НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

КУРСОВИЙ ПРОЄКТ

Фаховий молодший бакалавр
(освітньо-професійний ступінь)

на тему: «**Проєктування систем керування та збору інформації на базі мікроконтролерів сімейства PIC16»**

Виконав студент IV курсу, групи ***КІ-41***
Спеціальності *123 Комп'ютерна інженерія*
Варіант № 19
Пугач Степан-Іван Ігорович
(прізвище, ім'я по батькові)

Керівник _____ ***Роман СТОЛЯРЧУК***
(підпис) (ім'я прізвище)

Курсовий проєкт перевірений
і допущений до захисту:
«___»_____2025р.

Курсовий проєкт захищений «__ » _____2025 р.
з оцінкою « ___ _____ ___ »

Львів 2025

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ЗАТВЕРДЖЕНО
на засіданні циклової комісії
«Фундаментальної підготовки»
Протокол № 1 від 28 серпня 2025 р.
Голова комісії

_____ Ольга ЛАБАЗ

ЗАВДАННЯ

на курсовий проєкт варіант **№ 19**

Пугачу Степану-Івану Ігоровичу
(прізвище, ім'я та по батькові)

з навчальної дисципліни: МІКРОКОНТРОЛЕРИ ТА ЇХ ПРОГРАМУВАННЯ

Студент групи: **КІ-41**

1. Тема проекту: Система моніторингу температури.

2. Дата видачі завдання: ”22” вересня 2025 р

3. Термін здачі курсового проєкту: ”15” грудня 2025 р.

4. Вихідні дані до проєкту:

4.1 Пристрої введення інформації : DS1307, кнопки
4.2 Функції пристрою введення інформації: налаштування секундоміра-таймера, RTCC
4.3 Пристрій виведення інформації : CCI-4 р., (CA),біпер
4.4 Функції пристрою виведення інформації: по секундна інкрементація / декрементація часу
4.5 Використовувані мікроконтролери : PIC16F689, PIC16F884
4.6 Протокол обміну інформацією : SPI

5 Перелік обов’язкових демонстраційних креслень:

5.1 Узагальнена структурна схема МКС
5.2 Блок-схема алгоритму роботи МКС
5.3 Скриншот роботи проєкту у програмі PROTEUS

6. Склад розрахунково – пояснювальної записки (перелік питань до розробки):

ВСТУП
1 Розробка структурної схеми МКС
  1.1 Аналіз завдання та синтез структурної схеми МКС
  1.2 Обгрунтування вибору мікроконтролерів
  1.3 Призначення, принцип роботи, особливості периферійних пристроїв
2 Розробка електричної принципової схеми МКС
  2.1 Розробка схеми електричної принципової мікроконтролера, периферійних пристроїв та їх інтерфейсу.
  2.2 Створення проекту з використанням програми PROTEUS
3 Розробка програмного забезпечення
  3.1 Розробка алгоритму функціонування МКС
  3.2 Написання керуючої програми для мікроконтролера :
    - робота з програмою MPLAB, створення проекту;
    - конфігурування МК з використанням директиви __CONFIG;
    - ініціалізація мікроконтролерів;
    - ініціалізація периферійних модулів;
    - пояснення до написання основної частини програми.
  3.3 Використання програми PROTEUS для аналізу роботи проекту.
ВИСНОВКИ
ПЕРЕЛІК ПОСИЛАНЬ

Календарний план

Назва етапів	Термін виконання	Примітка
Вступ		
1 Розробка структурної схеми		
2 Розробка електричної принципової схеми		
3 Розробка програмного забезпечення		
Висновки		
Перелік посилань		

Студент _____ **Степан-Іван ПУГАЧ**
(підпис) (імя та прізвище)

Керівник проекту _____ **Роман СТОЛЯРЧУК**
(підпис) (імя та прізвище)

РЕФЕРАТ

Текстова частина курсового проєкту: 37 сторінок, 16 рисунків, 8 посилань.

