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 ТОЧНОЇ РЕКОНСТРУКЦІЇ ПОДІЇ ПРИ ДОРОЖНЬО-ТРАНСПОРТНІЙ ПРИГОДІ (ДТП)

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Христина БИК

Науковий керівник:
доцент кафедри ІТ
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

АЗ/ПЗ – відповідно, апаратне забезпечення/програмне забезпечення.

AVR – сімейство восьмибітних мікроконтролерів на RISC-архітектурі.

АЦП (Аналого-цифровий перетворювач) – електронний модуль, який трансформує неперервний аналоговий сигнал у цифрову форму (код).

ЕК – електрична схема (принципова, структурна або функціональна), що відображає взаємозв'язок елементів пристрою.

МК (Мікроконтролер) — мікросхема, що поєднує процесор, пам'ять та периферію, призначена для автоматизації роботи електронних систем.

САПР (Комп'ютерне проєктування) – програмне середовище, призначене для автоматизованої розробки та моделювання технічних об'єктів.

ШІМ (Широтно-імпульсна модуляція) — спосіб регулювання потужності, шляхом варіювання тривалості імпульсів за постійної частоти сигналу.

РКД/LCD — рідкокристалічний дисплей, який використовується для візуального відображення текстової чи графічної інформації.

ПЗП (ROM) – (англ. read-only memory) енергонезалежний тип пам'яті, призначений для зберігання постійної інформації;

EEPROM — тип енергонезалежної пам'яті, що допускає побайтне зчитування, перезапис та очищення за допомогою електричних сигналів.

Flash – різновид напівпровідникової енергонезалежної пам'яті, що забезпечує перезаписування та зберігає дані за умови відсутності електроживлення.

I^2C — послідовна двопровідна шина обміну даними, яка використовує окрему магістраль синхронізації (SCL) та лінію передачі даних (SDA).

SPI – (англ. Serial Peripheral Interface, SPI bus) – швидкісний послідовний синхронний інтерфейс, який функціонує в повнодуплексному режимі та забезпечує обмін даними між мікроконтролером і периферійними пристроями.

ОЗП (RAM) – енергозалежна оперативна пам'ять, яка слугує для тимчасового збереження даних під час виконання програмного коду.

АНОТАЦІЯ

Бик Х., Фармага І. (керівник). Розроблення апаратного і програмного забезпечення цифрового висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП). Бакалаврська кваліфікаційна робота. – Львівський державний університет внутрішніх справ, Львів, 2026.

У дипломній роботі реалізовано програмно-апаратний комплекс для вимірювання абсолютної та відносної висоти 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 (режим висотоміра), контролює зміну тиску та виводить виміряні значення на РК-дисплей. Розроблена система забезпечує конвертацію показників атмосферного тиску з можливістю їх відображення у паскалях (Па), гектопаскалях (гПа) або міліметрах ртутного стовпа (мм рт. ст.), Значення висоти виводиться в метрах. У пристрої передбачено два режими роботи: визначення абсолютної та відносної висоти. На поточний робочий режим вказує відповідна світлодіодна індикація. Крім того, висотомір оснащено інтегрованим меню, яке дозволяє гнучко налаштовувати режими роботи функціонування цифрового альтиметра. Базовим вузлом розробленої системи є мікроконтролер сімейства AVR. До нього під’єднано сенсор тиску BMP180, блок керуючих клавіш та символьний РК-дисплей формату 16x2. У межах дослідження спроєктовано електричну принципову схему приладу та створено її віртуальний аналог у середовищі Proteus VSM. Етап програмно-апаратного налагодження та верифікації функцій альтиметра реалізовано за допомогою Proteus VSM (ISIS). Окрім цього, сформовано логічну схему функціонування цифрового пристрою та написано керуючу програму мовою С в середовищі розробки CodeVisionAVR.

Ключові слова: датчик BMP180, цифровий барометр-альтиметр, мікроконтролер AVR, символьний РК-дисплей, Proteus VSM, вбудовані системи, мова програмування С, CodeVisionAVR.

ABSTRACT

Byk Kh., Farmaha I. (supervisor). Development of hardware and software for a digital altimeter for accurate accident reconstruction in traffic accidents (RTAs).. Bachelor's thesis. – Lviv State University of Internal Affairs, Lviv, 2026.

The thesis implements a hardware and software complex for measuring absolute and relative altitude using 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 and displays the measured values on a liquid crystal display (LCD). The developed system provides conversion of atmospheric pressure readings with the ability to display them in pascals (Pa), hectopascals (hPa), or millimeters of mercury (mmHg). Altitude values are displayed in meters. The device features two operating modes: absolute and relative altitude determination. The current operating mode is indicated by the corresponding LED status. Furthermore, the altimeter is equipped with an integrated menu that allows flexible configuration of the digital altimeter's operating modes. The core node of the developed system is an AVR family microcontroller. A BMP180 digital MEMS pressure sensor, a control keypad block, and a 16x2 character liquid crystal display are connected to it. Within the scope of the study, the schematic electrical diagram of the device was designed, and its virtual counterpart was created in the Proteus VSM computer simulation environment. The stage of hardware-software debugging and verification of the altimeter's functions was implemented using the tools of the Proteus VSM (ISIS) package. In addition, the logical flowchart of the digital device operation was formed, and the control software was written in C language within the CodeVisionAVR integrated development environment (IDE).

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

ЗМІСТ

ВСТУП

Використання висотоміра (або барометричного альтиметра) під час оформлення чи розслідування дорожньо-транспортної пригоди (ДТП) є нестандартним, але в окремих ситуаціях критично важливим елементом для встановлення істини. Найчастіше це стосується аварій у гірській місцевості, на складних розв'зках, естакадах або за складних погодних умов.

При аналізі ДТП експертам-автотехнікам важливо знати точний ухил дороги (підйом чи спуск), оскільки він безпосередньо впливає на: гальмівний шлях: на спуску гальмівний шлях значно подовжується, а на підйомі — скорочується; швидкість транспортного засобу: Дані про зміну висоти допомагають розрахувати, чи могла машина розвинути певну швидкість, або навпаки — чи справно працювали гальма під час тривалого спуску.

У мегаполісах або на сучасних магістралях аварії часто стаються на естакадах, мостах або підземних тунелях, де кілька доріг розташовані одна над одною. Проблема звичайного GPS: Стандартна двовимірна навігація (X, Y) може помилково "прив'язати" координати ДТП до нижнього ярусу дороги, хоча подія відбулася на естакаді над нею. Висотомір додає третю координату (Z), фіксуючи точний ярус (висоту) події. Це критично для екстрених служб (поліції, швидкої), щоб вони розуміли, куди саме під'їжджати, для правильної геолокації місця пригоди. Стрімкий розвиток сучасної мікроелектронної бази, передусім масове виробництво недорогих мікроконтролерів та інтегральних сенсорів, створив передумови для конструювання економічно ефективних програмованих пристроїв. Подібні технічні рішення забезпечують прецизійне вимірювання широкого спектра фізичних величин, зокрема барометричного тиску, поточної температури повітря, а також обчислення абсолютного й відносного рівнів висоти. В основі барометричного методу вимірювання висоти лежить її безпосередня залежність від атмосферного тиску, який, у свою чергу, підвладний впливу поточних метеоумов. Оскільки погодні умови відображають динаміку повітряних мас у конкретній географічній локації та

певному часовому інтервалі, точна оцінка відносної висоти вимагає систематичного врахування метеорологічних даних. Основними параметрами є температурний режим та тиск, оскільки саме барометричні коливання зумовлюють процеси конденсації вологи, утворення хмар та випадіння опадів. Історична потреба людства у передбаченні таких атмосферних явищ стала головним рушієм для проектування та вдосконалення засобів вимірювання висоти, екологічного й метеорологічного моніторингу.

Метою роботи є проектування апаратного і програмного забезпечення цифрового висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП), який вимірює висоту та метеорологічні параметри, а їх значення аналізує і результати аналізу виводить на символьний індикатор та в послідовний порт комп'ютера.

Апаратною платформою для проектованого пристрою було обрано мікроконтролер ATmega128 (архітектура AVR) виробництва Atmel. Використання мікроконтролерів (МК) є цілком виправданим, оскільки вони виконують функцію центрального обчислювального ядра у більшості сучасних вимірювальних систем. Із точки зору розробника, застосування МК дозволяє значно прискорити та економічно оптимізувати процес програмної реалізації складних логічних алгоритмів. Висока щільність інтеграції вбудованої периферії на одному напівпровідниковому кристалі забезпечує ефективне керування зовнішніми пристроями та збирання інформації з датчиків за мінімальної кількості елементів дискретної логіки. Такий підхід скорочує тривалість конструкторського циклу, мінімізує масогабаритні показники та знижує собівартість кінцевого виробу.

Для реалізації поставленої мети треба виконати такі науково-технічні завдання:

- проектування апаратного і програмного забезпечення цифрового висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП) із застосуванням такої елементної бази: МК ATmega128;

- налаштування символьного рідкокристалічного індикатора формату 16×2 (модель WH1602B-YGK-CTK);
- проєктування та відлагодження віртуального прототипу приладу в середовищі САПР Proteus;
- розробка логічної структури та алгоритмічної схеми функціонування мікропроцесорної системи;
- створення ПЗ для отримання й обробки інформації з сенсора тиску BMP180;
- створення програмних функцій для виводу даних на символьний дисплей.
- перевірка та аналіз роботи спроєктованого висотоміра шляхом моделювання в програмному комплексі Proteus ISIS.

Об’єкт дослідження – проектуваннння апаратного і програмного забезпечення цифрового висотоміра для точної реконструкції події при дорожньо-транспортній пригоді.

Предмет дослідження – моделі та засоби проєктування висотоміра.

Методи досліджень. Методологічну основу досягнення поставленої мети становить системний підхід та сукупність загальнонаукових методів. На етапі аналізу науково-технічних джерел та профільної літератури було визначено провідні напрямки у сфері проектування комплексів моніторингу. Безпосереднє обґрунтування та формування технічних рішень здійснювалося із залученням методів комп’ютерного моделювання, системного аналізу і синтезу. Окрім цього, теоретичне підґрунтя роботи базується на використанні логічних прийомів індукції та дедукції, а також засад структурного аналізу.

Структура роботи. Структурно робота містить вступну частину, чотири основні розділи, загальні висновки, перелік використаної літератури та додатки. Пояснювальна записка має 75 сторінок основного тексту, які проілюстровані 46 рисунками та 15 таблицями. Довідкову базу дослідження складають 12 бібліографічних джерел. Загальний об’єм роботи, з урахуванням 2 додатків, становить 88 сторінок.

РОЗДІЛ 1
СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Метеорологічні величини та їх вимірювання

Під час застосування барометричних сенсорів для дистанційного обчислення висоти головним чинником, що зумовлює похибку вимірювального процесу, виступає нестабільність повітряного середовища. Метеорологічні параметри відображають комплекс фізичних характеристик атмосфери, серед яких визначальними є барометричний тиск, температура, відносна вологість, а також кінематичні властивості повітряних мас (напрямок і швидкість вітрових потоків) та щільність хмарного покриву. Своєю чергою, атмосферні явища являють собою інтенсивні фізико-динамічні процеси (включаючи випадіння опадів у вигляді снігу чи дощу, грозову активність, а також оптичні ефекти, як-от полярне сяйво чи веселка), що спричиняють суттєві та раптові збурення в стані газової оболонки Землі.

До переліку ключових метеорологічних параметрів належать: Температура повітря, яка слугує мірою кінетичної (теплової) енергії молекул певного середовища (водних мас, повітряного простору або ґрунтового покриву) і визначає його тепловий рівень. Термічний стан земної атмосфери відзначається високою нестабільністю.

У приземному шарі температурні показники коливаються у вкрай великих межах: екстремальний максимум, зареєстрований на планеті, сягає майже +60 °C (у тропічних широтах), тоді як абсолютний мінімум становить близько -90 °C (в Антарктиді). Градієнт зміни температури залежно від віддалення від поверхні Землі є нелінійним. У нижній товщі повітря (до межі 10–15 км) спостерігається охолодження, після чого на ділянці між 50 та 60 км фіксується термічне зростання, яке згодом знову змінюється спадним трендом.

Під час виконання прикладних фізичних обчислень та інженерно-технічних вимірювань базовими є дві шкали:

- Шкала Цельсія (t °C) — найбільш поширена у повсякденному житті та інженерній практиці система вимірювання. Як базові точки в ній використовуються термодинамічні стани води за умов стандартного атмосферного тиску (1013 гПа): значення 0 °C фіксує момент кристалізації (замерзання), а 100 °C — процес переходу в пароподібний стан (кипіння).
- Шкала Кельвіна (T K) — термодинамічна шкала для відліку абсолютних значень температури. Рівень 0 K (абсолютний нуль) відображає теоретичний стан речовини, за якого тепловий рух елементарних частинок (молекул та атомів) повністю припиняється. У системі Цельсія ця межа еквівалентна значенню -273,15 °C.

Зважаючи на те, що інтервал ціни поділки (одного градуса) в обох зазначених системах є однаковим (1 K=1 °C), температурні показники за Кельвіном ніколи не набувають від'ємних значень. Математичний перерахунок величин із системи Цельсія у шкалу Кельвіна здійснюється за такою формулою:

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

Шкала Фаренгейта (°F) – альтернативна система вимірювання температури, запропонована Даніелем Габріелем Фаренгейтом у 1724 році, яка до сьогодні активно застосовується у низці країн (зокрема, в США) паралельно з метричними стандартами. Основними константами цієї шкали є: за умов нормального тиску кристалізація (замерзання) чистої води відбувається за температури +32 °F, тоді як процес її кипіння фіксується на позначці +212 °F. Величина одного градуса Фаренгейта еквівалентна 1/180 частині температурного діапазону, що лежить між точками танення льоду та пароутворення води. Для математичного перерахунку виміряних значень із градусів Цельсія (t °C) у шкалу Фаренгейта використовується така формула:

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

Нуль шкали Фаренгейта рівний температурі -17,8°C за стоградусною шкалою Цельсія. Нулі цих шкал не збігаються. Градус Фаренгейта менший за градус шкали Цельсія.

Атмосферний тиск визначається як механічна сила, з якою вертикальний стовп повітря тисне на одиничну площу земної кори. Сукупна маса газової мантії Землі, що генерує це навантаження, є колосальною та оцінюється приблизно у приблизно $5{,}15\times10^{15}$ тонн. Починаючи з XVII сторіччя, реєстрацію барометричних показників виконували за допомогою рідинних приладів, фіксуючи висоту підняття ртуті у лінійних вимірах (дюймах або міліметрах). Проте з автоматизацією синоптичних досліджень та переходом до комп'ютерного моделювання використання міліметрів ртутного стовпа (мм рт. ст.) визнали нераціональним. Ця одиниця є суто інструментальною та не демонструє природу тиску як сили, що ускладнює її інтеграцію у сучасні диференціальні рівняння та інженерні розрахунки. Для подолання цього недоліку за рекомендацією норвезького геофізика В. Б’єркнеса в метеорологію було впроваджено мілібар (мбар): мілібар (мбар): еквівалентний тиску, при якому сила у 1000 дин діє на площу в 1 см². (де дина визначається як сила, що надає тілу масою 1 г прискорення 1 см/с²).

Згідно з міжнародними стандартами SI (CI), загальноприйнятою одиницею тиску є Паскаль (Па): Паскаль (Па): фундаментальна одиниця вимірювання, що характеризує зусилля в 1 Ньютон, яке перпендикулярно та рівномірно тисне на площину розміром 1 м². У прикладній аеродинаміці та метеорологічних розрахунках базовими є два еталонні показники:

1. Нормальний атмосферний тиск: константа, визначена на нульовій висоті над рівнем моря (на 45-й географічній паралелі) за термічних умов 0 °C. Ця величина фіксується на позначці 1013,25 мбар (або 760 мм рт. ст.).
2. Стандартний атмосферний тиск: уніфіковане значення для інженерно-технічного нормування, яке усереднено приймають за 1000 гПа (або 750 мм рт. ст.).

Одиниці вимірювання тиску: мм рт.ст, гПа, мбар: 1 гПа=100 Па=1 мбар; 1 гПа=3/4 =0,75 мм рт.ст; [P]=[H/m]=[Па]; 1 мм. рт.ст.=1,333 гПа.

Величини атмосферного тиску є динамічними; їхні варіації безпосередньо визначають поточну синоптичну ситуацію, зокрема процеси хмароутворення, рівень вологості та термічний режим повітряних мас. З погляду фізіології людини, оптимальним для життєдіяльності є діапазон від 750 до 760 мм рт. ст.

Упродовж доби барометричні показники демонструють двофазний добовий хід: пікові значення фіксуються вранці та ввечері, тоді як спади (мінімуми) припадають на пополудневий час і період після опівночі. Крім того, спостерігається чітко виражена сезонна залежність: взимку, внаслідок підвищеної щільності та більшої ваги охолодженого повітря, тиск є вищим порівняно з літнім сезоном, коли нагріті повітряні маси розширюються і стають легшими.

Головним чинником, що впливає на абсолютне значення тиску, виступає географічна висота точки над рівнем Світового океану. У міру підняття вгору щільність газової оболонки та вертикальний стовп повітря над об'єктом зменшуються, що зумовлює експоненціальне падіння барометричного тиску. Саме цей стійкий фізичний взаємозв'язок покладено в основу функціонування авіаційних та топографічних висотомірів (альтиметрів).

1.2. Огляд існуючих підходів для розробки барометрів та висотомірів

Електронні барометричні пристрої є базовими елементами автоматизованих метеорологічних станцій, бортових авіаційних альтиметрів та комплексів екологічного контролю. Поряд із професійним сегментом, ці первинні перетворювачі масово інтегруються у бюджетну споживчу електроніку. Огляд науково-технічних першоджерел та профільних інтернет-публікацій свідчить про стійку актуальність до проектування засобів вимірювання атмосферного тиску та обчислення висотних координат.

Систематизація наявних на ринку та в академічному середовищі рішень дозволяє окреслити такі тенденції:

- Автоматизація на базі мікроконтролерів: переважна більшість сучасних розробок проектується з використанням мікроконтролерних ядер, які беруть на себе функції збирання, фільтрації та математичної обробки інформації.
- Телеметричний обмін даними: значний сегмент ринку займають пристрої з інтегрованими GSM-модемами. Вони реалізують дистанційну передачу виміряних значень у режимі реального часу через стільникові канали, що є критично важливим для автономних постів спостереження.
- Еволюція первинних сенсорів: для реєстрації тиску використовуються як традиційні механічні барометри-анероїди, так і передові твердотільні (аналогові та цифрові) напівпровідникові мікросхеми.
- Застосування новітньої напівпровідникової техніки відкриває можливість створювати вимірювальні прилади, які вдало поєднують прецизійну точність із мінімальними габаритами та високою енергоефективністю.

Класифікація барометрів. Рідинний (ртутний) барометр – історичний еталонний прилад для визначення барометричного тиску, де в ролі робочого середовища застосовується ртуть. Його конструкція та фізичний принцип дії мають такі особливості: конструктивно апарат містить запаяну з одного боку скляну капілярну трубку (колбу). Ця трубка заповнюється ртуттю і занурюється відкритим кінцем у чашу, також наповнену ртуттю. Відлік результатів здійснюється за допомогою градуйованої шкали, розташованої паралельно або нанесеної безпосередньо на скло. Газова оболонка планети тисне на відкриту дзеркальну поверхню ртуті в резервуарі, що змушує рідкий метал переміщатися всередині капіляра. За умови підвищення атмосферного тиску ртутний стовпчик рухається вгору, а при барометричних спадах – опускається, зупиняючись навпроти відповідного числової позначки шкали. На Рис.1.1. зображено ртутний барометр.

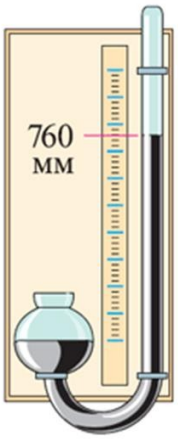


Рис. 1.1. Ртутний барометр

Барометр-анероїд представляє собою безрідинний вимірювальний прилад, призначений для реєстрації показників атмосферного тиску. Фізичний принцип його роботи базується на механічній деформації внутрішнього чутливого елемента, що виникає внаслідок силового впливу повітряного стовпа. Конструктивні елементи барометра-анероїда:

- Герметична мембранна капсула (анероїдна коробка): порожниста металева ємність із гофрованими поверхнями, виконана зі спеціальних нікель-срібних сплавів або термічно обробленої сталі. Всередині капсули забезпечено високий рівень вакууму (глибоке розрідження повітря).
- Протидійна пружина (пружний елемент): компенсує зусилля від зовнішнього барометричного тиску, не дозволяючи атмосфері повністю сплющити коробку.
- Передавально-множинний важільний привід: кінематична система, яка перетворює лінійні мікропереміщення гофрованих стінок у круговий рух індикаторного елемента.
- Циферблат та індикаційна стрілка: вузол візуалізації, зазвичай відкалібрований у значеннях, еквівалентних міліметрам ртутного стовпа або гектопаскалям.

Робота пристрою базується на деформації металевої камери під впливом мінливого навантаження з боку газового середовища. За умов підвищення тиску відбувається стиснення гофрованих стінок коробки, які долають

механічний опір внутрішньої пружини. У разі барометричного спаду пружний елемент розтискає стінки мембрани. Ці мікроскопічні коливання геометричних розмірів через систему тяг і важелів транслюються на вказівну стрілку, що переміщується над циферблатом. Через екологічну безпеку (відсутність ртутної пари), ергономічність та мобільність, безрідинні прилади отримали найбільше розповсюдження у побутовому секторі та експедиційній практиці. Залежно від призначення, корпус може бути пластиковий, металевий чи дерев’яний (для інтер’єрних рішень).

З огляду на те, що механічні властивості конструкційних матеріалів зазнають змін внаслідок тривалої експлуатації, для стабілізації метрологічних характеристик у конструкцію інтегрують вузли налаштування. Калібрування та звірення барометрів-анероїдів реалізують шляхом порівняння з первинним ртутним еталоном за такими параметрами:

- Поправка шкали: нівелює інструментальну похибку, викликану нелінійністю пружної деформації металу на різних відрізках робочого діапазону.
- Температурне коригування: усуває температурний дрейф показників, зумовлений лінійним розширенням деталей механізму при зміні температури довкілля.
- Залишкова (додаткова) поправка: враховує “старіння” металу та деградацію еластичних властивостей металевих компонентів .

На Рис.1.2. зображено безрідинні, механічні барометри (анероїди).

