МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ

Кафедра інформаційних технологій

РОЗРОБЛЕННЯ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ВИСОТОМІРА ДЛЯ БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ (ДРОНА) НА МЕМС-ДАВАЧІ BMP180

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Яна ХАРЧИНА

Науковий керівник:
доцент, кандидат технічних наук
Ігор ФАРМАГА

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім'я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____**Олег ЗАЧЕК**_
(підпис)

Львів
2026

СПИСОК УМОВНИХ СКОРОЧЕНЬ

AVR – архітектура 8-бітних мікроконтролерів, компанії Atmel.

АЗ/ПЗ – відповідно, апаратна частина (технічне оснащення) та програмна складова системи.

АЦП (Analog-to-Digital Converter) – перетворювач, що трансформує аналоговий сигнал у цифровий код.

МК (Мікроконтролер) – інтегральна схема, для керування електронними пристроями.

ЕК – електрична схема;

САПР (Computer-Aided Design) – комплекс програм для автоматизації процесів проектування та конструювання.

РКД/LCD – дисплеї відображення інформації на основі рідких кристалів.

ШІМ (Pulse-Width Modulation) – метод керування потужністю сигналу шляхом зміни тривалості імпульсів при незмінній частоті.

ОЗП (RAM) – оперативна пам’ять або оперативний запам’ятовуючий пристрій (англ. Random Access Memory, RAM);

ПЗП (ROM) – (англ. read-only memory) постійний запам’ятовуючий пристрій. Незалежна пам’ять, використовується для зберігання масиву незмінних даних;

EEPROM – постійна пам’ять з можливістю електричного стирання та повторного програмування окремих байтів.

Flash- напівпровідникова пам’ять, яка зберігає записану інформацію при відключенні живлення та підтримує багаторазове перезаписування.

SPI – (англ. Serial Peripheral Interface, SPI bus) - послідовний периферійний інтерфейс, шина SPI) - послідовний синхронний стандарт передачі даних в режимі повного дуплексу, призначений для високошвидкісного обміну даними мікроконтролерів і периферії;

I^2C – двопровідна шина для зв’язку, що використовує лінії даних (SDA) та синхронізації (SCL).

АНОТАЦІЯ

Харчина Я., Фармага І.В. (керівник). Розроблення апаратного і програмного забезпечення цифрового висотоміра для безпілотного літального апарату (дрона) на мемс-давачі bmp180. Бакалаврська кваліфікаційна робота. – Львівський державний університет внутрішніх справ, Львів, 2026.

У дипломній роботі реалізовано програмно-апаратний комплекс для взаємодії з цифровим сенсором BMP180. Система забезпечує зчитування та обробку даних про атмосферний тиск і температуру середовища, а також розрахунок абсолютної та відносної висоти. Пристрій вимірює навколишню температуру, атмосферний тиск, висоту над рівнем моря (альтитуду), відносну висоту (режиим висотоміра), контролює зміну тиску та видає ймовірний прогноз про можливі опади і хмарність та виводить виміряні значення на рідкокристалічний дисплей.

Передбачено конвертацію одиниць вимірювання у градусах Цельсія, Фаренгейта, або Кельвіна. Атмосферний тиск можна виводити у Паскалях, гекто Паскалях, міліметрах ртутного стовпчика, висота виводиться в метрах. Програмно-апаратний комплекс має два режими роботи – вимірювання атмосферного тиску та вимірювання висоти. Про вибір режиму вказують відповідні світлодіоди. Цифровий висотомір має вбудоване меню, що дає змогу користувачу налаштувати режими роботи.

В роботі спроектовано принципову електричну схему пристрою і його модель в САПР Proteus VSM. Виконано комп’ютерне моделювання та налагодження роботи висотоміра в програмному пакеті Proteus VSM (ISIS). Розроблено алгоритм роботи цифрового висотоміра і програмне забезпечення на мові С в середовищі розробки вбудованого ПЗ для МК сімейства AVR Code Vision AVR.

Ключові слова: цифровий барометр-висотомір, сенсор BMP180, МК сімейства AVR, символьний LCD-дисплей, Proteus VSM, мова програмування С, вбудовані системи (embedded systems), CodeVisionAVR.

ABSTRACT

Harchyna Ya., Farmaga I. (Supervisor). Development of hardware and software for a digital altimeter for an unmanned aerial vehicle (drone) based on the BMP180 MEMS sensor. Bachelor's thesis. – Lviv State University of Internal Affairs, Lviv, 2026.

The thesis implements a hardware-software complex for interacting with the BMP180 digital sensor. The system provides reading and processing of atmospheric pressure and ambient temperature data, as well as the calculation of absolute and relative altitude. The device measures ambient temperature, atmospheric pressure, altitude above sea level, and relative altitude (altimeter mode). It also monitors pressure changes, provides a likely forecast of potential precipitation and cloud cover, and displays the measured values on a liquid crystal display.

The system features unit conversion for temperature in Celsius, Fahrenheit, or Kelvin. Atmospheric pressure can be displayed in Pascals (Pa), hectopascals (hPa), or millimeters of mercury (mmHg), while altitude is displayed in meters. The hardware-software complex operates in two modes: atmospheric pressure measurement and altitude measurement. Corresponding LEDs indicate the selected mode. The digital altimeter includes an integrated menu that allows the user to configure the operating modes.

As part of the project, the electrical schematic diagram and a functional model were designed in Proteus VSM. Computer simulation and debugging of the altimeter were performed using the Proteus ISIS software package. The operational algorithm and software were developed in the C programming language within the CodeVisionAVR integrated development environment for AVR microcontrollers.

Keywords: digital barometer-altimeter, BMP180 sensor, AVR family MCU, character LCD, Proteus VSM, C programming language, embedded systems, CodeVisionAVR.

ЗМІСТ

ВСТУП

Сучасний прогрес у сфері мікроелектроніки, зокрема поява доступних мікроконтролерів та різноманітних датчиків, відкрив широкі можливості для проектування бюджетних програмованих комплексів. Такі системи дозволяють із високою точністю фіксувати різноманітні фізичні показники, як-от абсолютну та відносну висоту, показники атмосферного тиску та температуру навколишнього середовища. Фундаментальною основою для визначення висоти є її взаємозв'язок із барометричним тиском, який також коливається залежно від поточних метеоумов. Оскільки погода визначається динамічним станом атмосфери у конкретній точці та часовому проміжку, для коректного розрахунку відносної висоти необхідно враховувати якісні та кількісні метеорологічні дані. Ключовими показниками тут виступають температура та тиск, адже саме коливання останнього безпосередньо впливають на формування хмарності та ймовірність опадів. Прогнозування погодних змін завжди було пріоритетом для людини, що стимулювало створення приладів для моніторингу навколишнього середовища.

Мета даної роботи – є проектування апаратного та вбудованого програмного забезпечення цифрового висотоміра для безпілотного літального апарату (БПЛА), який вимірює висоту та метеорологічні параметри, а їх значення аналізує і результати аналізу виводить на символьний індикатор.

В основі апаратної архітектури приладу лежить мікроконтролер AVR ATmega128 від компанії Atmel. Вибір саме мікроконтролерної бази (МК) обумовлений тим, що вони є технологічним ядром більшості сучасних засобів вимірювання. З позиції інженерної розробки, використання МК суттєво прискорює та здешевлює процес впровадження складних алгоритмів. Завдяки високому рівню інтеграції периферійних вузлів безпосередньо на кристалі, МК здатен ефективно керувати зовнішніми модулями та обробляти вхідні сигнали при мінімальній кількості додаткових компонентів. Це сприяє не лише оптимізації термінів проектування, а й зменшенню габаритів та

кінцевої вартості готового виробу. Для реалізації поставленої мети необхідно виконати науково-технічні завдання:

- Проєктування архітектури цифрового висотоміра для безпілотного літального апарату із застосуванням такої елементної бази:
- мікроконтролер Atmega128;
- символьний рідкокристалічний модуль 16×2 (WH1602B-YGK-CTK);
- Розробка прототипу пристрою в середовищі САПР Proteus.
- Формування алгоритму функціонування системи.
- Написання програмного коду для забезпечення коректного зчитування даних із датчика тиску BMP180.
- Створення програмних модулів для виводу показників на дисплей.
- Перевірка та аналіз роботи спроєктованої моделі шляхом моделювання в програмному комплексі Proteus ISIS.

Об’єкт дослідження – проєктування цифрового висотоміра для безпілотного літального апарату.

Предмет дослідження – моделі та засоби проєктування цифрового висотоміра.

Методи досліджень. Реалізація мети дослідження базується на застосуванні системного підходу та комплексу наукових методів. Було опрацьовано фахову літературу, це дозволило виявити ключові напрямки у розвитку технологій моніторингу. Розробка технічних рішень ґрунтується на використанні моделювання, системного аналізу та синтезу, а також застосуванні методів дедукції, індукції, та структурного методів.

Структура роботи. Структурно робота включає вступну частину, чотири основні розділи, загальні висновки, перелік використаної літератури та додатки. Пояснювальна записка містить 81 сторінку основного тексту, проілюстрованого 60 рисунками та 15 таблицями. Інформаційну базу дослідження складають 9 бібліографічних джерел. Повний обсяг роботи, враховуючи 2 додатки, становить 91 сторінка.

РОЗДІЛ 1.
СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Основні метеорологічні величини та їх вимірювання

При використанні датчиків тиску для безконтактного визначення висоти ключовим фактором, що впливає на похибку вимірювань, є стан атмосфери. Метеорологічні величини – це сукупність фізичних показників, таких як атмосферний тиск, температурні показники, рівень вологості, а також вектор руху повітряних мас (швидкість і напрямок вітру) та ступінь хмарності.

Атмосферні явища – це динамічні фізичні процеси (опади у вигляді дощу чи снігу, грозові розряди, оптичні ефекти типу веселки або полярного сяйва), що зумовлюють різкі коливання стану повітряного середовища.

Погода – це фіксація поточного стану метеорологічних величин та явищ у конкретній локації в певний момент часу.

Клімат – це сталий режим погодних умов, характерний для певної місцевості, що базується на результатах багаторічних спостережень.

Метеорологічними величинами є:

Температура, яка відображає ступінь теплової енергії об'єкта (повітря, води чи ґрунту) та характеризує його тепловий стан. Температурний режим атмосфери є надзвичайно динамічним.

Біля земної поверхні температура повітря змінюється у широкому діапазоні: найбільші її значення, що зафіксовані на сьогодні , це +60 °C (у тропіках) і близько -90 °C в антарктичних регіонах . Зміна температури з висотою має нелінійний характер. Спочатку вона знижується до висоти 10 - 15 км, потім зростає на висотах до 50-60 км, а згодом знову змінюється падінням.

Для фізичних розрахунків і технічних вимірювань використовують дві основні системи:

- Шкала Цельсія (t °C): Найбільш розповсюджена у побуті та техніці. За опорні точки взято фазові переходи води при нормальному тиску (1013 гПа): 0 °C — замерзання, 100 °C — кипіння.
- Шкала Кельвіна (T, K): Система відліку абсолютної температури. Точка 0 К (абсолютний нуль) відповідає стані повної відсутності теплового руху молекул, що за Цельсієм становить приблизно -273,1 °C.

Оскільки величина одного градуса в обох системах ідентична (1 К = 1 °C), значення за Кельвіном завжди є додатним.

Для конвертації значень із градусів Цельсія у Кельвіни використовується наступне співвідношення:

$$TK = t^{\circ}C + 273{,}1 \tag{1.1}$$

Окрім загальноприйнятих метричних систем, у деяких країнах (зокрема в США) для вимірювання температури досі використовують шкалу, розроблену Даніелем Габріелем Фаренгейтом у 1724 році. Ключові параметри шкали Фаренгейта (°F): За цією системою чиста вода замерзає при значенні +32 °F, а закипає при +212 °F. Ціна поділки: Один градус за Фаренгейтом визначається як 1/180 частка температурного інтервалу між фазовими переходами води (таненням та кипінням).

Щоб перевести значення температури з градусів Цельсія (t °C) у градуси Фаренгейта, використовують таку формулу:

$$t^{\circ}C = 5/9 \cdot (t^{\circ}F - 32) \tag{1.2}$$

Градус Фаренгейта менший за градус шкали Цельсія, нулі цих шкал не збігаються. Нуль шкали Фаренгейта рівний температурі -17,8°C за стоградусною шкалою Цельсія. Атмосферний тиск визначається як сила, що діє з боку повітряного стовпа на одиницю земної поверхні. Сумарна маса газової оболонки планети, яка створює цей тиск, колосальна і становить приблизно $5{,}15 \times 10^{15}$ тонн.

Починаючи з XVII століття, тиск традиційно визначали за допомогою барометрів, вимірюючи висоту ртутного стовпчика в лінійних одиницях (міліметрах або дюймах).

З розвитком обчислювальної метеорології та методів прогнозування стало очевидним, що використання міліметрів ртутного стовпчика (мм. рт. ст.) є недоцільним. Така міра є суто візуальною і не відображає безпосередню фізичну природу тиску як сили, що робить її незручною для сучасних математичних моделей і технічних розрахунків. За ініціативи норвезького вченого В. Б'єркнеса в метеорологічну практику було впроваджено мілібар (мбар). Мілібар: Визначається як тиск, що відповідає силі у 1000 дин на площу в 1 см². (При цьому дина — це фізична сила, що надає об'єкту масою 1 г прискорення 1 см/с²). В міжнародній системі одиниць (СІ) тиск вимірюють в Паскалях (Па). Паскаль (Па): Основна одиниця в системі SI. Один Паскаль представляє силу в 1Ньютон, яка рівномірно діє на поверхню площею 1 м². У наукових розрахунках прийнято розрізняти два тиски атмосфери:

1. Нормальний тиск: Показник, зафіксований на рівні моря (паралель 45°) за умови нульової температури (0 °C). Він дорівнює 1013,25 мбар (або 760 мм рт. ст.).
2. Стандартний тиск: Спрощене значення для технічних нормативів, що становить 1000 гПа (або 750 мм рт. ст.).

Різні одиниці для вимірювання тиску: мм рт.ст, гПа, мбар:

1 мм. рт.ст. = 1,333 гПа

1 гПа = 100 Па = 1 мбар

1 гПа = 3/4 = 0,75 мм рт.ст

[P] = [H/m] = [Па],

Атмосферний тиск не є сталим показником; його коливання безпосередньо впливають на метеорологічні умови, зокрема на хмарність, вологість та температурний режим. Для людського організму найбільш сприятливим вважається діапазон від 750 до 760 мм рт. ст. Протягом доби

спостерігається певна циклічність – два максимуми (у ранкові та вечірні години) та два мінімуми (після полудня та після півночі). У зимовий період, через більшу щільність та вагу холодних повітряних мас, показники тиску зазвичай вищі, ніж улітку, коли прогріте повітря стає легшим. Ключовим фактором, що визначає значення тиску, є висота місцевості над рівнем моря. При підйомі вгору щільність повітря та висота атмосферного стовпа, що тисне на поверхню, зменшуються, що призводить до закономірного зниження барометричних показників. Цю властивість використовують для виготовлення висотомірів. Спостереження за динамікою тиску дозволяє прогнозувати зміну погодних умов: зниження тиску зазвичай передує погіршенню погоди та випаданню опадів; стабільне зростання тиску є ознакою встановлення сухої та ясної погоди.

1.2. Огляд існуючих підходів до розробки цифрових барометрів та висотомірів

Цифрові барометричні пристрої на сьогодні є критично важливими компонентами автоматичних метеостанцій, авіаційних висотомірів та різноманітних систем моніторингу навколишнього середовища. Окрім професійного обладнання, такі сенсори широко застосовуються у доступних побутових вимірювальних приладах. Дослідження наукової літератури та сучасних інтернет-ресурсів підтверджує значний інтерес до проектування засобів вимірювання атмосферного тиску та альтитуди (висоти). Аналіз існуючих рішень дозволяє виділити такі особливості:

- Мікроконтролерне керування: Більшість сучасних проектів базується на використанні мікроконтролерів, що дозволяє автоматизувати процес збору та обробки даних.
- Дистанційний моніторинг: Окрему категорію складають системи, оснащені GSM-модулями. Такі рішення забезпечують передачу показників у реальному часі через мережі стільникового зв’язку, що є незамінним для віддалених об’єктів.

- Різноманітність сенсорів: Для фіксації тиску застосовуються як класичні барометри-анероїди, так і високотехнологічні цифрові та аналогові напівпровідникові датчики.

Використання сучасної елементної бази дозволяє створювати пристрої, що поєднують високу точність вимірювань із компактністю та низьким енергоспоживанням.

Види барометрів. Ртутний барометр – це класичний прилад для вимірювання атмосферного тиску, де робочим тілом виступає ртуть. Його конструкція та функціонування базуються на наступних елементах: пристрій складається з герметичної скляної колби (трубки), один кінець якої запаяний. Ця трубка заповнюється ртуттю і занурюється відкритим кінцем у резервуар (чашу), також наповнений цією рідиною. Рівень тиску визначається за допомогою спеціальної шкали, що нанесена безпосередньо на скляну трубку або закріплена поруч із нею.Під впливом атмосферного тиску, що тисне на поверхню ртуті в чаші, висота металевого стовпчика в трубці змінюється. При зростанні тиску ртуть піднімається, а при зниженні — опускається, фіксуючись навпроти відповідної позначки на шкалі. На Рис.1.1. показано ртутний барометр.

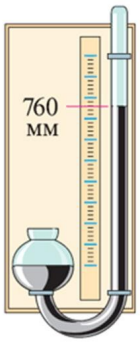


Рис. 1.1. Ртутний барометр

Барометр-анероїд – це прилад для визначення атмосферного тиску безвикористання рідини. Його робота базується на механічній деформації чутливого елемента під дією повітряних мас. Конструктивні елементи:

- Анероїдна капсула: Герметична коробка з гофрованими стінками, виготовлена із загартованої сталі або нікелево-срібного сплаву. Усередині неї створено вакуум (сильне розрідження).
- Пружний елемент (пружина): Протидіє зовнішньому тиску, запобігаючи повному стисненню капсули.
- Передавальний важільний механізм: Трансформує мікроскопічні рухи стінок коробки у рух стрілки.
- Шкала та вказівна стрілка: Візуалізують показники тиску, зазвичай градуйовані у відповідності до значень ртутного стовпа.

Функціонування приладу засноване на реакції металевої капсули на зовнішній тиск. При зростанні тиску стінки коробки стискаються, долаючи опір пружини. При зниженні тиску пружина розпрямляє стінки капсули. Ці механічні коливання через систему важелів передаються на стрілку, яка переміщується вздовж циферблата. Завдяки відсутності токсичної ртуті, компактності та портативності, анероїди є найбільш поширеними у побуті та польових умовах. Їхні корпуси можуть виготовлятися з металу, міцного пластику або деревини (для декоративних інтер'єрних моделей). З часом пружні властивості металу можуть змінюватися, тому для підтримання точності в сучасних приладах передбачені спеціальні компенсатори. Повірка анероїдів здійснюється шляхом порівняння з еталонним ртутним барометром за такими параметрами: Шкалова поправка: Усуває похибки, що виникають через нерівномірну реакцію механізму на різних ділянках діапазону. Температурна компенсація: Нівелює вплив термічного розширення металевих деталей на точність вимірів. Додаткова поправка: Враховує поступову зміну еластичності пружини та капсули внаслідок "старіння" металу. На Рис.1.2. зображено анероїди (безрідинні, механічні барометри).

Рис.1.2. Барометри - Анероїди

Електронний барометр можна вважати сучасною еволюцією класичного анероїда. У таких пристроях механічна деформація чутливого елемента трансформується в електричний сигнал. Цей сигнал проходить через аналого-цифрове перетворення, обробляється мікропроцесором і виводиться на дисплей у зручному для користувача форматі. Завдяки стрімкому розвитку мікроелектромеханічних систем (MEMS), на ринку з'явилися надкомпактні та високоточні датчики тиску. Їхні мініатюрні розміри дозволяють інтегрувати функціонал барометра та альтиметра у широкий спектр портативної електроніки: смартфони та планшети; смарт-годинники та фітнес-трекери; спеціалізовані портативні висотоміри.

В інженерних проектах та промислових рішеннях найчастіше використовують такі компоненти:

- MPX4115: П'єзорезистивний датчик, який формує на виході аналоговий сигнал, пропорційний атмосферному тиску.
- BMP085: Цифровий модуль, що забезпечує передачу даних через послідовний інтерфейс I^2C, що спрощує його підключення до мікроконтролерів.
- BMP180: Вдосконалена версія попередньої моделі. Він також працює по шині I^2C, але вирізняється покращеною точністю, меншим енергоспоживанням та стабільністю роботи, що робить його одним із найпопулярніших рішень для сучасних вбудованих систем.

Рис. 1.3. МЕМС датчики тиску BMP180, BMP085

Рис. 1.4. П'єзорезистивний датчик тиску MPX4115

Завдяки мініатюризації компонентів, сучасні датчики інтегруються в компактні гаджети, що дозволяє користувачам отримувати метеорологічні дані в реальному часі. На наведених ілюстраціях представлені приклади сучасних портативних пристроїв:

Рис. 1.5. Метеостанція TFA "SEASON"

Рис. 1.6. Метеостанція Explore Scientific Color

Рис. 1.7. барометр - Висотомір

Рис. 1.8. Висотомір УВ-57

Механічні барометри, що мають шкалу, за якою визначають висоту, називають висотомірами. Ці прилади застосовують в авіації та при сходженях в гори.

Рис. 1.9. Годинник з барометром

На Рис.1.5-1.9 зображено різні вироби цифрових метеостанцій, висотомірів, годинників з сенсорами тиску на МК.

1.3. Вимірювання метеопараметрів давачами та обробка їх значень

При використанні ртутних барометрів або класичних анероїдів отримання даних базується на суб'єктивному візуальному зчитуванні. Користувач фіксує положення стрілки або рівень рідини відносно нанесеній на корпус градуйованій шкалі. Напівпровідникові сенсори (на прикладі MPX4115): На відміну від механічних аналогів, робота з датчиком MPX4115 потребує обов'язкової програмної обробки сигналів. Функціонування сенсора MPX4115 ґрунтується на зміні електричних параметрів. Його вихідна напруга є безпосередньою функцією прикладеного тиску. Отже, для отримання реального значення тиску в одиницях кПа чи мм рт. ст., необхідно

перетворити отриману аналогову напругу за допомогою відповідного математичного алгоритму, що реалізується на базі мікроконтролера. Вихідна напруга давача тиску MPX4115 є функцією від тиску:

$$V_{\text{out}} = V_s(0,0009P - 0,095) \quad (1.3)$$

де V_s – напруга живлення давача. МК AVR мають вбудований аналого-цифровий перетворювач з роздільною здатністю 10 біт. Зміна наймолодшого біта зумовлена зміною вхідної напруги на 5 В/1023 = 0,00488 В. Зчитане значення коду N з регістра АЦП відповідає напрузі $V_{out} = N * 0,00488$ В. Отже, формула залежності вихідної напруги від виміряного тиску:

$$P = (N * 0,00488 + 0,095 * V_s)/(0,009 * V_s) \quad (1.4)$$

$$P = N * 0,108336 + 10,545 \text{ або } P = (V_{out} * 22,2) + 10,55 \text{ для } V_s = 5 \text{ В}$$

Похибка вимірювання тиску давачем MPX4115 в діапазоні температур 0° to 85°C становить 1,5%. Нижче наведено графік залежності вихідної напруги від тиску.

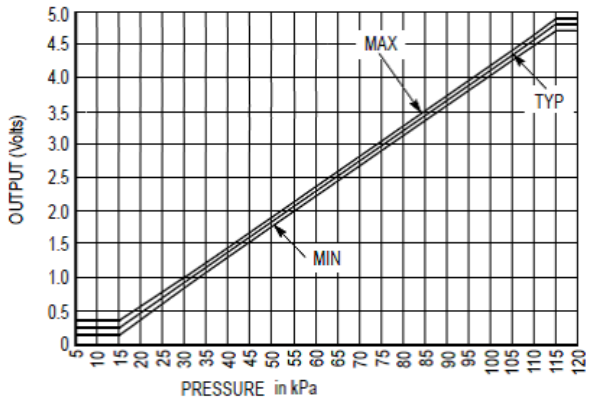


Рис. 1.10. Залежність вихідної напруги від тиску для MPX4115

Сучасні модулі BMP085 та BMP180 належать до категорії цифрових вимірювальних пристроїв, що забезпечують передачу даних через послідовний протокол I2C. Обидва компоненти спроєктовані для одночасної фіксації показників атмосферного тиску та температури навколишнього середовища. При цьому модель BMP180 є вдосконаленою версією, яка вирізняється вищою точністю вимірювань та покращеною стабільністю роботи. Давач BMP180 складається з п'єзорезистивного вимірювального елемента, АЦП, Керуючого логічного блоку з E2PROM та інтерфейсу – I2C. Пам'ять E2PROM містить176 біт індивідуальних калібрувальних даних.

Отримання достовірних даних за допомогою цих сенсорів вимагає реалізації складного математичного алгоритму. Процесор системи керування зчитує “сирі” значення з відповідних регістрів: регістр UP – дані тиску (16… 19 біт); регістр UT – дані температури (16 біт). Детальна послідовність операцій під час циклу вимірювання представлена на схемі алгоритму Рис.1.11.

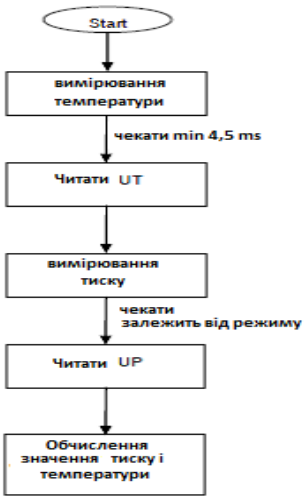


Рис. 1.11. Алгоритм вимірювання тиску датчиком BMP180

Кожен датчик BMP180 має унікальні калібрувальні коефіцієнти, які треба враховувати при обчисленні тиску і температури. Пам’ять E^2PROM BMP180 об’ємом 176 біт розбита на 11 слів по 16 біт, які містять 11 калібрувальних коефіцієнтів. Перед першим вимірюванням МК читає з E^2PROM ці дані. Назва та адреса коефіцієнтів наведена в Табл. 1.1.

Таблиця 1.1. Калібрувальні коефіцієнти BMP180

	BMP180 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

Для адаптації датчика до конкретних завдань розробнику доступні кілька варіантів роздільної здатності. Програмне керування режимами роботи сенсора здійснюється шляхом передачі числових значень від 0 до 3, що відповідають наступним рівням точності та енергоспоживання: 0 (Ultra Low Power): Режим мінімального споживання енергії. 1 (Standard): Стандартний режим вимірювання. 2 (High): Режим підвищеної точності. 3 (Ultra High Resolution): Режим максимальної роздільної здатності.

Процес математичного перетворення отриманих із сенсора цифрових кодів у реальні фізичні показники температури та атмосферного тиску описано в алгоритмі, який наведено на Рис. 1.12..

