприяє висока швидкодія ядра сигнального процесора (СП), багатопотокова система обслуговування пам'яті і наявність апаратних математичних команд, наприклад, для швидкого перетворення Фур'є. Звичайні МК такими можливостями не володіють.

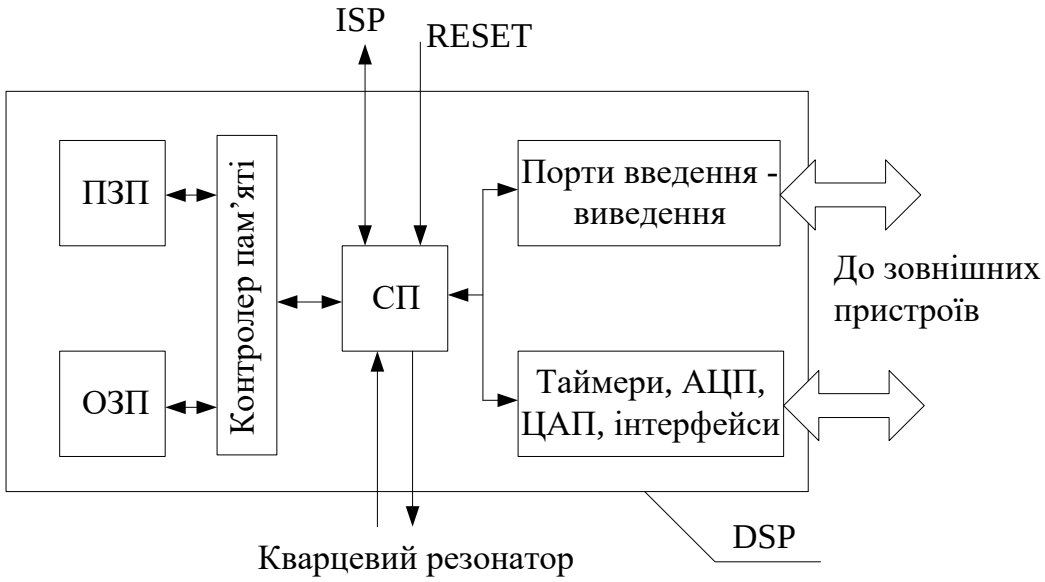

Рисунок 1.3 — Функціональна схема DSP

Перші DSP з'явилися в кінці 1970-х років через декілька років після перших МК, проте висока ціна і технологічні обмеження того часу не дозволили їм змагатися на рівних. Лише у 1982 р. фірма Texas Instruments зробила революційний прорив, випустивши в продаж перший універсальний програмований DSP TMS32010. Його концепція стала стандартом «де-факто» для всіх подальших сигнальних процесорів.

Відмінності в архітектурі і вузька спеціалізація привели до того, що напрям DSP/DSC виділився в окрему від МК сферу розробок з кількістю різновидів моделей більше 300. Вважається, що основною відмінністю DSP є відсутність

розвиненої системи команд управління, тобто умовних переходів, непрямих викликів підпрограм, які необхідні для виконання завдань з’єднання із зовнішніми об'єктами. Процесор в DSP і його системі команд орієнтовані на найвищу швидкість перетворення вхідних даних, що поступають. На управлінські «дрібниці» обчислювальних ресурсів вже не вистачає.

Сучасні МК запозичують від DSP апаратне множення і спеціалізацію команд, а DSP запозичують від МК універсальні інтерфейси введення/виведення і гнучкість в платформі програмування. Грані відмінностей поступово стираються.

На початку 1980-х років японська фірма Hitachi почала використовувати термін «мікрокомп'ютер», яким стали називати швидкодіючі процесори лінійки «Hitachi SUPERH». У рекламі можливостей чипів «SUPERH microcomputer SH7000 series» підкреслювалося, що на одній мікросхемі тепер можна побудувати систему керування реального часу, що перевищує за продуктивністю звичайний настільний мікрокомп'ютер.

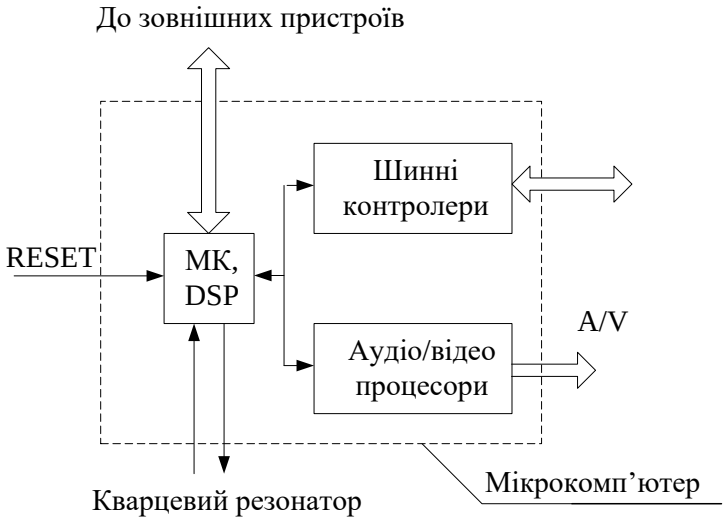

Рисунок 1.4 — Функціональна схема мікрокоп‘ютера

Сучасний мікрокомп'ютер (рис. 1.4) містить всі складові МК або DSP, але додатково має контролер шин для підключення зовнішньої високошвидкісної пам'яті, а також аудіо- і відеопроцесори. Останні, як правило, не поступаються

ЦПП за складністю та функціональністю. Приклади спрощених мікрокомп'ютерів з повсякденного життя — це однокристальні ВІС китайських клонів ігрових приставок «Dendy», «SEGA Mega Drive».

Мікропроцесор – це пристрій, який здійснює прийом, обробку і видачу інформації. Конструктивно МП містить одну або декілька інтегральних схем і виконує дії за програмою, записаною в пам'яті.

Мікропроцесорна система – обчислювальна, контрольно-вимірювальна або система керування, в якій основним пристроєм обробки інформації є МП. Мікропроцесорна система будується з набору мікропроцесорних ВІС.

Мультимікропроцесорна (або мультипроцесорна) система – система, яка утворюється об'єднанням деякої кількості універсальних або спеціалізованих МП, завдяки чому забезпечується паралельна обробка інформації і розподілене керування.

Мікропроцесорний комплект (МПК) — сукупність інтегральних схем, сумісних за електричними, інформаційними та конструктивними параметрами і призначених для побудови мікропроцесорних систем керування. Зазвичай МПК містить: ВІС МП (один чи кілька корпусів інтегральних схем); ВІС оперативних запам'ятовувальних пристроїв (ОЗП); ВІС постійних запам'ятовувальних пристроїв (ПЗП); інтерфейси або контролери зовнішніх пристроїв; службові ВІС (тактовий генератор, регістри, шинні формувачі, контролери шин, арбітри шин).

Жорстке розділення мікросхем на МК, мікропроцесори і DSP було характерне в кінці XX століття. У сьогодення грані відмінностей поступово стираються. МК усе частіше відносять до класу процесорів для вбудованих застосувань або, по-іншому, процесорів вбудованих систем (embedded processor). За визначенням, вбудовані обчислювальні системи – це системи, які безпосередньо, без постійної присутності людини, взаємодіють з датчиками і виконавчими пристроями керованого об'єкту.

Прикладами вбудованих систем є бортові і панельні комп'ютери, портативні вимірювальні прилади, системи відеоспостереження, роботи, мережеве устаткування, стільникові телефони.

Для досягнення максимальної універсальності і спрощення протоколів обміну інформацією в мікропроцесорних системах застосовується шинна структура зв'язку між окремими пристроями, що входять в систему.

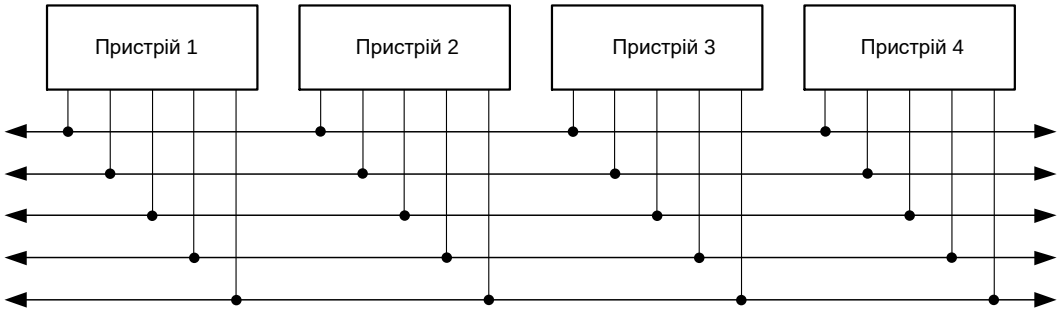

Рисунок 1.5 — Шинна структура зв’язків

При шинній структурі зв'язку (рис.1.5) всі сигнали між пристроями передаються по одних і тих же лініях зв'язку, але у різний час. Передача по всіх лініях зв'язку може здійснюватися в обох напрямах. У результаті кількість ліній зв'язку істотно скорочується, а правила обміну (протоколи) спрощуються. Група ліній зв'язку, по яким передаються сигнали або коди називається шиною (англ. bus). При шинній структурі зв'язку легко здійснюється пересилка всіх інформаційних потоків у потрібному напрямку, наприклад, їх можна пропустити через один процесор, що дуже важливо для мікропроцесорної системи. Проте при шинній структурі зв'язку вся інформація передається по лініях зв'язку послідовно в часі, по черзі, що знижує швидкодію системи у порівнянні з класичною структурою зв'язку.

Перевагою шинної структури зв'язку є те, що всі пристрої, що підключені до шини, повинні приймати і передавати інформацію за одними і тими ж правилами (протоколам обміну інформацією по шині). Відповідно, всі вузли, що відповідають за обмін з шиною в цих пристроях, повинні бути одноманітні, уніфіковані.

Недоліком шинної структури є те, що всі пристрої підключаються до кожної лінії зв'язку паралельно. Тому будь-яка несправність будь-якого пристрою може вивести з ладу всю систему. З цієї ж причини настройка системи з шинною структурою зв'язку досить складна і вимагає спеціального устаткування.

Типова структура мікропроцесорної системи наведена на рис. 1.6. Вона включає три основних типу пристроїв: процесор; пам'ять, що включає оперативну пам'ять ОЗП і постійну пам'ять ПЗП, яка служить для зберігання даних і програм; пристрої введення/виведення (ПВВ, I/O - Input/Output Devices), які призначені для зв’язку мікропроцесорної системи із зовнішніми пристроями; для приймання (введення, читання, Read) вхідних сигналів і передавання (виведення, запис, Write) вихідних сигналів.

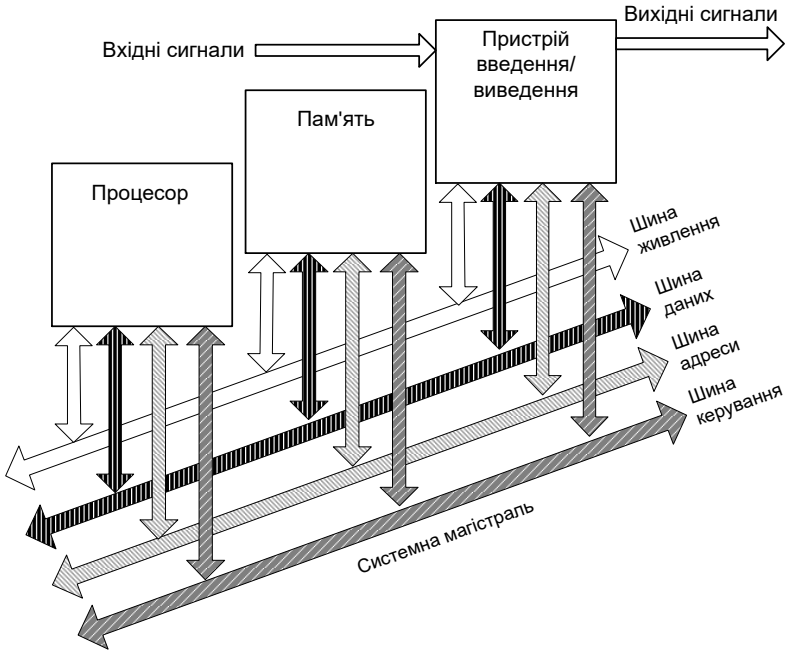

Рисунок 1.6 — Структура мікропроцесорної системи

Усі пристрої мікропроцесорної системи об'єднуються загальною системною шиною (вона ж називається ще системною магістраллю або каналом). Системна магістраль включає чотири основні шини нижнього рівня: шина адреси (Address Bus); шина даних (Data Bus); шина керування (Control Bus); шина живлення (Power Bus).

Шина адреси служить для визначення адреси пристрою, з яким процесор обмінюється інформацією в даний момент. Кожному пристрою (окрім процесора), кожному елементу пам'яті в мікропроцесорній системі задається власна адреса. Коли код якоїсь адреси виставляється процесором на шині адреси, пристрій, якому ця адреса призначена, розуміє, що його чекає обмін

інформацією.

Шина даних — це основна шина, яка використовується для передачі інформаційних кодів між всіма пристроями мікропроцесорної системи. Звичайно в пересилці інформації бере участь процесор, який передає код даних в якийсь пристрій або в елемент пам'яті чи ж приймає код даних з якогось пристрою або з елементу пам'яті. Але можлива також і передача інформації між пристроями без участі процесора. Шина даних завжди двохнаправлена.

Шина керування на відміну від шини адреси і шини даних складається з окремих сигналів керування. Кожний з цих сигналів під час обміну інформацією має свою функцію. Деякі сигнали служать для того щоб синхронізувати дані, що передаються або приймаються. Інші сигнали керування можуть використовуватися для підтвердження прийому даних, для скидання всіх пристроїв в початковий стан. Лінії шини керування можуть бути однонаправленими або двохнаправленими.

Шина живлення призначена не для пересилки інформаційних сигналів, а для живлення системи. Вона складається з ліній живлення і загального дроту. У мікропроцесорній системі може бути одне джерело живлення (частіше +5В) або декілька джерел живлення (звичайно ще -5В +12В і -12В). Кожній напрузі живлення відповідає своя лінія зв'язку. Всі пристрої підключені до цих ліній паралельно.

У мікропроцесорній системі всі інформаційні коди і коди команд передаються по шинах послідовно, по черзі. Це визначає порівняно невисоку швидкодію мікропроцесорної системи. Воно обмежене навіть не швидкодією процесора (яке теж дуже важливо) і не швидкістю обміну по системній шині, а саме послідовним характером передачі інформації по системній шині.

Лекція 2. Архітектура AVR мікроконтролерів

На рис. 1.7 наведена узагальнена структурна схема плати Arduino-UNO. Ядром Arduino є AVR-контролер, що тактується від кварцового резонатора частотою 16 МГц. Лінії портів МК виводяться назовні на контактну «гребінку» плати без яких-небудь обмежувальних або захисних елементів. Початкове скидання проводиться кнопкою SB1. На платі є 4 світлодіодних індикатора, з яких 3 службові та один («L») користувача.

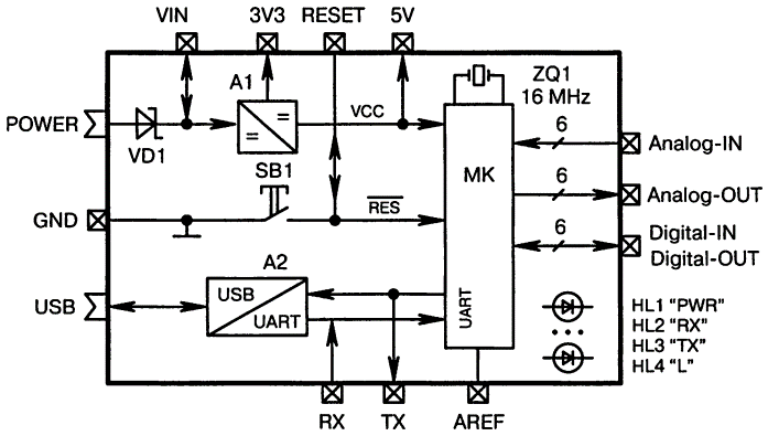

Рисунок 1.7 — Структурна схема Arduino-UNO

Зв'язок з комп'ютером здійснюється через конвертор USB-UART. Живлення на нього та на Arduino 5В надходить від комп'ютера. Також передбачено зовнішнє живлення через роз'єм POWER від «мережевої вилки» з напругою 9... 12 В. Система живлення Arduino-UNO показана на рис. 1.8.

Вхідні і вихідні сигнали МК поділяються за функціональною ознакою на такі групи: цифрові входи (IN), цифрові виходи (OUT), аналогові входи (АЦП), аналогові виходи (ШІМ). Схемотехніка підключення зовнішніх вузлів по входу та виходу буде такою ж, як і для звичайних МК.

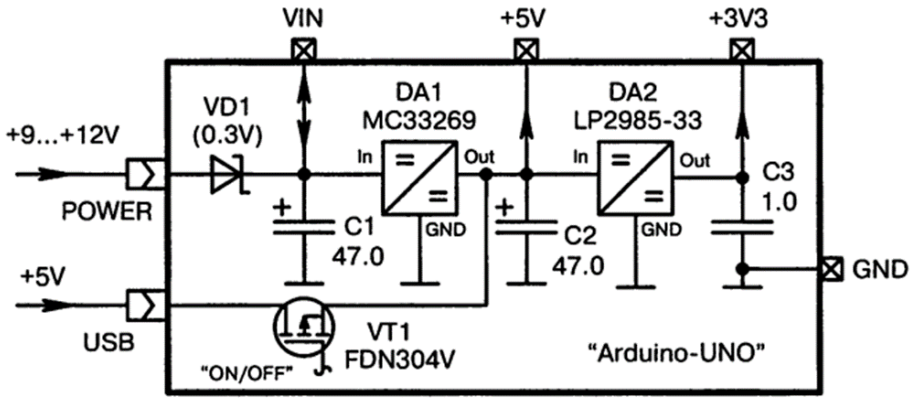

Рисунок 1.8 — Схема організації живлення в Arduino-UNO

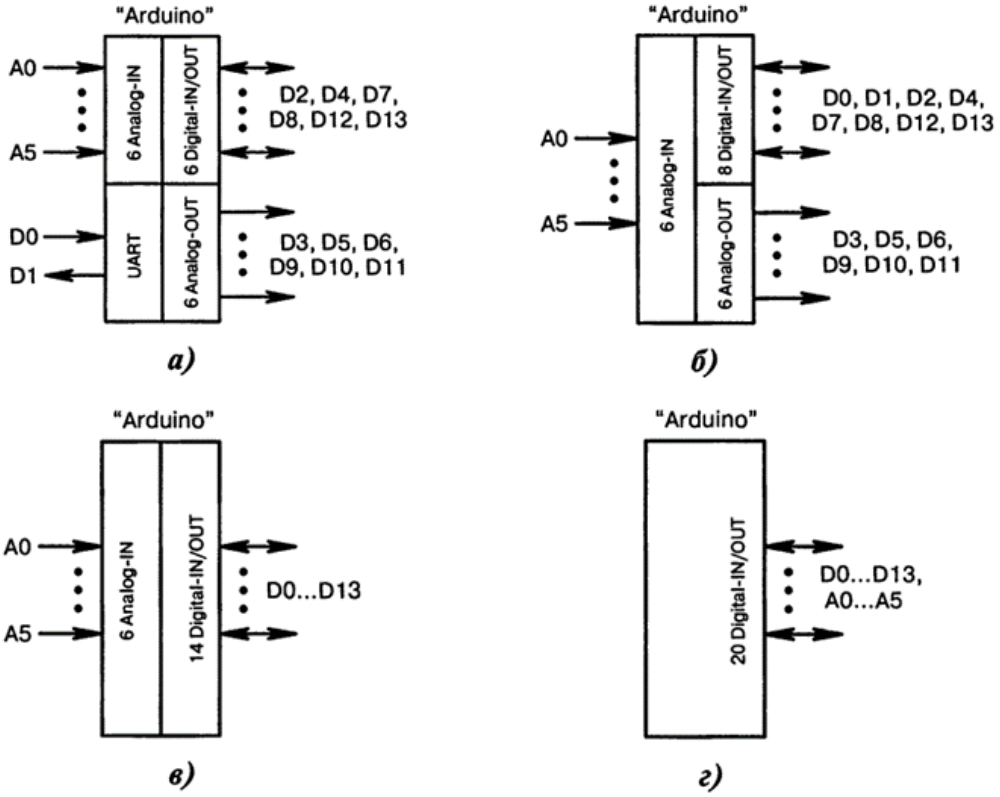

Рисунок 1.9 — Конфігурація пінів Arduino: а) повна; б) без сигналів UART: в) без аналогових виходів; г) без аналогових входів

Контакти «гребінки» 5V та 3V3 захищені від короткого замикання на загальний провід GND внутрішніми стабілізаторами DA1, DA2.