Об’єкт проєктування - “Секундомір-таймер”.

Мета проєкту - довершити свої навики у розробці структурної схеми, електричної принципової схеми МКС, створення проєкту з використанням програми PROTEUS, розробці алгоритму функціонування МКС, а також написання керуючої програми для мікроконтролерів. Закріпити теоретичні знання з основних понять дисципліни “Мікроконтролери та їх П”.

Результат роботи над курсовим проєктом показав, що завдяки використанню двох простих мікроконтролерів та модуля реального часу можна створити повноцінний секундомір-таймер.

МК, DS1307, SPI, CCI-4 p., (CA), SPLAN, MPLAB IDE.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЦП	Аналого-цифрові перетворювачі
ВІС	Великі інтегральні схеми
ГЗ	Глобальна задача
ГТІ	Генератор тактових імпульсів
ЗЗ	Загальна задача
ЗМП	Задачі малої розмірності
МК	Мікроконтролер
МКП	Мікроконтролерні пристрої
МКС	Мікроконтролерна схема
НЗ	Найпростіші задачі
ОМК	Однокристальні мікроконтролери
ПЗП	Постійний запам’ятовуючий пристрій
ЦАП	Цифроаналогові перетворювачі
ЦПП	Центральний процесорний пристрій
ШІМ	Широтно-імпульсна модуляція
I2C	Inter-Integrated Circuit (Міжінтегральна схема)
SPI	Serial Peripheral Interface (Послідовний периферійний інтерфейс)
UART	Universal Asynchronous Receiver Transmitter (Універсальний асинхронний приймач-передавач)

ЗМІСТ

ВСТУП

Типовими представниками RISC-процесорів є PIC-контролери (Periferial Interface Controller – контролери периферійних інтерфейсів) виробництва фірми MicroChip.

PIC-контролери застосовуються у системах високошвидкісного керування автомобільними й електричними двигунами, приладах побутової електроніки, телефонних приставках з АВН, системах охорони із сповіщенням по телефонній лінії, міні-АТС. Окремі вимірювальні інформаційні системи відрізняються розрядністю ПЗП(постійно запам'ятовуючого пристрою): від 12 до 14 біт для серії PIC16Cxx, 16 біт – для серії PIC17Cxx.

Завдяки скороченій кількості команд (від 33 до 35) усі команди займають у пам'яті одне слово. Час виконання кожної команди, крім команд розгалуження, становить чотири такти – один цикл (200 нс на частоті 20 МГц). Оперативний запам'ятовувальний пристрій виконано за схемою з довільною вибіркою з можливістю безпосередньої адресації у коді команди до будь-якої комірки.

Стек реалізовано апаратно з глибиною 2, 8 або 16 комірок. Майже в усіх PIC-контролерах є система переривань, джерелом яких може бути таймер, а також зміна станів сигналів на деяких входах. У PIC-контролерах передбачений біт захисту ПЗП, що запобігає нелегальному копіюванню.

Великі інтегральні схеми PIC16Cxx мають вбудовані ПЗП ємністю від 0,5 до 4 кілослів і ОЗП(оперативно-запам'ятовуючий пристрій) ємністю 32-256 байт. Основна частина контролерів має однократно програмований ПЗП, однак деякі контролери містять ПЗП з ультрафіолетовим стиранням, а PIC 16C84 містить пам'ять програм і пам'ять даних на базі ПЗП з електричним стиранням.

Крім того, контролери мають від одного до трьох таймерів, вбудовану систему скидання, watchdog таймер, внутрішній тактовий генератор, який може запускатися як від кварцового резонатора, так і від RC-ланцюга у широкому діапазоні частот – 0-25 МГц. Кількість розрядів портів – 12-33. Кожний розряд порту можна запрограмувати на введення або на виведення.