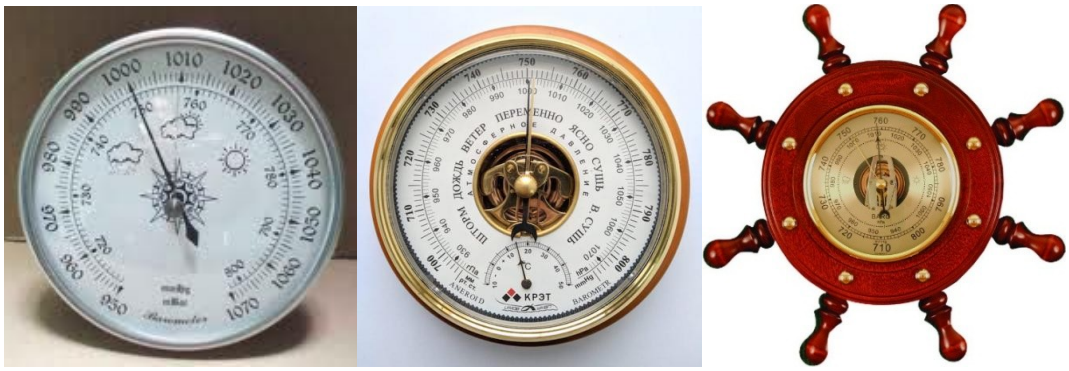


Рис.1.2. Барометри-Анероїди

Цифрові барометричні є результатом еволюційного розвитку традиційних механічних анероїдів. Фізичний принцип їхньої дії базується на перетворенні механічного напруження чутливої мембрани в еквівалентний електричний сигнал. Надалі цей первинний сигнал оцифровується за допомогою АЦП, аналізується мікроконтролером і транслюється на пристрій відображення у числовому чи графічному вигляді.

Стрімкий прогрес у галузі мікроелектромеханічних структур (MEMS) зумовив появу малогабаритних і прецизійних первинних перетворювачів тиску. Завдяки мікроскопічним розмірам, інтеграція функціональних блоків барометра та альтиметра стала можливою у різноманітних мобільних гаджетах, зокрема:

- мобільних телефонах і планшетних комп’ютерах;
- розумних годинниках, а також трекерах активності;
- автономних спеціалізованих альтиметрах.

У сучасних електронних розробках та промисловому приладобудуванні найбільшого поширення набули такі компоненти:

- MPX4115: тензорезистивний (п’єзорезистивний) компонент, який генерує на вихідному виводі аналогову напругу, пропорційну поточному барометричному тиску.
- BMP085: напівпровідниковий цифровий модуль, що підтримує обмін інформацією за допомогою послідовного двопровідного інтерфейсу I2C.
- BMP180: модернізоване покоління сенсорів, що прийшло на зміну BMP085. Цей чіп так само використовує шину I2C, проте характеризується підвищеною точністю вимірювань, оптимізованим струмом споживання та покращеною термостабільністю, внаслідок чого він став індустріальним стандартом для вбудованих систем (embedded systems).

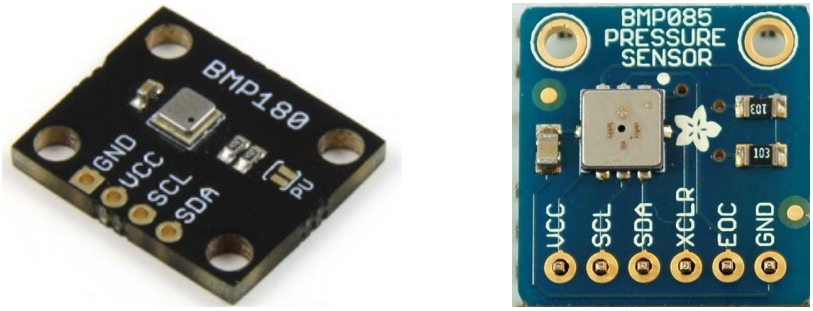


Рис.v1.3. МЕМС модуль з сенсорами тиску BMP180, BMP085

Рис.v1.4. П’єзорезистивний сенсор тиску MPX4115

Процеси мініатюризації електронних компонентів відкрили можливість для інтеграції первинних перетворювачів у малогабаритні споживчі пристрої. Це забезпечує кінцевим користувачам доступ до актуальної метеорологічної інформації. Нижче проїлюстровано вироби для вимірювання тиску і альтитуди:

Рис. 1.5. Метеостанція Explore Scientific Color

Рис. 1.6. Метеостанція TFA “SEASON”

Рис. 1.7. Барометр-висотомір

Рис. 1.8. Альтиметр УВ-57

Пристрої механічного типу, конструкція яких передбачає наявність спеціалізованої відлікової шкали для розрахунку висоти, класифікують як альтиметри (висотоміри). Зазначена категорія вимірювальних приладів є невіддільною частиною бортового обладнання в авіації, а також використовується під час проведення альпіністських експедицій та гірських сходжень.

Рис.1.9. Барометр з годинником

На Рис.1.5 – 1.9 зображено вироби цифрових висотомірів, метеостанцій, годинників з датчиками тиску.

1.2. Вимірювання метеопараметрів сенсорами та обробка їх значень

Традиційні ртутні барометри та класичні анероїди вимагають від оператора суб’єктивного візуального зчитування. Значення фіксуються безпосередньо за положенням меніска рідини або стрілки на круговій шкалі приладу. Натомість твердотільні датчики, такі як напівпровідниковий давач MPX4115, функціонують на інших принципах і потребують обов’язкової

цифрової обробки даних. Робота сенсора базується на зміні його електричних характеристик, де вихідний аналоговий сигнал (напруга) є пропорційний рівню зовнішнього тиску.

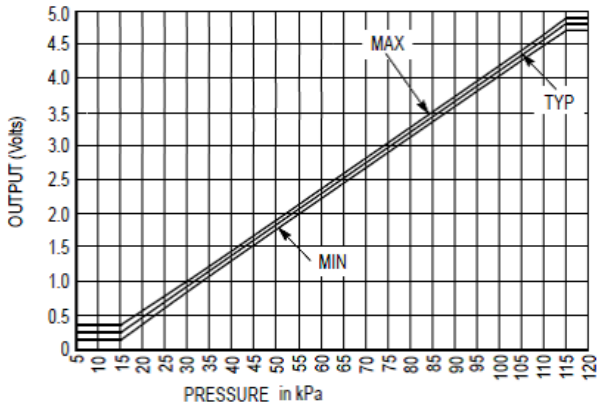


Рис. 1.10. Залежність напруги на виході MPX4115 від тиску

Електронні модулі BMP085 та BMP180 представляють групу цифрових сенсорів, які передають дані за допомогою послідовного інтерфейсу I^2C. Ці мікросхеми призначені для моніторингу параметрів довкілля – температури та атмосферного тиску. Модель BMP180 є наступним поколінням датчиків цієї лінійки, що забезпечує кращу стабільність характеристик та мінімізує похибку вимірювань.

Архітектура чипа BMP180 містить п’єзорезистивний чутливий елемент, аналого-цифровий перетворювач (АЦП), логічний контролер керування, інтерфейс I^2C та енергонезалежну пам’ять E2PROM. В останній зберігається 176 біт заводських коефіцієнтів для калібрування.

Процес обчислення достовірних фізичних величин базується на математичній корекції вихідних сигналів. Мікропроцесор зчитує з внутрішніх регістрів датчика невідкореговані (“сирі”) параметри:

- необроблене значення тиску з регістру UP (розрядністю від 16 до 19 біт);
- необроблене значення температури з регістру UT (16 біт).

Послідовність дій під час циклу вимірювання зображена на схемі Рис.1.11.

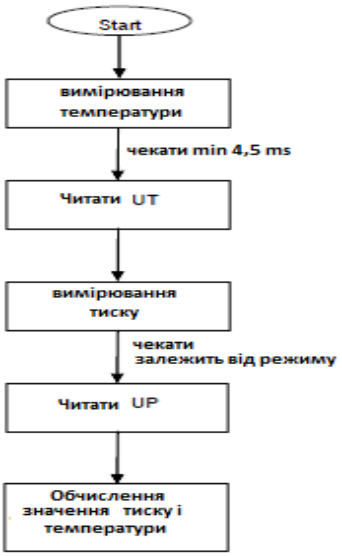


Рис. 1.11. Алгоритм вимірювання тиску сенсором BMP180

Для коректного обчислення тиску і температури необхідно враховувати специфічні калібрувальні параметри, унікальні для кожного екземпляра датчика BMP180. Ці дані зберігаються у внутрішній E^2PROM обсягом 176 біт, яка розділена на 11 регістрів по 16 біт. До початку вимірювання мікроконтролер здійснює одноразове зчитування цих масивів. Позначення коефіцієнтів, а також їхні адреси для доступу через шину обміну даними представлені у Табл. 1.1.

Таблиця 1.1. Адреси калібрувальні коефіцієнти для 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 гнучкого налаштування пристрою під специфіку конкретної інженерної задачі передбачено вибір роздільної здатності сенсора. Конфігурування режимів функціонування датчика реалізується програмно шляхом запису чисел від 0 до 3 у відповідний керуючий регістр. Кожне з цих чисел визначає параметри між точністю вимірювань та рівнем енерговитрат: 0

(Ultra Low Power) – режим із найменшим споживанням струму; 1 (Standard) – стандартний (номінальний) режим роботи; 2 (High) – конфігурація з підвищеною точністю; 3 (Ultra High Resolution) – режим гранично високої роздільної здатності.

Послідовність математичних обчислень, необхідних для отриманих із сенсора реальних фізичних показників температури та атмосферного тиску дано в алгоритмі, який зображено на Рис. 1.12.

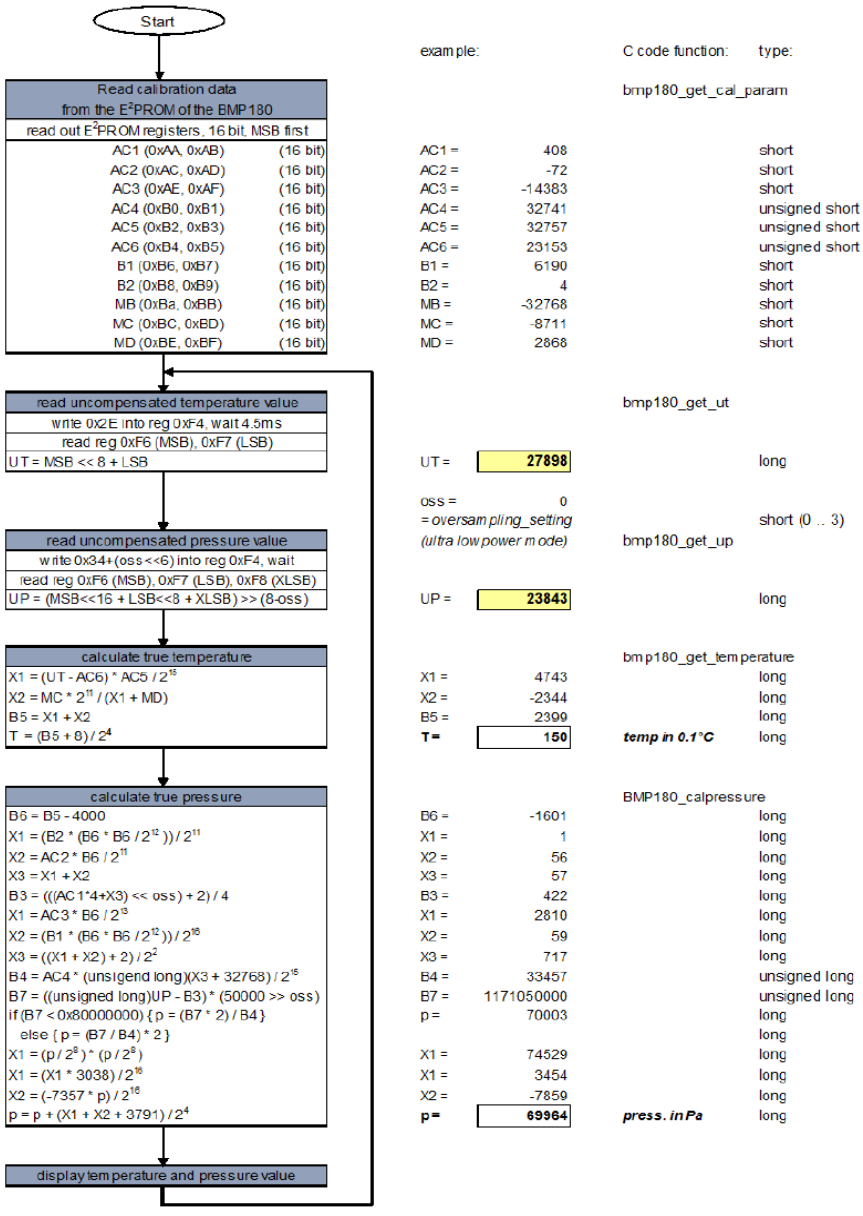


Рис. 1.12. Алгоритм отримання реальних значень температури та тиску

Для обчислення висоти над рівнем океану на основі даних, одержаних від сенсора BMP180, використовується наступна барометрична формула. Розрахунок базується на аналізі поточного показника тиску P та еталонного

значення P0, що відповідає нормальному атмосферному тиску на нульовій відмітці (рівень океану) та становить 1013,25 гПа.

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

Досліджено, що зміна тиску на 1 hPa відповідає зміні висоти на 8,43м.

На Рис. 1.13 бачимо графік залежності тиску від висоти (над рівнем океану).

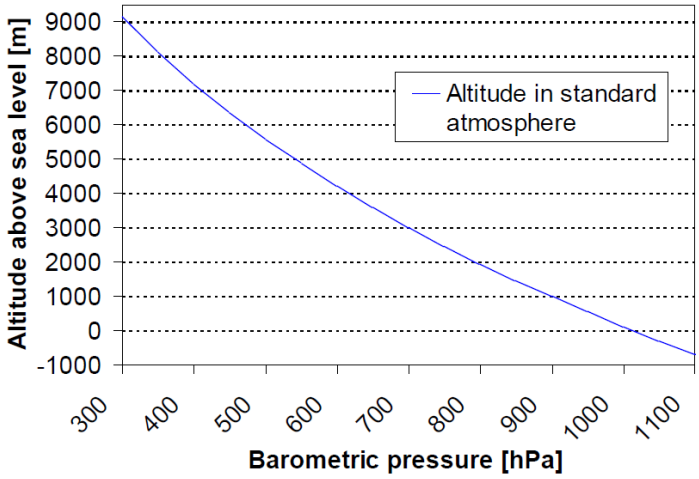


Рис. 1.13. Графік залежності тиску від висоти (над рівнем океану)

Для сенсора BMP180 точність у всьому діапазоні вимірювання становить для тиску ±0,12hPa, це для висоти ±1м та температури ±0,5 ºC.

РОЗДІЛ 2
ЗАСОБИ РОЗРОБКИ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВИСОТОМІРА

2.1. Система автоматизованого проектування та симулювання програмованих пристроїв Proteus VSM

Середовище комп’ютерного моделювання Proteus VSM, розроблене компанією Labcenter Electronics, є потужним інструментарієм для проєктування та комп'ютерного аналізу цифрових і аналогових пристроїв. Головна перевага цього програмного забезпечення полягає у здатності інтегрувати симуляцію мікроконтролерів (включаючи платформи Arduino, архітектури AVR, ARM тощо) із віртуальними аналогами вимірювальних приладів. Це дозволяє здійснювати наскрізне налагодження прошивки та верифікацію схемотехнічних рішень в єдиному середовищі, мінімізуючи потребу в макетуванні.

Пакет Proteus об’єднує засоби для розробки схем і модуль трасування друкованих плат (PCB Layout/ARES). Велика бібліотека інтегральних схем і дискретних елементів дозволяє створювати складні мікропроцесорні системи. У випадку відсутності компонента, користувач може інтегрувати сторонні SPICE-моделі від виробників напівпровідників або спроєктувати власний макромодельний блок. Функціональна архітектура комплексу базується на інтеграції двох основних підсистем:

1. Schematic Capture (раніше ISIS) — графічний інтерфейс для формування принципових схем та запуску інтерактивної симуляції. Створена тут електрична мережа є вихідними даними для побудови топології плати.
2. PCB Layout (раніше ARES) — спеціалізований редактор друкованих плат, оснащений алгоритмами автоматичного розміщення компонентів та інтегрованим автотрасувальником ELECTRA. Інструмент автоматизує рутинні операції та дозволяє експортувати технологічні

файли (Gerber, Excellon) для виготовлення плат на виробництві або верстатах із ЧПУ.

Окрім стандартних інструментів, середовище моделювання Proteus містить додаткові модулі для інтеграції віртуальних пристроїв із фізичним апаратним забезпеченням ПК. Зокрема, для цього використовуються такі компоненти: COMPIM – спеціалізований програмний вузол, який пов'язує модельовану схему із реальним послідовним інтерфейсом (COM-портом / UART) комп’ютера. USBCONN – модуль, призначений для організації прямого обміну даними між симуляцією та фізичним USB-портом хост-машини.

Цей програмний комплекс є високоефективним рішенням на етапі попереднього тестування та верифікації схемотехніки. При цьому мікроконтролерне програмне забезпечення, що імпортується у Proteus у вигляді бінарних файлів, може бути скомпільоване у сторонніх інтегрованих середовищах розробки (IDE). Серед них найчастіше застосовуються: CodeVisionAVR та WinAVR (орієнтовані на оптимізацію коду під 8-бітні архітектури AVR); ICC (універсальне середовище, що підтримує роботу з архітектурами AVR, MSP430 та ARM7); HiTECH (спеціалізоване ПЗ для прошивки мікроконтролерів сімейства PIC та архітектури 8051).

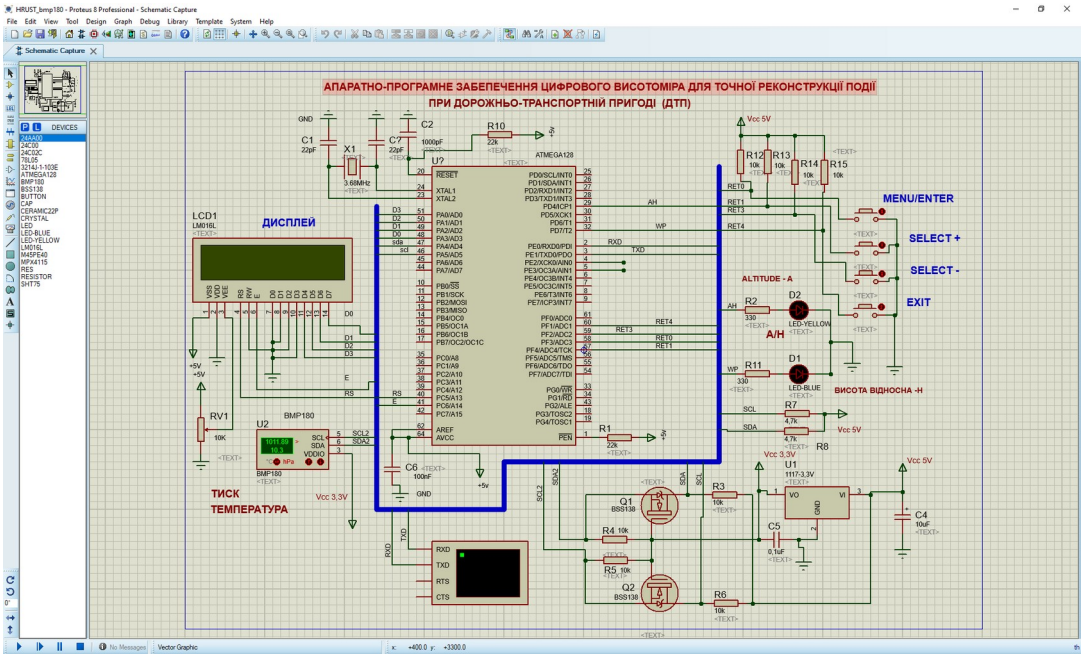


Рис. 2.1. Середовище Schematic Capture

2.2. Мова С та ПЗ CodeVision AVR для систем на МК AVR

Інтегроване середовище розробки CodeVisionAVR є високоефективним програмним забезпеченням, призначеним для написання та компіляції прикладного коду. Цей інструментарій оптимізовано спеціально для роботи з 8-бітними мікроконтролерами сімейства AVR виробництва компанії Atmel (наразі Microchip). Під час розробки програмного забезпечення у середовищі CodeVisionAVR підтримується стандартний набір типів даних мови С, а також їхні специфічні апаратні модифікації, базові характеристики яких систематизовано в Табл. 2.1.

Таблиця 2.1. Типи даних в мові С для МК AVR

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 інтегрованому середовищі CodeVisionAVR не можна вживати як ідентифікатори такі 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	

Для забезпечення наскрізного циклу проєктування пристроїв на базі архітектур AVR та AVR32 використовується комплекс програмних засобів,

що включає інструменти з відкритим вихідним кодом. Зокрема, популярний набір утиліт WinAVR базується на крос-компіляторі GNU GCC, який підтримує розробку програмного забезпечення мовами C та C++.

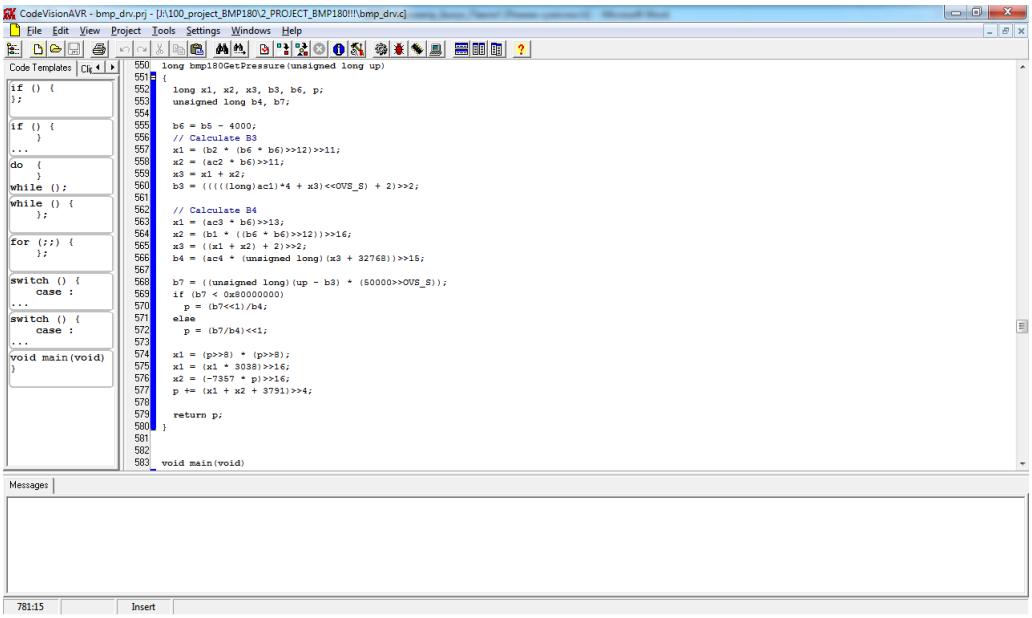


Рис. 2.2. CodeVisionAVR. Середовище програмування

CodeVisionAVR відоме високою якістю генерації оптимізованого й компактного машинного коду, а також докладним документаційним супроводом. Середовище функціонує на базі стандартного синтаксису мови C, що істотно спрощує процес проектування прошивок. Важливою архітектурною перевагою є можливість інтеграції з Atmel Studio (Microchip Studio), що дозволяє створити наскрізний цикл написання, компіляції та відлагодження програмного забезпечення для мікроконтролерів AVR. Робочий простір має англомовний інтерфейс, а в каталозі інсталяції \cvavr\bin розробнику доступна велика база практичних прикладів та інструкцій. Сучасні версії пакету адаптовані для роботи з операційними системами сімейства MS Windows. Ключові характеристики середовища CodeVisionAVR: сумісність з актуальними версіями MS Windows; мови програмування: високорівнева мова C (власний комерційний компілятор) та низькорівневий Асемблер; вбудований текстовий редактор, інтегрований менеджер проєктів та автоматизований генератор початкового коду; пряма інтеграція із середовищем проектування Atmel Studio; програма підтримує розширений

перелік апаратних засобів програмування та налагодження – як оригінальних від Atmel (STK500, AVRISP, AVR Dragon, JTAGICE), так і сторонніх (STK200, та інші ISP-програматори).