Кількість програмно доступних портів в Arduino не є константою. В залежності від налаштувань регістрів допускається різне поєднання цифрових та аналогових входів та виходів (рис. 1.9). Лінії портів Arduino прийнято називати «пінами» (pin). В Інтернеті часто застосовуються терміни «порт», «лінія». Нумерація пінів наскрізна. Цифрові піни позначаються літерою «D», аналогові

піни — літерою «А». Цифрові піни D0 та D1 відіграють особливу роль. Вони налаштовані на канал UART, який може в будь-який час необхідний для передачі даних в комп'ютер при налагодженні програми.

Розробники не встановлюють обмежень на тип МК, застосовуваний у Arduino, тому в його численних клонах використовують 8...32-бітні AVR, PIC-ARM-, Cortex-контролери. Кількість пінів (портів вводу/виводу) також може відрізнятися. Але для всіх моделей Arduino залишається незмінним принцип поділу портів на цифрові та аналогові, на входи та виходи.

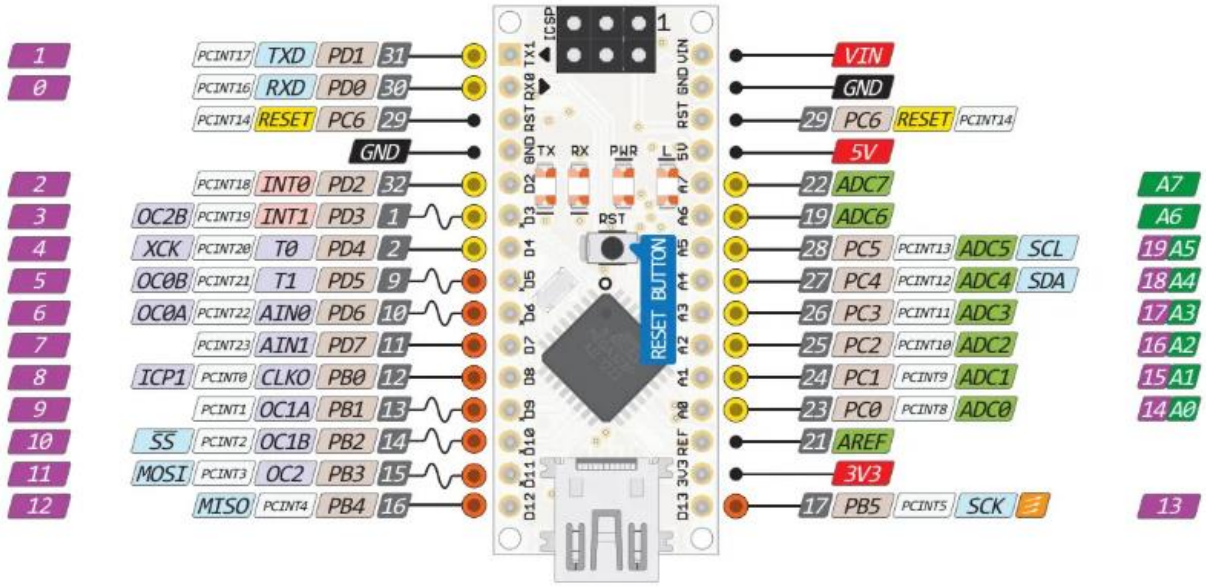

Рисунок 1.10 — Призначення виводів плати Arduino NANO

Плата Arduino NANO – це пристрій на основі мікроконтролера ATmega328 (рис. 1.10). До його складу входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів / виходів (з них 6 виводів можуть використовуватися як ШІМ-виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньо схемного програмування (ICSP) та кнопка скидання. Для початку роботи з платою необхідно подати живлення від AC / DC-адаптера, або підключити його до комп'ютера за допомогою USB-кабелю. Для роботи з платою Arduino Nano в операційній системі Windows необхідно встановити на комп'ютер інтегроване середовище розробки Arduino IDE (Integrated Development

Environment).

Входи і виходи плати Arduino NANO:

- послідовний інтерфейс: виходи 0 (RX) і 1 (TX), що використовуються для отримання (RX) та передачі (TX) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними виводами мікросхеми ATmega8U2 на платі Arduino Nano, яка виконує роль перетворювача USB-UART;
- зовнішні переривання: виводи 2 і 3, які можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих виводах.
- ШІМ-виводи 3, 5, 6, 9, 10 і 11 — інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK);
- світлодіод є вбудованимим, приєднаний до виводу 13. Рівень HIGH включає світлодіод, LOW — виключає;
- 6 аналогових входів (A0 - A5), кожен з яких може представити аналогову напругу у вигляді 10-бітного числа. За замовчуванням, вимір напруги здійснюється у діапазоні від 0 до 5 В;
- Reset: формування низького рівня (LOW) на цьому виводу призведе до перезавантаження мікроконтролера. Зазвичай цей вивід служить для функціонування кнопки скидання на платах розширення;
- 3V3: напруга на даному виводі +3.3 В, що генерується вбудованим регулятором на платі. Максимальне споживання струму 50 мА. Від цього виводу можуть живитися деякі апаратні модулі;
- 5V: напруга на даному виводу +5 В, що генерується вбудованим регулятором на платі Arduino NANO;
- GND: земля, або загальний мінус;
- Vin: використовується для подачі живлення від зовнішнього джерела в відсутності живлення 5 В від роз'єму USB, наприклад, коли необхідно запустити плату Arduino окремо від комп'ютера;
- TWI / I2C: вивід A4 або SDA та вивід A5 або SCL;
- AREF: опорна напруга для аналогово-цифрового перетворювача.

Рівень напруги на виводах обмежений 5В, максимальний струм, який може віддавати або споживати один порт, становить 40 мА. Всі виводи пов'язані з внутрішніми резисторами номіналом 20-50 кОм.

Лекція 3. Система команд і програмна модель AVR

Процес написання програм для МК AVR як і для будь-яких інших, складається з декількох етапів: підготовка вихідного тексту програми на якій-небудь мові програмування; компіляція програми; налагодження й тестування програми; остаточне програмування й підготовка до серійного виробництва.

Рисунок 1.11 – Етапи розробки програмного забезпечення мікроконтролерів AVR

Мікропрограма пристрою повинна бути написана на одній з мов програмування. На даний час для МК AVR існують декілька мов програмування, а також різних засобів підтримки розробки, що використовують одну мову, але різняться за функціональністю. На кожному з етапів необхідне застосування спеціальних програмних й апаратних засобів. Варто відзначити, що базовий

набір програмного забезпечення (компілятор асемблера, ПЗ для програмування) поширюється фірмою Atmel безкоштовно. Однак за досить довгий період часу, що пройшов з моменту появи цих МК, з'явилася велика кількість програмного забезпечення сторонніх виробників [2].

Етапи розробки програмного забезпечення мікроконтролерів AVR наведені на рисунку 1.11.

Інтегроване середовище розробки Arduino IDE – це кросплатфомовий додаток на Java, що включає в себе редактор коду, компілятор та модуль передачі прошивки в плату.

Щоб почати використовувати Arduino IDE, треба зайти на сайт https://www.arduino.cc, перейти на вкладку SOFTWARE > DOWNLOADS та завантажити середовище розробки Arduino (рис.1.12).

Рисунок 1.12 — Завантаження середовища розробки

Далі необхідно інсталювати драйвер для плати Arduino, з якою треба працювати (це робить тільки з підключеною платою через USB):

1) Підключіть плату до комп'ютера і дочекайтеся, поки Windows не почне процес установки драйверів. Незважаючи на всі зусилля системи, через кілька секунд процес завершиться невдачею.
2) Зайдіть в Пуск, відкрийте Панель керування.
3) У Панелі керування перейдіть на сторінку Система і безпека. Далі клацніть по пункту Система і відкрийте Диспетчер пристроїв.

4) Знайдіть розділ Порти (COM & LPT). У ньому ви побачите відкритий порт під ім'ям «Arduino UNO (COMxx)».
5) Клацніть правою кнопкою по пункту «Arduino UNO (COMxx)» і виберіть «Оновити драйвер».
6) Далі, у вікні, виберіть пункт «Виконати пошук драйверів на цьому комп'ютері»
7) На завершення, виберіть файл драйвера під ім'ям «*arduino.inf*», розташований в папці «Drivers» в директорії завантаженого програмного забезпечення Arduino. Windows завершить інсталяцію драйвера.
8) Відкрити середовище розробки Arduino та запустити тестову програму File > Examples > 1.Basics > Blink.
9) Тепер в меню Tools>Board необхідно вибрати пункт меню, що відповідає вашій моделі Arduino.
10) У меню Tools>Serial Port виберіть послідовний порт, до якого підключений лабораторний макет. Як правило, це COM-порт з номером 3 (COM3) або вище (COM1 і COM2 зазвичай асоційовані з апаратними портами). Тепер можна працювати та завантажувати скетчі в Arduino.

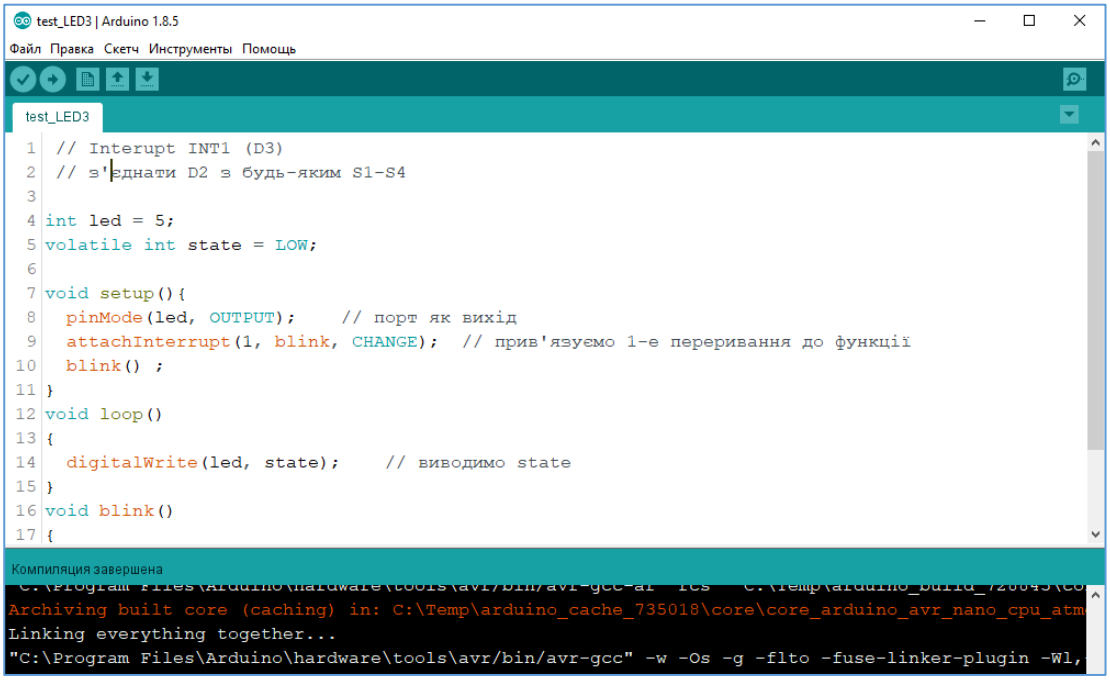

Рисунок 1.13 – Загальний вигляд програми Arduino IDE

Середовище розробки Arduino (рис.1.13) складається із вбудованого текстового редактора програмного коду, області повідомлень, вікна виведення тексту (консолі), панелі інструментів з кнопками часто використовуваних команд і декількох меню. Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини Arduino.

Програма, написана в середовищі Arduino, називається *скетч*. Скетч пишеться в текстовому редакторі, що має інструменти: вирізати / вставити, пошук / заміна тексту. Під час збереження та експорту проекту в області повідомлень з'являються пояснення, також можуть відображатися виникли помилки. Вікно виведення тексту (консоль) показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дозволяють перевірити та записати програму, створити, відкрити та зберегти скетч, відкрити моніторинг послідовної шини.

Блокнот (Sketchbook)

Середовищем Arduino використовується принцип блокнота: стандартне місце для зберігання програм (скетчів). Скетчі з блокнота відкриваються через меню File → Sketchbook або кнопкою Open на панелі інструментів. При першому запуску програми Arduino автоматично створюється директорія для блокнота. Розташування блокнота змінюється через діалогове вікно Preferences.

Закладки, Файли і Компіляція

Дозволяють працювати з декількома файлами скетчів (кожен відкривається в окремій закладці). Файли коду можуть бути стандартними Arduino (без розширення), файлами C (розширення * .c.), файлами C ++ (*.cpp) або файлами заголовків (.h).

Завантаження скетчу в Arduino

Після вибору порту та платформи необхідно натиснути кнопку завантаження на панелі інструментів або вибрати пункт меню File → Upload to I/O Board. Сучасні платформи Arduino перезавантажуються автоматично перед завантаженням. На більшості плат під час процесу будуть мигати

світлодіоди RX і TX. Середовище розробки Arduino виведе повідомлення про закінчення завантаження або про помилки.

При завантаженні скетчу використовується завантажувач (Bootloader) Arduino, невелика програма, що завантажується в мікроконтролер на платі. Вона дозволяє завантажувати програмний код без використання додаткових апаратних засобів. Завантажувач (Bootloader) активний протягом декількох секунд при перезавантаженні платформи і при завантаженні будь-якого з скетчів в мікроконтролер. Робота завантажувача (Bootloader) розпізнається по миготінню світлодіода (13 вивід) (наприклад, при перезавантаженні плати).

Експорт бінарного файлу

Для роботи з віртуальним стендом «Arduino Learner Kit» у середовищі Proteus 8 необхідно підключити файл типу .HEX (рис. 1.14). Для його отримання потрібно вибрати пункт меню Sketch →Export Compiled Binary. HEX файл буде створений у директорії зі скетчим.

Бібліотеки

Бібліотеки додають додаткову функціональність скетчам, наприклад, при роботі з апаратною частиною або при обробці даних. Для використання бібліотеки необхідно вибрати меню Sketch → Include Library. Можна вибрати бібліотеки зі списку або завантажити через Library Manager (рис.1.14).

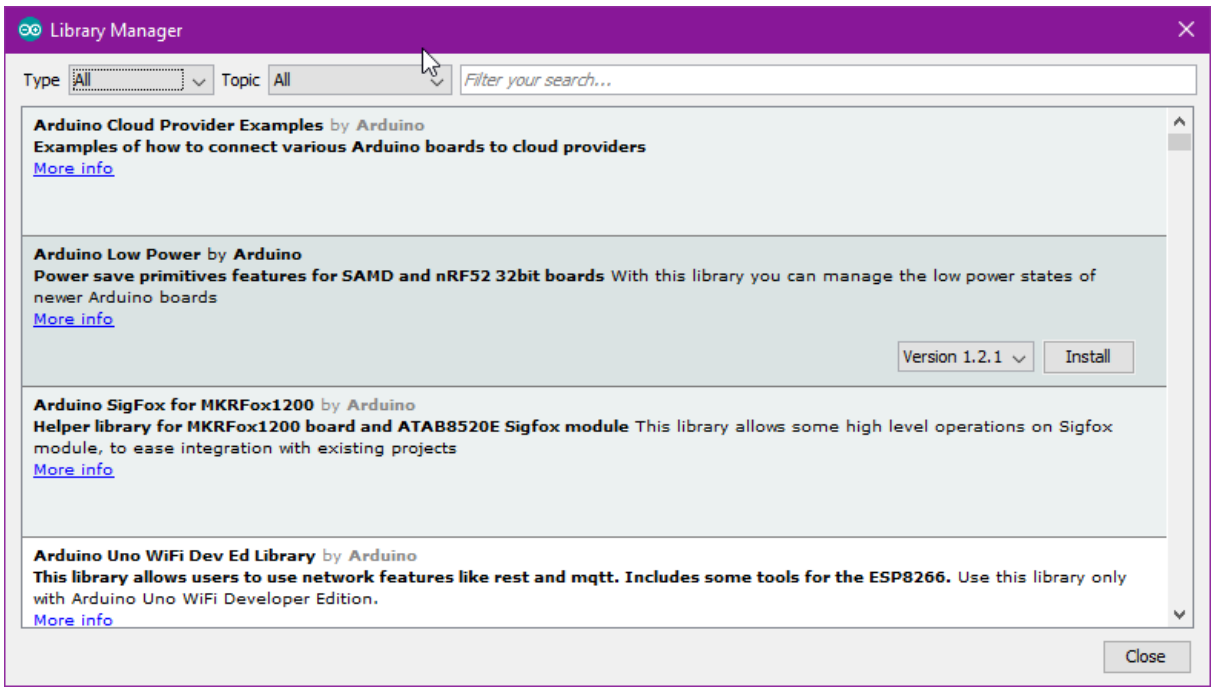

Рисунок 1.14 — Менеджер завантаження бібліотек

Одна або кілька директив *#include* будуть розміщені на початку коду скетчу з подальшою компіляцією бібліотек разом зі скетчем. Завантаження бібліотек вимагає додаткового місця в пам'яті Arduino. Невикористані бібліотеки можна видалити з скетчу прибравши директиву *#include*.

Основні бібліотеки: EEPROM, SD, SPI, SoftwareSerial, Wire. Деякі бібліотеки включені в середовище розробки Arduino. Інші можуть бути завантажені з різних ресурсів. Для встановлення додаткових бібліотек необхідно створити директорію «libraries» у папці блокнота та потім розпакувати архів. Наприклад, для встановлення бібліотеки DateTime її файли повинні знаходиться в папці / libraries / DateTime папки блокнота.

Лекція 4. Програмування в машинних кодах

Середовище розробки засновано на мові програмування Processing і спроектовано для програмування новачками, які не знайомі близько з розробкою програмного забезпечення. Строго кажучи, це C/C++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC.

Мова Arduino має чотири складових: оператори, дані, функції, бібліотеки.

Оператори:

- setup (),
- loop (),
- оператори мови С.

Дані: типи даних з мови С.

Функції:

- цифрове введення / виведення,
- аналогове введення / виведення,
- час,
- математичні обчислення,
- тригонометрія,
- випадкові числа,
- біти і байти,
- зовнішні переривання,
- переривання.

Бібліотеки:

- EEPROM,
- SD,
- SPI,
- SoftwareSerial,
- Wire,

- допоміжні класи,
- клас Serial,
- клас Stream.

Оголошення змінної відбувається так: спочатку вказується тип даних для змінної, а потім назва змінної. Оператор присвоювання (=) — не є знаком рівності та не може використовуватися для порівняння значень. Оператор рівності записується так — ==. Присвоєння використовується для збереження певного значення в змінній. Наприклад, запис виду *a = 10* задає змінній *a* значення числа *10*.

Якщо знаємо точну кількість дій (ітерацій) циклу, то можемо використовувати цикл *for*. Синтаксис його виглядає так:

```
for (дія до початку циклу; умова продовження циклу; дія в кінці кожної ітерації циклу)
    {
        інструкція циклу 1;
        інструкція циклу 2;
        інструкція циклу N;
    }
```

Ітерацією циклу називається один прохід цього циклу.

Коли не відомо скільки ітерацій повинен зробити цикл, тоді використовується цикл *while* або *do ... while*. Синтаксис циклу *while* виглядає так:

```
while (Умова){
    Тіло циклу;}
```

Даний цикл буде виконуватися, поки умова, вказана в круглих дужках є істиною.