Контролер PIC16C64 додатково має вихід з ШІМ, за допомогою якого можна реалізувати ЦАП з розрядністю до 16 розрядів, а також послідовний двонапрямлений синхронний порт з інтерфейсами SPI, I2C, SCI/UART. PIC 16C71 і PIC 16C74 мають внутрішній 8-розрядний АЦП із пристроєм вибирання/зберігання і вхідним аналоговим мультиплексором.

1 РОЗРОБКА ТРУКТУРНОЇ СХЕМИ

1.1 Аналіз завдання та розробка структурної схеми МКС

При проектуванні мікроконтролерних пристроїв (МКП) або систем (МКС) можна використовувати блоково-ієрархічний підхід, при якому уявлення про МКП або МКС, що проектуються, розчленовуються на ієрархічні рівні. На вищому рівні використовується найменш деталізоване уявлення, що відображає лише тільки загальні риси і особливості системи, що проектується. На наступних рівнях ступінь подробиць розгляду зростає, при цьому система розглядається не загалом, а окремими блоками. Такий підхід дозволяє на кожному рівні формулювати і вирішувати задачі допустимої складності, що піддаються усвідомленню й розумінню людиною та рішенню за допомогою доступних засобів. Переваги такого підходу полягають в тому, що складна задача великої розмірності розбивається на групи задач малої розмірності, які послідовно вирішуються, причому всередині груп різні задачі можуть вирішуватися паралельно.

Структурна схема – це схема, яка визначає основні функціональні частини виробу, їх взаємозв'язки та призначення. Під функціональною частиною розуміють складову частину схеми: елемент, пристрій, функціональну групу, функціональну ланку.

Структурна схема призначена для відображення загальної структури пристрою, тобто його основних блоків, вузлів, частин та головних зв'язків між ними. Із структурної схеми повинно бути зрозуміло, навіщо потрібний даний пристрій і як він працює в основних режимах роботи, як взаємодіють його частини. Позначення елементів структурної схеми можуть обиратись довільно, хоча загальноприйнятих правил виконання схем слід дотримуватись.

Об’єктом дослідження даного курсового проекту є Секундомір-таймер, що працює на онові 2 мікроконтролерах PIC16F876, PIC16F883.

Робота приладу забезпечується двійково-десятковим годинником-календарем з низьким споживанням струму DS1307, а керування відбуватиметься за допомогою кнопок.

Вивід інформації здійснюватиметься через семисегментний індикатор ССІ-4 р., (СА) та акустичний елемент біпер. Ці елементи показуватимуть по секундну інкрементація / декрементація часу.

Усі прилади даного проекту з’єднуються за протоколом SPI (синхронної послідовної передачі даних).

Структурна схема МКС для Секундоміра-таймера визначає основні функціональні частини виробу та зазначена на рисунку 1.1.

Перший блок даної схеми відповідає за зчитування інформації, що буде керуватися кнопками. Наші дані після зчитування пройдуть через два мікроконтролера по протоколу SPI.

У другому блоці схеми Секундоміра-таймера зображено графічний та звуковий вивід інформації, що посекундно інкрементувалася/декрементувалася.