Важливою перевагою CodeVisionAVR є наявність розвиненої системи вбудованих низькорівневих бібліотек, які спрощують взаємодію з поширеними периферійними компонентами та інтерфейсами: символьні рідкокристалічні дисплеї (LCD) на базі контролера HD44780; послідовні шини обміну даними SPI та I^2C; напівпровідникові термодатчики (зокрема компаній Maxim/Dallas та серії LM75); електронні годинники реального часу (RTC типу DS1307, PCF8583); однопровідний інтерфейс 1-Wire.

Унікальним елементом екосистеми є інтегрований майстер налаштувань CodeWizardAVR. Цей інструмент автоматизує рутинні операції, дозволяючи за кілька кліків згенерувати готовий шаблон ініціалізації внутрішніх ресурсів кристала: конфігурації портів введення-виведення, таймерів-лічильників, векторів переривань, аналого-цифрового перетворювача (АЦП), вбудованого компаратора та інтерфейсів зовнішньої пам’яті.

РОЗДІЛ 3
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ВИСОТОМІРА

3.1. Вибір елементної бази для проектування висотоміра

Елементною базою для проектування висотоміра вибрано: МК AVR (ATmega128), датчик тиску BMP180, РКД 16x2 (WH1602B-YGK-CTK).

МК ATmega128. Восьмибітні мікроконтролери архітектури AVR, розроблені компанією Atmel, займають провідні позиції в індустрії вбудованих систем. Завдяки збалансованому поєднанню високої продуктивності, енергоефективності та оптимізованого набору команд, ці чипи залишаються надійним фундаментом для сучасних інженерних рішень. Аналіз критеріїв “вартість-функціональні можливості-обчислювальний ресурс” зумовив вибір мікросхеми ATmega128 як головного обчислювального вузла у цьому проєкті. В основі сімейства AVR лежить концепція RISC-архітектури, що гарантує швидке оброблення даних та гнучкість керування периферією. Практичність відлагодження прототипів забезпечується можливістю внутрішньосхемного перепрограмування пам'яті (ресурс становить до 10 000 циклів). Серія Mega AVR розроблена для задач із підвищеними вимогами до обчислень; вона пропонує швидкодію до 16 MIPS, інтегровану пам'ять програм місткістю до 128 Кбайт, а також розвинену периферію: статичну RAM, енергонезалежну EEPROM та 10-бітний аналого-цифровий перетворювач. Структурно лінійка AVR поділяється на кілька гілок, серед яких помітне місце посідають базові Classic AVR та малогабаритні Tiny AVR. Моделі Classic є стандартним індустріальним рішенням (продуктивність до 16 MIPS, Flash – до 8 Кбайт, SRAM — до 512 байт), тоді як чипи Tiny являють собою ультракомпактні бюджетні компоненти у маловивідних (часто 8-пінових) корпусах. Важливою перевагою МК AVR є низхідна програмна сумісність: прикладне ПЗ, створене для молодших моделей, з мінімальними змінами переноситься на потужніші кристали.

Модель ATmega128 є одним із найбільш функціональних представників 8-бітного сімейства, володіючи вбудованою Flash-пам'яттю об'ємом 128 Кбайт. RISC-ядро оперує 133 інструкціями, більшість із яких виконуються за один такт генератора, що еквівалентно швидкодії 16 MIPS на тактовій частоті 16 МГц. Архітектура має роздільні простори пам’яті даних та програм (Гарвардська структура), а також містить 32 регістри загального призначення. Енергонезалежні блоки пам’яті мають високу надійність: Flash гарантує 10 тисяч циклів стирання/запису, а масив EEPROM (4 Кбайт) – до 100 тисяч циклів. Інтегрована периферія ATmega128 охоплює два 8-розрядні та два 16-розрядні таймери-лічильники, а також 8-канальний 10-бітний АЦП. Для міжмодульної комунікації передбачено апаратні інтерфейси SPI, USART та порт JTAG. Захист від критичних збоїв реалізовано за допомогою сторожового таймера (Watchdog) та схеми скидання при просіданні живлення (BOD). Додатково мікроконтролер підтримує генерацію 6 каналів ШІМ (із розрядністю 2...16 біт) та обробку системи зовнішніх і внутрішніх переривань.

Для оптимізації споживання струму передбачено 6 режимів сну (енергозбереження), а для комутації із зовнішніми вузлами виведено 53 універсальні лінії вводу-виводу. Чіп постачається у 64-вивідних корпусах типу TQFP або MLF. Залежно від умов експлуатації доступні дві модифікації: низьковольтна версія “L” (діапазон живлення 2,7–3,3 В, тактова частота до 8 МГц) та базова версія (живлення 4,5–5,5 В, частота до 16 МГц).

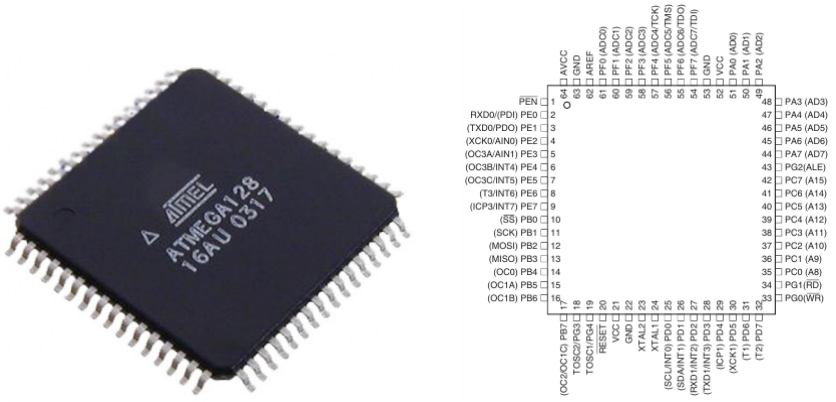


Рис. 3.1. ATmega128 в корпусах TQFP і MLF

Архітектурну основу чіпів AVR становить модернізована система інструкцій, адаптована для взаємодії з 32 регістрами загального призначення (РЗП). Оскільки всі РЗП мають пряме апаратне підключення до арифметично-логічного пристрою (АЛП), мікроконтролер здатний виконувати операції над двома незалежними регістрами всього за один такт синхронізації. Таке схемотехнічне рішення гарантує високу швидкість обчислень, завдяки чому пристрої AVR за продуктивністю майже вдесятеро випереджають класичні МК на базі CISC-архітектури.

Інтегрована пам’ять мікроконтролера ATmega128/L містить 128 Кбайт Flash-пам’яті програм із підтримкою апаратної функції паралельного доступу Read-While-Write. Для збереження даних користувача та змінних передбачено по 4 Кбайт енергонезалежної пам’яті EEPROM та статичної оперативної пам'яті (SRAM). Доступні функції розширено завдяки інтеграції чотирьох незалежних таймерів-лічильників, комунікаційних вузлів USART, SPI, I^2C, а також 10-розрядного аналого-цифрового перетворювача на 8 вхідних каналів. Запобігання критичним зависанням програмного коду забезпечується незалежним сторожовим таймером (Watchdog), який тактується від власного RC-осцилятора. З метою керування енергоспоживанням у чіпі реалізовано шість енергоощадних конфігурацій:

- режим Idle: зупиняє роботу обчислювального ядра, залишаючи активною периферію;
- режим зменшення шумів АЦП (ADC Noise Reduction): мінімізує цифрові завади під час перетворення завдяки вимкненню більшості модулів, крім самого АЦП та асинхронного таймера;
- режим глибокого сну (Power-down): повністю зупиняє роботу головного осцилятора та внутрішніх блоків із збереженням поточного стану регістрів. Пробудження мікросхеми здійснюється виключно за допомогою апаратного скидання (Reset) чи через зовнішні переривання;

- режим очікування (Standby): відрізняється від попереднього збереженням генерації основного тактового сигналу, що забезпечує надшвидкий перехід системи у робочий стан;
- режим розширеного очікування (Extended Standby): підтримує одночасне функціонування як основного, так і допоміжного асинхронного генераторів.

Запис керуючої прошивки у Flash-пам'ять пристрою може виконуватися за допомогою послідовного інтерфейсу SPI або через виділену область завантажувача (Boot Loader). Висока гнучкість конфігурування та обчислювальний потенціал RISC-ядра роблять цей пристрій ефективним базовим елементом для вбудованих систем керування. Схема внутрішньої організації мікроконтролера представлена нижче на Рис. 3.2.

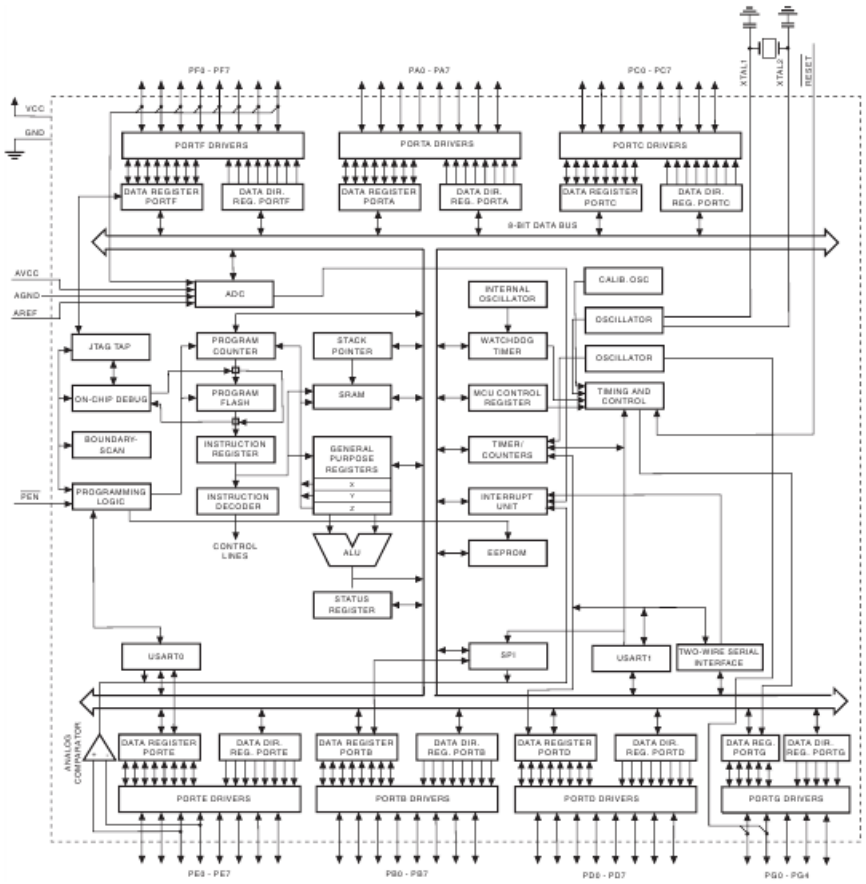


Рис. 3.2. Структура МК AVR ATmega128

Аналого-цифровий перетворювач (АЦП) ATmega128. Взаємодія мікроконтролера із зовнішніми пристроями здійснюється через універсальні порти вводу-виводу, які здебільшого функціонують у дискретному

(двійковому) форматі. За умов живлення схеми напругою 5 В, стан логічної “1” еквівалентний високому рівню напруги на відповідній лінії, тоді як логічний “0” відображає низький потенціал. Проте для моніторингу реальних фізичних процесів часто виникає потреба в реєстрації сигналів, що змінюються безперервно у часі, де стандартні цифрові інтерфейси є неефективними. Для детектування та квантування таких аналогових величин, чий рівень коливається в межах від нуля до опорної напруги, в архітектуру мікроконтролера ATmega128 інтегровано апаратний модуль аналого-цифрового перетворювача (АЦП), загальну схему якого зображено на Рис.3.3.

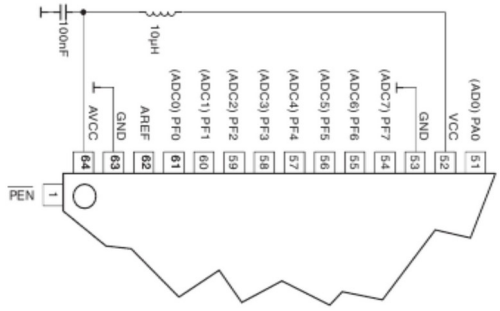


Рис. 3.3. Порт АЦП МК ATmega128

Апаратна реалізація аналого-цифрового перетворення в мікроконтролері ATmega128 інтегрована з лініями порту F (PORTF). Програмна взаємодія з модулем, налаштування його робочих режимів та імпорт результатів обчислень виконуються за допомогою спеціальних керуючих регістрів:

- ADMUX – використовується для вибору джерела опорної напруги (VREF) та керування вхідним аналоговим мультиплексором;
- ADCSRA – центральний регістр керування та стану, що забезпечує увімкнення блока АЦП, запуск циклу оцифрування, вибір коефіцієнта ділення тактової частоти (прескалера) та фіксацію прапорця завершення операції;
- ADCH та ADCL – пара вихідних регістрів (старший і молодший байти відповідно), які сумарно зберігають остаточний цифровий код після закінчення вимірювання;

- SFIOR – регістр спеціальних функцій, який відповідає за конфігурування додаткових тригерних джерел для автоматичного запуску процесу перетворення.

Регістр ADMUX

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. Вибір джерела опорної напруги для ADC

REFS1	REFS0	Джерело напруги ADC
0	0	Зовнішня опорна напруга , підключена до AREF, а внутрішня VREF відключена
0	1	AVCC із зовнішнім конденсатором на виводі AREF.
1	0	Не використовується
1	1	Внутрішня опорна напруга 2,56В із зовнішнім конденсатором - вивід AREF.

Граничні межі вимірювання інтегрованого АЦП безпосередньо залежать від обраної опорної напруги AREF. У випадках, коли величина вимірюваного сигналу стає більшою за рівень AREF, цифровий код приймає своє максимальне 10-бітне значення – 0x3FF. В архітектурі мікроконтролера ATmega128 реалізовано три альтернативні режими формування опорної напруги: 1. Використання напруги живлення аналогового каскаду AVCC, підключення якої відбувається за допомогою внутрішнього електронного ключа. 2. Активація інтегрованого джерела стабілізованої напруги з номіналом 2,56 В, що генерується внутрішнім прецизійним еталоном. 3. Подача зовнішнього опорної напруги безпосередньо на вивід AREF.

Зважаючи на високий вхідний опір виводу AREF, цей вузол є чутливим до наведень і розрахований на роботу суто з ємнісним навантаженням. Для придушення високочастотних завад і стабілізації опорного рівня рекомендується підключати блокувальний конденсатор між цим виводом та шиною заземлення (GND). Рівень опорної напруги можна проконтролювати шляхом підключення високоомного вольтметра до виводу AREF.

Результат оцифрування має 10-розрядну структуру і розподіляється між двома байтовими регістрами – ADCH та ADCL. Напрямок розрядного зсуву визначається конфігурацією прапорця ADLAR у регістрі ADMUX: за замовчуванням застосовується вирівнювання по правому краю, проте для завдань, де достатньо 8-бітної точності, можна увімкнути ліве вирівнювання, що дозволяє читати лише старший байт (ADCH).

Таблиця 3.2. Регістри даних АЦП

ADLAR=0:

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

ADLAR=1:

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

Комутація необхідного аналогового каналу та налаштування параметрів диференційного підсилювального каскаду реалізуються за допомогою конфігурування бітового поля MUX у регістрі ADMUX. За умов активації однополярного режиму вимірювання як джерело вхідного сигналу для блоку АЦП можна використовувати будь-яку з ліній ADC0–ADC7, шину нульового потенціалу (GND) або внутрішнє прецизійне джерело опорної напруги з номіналом 1,22 В. У диференційному режимі функціонування розробник може самостійно визначити інвертуючий та неінвертуючий вхідні канали

підсилювача. За такої конфігурації модуль АЦП оцифровує різницю потенціалів між цими двома виводами, яка попередньо масштабується відповідно до обраного коефіцієнта підсилення (Gain). Натомість в однополярній конфігурації підсилювальний тракт повністю вимикається з ланцюга проходження сигналу, і вимірювана напруга подається безпосередньо на вхід аналого-цифрового перетворювача. Нижче представлено архітектуру та розподіл бітів у регістрі 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	

Тактування інтегрованого АЦП забезпечується через спеціалізований попередній подільник (прескалер), який масштабує головну частоту процесора. Налаштування коефіцієнта ділення здійснюється ступінчасто по 2N (у межах від 2 до 128) за допомогою конфігурування бітового поля ADPS у регістрі ADCSRA (Табл.3.3).

Таблиця 3.3. Управління подільником ADC

ADPS2	ADPS1	ADPS0	Коефіцієнт поділу
0	0	0	1
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

Запуск прескалера відбувається автоматично в момент активації всього аналого-цифрового модуля шляхом встановлення прапорця ADEN. Відповідно, скидання цього біта (ADEN=0) повністю зупиняє роботу подільника частоти. Щоб забезпечити максимальну 10-бітну точність квантування, частоту синхронізації АЦП слід витримувати в оптимальному діапазоні від 50 до 200 кГц. Тривалість стандартного вимірювального циклу становить 13 тактів внутрішньої частоти АЦП. Винятком є перший (ініціалізаційний) запуск після увімкнення модуля, який триває довше – 25 тактів, що необхідно для стабілізації внутрішніх кіл. Після фіксації результату

дані переносяться у вихідні регістри, а апаратний прапорець переривання ADIF переходить у стан логічної одиниці. Особливості керування процесом залежать від обраного режиму роботи:

- Режим поодинокого вимірювання (Single Conversion): запуск оцифрування виконується встановленням біта ADSC, який автоматично скидається апаратурою після фіксації результату. Наступні цикли потребують повторного програмного встановлення цього прапорця.
- Режим безперервного сканування (Free Running): новий цикл вимірювання ініціюється автоматично відразу після завершення попереднього, при цьому біт ADSC постійно перебуває в активному стані.

Ініціалізація поодинокого циклу квантування реалізується шляхом встановлення прапорця ADSC у стан логічної одиниці. Цей біт підтримує високий рівень протягом усього часу оцифрування, після чого апаратно скидається в нуль.

У конфігурації безперервного сканування (Free Running) модуль АЦП виконує постійний циклічний огляд вхідної лінії, регулярно оновлюючи вміст вихідних регістрів. Для активації цього режиму необхідно записати логічну “1” у біт ADFR регістра ADCSRA. Первинний старт процесу все одно потребує одноразового програмного зведення біта ADSC. Після цього перетворення повторюються автономно, незалежно від того, чи було скинуто прапорець переривання ADIF.

Запуск автоматичного циклу базується на біті ADFR. Коли він активний, АЦП працює у фоновому режимі. За наявності в архітектурі апаратного автотригерування, керування початком вимірювань за певною подією координується за допомогою конфігурування бітового поля ADTS у регістрі спеціальних функцій SFIOR.

Ознакою завершення поточного вимірювального циклу є прапорець ADIF. Він автоматично переходить у стан логічної “1” одразу після того, як нові дані запишуться в байтову пару ADCH:ADCL. Якщо в системі програмно

дозволено обробку таких подій (активовано біт ADIE та зведено прапорець глобальних переривань I у регістрі стану SREG), мікроконтролер переходить до виконання відповідного вектора переривання. При цьому апаратний прапорець ADIF очищується автоматично під час виклику підпрограми обробки.

Дозвіл переривань за допомогою біта ADIE- цей біт діє як локальний дозвіл для генерації запитів на переривання від АЦП. Для того щоб система відреагувала на завершення оцифрування, необхідне одночасне виконання двох умов: встановлення біта ADIE та наявність глобального дозволу переривань у регістрі SREG.

Регістр SFIOR:

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. Біти (ADTS0, ADTS2, ADTS1) регістра SFIOR

Комбінація бітів ADTS[2:0]	Тригер активації (Джерело запуску)
000	Безперервне перетворення (Free Running)
001	Спрацьовування аналогового компаратора
010	Запит від зовнішнього переривання INT0
011	Збіг A для Таймера/Лічильника 0
100	Переповнення Таймера/Лічильника 0
101	Збіг B для Таймера/Лічильника 1
110	Переповнення Таймера/Лічильника 1
111	Подія захоплення (Capture Event) Таймера/Лічильника 1

Спосіб ініціалізації процесу аналого-цифрового перетворення координується станом прапорця ADATE у регістрі ADCSRA, а також конфігурацією бітового поля ADTS. Якщо біт автоматичного тригерування ADATE зведений у логічну “1”, вибір події повністю покладається на комбінацію бітів ADTS. Якщо ж цей прапорець скинутий (ADATE=0), значення в полі ADTS ігноруються мікроконтролером. Безпосередня активація вимірювального циклу відбувається по висхідному фронту обраного імпульсного сигналу.

Необхідно підкреслити, що перехід до конфігурації безперервного циклічного опитування (коли встановлено код ADTS[2:0]=0) не генерує

автоматичного стартового імпульсу самостійно, навіть за умови активного статусного прапорця ADIF. Синтез сигналів із широтно-імпульсною модуляцією (ШІМ) реалізується за допомогою відповідних апаратних виводів порівняння таймерів (OC1A/B/C та OC3A/B/C). Структура периферії ATmega128 містить два потужні 16-розрядні модулі: Таймер/Лічильник 1 та Таймер/Лічильник 3. Їхні регістри здатні оперувати даними в діапазоні від 0 до 65536 (2^{16}). Функціональне призначення цих таймерів охоплює вирішення таких інженерних завдань: гнучка генерація сигналів із широтно-імпульсною модуляцією (ШІМ); робота в режимі високоточного формувача прямокутних імпульсів із заданою частотою (режим CTC); інтеграція трьох незалежних каналів апаратного порівняння для кожного з модулів (OC1A/B/C у першому таймері та OC3A/B/C — у третьому).