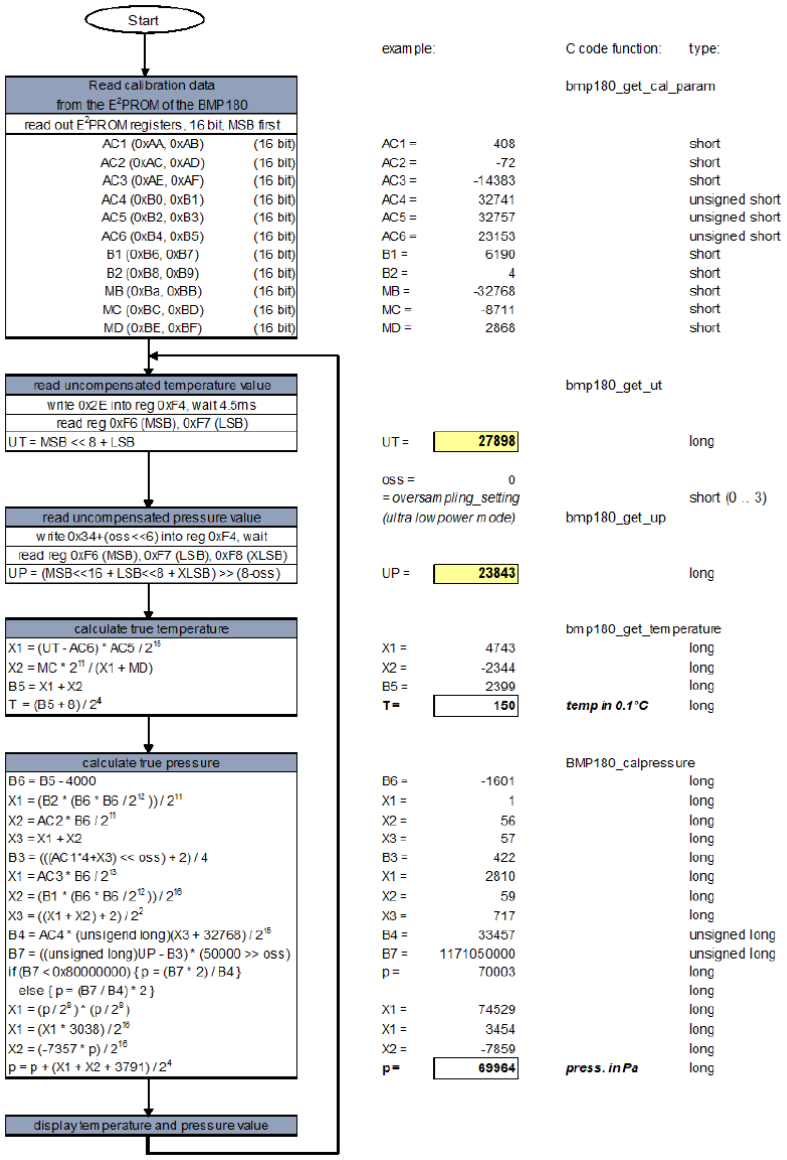


Рис. 1.12. Алгоритм обчислення значень температури та тиску

Для обчислення висоти над рівнем моря на основі даних, отриманих від сенсора BMP180, використовується барометрична формула. Розрахунок базується на порівнянні поточного показника тиску P із еталонним значенням P0, яке відповідає нормальному атмосферному тиску на нульовій відмітці (рівень моря) і становить 1013,25 гПа.

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right) \tag{1.5}$$

Зміна тиску на 1 hPa дорівнює зміні висоти на 8,43м.

На Рис.1.13 показано графік залежності тиску від висоти (над рівнем моря).

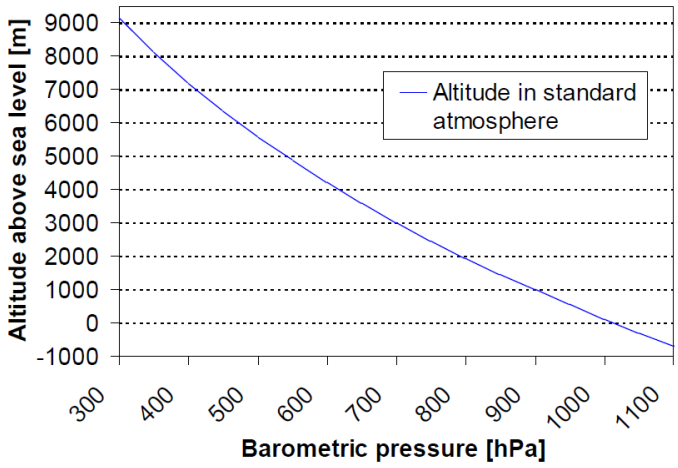


Рис. 1.13. Графік залежності тиску від висоти (над рівнем моря)

Точність у всьому діапазоні вимірювання датчиком BMP180: температури ±0,5 ºC, тиску ±0,12hPa, висоти ±1м.

РОЗДІЛ 2
ЗАСОБИ РОЗРОБКИ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ВИСОТОМІРА

2.1. Proteus VSM - автоматизована система проектування та моделювання пристроїв

Програмне забезпечення Proteus VSM від Labcenter Electronics є ефективним інструментом для розробки та аналізу електронних пристроїв. Система підтримує симуляцію роботи мікропроцесорної техніки, зокрема платформ Arduino та різних МК-модулів. Основна цінність ПЗ це можливість віртуального тестування електричних схем, що дозволяє проводити налагодження програмного коду та перевіряти працездатність апаратної частини без використання фізичних прототипів, застосовуючи вбудовані віртуальні прилади. Програмний комплекс Proteus VSM поєднує в собі інструменти для розробки схем та модуль ARES для трасування друкованих плат. Система має потужну базу моделей для популярних сімейств МК (від 8051 до ARM7), що забезпечує можливість комплексного проектування мікропроцесорних пристроїв. Симуляція в Proteus дозволяє ефективно відпрацювати логіку роботи ПЗ на етапі розробки. Якщо стандартна бібліотека не містить потрібного компонента, користувач може створити його власноруч або імпортувати SPICE-модель безпосередньо з офіційного ресурсу виробника електронних компонентів. Структура програмного комплексу Proteus базується на взаємодії двох ключових модулів:

1. ISIS – графічний редактор для побудови принципових електричних схем і проведення подальшої симуляції. Дані з цього модуля є основою для розробки топології в середовищі ARES. Останній забезпечує можливість автоматичного компонування деталей та трасування провідників.
2. ARES – спеціалізований інструментарій для проектування друкованих плат. Модуль оснащений системою автотрасування ELECTRA,

функцією автоматичного розміщення елементів та зручним менеджером бібліотек. Функціонал ARES дозволяє генерувати файли для виробництва: фотошаблони та керуючі програми для свердлильних верстатів з ЧПУ.

Окрім базових модулів, програмний пакет Proteus включає додаткові функціональні модулі:

- COMPIM – інструмент, що забезпечує взаємодію віртуальної моделі з фізичним послідовним COM-портом (UART) комп’ютера.
- USBCONN – спеціалізований модуль для організації зв’язку зі справжнім USB-інтерфейсом ПК.

САПР Proteus VSM є незамінною для попереднього проектування та верифікації пристроїв. Розробка програмного коду для симуляції може здійснюватися у різних середовищах (IDE), серед яких:

- WinAVR та CodeVisionAVR (орієнтовані на мікроконтролери архітектури AVR);
- ICC (підтримує AVR, msp430, а також ARM7);
- HiTECH (використовується для роботи з лінійками PIC та 8051).

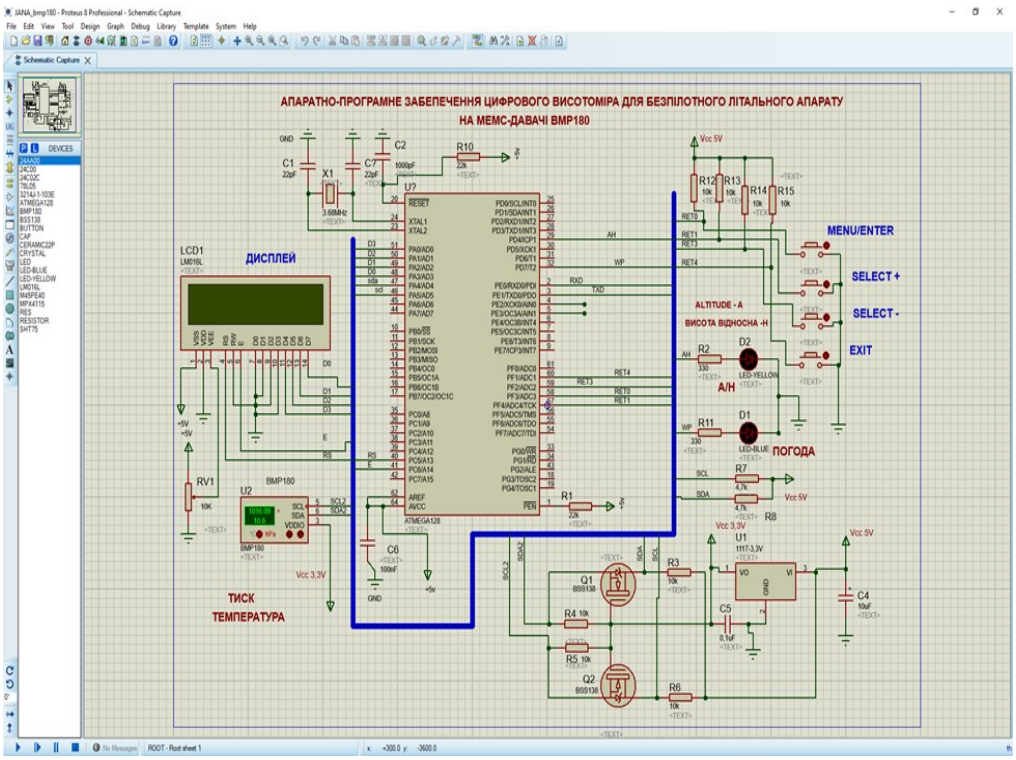


Рис. 2.1. Середовище Proteus ISIS

2.2. Мова С та інструментарій розробки ПЗ CodeVisionAVR для вбудованих систем на МК AVR

CodeVisionAVR – це потужне ПЗ для створення програмного коду. Воно орієнтоване для мікроконтролерів серії AVR компанії Atmel. При програмуванні в середовищі CodeVisionAVR можна використовувати наступні типи даних, які наведено в Табл.2.1.

Таблиця 2.1. Мова С для AVR. Типи даних

Тип	Розмір (біт)	Діапазон значень
char	8	-128…127
bit	1	0, 1
signed char	8	-128…127
unsigned char	8	0…255
int	16	-32768…32767
unsigned int	16	0…65535
short int	16	-32768…32767
signed int	16	-32768…32767
unsigned long int	32	0…4294967295
long int	32	-2147483648…2147483647
float	32	±1.175e-38…±3.402e38
double	32	±1.175e-38…±3.402e38
signed long int	32	-2147483648…2147483647

В середовищі CodeVisionAVR не можна використовувати як ідентифікатори наступні зарезервовані слова (Табл.2.2).

Таблиця 2.2. Зарезервовані слова

bit	enum	interrupt	switch
break	else	int	struct
case	extern	long	typedef
continue	for	short	void
char	flash	register	union
const	float	return	unsigned
double	if	sfrw	
default	funcused	signed	volatile
do	goto	sizeof	while
eeprom	inline	static	

До складу інструментарію CodeVisionAVR входить низка компонентів, що забезпечують повний цикл розробки програмного забезпечення для архітектур AVR та AVR32. Зокрема, пакет включає компілятор GNU GCC, який дозволяє працювати з мовами C, C++ та Objective-C, надаючи всі

необхідні засоби для програмування мікроконтролерів. Серед ключових елементів середовища варто виділити:

- AVR-LibC – спеціалізована бібліотека стандартних функцій для МК AVR;
- Programmers Notepad – інтегрований текстовий редактор для написання та редагування коду;
- GNU Debugger (GDB) – потужний засіб для пошуку помилок та налагодження програм.

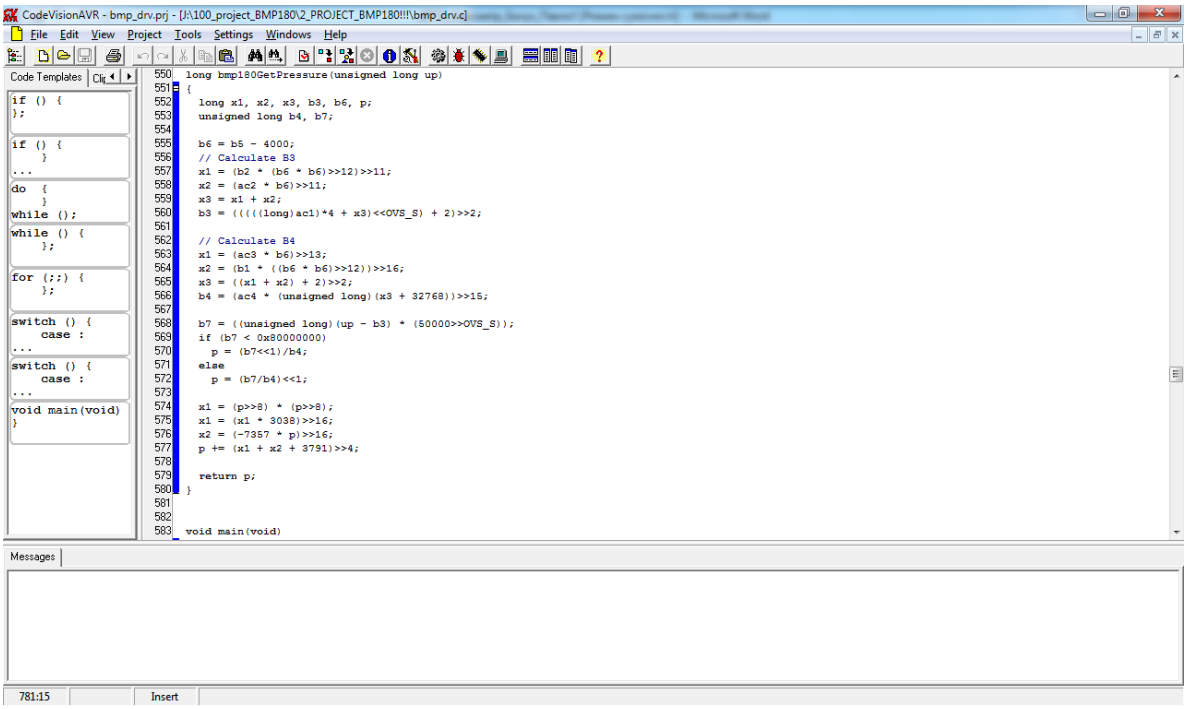


Рис. 2.2. Середовище програмування CodeVisionAVR

Середовище CodeVisionAVR вирізняється якісним документаційним супроводом та генерацією оптимізованого компактного коду. Програма базується на класичному синтаксисі мови С, що спрощує розробку. Важливою особливістю є можливість повної інтеграції з Atmel Studio. Спільне використання цих інструментів забезпечує ефективний цикл програмування та тестування мікроконтролерів серій Atmel, PIC та інших архітектур. Англомовний інтерфейс програми доповнюється наявністю навчальних прикладів та інструкцій, які можна знайти у робочій директорії `\cvavr\bin`. Пакет сумісний з усіма 32-розрядними версіями ОС Windows. Функціональні параметри CodeVisionAVR:

1. Операційне середовище: Будь-які версії MS Windows.
2. Мови програмування: Низькорівневий Assembler та високорівнева мова C.
3. Компілятор: Безкоштовний пакет GNU GCC.
4. Комплектація: Набір інструментів включає бібліотеку елементів avr-libc, редактор проектних файлів MFile та текстовий процесор Programmers Notepad.
5. Сумісність: Тісна інтеграція з AVR Studio.
6. Інтерфейс: Англомовний.

CodeVisionAVR сумісний із широким спектром 32-бітних операційних систем сімейства Windows (від версії 95 до Vista). Програмний комплекс забезпечує підтримку великої кількості апаратних засобів програмування, серед яких пристрої від Atmel (STK500, AVRISP, Dragon, JTAGICE), а також рішення від сторонніх виробників, як-от VTEC-ISP, STK200+, DT006 та інші. CodeVisionAVR має набір вбудованих бібліотек, що спрощують взаємодію з:

- Рідкокристалічними (LCD) індикаторами символьного типу.
- Інтерфейсами I2C (Philips) та SPI.
- Датчиками температури (зокрема серії LM75 та компонентами від Maxim/Dallas).
- Мікросхемами годинників реального часу (DS1307, PCF8583 тощо).
- Шиною 1-Wire.

Окремою перевагою є наявність автоматичного генератора коду (CodeWizard). Цей інструмент дозволяє миттєво створювати шаблони для ініціалізації периферії: налаштування портів I/O, таймерів, обробки переривань, роботи з АЦП, аналоговим компаратором та різними типами зовнішньої пам'яті.

РОЗДІЛ 3
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ВИСОТОМІРА

3.1. Вибір елементної бази для проектування цифрового висотоміра

Елементною базою для проектування цифрового висотоміра було вибрано:

- МК AVR (ATmega128);
- цифровий датчик тиску BMP180;
- символьний монохромний (РКД) 16x2 (WH1602B-YGK-CTK).

Опис МК ATmega128. Восьми-розрядні мікросхеми сімейства AVR від компанії Atmel здобули значну популярність на ринку вбудованих систем. Завдяки оптимізованій архітектурі, широкому набору інструкцій та високій продуктивності за умов низького енергоспоживання, ці МК залишаються актуальними для сучасних розробок. Оцінка балансу між вартістю, функціональними можливостями та обчислювальною потужністю зумовила вибір моделі ATmega128 як базового компонента проєкту.

Сімейство МК AVR базується на RISC-архітектурі, що забезпечує гнучкість та швидку обробку даних. Важливою перевагою є можливість багаторазового внутрішньосхемного програмування (до 10 тисяч циклів), що спрощує процес налагодження пристроїв. Лінійка Mega AVR орієнтована на складні обчислювальні завдання, з продуктивністю до 16 MIPS, великий об’єм Flash-пам’яті (до 128 Кб), а також вбудовані периферійні модулі, зокрема 10-бітний АЦП та незалежну пам’ять EEPROM і SRAM.

Мікроконтролери AVR мають кілька основних відгалужень, серед яких виділяються Classic AVR та Tiny AVR. Перші представляють стандартну лінійку з обчислювальною здатністю до 16 MIPS, об’ємом Flash до 8 Кб та SRAM до 512 байт. Другі – це компактні та бюджетні рішення, що зазвичай випускаються у 8-пінових корпусах. Важливою перевагою архітектури є програмна сумісність: код, написаний для менш потужних моделей, легко адаптується до продуктивніших чипів.

Одним із найбільш потужних представників є 8-бітний мікроконтролер ATmega128 – із можливістю внутрішньосхемної прошивки 128 Кб Flash-пам’яті. Його RISC-архітектура базується на 133 інструкціях, більшість із яких завершуються за один такт, що дозволяє досягти продуктивності 16 MIPS на частоті 16 МГц. Пристрій оснащений 32 регістрами загального призначення та має роздільну структуру пам’яті програм і даних. Надійність зберігання інформації забезпечується ресурсом Flash-пам’яті у 10 тис. циклів та EEPROM (4 Кб) у 100 тис. циклів.

ATmega128 має широкий спектр модулів: від подвійних 8-бітних та розширених 16-бітних таймерів до 8-канального 10-бітного АЦП. Для комунікації використовуються інтерфейси USART, SPI та JTAG, а для стабільної роботи передбачено сторожовий таймер (Watchdog) та систему Reset при коливаннях напруги, має шість каналів ШІМ з роздільною здатністю від 2 до 16 біт, зовнішні та внутрішні джерела переривань.

Контролер підтримує шість режимів енергозбереження та має 53 лінії вводу-виводу, що робить його універсальним для створення складних систем. МК виготовляється у корпусах: 64-піновий TQFP або MLF.

Живлення та частота: версія “L” працює при 2,7–5,5 В (до 8 МГц), стандартна версія – при 4,5–5,5 В (до 16 МГц).

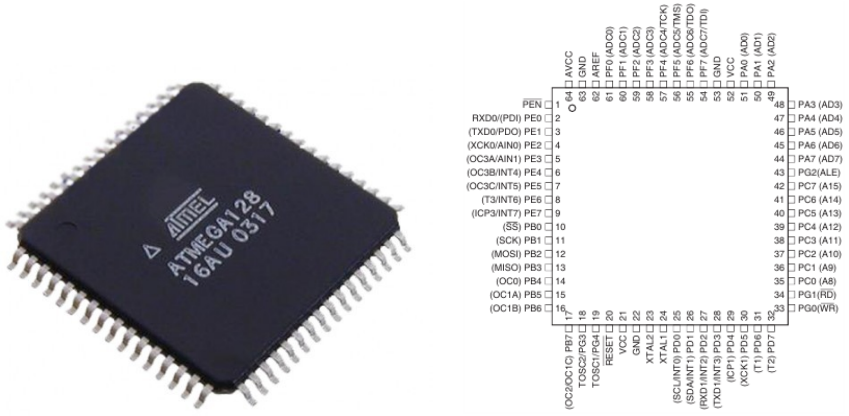


Рис. 3.1. МК AVR ATmega128 в корпусах TQFP і MLF

Основою архітектури AVR є вдосконалена система команд, що взаємодіє з 32 регістрами загального призначення. Пряме сполучення всіх регістрів з арифметично-логічним пристроєм (АЛП) дозволяє маніпулювати

двома незалежними регістрами протягом одного командного циклу. Така логіка побудови забезпечує значну перевагу в обчислювальній потужності: продуктивність AVR майже вдесятеро перевищує показники аналогічних мікроконтролерів із CISC-архітектурою.

Модель ATmega128/L оснащена 128 Кбайт Flash-пам'яті з підтримкою технології Read-While-Write (читання під час запису). Ресурсна база чипа також включає 4 Кбайт EEPROM та аналогічний об'єм оперативної пам'яті (SRAM). Периферійний набір складається з чотирьох таймерів-лічильників, інтерфейсів USART, I2C та SPI, а також 8-канального АЦП із розрядністю 10 біт. Безпеку роботи гарантує сторожовий таймер із власним тактовим генератором. Для оптимізації споживання енергії передбачено шість спеціальних режимів, зокрема "Idle", у якому зупиняється робота центрального процесора. Окремо виділено режим зниження шумів АЦП, де активними залишаються лише блоки перетворення та асинхронний таймер.

У стані “глибокого сну” (Power-down) мікроконтролер призупиняє роботу тактового генератора та більшості периферійних модулів, при цьому дані в регістрах залишаються незмінними. Вихід із цього режиму можливий лише за умови надходження сигналу зовнішнього скидання (Reset) або відповідного переривання. Режим очікування (Standby) відрізняється тим, що активним залишається лише один тактовий генератор, що дозволяє системі миттєво повернутися до виконання завдань. Завдяки здатності до швидкої активації, ATmega128 демонструє високу енергоефективність у динамічних режимах роботи. Для специфічних завдань передбачено розширене очікування (Extended Standby), де паралельно функціонують головний та асинхронний генератори. Запис програмного забезпечення у Flash-пам’ять реалізується двома способами: через послідовний інтерфейс SPI за допомогою стандартних технічних засобів або через завантажувач (Boot Loader). Архітектурні особливості RISC-ядра та гнучкість налаштувань пам'яті роблять цей контролер оптимальним рішенням для інтегрованих

систем автоматизації. Зображення внутрішньої структури МК наведено на Рис. 3.2.

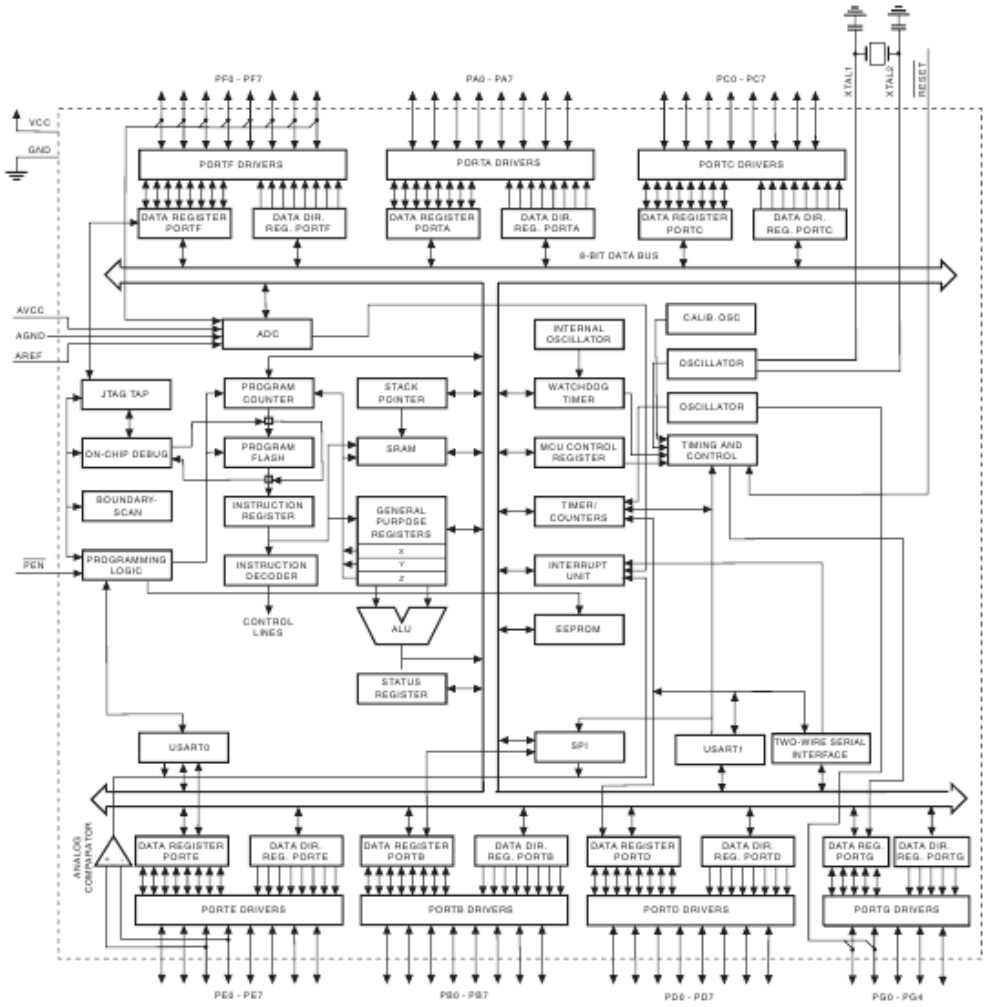


Рис. 3.2. Внутрішня структура МК AVR ATmega128

Аналого-цифровий перетворювач (АЦП) ATmega128. Для обміну даними мікроконтролер використовує універсальні порти вводу/виводу, які переважно оперують бінарними станами. При 5-вольтовому живленні рівень логічної “1” відповідає високому потенціалу на виводі, а логічний “0” - низькому. Оскільки реальні фізичні процеси часто вимагають вимірювання напруги, що змінюється безперервно, стандартних цифрових входів недостатньо. Саме для обробки аналогових сигналів, амплітуда яких варіюється від нуля до опорної напруги живлення, у мікроконтролері ATmega128 реалізовано модуль АЦП (Рис.3.3.).