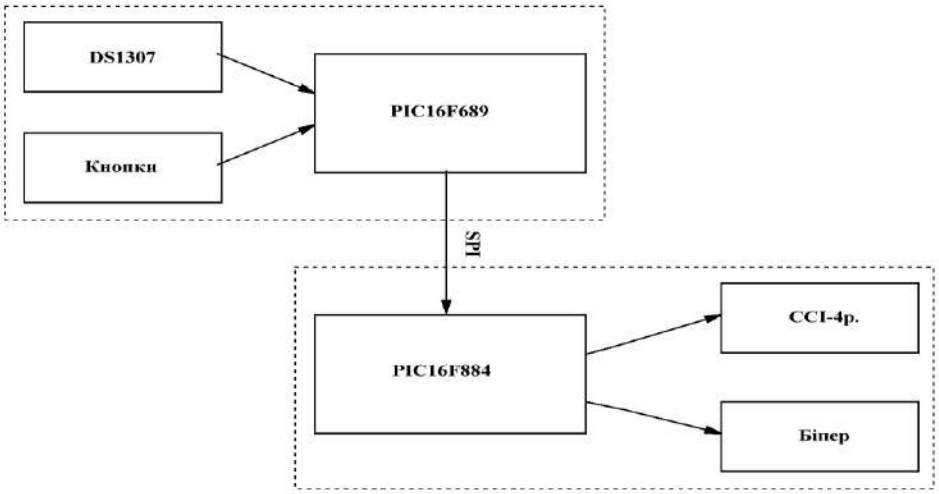

Рисунок 1.1 - Структурна схема системи секундоміра-таймера

1.2 Вибір мікроконтролерів та аналіз їх параметрів

В останні роки при розробці систем керування все більше уваги приділяється мікроконтролерній техніці. Це пов'язано з її бурхливим розвитком і широким асортиментом пропонованої продукції. Використання мікроконтролерів дозволяє конструювати пристрої з невеликими габаритами, що є відносно дешевими, простими і надійними, сумісними з персональним комп'ютером через стандартні інтерфейси.

Основна мета при виборі мікроконтролера - обрати мікроконтролер з мінімальною ціною, але в той же час він повинен відповідати вимогам продуктивності, надійності, умовам застосування, тощо.

Наступний крок при виборі мікроконтролера – пошук моделей, які задовольняють системним вимогам. Він включає підбір літератури, технічних описів і технічних комерційних журналів, а також демонстраційні консультації. Остання стадія вибору складається з кількох етапів, мета яких - звузити список прийнятних мікроконтролерів до одного. Ці етапи включають в себе аналіз ціни, доступності, засобів розробки, підтримки виробника, стабільності та наявності інших виробників. Проведення системного аналізу проекту дозволяє визначити вимоги до мікроконтролера:

- розрядність обчислювального ядра;
- набір вбудованих периферійних пристроїв (таймери, АЦП тощо...);
- апаратна організація обробки даних (структура машинного циклу, співвідношення тактів ГТІ і машинних циклів);
- можливість роботи по перериванню, за зовнішніми сигналами готовності або по командам людини;
- кількість керованих портів введення/виводу, характер передачі - байт або біт, програмне налаштування напрямку передачі;

- тип пристроїв введення/виводу, якими повинен керувати обирають мікроконтролер в проектованій системі (термінали, вимикачі, реле, клавіші, давачі, цифрові пристрої візуальної індикації, аналого-цифрові й цифро-аналогові перетворювачі, модулятори тощо);
- підтримувані способи завантаження програм в мікроконтролер, -можливість внутрішньосистемного програмування (ISP), використання при цьому стандартизованих інтерфейсів (SPI, I2C);
- кількість і тип напруги живлення;
- малогабаритні та естетичні обмеження;
- умови навколишнього середовища, необхідні для експлуатації.

Мікроконтролери PIC16F689 та PIC16F884 мають режими енергозбереження і можливість захисту коду програми.

У мікроконтролери вбудований сторожовий таймер WDT, який може бути вимкнений тільки в бітах конфігурації мікроконтролера. Для підвищення надійності сторожовий таймер WDT має власний RC генератор.

Додаткових два таймера виконують затримку старту роботи мікроконтролера. Перший, таймер запуску генератора (OST), утримує мікроконтролер в стані скидання, поки не стабілізується частота тактового генератора. Другий, таймер включення живлення (PWRT), спрацьовує після включення живлення і утримує мікроконтролер в стані скидання протягом 72мс (типове значення), поки не стабілізується напруга живлення. У більшості додатків ці функції мікроконтролера дозволяють виключити зовнішні схеми скидання.

Режим SLEEP призначений для забезпечення наднизького енергоспоживання. Мікроконтролер може вийти з режиму SLEEP по сигналу зовнішнього скидання, по переповненню сторожового таймера або при виникненні переривань.