Завдяки своїй 16-бітній організації, таймер 3 є універсальним інструментом для точного вимірювання часових інтервалів, фіксації кількості зовнішніх імпульсів, а також формування ШІМ-сигналів із керованою тривалістю та шпаруватістю на вихідних лініях порівняння. Абсолютно ідентичний набір інструментів та режимів доступний розробнику і при роботі з таймером 1. Детальна внутрішня топологія та алгоритми взаємодії цих 16-розрядних таймерів представлені на схемі (Рис.3.4).

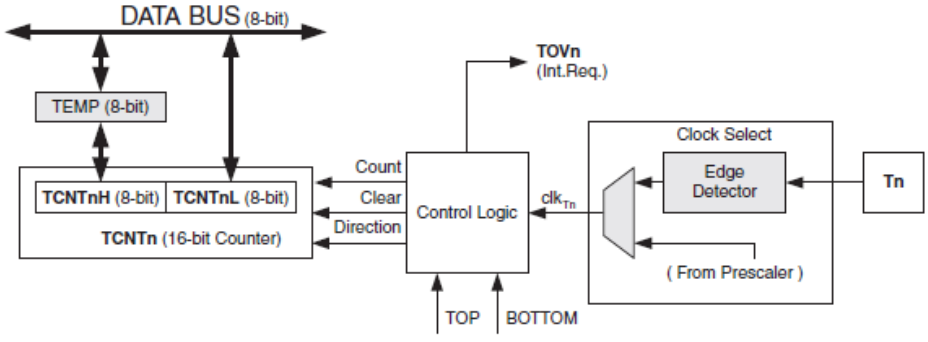


Рис.3.4. 16-ти розрядні таймери. Блок схема

Алгоритм функціонування лічильника базується на взаємодії сукупності сигналів та констант:

- Count (Рахунок): ініціює зміну поточного вмісту лічильного регістра TCNTn на одну дискрету (виконує інкремент або декремент).

- Direction (Напрямок): визначає напрям рахунку — на збільшення чи на зменшення значення.
- Clear (Очищення): здійснює апаратне обнулення регістра TCNTn.
- Clk (Тактування): послідовність стробуючих імпульсів від прескалера, що забезпечує часову синхронізацію лічильника.
- TOP (Верхня межа): поріг, що фіксує досягнення лічильником максимального значення для поточного режиму.
- BOTTOM (Нижня межа): прапорець, який активується у момент обнулення або досягнення мінімального стану в регістрі TCNTn.

Оскільки внутрішня архітектура мікроконтролера має 8-розрядну шину даних, доступ до 16-бітної регістрової пари TCNTn, яка розділена на старший (TCNTH) та молодший (TCNTL) байти, організовано через спеціальний тимчасовий регістр (TEMP). Зчитування та модифікація лічильника вимагають чіткої послідовності дій:

1. Операція запису: спочатку програма повинна завантажити дані у старший байт (TCNTH), і лише після цього здійснюється запис у молодший байт (TCNTL).
2. Операція зчитування: першим обов’язково опитується молодший байт (TCNTL), а за ним – старший (TCNTH).

Нижче наведено перелік режимів роботи 16-бітного таймера/лічильника 3. Виділяють такі базові режими роботи модуля:

Звичайний режим (Normal Mode). Найбільш проста конфігурація, за якої вміст регістра TCNT3 безперервно інкрементується з кожним тактом Clk. Рахунок триває до досягнення верхньої межі $FFFF (рівень TOP). Наступний імпульс викликає циклічний перехід через нуль (апаратне переповнення), внаслідок чого лічильник скидається у стан $0000. У цей момент генерується прапорець переповнення TOV3 у регістрі TIFR. За умови активації маски TOIE3 (в регістрі TIMSK) мікроконтролер переходить до обробки відповідної підпрограми переривання.

Режим скидання при збігу (CTC – Clear Timer on Compare Match). Цей режим за структурою схожий на Normal Mode, проте максимальна межа рахунку (TOP) є динамічною. Замість фіксованого порогу $FFFF, верхня межа задається розробником шляхом запису константи у регістри порівняння OCR3A/B/C або регістр захоплення ICR3. Як тільки значення в TCNT3 зрівнюється із вмістом обраного регістра, відбувається автоматичне обнулення таймера. У момент такого збігу встановлюється прапорець події порівняння OCF3A/B/C (або ICF3).

Режим швидкої ШІМ (Fast PWM). Дана конфігурація орієнтована на генерацію високочастотних сигналів із широтно-імпульсною модуляцією. Лічильник працює в односпрямованому режимі від $0000 до точки TOP, що робить його схожим на Normal Mode. Проте ключова особливість полягає в апаратному керуванні станом вихідних ліній порівняння при переході через нуль та при збігу зі значенням в OCR3x, що дозволяє формувати стабільну ШІМ-послідовність.

Повний набір регістрів, необхідних для налаштування 16-бітного таймера 3, наведено нижче:

Таблиця 3.5. Регістр управління 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 5:4 – COMnB1:0: налаштування каналу B;
- Bit 7:6 – COMnA1:0: налаштування каналу A;
- Bit 3:2 – COMnC1:0: налаштування каналу C;

Таблиця 3.6. Регістр управління 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: Встановлення біту, вмикає фільтрування шуму на вході ICP3:

- Bit 5 – зарезервовано;
- Bit 4:3 – WGMn3:2: для вибору режиму роботи таймера (Табл.3.7).

Таблиця 3.7. Режими роботи таймера

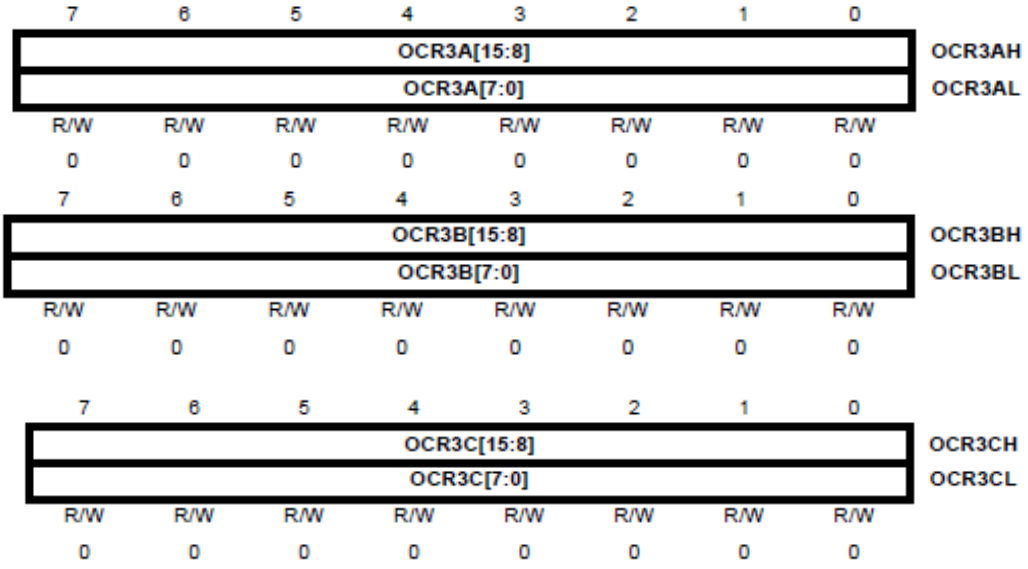
Режим	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter режими роботи	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	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 таймера (Табл.3.8)

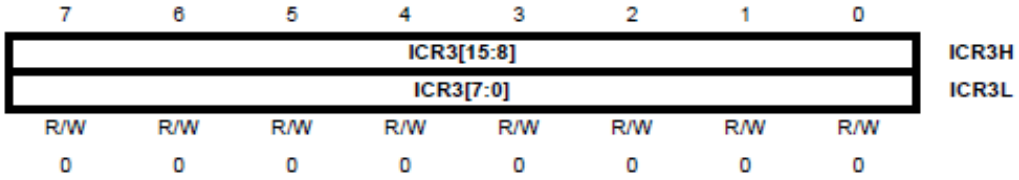
Таблиця 3.8. Джерела імпульсів синхронізації таймера

CSn2	CSn1	CSn0	Пояснення
0	0	0	**Джерело синхронізації лічильника не вибрано**
0	0	1	$clk_{I/O}$/1 **Подільник рівний 1**
0	1	0	$clk_{I/O}$/8 **Подільник рівний 8**
0	1	1	$clk_{I/O}$/64 **Подільник рівний 64**
1	0	0	$clk_{I/O}$/256 **Подільник рівний 256**
1	0	1	$clk_{I/O}$/1024 **Подільник рівний 1024**
1	1	0	**Зовнішнє джерело на** Tn pin. **Перемикання по спаду**
1	1	1	**Зовнішнє джерело на** Tn pin. **Перемикання по фронту**

Таблиця 3.9. Регістри OCR3An, OCR3Bn, OCR3Cn.



Таблиця 3.10. Регістр ICR3n



Регістр ICR3n можна налаштівати для встановлення значення TOP (режим 14. Табл.3.7). В Табл.3.11. наведено сигнали порта PE (OC3C, OC3A, OC3B,) для режиму Fast PWM, вони залежать від бітів регістра TCCR3A (Табл.3.5).

Таблиця 3.11. Режими Fast PWM

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)

Апаратний спосіб генерації широтно-імпульсної модуляції (ШІМ) має суттєві переваги порівняно з його програмною імітацією. Завдяки задіянню

інтегрованих периферійних модулів таймера повністю відпадає потреба в постійному використанні ресурсів центрального процесора для відліку часових інтервалів та формування імпульсів. Це дозволяє розвантажити обчислювальне ядро МК і перенаправити його потужність на виконання складніших логічних та пріоритетних завдань у режимі реального часу.

Після завершення етапу конфігурування регістрів таймер-лічильник переходить на автономне функціонування на схемотехнічному рівні. Процесор повністю звільняється від рутинного контролю ліній вводу-виводу, а його звернення до регістрів таймера відбувається виключно за потреби динамічного коригування коефіцієнта шпаруватості або частоти вихідного ШІМ-сигналу

Давач тиску BMP180. Цифровий барометричний сенсор BMP180, розроблений компанією Bosch Sensortec GmbH, виготовлений за передовою технологією мікроелектромеханічних систем (MEMS).

Конструктивно пристрій реалізовано у вигляді монолітного кремнієвого кристала, який об'єднує первинний п'єзорезистивний перетворювач та спеціалізовану мікросхему для оцифрування й первинної обробки сигналів. Основними цільовими напрямками використання датчика є створення портативних навігаційних приладів, цифрових альтиметрів (висотомірів) та локальних метеорологічних станцій.

Функціональні можливості сенсора дозволяють вимірювати абсолютний атмосферний тиск у діапазоні від 300 до 1100 гПа. Це забезпечує визначення висоти в інтервалі від -500 до +9000 метрів відносно середнього рівня моря. Інтегрований температурний датчик забезпечує компенсацію термічної похибки та працює в межах від -40°C до +85°C. Модуль адаптований для низьковольтних систем із напругою живлення 1,8...3,6 В, що гарантує низьке енергоспоживання, а обмін даними з мікроконтролером відбувається через послідовну шину I^2C.

На основі первинних телеметричних даних BMP180 можна обчислити не лише тиск і температуру, а й абсолютну/відносну висоту об’єкта, а також динаміку його вертикального руху (швидкість підйому чи спуску). Цей пристрій є модернізованою та повністю сумісною заміною застарілого сенсора BMP085. Експлуатаційні параметри сенсора BMP180:

- граничні температурні умови роботи: від -40 °C до +85 °C;
- зона робочих температур: 0 °C … +65 °C (забезпечує максимальну точність вимірів);
- діапазон напруг живлення становить 1,8–3,6 В (номінальне значення — 3,3 В);
- точність вимірювання тиску: абсолютна похибка < ±0,12 гПа;
- похибка розрахунку висоти: становить ±1 м;
- ооздільна здатність (дискретність): для каналу тиску дорівнює 0,01 гПа, для каналу температури — 0,1 °C;
- похибка вбудованого термометра: у прецизійній зоні (0…+65 °C) становить ±1 °C.

Час, необхідний для виконання одного циклу вимірювання тиску в архітектурі BMP180, визначається поточним рівнем передискретизації (oversampling). Розробнику доступно кілька програмних конфігурацій, які дозволяють гнучко комбінувати швидкість роботи пристрою із точністю отриманих даних:

1. Профіль енергозбереження (Ultra Low Power): швидкісне опитування за 4,5 мс.
2. Стандартна конфігурація (Standard): тривалість формування відліку дорівнює 7,5 мс.
3. Покращена роздільна здатність (High Resolution): час вимірювання становить 13,5 мс.
4. Максимальна точність (Ultra High Resolution): для апаратної обробки становить 25,5 мс.

5. Гранична (екстремальна) прецизійність: тривалість одного перетворення рівна 76,5 мс.

На відміну від тиску, тривалість циклу для температурного датчика є стабільною і завжди становить 4,5 мс. Для забезпечення високої швидкості передачі даних апаратний інтерфейс датчика здатний працювати на тактовій частоті до 3,4 МГц. У продажі є готові модулі з сенсором тиску BMP180 (Рис.3.5).

Рис. 3.5. Модуль з BMP180 фірми Bosch Sensortec GmbH

Призначення пінів BMP180 зображено на Рис.3.6.

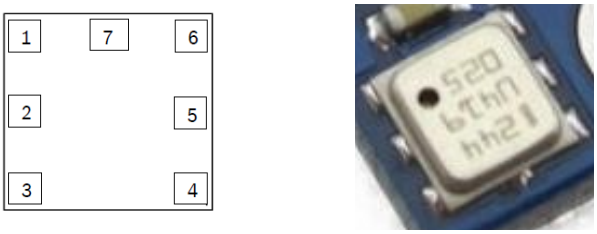


Рис. 3.6. Нумерація виводів давача BMP180

Призначення виводів BMP180: вивід 1: резервний (не задіяний); вивід 2 (VDD): вхід джерела живлення; вивід 3 (VDDIO): лінія живлення інтерфейсу I2C. вивід 4: не задіяний; вивід 5 (SCL): вхід тактування послідовної шини; вивід 6 (SDA): лінія прийому – передачі послідовних даних; вивід 7 (GND): загальний – шина (“земля”).

Конструктивне виконання барометра BMP180 має високу механічну міцність, що дозволяє йому витримувати перевантаження тиском до 10 000 гПа без порушення функціональності та втрати точності.

Обмін інформацією з головним мікроконтролером здійснюється за допомогою стандартного двопровідного інтерфейсу I^2C. З метою компенсації інструментальних похибок, які виникають під час детектування фізичних величин, у системі застосовується алгоритм термічної та барометричної

корекції за допомогою персональних коефіцієнтів. Ці калібрувальні константи визначаються індивідуально для кожної мікросхеми під час заводського тестування та інтегруються в його внутрішню енергонезалежну пам’ять (EEPROM). Датчик містить такі базові апаратні вузли: п’єзорезистивний чутливий елемент (первинний перетворювач); вбудований аналого-цифровий перетворювач (АЦП); логічний контролер керування периферією та інтерфейсом; енергонезалежний масив пам’яті EEPROM об’ємом 176 біт, призначений для зберігання заводських коефіцієнтів корекції.

На виході сенсора формуються два інформаційні масиви: 16-розрядне невідкориговане значення температури (UT) та цифровий код тиску (UP), розрядність якого динамічно змінюється в межах від 16 до 19 біт залежно від заданого рівня передискретизації. Типова схема підключення модуля BMP180 до мікропроцесорного ядра наведена далі на рисунку (Рис.3.7).

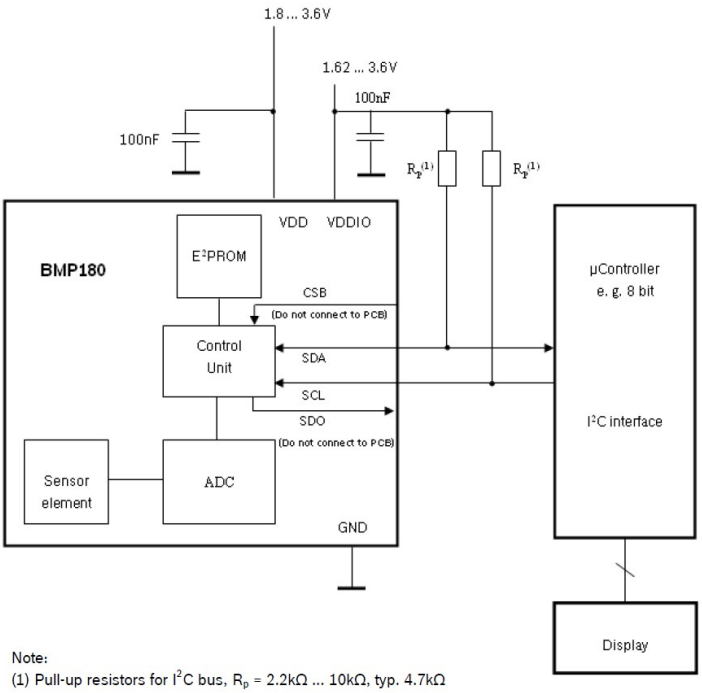


Рис. 3.7. Стандартне вмикання BMP180 до МК

Вимірювання температури і тиску. Процес зняття показників ініціюється мікроконтролером, який надсилає стартову команду для запуску внутрішнього циклу аналого-цифрового перетворення (температури або атмосферного тиску). Після завершення апаратного обчислення всередині

сенсора, первинні некомпенсовані масиви даних (UT або UP) передаються по шині через послідовний інтерфейс I^2C.

Переведення цифрових кодів у загальноприйняті фізичні одиниці (градуси Цельсія та гектопаскалі [гПа]) здійснюється шляхом математичного опрацювання з допомогою унікальних калібрувальних констант. Опитування цих коефіцієнтів із внутрішньої пам'яті датчика є одноразовою операцією, що виконується під час стартового завантаження мікропроцесорної системи.

У базовому робочому режимі інтенсивність квантування барометричного каналу може становити до 128 відліків за одну секунду. За такої високої дискретизації канал температури опитується з періодичністю всього один раз на секунду. Отриманий температурний показник фіксується і застосовується як опорна величина для математичної термокомпенсації кожного зі 128 барометричних вимірювань. Детальний опис топології внутрішнього регістрового простору BMP180 подано в Табл.3.12.

Таблиця 3.12. Регістри 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	BFh down to AAh	calib21<7:0> downto calib0<7:0>								n/a

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

Керуючі та службові регістри барометричного датчика. Регістр конфігурування вимірювань (CTRL_MEAS, доступ за адресою 0xF4): цей апаратний регістр застосовується для ініціалізації циклів опитування датчика, а також для встановлення параметрів точності квантування.

- Біт SCO (Bit 5): виконує роль тригера запуску та прапорця стану перетворення. Запис логічної одиниці в цю позицію дає команду сенсору розпочати вимірювальний цикл. Протягом усього часу апаратної обробки сигналу біт підтримує високий потенціал, а після занесення результату у вихідні буфери автоматично очищується процесором датчика.

- Бітове поле OSS (Bits 7:6): задає коефіцієнт передискретизації (oversampling), визначаючи баланс між роздільною здатністю та часовою затримкою. Архітектурою передбачено такі комбінації: 00 — базовий режим (тривалість перетворення становить 4,5 мс); 01 — подвійне програмне осереднення (час очікування — 7,5 мс); 10 — чотирикратне згладжування (апаратна затримка — 13,5 мс); 11 — восьмикратна передискретизація (робочий цикл триває 25,5 мс).

Регістр програмної ініціалізації скидання (SOFT_RESET, адреса 0xE0): спеціалізована комірка пам’яті, доступна виключно в режимі запису. Надсилання в цей регістр коду 0xB6 викликає повне перезавантаження внутрішньої логіки датчика, що за своєю ефективністю еквівалентно тимчасовому вимкненню лінії живлення.

Регістр ідентифікатора пристрою (CHIP_ID, доступ за адресою 0xD0): енергонезалежний регістр, який містить фіксовану заводську константу 0x55. Використовується майстер-пристроєм (мікроконтролером) для верифікації сенсора та перевірки його успішного підключення до шини I^2C. Зчитування зафіксованих результатів вимірювання у внутрішню пам'ять мікроконтролера здійснюється шляхом звернення до вихідних регістрів даних, що розташовані за адресами 0xF6, 0xF7 та 0xF8. Архітектурною перевагою цього барометричного модуля є гнучкість інтерфейсу, яка дозволяє зчитувати зазначені байти у довільній послідовності.

Комунікація із зовнішнім МК реалізована на базі двопровідного протоколу I^2C, який у даній модифікації мікросхеми здатний функціонувати у високошвидкісних режимах (High-Speed Mode) із пропускною здатністю до 3,4 Мбіт/с. Для забезпечення стабільного фронту імпульсів та правильного формування логічних рівнів, сигнальні лінії шини (SCL та SDA) необхідно підключити до позитивної шини живлення (VCC) через підтягуючі (pull-up) резистори. Базовий рекомендований номінал цих елементів дорівнює 4,7 кОм, проте залежно від ємності лінії допускається використання резисторів у межах від 2,2 до 10 кОм.

На фізичному рівні ідентифікація модуля під час сесії зв’язку визначається типом транзакції: для надсилання директив і команд інтерфейс використовує 8-бітну адресу 0xEE, тоді як для операцій приймання інформаційних байтів застосовується адреса 0xEF.

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

Алгоритм обміну інформаційними пакетами за двопровідним протоколом I2C (ілюстрація часових діаграм наведена на Рис.3.8) будується на послідовній зміні трьох базових апаратних станів: генерації умови START, безпосередньої фази трансляції байтів та фіксації умови STOP. Покрокова логіка взаємодії пристроїв у такій мережі включає такі фази:

1. Формування умови старту (START condition): Будь-яка сесія зв’язку ініціюється виключно ведучим пристроєм (мастером/ мікро-контролером). Апаратний стан «Старт» детектується на шині тоді, коли на тактовій лінії SCL утримується стабільний високий потенціал, а на інформаційній лінії SDA фіксується спадаючий фронт (перехід від логічної одиниці до нуля).
2. Фаза адресації периферії: Одразу після стартового імпульсу мікроконтролер передає в мережу 7-бітний ідентифікатор (адресу) потрібного відомого пристрою. Наступний за ним 8-й біт (R\W) координує напрямок потоку даних: він визначає, чи буде здійснюватися запис конфігураційних команд у датчик BMP180, чи відбуватиметься зчитування телеметрії з його регістрів.
3. Апаратне підтвердження (ACK - Acknowledge): У випадку верифікації надісланої адреси, підключений датчик сигналізує про готовність до обміну. Для цього він примусово занижує лінію SDA (встановлює логічний нуль) на період тривалості дев'ятого синхроімпульсу на лінії SCL.
4. Трансляція інформаційних байтів: Зміна інформаційних рівнів на провіднику SDA допускається в ті моменти, коли на лінії SCL діє

низький рівень. Натомість у фазі високого стану тактового сигналу шина даних SDA повинна залишатися незмінною.