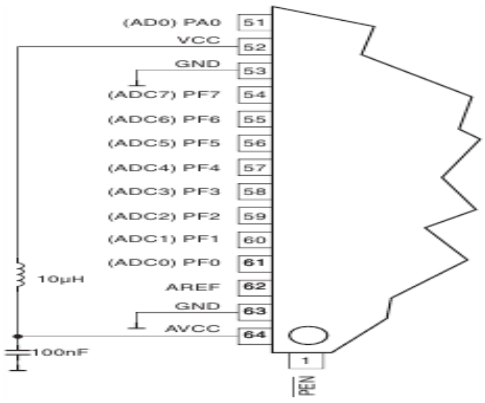


Рис. 3.3. Входи АЦП МК ATmega128

Функціонування аналого-цифрового перетворення в мікроконтролері ATmega128 доступне через лінії порту F (PORTF). Програмне керування процесом перетворення та зчитування даних здійснюється за допомогою наступного набору регістрів:

- ADMUX – призначений для конфігурування каналів мультиплексора та вибору опорної напруги.
- ADCSRA – основний регістр контролю, що відповідає за активацію перетворювача, вибір частоти дискретизації та моніторинг стану готовності.
- ADCL та ADCH – пара регістрів, що містять результат завершеного циклу перетворення (молодший та старший байти відповідно).
- SFIOR – регістр спеціальних функцій, для налаштування додаткових режимів роботи периферії, зокрема автозапуску АЦП.

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таблиця 3.1. Конфігурація джерела опорної напруги АЦП (регістр ADMUX)

REFS1	REFS0	Варіант вибору опорної напруги (Vref)
0	0	AREF: зовнішня опорна напруга, внутрішнє джерело вимкнено.
0	1	AVCC: опорною напругою є напруга живлення (з конденсатором на AREF).
1	0	Зарезервовано (не використовується).
1	1	Internal: внутрішнє стабільне джерело 1,1 В (з конденсатором на AREF).

Діапазон значень аналого-цифрового перетворення визначається величиною опорної напруги AREF. Якщо вхідний сигнал перевищує рівень AREF, цифровий результат сягатиме свого максимуму – 0x3FF. В архітектурі ATmega128 передбачено три варіанти джерела опорної напруги:

1. Напруга живлення аналогової частини (AVCC), що комутується через внутрішній ключ.
2. Вбудоване джерело з номіналом 2,56 В, сформоване внутрішнім еталоном.
3. Зовнішній сигнал, що подається безпосередньо на пін AREF.

Оскільки вивід AREF має високий вхідний опір, він розрахований виключно на ємнісне навантаження. Для мінімізації впливу перешкод рекомендується встановлювати фільтрувальний конденсатор між цим виводом та загальним проводом («землею»). Поточне значення опорної напруги можна перевірити за допомогою вольтметра з високим внутрішнім опором на відповідному піні. Отримані дані АЦП мають 10-бітну розрядність і зберігаються у регістрах ADCH та ADCL. Залежно від стану біта ADLAR у регістрі ADMUX, результат може бути вирівняний як по правому краю (стандартно), так і по лівому, що зручно для швидкого зчитування 8-бітного результату.

Таблиця 3.2. Регісти даних ADC – ADCL і ADCH

ALAR=0:

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

ALAR=1:

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

запис									
	R	R	R	R	R	R	R	R	
Вих. значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Вибір конкретного каналу аналогового вводу та конфігурація диференційного підсилювача здійснюється шляхом встановлення відповідних значень бітів MUX у регістрі ADMUX. У режимі однополярного вимірювання джерелом сигналу для АЦП може виступати будь-який із портів ADC0–ADC7, шина заземлення (GND) або внутрішній еталон напруги 1,22 В.

При використанні диференційного режиму розробник має можливість призначити інвертуючий та неінвертуючий входи підсилювального каскаду. У цьому випадку АЦП обробляє різницю потенціалів між обраною парою входів, попередньо помножену на встановлений коефіцієнт підсилення (Gain). Якщо ж активовано однополярний режим, каскад підсилення виключається з ланцюга обробки, і сигнал надходить безпосередньо на вхід перетворювача. Нижче наведено структуру бітів регістра **ADCSRA**:

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таблиця 3.3. Управління подільником ADC

ADPS2	ADPS1	ADPS0	Подільник частоти
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Для синхронізації роботи АЦП використовується спеціальний подільник частоти, який що працює від тактової частоти мікроконтролера. Коефіцієнт масштабування встановлюється ступінчасто по 2^N (у діапазоні від 2 до 128) за допомогою бітів ADPS у регістрі ADCSRA (Табл.3.3). Активація подільника відбувається одночасно з увімкненням модуля АЦП шляхом

встановлення біта ADEN. При скиданні цього біта (ADEN=0) подільник припиняє функціонувати.

Для досягнення максимальної точності оцифрування (10-бітна роздільна здатність) рекомендується підтримувати частоту дискретизації в межах 50–200 кГц.

Стандартний цикл перетворення займає 13 тактів синхросигналу АЦП, проте перша ініціалізація після ввімкнення модуля потребує подовженого циклу — 25 тактів. По завершенні процесу дані записуються в регістри результату, а статусний прапорець ADIF набуває значення логічної одиниці.

У режимі поодинокого вимірювання біт ADSC автоматично скидається після завершення циклу. Для повторного запуску цей біт необхідно встановити програмно. Натомість у режимі безперервного перетворення (Free Running) новий цикл стартує автоматично одразу після попереднього, а біт ADSC постійно залишається активним.

Біт ADSC: Для ініціації поодинокого вимірювання необхідно встановити біт ADSC у стан логічної одиниці. Протягом усього циклу оцифрування цей біт зберігає високий рівень, а по завершенні операції автоматично скидається в «0». У режимі безперервного моніторингу АЦП здійснює циклічне оцифрування вхідного сигналу з постійним оновленням вихідних регістрів даних. Цей режим встановлюється записом лог. “1” в ADFR - регістр ADCSRA. Перше перетворення стартує при записі лог. “1” в біт ADSC - регістр ADCSRA. У цьому режимі виконуються періодичні обчислення, незважаючи чи скидається прапорець переривання АЦП ADIF чи ні.

Режими автоматизації (біти ADFR та ADATE) Активація циклічного режиму здійснюється шляхом встановлення біта ADFR (у регістрі ADCSRA). При цьому перший запуск все одно потребує запису одиниці в біт ADSC. У такому стані АЦП працює автономно, незалежно від стану прапорця переривання ADIF. Додатково, використання біта ADATE дозволяє перевести модуль у режим автоматичного перезапуску за подією. Вибір

конкретного джерела тригера (події), що ініціює старт, визначається конфігурацією бітів ADTS у регістрі SFIOR.

Біт ADIF: Індикація та переривання. Статус завершення операції відображається прапорцем ADIF, який активується після оновлення регістрової пари ADCH:ADCL. Якщо в системі дозволені переривання (встановлені біти ADIE та загальний прапорець I у регістрі SREG), після закінчення перетворення буде викликано відповідну підпрограму обробки. Очищення прапорця ADIF відбувається автоматично під час переходу процесора до вектора переривання.

Біт ADIE: Даний біт відповідає за активацію переривань після завершення циклу АЦП-перетворення. Генерація запиту на переривання стає можливою лише за умови одночасного встановлення біта ADIE у стан логічної одиниці та наявності дозволу на глобальні переривання (біт I у регістрі статусу SREG).

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таблиця 3.4. Формат регістра SFIOR (біти ADTS0, ADTS2, ADTS1)

ADTS2	ADTS1	ADTS0	Джерело запуску перетворення АЦП
0	0	0	Постійне перетворення (Free Running mode)
0	0	1	Аналоговий компаратор
0	1	0	Зовнішній запит на переривання 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Compare0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Вибір джерела для запуску перетворення залежить від стану біта ADATE у регістрі ADCSRA та конфігурації ADTS. Якщо ADATE встановлено у логічну одиницю, джерело сигналу визначається бітами ADTS. У випадку, коли ADATE дорівнює нулю, параметри ADTS не враховуються. Процес оцифрування активується за переднім фронтом обраного сигналу.

Варто зауважити, що переведення модуля у режим безперервного циклічного перетворення (конфігурація ADTS[2:0]=0) не створює автоматичного пускового сигналу, навіть якщо активний прапорець переривання ADIF. Для реалізації функцій широтно-імпульсної модуляції (ШІМ) у використовують виводи PF0, PF5 та PF6. Мікроконтролер ATmega128 включає два 16-розрядні таймери/лічильники (Timer/Counter1 та Timer/Counter3), регістри яких здатні приймати значення в діапазоні від 0 до 65536 (2^{16}). Функціональні можливості інтегрованих таймерів охоплюють широке коло завдань, зокрема:

- Формування сигналів із широтно-імпульсною модуляцією (ШІМ).
- Роботу в режимі прецизійного генератора імпульсів прямокутної форми.
- Наявність трьох незалежних каналів порівняння для кожного модуля (OC1A/B/C для Timer1 та OC3A/B/C для Timer3).

Таймер/лічильник 3, маючи 16-розрядну архітектуру, є універсальним інструментом для вимірювання часових проміжків, реєстрації кількості зовнішніх подій та синтезу ШІМ-сигналів із регульованими параметрами тривалості й шпаруватості на відповідних виводах. Аналогічний набір функцій притаманний і Таймеру/лічильнику 1. Внутрішню організацію та логіку роботи цих 16-бітних вузлів продемонстровано на блок-схемі (Рис.3.4).

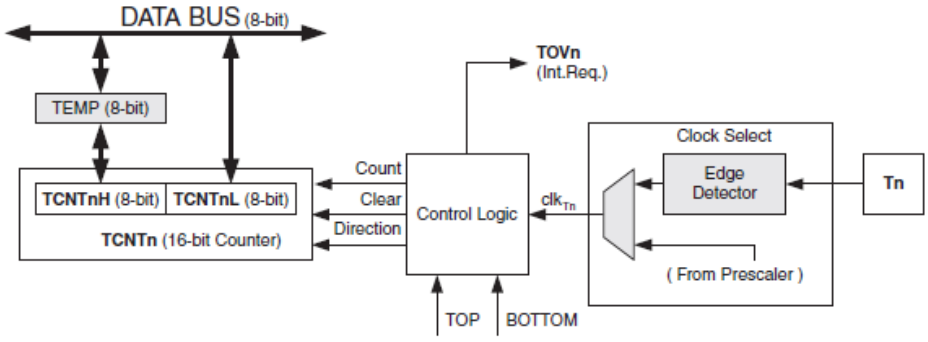


Рис. 3.4. Блок схема 16- розрядних Таймерів/лічильників

Процес функціонування лічильника базується на роботі таких керуючих сигналів та параметрів:

- Count - зміна поточного стану регістра TCNTn на одиницю (інкремент/ декремент).
- Direction- визначає режим роботи лічильника на збільшення-зменшення Clear - виконує скидання вмісту регістра TCNTn, встановлюючи всі його біти в нульовий стан.
- Clk - послідовність імпульсів, що забезпечує синхронізацію роботи таймера.
- TOP - індикатор, що спрацьовує при досягненні лічильником встановленого максимального порогу.
- BOTTOM - сигнал, що фіксує момент досягнення мінімально можливого значення в регістрі TCNTn.

Для зберігання значень 16-розрядного лічильника використовується регістрова пара TCNTn, що складається з двох 8-бітних регыстрів: TCNTH (старша частина) та TCNTL (молодша частина). Архітектура передбачає доступ до цих регістрів для зчитування поточного стану, так і для запису нових даних. Оскільки розрядність шини даних складає 8 біт, звернення до лічильника здійснюється шляхом виконання двох послідовних операцій.

- Запис даних: першим завантажується старший байт (TCNTH), а після нього — молодший (TCNTL).
- Читання даних: спочатку зчитується молодший байт (TCNTL), а потім — старший (TCNTH).

Динаміка зміни вмісту TCNTn залежить від конфігурації таймера. З кожним синхроімпульсом значення може інкрементуватися, декрементуватися або скидатися до нуля. Нижче наведено перелік ключових режимів роботи 16-бітного Таймера/лічильника3:

Normal Mode (Звичайний режим) Найпростішим алгоритмом функціонування таймера є режим Normal. У цьому стані вміст лічильного регістра TCNT3 інкрементується з кожним приходом тактового імпульсу. Процес триває до досягнення максимального значення $FFFF (рівень TOP). Після цього наступний сигнал викликає переповнення, і лічильник

скидається до початкового стану $0000, починаючи новий цикл відліку. Одночасно з переповненням активується прапорець TOV3 у регістрі TIFR. Якщо в регістрі TIMSK попередньо встановлено біт TOIE3, система генерує відповідне переривання.

Режим CTC – Clear Timer on Compare Match . Логіка роботи в режимі CTC багато в чому ідентична режиму Normal, проте з істотною різницею: межа рахунку (значення TOP) не є фіксованою. Замість максимального значення $FFFF, поріг визначається вмістом одного з регістрів порівняння (OCR3A/B/C) або регістром захоплення ICR3, залежно від обраної конфігурації. Щойно значення в TCNT3 стає рівним числу в контрольному регістрі, відбувається автоматичне обнулення лічильника. У момент такого збігу прапорець переповнення TOV3 набуває значення логічної одиниці.

Fast PWM - (Режим швидкої широтно-імпульсної модуляції). Даний режим функціонування призначений для формування високочастотних ШІМ-сигналів. За своєю базовою логікою та послідовністю операцій цей режим подібний до звичайного режиму Normal. Основна відмінність полягає у специфіці формування вихідного сигналу при досягненні лічильником встановлених порогів. Перелік регістрів, необхідних для налаштування 16-бітного Таймера 3, представлено нижче:

Таблиця 3.5. Регістр управління A TCCR3A

7	6	5	4	3	2	1	0	
COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

- Bit 7:6 – COMnA1:0: налаштування каналу A
- Bit 5:4 – COMnB1:0: налаштування каналу B
- Bit 3:2 – COMnC1:0: налаштування каналу C

Таблиця 3.6. Регістр управління B TCCR3B

7	6	5	4	3	2	1	0	
ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

- Bit 7 – ICNCn: Фільтр вхідних шумів. Встановлення біту, вмикає фільтрування шуму на вході ICP3.
- Bit 5 – зарезервовано.
- Bit 4:3 – WGMn3:2: Слугує для вибору режиму роботи лічильника див. (Табл.3.7).

Таблиця 3.7. Режим роботи таймера-лічильника

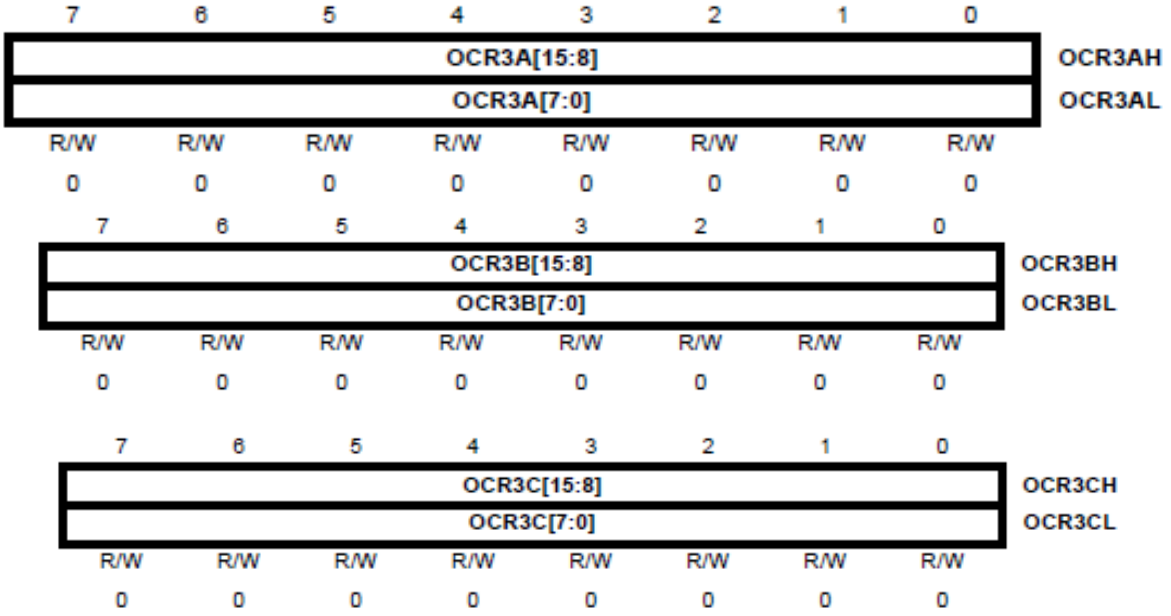
Режим	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter режими роботи	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

- Bit 2:0 – CSn2:0: Джерело імпульсів для синхронізації лічильника (Табл.3.8).

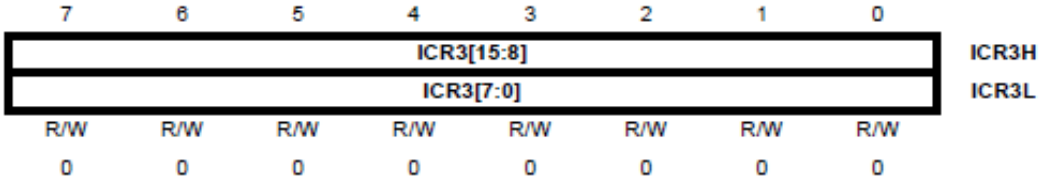
Таблиця 3.8. Джерела імпульсів синхронізації лічильника

CS12	CS11	CS10	Характеристика джерела синхронізації
0	0	0	Тактування відсутнє (зупинка таймера/лічильника)
0	0	1	Пряме тактування від системної шини clk_I/O (коефіцієнт 1)
0	1	0	Синхронізація через дільник частоти clk_I/O/8
0	1	1	Синхронізація через дільник частоти clk_I/O/64
1	0	0	Синхронізація через дільник частоти clk_I/O/256
1	0	1	Синхронізація через дільник частоти clk_I/O/1024
1	1	0	Зовнішнє джерело (вивід T1), спрацьовування за спадним фронтом
1	1	1	Зовнішнє джерело (вивід T1), спрацьовування за висхідним фронтом

Таблиця 3.9. Регістри порівняння OCR3An , OCR3Bn, OCR3Cn.



Таблиця 3.10. Регістр ICR3n



Регістр захоплення ICR3n можна використати для встановлення значення TOP (режим 14. Табл. 3.7). В Таблиці 3.11. проілюстровано сигнали порта PE (OC3C, OC3A, OC3B,) для режиму Fast PWM, які залежать від бітів регістра управління A TCCR3A (Табл.3.5.).

Таблиця 3.11. Режими Fast PWM, сигнали на PE (OC3A, OC3B, OC3C)

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	**Пояснення**
0	0	Normal port operation, OCnA/OCnB/OCnC **від'єднано**
0	1	WGMn3:0 = 15: Toggle OCnA on Compare Match, OCnB/OCnC disconnected (normal port operation). For all other WGMn settings, normal port operation, OCnA/OCnB/OCnC **від'єднано**
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM, (non-inverting mode)
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM, (inverting mode)

Апаратна реалізація ШІМ має незаперечні переваги над програмним методом. Використання інтегрованих модулів таймера дозволяє суттєво розвантажити центральний процесор, позбавляючи його потреби виконувати громіздкі програмні цикли для формування сигналів. Це вивільняє обчислювальні ресурси для виконання більш пріоритетних завдань системи.

Після завершення ініціалізації та конфігурування відповідних регістрів, таймер/лічильник переходить у автономний режим функціонування. ЦП при цьому залишається вільним для інших операцій, взаємодіючи з таймером лише епізодично — наприклад, для корекції параметрів сигналу або зміни алгоритму роботи.

Опис датчика тиску BMP180. BMP180 – це сучасний цифровий датчик тиску, виконаний за технологією MEMS, від німецького виробника Bosch Sensortec GmbH. Пристрій являє собою монолітний кристал, що поєднує в собі чутливий механічний елемент та інтегровану схему для цифрової обробки сигналів. Основними сферами застосування сенсора є портативні барометри, альтиметри та системи моніторингу погоди.

Датчик здатен реєструвати атмосферний тиск у межах 300–1100 гПа, що відповідає висотному діапазону від +9000 до -500 метрів щодо рівня Світового океану. Термометр, вбудований у пристрій, працює в температурному діапазоні від --40°C до +85°C. Модуль вирізняється низьким енергоспоживанням (напруга живлення від 1,8 до 3,6 В) та використовує для комунікації з мікроконтролером послідовний інтерфейс I^2C. За допомогою BMP180 можна міряти не лише тиск і температуру, а й визначати абсолютну чи відносну висоту, а також швидкість вертикального переміщення. Варто зазначити, що даний пристрій є функціональним наступником попередньої моделі BMP085. Експлуатаційні та метрологічні параметри сенсора BMP180:

- робочий температурний режим від -40 °C до +85 °C;
- оптимальний температурний діапазон (висока точність) 0 °C … +65 °C;
- електроживлення: 1,8–3,6 В (стандарт - 3,3 В);

- похибка вимірювання тиску: ± 0,12 гПа (гектоПаскаль);
- похибка визначення висоти: ±1 м;
- дискретність (тиск / температура): 0,01 гПа/0,1 °С;
- точність термометра (при 0…+65 °С) ±1 °С.

Тривалість циклу вимірювання тиску в датчику BMP180 безпосередньо залежить від обраної точності. Виробником передбачено кілька програмних профілів, що дозволяють балансувати між швидкістю та роздільною здатністю:

- режим мінімальної потужності (Ultra Low Power): час обробки складає 4,5 мс;
- базовий режим (Standard): формування результату триває 7,5 мс;
- режим підвищеної точності (High Resolution): цикл вимірювання = 13,5 мс;
- профіль максимальної прецизійності (Ultra High Resolution): потребує 25,5 мс;
- екстремальна роздільна здатність: тривалість обчислень сягає 76,5 мс.

Визначення температурних показників виконується фіксовано за 4,5 мс. Для забезпечення високої швидкості передачі даних інтерфейс пристрою підтримує тактову частоту до 3,4 МГц. В продажі доступні готові модулі з датчиком тиску BMP180 (Рис.3.5.).

Рис. 3.5. Модуль з датчиком BMP180 фірми Bosch Sensortec GmbH

Призначення пінів датчика BMP180 зображено на Рис.3.6.

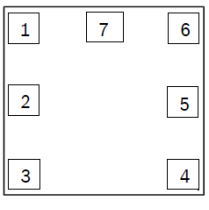


Рис. 3.6. Нумерація пінів датчика BMP180

Призначення виводів датчика: Вивід 1: Резервний (не задіяний). Вивід 2 (VDD): Вхід основного джерела живлення. Вивід 3 (VDDIO): Лінія живлення цифрового інтерфейсу I2C. Вивід 4: Не використовується. Вивід 5 (SCL): Вхід для сигналу тактування послідовної шини. Вивід 6 (SDA): Лінія передачі/прийому послідовних даних. Вивід 7 (GND): Загальний провід системи (“земля”). Конструкція датчика BMP180 розрахована на значні механічні навантаження, зокрема він здатний витримувати тиск до 10 000 гПа без втрати працездатності. Взаємодія з керуючим мікроконтролером реалізована через інтерфейс I^2C. Для усунення похибок, що виникають під час зняття показників тиску й температури, використовуються індивідуальні калібрувальні параметри. Ці коефіцієнти попередньо записані в енергонезалежну пам’ять (E^2PROM) пристрою на етапі виробництва.

Структурно модуль складається з таких компонентів: п’єзорезистивний вимірювальний елемент; аналого-цифровий перетворювач (АЦП); контролер керування; блок пам’яті E^2PROM, що містить 176 біт індивідуальних калібрувальних даних. У процесі роботи сенсор формує два типи значень: UT (необроблені дані температури, 16 біт) та UP (дані тиску, від 16 до 19 біт залежно від обраної точності). Стандартний варіант вмикання BMP180 у мікропроцесорну систему представлений на схемі (Рис. 3.7).

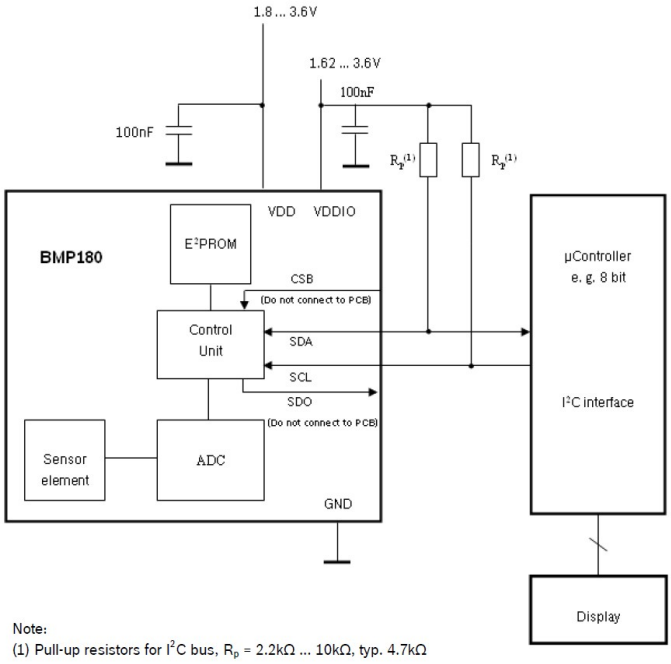


Рис. 3.7. Стандартний варіант вмикання BMP180 до МК

Вимірювання тиску і температури. Процес отримання даних розпочинається з надсилання мікроконтролером стартової команди на ініціацію вимірювання температури або тиску. Після завершення внутрішнього циклу обробки сигналу в сенсорі, отримані «сирі» значення (UT або UP) зчитуються через послідовний інтерфейс I^2C. Для перерахунку цих кодів у фізичні величини (градуси Цельсія та гектоПаскалі) застосовуються індивідуальні калібрувальні коефіцієнти. Читання цих параметрів з пам’яті датчика виконується одноразово на етапі ініціалізації системи. У стандартному режимі функціонування частота оцифрування тиску може сягати 128 вимірів на секунду. За такої інтенсивності зчитування температури проводиться лише раз на секунду, а отримане значення використовується як константа для температурної компенсації всіх 128 барометричних вимірів. Детальний опис регістрової структури BMP180 подано в Табл.3.12.

Таблиця 3.12. Регістри датчика тиску BMP180

Register Name	Register Adress	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
out_xlsb	F8h	adc_out_xlsb<7:3>					0	0	0	00h
out_lsb	F7h	adc_out_lsb<7:0>								00h
out_msb	F6h	adc_out_msb<7:0>								80h
ctrl_meas	F4h	oss<1:0>		sco	measurement control					00h
soft reset	E0h	reset								00h
id	D0h	id<7:0>								55h
calib21 downto calib0	B7h down to AAh	calib21<7:0> down to calib0<7:0>								n/a

Registers:	Control registers	Calibration registers	Data registers	Fixed
Type:	read / write	read only	read only	read only

Регістр керування (CTRL_MEAS, адреса F4h) Даний регістр відповідає за ініціалізацію процесу вимірювання та налаштування параметрів точності.