Вибір режиму роботи тактового генератора дозволяє використовувати мікроконтролери в різних додатках. Режим тактового генератора RC дозволяє зменшити вартість пристрою, а режим LP знизити енергоспоживання. Біти конфігурації мікроконтролера використовуються для вказівки режиму його роботи.

1.3 Опис периферійних пристроїв та робота з ними

1.3.1 Давач температури DS1307

Модуль реального часу з послідовним інтерфейсом DS1307 –це двійково-десятковий годинник-календар з низьким споживанням струму. Даний модуль відраховує секунди, хвилини, години, день, дату, місяць і рік. Остання дата місяця автоматично коректується для місяців з кількістю днів менше 31, включаючи корекцію для високосного року. Модуль працює як в 24-годинному, так і в 12-годинному режимах з індикацією AM/PM. DS1307 має інтегровану схему контролю живлення, яка відслідковує проблеми живлення і у випадку їх детектування автоматично переключає модуль на живлення від іншого джерела

живлення. На рисунку 1.2 представлена типова схема підключення модуля DS1307 до мікроконтролера. Розміщення виводів наведено на рисунку 1.3.

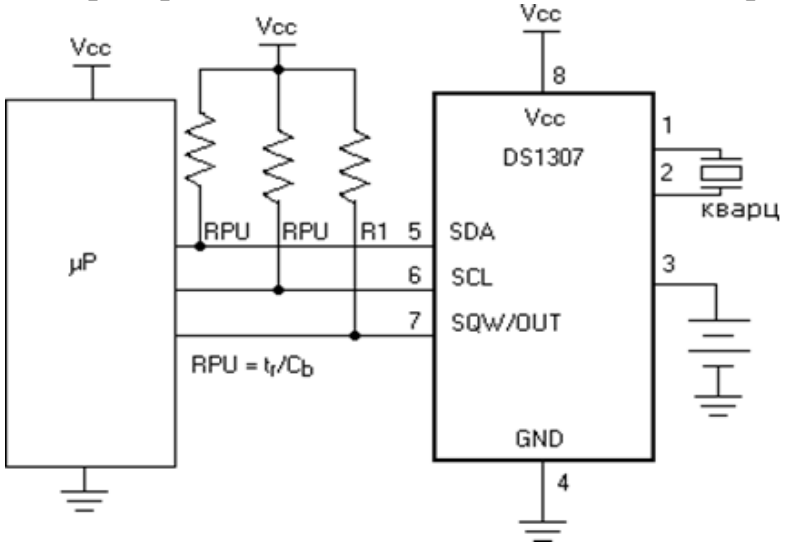

Рисунок 1.2 - Типова схема підключення модуля DS1307 до мікроконтролера

1.3.2 Робота з семисегментними індикаторами

CCI-4 р., (CA) - чотирьох розрядний семисегментний індикатор з типом підключення - спільний анод. Він має матрицю із семи світлодіодів, розмішених таким чином, що, засвічуючи їх у різних комбінаціях можна відобразити будь-яку десяткову цифру (від 0 до 9) та деякі латинські літери. Доповнюють його восьмим маленьким сегментом, що призначений для відображення десяткової коми.

Умовно кожен світлодіод в індикаторі має свій буквений ідентифікатор (a, b, c, d, e, f, g, h), і одна з ніжок світлодіода підключена до відповідного зовнішнього виводу. Інакші ніжки з'єднані разом і підключені до загального виводу.

Типовий спосіб підключення декількох індикаторів полягає в тому, щоб включити їх паралельно й потім керувати протіканням струму через загальні виводи окремих індикаторів. Через те, що величина цього струму перевищує припустиме значення вихідного струму МК, то для керування струмом включаються додаткові транзистори, які вибирають, котрий з індикаторів буде перебувати в активному стані.