Оператор if служить для того, щоб виконати будь-яку операцію в тому випадку, коли умова є вірною. Умовна конструкція завжди записується в круглих дужках після оператора *if*. Усередині фігурних дужок вказується тіло умови. Якщо умова виконається, то почнеться виконання всіх команд, які знаходяться між фігурними дужками.

Оператор else. Кожному оператору *if* відповідає тільки один оператор

else. Сукупність цих операторів — *else if* означає, що якщо не виконалася попередня умова, то перевірити дану. Якщо жодна з умов не є вірною, то виконується тіло оператора *else*.

Оператори порівняння

== (дорівнює);

! = (не дорівнює);

< (менше ніж);

> (більше ніж);

<= (менше або дорівнює);

> = (більше або дорівнює).

Логічні оператори

&& (І);

|| (АБО);

! (НЕ);& (побітове І);

| (побітове АБО).

Бітові оператори

^ (побітове XOR або виключне АБО);

~ (побітове НЕ);

<< (побітовий зсув вліво);

>> (побітовий зсув вправо).

Складні оператори++ (інкремент)

– – (декремент);

+ = (складене додавання);

– = (складене віднімання);

* = (складене множення);

/ = (складене ділення);

& = (складене побітове І);

| = (складене побітове АБО).

Будь-яка функція має тип, як і будь-яка змінна. Функція може повертати значення, тип якого аналогічний типу самої функції. Якщо

функція не повертає ніякого значення, то вона повинна мати тип *void* (такі функції іноді називають процедурами).

При оголошенні функції, після її типу має стояти ім'я функції і дві круглі дужки — відкриваюча і закриваюча, всередині яких можуть знаходитися один або кілька аргументів функції, яких також може не бути взагалі. Після списку аргументів функції ставиться відкриваюча фігурна дужка, після якої знаходиться саме тіло функції. В кінці тіла функції обов'язково ставиться фігурна дужка, що закриває його.

Під час виклику функції *setup(),* програма ініціалізується та встановлює початкові значення. Функція *setup()* викликається, коли стартує скетч. Використовується для ініціалізації змінних, визначення режимів роботи виводів, запуску використовуваних бібліотек. Функція *setup()* запускається тільки один раз, після кожної подачі живлення або скидання плати Arduino.

Функція *loop ()* забезпечує нескінченний робочий цикл програми. У циклі виконується опитування стану виводів, зміна їх стану, прийом-передача даних робота з АЦП та ін.

Приклад:

```
    int buttonPin = 3;
void setup()
    {
    // put your setup code here, to run once:
    }
    void loop()
    {
    // ...
    }
```

Типи даних

void — ключове слово void використовується тільки при оголошенні функцій. Воно вказує на те, що оголошувана функція не повертає ніякого значення.

boolean — змінні типу boolean можуть приймати одне з двох значень: true або false. Кожна змінна типу boolean займає в пам'яті один байт.

char — тип даних, який займає в пам'яті 1 байт та зберігає символьне

значення. Символи пишуться в одинарних лапках, наприклад: 'A' (сукупність символів (рядки) пишуться у подвійних лапках: "ABC").

int — цілочисельний тип даних. Це основний тип даних для зберігання чисел. В Arduino Uno змінні типу int зберігають 16-бітові (2-байтові) значення у діапазоні від –32768 до 32767.

unsigned int — беззнакові цілі містять двобайтові значення. Замість негативних чисел зберігаются лише позитивні значення в діапазоні від 0 до 65535.

long — змінні типу long маю розширений розмір для зберігання чисел та мають розмірність 32 біта (4 байта), що дозволяє їм зберігати числа в діапазоні від –2 147 483 648 до 2 147 483 647.

unsigned long — мають розмірність 32 біта (4 байта). Змінні типу *unsigned long*, на відміну від звичайного long, зберігають тільки позитивні числа в діапазоні від 0 до 4 294 967 295.

short — це 16-бітний тип даних.

float — тип даних для чисел з плаваючою точкою. Числа з плаваючою точкою часто використовуються для подання аналогових або безперервних величин, оскільки дають можливість окреслити їх більш точно, ніж цілі числа. Числа з плаваючою точкою мають 32 біта (4 байта) інформації та можуть досягати величезних значень від —3.4028235E + 38 до 3.4028235E + 38.

Точність дрібних чисел типу *float* становить 6-7 десяткових знаків. Мається на увазі загальна кількість цифр, а не кількість знаків після коми.

double — займають 4 байта. Аналогічні змінним float.

Створення (оголошення) *масиву*. Масив — це область пам'яті, де можуть послідовно зберігатися кілька значень.

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

string — текстовий рядок. Може бути оголошений двома способами: можна використовувати тип даних *string* або оголосити рядок як *масив*

символів char з нульовим символом в кінці.

```
char Str1 [15];
char Str2 [8] = { 'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3 [8] = { 'a', 'r', 'd', 'u', 'i', 'n', 'o', '\ 0'};
char Str4 [] = "arduino";
char Str5 [8] = "arduino";
char Str6 [15] = "arduino".
```

Рядки завжди оголошуються в подвійних лапках ("Abc"), а символи завжди оголошуються в одинарних лапках ('A').

Лекція 5. Порти введення/виведення AVR. Програмне введення/виведення інформації

За замовчуванням усі порти Arduino визначаються як входи, і немає потреби описувати це в коді. Порти зазвичай прописуються в функції ініціалізації змінних.

Ініціалізація порту вводу-виводу Arduino:

pinMode **(pin, mode).** Параметри:

– pin: номер виводу, режим роботи якого задається;

– mode: приймає значення: INPUT — вхід, у цьому режимі відбувається зчитування даних з датчиків, стану кнопок, аналогового та цифрового сигналу. Порт знаходиться в так званому високоімпедансному стані, тобто на вході високий опір. OUTPUT — вихід, залежно від команди прописаної в коді, порт приймає значення одиниці або нуля. Вихід стає свого роду керованим джерелом живлення та видає максимальний струм (20 мА та 40 мА в піковому значенні) у навантаження, що до нього підключене. INPUT_PULLUP — порт працює як вхід, але до нього підключається «PushUp» резистор з номіналом 20 – 50 кОм;

– значення, що повертаються – немає.

digitalWrite **(pin, value).** Параметри:

– pin: номер виводу;

– value: значення HIGH або LOW;

– значення, що повертаються — немає.

digitalRead **(pin).** Параметри:

– pin: номер цифрового виводу, з якого необхідно зчитати значення (int);

– значення, що повертаються HIGH або LOW.

Регістри портів дозволяють низько рівневі високошвидкісні маніпуляції з портами мікроконтролера. Мікроконтролери, що використовуються в Arduino мають три порти : B (D8-D13), C(A0-A7), D(D0-D7) (рис. 1.10).

Кожен порт контролюється трьома регістрами, кожен з яких відповідає за певний стан. Регістр DDR визначає, який біт порта вхідний, а який вихідний. Регістр PORT встановлює біт порта у відповідний стан HIGH або LOW, регістр PIN читає стан вхідного порта.

Регістри DDR та PORT можуть бути як прочитані, так і записані. Регістр PIN відповідає за стан вхідних портів, тому може бути лише прочитаний.

PORTD відповідає за виводи 0 - 7.

DDRD — регістр напряму порту D;

PORTD — регістр даних порту D;

PIND — регістр вхідних даних порту D.

PORTB відповідає за виводи 8 - 13. Два старших біта (6 та 7), що відповідають за виводи кварцу, не використовуються.

DDRB — регістр напряму порту B;

PORTB — регістр даних порту B;

PINB — регістр вхідних даних порту B.

PORTC відповідає за аналогові виводи 0 - 5.

DDRC — регістр напряму порту C;

PORTC — регістр даних порту C;

PINC — регістр вхідних даних порту C.

Кожен біт в цих регістрах відповідає за відповідний вивід, так молодший біт у DDRB, PORTB, та PINB посилається на вивід PB0 (цифровий порт D8) (рис. 1.10). Слід пам'ятати, що виводи D0 та D1 задіяні послідовним портом та робота з ними можлива тільки в тому випадку, якщо налагодження та послідовний порт не потрібні. Приклад роботи з портом D:

```
// призначаємо виводи Arduino 1-7 вихідними, вивід 0-вхідним
DDRD = B11111110;
// виводи з 2 по 7 вихідні, стан виводів 0 та 1 не змінюється
DDRD = DDRD | B11111100;
// встановлюємо рівень HIGH на цифрових виводах 7,5,3
PORTD = B10101000;
```

Лекція 6. Таймери/лічильники. Модуль переривань

У ATMega328 передбачені три таймери/лічильники, на яких реалізовано функції часу, які використовують для формування та вимірювання часових інтервалів.

Основні функції часу:

delay (ms) Параметри:

- ms — кількість мілісекунд, на які необхідно призупинити програму;
- значення, що повертаються — немає;
- опис: припиняє виконання програми на вказаний проміжок часу (в мілісекундах).

delayMicroseconds (us). Параметри:

- us - кількість мікросекунд, на які необхідно призупинити програму;
- значення, що повертаються – немаєопис: припиняє виконання програми на вказаний проміжок часу (в мікросекундах). Найбільше число для формування затримки — 16383.

millis ():

- параметри — немає;
- значення, що повертаються — кількість мілісекунд, що пройшли з моменту старту програми;
- опис: повертає кількість мілісекунд, що пройшли з моменту старту програми Arduino. Число, що повертається, скинеться в 0) через приблизно 50 днів.

micros():

- параметри — не має;
- значення, що повертаються — кількість мікросекунд, що минули з моменту старту програми;
- опис: повертає кількість мікросекунд, що минули з моменту початку виконання програми Arduino. Число, що повертається, скинеться в 0 через приблизно 70 хвилин. Роздільна здатність цієї функції становить

чотири мікросекунди.

pulseIn (pin, value) / pulseIn (pin, value, timeout). Параметри:

- pin : номер виводу, на якому буде очікуватися сигнал;
- value: тип імпульсу (HIGH або LOW);
- timeout (опціонально): час очікування імпульсу в мікросекундах (значення за замовчуванням - одна секунда);
- значення, що повертаються — тривалість імпульсу (в мікросекундах) або 0 в разі відсутності імпульсу протягом таймаута;
- опис: зчитує тривалість імпульсу (будь-якого – HIGH або LOW) на виведення. Наприклад, якщо задане значення value = HIGH, то функція *pulseIn ()* очікує появи на виведення сигналу HIGH, потім вимірює час та чекає перемикання в стан LOW, після чого зупиняє відлік часу. Функція повертає тривалість імпульсу в мікросекундах, або 0 в разі відсутності імпульсу протягом певного часу очікування. Функція працює з імпульсами тривалістю від 10 мікросекунд до 3 хвилин.

Переривання – це сигнали, що переривають нормальний перебіг програми. Вони використовуються для апаратних пристроїв, що вимагають негайної реакції на появу подій. Обробка переривань у мікроконтролері відбувається за допомогою модуля переривань, який приймає запити переривання й організовує перехід до виконання визначеної програми. Запити переривання можуть надходити як від зовнішніх джерел, так і від джерел, розташованих у різних внутрішніх модулях мікроконтролера.

Як входи для прийому запитів від зовнішніх джерел, найчастіше використовуються виводи паралельних портів вводу/виводу, для яких ця функція є альтернативною. Джерелами запитів зовнішніх переривань також можуть бути будь-які зміни зовнішніх сигналів на деяких спеціально виділених лініях портів вводу/виводу.

Arduino надає свої функції для роботи з зовнішніми перериваннями. Ці функції оголошені у файлі:\ *Hardware \ cores \ arduino \ wiring.h* та реалізовані в файлі: \ *Hardware \ cores \ arduino \ WInterrupts.c*

Функції переривання Arduino:

attachInterrupt (interrupt, function, mode). Параметри:

- interrupt: номер переривання (0 – pin D2, 1 – pin D3);
- function: функція, яку необхідно викликати при виникненні переривання; ця функція повинна бути без параметрів і не повертати ніяких значень (таку функцію іноді називають оброблювачем переривання);
- mode: визначає умову, за якої має спрацьовувати переривання. Може приймати одне з чотирьох визначених значень: LOW — переривання буде спрацьовувати щоразу, коли на виводі присутній низький рівень сигналу, CHANGE — переривання буде спрацьовувати щоразу, коли змінюється стан виводу, RISING — переривання спрацює, коли стан виводу зміниться з низького рівня на високий, FALLING — переривання спрацює, коли стан виводу зміниться з високого рівня на низький;
- значення, що повертаються – немає.

detachInterrupt (pin):

- параметри — немає;
- значення, що повертаються — немає;
- опис: забороняє задане переривання. Забороняє переривання, для того, щоб відключити доступ до даних в процесі виконання переривання.

interrupts ():

- параметри — немає;
- значення, що повертаються — немає;
- опис: повторно дозволяє переривання.

Таймери, як і зовнішні переривання, працюють незалежно від основної програми. У стандартних платах Arduino є три таймера Timer0, Timer1 і Timer2. Timer0 є 8 бітним таймером, це означає, що його рахунковий регістр може зберігати числа до 255. Timer0 використовується стандартними часовими функціями Arduino такими як delay() і millis(), так що краще його не використовувати у своїх проектах.

Timer1 це 16 бітний таймер з максимальним значенням 65535. Timer2 —

8 бітний і дуже схожий на Timer0. Він використовується в функції tone () Arduino.

Для обробки переривань у мові програмування Arduino використовується функція ISR ().

Лекція 7. Мікроконтролери Arduino та ESP8266

Будь-який розробник мікропроцесорних систем управління використовує мікроконтролери Arduino (UNO, Micro, Nano, Mini, Mega), ESP8266 (WeMos D1, WeMos D1 mini NodeMCU), AirBoard, ChipKIT (UNO32, DP32, uC32, Max32) [9, 10]. Для зв'язку між контролером та мобільним пристроєм можна скористатися Bluetooth HC-05, HC-06, WiFi ESP8266, Ethernet Shield W5100. Для створення пристрою з віддаленим керуванням пропонуються кілька безкоштовних систем розробки та використання мобільних графічних інтерфейсів для управління контролерами зі смартфона або планшета.

Основним елементом керування є модуль з мікроконтролером, який здійснює приймання та обробку команд по каналу Wi-Fi і керує виконавчим пристроєм.

Блок керування можна реалізувати такими способами. Перший спосіб зробити зв’язку на Arduino Uno та Wi-Fi шилд на базі HDG04 [9, 10]. Апаратне зв’язування даних модулів відбувається накладанням Wi-Fi шилда на Arduino Uno, у вигляді «бутерброда».

Wi-Fi шилд побудований на базі модуля HDG104, представляє собою систему на кристалі, яка забезпечує підключення Arduino до мережі Інтернет по безпровідному інтерфейсу LAN 802.11b/g (Wi-Fi). Мікроконтролер ATmega 32UC3 підтримує стек мережевих протоколів (IP) та дозволяє працювати як з TCP, так і з UDP-протоколами. Дана збірка має сильні недоліки, а саме: велика ціна збірки Arduino Uno + Wi-Fi шилд.

Другий спосіб зробити зв’язку з Arduino Pro Mini nf ESP-01(ESP-12E). В даній зв’язці в ролі мікроконтролера виступає Arduino Pro Mini, а в ролі пристрою передачі через інтернет виступає ESP-01(ESP8266). Оскільки на виході мікроконтролера рівень виходів дорівнює 5В, в пристрої перетворення рівнів немає необхідності.

Третій спосіб використати модуль Node MCU V3, який виступає у ролі пристрою передачі даних через Інтернет і мікроконтролером керування. Технічні

характеристики модуля Node MCU V3 наведені в [10].

Node MCU V3 [10] – це модуль, на якому розташований чіп ESP 8266, який містить Wi-Fi мікросхему і мікроконтролер. Завдяки цьому можна створювати мікропроцесорні системи інтернету речей, що з'єднуються між собою через Wi-Fi з'єднання. Характерними особливостями цього модуля є частота мікроконтролера, яка складає 160МГц, велика кількість портів введення/виведення мале енергоспоживання і компактні розміри. Під даний модуль можна писати програму на мові програмування Processing та JavaScript.

Wi-Fi модуль розроблений компанією Ai-thinker і побудований на базі процесора з ядром ESP8266, відмінною рисою якого є наявність радіоінтерфейсу Wi-Fi. Ядро ESP8266 інтегровано в Tensilica L106 - 32-бітний мікроконтролер з ультранизьким енергоспоживанням. Підтримка тактових частот 80 і 160 МГц, підтримка RTOS, вбудовані Wi-Fi MAC / BB / RF / PA / LNA. Флеш-пам'ять, інтегрована в модуль - це SPI флеш-пам'ять, ємність якої становить 4 Мбайта, в корпусі SOP-210mil. Антена, що застосовується в модулі, мікросмугова антена на платі з коефіцієнтом посилення 3 дБ. Однокристальна Wi-Fi система ESP8266EX вбудовується разом з контролером пам'яті, включаючи SRAM і ROM. MCU може звертатися до пам'яті через інтерфейси iBus, dBus і AHB. Розмір RAM <36 Кбайт, тобто, коли ESP8266EX працює в режимі клієнтської станції і підключений до роутера, програмований простір, доступний користувачеві разом з секцією Data, становить близько 36 Кбайт. В однокристальній системі немає програмованої пам'яті ROM; призначена для користувача програма повинна зберігатися у зовнішньому SPI флеш-пам'яті.

Модуль містить 11 портів загального призначення. Деякі з портів мають додаткові функції: D9, D10 – UART, D1, D2 – I²C/TWI, D5..D8 – SPI, D1..D10 — виходи з ШІМ (PWM), A0 – аналоговий вихід з АЦП.

Лекція 8. Аналого-цифрові перетворювачі. Цифро-аналогові перетворювачі

Для передавання аналогового сигналу до МПС використовують аналогово-цифровий перетворювач (АЦП). АЦП сприймає аналоговий сигнал, напругу або струм і перетворює його в цифрове слово, зрозуміле МП. На рис. 1.15 наведена структурна схема модуля (АЦП), що застосовується в МК AVR.

Перетворення «аналог-цифра» здійснюється в 10-бітовому АЦП послідовного наближення. Для його нормальної роботи потрібно три сигнали: вхідний V_{IN}, тактовий $F_{АЦП}$, опорний V_{REF}.

Сигнал V_{IN} поступає від мультиплексора, що комутує вісім аналогових каналів з ліній PA0...PA7 та дві тестових напруги 0 і +1,22В. Вибір джерела сигналу здійснюється програмним способом через регістр ADMUX. Сигнал $F_{АЦП}$ виходить з тактового сигналу F_{CLK} шляхом ділення на коефіцієнт 2...128, що програмно задається регістром ADCSRA. Усередині блоку АЦП частота $F_{АЦП}$ ділиться ще раз на 13 або 14 залежно від одноразового або безперервного режиму вимірів. Це і буде істинним значенням частоти дискретизація сигналу Fд.

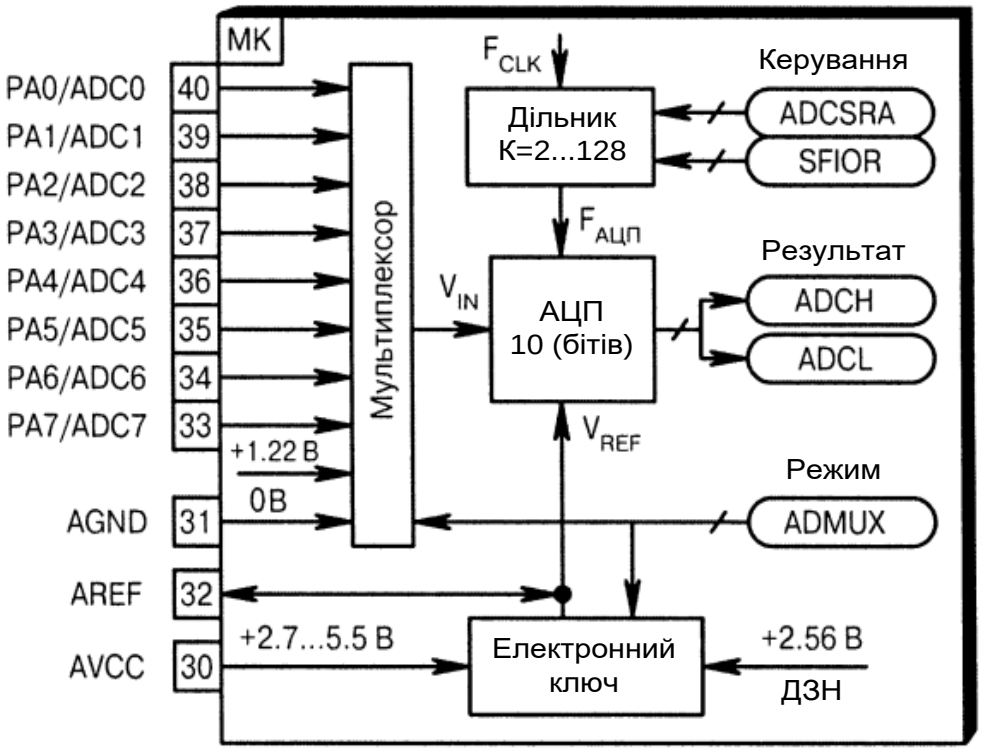

Рисунок 1.15 – Структурна схема модуля АЦП в мікроконтролерах AVR

Припустимо, що F_{CLK} = 8 МГц, коефіцієнт ділення встановлений 16, відповідно, $F_{АЦП}$ = 500 кГц, Fд = 38,5 кГц (13тактів) або Fд= 35,7 кГц (14тактів).