- SCO (5-й біт): виконує функцію індикатора та запуску перетворення. При записі логічної одиниці сенсор розпочинає цикл вимірювання. Протягом усього часу обробки даних цей біт зберігає високий рівень (лог. 1), а після завершення обчислень автоматично скидається в логічний нуль.
- OSS (біти 7:6): визначає рівень (oversampling), що впливає на роздільну здатність і тривалість циклу вимірювання. Доступні наступні конфігурації: 00: одинарне перетворення (час затримки - 4,5 мс); 01: 2-кратне осереднення (затримка - 7,5 мс); 10: 4-кратне осереднення (затримка - 13,5 мс); 11: 8-кратне осереднення (затримка - 25,5 мс).

Регістр програмного скидання (SOFT_RESET, адреса 0xE0): призначений виключно для запису. Внесення значення 0xB6 ініціює повний перезапуск (reset) датчика аналогічно апаратному вимкненню живлення.

Регістр ідентифікації (CHIP_ID, адреса 0xD0): містить незмінне заводське значення 0x55, яке використовується для перевірки наявності пристрою на шині.

Зчитування результатів вимірювання здійснюється через регістри даних з адресами 0xF6, 0xF7 та 0xF8. Особливістю модуля є можливість довільної послідовності доступу до цих байтів. Взаємодія з мікроконтролером базується на протоколі I^2C, який у даній моделі підтримує швидкісні режими до 3,4 Мбіт/с. Для стабільної роботи шини лінії SCL та SDA мають бути з’єднані з позитивною шиною живлення через підтягуючі

резистори, рекомендований номінал яких становить 4,7 кОм (допустимий діапазон від 2,2 до 10 кОм).

На фізичному рівні адресація модуля на шині залежить від типу операції: для передачі команд використовується адреса 0xEE, а для отримання даних — 0xEF.

A7	A6	A5	A4	A3	A2	A1	W/R
1	1	1	0	1	1	1	0/1

Процес передачі даних за протоколом I^2C (Рис. 3.8) базується на чергуванні трьох основних станів: умови START, фази обміну даними та умови STOP.

1. Ініціалізація (START): Сеанс зв’язку завжди розпочинається мікроконтролером (ведучим пристроєм). Стан “Старт” фіксується, коли на лінії тактування SCL утримується високий рівень, а на лінії даних SDA відбувається перехід з логічної одиниці в нуль.
2. Адресація: Після стартового сигналу МК транслює 7-бітну адресу цільового сенсора. Восьмий біт (R/W) визначає характер наступної операції: запис даних у регістри BMP180 чи їх зчитування.
3. Підтвердження (ACK): Якщо адреса збігається з внутрішньою адресою датчика, він підтверджує готовність, встановлюючи низький рівень на шині SDA протягом дев'ятого такту SCL.
4. Передача даних: Зміна інформаційних бітів на лінії SDA дозволена лише за умови низького рівня на SCL. Коли ж на шині тактування встановлено високий рівень, стан лінії даних має залишатися стабільним для коректного зчитування.
5. Завершення (STOP): Робота припиняється за умови «Стоп»: лінія SCL переходить у високий стан, після чого на SDA також встановлюється логічна одиниця (Рис.3.8.).

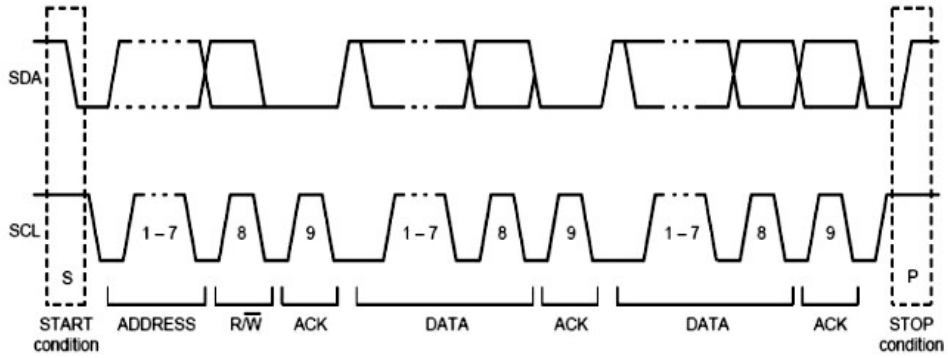


Рис. 3.8. Протокол обміну по інтерфейсу – I^2C

Алгоритм отримання даних про температуру та тиск через інтерфейс I2C реалізується у чіткій послідовності операцій. Процес розпочинається з генерації умови START, після чого мікроконтролер транслює адресу сенсора з бітом запису. Далі передається номер цільового регістра та відповідна команда для ініціації вимірювання. Характерною особливістю протоколу є те, що після кожного прийнятого байта модуль BMP180 генерує сигнал підтвердження (ACKS). Детальна часова діаграма взаємодії вузлів представлена на Рис. 3.9.

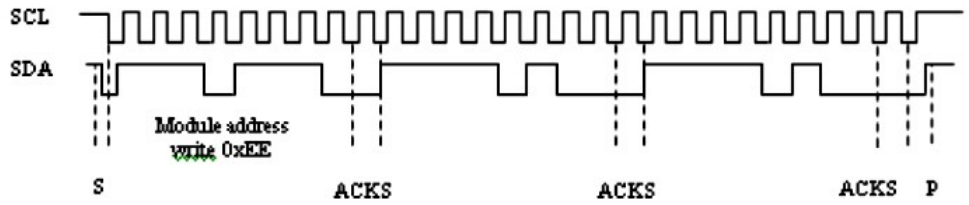


Рис. 3.9. Читання даних по інтерфейсу I^2C

Опис та вивід значень на РКД (LCD). Для візуалізації даних про поточний стан та робочі параметри системи доцільно використовувати рідкокристалічні дисплеї (РКД). Дані модулі вирізняються здатністю відображати значні масиви інформації при мінімальних витратах електроенергії. Завдяки інтегрованому підсвічуванню такі екрани можна читати навіть в умовах недостатнього або повністю відсутнього зовнішнього освітлення. Широкий температурний діапазон експлуатації (від -20°C до +70°C) дозволяє інтегрувати РК-модулі в мобільні пристрої та бортову апаратуру. Конструктивно виріб складається з рідкокристалічної панелі та спеціалізованого контролера керування. Типовий модуль, представлений на

рис. 3.10, здатний відображати 16 символів у два рядки. Формування кожного знака відбувається в матриці форматом 5x8 пікселів, при цьому між сусідніми символами передбачено технологічний розрив завширшки в одну точку.

Рис. 3.10. РКД (WH1602B-YGK-CTK)

Архітектура модуля WH1602B-YGK-CTK:

- Модуль побудований на базі контролера, що повністю сумісний із галузевим стандартом HD44780.
- Дисплей підтримує відображення текстової інформації у форматі 2 рядки по 16 знакомісць.
- Використовується надійне світлодіодне (LED) підсвічування з класичним жовто-зеленим спектром випромінювання.
- Функціонування дисплея базується на взаємодії внутрішньої логіки керування РК-панеллю та кількох типів пам'яті. В оперативній пам'яті модуля кожному знаку на екрані призначено конкретну адресу комірки, де зберігається його код.

Архітектура пристрою передбачає наявність двох сегментів знакогенератора: один містить стандартні коди символів, а інший дозволяє користувачеві самостійно програмувати унікальні графічні знаки. Функціональні характеристики модуля:

- Вбудований знакогенератор містить дві активні сторінки, що забезпечує коректне відображення як латинських символів, так і кириличного алфавіту.
- Модуль підтримує роботу за двома протоколами обміну даними — 4-бітним або 8-бітним. Конкретний режим встановлюється програмно під час ініціалізації дисплея.

- Пристрій здатен приймати команди керування від мікроконтролера, а також виконувати операції запису та зчитування даних безпосередньо через шину даних.
- Реалізована можливість зчитування прапорця стану (статус-байта), що дозволяє контролеру перевіряти готовність модуля до наступної операції.
- Передбачено наявність області пам’яті для створення та зберігання до 8 унікальних графічних символів, визначених користувачем.
- Програмне налаштування типу курсору (статичний або з ефектом мерехтіння), а також можливість апаратного або програмного регулювання інтенсивності підсвічування та рівня контрастності зображення.

Для розуміння алгоритмів роботи символьних РК-модулів необхідно проаналізувати внутрішню структуру базового контролера HD44780. У процесі функціонування програма взаємодіє безпосередньо з цим чіпом, тоді як допоміжні елементи (регістри зсуву, драйвери сегментів, знакогенератор) забезпечують апаратну регенерацію зображення та формування курсора без втручання зовнішнього коду.

Взаємодія керуючої системи з контролером базується на використанні двох ключових регістрів: IR (інструкцій) та DR (даних). Вибір конкретного регістра здійснюється за допомогою сигналу на лінії RS: При RS=0 виконується звернення до регістра команд (IR). При RS=1 активується доступ до регістра даних (DR). Інформація, що надходить до регістра DR, перенаправляється до відеопам’яті (DDRAM) або пам’яті знакогенератора (CGRAM) відповідно до поточного значення лічильника адреси (AC). Регістр IR, у свою чергу, передає отримані коди до блоку виконання інструкцій. Обсяг відеопам’яті становить 80 байт, що дозволяє зберігати коди символів у форматі двох рядків по 40 знакомісць у кожному. Така логічна організація є універсальною: незалежно від фізичної конфігурації екрана (наприклад, 20x4 чи 8x1), контролер завжди сприймає пам’ять як два рядки по 40 символів.

Оскільки HD44780 працює за принципом динамічної індикації, він циклічно оновлює стан сегментів РК-панелі. Власні ресурси драйвера дозволяють керувати 40 сегментами (SEG1...SEG40), чого достатньо для підтримки 8-символьних дисплеїв. Модулі з більшою кількістю знакомісць потребують додаткових драйверів (наприклад, HD44100), кожен з яких забезпечує роботу ще 40 сегментів.

Підключення символьного LCD. CodeVisioAVR має готову бібліотеку для роботи з такими РКД – модулями. Бібліотека дозволяє працювати з різними форматами LCD-модулів — від компактних 1x8 до повнорозмірних 2x40, включаючи популярні типорозміри 2x16 та 4x20. Схема з'єднання модуля LCD з МК ATmega128 (Рис.3.11)

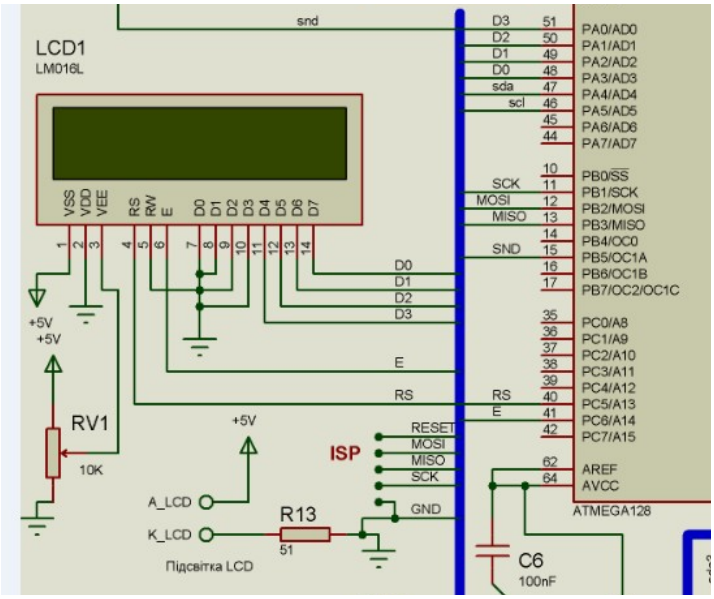


Рис. 3.11. Схема з'єднання символьного LCD з МК ATmega128

3.2. Проектування цифрового висотоміра в САПР Proteus VSM

Основним призначенням проектованого цифрового висотоміра є моніторинг ключових метеорологічних показників таких як: статичного тиску, температуру повітря, висоту над рівнем океану (альтитуду), відносну висоту, Крім того, пристрій здатен формувати короткостроковий прогноз погоди та забезпечувати взаємодію з користувачем через систему меню та РК-дисплей. Структурна схема приладу, наведена на Рис.3.12, відображає єдність апаратної та програмної складових. Роль центрального

обчислювального модуля виконує мікроконтролер AVR (ATmega128), який обробляє дані, що надходять від цифрового сенсора BMP180. До складу апаратної частини також інтегровано вузол узгодження рівнів, блок кнопок керування, світлодіодну індикацію станів та символьний дисплей для візуалізації вимірюваних параметрів для подальшого аналізу та обробки.

Для забезпечення коректної взаємодії компонентів пристрою передбачено вузол узгодження рівнів, до складу якого входять регулятор напруги U1, транзисторні ключі Q1, Q2, резистивна обв’язка R3–R6 та фільтруючі конденсатори C4, C5. Цей блок виконує дві функції: знижує напругу живлення до 3,3 В для модуля BMP180 та адаптує логічні рівні сигналів обміну даними між сенсором і мікроконтролером. Керування функціоналом висотоміра здійснюється за допомогою чотирьох кнопок: “MENU/ENTER” та “EXIT” - для навігації та виходу з налаштувань; “SELECT+” та “SELECT-“ - для інкременту/декременту параметрів. Обчислювальним ядром системи є мікроконтролер ATmega128 (архітектура AVR), який відповідає за зчитування даних із датчика, опрацювання команд від кнопок, керування світлодіодною індикацією та вивід результатів на символьний РК-дисплей формату 16x2. Кольорова індикація дозволяє швидко визначити поточний стан пристрою: жовтий світлодіод активується в режимі альтиметра, а синій - під час барометричного прогнозування погоди.

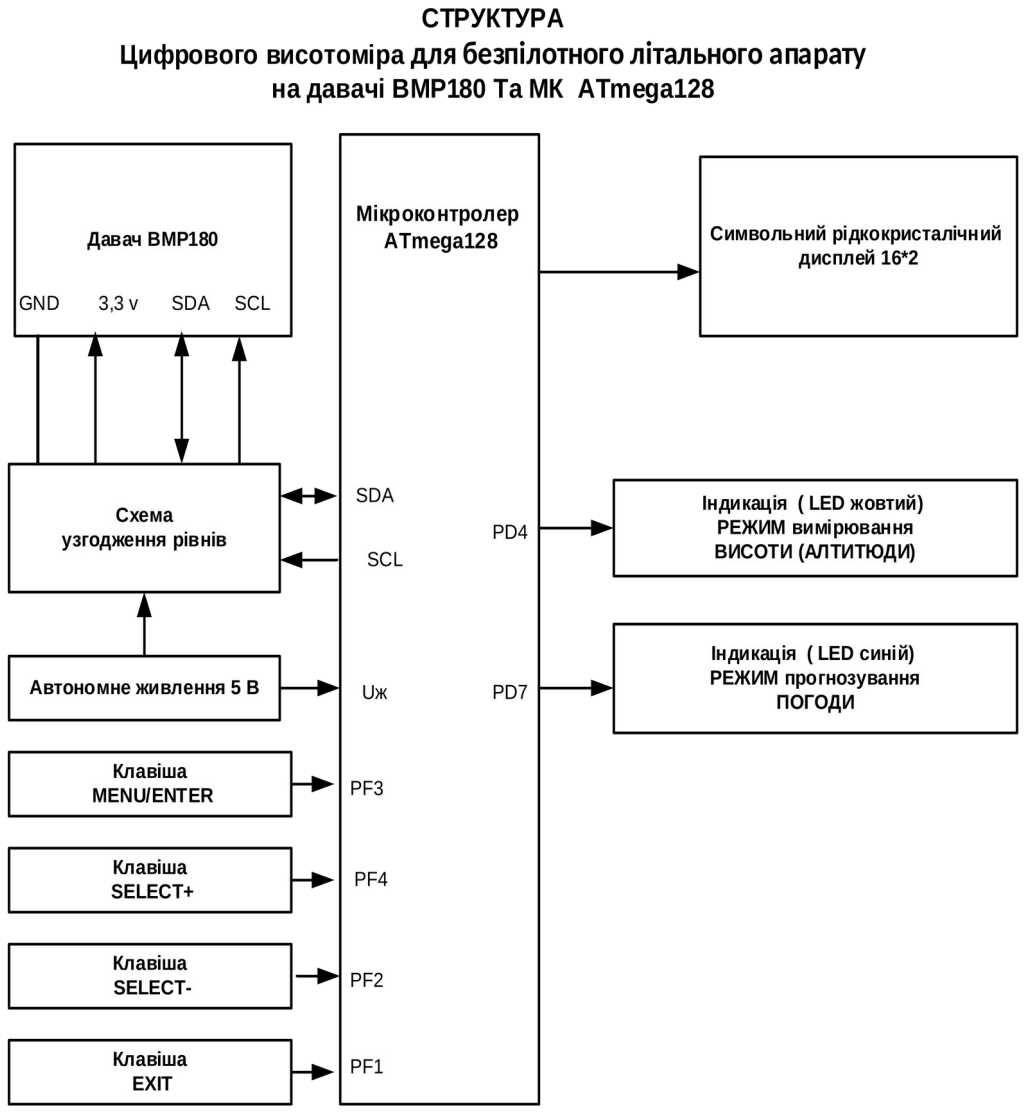


Рис. 3.12. Загальна структура цифрового висотоміра

На Рис.3.13 проілюстровано апаратне забезпечення цифрового висотоміра з МК управлінням створене в САПР Proteus VSM.

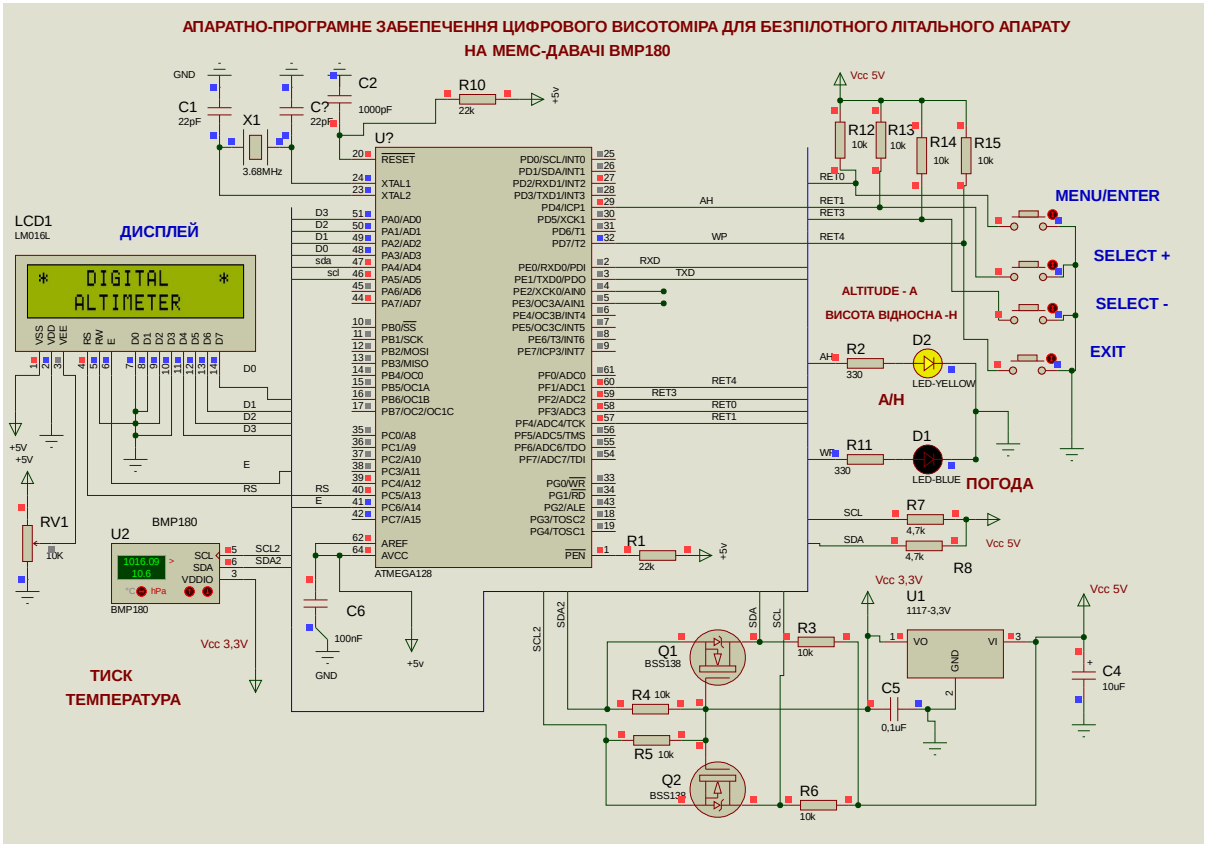


Рис. 3.13. Апаратне забезпечення цифрового висотоміра спроектоване в САПР Proteus VSM

МК здійснює опрацювання отриманих даних (параметрів тиску та температури) з подальшою візуалізацією на РК-модулі. Взаємодія з цифровим сенсором U2 реалізована через інтерфейсні лінії SDA та SCK, які через вузол узгодження рівнів під'єднані до виводів PA4 та PA5 відповідно. Відображення текстової інформації забезпечує символьний дисплей моделі WH1602B-YGK-CTK, підключений до шини порту A. Система керування та світлової індикації реалізована наступним чином:

- клавіша виклику меню (MENU/ENTER) заведена на вивід PF3, кнопки навігації SELECT+ та SELECT- на PF4 і PF2, а кнопка повернення до робочого стану (EXIT) на PF1.
- жовтий світлодіод D2 підключено до піна PD4, а синій індикатор D1 до PD7.

РОЗДІЛ 4
ПРОГРАМНО-АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ВИСОТОМІРА

4.1. Алгоритм роботи цифрового висотоміра

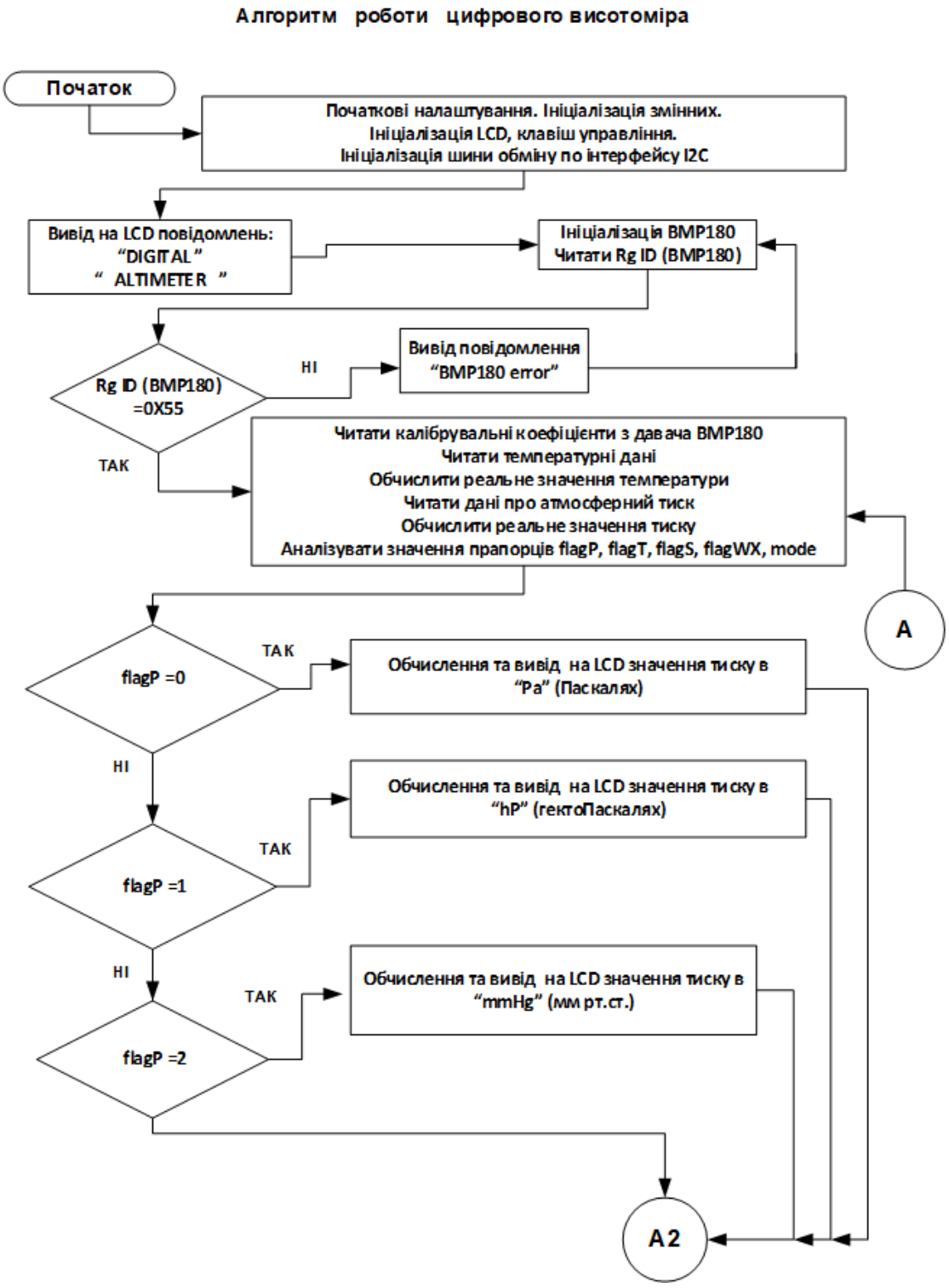


Рис. 4.1 Алгоритм роботи цифрового висотоміра (початок)