5. Формування умови стопу (STOP condition): Завершення поточної сесії обміну реалізується генеруванням стану “Стоп”. Для цього на лінії SCL спочатку фіксується високий рівень, після чого на лінії SDA формується висхідний фронт (перехід у лог. “1”), як показано на графіку (Рис. 3.8).

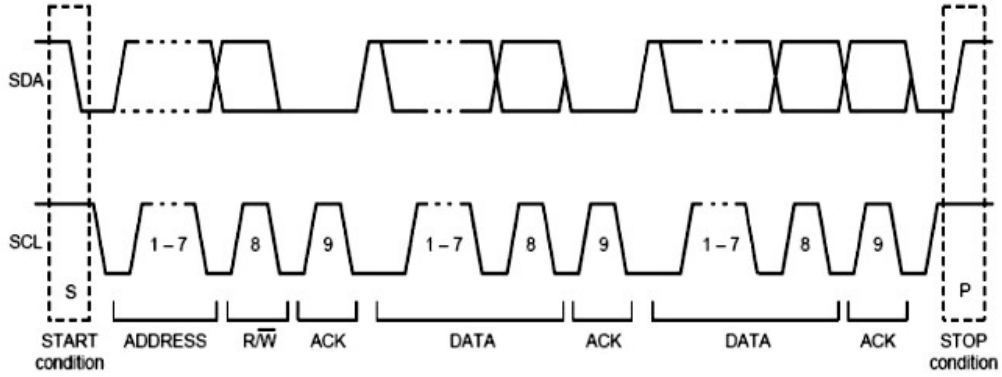


Рис. 3.8. Протокол обміну по інтерфейсу – I^2C

Процедура зчитування телеметричних показників тиску й температури за допомогою послідовного інтерфейсу I^2C виконується у вигляді строго регламентованої послідовності апаратних кроків. Початок транзакції починається генерацією ведучим пристроєм (мікроконтролером) умови START. Одразу після цього хост-процесор передає в лінію 7-бітну фізичну адресу датчика, доповнену бітом напрямку транзакції (в даному разі – запис).

Наступним етапом є надсилання внутрішньої адреси цільового регістру та відповідного керуючого коду (директиви), що запускає процес оцифрування всередині сенсора. Специфіка реалізації цього обміну полягає в тому, що після успішного приймання кожного окремого байта інформації внутрішня логіка модуля BMP180 формує у відповідь апаратний імпульс підтвердження (ACKS). Вичерпна часова діаграма взаємодії між обчислювальними компонентами зафіксована далі (Рис.3.9).

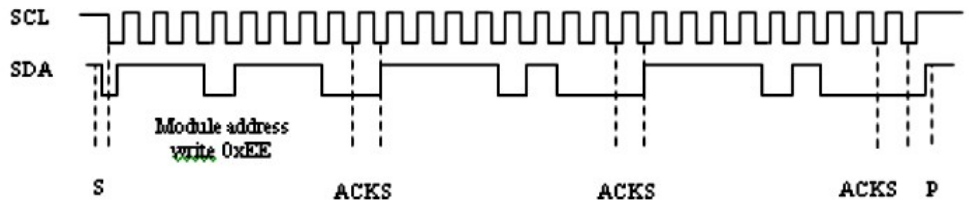


Рис. 3.9. Читання даних по інтерфейсу I^2C

Опис РКД (LCD). Для забезпечення наочного виведення телеметрії та поточних параметрів розроблюваного пристрою оптимальним рішенням є застосування символьних рідкокристалічних дисплеї . Головними перевагами цих засобів індикації є висока інформативність при вкрай низькому рівні споживання струму. Наявність інтегрованого модуля світлодіодного підсвічування гарантує чітке зчитування текстових даних оператором як при слабкому зовнішньому освітленні, так і в умовах його повної відсутності.

Надійне функціонування у широких температурних межах (від -20°С до +70°С дозволяє успішно експлуатувати такі РК-модулі у складі бортової та мобільної апаратури. З конструктивного погляду дисплей являє собою моноблок, що об'єднує рідкокристалічну матрицю та спеціалізований контролер (драйвер). Базова модель екрана, яка проілюстрована на Рисунку 3.10, орієнтована на виведення двох рядків тексту по 16 знакомісць у кожному. Кожен окремий символ генерується всередині піксельної матриці розмірністю 5x8 пікселів, причому для забезпечення нормального читання тексту між сусідніми символами закладено технологічний зазор шириною в один піксель.

Рис. 3.10. РКД (WH1602B-YGK-CTK)

Апаратною основою модуля є інтегрований драйвер, який за своєю системою команд та архітектурою повністю сумісний із промисловим стандартом HD44780. Екранна матриця орієнтована на виведення буквенно-цифрових даних у конфігурації 2 рядки по 16 позицій (знакомісць). Оптимальна видимість інформації досягається завдяки світлодіодному (LED) підсвічуванні із жовто-зеленим свіченням. Логіка функціонування дисплея базується на апаратній взаємодії внутрішнього контролера РК-матриці та інтегрованого масиву пам’яті. В оперативній пам'яті дисплея (DDRAM) за

кожною координатою знакомісця на екрані жорстко закріплена адреса відповідної комірки, де зберігається цифровий код символу.

Внутрішня топологія знакогенератора розділена на два функціональні сегменти: перший (CGROM) містить фіксовані заводські матриці стандартних символів, а другий (CGRAM) надає розробнику можливість програмно створювати та записувати власні унікальні графічні знаки або піктограми. Матриця знакогенератора містить дві кодові сторінки, що дозволяє паралельно відображати символи європейських алфавітів (латиницю) та знаки кириличного шрифту.

Спряження модуля з мікроконтролером може здійснюватися за допомогою двох варіантів паралельного інтерфейсу – з 4-розрядною або 8-розрядною шиною даних. Вибір конкретного режиму комунікації визначається програмним шляхом під час стартового налаштування (ініціалізації) дисплея. Апаратна логіка індикатора адаптована для прийому команд від мікроконтролера, а також підтримки операцій читання/запису безпосередньо через паралельну шину.

Передбачено механізм зчитування статус-байта (прапорця зайнятості Busy Flag), що дозволяє МК моніторити поточний стан модуля і блокувати передачу нових даних до завершення поточного циклу. Створено окрему область пам’яті для проектування та збереження до 8 власних унікальних графічних піктограм або символів користувача.

Реалізовано програмне керування конфігурацією курсора (фіксований маркер або миготливе знакомісце), а також апаратна й програмна корекція яскравості підсвічування та рівнів контрасту матриці. Для детального вивчення алгоритмів функціонування знакосинтезуючих екранів необхідно розглянути внутрішню топологію базової мікросхеми HD44780. Програмний код взаємодіє безпосередньо з цим ядром, тоді як допоміжні апаратні вузли (блоки зсуву, контролери сегментів, знакогенератор) беруть на себе завдання фонової регенерації зображення і позиціонування курсора без залучення ресурсів зовнішнього МК.

Узгодження дій між МК та РК-контролером реалізовано через два автономні регістри: командний (IR) та інформаційний (DR). Селекція потрібного буфера здійснюється логічним рівнем на керуючій лінії RS:

- При низькому рівні сигналу (RS=0) виконується доступ до регістра інструкцій (IR).
- При високому рівні сигналу (RS=1) активується лінія зв’язку з регістром даних (DR).

Вміст, що надходить у буфер DR, автоматично перенаправляється у відеопам’ять екранного буфера (DDRAM) або в область користувацьких шрифтів (CGRAM), керуючись поточним станом лічильника адреси (AC). Регістр інструкцій (IR) транслює отримані коди команд безпосередньо на блок дешифратора та виконання. Загальний обсяг екранного ОЗП становить 80 байт, що еквівалентно збереженню кодів текстових символів у конфігурації двох рядків по 40 знакомісць. Дана логічна організація є універсальним стандартом: незалежно від фізичних габаритів матриці (наприклад, 20x4 або 8x1), вбудований контролер завжди оперує пам’яттю як віртуальним масивом розміром 2x40символів.

Оскільки мікросхема HD44780 функціонує за алгоритмом динамічного сканування, вона безперервно оновлює стан рідкокристалічних комірок. Власних потужностей базового драйвера вистачає для безпосереднього керування 40 виходами сегментів (SEG1...SEG40), чого достатньо для обслуговування екранів ємністю до 8 знаків. Дисплеї з більшою щільністю інформації потребують інтеграції додаткових мікросхем розширення (наприклад, каскадних драйверів HD44100), кожен з яких додає до системи ще 40 каналів керування сегментами.

Для спрощення розробки ПЗ у середовищі CodeVisionAVR передбачено спеціалізовану бібліотеку, що містить готовий набір функцій для конфігурування таких індикаторів і забезпечує коректне спряження з різноманітними модифікаціями РК-модулів – від малогабаритних однорядкових моделей 1x8 до повноформатних дисплеїв 2x40, включаючи

поширені інженерні стандарти 2x16 та 4x20. Схема модуля LCD з МК ATmega128 (Рис.3.11.)

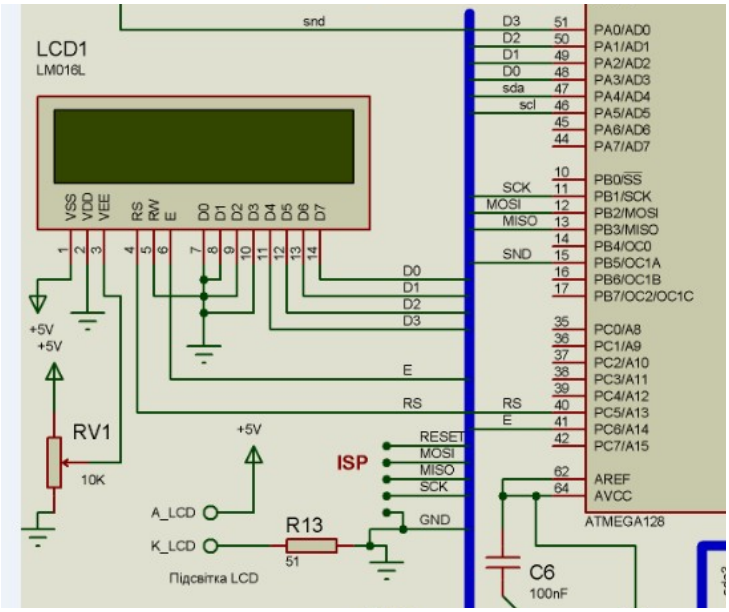


Рис. 3.11. Схема під’єднання символьного LCD до МК ATmega128

3.2. Проектування висотоміра в САПР Proteus VSM

Цифровий альтиметр орієнтований на безперервний моніторинг базових метеорологічних та навігаційних параметрів, серед яких: абсолютний атмосферний тиск, температура навколишнього середовища, геодезична висота над рівнем моря (абсолютна альтитуда), а також відносна висота об’єкта. Окрім локальної індикації, в архітектурі пристрою реалізовано функцію трансляції інформаційних масивів через послідовний комунікаційний інтерфейс до персонального комп'ютера, а також інтерактивну взаємодію з оператором за допомогою структурованого екранного меню та рідкокристалічного екрана.

Наведені на Рис.3.12 блоки структурної схеми демонструють взаємозв'язок між програмним забезпеченням та фізичними компонентами системи. Головним елементом обробки даних є 8-бітний мікроконтролер архітектури AVR – ATmega128, який здійснює керування інформаційними потоками, що надходять від цифрового MEMS-сенсора BMP180. Схемотехніка приладу додатково містить спеціалізований модуль конвертації

логічних рівнів, клавіатурну матрицю керування, світлодіодні індикатори поточного стану та знакосинтезуючий дисплей, призначений для виведення результатів вимірювань з метою їхнього оперативного аналізу.

Для забезпечення апаратної сумісності периферійних вузлів інтегровано схему узгодження рівнів, побудовану на базі лінійного стабілізатора напруги U1, польових транзисторів Q1 та Q2, резистивного дільника R3–R6 і блокувальних конденсаторів C4, C5. Зазначений вузол формує стабільну напругу живлення 3,3 В для безпечної роботи сенсора BMP180 та здійснює двонаправлене перетворення амплітуди сигналів на лініях зв'язку між сенсором і портами введення-виведення мікроконтролера.

Зміна режимів та навігація по внутрішньому інтерфейсу висотоміра реалізована за допомогою чотирьох клавіш:

- “MENU/ENTER” та “EXIT” – призначені для відкриття пунктів меню, підтвердження дій та повернення на попередній рівень меню;
- “SELECT+” та “SELECT-“ – слугують для збільшення або зменшення числових значень параметрів.

Система базується на мікроконтролері ATmega128, функціональні обов’язки якого охоплюють циклічне опитування датчика по шині, дешифрацію сигналів від кнопок, комутацію світлодіодних ліній, а також дублювання вимірювальних даних на LCD-екран формату 16x2 та у послідовний порт UART для зв’язку з ПК.

Візуальний контроль поточного режиму роботи пристрою забезпечується кольоровою LED-індикацією: світлодіод жовтого кольору сигналізує про активність режиму вимірювання абсолютної альтитуди, тоді як синій індикатор вмикається під час підрахунку відносної висоти.

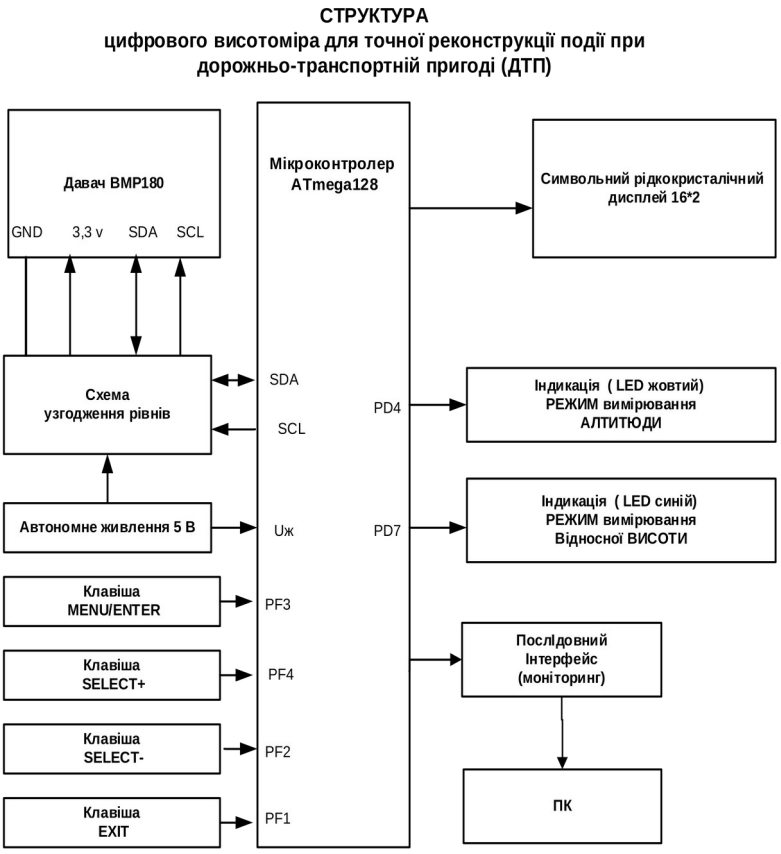


Рис. 3.12. Загальна структура висотоміра

На Рис.3.13 зображено апаратне забезпечення висотоміра створене в САПР Proteus VSM.

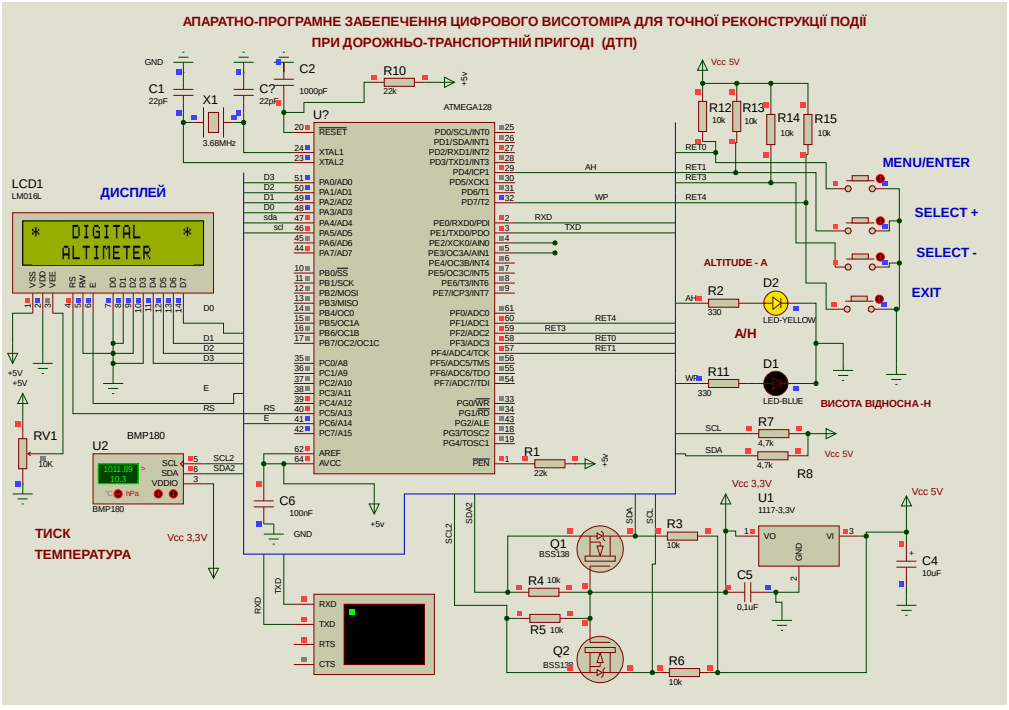


Рис. 3.13. Апаратне забезпечення висотоміра спроектоване в САПР Proteus VSM

Зібрані дані щодо температури та тиску обробляються мікроконтролером, після чого виводяться на рідкокристалічний екран та передаються на персональний комп’ютер. Комунікація з цифровим датчиком U2 організована через сигнальні лінії SDA і SCK, які підключені до портів PA4 та PA5 через схему узгодження логічних рівнів. Для виведення текстових даних використано символьний індикатор марки WH1602B-YGK-CTK, що взаємодіє з шиною порту A. Блок індикації та інтерфейсу користувача має таку конфігурацію:

- Кнопка переходу в меню (MENU/ENTER) комутується з виводом PF3;
- Клавіші вибору SELECT+ та SELECT- підведені до пінів PF4 і PF2 відповідно;
- Кнопка скидання/повернення в початковий режим (EXIT) задіює пін PF1;
- Сигнальні світлодіоди D2 (жовтого кольору) та D1 (синього кольору) підключені до контактів PD4 і PD7 відповідно.

РОЗДІЛ 4
ПРОГРАМНО-АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ВИСОТОМІРА

4.1. Алгоритм роботи висотоміра

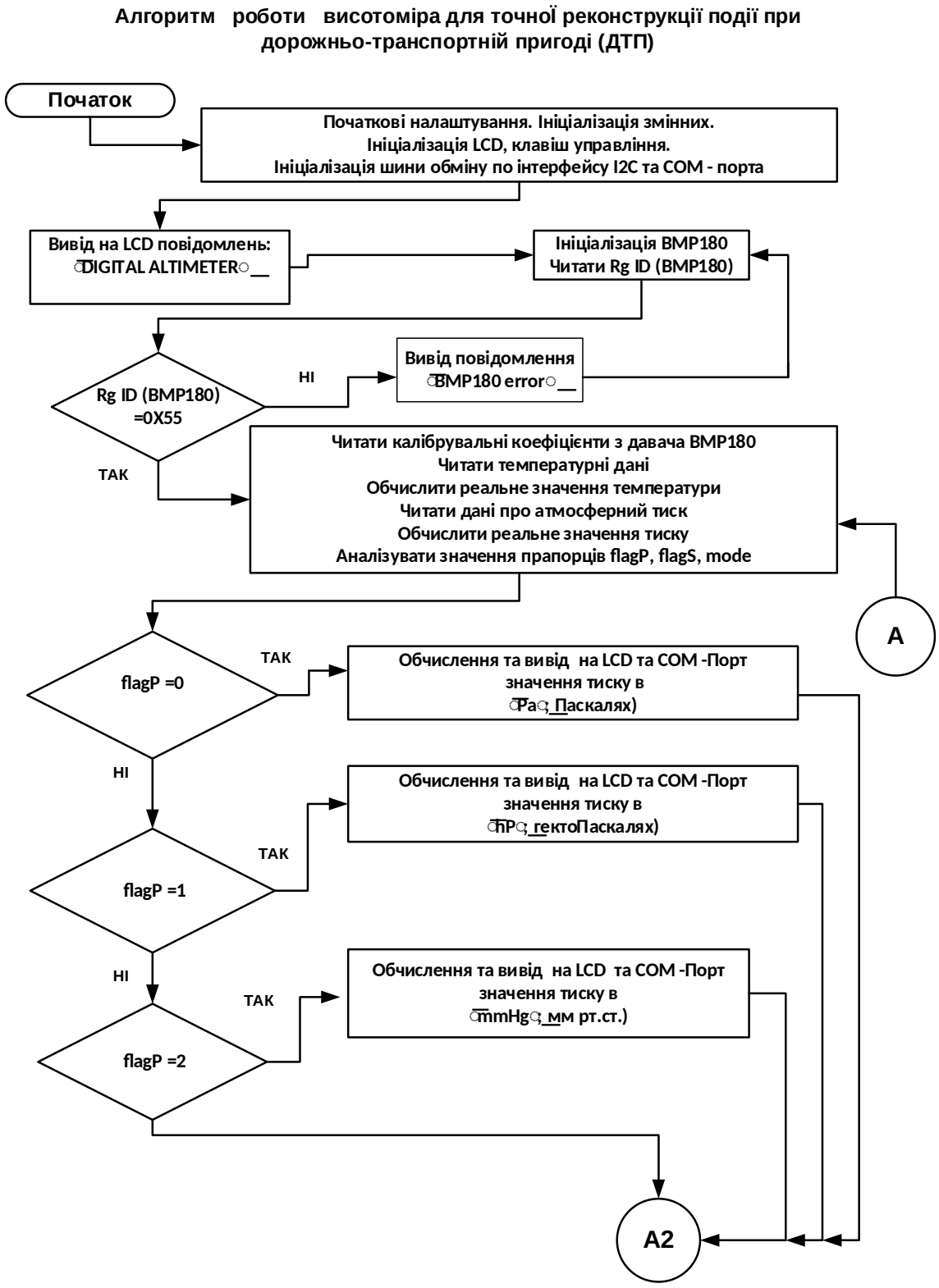


Рис. 4.1 Алгоритм роботи висотоміра (початок)

Алгоритм роботи висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП) (продовження 2)