Семисегментним індикатором можна керувати статично або динамічно. При статичному керуванні розряди індикатора підключені до мікроконтролера незалежно й інформація на них виводиться постійно. Суть статичної індикації полягає в постійному підсвічуванні індикатора від одного джерела. Тобто необхідно підключити індикатор до портів мікроконтролера, налаштувати ці порти на вихід і записати в ці порти дані так, щоб на індикаторах висвітилося щось зрозуміле, і відповідно міняти значення в портах при необхідності.

Щоб зменшити кількість необхідних виводів, застосовується динамічна індикація. Сутність динамічної індикації полягає в почерговому включенні індикаторів через загальний ланцюг перетворення коду. Підключення індикаторів необхідно робити із частотою f=120...140Гц, щоб уникнути мерехтіння індикаторів.

1.3.3 Модуль TMR0

Модуль TMR0 - це апаратно реалізований 8-розрядний таймер/лічильник з можливістю читання й запису поточного значення регістра-лічильника. Реалізований на основі програмно доступного 8-розрядного регістра TMR0, який знаходиться в 0-вому банку пам'яті даних за адресою 01h.

Модуль TMR0 конструктивно містить 8-розрядний програмований попередній дільник. Працює на основі сигналів внутрішнього або зовнішнього джерела тактового сигналу, при переповненні (перехід від FFh до 00h) формує сигнал переривання TOIF, який може бути використаний для зупинки основної програми та переходу на обробку підпрограми переривання.

Керування роботою модуля TMR0 здійснюється з використанням програмно доступного регістра OPTION, який знаходиться в 1-ому банку пам'яті даних за адресою 81h.Регістр OPTION доступний для читання й запису, містить біти керування:

біт 5: T0CS: Вибір тактового сигналу для TMR0
1= зовнішній тактовий сигнал з виводу RA4/T0CKI
0 = внутрішній тактовий сигнал CLKOUT

біт 4: T0SE: Вибір фронту збільшення TMR0 при зовнішньому тактовому сигналі
1= збільшення по задньому фронті сигнала на виводі RA4/T0CKI
0 = збільшення по передньому фронті сигнала (до високого рівня)

біт 3: PSA: Вибір включення попереднього дільника
1= попередній дільник включений перед WDT
0 = попередній дільник включений перед TMR0

біти 2-0: PS2: PS0: встановлення коефіцієнта ділення попереднього дільника

Біти PS0,PS1,PS2 визначають коефіцієнт попереднього дільника. (Попередній дільник - це послідовний ланцюжок з 8-ми тригерів, кожний з яких ділить на 2. Таким чином, максимальний коефіцієнт поділу = 256, і він, у межах дозволених таблицею значень, може задаватися значеннями бітів PS0,PS1,PS2.

Попередній дільник може бути включений або перед таймером TMR0, або після сторожового таймера WDT. Це визначає 3-й біт регістра OPTION (PSA).

Через кожні 256 імпульсів (при переході зі стану FFh у стан 00h) відбувається переповнення таймера (перехід на нове кільце рахунку), кількість яких (при необхідності підрахунку кількості імпульсів більшого, ніж 256) підраховується. Наприклад: за 1сек. відбулося 10 переповнень, значить TMR0 порахував 2560 імпульсів. Якщо приплюсувати до цієї кількості вміст TMR0 на момент закінчення рахунку, то одержимо точну кількість імпульсів, що надійшли на вхід TMR0 за 1сек. Якщо перед TMR0 включений попередній дільник, то в підсумок підрахунку вносяться відповідні корективи, що зумовлені заданим коефіцієнтом поділу попереднього дільника й числом, що знаходиться у попередньому дільнику на момент закінчення рахунку.

На такому підрахунку й ґрунтується принцип роботи пристроїв, що роблять облік імпульсів за заданий інтервал часу.