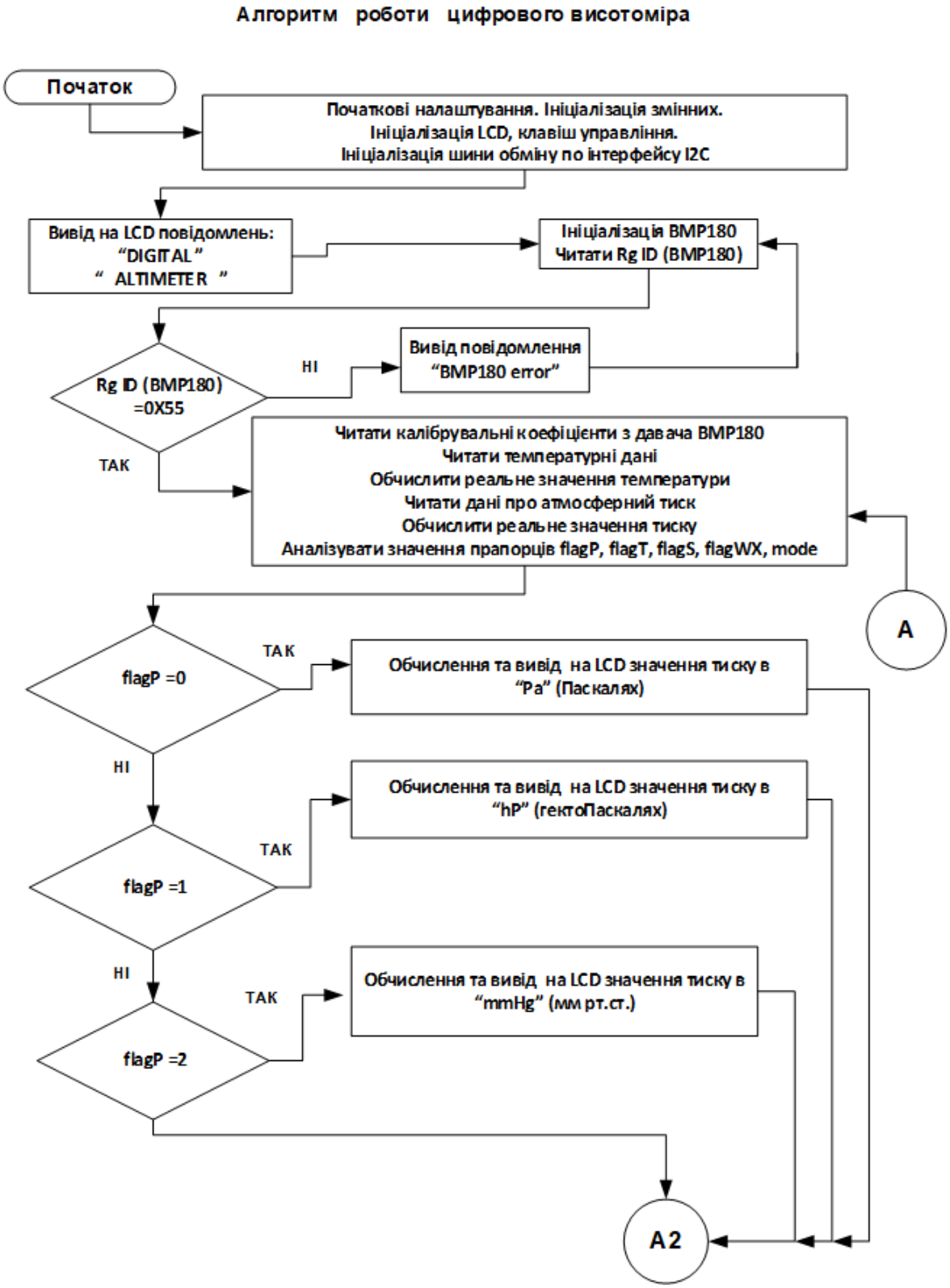


Рис. 4.2 Алгоритм роботи цифрового висотоміра (продовження)

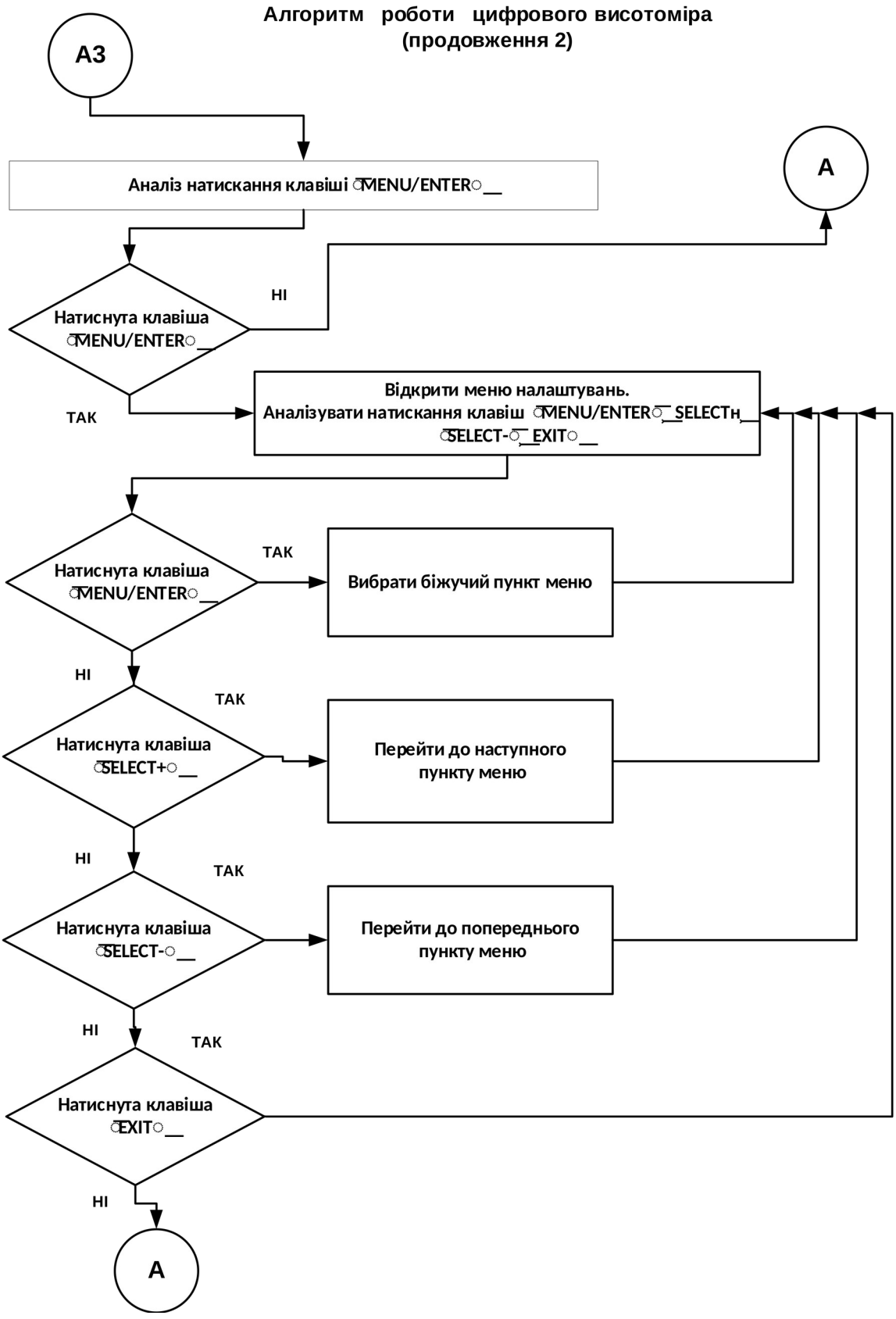


Рис. 4.3. Алгоритм роботи цифрового висотоміра (продовження 2)

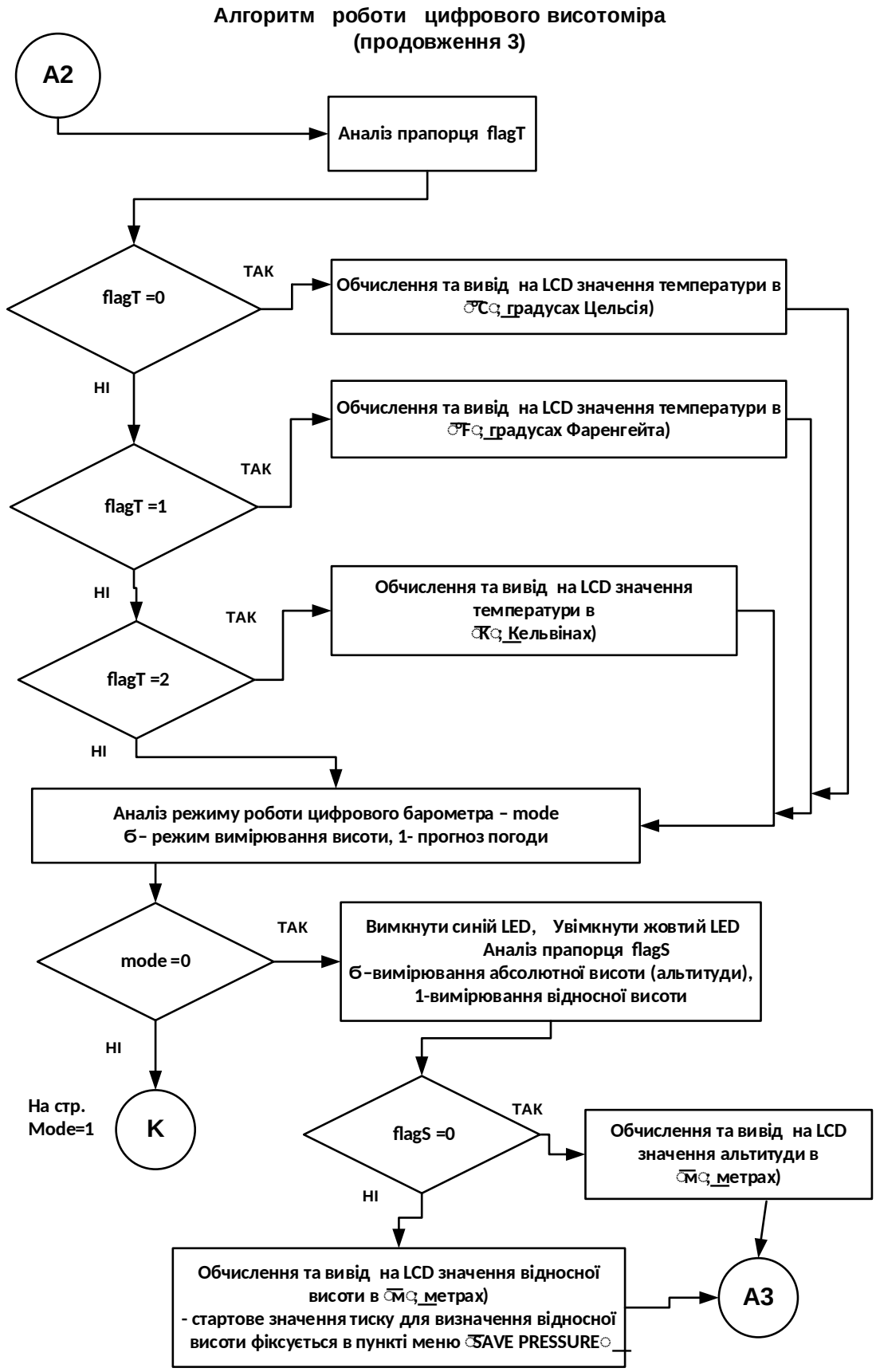


Рис. 4.4. Алгоритм роботи цифрового висотоміра (продовження 3)

Алгоритм роботи цифрового висотоміра (продовження 4)

Пункти меню, та алгоритм їх роботи

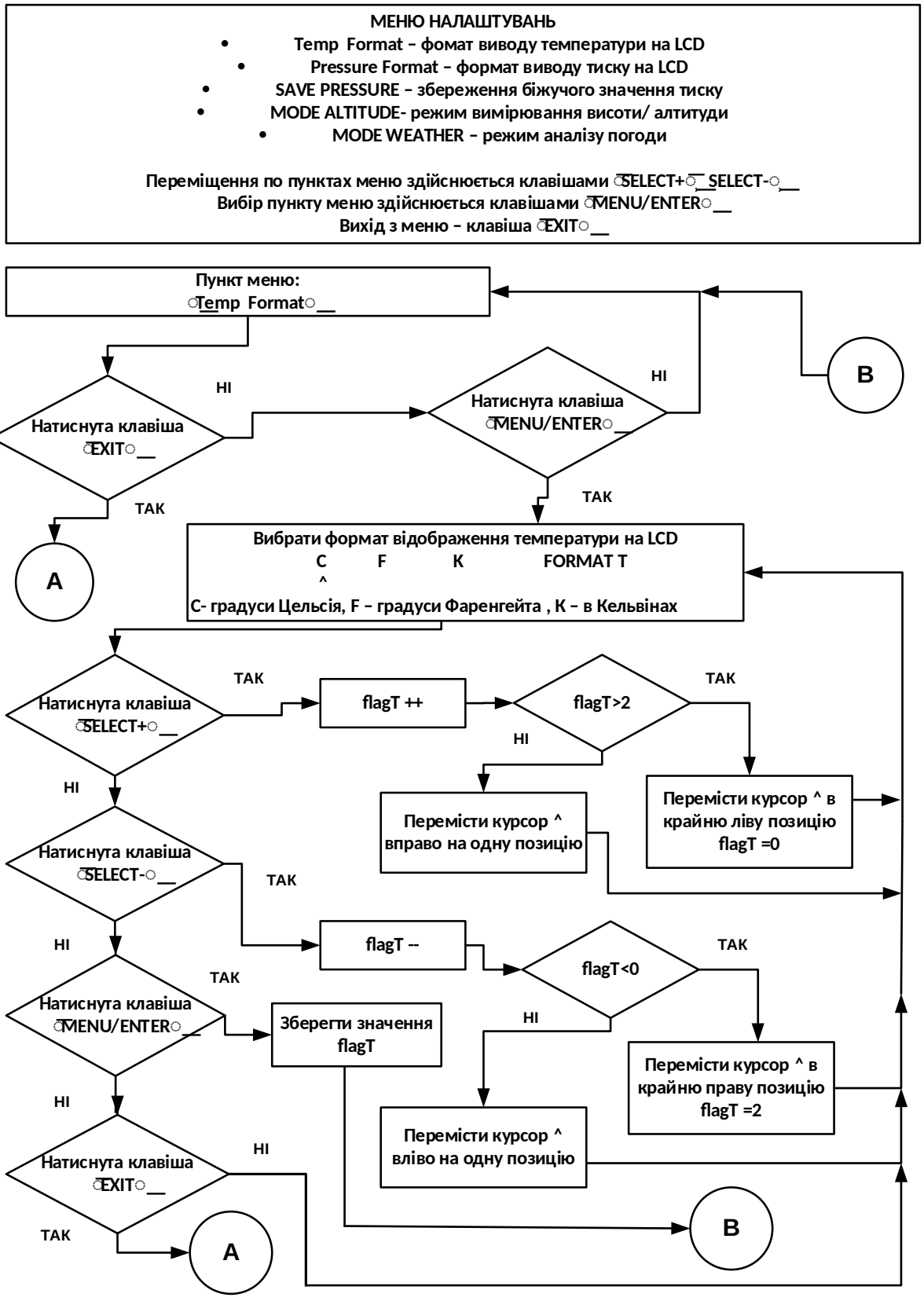


Рис. 4.5. Алгоритм роботи цифрового висотоміра (продовження 4)

Алгоритм роботи цифрового висотоміра (продовження 5)

Пункти меню, та алгоритм їх роботи

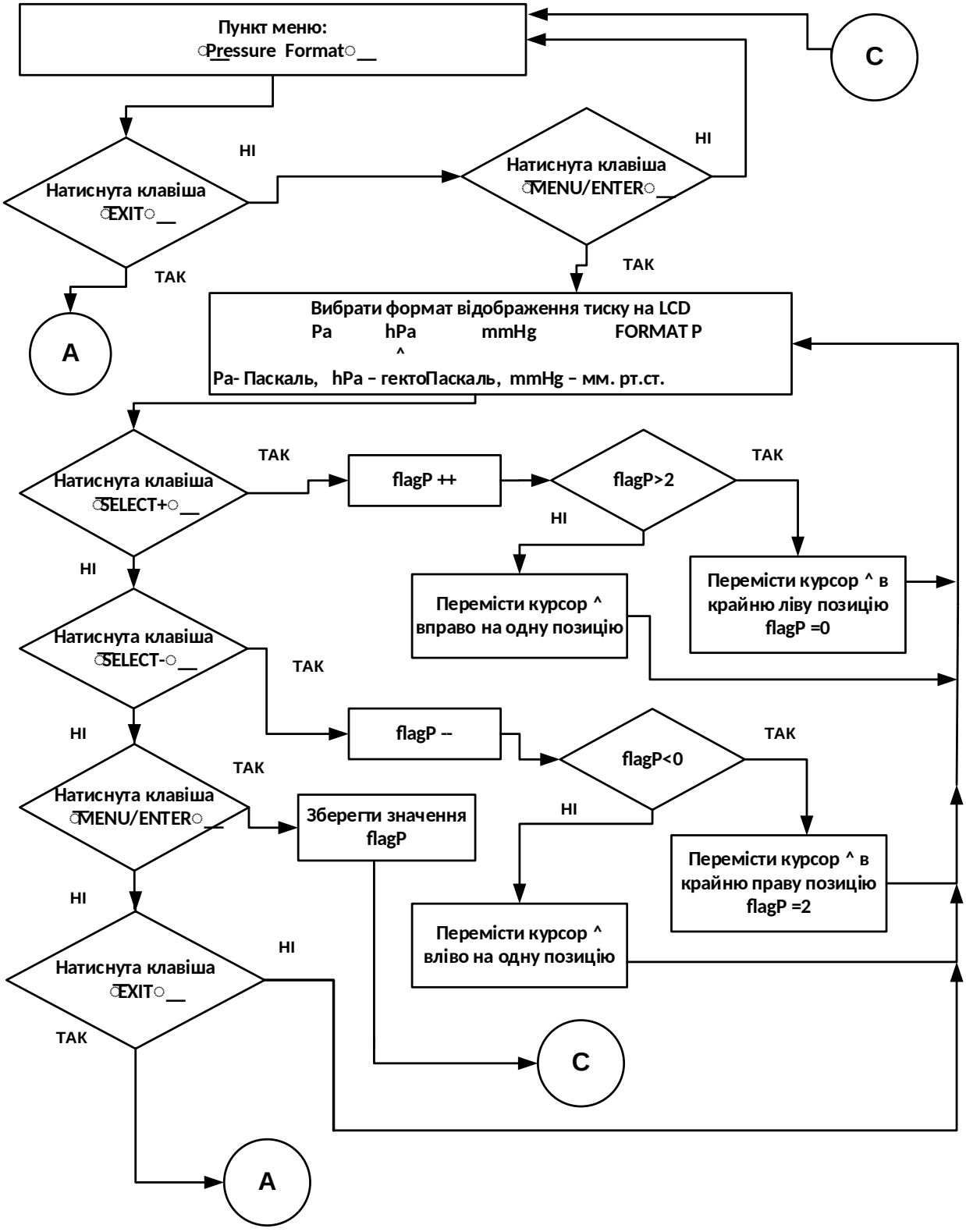


Рис. 4.6. Алгоритм роботи цифрового висотоміра (продовження 5)

Алгоритм роботи цифрового висотоміра (продовження 6)

Пункти меню, та алгоритм їх роботи

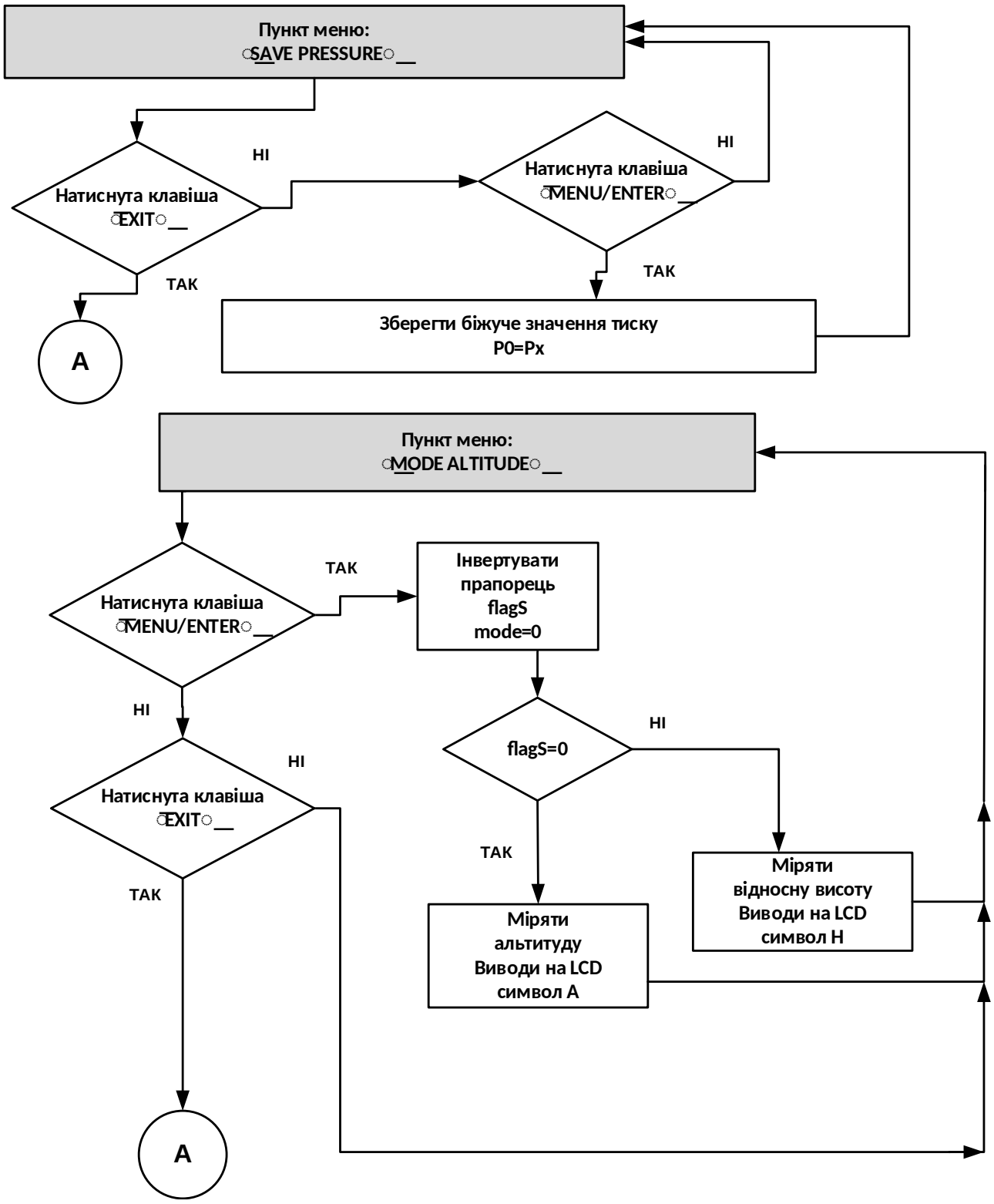


Рис. 4.7. Алгоритм роботи цифрового висотоміра (продовження 6)

Алгоритм роботи цифрового висотоміра в режимі ПОГОДА (MODE=1)
В цьому режимі відбувається прогнозування погоди (продовження 7)

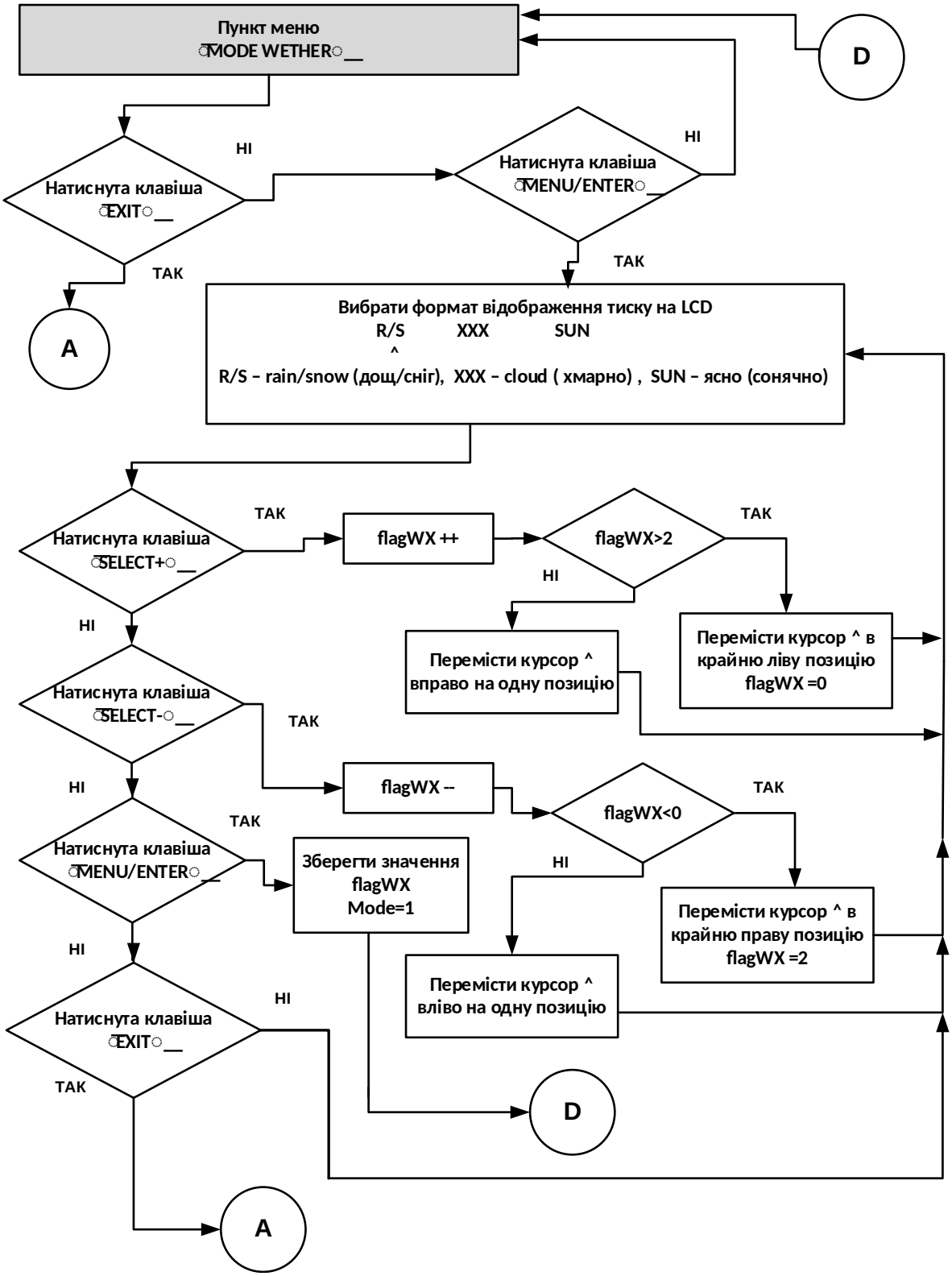


Рис. 4.8. Алгоритм роботи цифрового висотоміра (продовження 7)

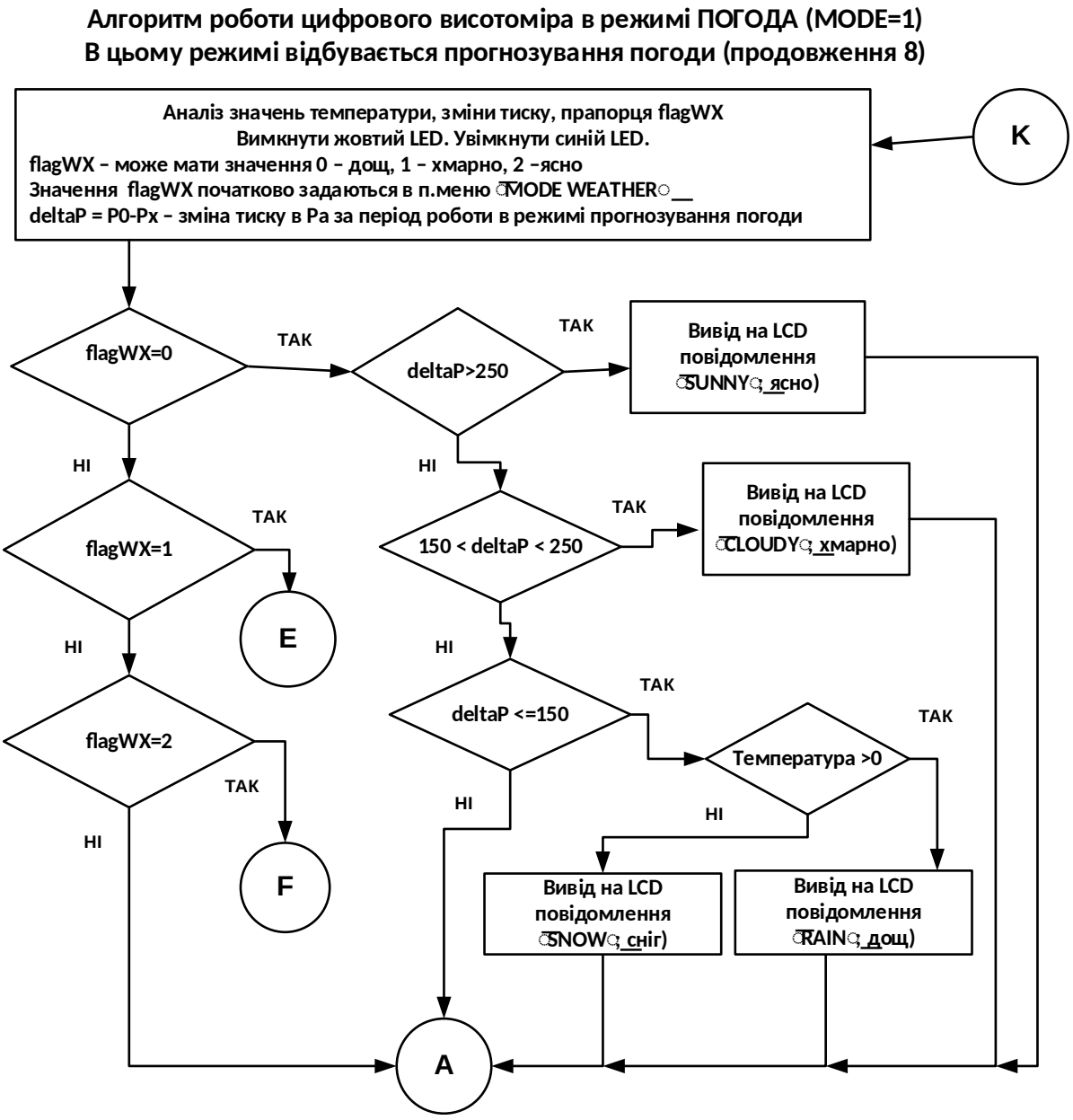


Рис. 4.9. Алгоритм роботи цифрового висотоміра (продовження 8)

Після подачі напруги живлення розпочинається виконання вбудованого програмного забезпечення мікроконтролера. Початковий етап роботи передбачає ініціалізацію внутрішніх змінних, після чого відбувається конфігурування апаратних ресурсів та периферії. Зокрема, система налаштовує інтерфейс I^2C, готує до роботи сенсор BMP180, символьний РК-дисплей, а також порти введення-виведення для клавіш керування та світлодіодних індикаторів. Процес запуску супроводжується виводом на екран на 2 секунди повідомлення “DIGITAL ALTIMETER”. Паралельно виконується перевірка цілісності шини даних та опитування датчика

BMP180. Якщо під час самодіагностики виявляється збій, на дисплей виводиться сповіщення “BMP180 error”. За умови успішного тестування програма переходить до основного циклу, що включає безперервне вимірювання метеоданих, їх математичну обробку та моніторинг стану клавіші “MENU/ENTER” для взаємодії з користувачем.