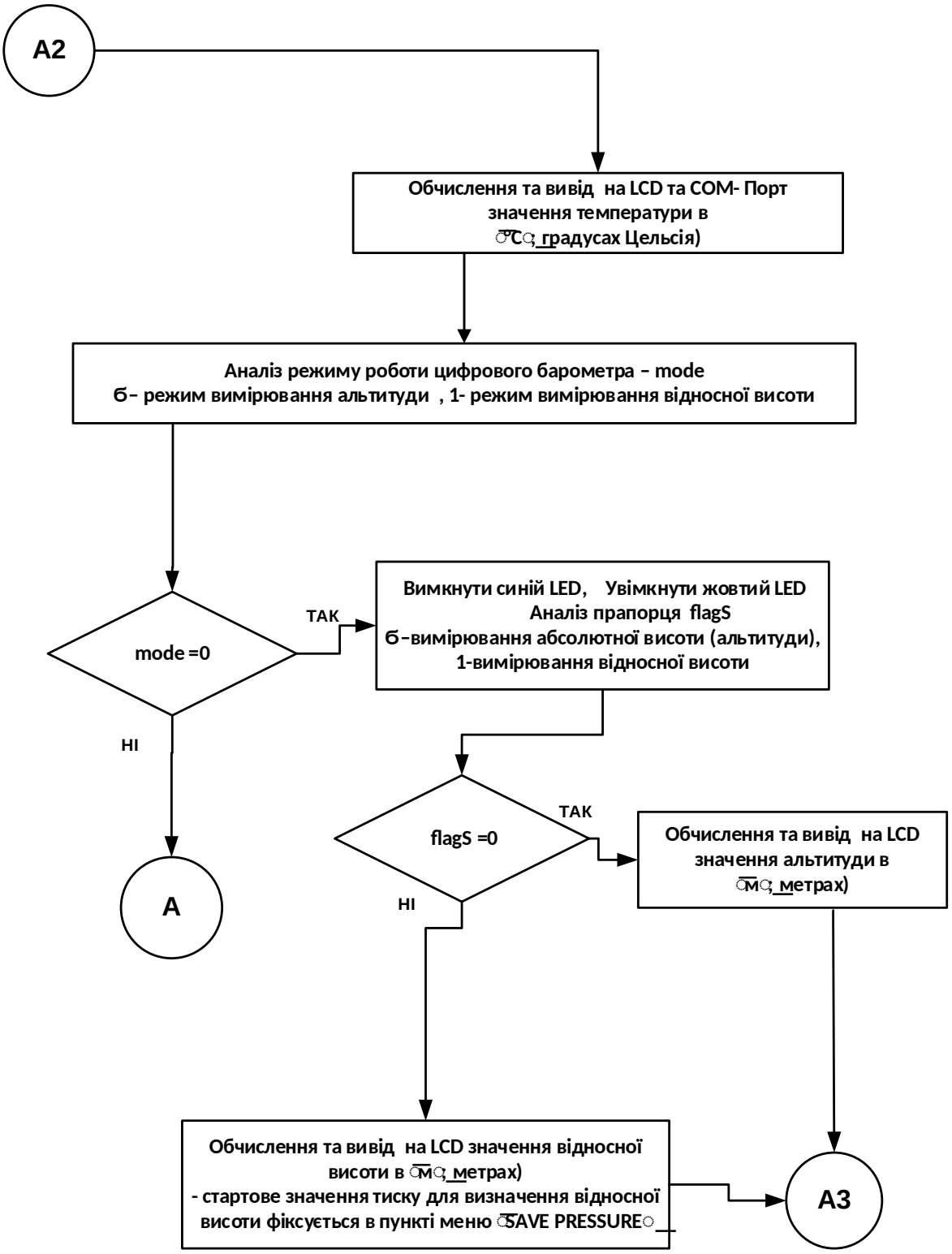


Рис. 4.2. Алгоритм роботи висотоміра (продовження 2)

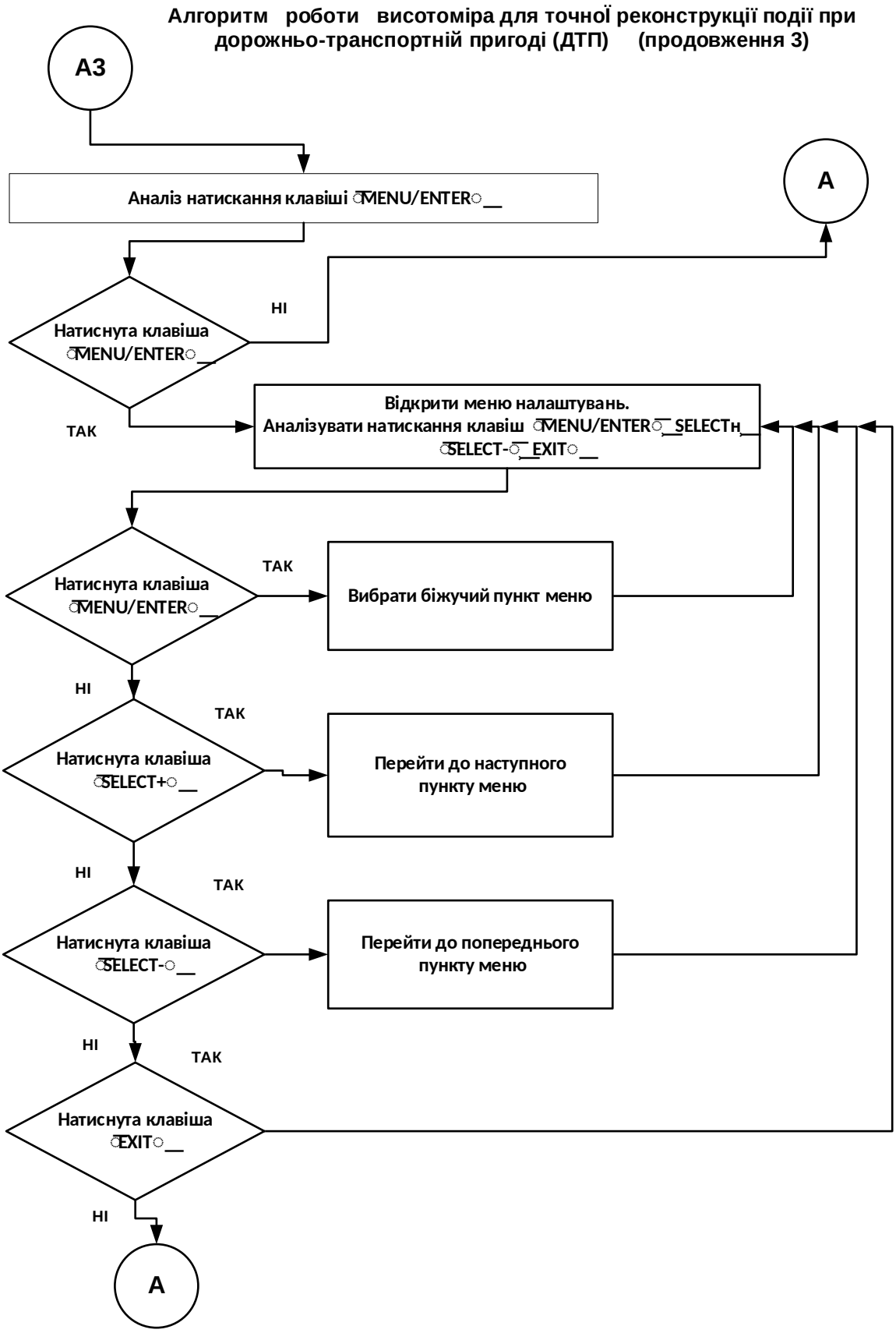


Рис. 4.3. Алгоритм роботи висотоміра (продовження 3)

Алгоритм роботи висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП) (продовження 4)

Пункти меню, та алгоритм їх роботи

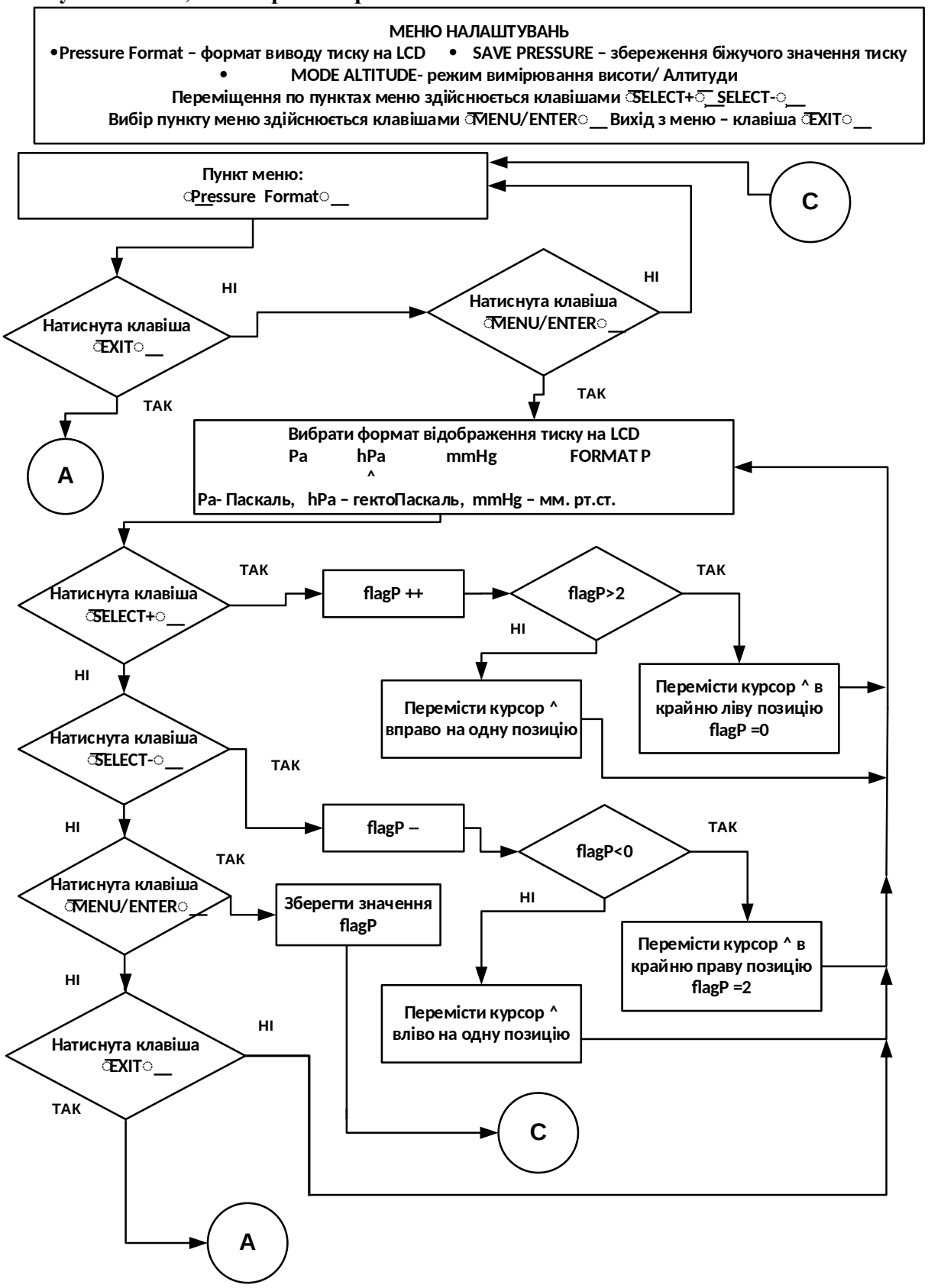


Рис. 4.4. Алгоритм роботи висотоміра (продовження 4)

Алгоритм роботи висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП) (продовження 5)

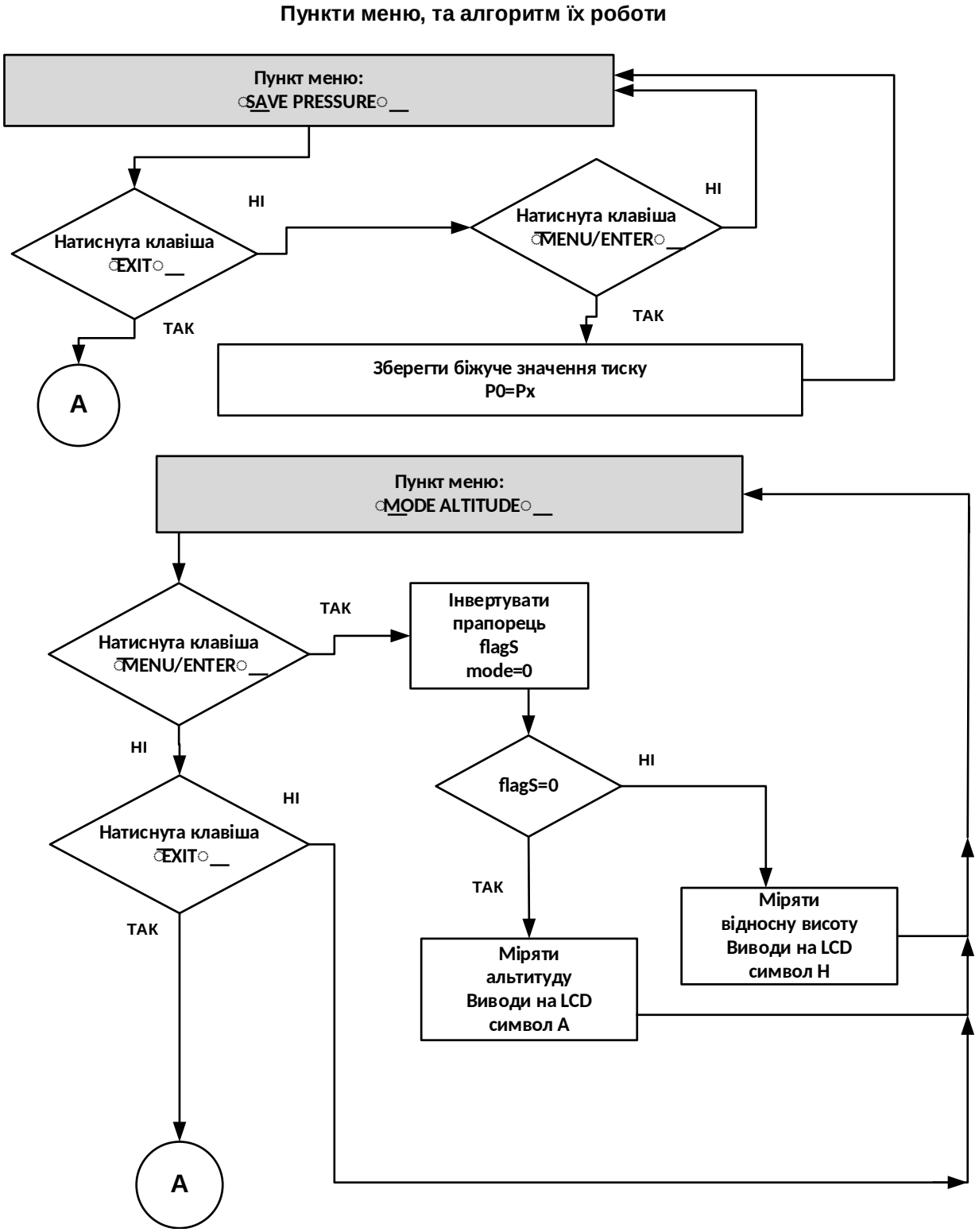


Рис. 4.5. Алгоритм роботи висотоміра (продовження 5)

Робота мікроконтролера стартує автоматично відразу після подачі живлення та запуску вбудованої прошивки. На старті система ініціалізує

внутрішні змінні, після чого конфігурує периферійні модулі та апаратну частину. Зокрема, виконується налаштування порту I2C, підготовка до роботи датчика BMP180, символьного РК-екрана, COM-порту, а також ліній вводу-виводу, до яких підключено світлодіоди та кнопки керування. Процедура завантаження передбачає двосекундне відображення на дисплеї вітального напису “DIGITAL ALTIMETER”. У цей же час система тестує працездатність комунікаційної шини та опитує сенсор BMP180. Якщо під час діагностики фіксується помилка, користувач бачить повідомлення “BMP180 error”.

У разі успішного завершення тестів мікроконтролер переходить до виконання головного циклу. Цей цикл забезпечує постійне зняття метеорологічних показників, їх програмну обробку та відстеження натискання кнопки “MENU/ENTER” для переходу до пунктів меню користувача.

Взаємодія з вимірювальним сенсором побудована на виклику спеціалізованих підпрограм. Функція bmp180Calibration() відповідає за первинне зчитування заводських коефіцієнтів калібрування. Після цього за допомогою BMP180ReadUT() зчитуються “сирі” температурні дані, необхідні для подальшої термокомпенсації результатів вимірювання тиску. Розрахунок реального атмосферного тиску (в Паскалях) виконує функція bmp180GetPressure(up). Спосіб подачі інформації на екран залежить від поточних значень конфігураційних прапорців:

- flagP (одиниці тиску): визначає, у чому показувати результат — Паскалях (при 0), гектопаскалях (при 1) чи мм рт. ст. (при 2).
- mode (режим роботи): перемикає логіку пристрою між калькуляцією абсолютної висоти (якщо дорівнює 0) та відносної висоти (якщо дорівнює 1).

Базовим станом при увімкненні є mode = 0. Цей режим візуально підтверджується активацією жовтого світлодіода, а алгоритм подальших обчислень у ньому задається станом параметра flagS. Якщо параметр flagS дорівнює 0, мікроконтролер обчислює та виводить на РК-модуль абсолютну висоту (альтитуду) в метрах відносно рівня моря. Коли flagS набуває значення

1, пристрій переходить до калькуляції та відображення відносної зміни висоти, що візуально підтверджується активацією синього світлодіода.

Для коректного розрахунку цієї величини система потребує попередньої фіксації опорного атмосферного тиску. Цей процес реалізується через пункт меню “SAVE PRESSURE” (схему процесу наведено на Рис. 4.2). Під час штатної роботи висотомір безперервно зчитує показники датчиків тиску та температури, показує інформацію на екрані й моніторить стан кнопки “MENU/ENTER”. Натискання на “MENU/ENTER” перемикає альтиметр у режим налаштувань. У цьому стані програма починає обробляти сигнали від чотирьох клавіш керування: “MENU/ENTER”, “SELECT+”, “SELECT-” та “EXIT”. Порядок та логіка взаємодії з кнопками деталізовані у вигляді блок-схем на Рис. 4.3 та 4.4. Архітектура меню користувача складається з трьох ключових пунктів:

- “PressureFormat” – налаштування відображення одиниць виміру тиску;
- “SAVE PRESSURE” – збереження поточного значення тиску в енергонезалежну пам’ять;
- “MODE ALTITUDE” – вибір режиму обчислення висотних координат.

Переміщення між розділами здійснюється за допомогою кнопок “SELECT+” (вперед) та “SELECT-“ (назад). Вибір конкретного параметра підтверджується клавішею “MENU/ENTER”, а вихід у головний робочий режим відбувається за натисканням кнопки “EXIT”.

Для прикладу розглянемо алгоритм налаштувань у меню “Pressure Format” (схему наведено на Рис. 4.4). Вибір необхідної одиниці вимірювання тиску (Pa, hPa чи mmHg) вибирається за допомогою клавіш “SELECT+” та “SELECT-“. Під час їх натискання горизонтальний маркер покажчика «^» переміщується у відповідний бік. Щоб підтвердити внесені зміни та зберегти новий стан прапорця flagP, слід натиснути “MENU/ENTER”, тоді як кнопка “EXIT” забезпечує повернення на попередній рівень інтерфейсу.

Підпрограма “SAVE PRESSURE” (Рис. 4.5) слугує для фіксації початкового (опорного) барометричного тиску. Цей етап є обов’язковим для

правильної роботи висотоміра (калібрування початкової точки відліку). У вікні “MODE ALTITUDE” (Рис. 4.5) за командою “MENU/ENTER” змінна mode набуває значення 0, а поточний стан прапорця flagS інвертується. Наступна логіка обчислень та екранна графіка повністю залежать від цього маркера: якщо flagS=0, система вираховує абсолютну висоту (на екрані виводиться індекс “A”); якщо flagS=1, обчислюється відносна висота (на дисплеї відображається індекс “H”).

4.2. Розробка програмних модулів для роботи з МЕМС сенсором BMP180

Обмін даними з барометричним сенсором BMP180 організовано через комплекс розроблених спеціалізованих функцій, які виконують такі завдання:

- імпорт заводських коефіцієнтів калібрування з пам’яті мікросхеми;
- зчитування первинних (некомпенсованих) значень температури та тиску;
- математичний розрахунок дійсного атмосферного тиску й поточної висоти над рівнем моря;
- низькорівневе керування шиною обміну (побайтовий запис та читання регістрів) через апаратний інтерфейс сенсора.

```
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 над рівнем океану

```
{
    float  altitude;   // змінна альтитуда
    float  temp;
 temp= (float)(pressure*0.01);   // обчислення
 temp= temp/1013.25;             // обчислення
 temp = 1-pow(temp, 0.19029);   // обчислення
 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 для меню налаштувань

Для інтерфейсу користувача розроблено наступні функції:

Функція для читання статусу кнопок: unsigned char getBtnStatus(unsigned char BUTTON_ID).

Функція читати попередній стан кнопки: unsigned char getPrevBtnStatus(unsigned char BUTTON_ID).

Функція форматування даних для виводу тиску у вибраних одиницях вимірювання: void setPressUnitFormat(void).

Функція керування меню налаштувань: void mainMenu(void).

4.4. Розробка програмного модуля для роботи з символьним дисплеєм

Для виводу даних на РКД створено функції які вмонтовані в бібліотеку “mylcd.c”:

```
void lcd_clear (void); // очистити інформацію РКД
void lcd_strobe (void); // вивести строб в РКД
void lcd_putchar (unsigned char c); // вивести один символ в РКД
void lcd_init (void); // проініціалізувати РКД
void lcd_puts(unsigned char *str); // вивести символьне значення  (string)  на РКД
void lcd_write (unsigned char c); // запис даних в LCD
void lcd_putsf (unsigned char __flash *str); // вивід символу з FLASH LCD
void cursor_on (void); // вивести значок курсора на LCD
void lcd_cmd (unsigned char c); // записати  команду в LCD
void cursor_off (void); // не виводити значок курсора на LCD
```

4.5. Програмна реалізація головного алгоритму роботи цифрового висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП)

Програмний код, який забезпечує виконання базового алгоритму роботи цифрового альтиметра, наведено у додатку 2. Після компіляції проєкту в інструментальному середовищі CodeVisionAVR було згенеровано файл прошивки з розширенням .hex, розрахований на завантаження в мікроконтролер ATmega128. Цей бінарний файл записується в енергонезалежну пам’ять МК за допомогою апаратного програматора.

4.6. Моделювання та аналіз результатів роботи цифрового висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП) в Proteus ISIS

Попереднє тестування висотоміра та верифікація працездатності алгоритму керування виконувалися шляхом симуляції у програмному пакеті Proteus ISIS. Результати комп’ютерного моделювання роботи системи наведено на графічних матеріалах нижче.

Під час подачі живлення на цифровий пристрій першочергово запускаються процеси конфігурування рідкокристалічного екрана, послідовного COM-порту, шини зв'язку I2C, системних змінних та портів опитування клавіатури. Після цього на дисплей та в послідовний порт передаються стартові інформаційні повідомлення (Рис. 4.6).