Сигнал V_{REF} може поступати з трьох напрямів: від вхідної лінії AREF, від внутрішнього джерела зразкової напруги +2,56 В, від джерела живлення AVCC. Перемикання здійснюється електронним ключем, який керується регістром ADMUX. Вивід AREF має безпосередній електричний зв'язок з модулем АЦП, тому, для зменшення наведень, його шунтують керамічним конденсатором ємністю 0,1 мкФ.

Результат кожного вимірювання поміщається в регістри ADCH (старші 2 біта) і ADCL (молодші 8 бітів). Разом в двох регістрах утворюється число в діапазоні 0... 1023. Ціна одного ділення – V_{REF}/1024. Це справедливо для «чистого» режиму 10 біт. У ATmega328 є ще один режим, що умовно називається 8/10 біт. У ньому вимірювання проводиться з точністю 10 бітів, але відображаються усього лише 8 старших бітів, тобто діапазон складає 0...255, ціна одного ділення V_{REF}/256.

При роботі АЦП потрібно на початку програми відключати вхідні «pull-up» резистори, оскільки лінії ADC0...ADC7 за сумісництвом ще є цифровими входами PA0...PA7.

Аналого-цифровий перетворювач дозволяє зчитати величину напруги на аналогових входах. Це дає можливість зчитувати дані з датчика освітленості, виміряти напругу живлення і т.д. Основні команди для роботи з АЦП в Arduino:

analogReference **(type).** Параметри:

- type: тип джерела опорної напруги (DEFAULT, INTERNAL, EXTERNAL);
- значення, що повертаються — немає;
- опис: встановлює джерело опорної напруги, що використовується при зчитуванні аналогового сигналу (задає максимальне значення вхідного діапазону). DEFAULT: опорна напруга за замовчуванням, рівна 5В.

INTERNAL: внутрішня опорна напруга рівна 1,1В. EXTERNAL: як опорна напруга буде використовуватися напруга, прикладена до виводу AREF (від 0 до 5В).

analogRead (pin). Параметри:

- pin: номер виводу, з якого буде зчитуватися напруга (A0 - A5 для більшості плат, A0 - A7 для Mini та Nano, A0 - A15 для Mega);
- значення, що повертаються: ціле число int (від 0 до 1023);
- опис: зчитує величину напруги з зазначеного аналогового виводу. АЦП перетворювач перетворює вхідну напругу з діапазону 0 - 5В в цілочисельні значення в межах від 0 до 1023 відповідно. Роздільна здатність АЦП становить: 5 В / 1024 значення або 0,0049 В (4.9 мВ) на одне значення. Вхідний діапазон та роздільна здатність можуть змінюватися за допомогою функції *analogReference ()*. Для зчитування значення з аналогового входу потрібно близько 100 мікросекунд, тому максимальна частота опитування виводу приблизно дорівнює 10 000 разів в секунду. Якщо аналоговий вхід ні до чого не підключений, значення, що повертається функцією *analogRead (),* буде випадковим (змінюється під впливом декількох факторів: величина напруги на інших аналогових входах, наведення від руки поблизу плати).

Цифроаналоговий перетворювач (ЦАП) призначений для перетворення числа, визначеного, як правило, у вигляді двійкових кодів, у напругу або струм пропорційно значенню цифрового коду.

Дуже часто ЦАП входить до складу МПС. У цьому випадку, якщо не потрібна висока швидкодія, цифроаналогове перетворення може бути дуже просто здійснено за допомогою широтно-імпульсної модуляції (ШІМ). Схема ЦАП з ШІМ наведена на рис.1.16.

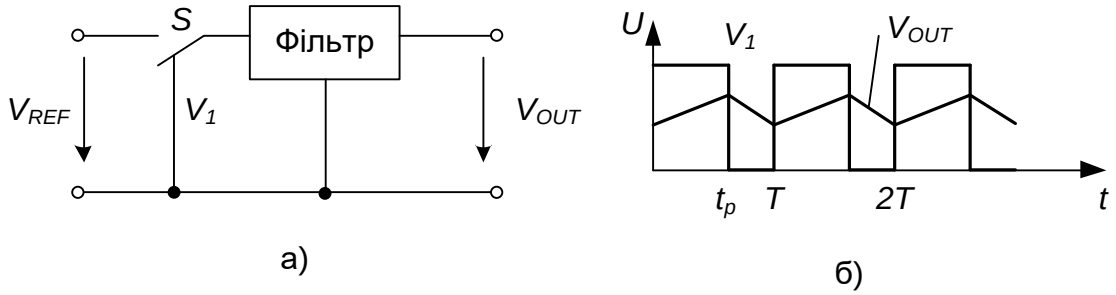

Рисунок 1.16 – ЦАП з широтно-імпульсною модуляцією:
а) структурна схема; б) часова діаграма

Найпростіше організовується цифроаналогове перетворення в тому випадку, якщо МК має вбудовану функцію широко-імпульсного перетворення. Вихід ШІМ керує ключем *S*. У залежності від заданої розрядності перетворення контролер за допомогою власного таймера/лічильника формує послідовність імпульсів, відносна тривалість яких $\gamma = t_p / T$ визначається співвідношенням:

$$\gamma = \frac{D}{2^N},$$

де N – розрядність перетворення, а D – код перетворення.

Фільтр нижніх частот згладжує імпульси, виділяє середнє значення напруги. У результаті вихідна напруга перетворювача:

$$V_{OUT} = \gamma V_{REF} = \frac{DV_{REF}}{2^N}$$

Розглянута схема забезпечує ідеальну лінійність перетворення, не містить прецизійних елементів (за винятком джерела опорної напруги). Основний її недолік – низька швидкодія.

Для формування сигналів ШІМ використовується команда ***analogWrite (pin, value)*** з параметрами:

- pin: вивід, на якому буде формуватися напруга широтно-імпульсної модуляції (ШІМ або PWM);
- value: коефіцієнт заповнення — лежить в межах від 0 (завжди вимкнений) до 255 (завжди включений);

– значення, що повертаються: немає;
– опис: формує задану аналогову напругу на виводі у вигляді ШІМ-сигналу. Може використовуватися для зміни яскравості світіння світлодіода або управління швидкістю обертання двигуна. Після виклику *analogWrite(),* на виводі буде безперервно генеруватися ШІМ-сигнал із заданим коефіцієнтом заповнення до наступного виклику функції *analogWrite()* (або до моменту виклику *digitalRead(),* або *digitalWrite(),* взаємодіючих з цим же виводом). Частота ШІМ становить приблизно 490 Гц. На більшості плат Arduino функція *analogWrite ()* працює з виводами 3, 5, 6, 9, 10 і 11. Функція *analogWrite ()* не має нічого спільного з аналоговими виводами і функцією *analogRead ().*

Лекція 9. Особливості живлення та формування тактової частоти

Для живлення будь-якого МК потрібно, як мінімум, два виводи: позитивний («плюс», «Power supply») і негативний («мінус», «Ground reference»). Позначають їх в Data-sheet і на схемах: Vcc (Voltage Collector - to - Collector) або V_{DD} (Voltage Drain - to - Drain); GND (Ground) або V_{SS} (Voltage Source - to - Source).

Двопровідне живлення застосовується в малогабаритних МК з числом виводів 6... 18, наприклад, в Atmel ATtiny, Microchip PIC10/12. З розвитком технології до складу МК стали вводити аналогові вузли АЦП/ЦАП, які дуже чутливі до завад. Додавання ланок AVCC (Analog VCC) і AGND (Analog GND) дозволяє розв'язати між собою аналогові і цифрові частини мікросхеми, зменшити імпульсні завади, підвищити інструментальну точність каналів АЦП і ЦАП.

Якщо подивитися на осцилограму струму споживання МК, то в ній можна помітити низькочастотну (НЧ) і високочастотну (ВЧ) складові. Як наслідок, коливання струму призводять до появи НЧ- і ВЧ - завад на клемах живлення. Для їх послаблення використовують стандартні рішення у вигляді зв'язки конденсаторів (рис. 2.1), LC - і RC - фільтрів (рис. 2.2).

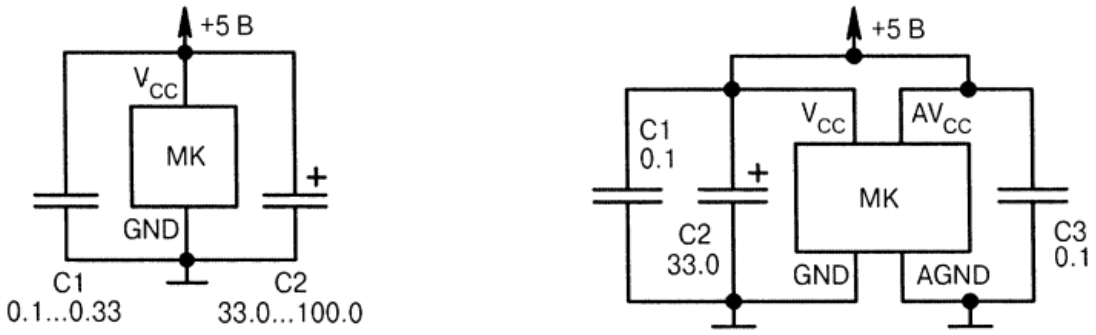

Рисунок 2.1 — Фільтрація завад в схемі живлення

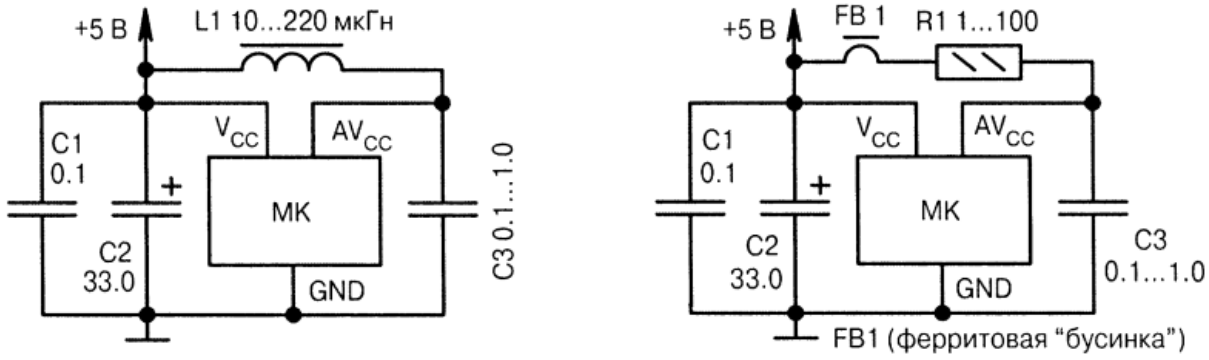

Рисунок 2.2 — Фільтрація завад LC-фільтром та RC-фільтром

Неполярні конденсатори С1, С3 (рис. 2.1) послабляють ВЧ-завади. Їх наявність обов'язкова біля будь-якого МК, причому максимально близько від виводів живлення (не більше 50 мм). Конденсатори мають бути керамічні, наприклад, К10-17 або SMD типу 0603... 1206. Базовий номінал ємності 0,1 мкФ вибраний умовно, тому що його легко запам'ятати. Полярний конденсатор С2 бажано використати танталовий, оскільки він краще подавлює імпульсні завади. При виборі ємності можна керуватися таким правилом – 1000 мкФ на кожен ампер струму навантаження. Наприклад, якщо цифрова частина МК споживає струм 10...30 мА, то досить поставити конденсатор С2 ємністю 10...30 мкФ з робочою напругою не менше 6,3В. Рекомендують вибирати конденсатори з напругою 10... 16 В, оскільки підвищується надійність в експлуатації і, головне, знижується внутрішній імпеданс, що дозволяє краще фільтрувати завади.

Конденсатор С2 обов'язковий при живленні від акумулятора як накопичувач енергії, а також при значних коливаннях і скачках напруги. У деяких випадках його функцію виконує конденсатор фільтру мережевого випрямляча або стабілізатора напруги.

Котушка індуктивності L1 (рис. 2.2) розв'язує цифрову і аналогову частини по високій частоті. Якщо її не ставити, то може зменшитись точність виміру АЦП і стабільність порогу спрацьовування аналогового компаратора. Значну частину завад по живленню створюють внутрішні цифрові вузли МК, тому LC - і RC - фільтри захищають контролер від самого себе. Номінал індуктивності L1 не особливо критичний і може змінюватися в широких межах.

Феритова «бусинка» FBI (Ferrite Bead) є провідник, пропущений через феритове кільце або циліндр. Цей елемент сприяє зниженню високочастотного випромінювання.

МК складається із статичних тригерів, регістрів і лічильників. Після подання живлення їх потрібно примусово встановити в певний логічний стан, інакше із-за загального хаосу виконання програми стане непередбачуваним. Імпульс початкового скидання подається на виводи RST (ReSeT) або RES (інверсний RESet). Відрізняються вони між собою, відповідно, позитивною і негативною формою сигналу.

Початковий скид у сучасних МК проводиться в таких випадках:

- Power-On — внутрішне автоматичне скидання, яке активізується відразу після подачі живлення;
- Brown-Out — скидання від внутрішнього детектора «просідань» напруги живлення;
- External Reset — зовнішній скид низьким рівнем на виводі RES;
- Watch-Dog — скидання від внутрішнього «сторожового» таймера при випадковій зупинці роботи центрального процесора або зависанні програми;
- JTAG - програмний скид через налагоджувальний інтерфейс JTAG.

Усі джерела скидання рівноцінні. Установка режимів скидання виконується бітами конфігурації, а також програмно-доступними регістрами з області SFR. Настроюватися можуть: поріг спрацьовування детектора напруги, що «просіла», тривалість часу затримки таймера очікування Watch - Dog.

Вузол апаратного скидання Power - On є присутнім в усіх без виключення МК. Якщо напруга живлення стабільна в часі і подається різким стрибком, то зовнішні елементи для скидання теоретично взагалі не потрібні. Скидання виконується автоматично вузлом Power - On при досягненню певного порогу. При високій швидкості наростання живлення (орієнтовно за час не більше 1...5 мс) вхід скидання RES підключають до кола VCC трьома способами:

безпосередньо, через зовнішній резистор опором 1-10 кОм або залишають вільним, покладаючись на внутрішній резистор МК.

Перший варіант повністю усуває шлях перешкодам, але виключає скидання кнопкою і можливість повторного програмування. Другий варіант дозволяє підключити кнопку скидання, що зручно при лабораторному макетуванні в домашніх умовах. Третій варіант дозволяється за відсутності перешкод і наявності усередині МК резистора підтяжки R_{RES} опором до 100 кОм.

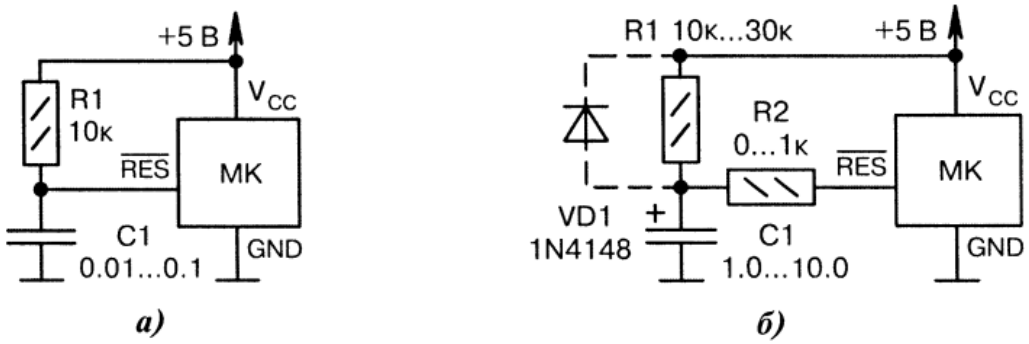

Рисунок 2.3 — Скидання зовнішньою RC ланкою: а) при середньому часу наростання живлення; б) при великому часі наростання живлення

Якщо напруга живлення наростає більше тривалий час (для різних сімейств МК по-різному), то рекомендується установка зовнішніх RC -ланок (рис. 2.3, а, б) із стандартними значеннями:

- R1 = 10 кОм, C1 = 0,1 мкФ при часі наростання 5...20 мс. Наприклад, подача живлення Vcc перемикачем, який розташовується між інтегральним стабілізатором напруги +5 В і МК;
- R1 = 10 кОм, C1 = 10мкФ при часі наростання 20...100мс. Це актуально, наприклад, при включенні пристрою в мережу 220 В загальним тумблером. Якщо ємність конденсатора C1 більше 1 мкФ, то для прискорення його розряду ставлять діод VD1 типу 1N4148 (КД522Б), а для захисту входу скидання від перенапруги ще і резистор R2. Ці перестраховки подовжують МК життя.

Вибір конкретної схеми скидання залежить від умов експлуатації. Наприклад, якщо сумарна ємність конденсаторів фільтру між VCC і GND складає більше 1000 мкФ, то, швидше за все, знадобиться зовнішня RC – ланка

(рис. 2.3, а, б). Якщо поряд з виводом RES на друкованій платі проходить силове комутаційне коло, то для з'ясування причин збоїв корисно тимчасово з’єднати лінію скидання МК з живленням. Якщо прилад розташовується поблизу від джерела потужних індустріальних завад, то на вході скидання рекомендується поставити додаткову мікросхему супервізора живлення, яка продублює вузол Brown - Out.

Для того, щоб МК запрацював, необхідно подати на центральний процесор тактові імпульси. Чим вище їх частота, тим швидше виконуються операції, а чим нижче їх частота, тим менше споживання струму. Формуванням тактових частот займається підсистема синхронізації. На її структурній схемі (рис. 2.4) є декілька вбудованих генераторних вузлів (on - chip oscillator): HF(High Frequency) - високочастотний, LF (Low Frequency) - низькочастотний, CLK (CLocK) - тактування.

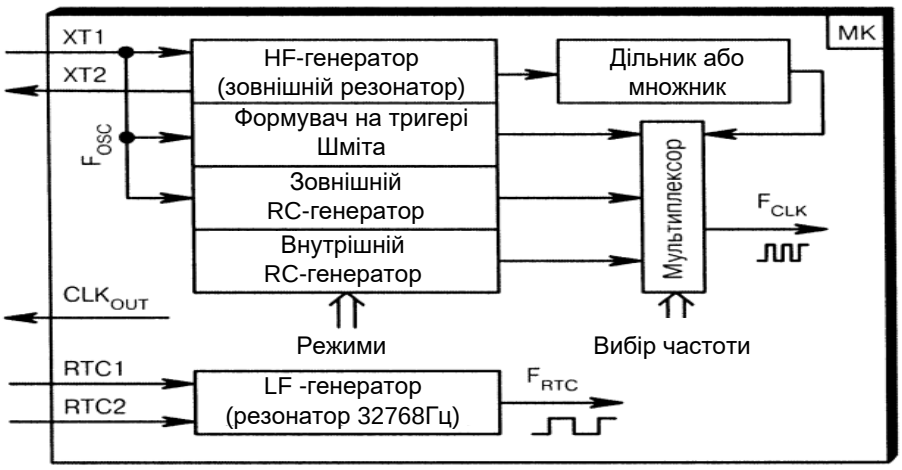

Рисунок 2.4 — Підсистема синхронізації МК

З чотирьох верхніх блоків, тільки HF-генератор використовує два виводи підключення XT1, XT2. Тим самим підкреслюється, що він розрахований на схему із зворотним зв'язком.