4-тим бітом регістра OPTION (TOSE) встановлюється момент спрацьовування таймера TMR0 від зовнішньої імпульсної послідовності. При встановленні біта TOSE в 1, інкремент відбувається по спадах імпульсної послідовності (перехід від 1 до 0), що є присутнім на виводі RA4/TOCKI, а при TOSE = 0 - по фронтах (перехід від 0 до 1).

Значення 5-го біта регістра OPTION (TOCS) визначає, який сигнал буде подаватися на вхід TMR0: або зовнішній сигнал з рахункового входу RA4/TOCKI (біт установлюється в 1), або внутрішній тактовий сигнал CLKOUT (біт установлюється в 0). Модуль TMR0 може працювати в одному з 2-х режимів: таймера або лічильника.

Переривання у модулі TMR0 відбувається тоді, коли формується переповнення регістра таймера/лічильника при переході від FFh до 00h. Тоді встановлюється біт запиту T0IF у регістрі INTCON<2>. Дане переривання можна замаскувати бітом T0IE у регістрі INTCON<5>. Біт запиту T0IF повинен бути скинутий програмно при обробці переривання.

Переривання по TMR0 не може вивести процесор з режиму SLEEP тому, що таймер у цьому режимі не функціонує.

При PSA=1 дільник буде приєднаний до сторожового таймера як попередній дільник (дільник на виході). При використанні попереднього дільника разом з TMR0, всі команди, що змінюють уміст TMR0, обнуляють попередній дільник. Якщо попередній дільник використовується разом з WDT, команда CLRWDT обнуляє вміст попереднього дільника разом з WDT.

Попередній дільник є 8-бітним лічильником, який використовується як дільник частоти перед TMR0 або після Watchdog таймера.

Якщо дільник з'єднаний з TMR0, то він не може бути підключений до Watchdog таймера і навпаки. Сторожовий таймер WDT це RC-очікуючий мультивібратор з перезапуском, що формує імпульс тривалістю приблизно

18мс. Якщо робота WDT дозволена, то після старту програми, він запускається, і якщо його, в інтервалі часу до 18мс., не запустити знову, то він закінчить формування імпульсу, а по його задньому фронті, сформується сигнал RESET. У випадку "зависання" програми, WDT, скинувши програму на початок, може вивести її із цього стану. Для забезпечення цього сторожового режиму, у ході виконання програми, необхідно періодично скидати WDT (не допускати його спрацювання).
Якщо після WDT поставити попередній дільник, то період скидання WDT можна збільшити (це залежить від заданого Кділ попереднього дільника)

Попередній дільник може підключатись по-різному. Коли він приєднаний до TMR0, всі команди, що здійснюють запис в TMR0 (наприклад, CLRF 1, MOVWF 1, BSF 1, тощо...) будуть одночасно обнулювати дільник. Коли дільник підключений до WDT, команда CLRWDT також обнулить дільник разом з Watchdog таймером. Підключення дільника – програмно кероване.

1.3.4 Особливості протоколу SPI

Послідовний периферійний інтерфейс (*Serial Peripheral Interface* - SPI) забезпечує високошвидкісний синхронний обмін даними між мікроконтролером і периферійними пристроями, а також між декількома мікроконтролерами.

В SPI режимі можливі одночасний синхронний прийом і передача 8-розрядних даних.

Режим ведучого SPI. Ведучий може ініціалізувати передачу даних у будь-який момент, оскільки він генерує тактовий сигнал, і визначає, коли ведений повинен передати дані відповідно до використовуваного протоколу.

У режимі ведучого дані передаються/прийняті після їхнього запису/читання з регістра SSPBUF. Якщо в SPI режимі потрібно тільки приймати дані, вивід SDO може бути заблокований (настроєний як вхід). Дані з виводу SDI послідовно зсуваються в регістр SSPSR із встановленою швидкістю. Кожний прийнятий байт завантажується в регістр SSPBUF (як нормально отриманий байт) з формуванням переривань і впливом на відповідні біти статусу. Ця функція може бути корисна при реалізації "монітора шини".