Програмний алгоритм взаємодії з датчиком базується на послідовному виклику створених функцій. Спочатку за допомогою bmp180Calibration() зчитуються заводські калібрувальні коефіцієнти. Далі функція BMP180ReadUT() отримує необроблені дані про температуру, оскільки вони є критично важливими для подальшої температурної компенсації показників тиску. Обчислення фактичного атмосферного тиску (в Паскалях) реалізовано через функцію bmp180GetPressure(up). Формат відображення даних на РК-дисплеї визначається станом системних прапорців:

- flagP (тиск): керує одиницями виміру – Паскалі (значення 0), гектопаскалі (значення 1) або міліметри ртутного стовпа (значення 2).
- flagT (температура): задає шкалу відображення — Цельсія (значення 0), Фаренгейта (значення 1) або Кельвіна (значення 2).
- mode (режим системи): визначає функціональний стан пристрою, де значення 0 відповідає вимірюванню висотних характеристик (абсолютної та відносної), а значення 1 активує барометричне прогнозування погоди.

За замовчуванням пристрій активується у стані mode=0. Даний режим супроводжується ввімкненням жовтого світлодіодного індикатора. Логіка обчислень у цьому стані визначається параметром flagS:

- При flagS = 0 система розраховує та відображає на екрані альтитуду (абсолютну висоту щодо рівня океану в метрах).
- При flagS = 1 прилад переходить у режим висотоміра, обчислюючи відносну зміну висоти.

Для коректного визначення відносної висоти необхідно попередньо зберегти базове значення атмосферного тиску. Ця процедура виконується

через програмний пункт меню “SAVE PRESSURE” (Рис. 4.2). У процесі функціонування висотомір безперервно проводить моніторинг температури й тиску, оновлює розрахункові дані на дисплеї та перебуває в режимі очікування натискання кнопки “MENU/ENTER”. Активація кнопки “MENU/ENTER” переводить пристрій у режим конфігурування, де програма починає відстежувати стан чотирьох керуючих клавіш : “MENU/ENTER”, “SELECT+”, “SELECT-”, “EXIT”. Логічна послідовність обробки сигналів від кнопок детально відображена в алгоритмах на Рис. 4.3 та 4.4. Структура користувацького інтерфейсу містить п’ять основних підпунктів:

- “TempFormat” – налаштування шкали відображення температури;
- “PressureFormat” - вибір одиниць вимірювання атмосферного тиску;
- “SAVE PRESSURE” - фіксація поточних показників тиску в пам’яті;
- “MODE ALTITUDE” - перемикання у режим вимірювання висотних параметрів;
- “MODE WEATHER” - перехід до функції барометричного прогнозування.

Навігація здійснюється кнопками “SELECT+” (наступний елемент) та “SELECT-“ (попередній елемент). Підтвердження вибору пункту меню виконується клавішею “MENU/ENTER”, а повернення до робочого циклу — натисканням “EXIT”. Як приклад розглянемо процедуру конфігурування у розділі “Temp Format” (Рис. 4.4). Вибір шкали відображення температури (C, F або K). здійснюється кнопками “SELECT+” та “SELECT-“. При цьому навігаційний покажчик «^» зміщується по горизонталі у відповідний бік. Для підтвердження вибору та запису стану прапорця flagT використовується клавіша “MENU/ENTER”, а повернення до попереднього стану меню виконується натисканням “EXIT”. За ідентичним алгоритмом налаштовується параметр flagP, що визначає формат виводу атмосферного тиску в розділі “Pressure Format” (Рис. 4.5). Функція “SAVE PRESSURE” (Рис. 4.6) призначена для реєстрації опорного показника тиску. Це критично важливо для режиму висотоміра (встановлення нульової відмітки) та для

коректної роботи алгоритму метеопрогнозування. У розділі “MODE ALTITUDE” (Рис. 4.6) натискання “MENU/ENTER” призводить до встановлення змінної mode=0 та інверсії стану прапорця flagS. Залежно від поточного значення цього маркера змінюється алгоритм розрахунків та індикація на екрані: При flagS=0 розраховується альтитуда (позначається символом “A”); При flagS=1 обчислюється відносна висота (індикація символом “H”). Пункт меню “MODE WEATHER” (Рис.4.7) призначений для активації режиму барометричного прогнозування. При натисканні кнопки “MENU/ENTER” відкривається діалогове вікно вибору поточного стану погодних умов. Користувачеві пропонується встановити один із трьох варіантів: “R/S” (дощ або сніг), “XXX” (хмарність) чи “SUN” (сонячна погода). Наприклад, якщо на дворы сонячно то вибираэмо пункт «SUN» . Навігація між опціями реалізована за допомогою клавіш “SELECT-“ та “SELECT+”, а запис обраного параметра у пам’ять здійснюється повторним натисканням “MENU/ENTER”. Результат вибору у формі цифрового коду (0, 1 або 2) записується у змінну flagWX, яка згодом використовується програмним алгоритмом для подальшого аналізу. Повернення до попереднього рівня ієрархії меню виконується клавішею “EXIT”. Функціонування приладу в режимі метеорологічного прогнозування базується на комплексному аналізі температурних показників, динаміки атмосферного тиску та початкового стану неба, заданого параметром flagWX (0 - опади, 1 - хмарність, 2 - ясно). Ключовим фактором для прогнозу є величина deltaP, що відображає зміну тиску в Паскалях за розрахунковий період. Встановлено, що коливання тиску в межах 150...250 Па свідчать про ймовірну зміну погоди. Наприклад, за умови сонячної погоди падіння тиску вказує на наближення хмарності, а при зниженні понад 250 Па - на високу ймовірність опадів. Вид опадів визначається за температурним показником: додатні значення передбачають дощ, а від’ємні – снігопад.

Програмна реалізація враховує всі можливі сценарії (Рис. 4.8 та 4.9). Якщо при початковому значенні flagWX=0 зафіксовано зростання тиску на

250 Па, система прогнозує прояснення (повідомлення “SUNNY”). Зростання на меншу величину (близько 160 Па) інтерпретується як мінлива хмарність без опадів (“CLOUDY”). При незначних коливаннях тиску стан погоди вважається стабільним, проте враховується зміна фазового стану опадів (“SNOW” або “RAIN”) залежно від переходу температури (температура менше нуля). Аналогічно відбувається аналіз для flagWX=1 та flagWX=2, Рис.4.9.

4.2. Розробка програмних модулів для роботи з цифровим МЕМС давачем BMP180

Взаємодія з модулем BMP180 реалізована за допомогою набору розроблених функцій, що забезпечують:

- отримання заводських калібрувальних параметрів;
- зчитування сирих (некомпенсованих) даних про тиск та температуру;
- програмне обчислення реального атмосферного тиску та поточної висоти;
- низькорівневий обмін даними (запис та читання окремих байтів) через інтерфейс давача.

```
void bmp180Calibration() // функція читання калібрувальних коефіцієнтів
{
  unsigned char msb, lsb;

    msb = BMP180ReadByte(0xAA); lsb = BMP180ReadByte(0xAB);
   ac1 = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xAC); lsb = BMP180ReadByte(0xAD);
   ac2 = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xAE); lsb = BMP180ReadByte(0xAF);
   ac3 = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xB0); lsb = BMP180ReadByte(0xB1);
   ac4 = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xB2); lsb = BMP180ReadByte(0xB3);
   ac5 = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xB4); lsb = BMP180ReadByte(0xB5);
   ac6 = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xB6); lsb = BMP180ReadByte(0xB7);
   b1  = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xB8); lsb = BMP180ReadByte(0xB9);
   b2  = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xBA); lsb = BMP180ReadByte(0xBB);
   mb  = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xBC); lsb = BMP180ReadByte(0xBD);
   mc  = ((int)msb<<8)+((int)lsb);
    msb = BMP180ReadByte(0xBE); lsb = BMP180ReadByte(0xBF);
   md  = ((int)msb<<8)+((int)lsb);
 }

int BMP180ReadUT()   // функція читання некомпенсованого значення температури
{
  unsigned char msb, lsb;

 BMP180WriteByte(0xF4, 0x2E);  // Write 0x2E into Register 0xF4
 delay_ms(15); // Wait at least 4.5ms

 msb = BMP180ReadByte(0xF6); lsb = BMP180ReadByte(0xF7);
 ut= ((int)msb<<8)+((int)lsb);
 return(ut);
}

// Обчислення temperature ut.
// Отримане значення в  0.1 deg C
short BMP180GetTemperature(int ut)
{
 long x1, x2;

 x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
 x2 = ((long)mc << 11)/(x1 + md);
 b5 = x1 + x2;
 return ((b5 + 8)>>4);
}
```

```
unsigned long int bmp180ReadUP() // функція читання некомпенсованого значення тиску
{
 unsigned char msb, lsb, xlsb;
 unsigned long int up;

 // Записати  0x34+(OSS<<6) в register 0xF4
 // Request a pressure reading w/ oversampling setting
BMP180WriteByte(0xF4, (0x34 + (OVS  S<<6)) );

 // Чекати на перетворення, часова затримка залежить  від OSS
switch (OVS_S)
 {
   case 0: delay_ms(5);  break; // Ultra low power (1 internal samples) - час перетворення 4,5 ms
   case 1: delay_ms(8);  break; // Standard (2 internal samples) - час перетворення 7,5 ms
   case 2: delay  ms(14); break; // HIGH(4 internal samples) - час перетворення 13,5 ms
   case 3: delay_ms(26); break; // ULTRA_HIGH (8 internal samples) - час перетворення 22,5 ms
 }

  // Читати register 0xF6 (MSB), 0xF7 (LSB), and 0xF8 (XLSB)
  msb  = BMP180ReadByte(0xF6);
  lsb  = BMP180ReadByte(0xF7);
  xlsb = BMP180ReadByte(0xF8);
  up = ((((long)msb <<16) | ((long)lsb <<8) | ((long)xlsb)) >> (8-OVS_S));
  return(up);  // не компенсоване значення тиску
}

long bmp180GetPressure(unsigned long up)  // функція  обчислення атмосферного  тиску
//  Обчислення тиску за даними up
//  Калібрувальні коефіцієнти мають бути відомі
// Значення температури отримати першим для обчислення b5
// Значення тиску отримаємо в  Pa.
{
 long x1, x2, x3, b3, b6, p;
 unsigned long b4, b7;

 b6 = b5 - 4000;
 // Обчисмлюємо B3
 x1 = (b2 * (b6 * b6)>>12)>>11;
 x2 = (ac2 * b6)>>11;
 x3 = x1 + x2;
 b3 = (((((long)ac1)*4 + x3)<<OVS_S) + 2)>>2;

 // Обчисмлюємо B4
 x1 = (ac3 * b6)>>13;
 x2 = (b1 * ((b6 * b6)>>12))>>16;
 x3 = ((x1 + x2) + 2)>>2;
 b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;
 b7 = ((unsigned long)(up - b3) * (50000>>OVS_S));
 if (b7 < 0x80000000)
   p = (b7<<1)/b4;
 else
   p = (b7/b4)<<1;
 x1 = (p>>8) * (p>>8);
 x1 = (x1 * 3038)>>16;
 x2 = (-7357 * p)>>16;
 p += (x1 + x2 + 3791)>>4;
 return p; // Значення тиску отримаємо в Pa
}
```

```
float  bmp180Altitude(long int  pressure) // функція обчислення висоти
  { float  altitude; float  temp;
   temp= (float)(pressure*0.01); temp= temp/1013.25;
   temp = 1-pow(temp, 0.19029);  altitude = 44330*temp; //обчислити  altitude в метрах
   return altitude;   }
  // функція читання байта з BMP180
  char BMP180ReadByte(unsigned char address)
    {
     unsigned char data;  // отримані дані з  BMP180
     i2c_start();
     i2c_write(BMP180W);   // адрес BMP180 write на  на шині i2c
     i2c_write(address);
     i2c_start();
     i2c_write(BMP180R );   // адрес BMP180 read на  на шині i2c
     data=i2c_read(0);
     i2c_stop();
     return(data);
    }

  // функція запису байта в BMP180
  void BMP180WriteByte(unsigned char address, unsigned char data)
  {
    i2c_start();
    i2c_write(BMP180W);  // адрес BMP180 write на  на шині i2c
    i2c_write(address);
    i2c_write(data);    // дані для запису
    i2c_stop();
  }
```

4.3. Розробка програмних модулів меню налаштувань

Для реалізації інтерфейсу користувача та меню налаштувань розроблено наступні функції:

- функція читання статусу кнопок;
  unsigned char getBtnStatus(unsigned char BUTTON_ID) // функція статус клавіші
// читати попередній статус клавіші
  unsigned char getPrevBtnStatus(unsigned char BUTTON_ID)
- функція форматування даних для виводу температури;
  void setTempUnitFormat(void) // пункт меню - формат виводу температури
- функція форматування даних для виводу тиску;
  void setPressUnitFormat(void) // пункт меню - формат виводу тиску

- функція налаштування біжучого значення для погоди;
  void setweather(void) // пункт меню – налаштування біжучого значення погоди
- меню – головна функція керування структурою меню.
  void mainMenu(void) // меню – функція

4.4. Розробка програмного модуля для роботи з рідкокристалічним символьним дисплеєм

Для виводу інформації на РКД створено функції які інтегровані в бібліотеку “mylcd.c”:

```
void lcd_clear (void); // очистити РКД
void lcd_strobe (void); // строб в РКД
void lcd_putchar (unsigned char c);
void lcd_init (void); // ініціалізація РКД
void lcd_puts(unsigned char *str); // вивід string на РКД
void lcd_write (unsigned char c); // запис даних в РКД
void lcd_putsf (unsigned char __flash *str); // вивід з FLASH РКД символу
void cursor_on (void); // виводити курсор
void lcd_cmd (unsigned char c); // запис команди в РКД
void cursor_off (void); // не виводити курсор
```

4.5. Програмна реалізація головного алгоритму роботи цифрового висотоміра

У додатку 2 міститься програмний код, що реалізує основний алгоритм функціонування цифрового висотоміра. У результаті компіляції проєкту в середовищі CodeVisionAVR було отримано прошивку у форматі .hex, призначену для мікроконтролера ATmega128. Цей файл завантажується в пам’ять МК за допомогою програматора. Попередня перевірка працездатності спроектованого висотоміра та верифікація керуючої програми здійснювалися в програмному середовищі Proteus ISIS. Результати моделювання роботи пристрою представлені на відповідних рисунках нижче.

Вмикаємо цифровий висотомір. Спочатку відбувається: ініціалізація РКД, інтерфейсу I^2C, змінних та кнопок управління. Послідовно на РКД виводяться наступні повідомлення (Рис.4.10).

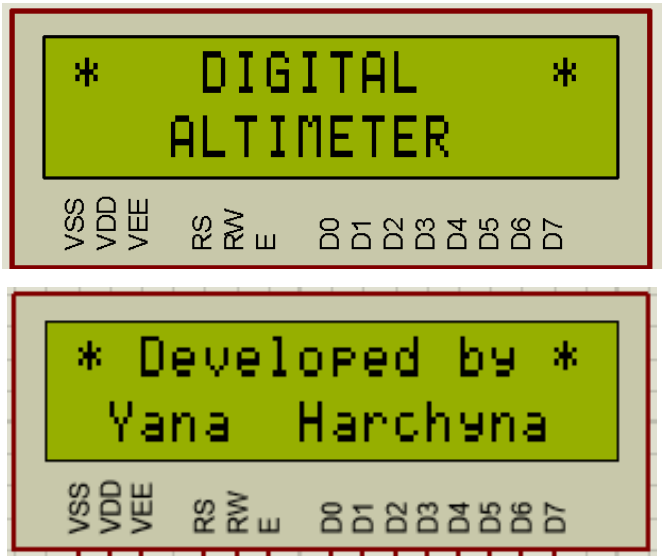


Рис. 4.10. Моделювання роботи цифрового висотоміра в Proteus ISIS – ініціалізація системи

Після ініціалізації, відбувається зчитування по інтерфейсу I^2C з датчика BMP180 – даних, їх обробка і вивід інформації на РКД (Рис.4.11).

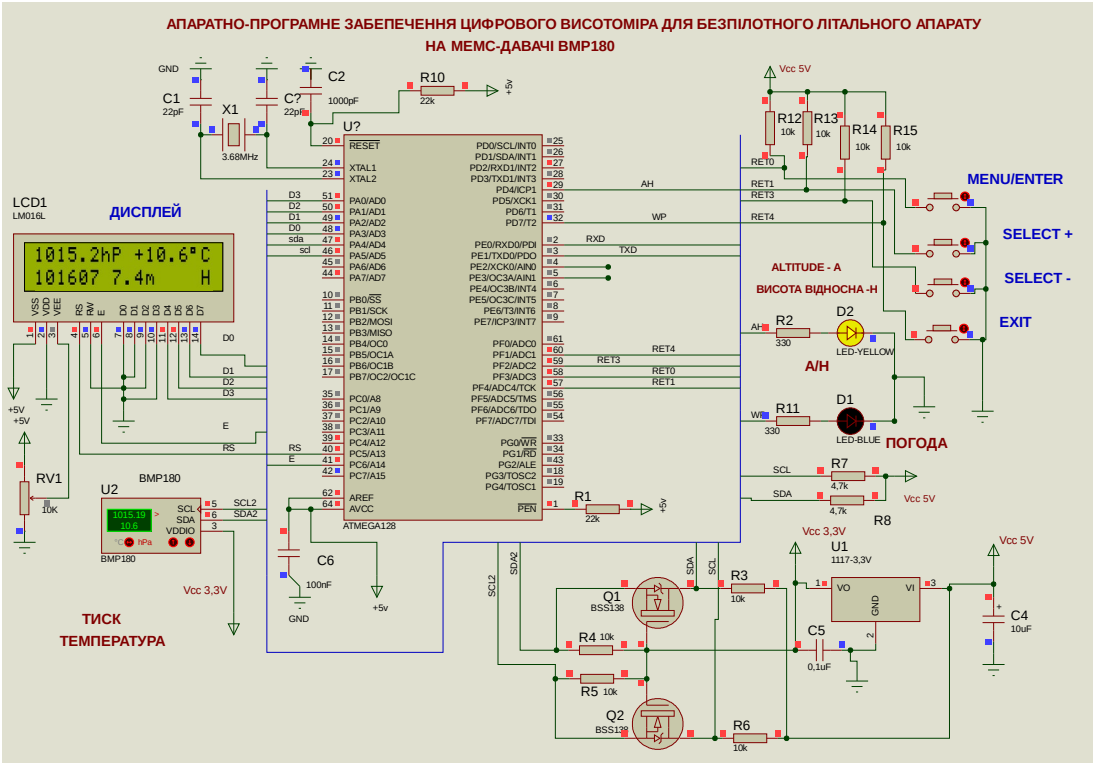


Рис. 4.11. Моделювання роботи цифрового висотоміра в Proteus ISIS. Робочий режим

На рідкокристалічному дисплеї відображається поточний тиск, температура повітря (верхній рядок), стартове значення тиску та режим (H) - вимірювання відносної висоти в метрах (нижній рядок). Базові налаштування

приладу передбачають відображення тиску в Паскалях (Па), а температури - у градусах Цельсія (°C). За замовчуванням активовано функцію розрахунку відносної висоти в метрах. Активність цього режиму підтверджується світінням жовтого світлодіода. Система працює в робочому режимі. Перехід до “Головного меню” (Рис. 4.12) здійснюється за допомогою кнопки “MENU/ENTER”.

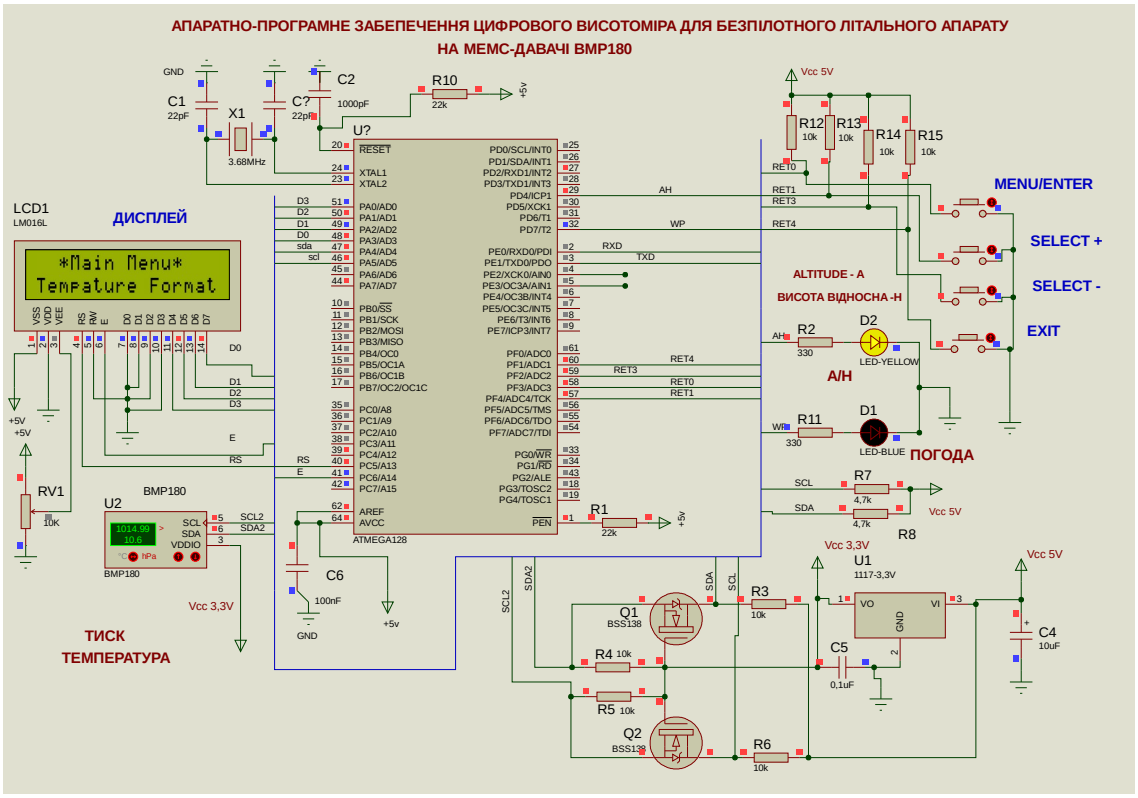


Рис.4.12. Моделювання роботи цифрового висотоміра. Головне меню

Головне меню конфігурації містить пункти , що представлені на Рис. 4.13 – 4.17. Навігація між пунктами виконується кнопками “SELECT–” та “SELECT+”, а підтвердження вибору – натисканням “MENU/ENTER”. Повернутися до стандартного режиму роботи можна за допомогою клавіші “EXIT”.