Виводи XT1, XT2 основного генератора в різних МК можуть позначатися по-різному: XTAL1, XTAL2, XI, X2, XIN, XOUT, OSC1, OSC2.

Управління режимами підсистеми синхронізації здійснюється через біти конфігурації. Вони перемикають канали мультиплексора, настроюють частоту

внутрішнього RC-генератора і так далі. Вони ж можуть дозволити/заборонити видачу сигналу CLKOUT (рис. 2.4) з окремої лінії порту, з частотою, у декілька разів менше тактовою. Ця функція є присутньою не в усіх МК.

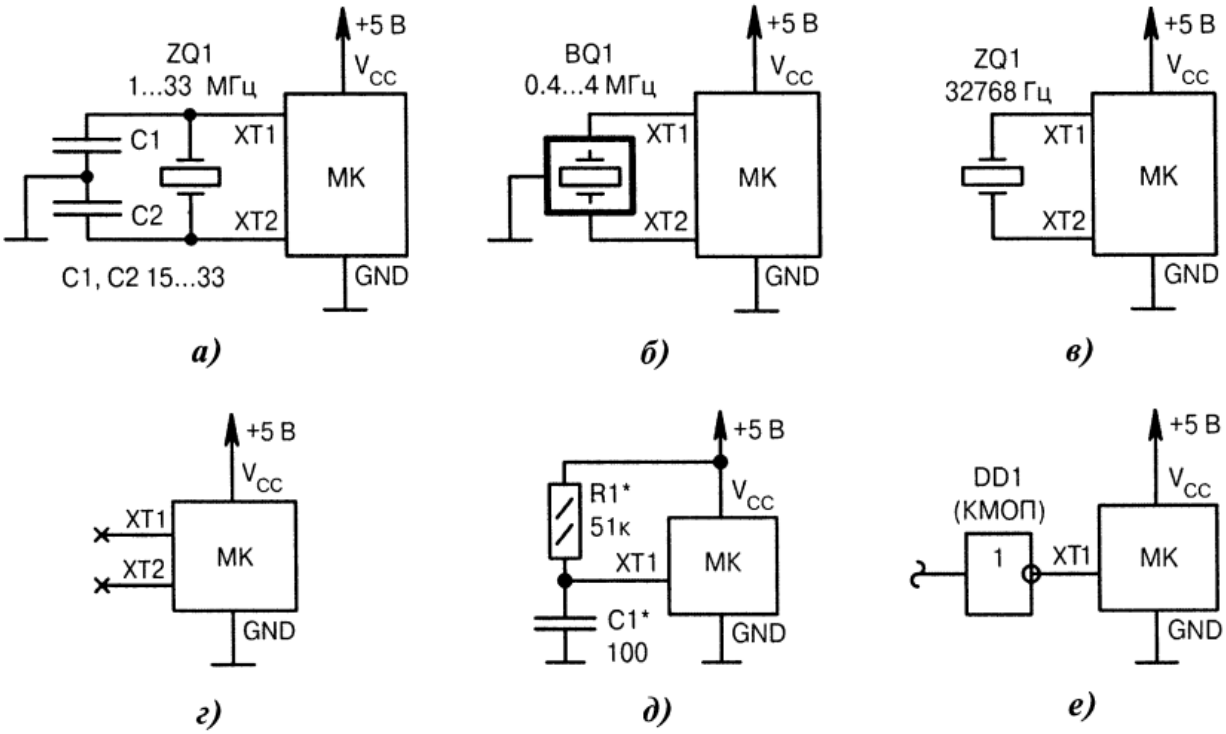

Рисунок 2.5 — Схеми формування тактової частоти : а) від ВЧ кварцового резонатора; б) від керамічного резонатора; в) від НЧ кварцового резонатора; г) від внутрішнього генератора; д) від RC-ланки; е) від зовнішніх імпульсів.

МК працює в таких режимах:

- від високочастотного кварцового резонатора 1...33 МГц (рис. 2.5, а);
- від середньо частотного керамічного резонатора 0,4...4 МГц (рис. 2.5, б);
- від низькочастотного кварцового резонатора 10... 100 КГц (рис. 2.5, в);
- від внутрішнього RC - генератора 1; 2; 4; 8 МГц (рис. 2.5, г);
- від зовнішньої RC - ланки 0,4...12 МГц (рис. 2.5, д);
- від зовнішніх синхроімпульсів 0...40 МГц (рис. 2.5, е).

Усі перераховані режими роботи задаються при програмуванні конфігураційних бітів

Лекція 10. Виконавчі пристрої мікропроцесорних систем управління

Практично жодна мікропроцесорна система управління не може обійтися без таких елементів, як виконавчі пристрої. Головне призначення будь-якої системи – це управління яким-небудь зовнішнім механізмом. Це можуть бути електродвигуни, нагрівачі, електромагнітні клапани. Тому, окрім датчиків, кнопок управління і елементів індикації до мікроконтролера обов'язково доведеться підключати і виконавчі пристрої. Для управління зовнішніми пристроями використовуються ті ж самі порти введення/виведення МК, які працюють на виведення. Сигнали з будь-якою з ліній будь-якого порту легко можуть бути використані для включення і виключення зовнішнього пристрою. Необхідно лише підсилити керуючий сигнал за потужністю до необхідного рівня. Для цього застосовуються різні схеми узгодження. Вибір схеми залежить від типу виконавчого пристрою.

У найпростішому випадку можна застосувати транзисторний ключ (рис 2.6). При використанні транзистора КТ315Г можна керувати зовнішніми колами із струмом споживання до 100 мА і напругою $U_Ж$ до 15 В. Транзистор допускає також високу напругу, проте підвищення напруги можливе при зменшенні струму.

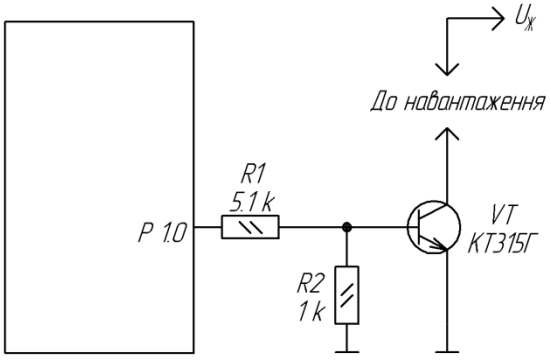

Рисунок 2.6 — Найпростіший транзисторний ключ

Для керування ланками з великим струмом потрібно застосувати потужніший транзистор або цілу транзисторну збірку. При виборі транзистора потрібно враховувати, що максимально допустимий струм навантаження для

будь-якого з виходів МК не повинен перевищувати величини 20 мА. При складанні програми потрібно не забувати, що будь-який транзисторний ключ інвертує сигнал. Якщо на виході P1.0 (рис. 2.6) встановити одиничний рівень, ключ відкривається і навантаження підключається до джерела живлення. При нульовому рівні на тому ж виході ключ закривається і навантаження відключається.

Якщо виконавчий механізм, яким повинна керувати МПС, живиться від мережі змінного струму 220 В, потрібно застосовувати схему управління з гальванічною розв’язкою. Один з можливих варіантів – релейна схема управління.

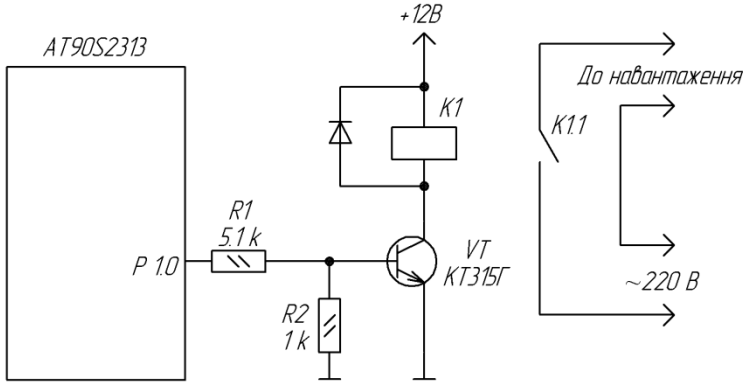

Рисунок 2.7 — Виконавчий пристрій з використанням реле

Типовий варіант схеми управління з використанням реле наведений на рис. 2.7. На схемі представлений електронний ключ, в навантаження якого включено електромагнітне реле К1. МК за допомогою ключа може вмикати і вимикати електромагнітне реле. Контакти реле, в свою чергу, керують навантаженням. Така схема забезпечує комутацію достатньо великої напруги і струму.

Гальванічна розв'язка між всіма колами МПС і силовою мережею 220 В забезпечує безпеку роботи з цією схемою. Діод VD1 призначений для захисту елементів схеми від напруги ЕРС самоіндукції, що виникає в котушці К1 у момент закривання ключа VT1. При виборі електромагнітного реле необхідно звертати увагу на такі параметри. По-перше, напруга спрацьовування реле. Для прикладу на рис. 2.7 вона має бути рівна 12 В. По-друге, максимально

допустимий струм комутації і максимально допустима напруга для виконавчих контактів реле. Вони повинні відповідати реальним значенням струму і напруги в колі навантаження.

Досить часто реле замінюють оптоелектронними комутаційними вузлами, які мають малі струми і напруги керування, беззвучні і довговічні у роботі, можливість робити в середовищах постійного і змінного струму, комутації напруги (деяких приладів) до 400...600 В і струмів до 0,5 А. На рис. 2.8 представлена одна з таких схем.

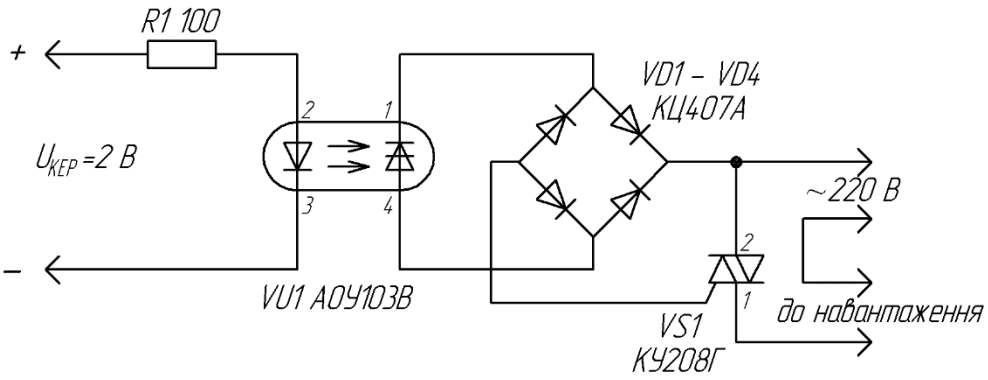

Рисунок 2.8 — Електрична схема із застосуванням оптрона

У цій схемі управління навантаженням (потужність якого може досягати 600 Вт) здійснюється симістором КУ208. Завдяки розв'язці по живленню - застосуванню оптоелектронного приладу АОУ103В, кола управління навантаженням в мережі 220 В і керуючі схеми повністю розв'язані. Керуюча постійна напруга (або імпульси) амплітудою 1,5...2 В потрапляє від схеми управління через обмежувальний резистор R1 на вхід оптопари VU1. Керуючий струм не перевищує 5 мА.

За наявності керуючого сигналу, тиристор усередині оптопари відкривається (його опір в прямому напрямку зменшується до декількох десятків Ом), і він шунтує діагональ випрямляючого моста VD1. Від випрямляючого моста напруга проходить на електрод керуючого симістора VST, завдяки чому він відкривається у відповідні напівперіоди напруги і в навантаженні тече струм. Використання оптопар АОУ103 залежить від напруги в електричному колі. Так,

для даної схеми та інших з напругою більше 200 В підходить лише оптопара АОУ103В.

При необхідності управління більш потужним навантаженням, наприклад до 1000 Вт, симістор, як основний пристрій в даній схемі, що комутує навантаження, слід встановити на охолоджуючий радіатор.

Схожа за принципом роботи схема представлена на рис. 2.9. Тут діагональ випрямного моста замикає оптосимістор ТО132-40 (або аналогічний ТО125-12,5, ТО106-10 та інші). Їх основна відмінність один від одного полягає у різних струмах і потужності комутації.

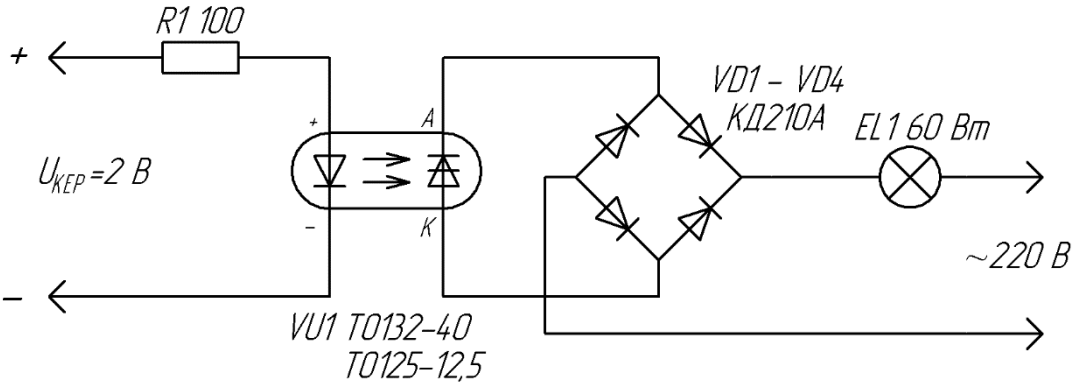

Рисунок 2.9 — Схема вузла управління навантаженням із застосуванням оптрона

На рис. 2.10 показаний ще один варіант включення - поєднання оптоелектронної розв'язки із застосуванням оптопари АОУ103В і симістора КУ208Г.

Управління пристроями навантаження ефективно здійснюється, якщо їх потужність не перевищує 600 Вт. Оптопара АОУ103В дозволяє самостійно комутувати високовольтне навантаження (з напругою до 350 В), проте струм комутації не повинен перевищувати 100 мА. Тому для управління потужним навантаженням в схему введений симістор КУ208Г.

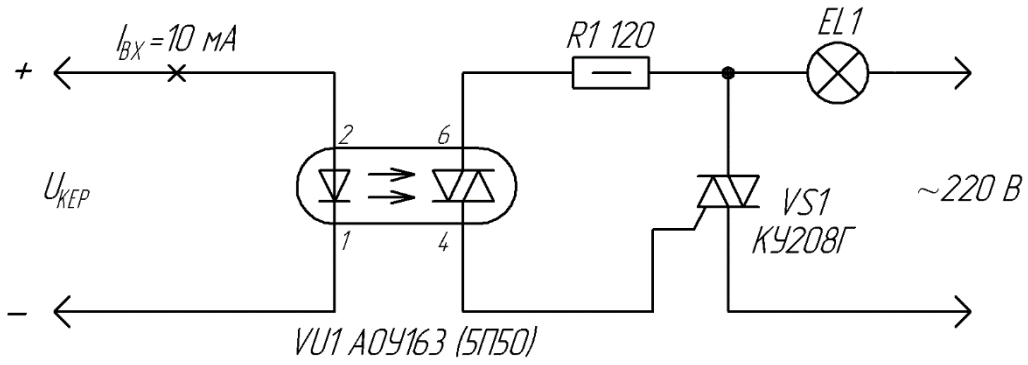

Рисунок 2.10 — Електрична схема оптоелектронної розв'язки

Лекція 11. Елементи індикації

Практично кожна мікропроцесорна система управління містить елементи індикації. Як індикатори в даний час найчастіше застосовуються світлодіоди. На ринку є величезний вибір світлодіодів найрізноманітніших видів і розмірів.

У мікропроцесорній системі управління LED індикатори можуть служити для відображення різних режимів роботи: попередження про критичні ситуації, відображення ходу прийому сигналів керування тощо. Підключити одиночний світлодіодний індикатор до МК дуже просто. На рис. 2.11 наведена схема підключення світлодіода безпосередньо до виводу порту МК.

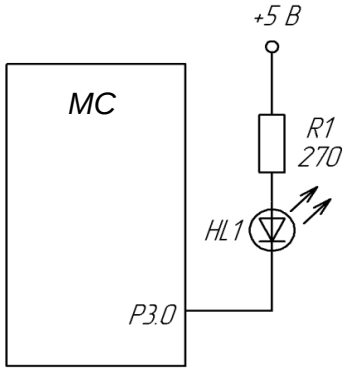

Рисунок 2.11 — Схема підключення світлодіодного індикатора

Усі вихідні каскади МК мають достатню навантажувальну здатність для того, щоб витримати підключення одного світлодіодного індикатора зі споживаним струмом у робочому режимі не більше 20 мА.

Для керування двома світлодіодами одним виходом у МК передбачено активні вихідні каскади, і для перемикання режиму роботи (введення або виведення) слугує спеціальний регістр. Таким чином, сигнал кожного виходу будь-якого порту може мати 3 значення – "0", "1" і високоімпедансний ("Z") стан. Це дозволяє керувати двома світлодіодами за допомогою одного виводу (рис. 2.12).

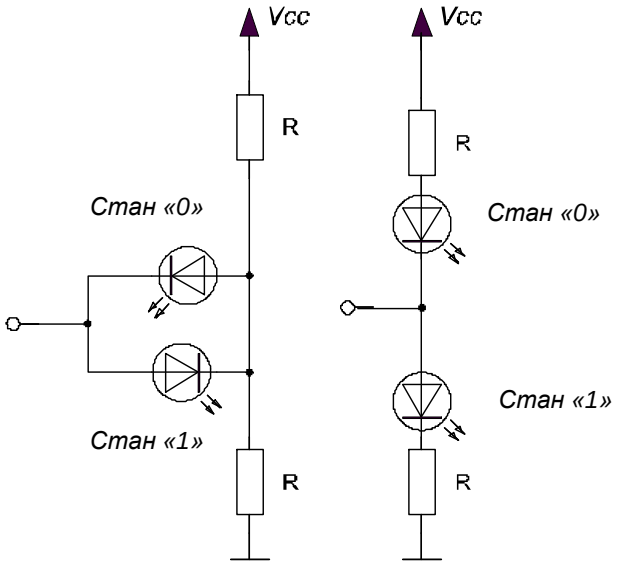

Рисунок 2.12 — Керування двома світлодіодами одним виходом МК

При роботі порту в режимі виходу, залежно від стану "0" або "1" горить відповідно верхній або нижній світлодіод. При перемиканні в Z-стан, і при відповідному виборі резисторів струм через світлодіоди дуже малий і їх світіння майже непомітно.

Дуже часто МК використовується не тільки для керування роботою, але й для того, щоб повідомити що-небудь користувачеві і/або отримати від нього які-небудь вказівки про подальшу роботу. Наприклад, електронний годинник, крім відліку часу, повинен його ще відображати, а також дозволяти змінювати покази (встановлювати точний час). Якщо вся «інформація» зводиться до мигання парою світлодіодів, яких-небудь спеціальних зусиль з відображення інформації з боку розробника не вимагається, але якщо таких світлодіодів виявляється два-три десятки, тут вже потрібне застосування додаткових засобів (як апаратних, так і програмних). Як правило, в цьому випадку відображення інформації виконують у режимі динамічний індикації — це найбільш економний за кількістю використовуваних ліній спосіб.

Для організації динамічної індикації застосовується матриця, що складається з ліній рядків і ліній стовпців (рис. 2.13). На перетині стовпця і рядка матриці розташований індикаторний елемент — світлодіод. Для того, щоб запалити той або інший елемент, необхідно подати на матрицю не один, як у

звичайних індикаторах, а два сигнали: логічна 1 на відповідному рядку і логічний 0 на відповідному стовпці матриці. Через односторонню провідність світлодіода кожна комбінація сигналів на входах рядків і стовпців однозначно включає рівно один індикаторний елемент.