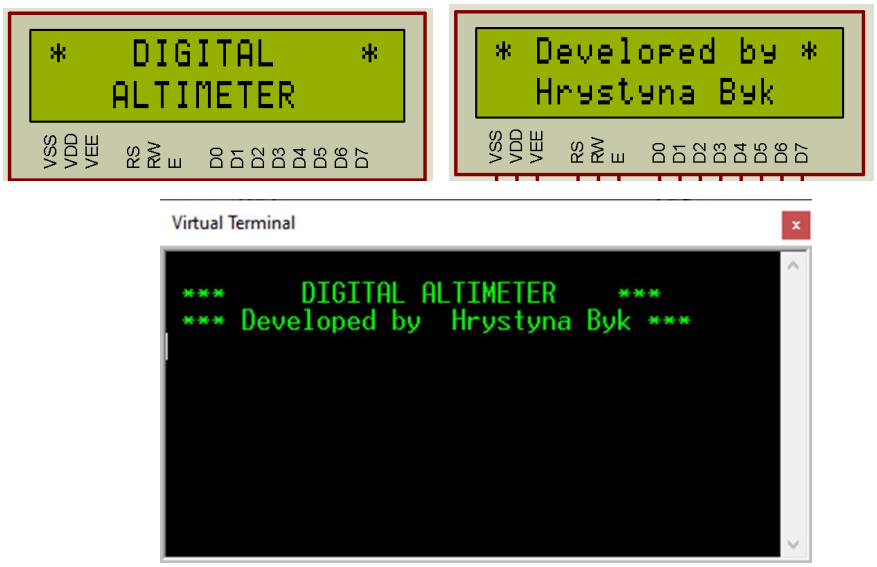


Рис. 4.6. Моделювання роботи висотоміра в Proteus ISIS – ініціалізація системи

Далі, відбувається зчитування через інтерфейс I^2C даних з сенсора BMP180, їх обробка та вивід інформації в COM-порт і на РКД (Рис.4.7-4.8).

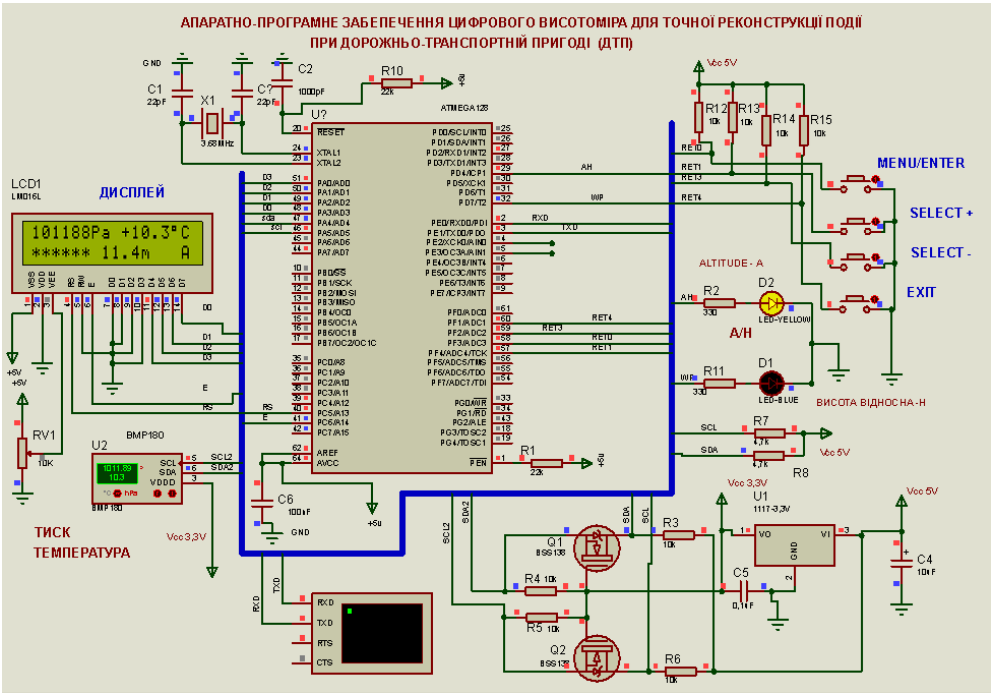


Рис. 4.7. Моделювання роботи висотоміра в Proteus ISIS. Робочий режим

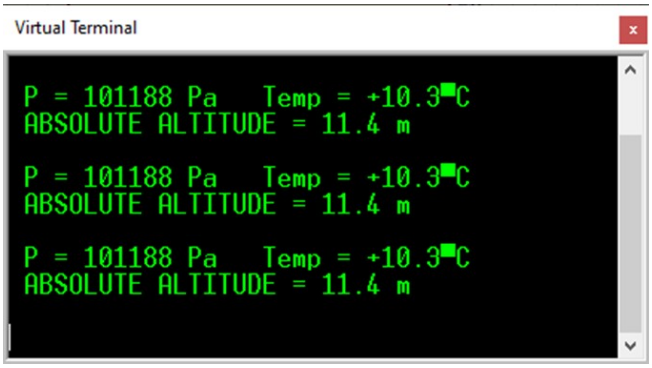


Рис. 4.8. Моделювання роботи висотоміра в Proteus ISIS.

Вивід інформації по COM-порту

На рідкокристалічний індикатор виводяться поточні барометричні показники та температура навколишнього середовища (у верхньому рядку), а також опорний рівень тиску й поточний режим роботи (символ «H» позначає калькуляцію відносної висоти в метрах, дані відображаються у нижньому рядку). Початкова конфігурація альтиметра передбачає виведення тиску в Паскалях (Па), а температурних значень – у градусах Цельсія (°C). За базових умов система запускається у режимі вимірювання абсолютної висоти в метрах. Індикація цього стану забезпечується безперервним світінням жовтого світлодіода. Пристрій функціонує у штатному режимі. Для переходу до “Головного меню” (Рис. 4.9) використовується клавіша “MENU/ENTER”.

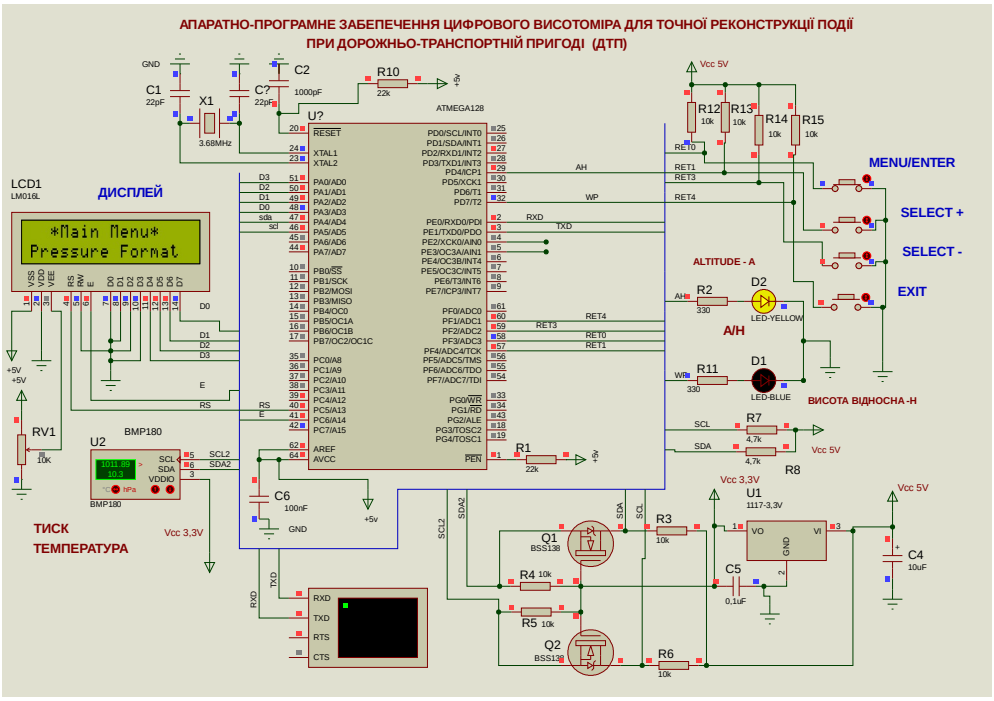


Рис.4.9. Моделювання роботи цифрового висотоміра. Головне меню

Головне меню містить пункти, що представлені на Рис.4.10 - 4.12. Перехід між пунктами виконується кнопками “SELECT–“ та “SELECT+”, а вибір пункту – натисканням “MENU/ENTER”. Повернутися до “робочого режиму” основного режиму роботи можна за допомогою клавіші “EXIT”.

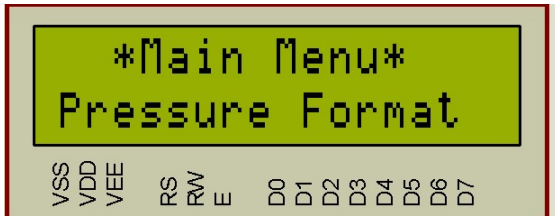


Рис. 4.10. Пункт меню налаштування формату тиску

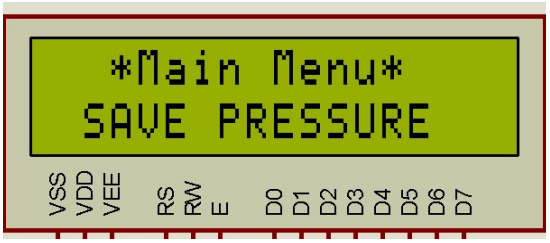


Рис. 4.11. Пункт меню для збереження у пам’яті біжучого значення тиску

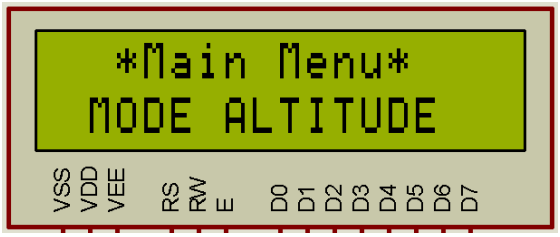


Рис. 4.12. Пункт меню. Вивід альтитуди або відносної висоти

Схема вибору одиниць виміру атмосферного тиску для його подальшого виведення на рідкокристалічний екран проілюстрована на Рис. 4.13. Оператор має можливість задати необхідний формат відображення – Паскалі (Pa), гектопаскалі (hPa) або міліметри ртутного стовпа (mmHg). Зміна налаштувань виконується шляхом пересування горизонтального маркера “^” за допомогою клавіш “SELECT–“ та “SELECT+”. Нове значення записується в пам’ять системи після натискання “MENU/ENTER”, а для повернення з цього вікна на попередній рівень інтерфейсу слугує кнопка “EXIT”.

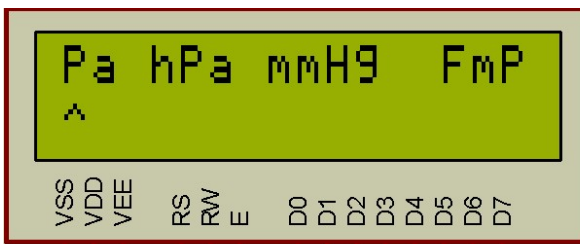


Рис. 4.13. Вибір одиниць вимірювання тиску

На Рис.4.14 та 4.16 виведено на РКД тиск у Паскалях та гектоПаскалях та мм.рт. ст.., відповідно.

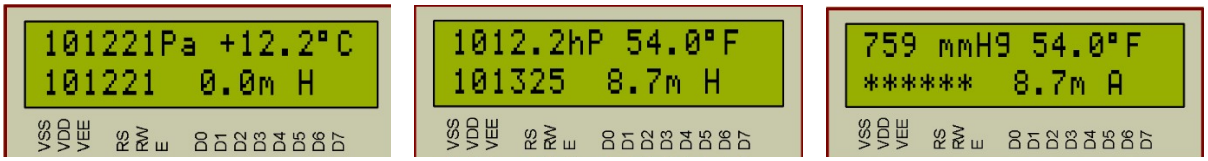


Рис. 4.14. Одиниці тиску у Паскалях, гектоПаскалях, мм.рт.ст

Архітектура цифрового висотоміра базується на двох ключових режимах: визначенні абсолютної (альтитуди) та відносної висоти.

Налаштування цих параметрів здійснюється через діалогове вікно “MODE ALTITUDE”. За допомогою кнопки “MENU/ENTER” оператор перемикає алгоритм розрахунку між абсолютною (“A”) та відносною (“H”) величинами, а вихід до попереднього пункту меню забезпечується натисканням “EXIT”. У разі розрахунку абсолютної висоти (процес проілюстровано на Рис. 4.7) РК-дисплей відображає поточну координату в метрах щодо рівня моря, де за базову величину прийнято стандартний тиск 101325 Па. Цей режим маркується літерою “A”. Для обчислення відносної висоти система потребує фіксації початкової точки відліку. Цей процес реалізується у розділі “SAVE PRESSURE”, де поточні барометричні дані записуються в пам’ять після натискання “MENU/ENTER” (Рис. 4.11).

На Рис. 4.17-4.18 наведено результати комп’ютерного моделювання роботи приладу під час вимірювання відносної висоти. Нижній рядок рідкокристалічного індикатора виводить зафіксований на початковій точці тиск, розраховану відносну висоту (в метрах) та буквений індекс “H”. У верхньому рядку екрана виводяться поточні параметри тиску в Паскалях та температура навколишнього середовища в градусах Цельсія. В цьому режимі світиться синій світлодіод “вимірювання відносної висоти”.

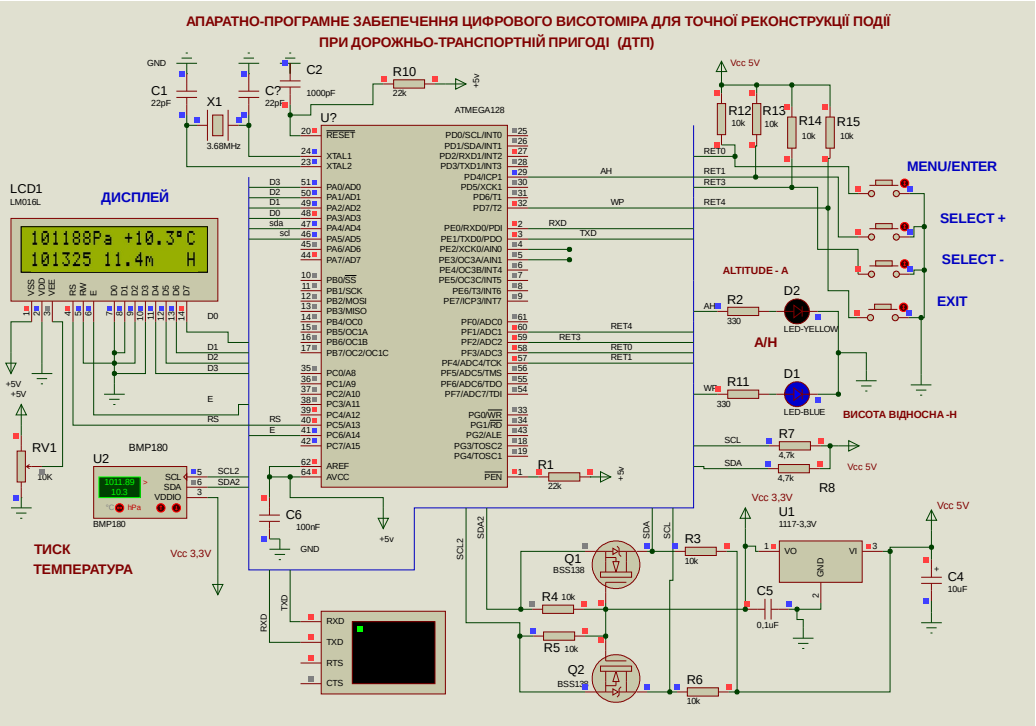


Рис. 4.17. Моделювання роботи висотоміра. Вимірювання відносної висоти

Рис. 4.18. Вимірювання відносної висоти. Вивід даних в COM -порт.

Моделювання роботи висотоміра для точноЇ реконструкції події при дорожньо-транспортній пригоді (ДТП) в різних режимах підтвердило функціонування апаратного та програмного забезпечення згідно розроблених алгоритмів.

ВИСНОВКИ

В дипломній роботі було спроектовано апаратну частину та програмне забезпечення висотоміра для точної реконструкції події при дорожньо-транспортній пригоді (ДТП).

Створений пристрій забезпечує точне визначення параметрів атмосферного тиску, температури довкілля, абсолютної альтитуди (висоти над рівнем моря) та відносної висоти. Також реалізовано функцію передачі виміряних даних на ПК через послідовний порт зв’язку. Візуальний контроль параметрів здійснюється на символьному рідкокристалічному індикаторі. Вбудоване меню надає користувачеві можливість оперативно перемикати режими роботи, зберігати опорні значення барометричного тиску та обирати одиниці відображення метеоданих між Па, гПа та мм рт. ст. Для відображення робочого стану системи є світлодіодна індикація.

Апаратна архітектура цифрового висотоміра базується на сучасних електронних компонентах, серед яких ключовими є мікроконтролер сімейства AVR, цифровий сенсор BMP180 та символьний РК-дисплей. Процес проектування охоплював побудову принципової електричної схеми та розробку її інтерактивного цифрової моделі в пакеті Proteus VSM. Вбудоване програмне забезпечення написане мовою С у середовищі CodeVisionAVR. До його складу входять розроблені драйвери для низькорівневого обміну з барометром BMP180 та символьним дисплеєм.

Комплексна симуляція та відлагодження системи в емуляторі Proteus ISIS повністю підтвердили функціональну придатність та точність роботи розроблених алгоритмів для використання у висотомірі для точної реконструкції події при дорожньо-транспортній пригоді.

Виконання роботи сприяло суттєвому розширенню практичного досвіду в галузі програмування мікроконтролерних систем та проектування сучасних цифрових пристроїв.

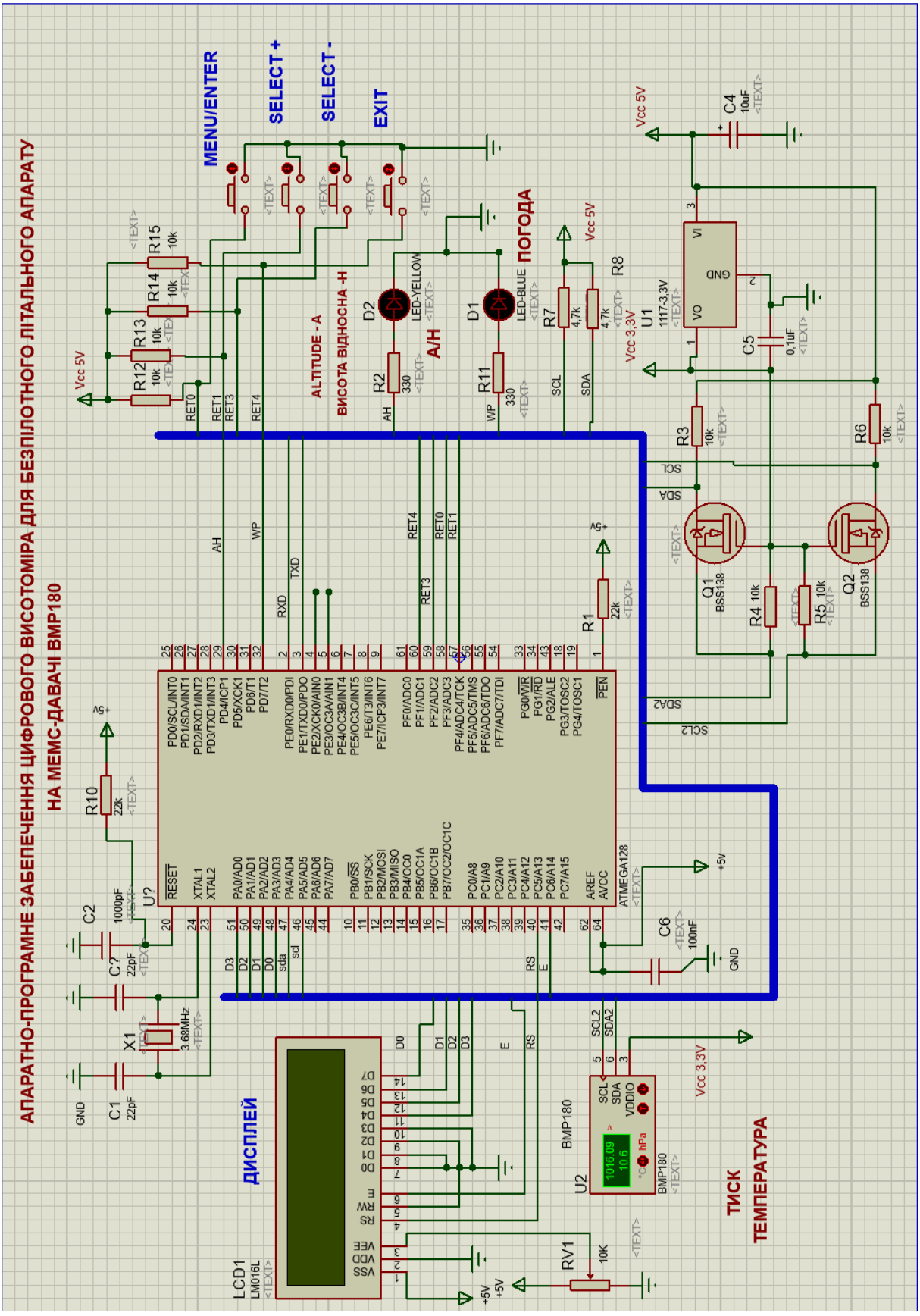
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

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.alldata-sheet.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/.
10. Dhananjay V. Gadre. Programming and Customizing the AVR Microcontroller. – Published by McGraw-Hill Education TAB, 2000. - p. 336.
11. Muhammad Ali Mazidi, Sarmad Naimi, Sepher Naimi. The AVR Microcontroller and Embedded Systems: Using Assembly and C – Published by Pearson, 1st ed., 2015. – p. 752.
12. Richard H. Barnett, Sarah A. Cox, and Larry O'Cull. Embedded C Programming and the Atmel AVR. – Published by Cengage Learning, 2nd ed., 2006. – p. 560.