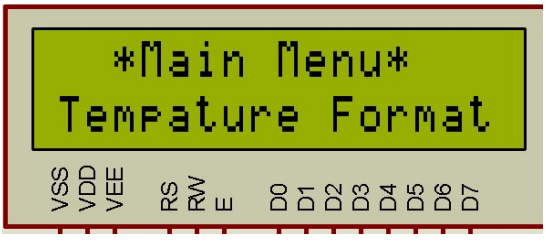


Рис. 4.13. Пункт меню налаштування формату температури для виводу на РКД

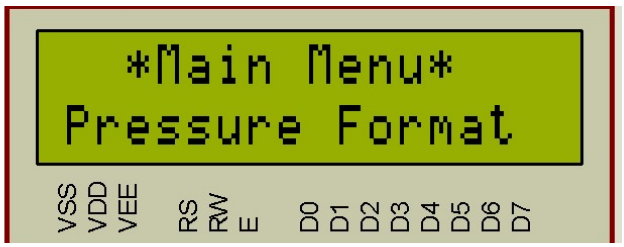


Рис. 4.14. Пункт меню налаштування формату тиску для виводу на РКД

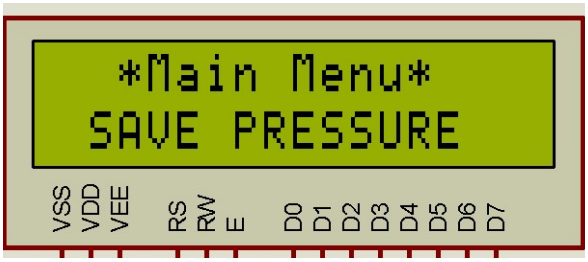


Рис. 4.15. Пункт меню для збереження біжучого значення тиску

Рис. 4.16. Пункт меню для налаштування режиму роботи висотоміра

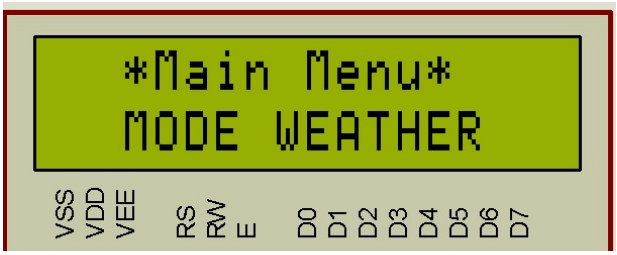


Рис. 4.17. Пункт меню для налаштування режиму "погода"

Процес налаштування одиниць вимірювання температури для відображення на РК-дисплеї змодельовано на Рис. 4.18. Зміна формату ("C" – Цельсій, "F" - Фаренгейт або "K" - Кельвін) здійснюється шляхом переміщення курсору "^" за допомогою кнопок "SELECT–" та "SELECT+". Для підтвердження та збереження обраного варіанта слід натиснути "MENU/ENTER", а для повернення з цього підменю передбачена клавіша "EXIT".

Рис. 4.18. Моделювання вибору формату температури

Способи відображення температурних показників у шкалах Фаренгейта та Кельвіна на рідкокристалічному дисплеї наведено відповідно на Рис. 4.19 та 4.20.

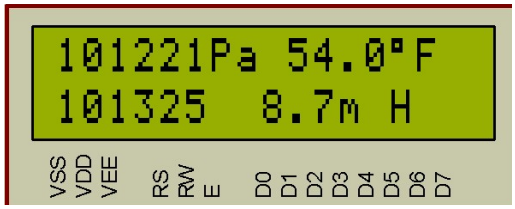


Рис. 4.19. Температура у градусах Фаренгейта

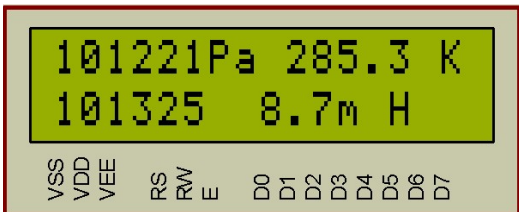


Рис. 4.20. Температура у Кельвінах

На Рис. 4.21 відображено процес налаштування одиниць вимірювання тиску для їх подальшої візуалізації на екрані. Користувач може обрати потрібний формат – Паскалі (Pa), гектопаскалі (hPa) або міліметри ртутного стовпа (mmHg) – переміщуючи покажчик “^” кнопками “SELECT–“ та “SELECT+”. Обраний параметр фіксується в системі натисканням “MENU/ENTER”, а вихід із цього підменю здійснюється клавішею “EXIT”.

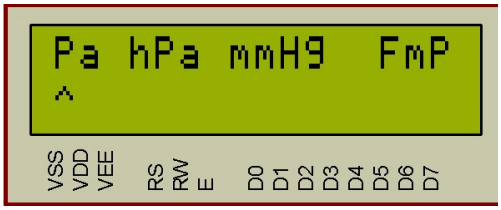


Рис. 4.21. Вибір одиниць вимірювання тиску

На Рис.4.22 та 4.24 відображено на РКД тиск у Паскалях та гектоПаскалях та мм.рт. ст.., відповідно.

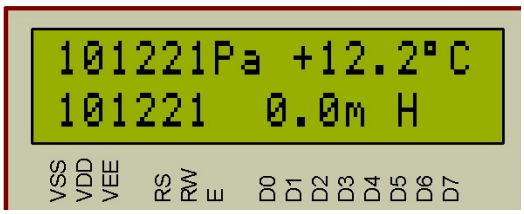


Рис. 4.22. Одиниці тиску у Паскалях

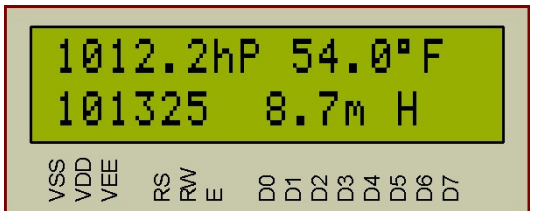


Рис. 4.23. Одиниці тиску у гектоПаскалях

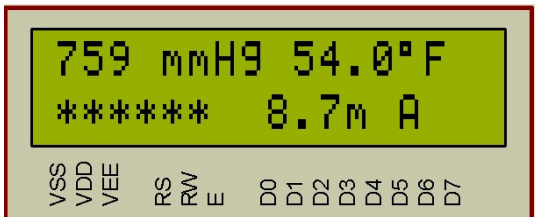


Рис. 4.24. Одиниці тиску у мм.рт.ст.

Цифровий висотомір передбачає два основні режими: вимірювання альтитуди та прогнозування погоди. Для роботи з приладом у режимі альтиметра необхідно перейти до пункту “MODE ALTITUDE”. Натисканням кнопки “MENU/ENTER” можна встановити режим вимірювання: абсолютну висоту (“A”) або відносну (“H”). Повернення до попереднього рівня меню здійснюється клавішею “EXIT”. При виборі абсолютної висоти (Рис. 4.24) на рідкокристалічному екрані відображається поточне значення в метрах відносно рівня моря (базовий тиск за замовчуванням - 101325 Па), що супроводжується індикатором **“A”**. Якщо потрібно визначити відносну висоту, необхідно попередньо зберегти поточний показник тиску як контрольну точку. Це виконується через пункт меню “SAVE PRESSURE” шляхом натискання клавіші “MENU/ENTER” (Рис. 4.25).

Рис. 4.25. Пункт меню для збереження бужучої величини тиску

На Рис.4.26 зображено моделювання роботи цифрового висотоміра в режимі вимірювання відносної висоти. В другому рядку на дисплеї відображається стартове значення тиску на певній (початковій висоті), виміряна відносна висота у метрах а також “H” - підказка (відносна висота). У верхньому рядку на дисплеї відображається тиск у Паскалях, температура у градусах Фаренгейта. Світиться жовтий світлодіод режиму вимірювання висоти.

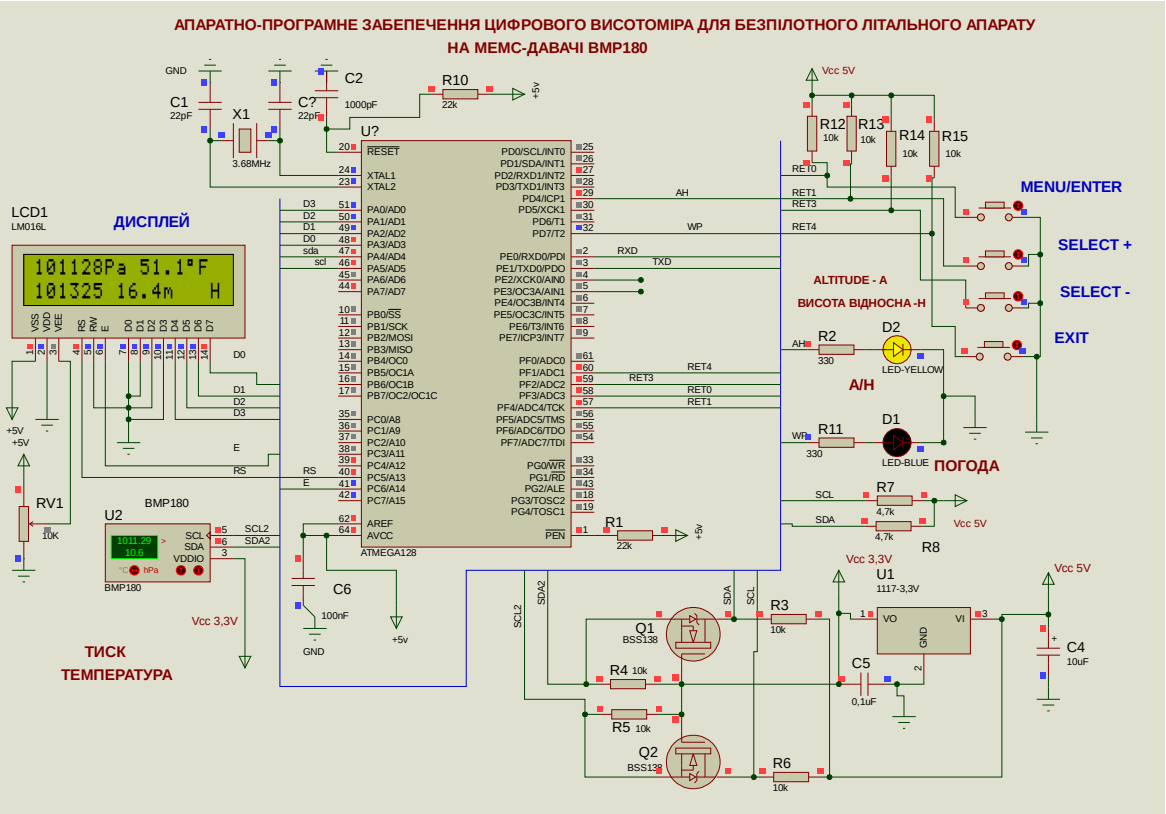


Рис. 4.26. Моделювання роботи цифрового пристрою- вимірювання висоти

Для моделювання функцій прогнозування погоди слід відкрити пункт “MODE WEATHER”, як показано на Рис. 4.27. Вибір актуальних погодних умов виконується шляхом зміщення курсора ““^” у відповідну сторону (вліво або вправо) за допомогою клавіш вибору “SELECT-”, “SELECT+”, (Рис. 4.28).

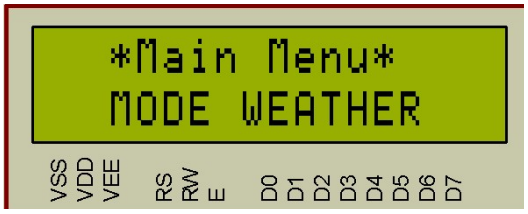


Рис. 4.27. Пункт меню “MODE WEATHER”

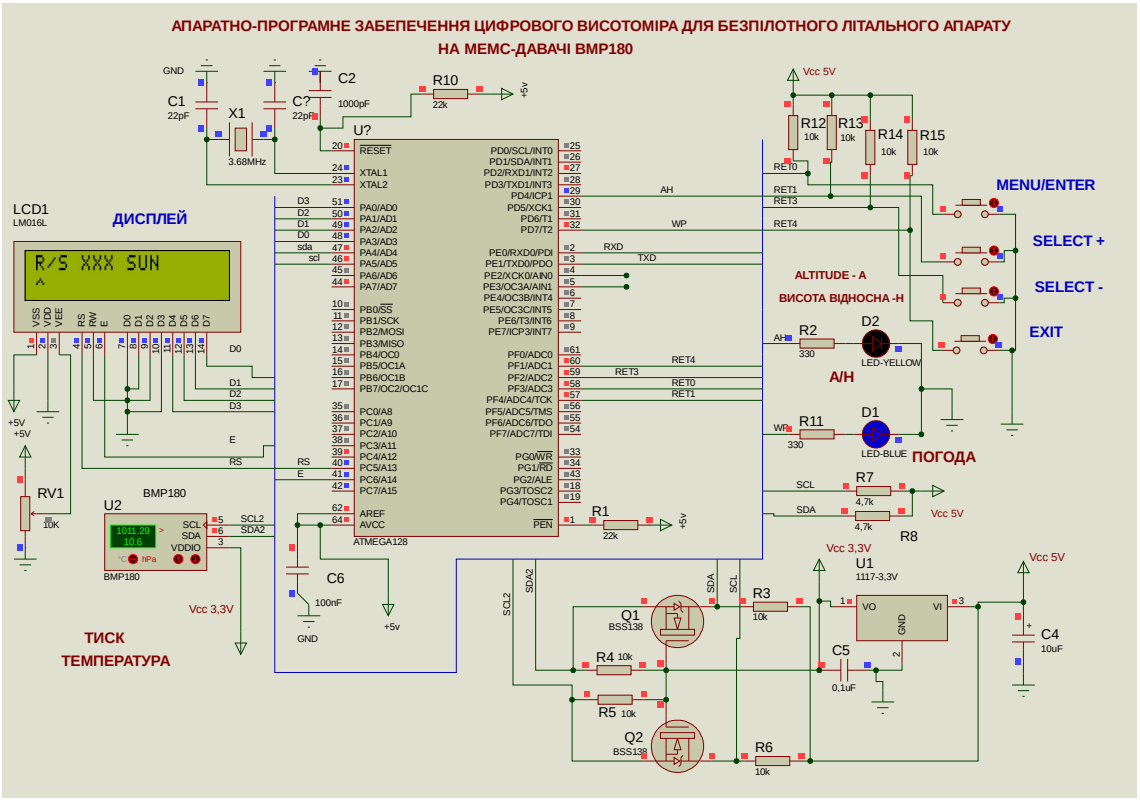


Рис. 4.28. Встановлення наявної погоди

Вибір метеоумов здійснюється серед наступних значень: “R/S” (опади), “XXX” (хмарність) та “SUN” (сонячно). Поточний стан погоди фіксується натисканням “MENU/ENTER”, а для повернення з налаштувань використовується клавіша “EXIT”. Щоб забезпечити точність прогнозу, необхідно зберегти актуальний показник тиску через розділ меню “SAVE PRESSURE” (Рис. 4.25). Активація цього режиму супроводжується синім світлодіодним індикатором “ПОГОДА”, а відповідні дані відображаються на дисплеї (Рис. 4.29 – 4.31).

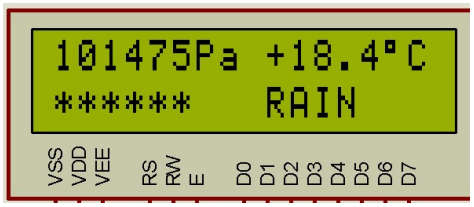


Рис. 4.29. Очікується дощ

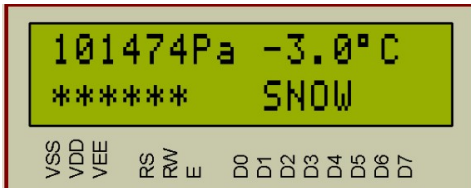


Рис. 4.30. Очікується сніг (якщо мінусова температура)

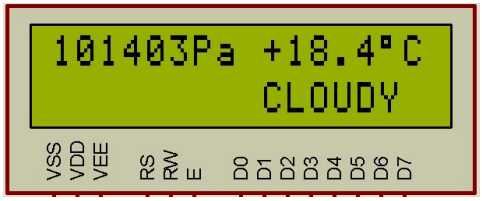


Рис. 4.31. Очікується хмарна погода

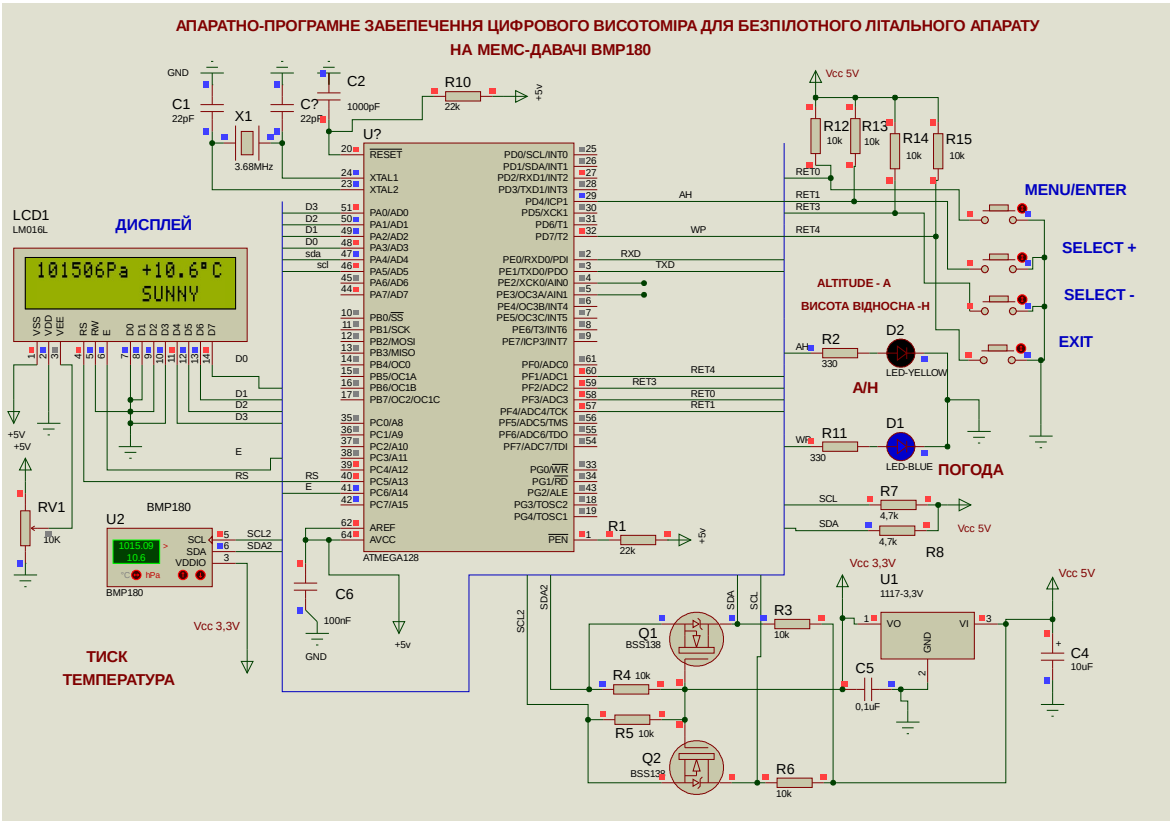


Рис. 4.32. Моделювання прогнозу погоди

Моделювання роботи цифрового висотоміра для безпілотного літального апарату в різних режимах показало коректність роботи апаратного та програмного забезпечення.

ВИСНОВКИ

В дипломній роботі було спроектовано апаратну частину та програмне забезпечення цифрового висотоміра для безпілотного літального апарату (дрона) на МЕМС-давачі BMP180.

Розроблений пристрій вимірює атмосферний тиск, температуру навколишнього середовища, альтитуду (абсолютну висоту над рівнем моря), відносну висоту (в режимі висотоміра), прогнозує ймовірну зміну погоди. Візуалізація даних здійснюється за допомогою рідкокристалічного дисплея (РКД).

Цифровий висотомір має вбудоване меню, що дозволяє обирати режими роботи, налаштувати одиниці виводу виміряних значень метеовеличин (градуси Цельсія/Фаренгейта для температури та гПа/мм рт. ст. для тиску). Для контролю стану системи пристрій має світлодіодну індикацію.

При розробці цифрового висотоміра використано сучасну елементну базу. Головними її елементами є МК AVR, цифровий давач BMP180, рідкокристалічний символьний дисплей.

Етап проектування включав розробку принципової схеми та створення віртуальної моделі в середовищі Proteus VSM. Програмне забезпечення, розроблене мовою С у середовищі CodeVisionAVR, містить спеціалізовані модулі для взаємодії з датчиком BMP180 та керування символьним дисплеєм. Успішне тестування в емуляторі Proteus ISIS підтвердило коректність роботи алгоритмів цифрового висотоміра для безпілотного літального апарату. Робота над проектом дозволила поглибити практичні навички програмування вбудованих систем та проектування цифрової мікропроцесорної техніки.

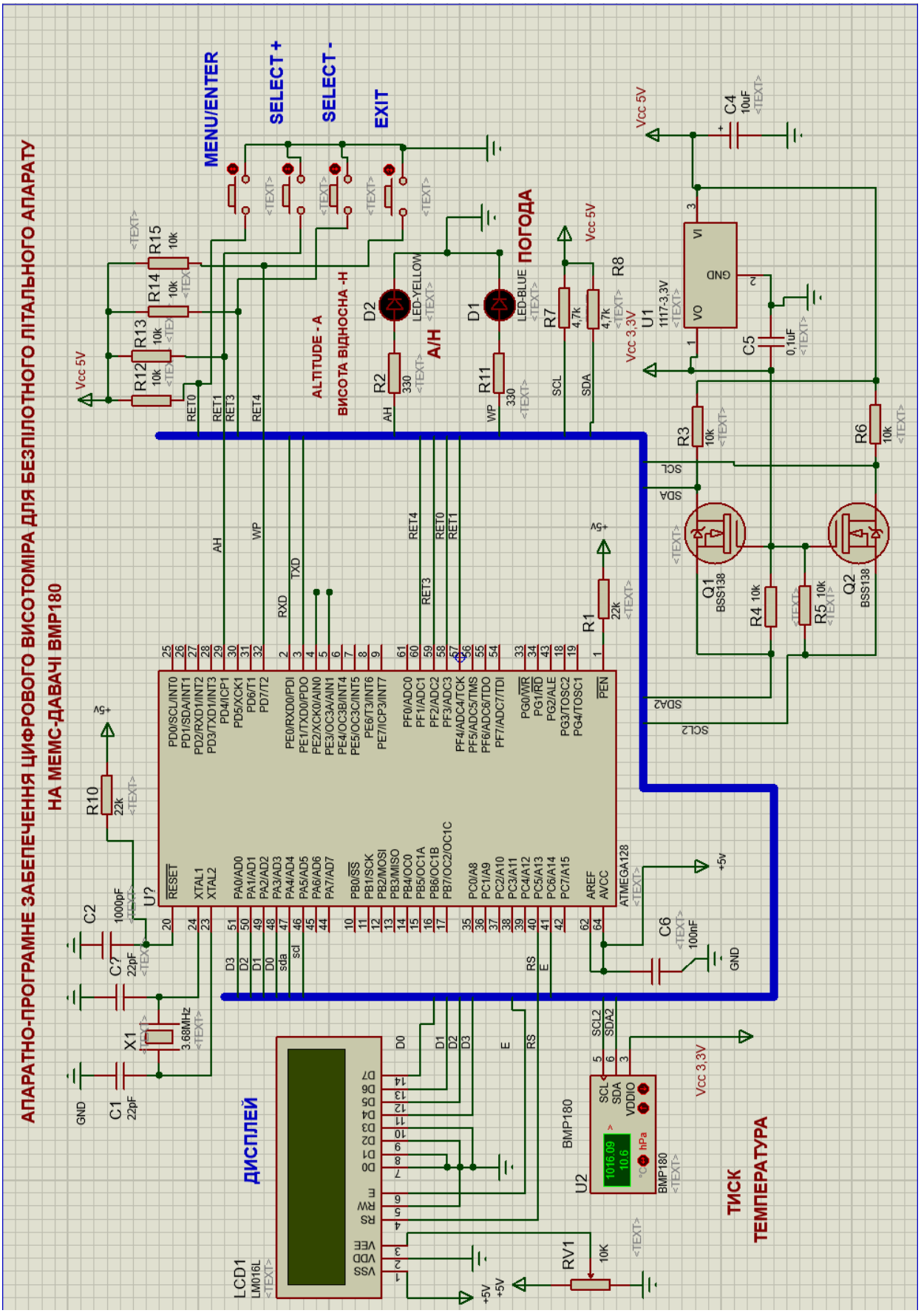
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Електронний ресурс: https://aebst.resource.bosch.com/media/_tech/-media/product_flyer/BST-BMP180-FL000.pdf
2. Elliot Williams. Make. AVR Programming. Learning to Write Software for Hardware. 2014. – 472 с.
3. Joe Padrue. C Programming for microcontrollers. 2005. – 300 с.
4. Barrett S.F., Pack D.J.. Atmel AVR Microcontroller Primer. Programming and Interfacing. 2008. – 198 с.
5. Електронний ресурс з datasheet на мікросхему МК ATMEGA128: https://www.alldatasheet.com/datasheet-pdf/pdf/56260/ATMEL/ATMEGA128.html
6. Електронний ресурс з datasheet на сенсор DS18B20: http://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html.
7. Електронний ресурс на LCD Hitachi HD44780: https://circuitdigest.com/sites/default/files/HD44780U.pdf.
8. Електронний ресурс CodeVisionAVR user manual https://www.thierry-lequeu.fr/data/CodeVisionAVR-3-20-User-Manual.pdf.
9. Електронний ресурс на Proteus VSM: https://www.labcenter.com/-downloads/.