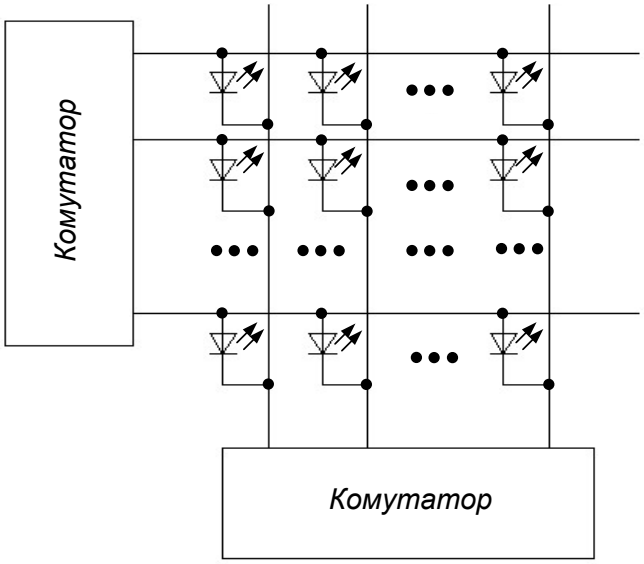

Рисунок 2.13 — Організація динамічної індикації

Головна перевага динамічної індикації – невелике число ліній, що керують: для матриці світлодіодів розміром N×N елементів потрібно всього 2N сигналів, що керують. За таку економію, втім, доводиться платити – справа в тому, що при почерговому виведенні інформації на кожен світлодіод матриці його яскравість світіння буде в N^2 разів нижча, ніж при безпосередньому виведенні інформації на один світлодіод, що "окремо стоїть". Тому в пристроях, що використовують динамічну індикацію, виведення інформації здійснюється не на кожен світлодіод окремо, а на один рядок або на один стовпець повністю – у цьому випадку яскравість світіння світлодіодів падає тільки в N разів.

Розглянемо декілька варіантів реалізації динамічної індикації із застосуванням МК [1, 2, 9, 10]. Як індикаторний елемент передбачається застосування семисегментних індикаторів, але кожен такий індикатор з легкістю можна замінити і групою світлодіодів.

Схема реалізації динамічної індикація без додаткових елементів наведена

на рис. 2.14. До порту B МК підключені катоди всіх світлодіодів матриці, а до порту A – аноди кожного з індикаторів, що створюють матрицю. На лініях порту A організовується одиниця, що «біжить». На лінії порту B при кожному положенні одиниці, що біжить, виводиться семисегментний код того символу, який повинен горіти в даному знакомісці. Для індикаторів із загальним катодом замість одиниці, що біжить, використовується нуль, що біжить. Перевага такого способу індикації — у відсутності яких-небудь додаткових компонентів (окрім самих світлодіодних індикаторів), головний недолік — значна перевитрата ліній портів. Таке рішення для МК може забезпечити роботу не більше 5 семисегментних індикаторів одночасно. При використанні інших МК з великою кількістю ніжок вказана проблема знімається.

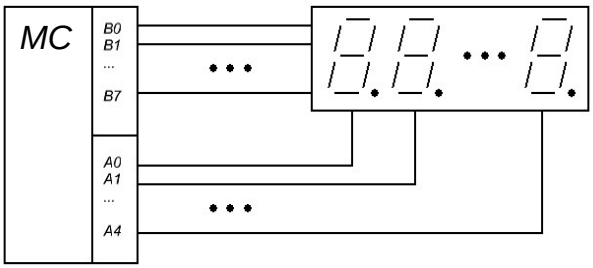

Рисунок 2.14 — Схема реалізації динамічної індикація без додаткових елементів

Схема реалізації динамічної індикації з одним додатковим елементом наведена на рис. 2.15. У цьому варіанті одиниця, що біжить, реалізується за допомогою регістра зсуву. Порт B у цій схемі також використовується в режимі часового мультиплексування, як для видачі символу, так і для занесення чергового біта до регістра зсуву. Схема також вимагає всього одну додаткову лінію (крім порту B).

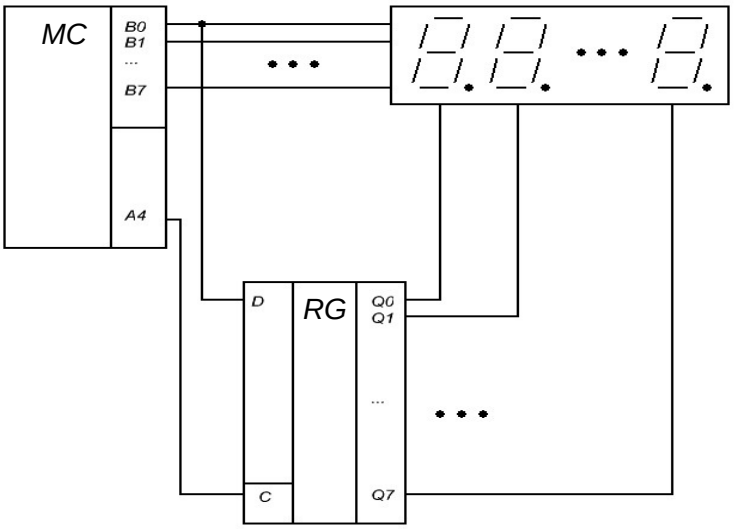

Рисунок 2.15 — Схема реалізації динамічної індикації з регістром зсуву

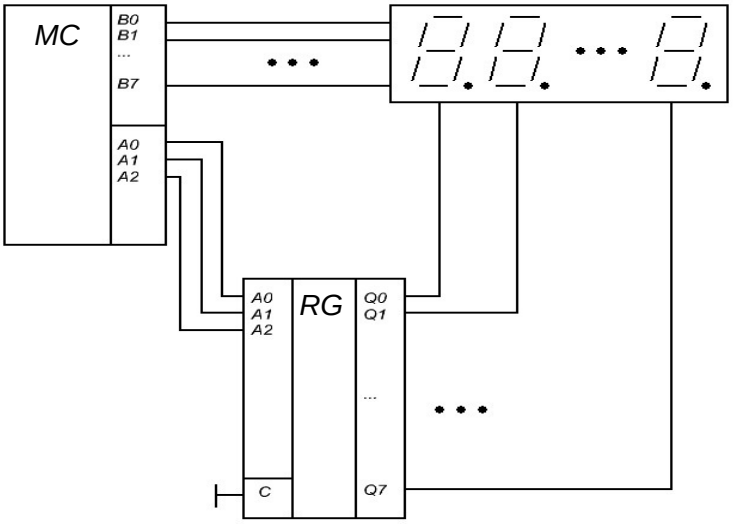

Рисунок 2.16 — Схема реалізації динамічної індикації з дешифратором

Ще один варіант реалізації схеми з одним додатковим елементом наведений на рис. 2.16. Така схема придатна тільки для індикаторів із загальним катодом, оскільки для організації нуля, що біжить, в ній використовується дешифратор. Схема вимагає три додаткові лінії (крім порту B), проте у багатьох випадках вона може виявитися корисною.

Лекція 12. Кнопки та датчики. Оптичні датчики

За допомогою кнопок і простих датчиків в мікропроцесорну систему управління надходить різна інформація, яка використовується для зміни алгоритму роботи програми. Схема підключення контактного датчика до МК наведена на рис. 2.17. У наведеному прикладі датчик підключений до лінії PD0 порту D МК. Через цей вхід МК проводить зчитування стану датчика. Датчик можна підключити і до будь-якої іншої лінії будь-якого з портів МК.

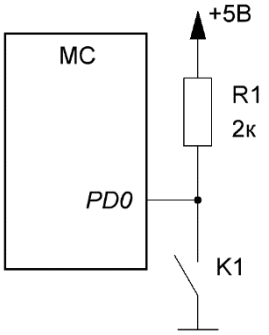

Рисунок 2.17 — Підключення контактного датчика до МК

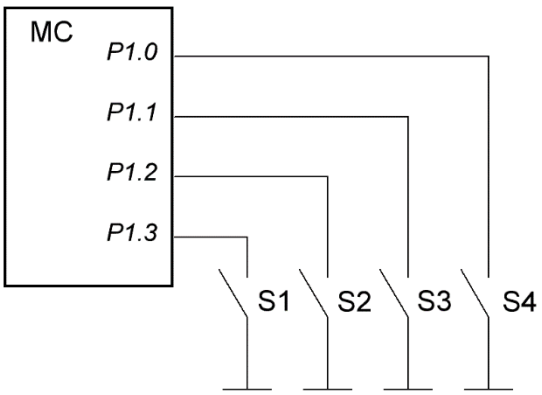

Рисунок 2.18 — Підключення кнопок або простих датчиків до МК

У початковому стані контакти датчика розімкнені. На вхід МК через резистор R1 прикладається напруга від джерела живлення + 5 В. МК сприймає цю напругу як сигнал логічної одиниці. При спрацьовуванні датчика контакти замикаються і з’єднують вивід МК із загальним дротом. Тепер мікросхема

сприймає вхідний рівень сигналу як логічний нуль. Резистор R1 при цьому служить струмообмежувальним елементом, запобігаючи короткому замиканню між шиною живлення і загальним дротом. Деякі МК мають свої внутрішні резистори навантаження, які можуть замінити зовнішній резистор. Схема підключення декількох датчиків або кнопок до МК зображена на рис. 2.18.

У схемі, що зображена рис. 2.19, при натисканні однієї з клавіш змінюється постійна напруга на відповідному вході процесора, яка розпізнається процесором і дешифрується в певну команду. Ця напруга максимальна (приблизно 5 В), коли кнопки не натиснуті, і мінімальна (0 В) при натиснутій клавіші S1 [1, 2, 10].

Існує два види клавіатур, що підключаються до МК: зі скануванням і з кодуванням.

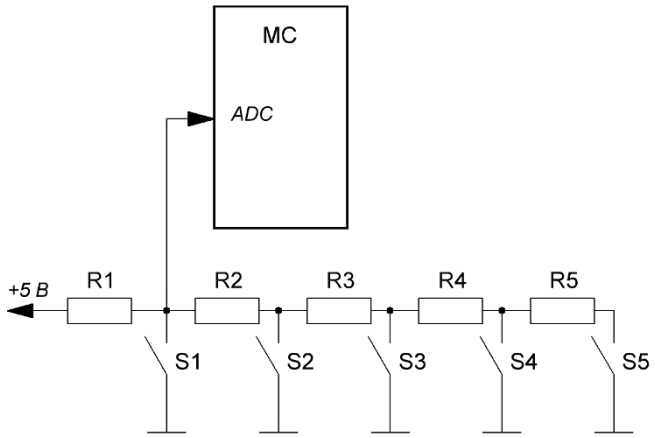

Рисунок 2.19 – Підключення кнопок зміною напруги на аналоговому вході МК

Блок-схема 12-клавішної клавіатури зі скануванням показана на рис.2.20. Клавіші розташовані у вузлах матриці, у якої чотири лінії рядків і три лінії стовпців. На лінії стовпців по черзі подається негативний імпульс (логічний «0»). У цей момент перевіряється стан чотирьох ліній рядків. Якщо натиснутих клавіш немає, всі лінії рядків мають високий рівень (вони підключені до напруги +5 В через резистори). Якщо ж клавіша натискається, і на лінії стовпця, відповідного натиснутій клавіші, все ще нуль, то адекватна лінія рядка також стає рівною

нулю. Знаючи номери стовпця і рядка, можна отримати позицію натиснутої клавіші.

У клавіатурі з кодуванням застосовують спеціалізовані мікросхеми, які виявляють натискання клавіші і передають її код. Прикладом такого пристрою є мікросхема MM74C922 (National Semiconductors) [2].

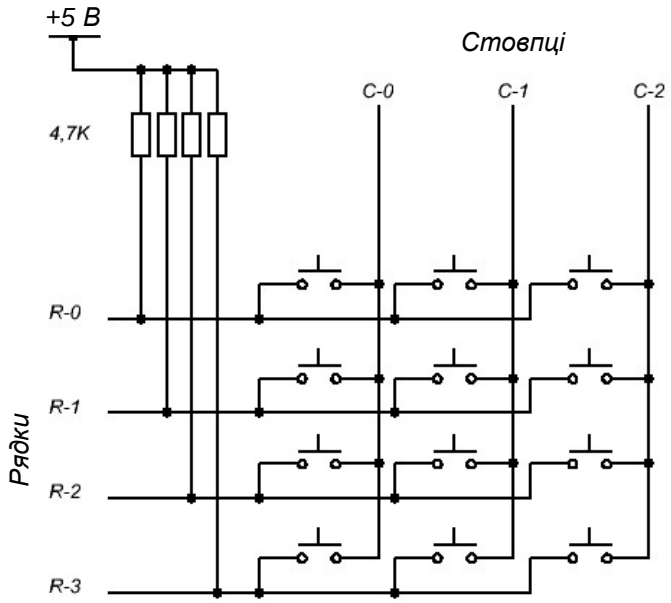

Рисунок 2.20 — Матрична клавіатура 4×3

Як оптичний датчик найчастіше виступає світлодіод та фотоприймач, який називається оптроном. Випускаються оптрони з закритим (optoisolator) оптичним каналом (у монолітному виконанні) і відкритим оптичним каналом (щілинні і відбивальні оптрони).

На рис. 2.21 показаний оптичний датчик — щілинний оптрон (slotted optical switch). Фототранзистор і направлений на нього світлодіод закріплені на пластиковій підставці і розділені проміжком так, що коли якийсь предмет рухається в зазорі, він перекриває світло між світлодіодом і датчиком. Щілинні оптрони часто використовуються для вимірювання швидкості двигуна за допомогою диска з прорізами, розміщеного на осі двигуна. Коли вісь обертається, диск перекриває світловий промінь. Інше застосування щілинного оптрона – це індикація того, відкриті чи закриті двері або, наприклад, кожух

охоронного приладу. Прапорець на дверях, потрапляючи в щілину, блокує світло, коли двері закриваються. Механічна комп'ютерна миша також використовує щілинні оптрони.

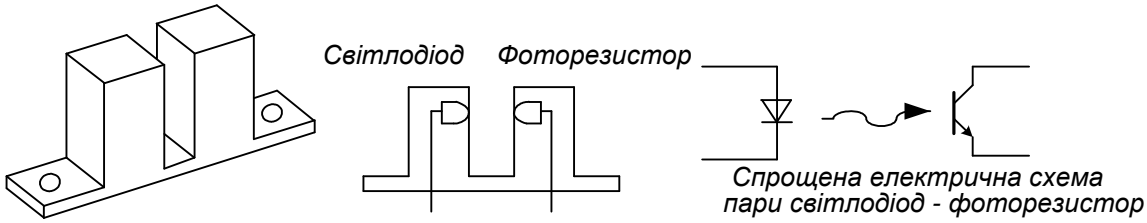

Рисунок 2.21 — Щілинний оптрон

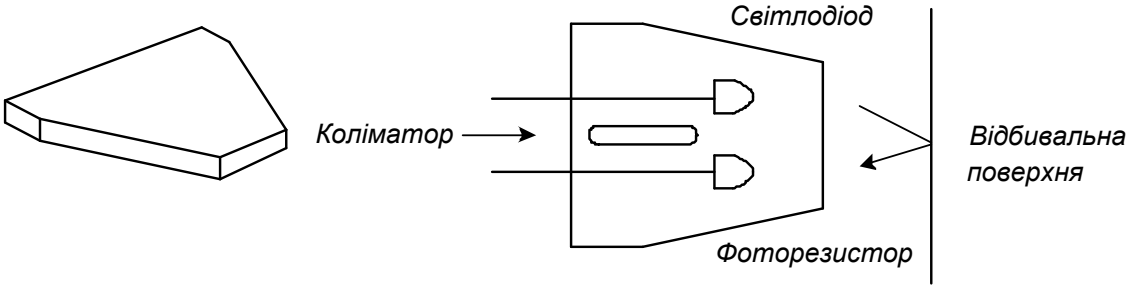

Рисунок 2.22 — Відбивальний оптрон

На рис. 2.22 показаний інший тип оптичного датчика – відбивальний оптрон (reflective sensor). Принцип роботи цього датчика такий же, як і щілинного, з тією різницею, що фототранзистор приймає відбите, а не пряме світло. Більшість датчиків відбиття характеризується фокусною відстанню – оптимальною відстанню, на якій має бути розміщений відбивальний об'єкт від датчика. Ця відстань дорівнює 0,254... 1,270 см (від 0,1 до 0,5 дюйма). Типове застосування відбивальних оптронів – це реєстрація обертання двигуна за нанесеними на його вісь темними мітками [2, 10].

Коли вісь обертається, датчик реєструє зміну темних і відбивальних ділянок. Обидва типи оптронів мають схожі характеристики, які необхідно враховувати при проектуванні систем.

Лекція 13. Пристрої формування звукових сигналів. Пристрої управління двигунами постійного струму

П'єзоелектричні динаміки служать для генерації звуків. Вони мають максимальну вхідну напругу 50 В і номінальний струм 10 мА. На рис. 2.23 зображена схема, що використовує буфер для керування таким динаміком. Схема пристрою керування на транзисторі ZTX300 наведена на рис. 2.23 [2, 9]. Щоб отримати звук, необхідно подати на вхід послідовність імпульсів.

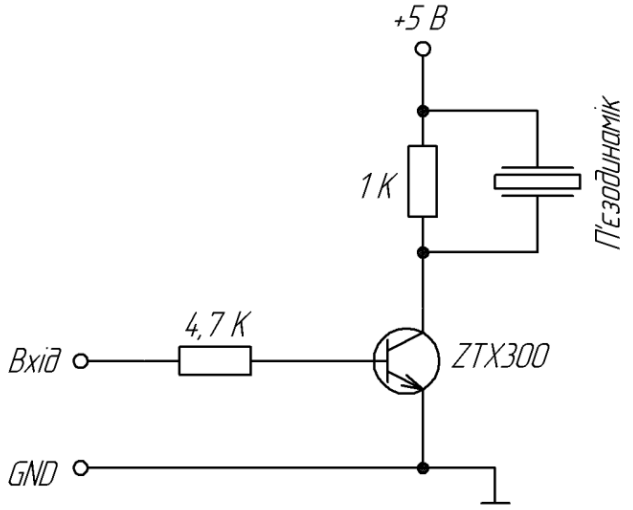

Рисунок 2.23 — Схема керування п’єзоелектричним динаміком

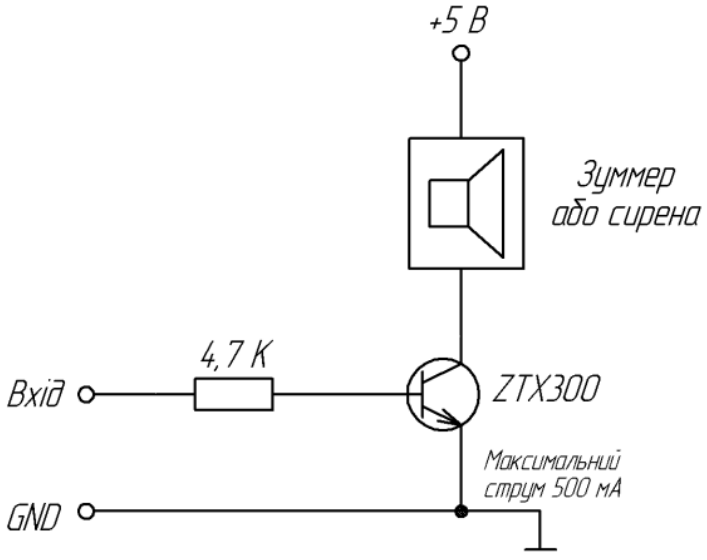

Рисунок 2.24 — Схема керування зумером або сиреною

Напівпровідникові зумери — це автономні динаміки, здатні генерувати тон,

частотою близько 450 Гц. На рис. 2.24 наведена схема керування на транзисторі *ZTX300*. Для генерації звуку на базу *ZTX300* необхідно подати високий рівень напруги. При керуванні сиренами можна використовувати такі ж схеми.