ДОДАТКИ

Додаток 1

Принципова схема цифрового висотоміра



Додаток 2

Головний програмний модуль цифрового висотоміра

```
/*****************************************************
Project :Digital altimeter for precise road accident reconstruction
Author  : BYK HRYSTYNA
Chip type           : ATmega128
Clock frequency     : 3,680000 MHz
*****************************************************/
#include <mega128.h>
#include <stdio.h>
#include <math.h>
#include <delay.h>
#include <i2c.h>          //
#include "mylcd.c"     //  бібліотека для LCD
// USART0  8 Data, 1 Stop, No Parity  Asynchronous 9600
#define RXB8 1   // Bit 1 – RXB8n: Receive Data Bit 8 - UCSR0B Status Register B
#define TXB8 0   // Bit 0 – TXB8n: Transmit Data Bit 8  UCSR0B Status Register B
#define UPE 2    // Bit 2 – UPEn: Parity Error UCSR0C Status Register A
#define OVR 3
#define FE 4     //Bit 4 – FEn: Frame Error - UCSR0C Status Register A
#define UDRE 5    // Bit 5 – UDREn: USART Data Register Empty-UCSR0C Status Register A
#define RXC 7    // Bit 7 – RXCn: USART Receive Complete - UCSR0C Status Register A
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
  // Функція для передачі рядка
void uart0_puts(char *s)
 {
   while (*s)   {
      putchar(*s++);    }
}
// I2C Bus functions
#asm
  .equ __i2c_port=0x1B ;PORTA
  .equ __sda_bit=4   .equ __scl_bit=5
#endasm

#define BMP180W 0xEE   // write-адрес BMP180    на шині i2c
#define BMP180R 0xEF   // read- адрес BMP180    на шині i2c
#define B1    0xB6   // -адрес коеф. b1
#define B2    0xB8   // -адрес коеф. b2
#define AC1   0xAA   // -адрес коеф. ac1
#define AC2   0xAC   // -адрес коеф. ac2
#define AC3   0xAE   // -адрес коеф. ac3
#define MC    0xBC   // -адрес коеф. mc
#define MD    0xBE   // -адрес коеф. md
#define MB    0xBA   // -адрес коеф. mb
#define AC6   0xB4   // -адрес коеф. ac6
#define AC5   0xB2   // -адрес коеф. ac5
```

```
#define AC4   0xB0   // -адрес коеф. ac4
#define BUTTON_DDR      DDRF
#define BUTTON_PORT     PORTF    // клавіші на порті F
#define BUTTON_PIN      PINF
#define SELECT_PLUS_BTN_PIN 4    // КН2
#define MENU_ENTER_BTN_PIN 3     // КН1
#define EXIT_BTN_PIN 1           // КН4
#define SELECT_MINUS_BTN_PIN 2   // КН3
#define TEMP_UNITS_NUM 3   // кількість пунктів підменю

const unsigned char OVS_S = 2; //(0,1,2,3 from ultra low power, to  hi-resolution)
// Declare  global variables
unsigned char PREV_BUTTON_PIN = 0xFF;    //  початкове значення клавіші
char *ttt[3] = {"FORMAT T", "FmP", " " };
volatile int flags=0,flagt=0, flagp=0, mode=0;    //  mode=0 - " MODE ALTITUDE "

int press_unit_ind = 0;    // для тиску
char *press_unit[3] = {"Pa", "hPa", "mmHg"};
volatile int flagwx=2;
volatile long int  pmm=101325,  p2=101325 ;    //paw=0,   pmm=101325
float ph=20.1, tk=0.1;      //  hPa
float  a0=0.1, a1=0.1;      // висоти над рівнем океану
float     dh=0.1;           // зміна висоти  відносна
int deltap=0;    // відносна зміна тиску
short temperatura=1;

char  buffer[33];         // для LCD конвертації
char id = 0xcc;           // BMP180 id =   55h

/* Calibration coefficients */
unsigned int ac5=32757, ac4=32741,  ac6=23153;
int b1=6190,b2=4,ac1=408,ac2=-72,ac3=-14383,mc=-8711,mb=-32768,md=2868 ;
volatile unsigned int ut;   // uncompensated tempera value - 0.1 -коеф.
volatile long int b5 ;
long p,upr;    // тиск
short t;    // температура
float  tf=-1.1;       //
//  статус клавіші
  unsigned char getBtnStatus(unsigned char BUTTON_ID)  // читати атрибут клавіші
{return (!(BUTTON_PIN & (1 << BUTTON_ID)));}

unsigned char getPrevBtnStatus(unsigned char BUTTON_ID)  // читати попередній атрибут
клавіші
{return (!(PREV_BUTTON_PIN & (1<< BUTTON_ID)));}

 void setPressUnitFormat(void)           //  формат виводу
    {
      int i, pos_x = 0;      // pos_x_max = TEMP_UNITS_NUM * 2;
      char oK = 0;
      lcd_clear();          //   очистити дисплей
      lcd_gotoxy(13,0);      //   перейти за координатами
      lcd_puts(ttt[1]);      //  "FmP"
```

```
    lcd_gotoxy(0,0);       //   перейти за координатами
    for (i = 0; i < TEMP_UNITS_NUM; i++)     // цикл
     {      lcd_puts(press_unit[i]);  lcd_putchar(' ');          // вивід на LCD
     }
    while(!oK)     {
       PREV_BUTTON_PIN = BUTTON_PIN;
       lcd_gotoxy(pos_x, 1);          //   перейти за координатами
       lcd_putchar('^');              // вивід на LCD
       if (getBtnStatus(SELECT_PLUS_BTN_PIN))       // клавіша  КН2=  "+"
        {
            if (!getPrevBtnStatus(SELECT_PLUS_BTN_PIN))      {
               lcd_gotoxy(pos_x, 1);          // перейти за координатами
            lcd_putchar(' ');          // вивід на LCD
                  if (pos_x < (TEMP_UNITS_NUM - 1)* 4)
                    pos_x += 4;
                  else
                    pos_x = 0;
          }            }

       if (getBtnStatus(SELECT_MINUS_BTN_PIN))       //  клавіша  КН3=  "-"
        {
            if (!getPrevBtnStatus(SELECT_MINUS_BTN_PIN))       {
            lcd_gotoxy(pos_x, 1);          // клавіша рядок 2
            lcd_putchar(' ');              // вивід на LCD
               if (pos_x ==0)   pos_x = (TEMP_UNITS_NUM-1) * 4;
            else    pos_x -= 4;
          }         }
       if (getBtnStatus(MENU_ENTER_BTN_PIN))   // аналіз клавіші ENTER
        {
         if (!getPrevBtnStatus(MENU_ENTER_BTN_PIN))      {
            press_unit_ind = pos_x / 4;
              return;          }        }
       if (getBtnStatus(EXIT_BTN_PIN))          // аналіз клавіші EXIT
        {
         if(!getPrevBtnStatus(EXIT_BTN_PIN))          {
            PREV_BUTTON_PIN = BUTTON_PIN;    // попереднє значення
            return;          }          }
       flagp=  pos_x/4;         //    формат  тиску  - прапорець
    }   }

  float  bmp180Altitude(long int  pressure)    //  вимірювання висоти
   {
float  altitude;   // абсолютна висота
    float  temp;
    temp= (float)(pressure*0.01);     // обчислення
    temp= temp/1013.25;              // обчислення
    temp = 1-pow(temp, 0.19029);     // обчислення
    altitude = 44330*temp;        // altitude - метри
    return altitude;

   }
```

```
 void mainMenu(void)          // головне меню
{
 char *menu_items[4] = {"Pressure Format ", " SAVE PRESSURE  ", " MODE ALTITUDE
"};
 unsigned char selected = 0; char menu_title[] = " *Main Menu*"; // Units Conversion
 while(1)  {
  PREV_BUTTON_PIN = BUTTON_PIN;    // аналіз клавіші
  lcd_gotoxy(1,0);                 // перейти за координатами
  lcd_puts(menu_title);
  lcd_gotoxy(0,1);                 // перейти за координатами
  lcd_puts(menu_items[selected]);

  if(getBtnStatus(SELECT_PLUS_BTN_PIN))                // клавіша  КН2=  "+"
    {
     if(!getPrevBtnStatus(SELECT_PLUS_BTN_PIN))
       if(selected == 2) selected = 0;             //  пунктів меню  селектор
       else selected++;                          // інкримент
    }

  if(getBtnStatus(SELECT_MINUS_BTN_PIN))        // клавіша КН3=  "-"
  {
   if (!getPrevBtnStatus(SELECT_MINUS_BTN_PIN))
     if (selected == 0) selected = 2;           // вибір пунктів меню 0,1,2,3,4
     else selected--;                          // декримент
  }

  if(getBtnStatus(MENU_ENTER_BTN_PIN))          // клавіша КН1="menu"
  if(!getPrevBtnStatus(MENU_ENTER_BTN_PIN))     // аналіз
    {
          switch(selected) {
       case 0: setPressUnitFormat();   break;
             case 1:
             lcd_gotoxy(15,1);
             flags=1;lcd_putsf("Y"); p2=p;   PORTD.4=0;PORTD.7=1;  // вивести на LCD і
зберегти значення тиску
             delay_ms(500);
             break;
             case 2:
                  mode=0; //  PORTD.4=1;PORTD.7=0;    lcd_gotoxy(15,1);   //----------------
                 if (flags==0) {flags=1;lcd_putsf("H");PORTD.4=0;PORTD.7=1;  } // led blue
ON
                  else {flags=0;lcd_putsf("A");PORTD.4=1;PORTD.7=0; };    //    led yellow ON
                delay_ms(500);       // затримка
                break;              //  висота
             default: break;
           }
     lcd_clear();               // очистити дисплей
    }
  if (getBtnStatus(EXIT_BTN_PIN))         // клавіша КН4="exit"
    {
     if (!getPrevBtnStatus(EXIT_BTN_PIN))   //   аналіз клавіші
       {
```

```
        PREV_BUTTON_PIN = BUTTON_PIN;       //
        return;         }     } }     }

/* читати байт BMP180 */
 char BMP180ReadByte(unsigned char address) {
    unsigned char data;  //  дані з  BMP180
    i2c_start();         // старт  i2c
    i2c_write(BMP180W);  // адрес BMP180 write на  на шині i2c
    i2c_write(address);  // запис
    i2c_start();
    i2c_write(BMP180R ); // адрес BMP180 read на  на шині i2c
    data=i2c_read(0);    // дані
    i2c_stop();
    return(data);
  }

/* записати байт в BMP180 */
void BMP180WriteByte(unsigned char address, unsigned char data){
   i2c_start();         // старт  i2c
   i2c_write(BMP180W);  // адрес BMP180 write на  на шині i2c
   i2c_write(address);  // запис
   i2c_write(data);     // дані для запису
   i2c_stop();
}

/* читати Calibration coefficients */
void bmp180Calibration() {
   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);
 }
```

```
 /* читати uncompensated temperature value */
int BMP180ReadUT(){
   unsigned char lsb,  msb;

  BMP180WriteByte(0xF4, 0x2E);  // записати 0x2E в Register 0xF4
  delay_ms(15); // затримка 4.5ms
  msb = BMP180ReadByte(0xF6); lsb = BMP180ReadByte(0xF7);
  ut= ((int)msb<<8)+((int)lsb);    // дані
  return(ut);
}

// обчислення temperature given ut.
// значення в units of 0.1 deg C
short BMP180GetTemperature(int ut){
  long x2,x1 ;
  x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15; // дані
  x2 = ((long)mc << 11)/(x1 + md);              // дані
  b5 = x1 + x2;
  return ((b5 + 8)>>4);
}

/* читати uncompensated pressure значення */
unsigned long int bmp180ReadUP() {
  unsigned long int up; unsigned char lsb, msb,  xlsb;
BMP180WriteByte(0xF4, (0x34 + (OVS_S<<6)) );
  // чекати на conversion,
 switch (OVS_S)  {
     case 0: delay_ms(5);  break; // Ultra low power  - час  4,5 ms
     case 1: delay_ms(8);  break; // Standard - час  7,5 ms
     case 2: delay_ms(14); break; // HIGH - час  13,5 ms
     case 3: delay_ms(26); break; // ULTRA_HIGH  - час  22,5 ms
  }

   // читати register 0xF6 (MSB),0xF8 (XLSB), 0xF7 (LSB)
   msb  = BMP180ReadByte(0xF6);     //  дані
   lsb  = BMP180ReadByte(0xF7);     //  дані
   xlsb = BMP180ReadByte(0xF8);     //  дані
   up = ((((long)msb <<16) | ((long)lsb <<8) | ((long)xlsb)) >> (8-OVS_S)); // pressure
uncompensated
   return(up);
}

// Calculate pressure given up
long bmp180GetPressure(unsigned long up) // значення returned will be pressure в Pa.
{
 unsigned long b4, b7; long x1,x3, x2,b6, b3,  p;
  b6 = b5 - 4000;
  // обчислення B3
  x1 = (b2 * (b6 * b6)>>12)>>11;  //  дані
  x2 = (ac2 * b6)>>11;            //  дані
  x3 = x1 + x2;                   //  дані
  b3 = (((((long)ac1)*4 + x3)<<OVS_S) + 2)>>2;  //  дані
```

```
  // обчислення B4
  x1 = (ac3 * b6)>>13;              //  дані
  x2 = (b1 * ((b6 * b6)>>12))>>16;   //  дані
  x3 = ((x1 + x2) + 2)>>2;          //  дані
  b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;  //  дані

  b7 = ((unsigned long)(up - b3) * (50000>>OVS_S)); //  дані
  if (b7 < 0x80000000)           //  аналіз
    p = (b7<<1)/b4;
  else
    p = (b7/b4)<<1;  x1 = (p>>8) * (p>>8);       //  дані
  x1 = (x1 * 3038)>>16;       //  дані
  x2 = (-7357 * p)>>16;      //  дані
  p += (x1 + x2 + 3791)>>4;   //  дані
  return p;
}

void main(void){
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
PORTA=0x0F;DDRA=0x0F; DDRC=0xF0; PORTC=0xF0;
PORTG=0x00;  DDRG=0x00;  // Port G
PORTB=0x00;   DDRB=0x00; // Port B
PORTE=0x00;  DDRE=0x00; // Port E
PORTF=0x00;  DDRF=0x00;  // Port F
PORTD=0x00;  DDRD=0x00; // Port D
// Timer 0 Stopped
// Normal top=FFh
// OC0 : Disconnected
  TCCR0=0x00;   ASSR=0x00;   OCR0=0x00;  TCNT0=0x00;
// Timer/Counter 1 initialization
// 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;
// Timer/Counter 2
//  Timer 2 Stopped
  TCNT2=0x00; OCR2=0x00; TCCR2=0x00;
// Timer/Counter 3
// Timer 3 Stopped
 TCCR3B=0x00; TCNT3H=0x00;   TCNT3L=0x00; TCCR3A=0x00;
 ICR3L=0x00;  OCR3AH=0x00;   OCR3AL=0x00; ICR3H=0x00;
 OCR3BL=0x00; OCR3CH=0x00;   OCR3CL=0x00;  OCR3BH=0x00;
// External Interrupt(s)
  EICRB=0x00;  EIMSK=0x00;EICRA=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;    ETIMSK=0x00;
// Analog Comparator
// Analog Comparator: Off
SFIOR=0x00; ACSR=0x80;
```

```
DDRA = 0x0F;            // D0-D3 ->  for LCD
DDRC = 0x60;            // Port C RC5, RC6 -> for LCD
DDRD.2 = 1;     // SD5/-> включити живлення +5X LCD
PORTD.2=1;      // SD5/=1 включити живлення +5X LCD
PORTA.7=1;      // LED1=1  for LCD
DDRC.7=1;       // LWR/-> for LCD
PORTC.7=0;      // LWR/=0   for LCD
DDRA.7=1;    // LED1 ->  for LCD
PORTD.4=1;      //
DDRD.4 = 1;     //
PORTD.7=0;      //
DDRD.7 = 1;     //
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;     // Asynchronous
UCSR0B=0xD8;     // Status Register B дозвіл Rx Tx
UCSR0C=0x06;    //Status Register C  8 Data
// UBBR = F_OSC / (16 * BAUD) - 1 ( Asynchronous Normal Mode;
UBRR0H=0x00;    //Baud Rate Register UART0 - Rg швидкості передачі H-byte
UBRR0L=0x17;    // Rg швидкості передачі L-byte  = 23 (9600 bps при focs=3.68 MHz)
lcd_init ();    //   initialization  lcd
lcd_clear ();   lcd_gotoxy(0,0);
lcd_putsf("*   DIGITAL    *");
lcd_gotoxy(0,1);                // координати
lcd_putsf("   ALTIMETER   ");     //--------------
delay_ms(900);                  // затримка
lcd_clear ();   lcd_gotoxy(0,0);     // координати
lcd_putsf("* Developed by *");
lcd_gotoxy(0,1); lcd_putsf("  Hrystyna Byk  ");
uart0_puts("\r\n");
 uart0_puts(" ***     DIGITAL ALTIMETER    *** \r\n");    // вивід інформації  в послідовний
порт
uart0_puts(" *** Developed by  Hrystyna Byk *** \r\n");
 delay_ms(20);
i2c_init();   //  initialization  I2C Bus
bmp180Calibration();
/* ініціалізація клавіш */
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);
  // очистка LCD
  lcd_clear ();

while (1)
  {
     // Place your code
```

```
          uart0_puts("\r\n");   // rs232
        delay_ms(10);  // затримка
      if (getBtnStatus(MENU_ENTER_BTN_PIN))    // аналіз
        {
          if (!getPrevBtnStatus(MENU_ENTER_BTN_PIN)) // аналіз
           {              lcd_clear();     //  очистка LCD
            mainMenu();      //   виклик меню
            lcd_clear();;    //    очистка LCD
           }
          PREV_BUTTON_PIN = BUTTON_PIN;    };
   id= BMP180ReadByte(0xD0);  // читаємо = 55h
   if (id!=0x55)       // аналіз
    {
     while (1)    {
      lcd_clear ();    //    очистка LCD
      lcd_gotoxy(0,0);    // координати
      lcd_putsf("BMP180 error");        // помилка !!
      uart0_puts(" BMP180 error !!! \r\n");
      delay_ms(1000);   // затримка !
     };
    }    else  {
    ut=BMP180ReadUT();   //  UT = 27898-
    t=BMP180GetTemperature(ut);  //  температура
    upr=bmp180ReadUP();   //
    p=bmp180GetPressure(upr); //   тиск
   lcd_gotoxy(9,0);   // координати
   lcd_putsf("       ");
   lcd_gotoxy(9,0);     // координати
    delay_ms(30);    // затримка !
  }
   lcd_gotoxy(0,0);
    switch(flagp)        // тиск {"Pa", "hPa", "mmHg"};
{         case 0:
              lcd_gotoxy(0,0);     // координати
              lcd_putsf("        ");  lcd_gotoxy(0,0);       // координати
              sprintf(buffer, "%lu",p);  // атмосферний тиск - в Паскалях
              lcd_puts(buffer);   // вивід
              lcd_putsf("Pa");    // вивід
              delay_ms(30);      // затримка !
              uart0_puts(" P = ");  // вивід
              uart0_puts(buffer);
              uart0_puts(" Pa  ");
              break;
          case 1:
              lcd_gotoxy(0,0);      // координати
              lcd_putsf("        ");
              lcd_gotoxy(0,0);      // координати
              ph=(float)(p*0.01);         //  атмосферний тиск - в hPa
              sprintf(buffer, "%1.1f",ph);
              lcd_puts(buffer);     // вивід
              lcd_putsf("hP");      // вивід
              uart0_puts(" P = "); uart0_puts(buffer);  uart0_puts(" hP  ");
```

```
            break;   // hPa
           case 2:
            lcd_gotoxy(0,0);     // координати
            lcd_putsf("        "); lcd_gotoxy(0,0);    // координати
            pmm=(p*75/10000);    // атмосферний тиск -   в mmHg
            sprintf(buffer, "%lu",pmm);
            lcd_puts(buffer);       // вивід
            lcd_putsf(" mmHg");    // вивід
            uart0_puts(" P = ");  uart0_puts(buffer);  uart0_puts(" mmHg  ");
            break;
           default: break;
          }
    lcd_gotoxy(0,0);  // координати
    switch(flagt)       // температура в {'C'};
          {
    case 0:
          lcd_gotoxy(9,0);    // координати
          lcd_putsf("      ");
          lcd_gotoxy(9,0);    // координати
          tf=(float) (t*0.1);
          if (tf>=0)
          { sprintf(buffer, "+%1.1f%cC",tf,0xdf);}  // вивід float в буфер температура   df
           else {sprintf(buffer, "%1.1f%cC",tf,0xdf);};
          lcd_puts(buffer);    // вивід даних на LCD - температура
          uart0_puts(" Temp = "); uart0_puts(buffer); uart0_puts(" \r\n"); // вивід в Com port
температури
         break;
     case 1:
          tf=(float)(t*0.18)+32;        //   Цельсій - Фаренгейт
           break;
     case 2:
          tk=(float)(t*0.1+273.15);     //   Цельсій - Кельвін
           default: break;
          }
  // end format
        // запис початкового  тиску для обчислення  відносної висоти
     if (flags==0)
        { lcd_gotoxy(0,1);        // координати
         lcd_putsf("******");
         if (mode==0)           // аналіз
           {
            a0= bmp180Altitude(p);    // висота над рівнем моря
            lcd_gotoxy(7,1);          // координати
            lcd_putsf("       ");
            lcd_gotoxy(7,1);
            sprintf(buffer, "%1.1f",a0);    //  дані
            lcd_puts(buffer);               //  дані
            lcd_putsf("m");
             lcd_gotoxy(14,1);        // координати
            lcd_putsf(" A");
            uart0_puts(" ABSOLUTE ALTITUDE = ");
            uart0_puts(buffer);       //  дані
```

```
            uart0_puts(" m  "); uart0_puts(" \r\n");
          };
       }
      else {
          lcd_gotoxy(0,1);    // координати
           if (mode==0)       // аналіз
             {
              //  запис тиску у Паскалях   p2=p;
              sprintf(buffer, "%lu",p2);    // paw
              lcd_puts(buffer);            //  дані
              a0= bmp180Altitude(p);        //  дані
              a1= bmp180Altitude(p2);       //  дані
              dh= a0-a1;                   //  дані
              lcd_gotoxy(7,1);           // координати
                lcd_putsf("       ");
              lcd_gotoxy(7,1);           // координати
              sprintf(buffer, "%1.1f",dh);   //  дані
              lcd_puts(buffer);            //  дані
              lcd_putsf("m");
              lcd_gotoxy(14,1);      // координати
              lcd_putsf(" H");
             uart0_puts(" RELATIVE HEIGHT = ");
             uart0_puts(buffer);        //  дані
             uart0_puts(" m  "); uart0_puts(" \r\n");     };   };
      if (mode==1)      //
      {
        deltap= p-p2; //  тиск - фіксований
        lcd_gotoxy(9,1);  lcd_putsf("      ");  lcd_gotoxy(9,1);
        temperatura=t;     // аналіз температури
       switch(flagwx)        //
         {
          case 0:         // flagwx-
            if (deltap > 250){ lcd_putsf("SUNNY");};        //
          break;
         case 1:         // flagwx-хмарно
           if (deltap > 150){ lcd_putsf("SUNNY");};
         break;
            case 2: // flagwx-сонячно
           if ((deltap < (-150)) && (deltap > (-200))) { lcd_putsf("CLOUD ");};
         break;
            default: break;
           }
        // ---------- end режим ------------
      };

  };
}
```