ДОДАТКИ

Додаток 1

Принципова схема цифрового висотоміра



Додаток 2

Головний програмний модуль цифрового висотоміра

```
/* Project : цифровий висотомір на BMP180
Chip type               : ATmega128
Program type            : Application
Clock frequency     : 3,680000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 1024 */
#include <mega128.h>
#include <stdio.h>
#include <i2c.h>
#include <delay.h>
#include <math.h>
#include "mylcd.c"     //  бібліотека для LCD
// I2C Bus functions
#asm
   .equ __i2c_port=0x1B ;PORTA
   .equ __sda_bit=4
   .equ __scl_bit=5
#endasm
#define BMP180R 0xEF   // read- адрес пристрою  на  на шині i2c
#define BMP180W 0xEE   // write-адрес пристрою  на  на шині i2c
#define AC1   0xAA   // старт-адрес коеф. ac1
#define AC2   0xAC   // старт-адрес коеф. ac2
#define AC3   0xAE   // старт-адрес коеф. ac3
#define AC4   0xB0   // старт-адрес коеф. ac4
#define AC5   0xB2   // старт-адрес коеф. ac5
#define AC6   0xB4   // старт-адрес коеф. ac6
#define B1    0xB6   // старт-адрес коеф. b1
#define B2    0xB8   // старт-адрес коеф. b2
#define MB    0xBA   // старт-адрес коеф. mb
#define MC    0xBC   // старт-адрес коеф. mc
#define MD    0xBE   // старт-адрес коеф. md
#define BUTTON_PORT     PORTF    // клавіші на порті F
#define BUTTON_PIN      PINF #define BUTTON_DDR      DDRF
#define MENU_ENTER_BTN_PIN 3        // КН1
#define SELECT_PLUS_BTN_PIN 4       // КН2
#define SELECT_MINUS_BTN_PIN 2   // КН3
#define EXIT_BTN_PIN 1                    // КН4
#define TEMP_UNITS_NUM 3   // кількість пунктів підменю
const unsigned char OVS_S = 2; // Oversampling (0,1,2,3 from ultra low power, to ultra hi-
resolution)
unsigned char PREV_BUTTON_PIN = 0xFF;    // попереднє - початкове значення клавіші
char temp_unit[3] = {'C','F','K'};     //  Цельсій, Фаренгейт, Кельвін
char *ttt[3] = {"FORMAT T", "FmP", " " };  volatile int flagt=0, flagp=0,flags=0, mode=0;
  // mode=1 - " MODE WEATHER  ", mode=0 - " MODE ALTITUDE "
int temp_unit_ind = 0;      // температура
int press_unit_ind = 0;    // тиск
char *press_unit[3] = {"Pa", "hPa", "mmHg"};
char *press_weather[3] = {"R/S", "XXX", "SUN"};      // опади,хмарно,ясно
```

```
volatile int flagwx=2;                                  // 1-дощ.сніг, 2 - хмарно, 3 - сонячно.
volatile long int  pmm=101325,  p2=101325 ;             //  pmm=101325  paw=0,
float ph=20.1, tk=0.1;      // тиск в hPa
float  a0=0.1, a1=0.1;                                  // висоти над рівнем моря
float     dh=0.1;                                       // відносна  зміна висоти
int deltap=0;                                           // зміна тиску
short temperatura=1;                                    // тест сніг- дощ
char  buffer[33];                                       // for LCD convertion
char id = 0xcc;                                         // BMP180 id змінна  має бути 55h
/* Calibration coefficients */
unsigned int ac4=32741, ac5=32757, ac6=23153;
int ac1=408,ac2=-72,ac3=-14383,b1=6190,b2=4,mb=-32768,mc=-8711,md=2868 ;
volatile long int b5 ; volatile unsigned int ut;   // uncompensated temperature value - 0.1 -коеф.
short t;    // температура
long p,upr;    /* тиск  */ float  tf=-1.1;        // pf=2.1,

 // функція статус клавіші
  unsigned char getBtnStatus(unsigned char BUTTON_ID)  // читати статус клавіші
{
        return (!(BUTTON_PIN & (1 << BUTTON_ID)));
}
 unsigned char getPrevBtnStatus(unsigned char BUTTON_ID)  // читати попередній статус
клавіші
{
        return (!(PREV_BUTTON_PIN & (1<< BUTTON_ID)));
}

  void setTempUnitFormat(void)            // підменю - формат виводу температури
    {
     int i, pos_x = 0;                          // pos_x_max = TEMP_UNITS_NUM * 2;
     char oK = 0;
     lcd_clear();      lcd_gotoxy(7,0);
     lcd_puts(ttt[0]);                          //  "FORMAT T"
     lcd_gotoxy(0,0);
     for (i = 0; i < TEMP_UNITS_NUM; i++)        {
        lcd_putchar(temp_unit[i]);
           lcd_putchar(' ');        }
     while(!oK)      {
        PREV_BUTTON_PIN = BUTTON_PIN;
        lcd_gotoxy(pos_x, 1);
        lcd_putchar('^');

        if (getBtnStatus(SELECT_PLUS_BTN_PIN))  {              //   КН2=  "+"
              if (!getPrevBtnStatus(SELECT_PLUS_BTN_PIN))
                {
                  lcd_gotoxy(pos_x, 1);          // рядок 2
              lcd_putchar(' ');
                    if (pos_x < (TEMP_UNITS_NUM - 1)* 2)
                      pos_x += 2;
                    else    pos_x = 0;
            }           }
       if (getBtnStatus(SELECT_MINUS_BTN_PIN))             //  КН3=  "-"
```

```
        {
            if (!getPrevBtnStatus(SELECT_MINUS_BTN_PIN))        {
            lcd_gotoxy(pos_x, 1);          //  рядок 2
            lcd_putchar(' ');
                  if (pos_x ==0)
                    pos_x = (TEMP_UNITS_NUM-1) * 2;
            else     pos_x -= 2;
           }     }
       if (getBtnStatus(MENU_ENTER_BTN_PIN))
        {
         if (!getPrevBtnStatus(MENU_ENTER_BTN_PIN))
          {
            temp_unit_ind = pos_x / 2;                    // значення для прапорця
                  return;          }      }
       if (getBtnStatus(EXIT_BTN_PIN))    {
          if(!getPrevBtnStatus(EXIT_BTN_PIN))     {
            PREV_BUTTON_PIN = BUTTON_PIN;
            return;
           }       }
       flagt=  pos_x/2;             // прапорець формату виводу температури

    }    // end while
}         // end setTempUnitFormat

void setPressUnitFormat(void)           // підменю - формат виводу температури
   {
    int i, pos_x = 0;      // pos_x_max = TEMP_UNITS_NUM * 2;
    char oK = 0;   lcd_clear();  lcd_gotoxy(13,0);  lcd_puts(ttt[1]);      //  "FmP"
    lcd_gotoxy(0,0);
    for (i = 0; i < TEMP_UNITS_NUM; i++)  {
       lcd_puts(press_unit[i]);  lcd_putchar(' ');       }
    while(!oK)      {
       PREV_BUTTON_PIN = BUTTON_PIN;
       lcd_gotoxy(pos_x, 1);         lcd_putchar('^');
       if (getBtnStatus(SELECT_PLUS_BTN_PIN))                //   КН2=  "+"
        {
            if (!getPrevBtnStatus(SELECT_PLUS_BTN_PIN))   {
               lcd_gotoxy(pos_x, 1);          // рядок 2
            lcd_putchar(' ');
                  if (pos_x < (TEMP_UNITS_NUM - 1)* 4)
                    pos_x += 4;
                  else    pos_x = 0;
           }       }
       if (getBtnStatus(SELECT_MINUS_BTN_PIN))             //  КН3=  "-"
        {
            if (!getPrevBtnStatus(SELECT_MINUS_BTN_PIN))     {
            lcd_gotoxy(pos_x, 1);          //  рядок 2
            lcd_putchar(' ');
               if (pos_x ==0)
                 pos_x = (TEMP_UNITS_NUM-1) * 4;
            else   pos_x -= 4;
           }         }
```

```
      if (getBtnStatus(MENU_ENTER_BTN_PIN))       {
         if (!getPrevBtnStatus(MENU_ENTER_BTN_PIN))         {
            press_unit_ind = pos_x / 4;
               return;
           }       }
      if (getBtnStatus(EXIT_BTN_PIN))       {
          if(!getPrevBtnStatus(EXIT_BTN_PIN))
           {
            PREV_BUTTON_PIN = BUTTON_PIN;
            return;             }       }
      flagp=  pos_x/4;          //   прапорець формату виводу тиску
    }    // end while
}         // end setTempUnitFormat

 float  bmp180Altitude(long int  pressure)    // функція вимірювання висоти
  { float  temp; float  altitude;
   temp= (float)(pressure*0.01);
   temp= temp/1013.25;
   temp = 1-pow(temp, 0.19029);
   altitude = 44330*temp;     //get altitude в метрах
   return altitude;
  }

 void setweather(void)           // підменю - формат виводу погоди
   {
    int i, pos_x = 0;      // pos_x_max = TEMP_UNITS_NUM * 2;
    char oK = 0;
    lcd_clear();   lcd_gotoxy(13,0);      lcd_gotoxy(0,0);
    for (i = 0; i < TEMP_UNITS_NUM; i++)    {
       lcd_puts(press_weather[i]); // вивід "SUNNY", "CLOUD", "PREC";
       lcd_putchar(' ');       }
    while(!oK)    {
       PREV_BUTTON_PIN = BUTTON_PIN;
       lcd_gotoxy(pos_x, 1);
       lcd_putchar('^');
        if (getBtnStatus(SELECT_PLUS_BTN_PIN))         //   КН2=  "+"
         {
             if (!getPrevBtnStatus(SELECT_PLUS_BTN_PIN))      {
                lcd_gotoxy(pos_x, 1);          // рядок 2
             lcd_putchar(' ');
                   if (pos_x < (TEMP_UNITS_NUM - 1)* 4)
                     pos_x += 4;
                   else    pos_x = 0;
            }
         }
       if (getBtnStatus(SELECT_MINUS_BTN_PIN))             //  КН3=  "-"
        {
            if (!getPrevBtnStatus(SELECT_MINUS_BTN_PIN))
           {     lcd_gotoxy(pos_x, 1);          //  рядок 2
            lcd_putchar(' ');
               if (pos_x ==0)
                 pos_x = (TEMP_UNITS_NUM-1) * 4;
```

```
            else   pos_x -= 4;
          }       }
       if (getBtnStatus(MENU_ENTER_BTN_PIN))
         {
          if (!getPrevBtnStatus(MENU_ENTER_BTN_PIN))
           {
            press_unit_ind = pos_x / 4;
               return;
           }
         }
       if (getBtnStatus(EXIT_BTN_PIN))       {
          if(!getPrevBtnStatus(EXIT_BTN_PIN))
           {          PREV_BUTTON_PIN = BUTTON_PIN;
            return;            }
         }
       flagwx=  pos_x/4;         //   прапорець формату виводу погоди
    }    // end while
}         // end setweather

 void mainMenu(void)          // меню
{
 char *menu_items[5] = {"Tempature Format", "Pressure Format ", " SAVE PRESSURE  "," 
MODE WEATHER ", " MODE ALTITUDE  "};
 char menu_title[] = " *Main Menu*";              // Units Conversion
 unsigned char selected = 0;

 while(1)
 {
  PREV_BUTTON_PIN = BUTTON_PIN;
  lcd_gotoxy(1,0);
  lcd_puts(menu_title);
  lcd_gotoxy(0,1);
  lcd_puts(menu_items[selected]);

  if(getBtnStatus(SELECT_PLUS_BTN_PIN))                //   КН2=  "+"
    {
     if(!getPrevBtnStatus(SELECT_PLUS_BTN_PIN))
       if(selected == 4) selected = 0;            // селектор пунктів меню 0,1,2,3,4 =char 
*menu_items[5]
       else selected++;
    }
  if(getBtnStatus(SELECT_MINUS_BTN_PIN))               //  КН3=  "-"
  {
   if (!getPrevBtnStatus(SELECT_MINUS_BTN_PIN))
     if (selected == 0) selected = 4;           // selected пунктів меню 0,1,2,3,4 =char 
*menu_items[5]
      else selected--;
  }
  if(getBtnStatus(MENU_ENTER_BTN_PIN))                 //  КН1="menu"
  if(!getPrevBtnStatus(MENU_ENTER_BTN_PIN))
    {
          switch(selected)    {
```

```
        case 0: setTempUnitFormat();   break;
              case 1: setPressUnitFormat();  break;       //    setPressUnitFormat();
              case 2:
              lcd_gotoxy(15,1);
              flags=1;lcd_putsf("Y"); p2=p;     //   зберегти значення тиску і вивести на LCD
              delay_ms(500);
              break;
              case 3: mode=1; PORTD.4=0;  PORTD.7=1;    delay_ms(500);    //   режим погода;
lcd_gotoxy(15,1); lcd_putsf("W");
                 setweather();              // функція параметри погоди
                 break;
              case 4:
                   mode=0; PORTD.4=1;PORTD.7=0; lcd_gotoxy(15,1);
                  if (flags==0) {flags=1;lcd_putsf("H"); }
                  else {flags=0;lcd_putsf("A");};
                 delay_ms(500);
                 break;  // режим висота
              default: break;
            }
       lcd_clear();
     }
    if (getBtnStatus(EXIT_BTN_PIN))                    //  КН4="exit"
      {
       if (!getPrevBtnStatus(EXIT_BTN_PIN))
         {
         PREV_BUTTON_PIN = BUTTON_PIN;
         return;
         }
      }
  }     // end while
}       // end mainMenu
/* read a byte from the BMP180 */
 char BMP180ReadByte(unsigned char address)
  {
    unsigned char data;  // отримані дані з  BMP180
    i2c_start();
    i2c_write(BMP180W);  // адрес BMP180 write на  на шині i2c
    i2c_write(address);
    i2c_start();
    i2c_write(BMP180R ); // адрес BMP180 read на  на шині i2c
    data=i2c_read(0);
    i2c_stop();
    return(data);
  }
/* write a byte to the BMP180 */
void BMP180WriteByte(unsigned char address, unsigned char data)
{
   i2c_start();
   i2c_write(BMP180W);  // адрес BMP180 write на  на шині i2c
   i2c_write(address);
   i2c_write(data);    // дані для запису
   i2c_stop();
```

```
}
/* read Calibration coefficients */
void bmp180Calibration()
{
  unsigned char msb, lsb;
    msb = BMP180ReadByte(0xAA); lsb = BMP180ReadByte(0xAB);
   ac1 = ((int)msb<<8)+((int)lsb);                                    //int  511
    msb = BMP180ReadByte(0xAC); lsb = BMP180ReadByte(0xAD);
   ac2 = ((int)msb<<8)+((int)lsb);                                    //int  -72
    msb = BMP180ReadByte(0xAE); lsb = BMP180ReadByte(0xAF);
   ac3 = ((int)msb<<8)+((int)lsb);                                    //int  -14383
    msb = BMP180ReadByte(0xB0); lsb = BMP180ReadByte(0xB1);
   ac4 = ((int)msb<<8)+((int)lsb);                                    // int 32741
    msb = BMP180ReadByte(0xB2); lsb = BMP180ReadByte(0xB3);
   ac5 = ((int)msb<<8)+((int)lsb);                                    //  int 32757
    msb = BMP180ReadByte(0xB4); lsb = BMP180ReadByte(0xB5);
   ac6 = ((int)msb<<8)+((int)lsb);                                    // int 23153
    msb = BMP180ReadByte(0xB6); lsb = BMP180ReadByte(0xB7);
   b1  = ((int)msb<<8)+((int)lsb);                                    //int 6190
    msb = BMP180ReadByte(0xB8); lsb = BMP180ReadByte(0xB9);
   b2  = ((int)msb<<8)+((int)lsb);                                    //int 4
    msb = BMP180ReadByte(0xBA); lsb = BMP180ReadByte(0xBB);
   mb  = ((int)msb<<8)+((int)lsb);                                    //int -32768
    msb = BMP180ReadByte(0xBC); lsb = BMP180ReadByte(0xBD);
   mc  = ((int)msb<<8)+((int)lsb);                                    //int -8711
    msb = BMP180ReadByte(0xBE); lsb = BMP180ReadByte(0xBF);
   md  = ((int)msb<<8)+((int)lsb);                                    //int 2868
 }
/* Read the uncompensated temperature value */
int BMP180ReadUT()
{
  unsigned char msb, lsb;
 BMP180WriteByte(0xF4, 0x2E);  // Write 0x2E into Register 0xF4
 delay_ms(15); // Wait at least 4.5ms
 msb = BMP180ReadByte(0xF6); lsb = BMP180ReadByte(0xF7);
 ut= ((int)msb<<8)+((int)lsb);
 return(ut);
}
// Calculate temperature given ut.
// Value returned will be in units of 0.1 deg C
short BMP180GetTemperature(int ut)
{
 long x1, x2;
 x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
 x2 = ((long)mc << 11)/(x1 + md);
 b5 = x1 + x2;
 return ((b5 + 8)>>4);
}
/* Read the uncompensated pressure value */
unsigned long int bmp180ReadUP()
{
 unsigned char msb, lsb, xlsb;
```

```
 unsigned long int up;
 // Write 0x34+(OSS<<6) into register 0xF4
 // Request a pressure reading w/ oversampling setting
BMP180WriteByte(0xF4, (0x34 + (OVS_S<<6)) );
 // Wait for conversion, delay time dependent on OSS
 switch (OVS_S)
 {
    case 0: delay_ms(5);  break; // Ultra low power (1 internal samples) - час перетворення 4,5
ms
    case 1: delay_ms(8);  break; // Standard (2 internal samples) - час перетворення 7,5 ms
    case 2: delay_ms(14); break; // HIGH(4 internal samples) - час перетворення 13,5 ms
    case 3: delay_ms(26); break; // ULTRA_HIGH (8 internal samples) - час перетворення 22,5
ms
 }
  // Read register 0xF6 (MSB), 0xF7 (LSB), and 0xF8 (XLSB)
  msb  = BMP180ReadByte(0xF6);
  lsb  = BMP180ReadByte(0xF7);
  xlsb = BMP180ReadByte(0xF8);
  up = ((((long)msb <<16) | ((long)lsb <<8) | ((long)xlsb)) >> (8-OVS_S)); // uncompensated
pressure value
  return(up);
}
// Calculate pressure given up
// b5 is also required so bmp180GetTemperature(...) must be called first.
// Value returned will be pressure in units of Pa.
long bmp180GetPressure(unsigned long up)
{
 long x1, x2, x3, b3, b6, p;   unsigned long b4, b7;   b6 = b5 - 4000;
 // Calculate B3
 x1 = (b2 * (b6 * b6)>>12)>>11;   x2 = (ac2 * b6)>>11;   x3 = x1 + x2;
 b3 = (((((long)ac1)*4 + x3)<<OVS_S) + 2)>>2;
 // Calculate B4
 x1 = (ac3 * b6)>>13;   x2 = (b1 * ((b6 * b6)>>12))>>16;   x3 = ((x1 + x2) + 2)>>2;
 b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;   b7 = ((unsigned long)(up - b3) *
(50000>>OVS_S));
 if (b7 < 0x80000000)     p = (b7<<1)/b4;
 else     p = (b7/b4)<<1;
 x1 = (p>>8) * (p>>8);   x1 = (x1 * 3038)>>16;   x2 = (-7357 * p)>>16;   p += (x1 + x2 +
3791)>>4;
 return p;
}

void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
PORTA=0x0F;DDRA=0x0F; PORTC=0xF0; DDRC=0xF0;
PORTB=0x00;   DDRB=0x00; // Port B initialization
PORTD=0x00;  DDRD=0x00; // Port D initialization
PORTE=0x00;  DDRE=0x00; // Port E initialization
PORTF=0x00;  DDRF=0x00;  // Port F initialization
```

```
PORTG=0x00;  DDRG=0x00;  // Port G initialization
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;    TCCR0=0x00; TCNT0=0x00;   OCR0=0x00;
// Clock value: Timer 1 Stopped
TCCR1A=0x00;   TCCR1B=0x00;  TCNT1H=0x00;   TCNT1L=0x00;
ICR1H=0x00;    ICR1L=0x00;  OCR1AH=0x00;   OCR1AL=0x00;
OCR1BH=0x00;   OCR1BL=0x00; OCR1CH=0x00;   OCR1CL=0x00;
// Clock value: Timer 2 Stopped
TCCR2=0x00;  TCNT2=0x00; OCR2=0x00;
// Clock value: Timer 3 Stopped
TCCR3A=0x00;   TCCR3B=0x00; TCNT3H=0x00;   TCNT3L=0x00;
ICR3H=0x00;    ICR3L=0x00;  OCR3AH=0x00;   OCR3AL=0x00;
OCR3BH=0x00;   OCR3BL=0x00; OCR3CH=0x00;   OCR3CL=0x00;
// External Interrupt(s) initialization
// INT0: Off  INT1: Off  INT2: Off  // INT3: Off  INT4: Off  INT5: Off
EICRA=0x00;   EICRB=0x00;  EIMSK=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;    ETIMSK=0x00;
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80; SFIOR=0x00;
DDRC = 0x60;            // Port C RC5, RC6 ->output used for LCD
DDRA = 0x0F;            // D0-D3 -> output used for LCD
DDRA.7=1;    // LED1 -> output used for LCD
PORTA.7=1;      // LED1=1 output used for LCD
DDRC.7=1;       // LWR/-> output used for LCD
PORTC.7=0;      // LWR/=0  output used for LCD
DDRD.2 = 1;     // SD5/-> включити живлення +5X LCD
PORTD.2=1;      // SD5/=1 включити живлення +5X LCD
DDRD.4 = 1;  PORTD.4=1; DDRD.7 = 1; PORTD.7=0;
lcd_init ();    // lcd  initialization
lcd_clear ();   lcd_gotoxy(0,0);
lcd_putsf("*   DIGITAL    *");
lcd_gotoxy(0,1);  lcd_putsf("   ALTIMETER   ");   delay_ms(900);
lcd_clear ();   lcd_gotoxy(0,0); lcd_putsf("* Developed by *");
lcd_gotoxy(0,1); lcd_putsf("Yana Harchyna");
i2c_init();   // I2C Bus initialization
bmp180Calibration();
/* ініціалізація кнопок */
BUTTON_DDR&=~(1<<MENU_ENTER_BTN_PIN)&~(1<<SELECT_PLUS_BTN_PIN)&~(
1<<SELECT_MINUS_BTN_PIN)&~(1<<EXIT_BTN_PIN);
BUTTON_PORT|=(1<<MENU_ENTER_BTN_PIN)|(1<<SELECT_PLUS_BTN_PIN)|
(1<<SELECT_MINUS_BTN_PIN)|(1<<EXIT_BTN_PIN);
  lcd_clear ();
while (1)  {
     // Place your code here
        delay_ms(10);
       if (getBtnStatus(MENU_ENTER_BTN_PIN))
         {
           if (!getPrevBtnStatus(MENU_ENTER_BTN_PIN)) {
```

```
            lcd_clear();   mainMenu(); lcd_clear();       }
          PREV_BUTTON_PIN = BUTTON_PIN;     };
 id= BMP180ReadByte(0xD0);  // має бути 55h
 if (id!=0x55)  { lcd_gotoxy(0,0);        lcd_putsf("BMP180 error");      }
 else  {
  // bmp180Calibration(); // read Calibration coefficients       !!!!!!!!!!!!!
   ut=BMP180ReadUT();   //  UT = 27898-datasheet
   t=BMP180GetTemperature(ut);  // виміряна температура
   upr=bmp180ReadUP();   // UP
   p=bmp180GetPressure(upr); //  атмосферний тиск
  lcd_gotoxy(9,0);   lcd_putsf("       ");  lcd_gotoxy(9,0);   delay_ms(30);
 }   lcd_gotoxy(0,0);
  switch(flagp) {       // тиск {"Pa", "hPa", "mmHg"};
      case 0:
           lcd_gotoxy(0,0);    lcd_putsf("        ");   lcd_gotoxy(0,0);
           sprintf(buffer, "%lu",p);  // long в буфер атмосферний тиск - в Паскалях
           lcd_puts(buffer); lcd_putsf("Pa");  break;
         case 1:
           lcd_gotoxy(0,0);  lcd_putsf("        ");   lcd_gotoxy(0,0);
           ph=(float)(p*0.01);         // long в буфер атмосферний тиск - в hPa
           sprintf(buffer, "%1.1f",ph);  lcd_puts(buffer); lcd_putsf("hP");  break;   // hPa
         case 2:
           lcd_gotoxy(0,0);
           lcd_putsf("        ");
           lcd_gotoxy(0,0);
           pmm=(p*75/10000);    // long в буфер атмосферний тиск -   в mmHg
           sprintf(buffer, "%lu",pmm);
           lcd_puts(buffer);
           lcd_putsf(" mmHg");  break;
         default: break;
       }
  lcd_gotoxy(0,0);
  switch(flagt)        // температура в {'C','F','K'};
       {
   case 0:
        lcd_gotoxy(9,0); lcd_putsf("       ");  lcd_gotoxy(9,0);
        tf=(float) (t*0.1);
        if (tf>=0)
        { sprintf(buffer, "+%1.1f%cC",tf,0xdf);}  // вивід float в буфер температура
         else {sprintf(buffer, "%1.1f%cC",tf,0xdf);};
        lcd_puts(buffer);                          // вивід даних на LCD - температура
       break;
   case 1:
        lcd_gotoxy(9,0);
        tf=(float)(t*0.18)+32;                     //  переведення Цельсій - Фаренгейт
        sprintf(buffer, "%1.1f%cF",tf,0xdf);  // вивід long int в буфер температура
        lcd_puts(buffer); break;
   case 2:
        lcd_gotoxy(9,0);  tk=(float)(t*0.1+273.15);        //  переведення Цельсій - Кельвін
        sprintf(buffer, "%1.1f K",tk);  // вивід long int в буфер температура
        lcd_puts(buffer);  default: break;
       }
```

```
   // запис початкового значення тиску для вимірів погоди або відносної висоти
if (flags==0)
  { lcd_gotoxy(0,1);   lcd_putsf("******");
   if (mode==0)   {
       a0= bmp180Altitude(p);    // висота над рівнем моря
       lcd_gotoxy(7,1); lcd_putsf("       ");   lcd_gotoxy(7,1);
       sprintf(buffer, "%1.1f",a0);  lcd_puts(buffer); lcd_putsf("m");
        lcd_gotoxy(14,1); lcd_putsf(" A");  };
  }       else {
     lcd_gotoxy(0,1);
      if (mode==0) {
        //  запис тиску у Паскалях   p2=p;
       sprintf(buffer, "%lu",p2);               // paw
       lcd_puts(buffer);  a0= bmp180Altitude(p);     //  p=101325,
       a1= bmp180Altitude(p2);    //   p2=101100 ;
       dh= a0-a1;  lcd_gotoxy(7,1);  lcd_putsf("       ");  lcd_gotoxy(7,1);
       sprintf(buffer, "%1.1f",dh);    //
       lcd_puts(buffer); lcd_putsf("m"); lcd_gotoxy(14,1); lcd_putsf(" H");
      };  };
if (mode==1)  {        // режим погода
  deltap= p-p2;       // біжучий тиск - фіксований
  lcd_gotoxy(9,1); lcd_putsf("      "); lcd_gotoxy(9,1);
 // deltap=-260;       // delete !!!     260 -sun   160 - cloudy   140 - опади
  temperatura=t;       // аналіз температури
 switch(flagwx)  {      /* погода ; 0- дощ, 1-хмарно, 2- сонячно*/
     case 0:         // flagwx-дощ
      if (deltap > 250){ lcd_putsf("SUNNY");};        // сонячно
     //  lcd_gotoxy(9,1);
      if ((deltap > 150) && (deltap <250)){lcd_putsf("CLOUDY ");} ;  // хмарно
      if ((deltap <= 150) && (temperatura<0)){lcd_putsf("SNOW");};   // хмарно
      if ((deltap <= 150) && (temperatura>0)){lcd_putsf("RAIN");};   // хмарно
     break;
    case 1:         // flagwx-хмарно
     if (deltap > 150){ lcd_putsf("SUNNY");};
     if ((deltap < (-150)) && (temperatura<0)){lcd_putsf("SNOW");}
     if ((deltap < (-150)) && (temperatura>=0)){lcd_putsf("RAIN");}
     if ((deltap <= 150) && (deltap > (-150))) {lcd_putsf("CLOUD");};
    break;
       case 2: // flagwx-сонячно
     if ((deltap < (-150)) && (deltap > (-200))) { lcd_putsf("CLOUD ");};
     if ((deltap < (-250)) && (temperatura<0)){ lcd_putsf("SNOW");}
     if ((deltap < (-250)) && (temperatura>=0)){ lcd_putsf("RAIN");}
     if (deltap > (-150)) { lcd_putsf("SUNNY ");};
    // else lcd_putsf("SUNNY");
    break;  default: break;
     }     }; };}
```