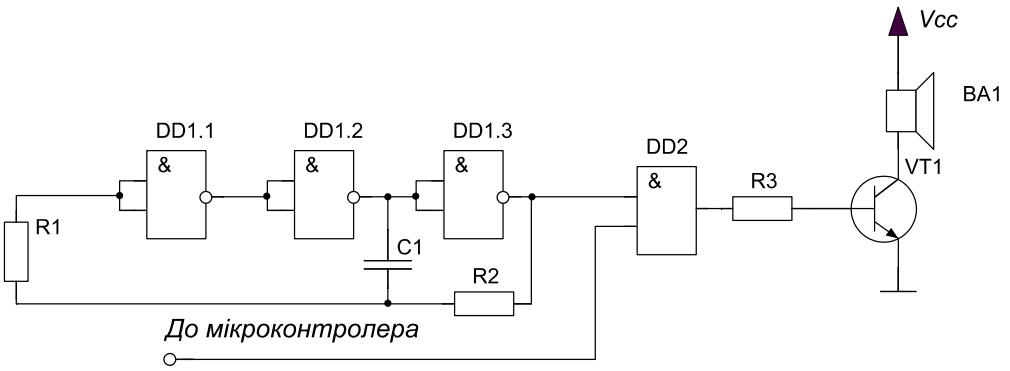

Рисунок 2.25 — Схема формування звукового сигналу апаратним способом

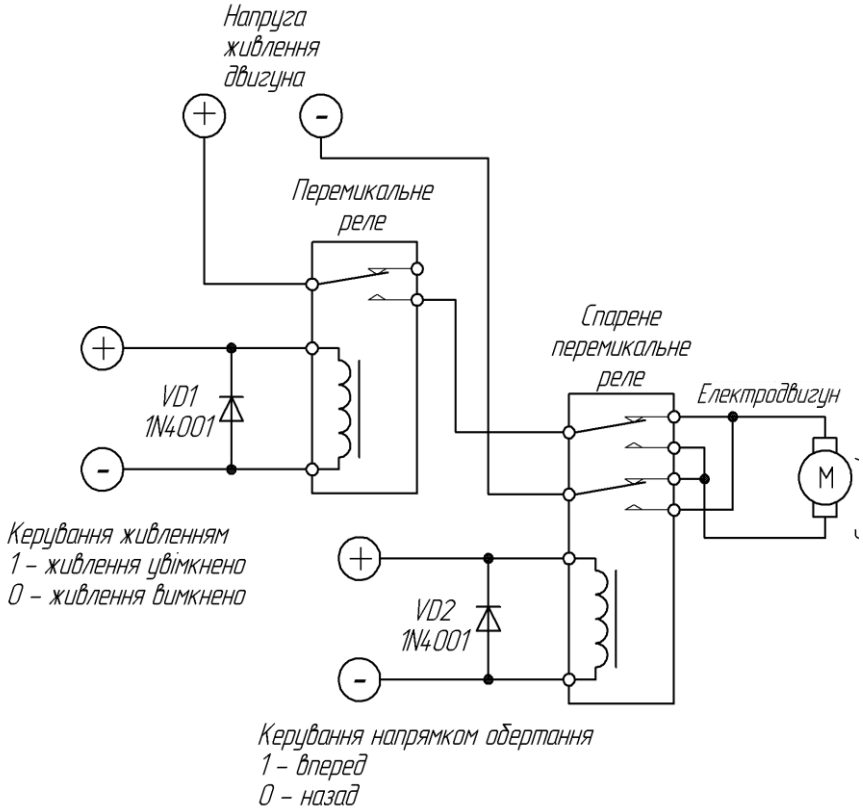

Рисунок 2.26 — Схема керування двигуном постійного струму на базі реле

На рис. 2.25 зображена схема формування звукового сигналу апаратним способом. Схема складається з генератора звукового діапазону на логічних

елементах І та схеми керування звуковим пристроєм. При формуванні МК сигналу логічної «1» імпульси від генератора проходять на звуковий пристрій.

Двигунами постійного струму можна керувати за допомогою реле або транзисторів (рис. 2.26). Одиночне перемикальне реле вмикає і вимикає двигун, а спарене реле відповідає за напрям обертання (рис. 2.26).

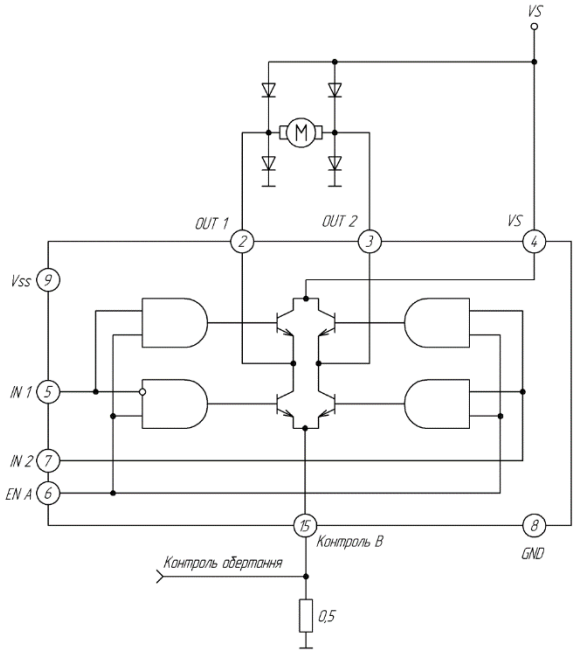

Рисунок 2.27 — Схема керування двигуном постійного струму

Інший спосіб керування двигунами постійного струму заснований на використанні мостових схем типу L298N (SGS-Thomson, RS636-384) [9, 10]. Це двоканальний пристрій, в якому присутня потужна напруга до 46 В, струм до 2А на кожен канал. Такий пристрій працює від рівнів ТТЛ (рис. 2.27). З виводу VS (контакт 4) подається напруга живлення для двигуна, на вивід Vss (контакт 9) подається напруга живлення схеми (+5В). Виводи EN A і EN B (контакти 6 і 11) відкривають входи двох каналів. Входи IN 1 і IN 2 (контакти 5 і 7) керують першим каналом, а IN 3 і IN 4 – другим. Емітери транзисторів з’єднані для підключення зовнішніх датчиків контролю. Типова схема включення для одного каналу наведена на рис. 2.27. Коли на вході EN A низький рівень, входи заблоковані і двигун не обертається. Якщо на цей вхід подати високий рівень,

входи відкриваються. Входи IN 1 і IN 2 керують режимами роботи двигуна таким чином: IN 1 «1», IN 2 «0» — двигун обертається за годинниковою стрілкою; IN 1 «0», IN 2 «1» — двигун обертається проти годинникової стрілки; IN 1 = IN 2 — двигун не обертається.

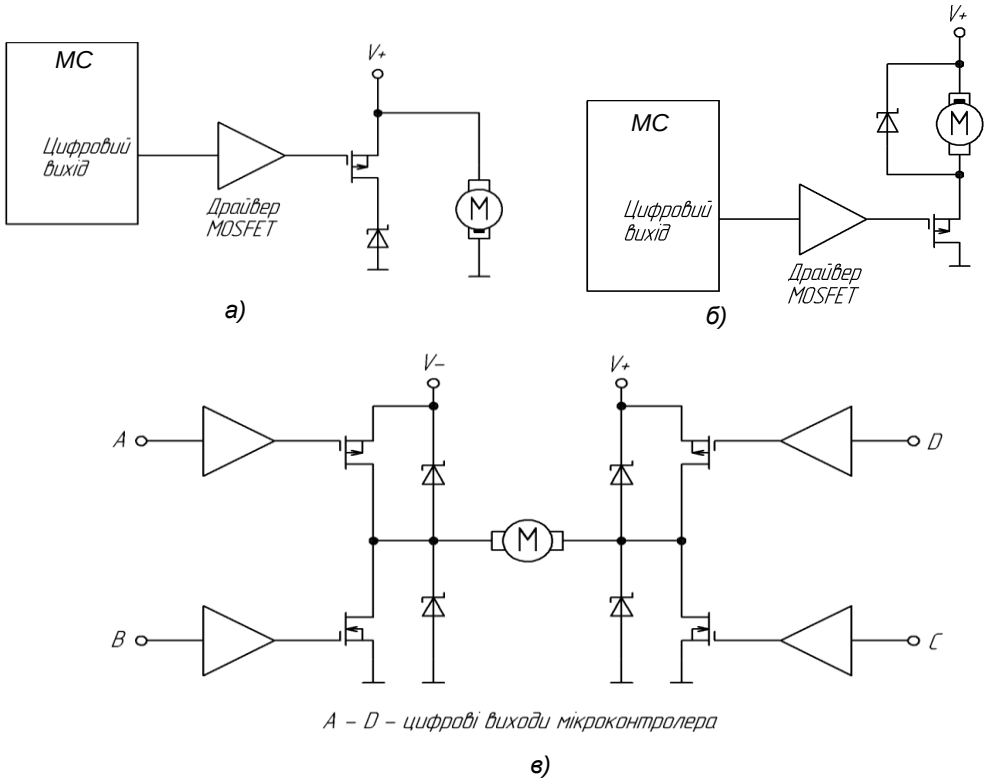

Рисунок 2.28 — Керування колекторним двигуном постійного струму: а) схема з верхнім розташуванням ключа; б) схема з нижнім розташуванням ключа; в) мостова схема

При керуванні колекторними двигунами постійного струму потрібно регулювати струм, що проходить через обмотки двигуна. Цей процес охоплює регулювання напрямку магнітного потоку і величини струму. Проста схема керування наведена на рис. 2.28. Дані схеми дозволяють керувати обертанням двигуна лише в одному напрямку.

Схема з верхнім розташуванням ключа часто застосовується в системах з підвищеними вимогами до безпеки – коротке замикання не призводить до увімкнення двигуна, у вимкненому стані обидва виводи обмотки підключено до спільної точки схеми. Схема з нижнім розташуванням ключа найдешевша,

оскільки для керування силовим транзистором MOSFET досить подавати на затвор сигнал з цифрового виходу МК без використання спеціального драйвера. Для реверсивного керування двигуном потрібно використовувати мостову схему включення, наведену на рис. 2.28. Частота обертання двигуна регулюється за допомогою зміни значення напруги на обмотці якоря. При використанні МК цю напругу можна регулювати за допомогою широтно-імпульсної модуляції.

Для вимірювання частоти обертання двигуна можна використовувати ефект зворотної ЕРС або використовувати оптоелектронний датчик положення ротора («ромашка») (рис. 2.29).

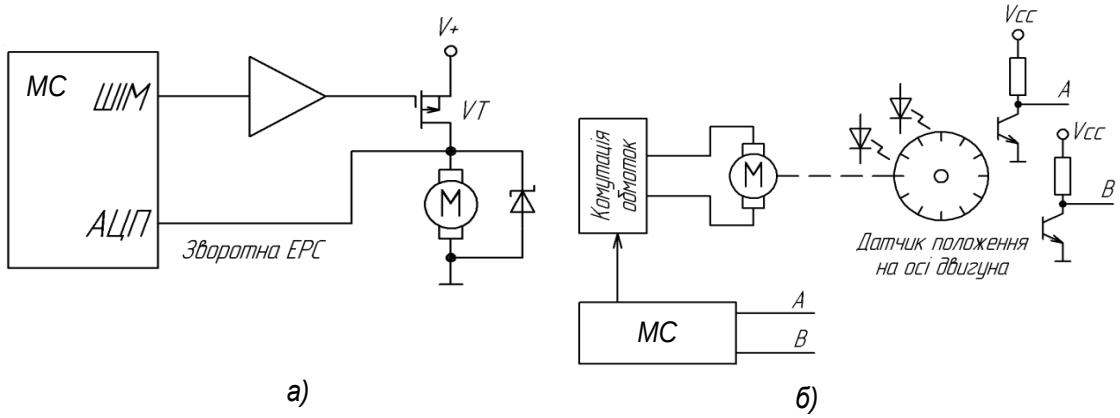

Рисунок 2.29 — Вимірювання швидкості обертання двигуна: а) з використанням зворотної ЕРС; б) з використанням датчика положення ротора

Безколекторні двигуни є прикладом спрощення конструкції з одночасним ускладненням схеми керування. Двигун не може самостійно перемикати обмотки (керувати струмом), тому схема керування повинна самостійно правильно регулювати величину струму в обмотках для забезпечення рівномірного обертання валу двигуна. Схема керування містить півмостову схему включення кожного з трьох виводів обмоток. Існують 2 основних типи керування безколекторним двигуном: з датчиками і без датчиків. Для того, щоб включати обмотки в потрібній послідовності, необхідно використовувати різні методи визначення положення ротора. Двигун з датчиком завжди повідомляє МК про положення ротора. Кожному положенню ротора відповідає певний набір керівних дій, які подаються на мостову схему включення обмоток.

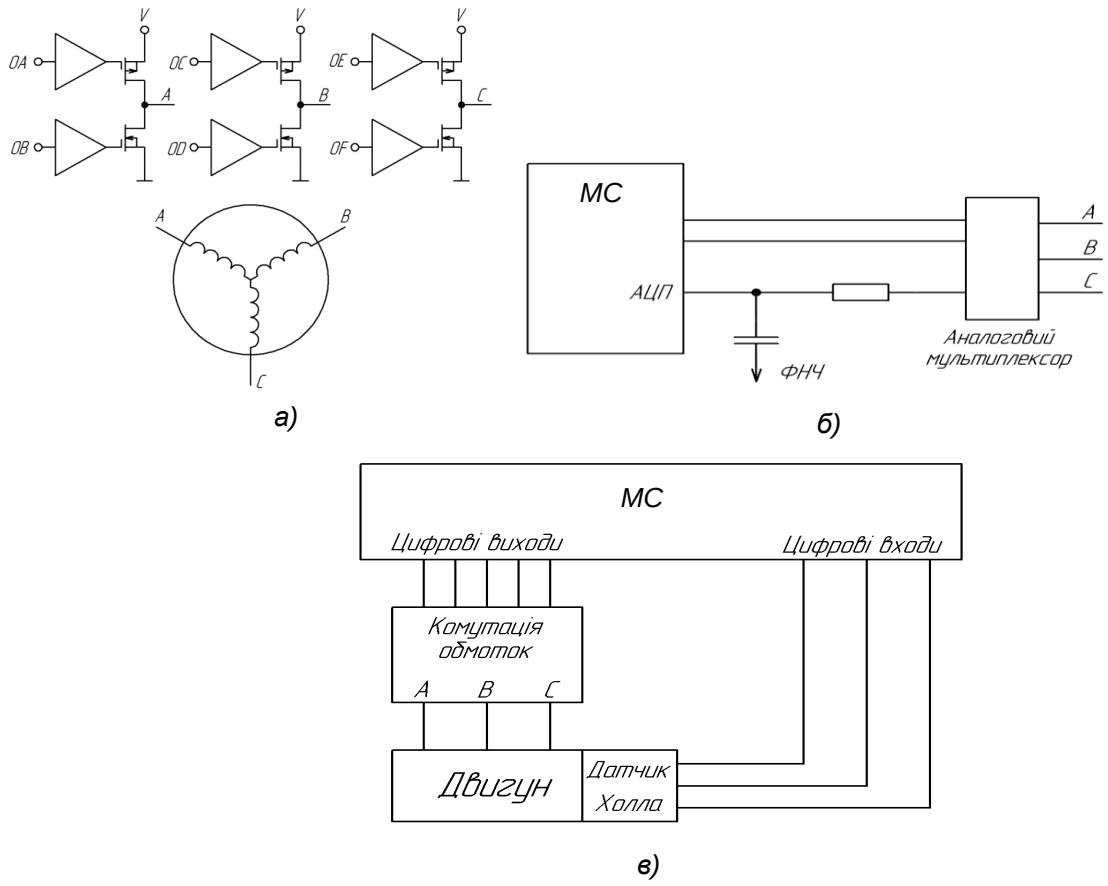

Рисунок 2.30 – Керування безколекторним двигуном постійного струму: а) спільна схема включення; б) схема включення без датчика положення; в) схема включення з датчиком положення.

У двигунах без датчика положення ротора визначається за величиною ЕРС, що виникає в непідключеній обмотці. Двигуни без датчиків простіші у виготовленні, але складніші в управлінні. Їх застосовують в умовах, що не вимагають частих запусків і зупинок. Двигуни з датчиками – кращий вибір для умов, пов'язаних з періодичними зупинками і запусками. Схеми включення двигуна наведені на рис. 2.30. Складність побудови схем керування не залежить від типу двигуна. Безколекторні двигуни мають кращі показники надійності, питомої потужності і економічності в порівнянні з колекторними, тому рекомендується використовувати безколекторні двигуни.

Лекція 14. Периферійний послідовний інтерфейс UART, SPI

До складу мікроконтролерів AVR входить універсальний дуплексний послідовний порт (UART). Його основні можливості: широкий діапазон швидкостей обміну даними; висока швидкість передачі при низькій частоті XTAL; 8 або 9-розрядний формат даних; виявлення помилок утрати даних при прийомі; виявлення помилок формату кадрів; виявлення помилкового стартового біта; три окремих переривання: по завершенню передачі, по порожньому регістру передавача і по завершенню прийому.

При передачі модуль UART додає до вхідного символу (8 або 9 біт) на початку — старт-біт (нуль), а в кінці — стоп-біт (одиниця), формуючи таким чином 10- або 11-бітову послідовність. Отримані значення передаються до регістра зсуву, який по черзі передає біти на вихід передавача TXD (вивід PD1). Швидкість видачі біт на вихід передавача визначається параметром *baud rate* (швидкість передачі інформації; вимірюється в бодах), яким можна керувати.

Приймач модуля UART безперервно перевіряє стан входу RXD, на якому за відсутності даних встановлюється рівень «1». Приймач зчитує інформацію з входу в 16 разів швидше. При виявленні на виводі RXD рівня «0» (тобто можливого старт-біта) мікроконтролер пропускає шість відліків, а потім робить три вибірки. Ці вибірки доводяться на відлік 8, 9 і 10 для кожного біта, що приймається, і, таким чином, зчитування значення біта відбувається в середині інтервалу його передачі, що дозволяє працювати з сигналами, що мають фронти великої тривалості. Якщо мікроконтролер виявляє, що на виводі RXD все ще присутній рівень «0», тобто прийшов стар-біт, модуль UART переходить в робочий режим і починає зчитувати байт. Якщо ж на виводі RXD вже присутній рівень «1», вважається, що перший відлік був просто шумом, і модуль переходить до очікування коректного символу. Якщо приймач визначив, що прийшов дійсний символ, він починає брати по три відліки кожного біта в середині інтервалу його передачі. Якщо значення всіх трьох відліків біта не збігаються, то значення біта набуває рівним значенню двох однакових відліків.

На завершення модуль зчитує вибірки, що відносяться до стоп-біту. Для того, щоб було вирішено про коректний прийом символу, принаймні, дві з цих вибірок мають дорівнювати одиниці. Інакше модуль вважає символ за невірно кадрований і реєструє помилку кадрування (framing error).

Периферійний послідовний інтерфейс SPI застосовується як для з’єднання МК між собою, так і МК з периферійними пристроями. У одному сеансі зв'язку беруть участь лише 2 пристрої, з яких один обов'язково МК, а інший або МК, або периферійний пристрій з інтерфейсом SPI (АЦП, датчик, пам'ять, виконавчий пристрій). У периферійному послідовному інтерфейсі SPI використовуються цифрові сигнали «такти» (clock), «вибір кристала (мікросхеми)» (chip select), «вхід даних» (data input) і «вихід даних» (data output), але немає адресних сигналів.

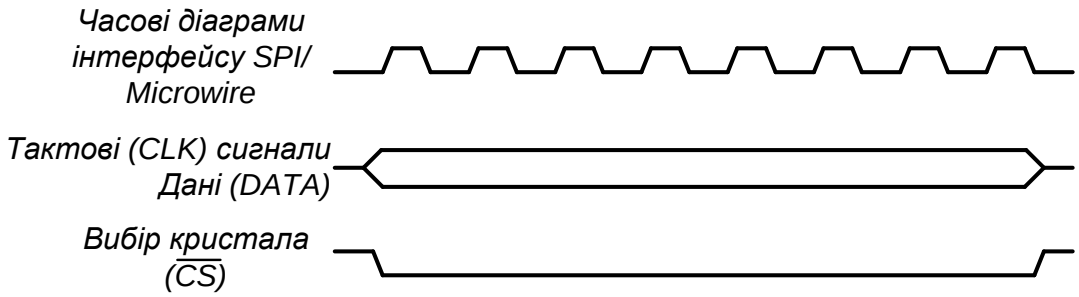

Рисунок 2.31 — Шина SPI

Кожен з пристроїв, підключених до шини SPI, вимагає наявності окремого сигналу $\overline{CS}$, яким він вибирається. Два пристрої працюють в режимі «ведучий-ведений». Дані, які потрібно передати, завантажуються в 8-бітові регістри портів і вхід даних ведучого пристрою з'єднується з виходом даних веденого. У свою чергу, вихід даних ведучого пристрою з'єднується з входом даних веденого. Також підключені до МК (ведучого пристрою) останні вище перелічені ведені пристрої, які включаються за окремими сигналами «вибір кристала». Хоча регістри мають по 8 бітів, але при такому підключенні входів і виходів утворюється спільний 16-бітовий регістр зсуву і, крім того, одночасно передаються по двох лініях дані в обох напрямах. Зсув проводиться тактовими сигналами від тактового генератора ведучого пристрою.

Лекція 15. Організація обміну даними інтерфейсом I²C, 1-Wire

У двонаправленій шині I²C використовується лише 2 лінії: послідовна лінія синхронізації SCL (SCLock) і послідовна лінія даних SDA (SDAta) [2]. Сигнал на лінії SCL формується процесором для синхронізації даних периферійного пристрою. Обидва виводи (SDA і SCL) виконуються з відкритим колектором або відкритим стоком, що визначається типом мікросхеми. Вони з'єднані з позитивним джерелом живлення через резистор навантаження за схемою «Монтажне І» і, таким чином, декілька пристроїв можуть одночасно використовувати шини SCL і SDA.

При передачі даних сигнал SDA можна змінювати лише доти, доки на SCL встановлений низький рівень. Коли на SCL високий рівень, перепади на лінії SDA з одного рівня в інший інтерпретуються як умови «СТАРТ» і «СТОП». Якщо SDA переходить у низький рівень, тоді як на лінії SCL високий, усі периферійні пристрої на шині сприйматимуть цю подію як старт-умову. Якщо SDA переходить у високий рівень, коли на SCL – високий, генерується стоп-умова. Процесор генерує старт-умову, а потім посилає одночасно всім периферійним пристроям адресу довжиною 7 бітів, повідомляючи, який з них вибраний, і восьмий біт читання/запису (0 – запис, 1 – читання). 7-бітова адреса дозволяє підключати по I²C-шині 2^7, або 128, периферійних пристроїв за однієї умови: ємність шини не повинна перевищувати 400 пФ.

Під час передачі в першому байті восьмого біта читання/запису (R/W) процесор встановлює напрям передачі: програмує R/W на запис (передачу) даних від МК до периферійного пристрою, якщо цей біт дорівнює 0. Інакше МК налаштовується на читання (прийом) даних від периферійного пристрою з адресою, вказаною в 7 бітах. Слід відмітити, що перші 7 бітів передаються, починаючи з старшого біта і закінчуючи молодшим бітом, а восьмим бітом передається біт R/W. Після прийому кожного байту, включаючи адресний, вибраний периферійний пристрій (приймальна сторона) по лінії SDA посилає сигнал підтвердження на виконання функції прийому переведенням рівня на лінії

SDA в низький, щоб показати, що він прийняв адресу і умову читання/запису.

Після прийому біта підтвердження процесор встановлює адресу іншого периферійного пристрою, з яким він хоче встановити зв'язок. Довжина цього поля залежить від периферійного пристрою. Потім приймається біт підтвердження і передаються дані. При виконанні операції запису процесор синхронізує вихідні 8 бітів, а при читанні МП встановлює виведення SDA як вхід і синхронізує вхідну 8-бітову послідовність. Дані завершуються бітом підтвердження.

Деякі периферійні пристрої дозволяють зчитувати або записувати декілька байтів за одну передачу. Процесор повторює послідовність команд «Дані/біт підтвердження» (data/acknowledge) до тих пір, поки всі байти не будуть передані. Периферійний пристрій збільшуватиме свою внутрішню адресу після кожної передачі.

Шина 1-Wire є основою мереж MicroLAN і розроблена наприкінці XX століття фірмою Dallas Semiconductor [2, 9]. Шина 1-Wire побудована за технологією Master/Slave. На шині повинен бути хоча б один ведучий пристрій (Master). Всі інші пристрої повинні бути веденими (Slave). Ведучий 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-пристрою до іншого безпосередньо неможлива. Для того, щоб Master міг звертатися до кожного з ведених пристроїв по шині, кожний ведений пристрій містить у собі індивідуальний код (ID-код).

Протокол l-Wire містить у собі спеціальну команду пошуку, за допомогою якої ведучий пристрій (Master) може здійснювати автоматичний пошук ведених пристроїв. У процесі пошуку Master визначає ID-коди для всіх підключених до мережі мікросхем. Пошук відбувається шляхом поступового відсіювання неіснуючих адрес. Тому для того, щоб знайти всі пристрої, що підключені до шини, потрібен досить значний час. Середня швидкість пошуку елементів у мережі MicroLAN становить близько 75 вузлів за секунду.

Обмін інформацією по шині 1-Wire відбувається за такими правилами.

1. Обмін завжди ведеться з ініціативи одного ведучого пристрою, що у більшості випадків є мікроконтролером.

2. Будь-який обмін інформацією починається з подачі імпульсу скидання («Reset Pulse» або просто RESET) у лінію 1-Wire ведучим пристроєм.

3. Для інтерфейсу 1-Wire у загальному випадку передбачається «гаряче» підключення й відключення пристроїв.

4. Будь-який пристрій, підключений до 1-Wire, після одержання живлення видає в лінію DQ імпульс присутності, який називається «Presence pulse». Цей же імпульс пристрій завжди видає в лінію, якщо виявить сигнал RESET.

5. Поява в шині 1-Wire імпульсу PRESENCE після видачі RESET однозначно інформує про наявність хоча б одного підключеного пристрою.

6. Обмін інформації ведеться так званими тайм-слотами: один тайм-слот служить для обміну одним бітом інформації.

7. Дані передаються побайтово, біт за бітом, починаючи з молодшого біта. Ймовірність переданих/прийнятих даних (перевірка відсутності спотворень) гарантується шляхом підрахунку циклічної контрольної суми.

На рис. 2.32 показана діаграма сигналів RESET і PRESENCE, з яких завжди починається будь-який обмін даними.

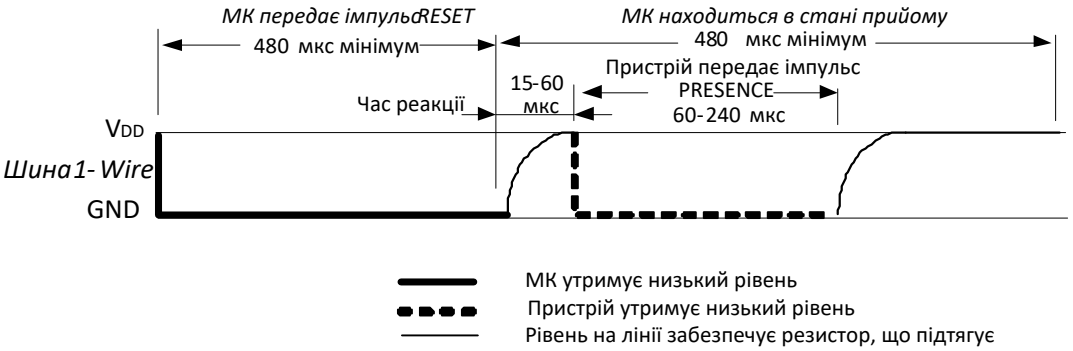

Рисунок 2.32 — Діаграма сигналів RESET і PRESENCE

Імпульс RESET формує МК, що переводить в низький логічний рівень шину 1-Wire і втримує її в цьому стані мінімум 480 мікросекунд. Далі МК повинний

«відпустити» шину. Через якийсь час, що залежить від ємності лінії й опору резистора, що підтягує, у лінії встановиться високий логічний рівень. Протокол 1-Wire обмежує цей час «релаксації» діапазоном від 15 до 60 мікросекунд, що і є визначальним для вибору резистора, що підтягує.

Виявивши імпульс RESET, ведений пристрій (Slave) приводить свої внутрішні вузли у вихідний стан і формує відповідний імпульс PRESENCE не пізніше 60 мікросекунд після завершення імпульсу RESET. Для цього пристрій переводить у низький рівень лінію і втримує її в цьому стані від 60 до 240 мікросекунд. Після цього пристрій так само «відпускає» шину.

Але після завершення імпульсу PRESENCE пристрою дається ще якийсь час для завершення внутрішніх процедур ініціалізації, таким чином, МК повинен приступити до будь-якого обміну з пристроєм не раніше, ніж через 480 мікросекунд після завершення імпульсу RESET.

Отже, процедура ініціалізації інтерфейсу, з якої починається будь-який обмін даними між пристроями, триває мінімум 960 мікросекунд, складається з передачі від МК сигналу RESET і прийому від пристрою сигналу PRESENCE. Якщо сигнал PRESENCE не виявлений – значить на шині 1-Wire немає готових до обміну пристроїв.

Обмін бітами інформації здійснюється певними тайм-слотами. Тайм-слот – це певна, досить жорстко лімітована за часом послідовність зміни рівнів сигналу в лінії 1-Wire. Розрізняють 4 типи тайм-слотів: передача «1» від МК, передача «0» від МК, прийом «1» від Slave-пристрою, прийом «0» від Slave-пристрою.

Будь-який тайм-слот завжди починає МК шляхом переведення шини 1-Wire у низький логічний рівень. Тривалість будь-якого тайм-слоту повинна перебувати в межах від 60 до 120 мікросекунд. Між окремими тайм-слотами завжди повинен передбачатися інтервал не менший 1 мікросекунди.

Тайм-слоти передачі відрізняються від тайм-слотів прийому поводженням МК: при передачі він тільки формує сигнали; при прийманні, крім того, ще й опитує рівень сигналу в лінії 1-Wire. Рис. 2.33 демонструє часові діаграми тайм-

слотів всіх 4-х типів: угорі показані тайм-слоти передачі від МК, унизу – прийому від Slave-пристрою.

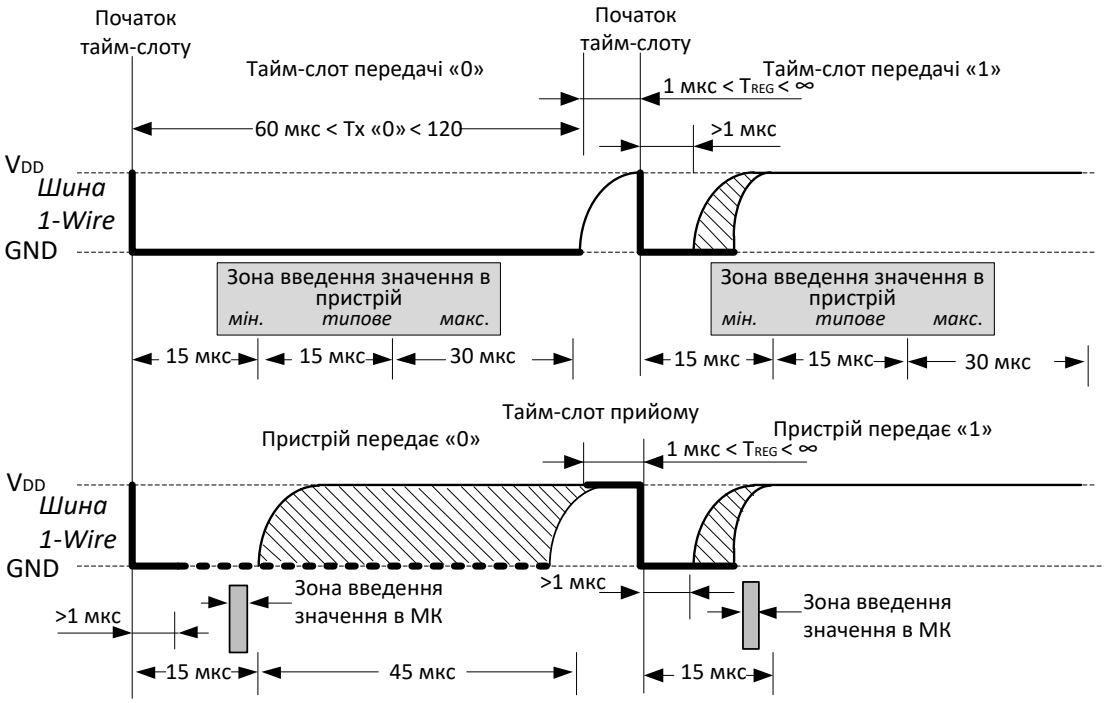

Рисунок 2.33 — Часові діаграми тайм-слотів на лінії 1-Wire

Кожний пристрій 1-Wire має унікальний ідентифікаційний 64-бітовий номер, який програмується на етапі виробництва мікросхеми.

На мережевому рівні відбувається адресація елементів на однопроводовій шині. Перебуваючи на цьому рівні, всі Slave пристрої очікують одну з команд адресації. Така команда повинна вибрати один або кілька пристроїв, з якими Master буде працювати на транспортному рівні.

Після того, як МК видасть команду READ ROM (0×33), від пристрою надійде 8 байтів його власної унікальної адреси — МК повинен їх прийняти. Будь-яка процедура обміну даними з пристроєм повинна бути завершена повністю або перервана посиланням сигналу RESET.

Якщо відправлено команду MATCH ROM (0×55), то після неї МК повинен передати так само й 8 байтів конкретної адреси пристрою, з яким буде здійснюватися наступний обмін даними. За цією командою кожний пристрій порівнює передану адресу зі своєю власною. Всі пристрої, адреси яких не збіглися, припиняють аналіз і видачу сигналів у лінії 1-Wire, а пристрій, що

пізнав адресу, продовжує роботу. Тепер всі дані, що передані МК, будуть потрапляти лише до цього пристрою. Які саме дані треба послати в пристрій або одержати від нього після його адресації, залежить від конкретного пристрою.

Якщо пристрій один на шині, то можна прискорити процес взаємодії з ним за допомогою команди SKI ROM (0×CC). Отримавши цю команду, пристрій відразу вважає, що адреса збіглася, хоча ніякої адреси для цієї команди не треба. Деякі процедури не вимагають прийому від пристрою ніяких даних, у цьому випадку команду SKI ROM можна використовувати для передачі деякої інформації відразу всім пристроям.

Всі мікросхеми мережі MicroLAN переходять на транспортний рівень після будь-якої команди мережного рівня. Набір команд транспортного рівня для різних мікросхем дещо відрізняється. Основні команди транспортного рівня — це команди «Запис пам'яті» і «Читання пам'яті».

Структура індивідуального коду, записаного в спеціальний ПЗП будь-якої мікросхеми, що підтримує інтерфейс 1-Wire, умовно зображена на рис.2.34. Код складається з трьох основних елементів:

Рисунок 2.34 — Структура індивідуального коду мікросхеми 1 -Wire

Перші 8 бітів, починаючи з молодшого розряду, – це груповий код мікросхеми. Груповий код однозначно визначає її вид. Наступні 48 бітів – це унікальний серійний номер мікросхеми. Останні 8 бітів – це контрольна сума (CRC).

Контрольна сума або CRC — це байт, значення якого передається найостаннішим і обчислюється за спеціальним алгоритмом на основі значення

усіх семи попередніх байтів. Алгоритм підрахунку такий, що якщо усі байти, що передаються-приймаються, без спотворень, то прийнятий байт контрольної суми співпадає з розрахованим в МК значенням. Тобто при реалізації програмного алгоритму обміну інформацією необхідно при передачі і прийомі байтів підраховувати їх контрольну суму за певним алгоритмом, а потім або передати отримане значення, або порівняти розрахункове значення з отриманим значенням CRC. Тільки при збігу обох CRC МК вважає прийняті дані достовірними. Інакше продовження обміну неможливе.

Організація обміну між ПК та МК по інтерфейсу USB

Для зв'язку з платою Arduino можна використовувати спеціальну програму моніторингу послідовного порту (Serial Monitor), вбудовану в програмне забезпечення Arduino. Монітор послідовної шини відображає дані, що посилаються в плату Arduino через віртуальний послідовний порт за допомогою USB. Для відправки даних вибирається швидкість передачі зі списку, відповідна значенню Serial.begin в скетчі. Потім необхідно ввести текст і натиснути кнопку Send або Enter.

Плата Arduino Nano має один послідовний порт (також відомий як UART або USART): Serial, що використовується для зв'язку з комп'ютером через USB. Він пов'язаний з цифровими виводами 0 (RX) і 1 (TX). Таким чином, під час роботи послідовного порту, порти D0 та D1 не можуть використовуватися як цифрові входи або виходи.

Для забезпечення зв'язку плати Arduino з комп'ютером або іншими пристроями використовується клас Serial. Клас Serial містить близько 20 функцій. Функції для роботи з послідовним портом плати Arduino:

if (Serial):

- параметри — немає;
- значення, що повертаються — boolean: повертає true, якщо вказаний послідовний порт готовий до роботи;

– опис: дозволяє перевірити готовність послідовного порту.

Serial.begin(speed). Параметри:

– speed: швидкість в бітах на секунду (бодах);

– опис: задає швидкість передачі даних по послідовному інтерфейсу в бітах в секунду (бодах). Для взаємодії з комп'ютером слід використовувати одну з попередньо встановлених швидкостей обміну: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200.

Serial.end ():

– параметри — немає;

– значення, що повертаються — немає.

– опис: функція розриває послідовний зв'язок, після чого виводи Rx і Tx знову можна використовувати як виводи загального призначення. Для відновлення послідовного з'єднання необхідно використовувати функцію *Serial.begin ().*

Serial.available():

– параметри — немає;

– значення, що повертаються: кількість байт, доступних для зчитування;

– опис: повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту. Під символами розуміються дані, які вже прийняті і зберігаються в послідовному приймальному буфері (який може зберігати максимум 64 байта).

Serial.read():

– параметри — немає;

– значення, що повертаються: перший байт прийнятих даних (або -1, якщо таких нема);

– опис: зчитує дані, що надходять по послідовному інтерфейсу.

Serial.print(val) / Serial.print(val, format). Параметри:

– val: значення, яке необхідно вивести (будь-який тип даних);

– format: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою). BIN (двійкова система), OCT (восьмерична система), DEC (десяткова система), HEX (шістнадцятирічна система). Для числа з плаваючою точкою цей параметр визначає кількість десяткових знаків після коми;
– значення, що повертаються: кількість виведених байт. Зчитування цього значення не є обов'язковим;
– опис: функція виводить через послідовний порт заданий ASCII текст у вигляді, зрозумілому для людини. При виведенні числа кожній його цифрі відповідає один ASCII-символ. Дробові числа теж виводяться у вигляді ASCII-цифр, при цьому після коми за замовчуванням залишається два десяткових знака. Байти виводяться у вигляді окремих символів, а символи та рядки виводяться без змін – «як є».

Приклад:

```
Serial.print("Hello world.") - виведе "Hello world."
Serial.print('N') - виведе "N"
Serial.print(78) - виведе "78"
Serial.print (78, BIN) - виведе "1001110"
Serial.print (78, OCT) - виведе "116"
Serial.print (78, DEC) - виведе "78"
Serial.print (78, HEX) - виведе "4E"
Serial.print(1.23456) - виведе "1.23"
Serial.println (1.23456, 0) - виведе "1"
Serial.println (1.23456, 2) - виведе "1.23"
Serial.println (1.23456, 4) - виведе "1.2346"
```

Serial.println (val) Serial.println (val, format). Параметри:

– val: значення, яке необхідно вивести (будь-який тип даних);
– format: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою);
– значення, що повертаються: кількість виведених байт;

– опис: виводить через послідовний порт ASCII-текст в зрозумілому для людини вигляді з символами повернення каретки (ASCII 13 або '\ r') і нового рядка (ASCII 10 або 'n'). Ця команда має такі ж форми, як і *Serial.print ()*.

Приклад коду для роботи з послідовним портом

```
int incomingByte = 0; void setup() {
    Serial.begin(9600);
}
void loop() {
if (Serial.available() > 0) { incomingByte = Serial.read();
    Serial.print("I received: "); Serial.println(incomingByte,
    DEC);
    }
}
```