Будь-яка небажана функція послідовного порта може бути виключена, налаштовуючи відповідні біти регістрів напрямку даних TRIS. Наприклад, якщо в режимі ведучого SPI виконується тільки передача даних, то виводи SDI й -SS можуть використовуватися як цифрові виходи, скинувши відповідні біти TRIS в '0'.

1.3.5 Робота з протоколом I2C

DS1307 підтримує обмін даними по протоколу I2C по двопровідній двонаправленій шині. Пристрій, який передає дані на шину, є передавачем, а пристрій, приймаючий дані, - відповідно приймачем. Ведучий пристрій генерує

синхроімпульси (serial clock - SCL), керує доступом до шини і генерує умови START і STOP. DS1307 працює на шині як ведений пристрій. Типова конфігурація шини з використанням протоколу I2C наведена на рисунку 1.3.

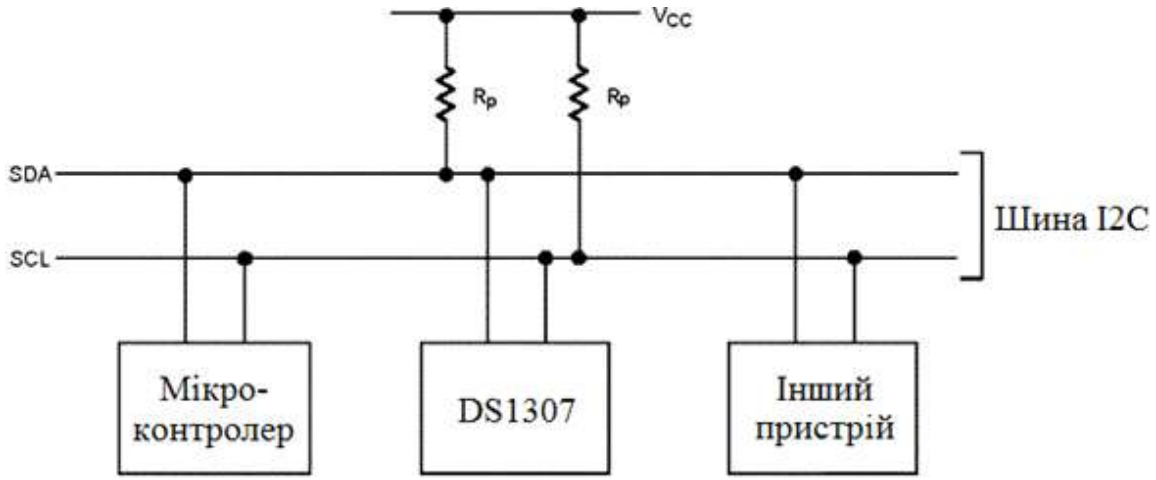

Рисунок 1.3 - Типова конфігурація двопровідної шини (I2C)

DS1307 на послідовній шині працює як ведений пристрій. Доступ до нього здійснюється встановленням умови START і передачею пристрою ідентифікаційного коду, після якого передається адреса регістру. До наступного за обраним регістром доступ здійснюється послідовно, поки не буде виконана умова STOP.

Кожна передача даних ініціюється умовою START і завершується умовою STOP. Кількість байтів даних, що передаються між умовами START і STOP, не обмежена і визначається ведучим пристроєм. Інформація передається по-байтово, і кожний байт приймач підтверджує дев‘ятим бітом (біт підтвердження - ACK). В специфікації двох провідного інтерфейсу визначені звичайний режим і швидкий режим (с тактовою частотою 400 кГц). DS1307 працює тільки в звичайному режимі (100 кГц).

Для підтвердження прийому пристрій має підтягнути до низького рівня лінію SDA під час тактового імпульсу таким чином, щоб на лінії SDA залишався стабільний низький рівень.

У режимі веденого приймача послідовні дані приймаються по SDA і синхронізуються по SCL. Після кожного прийнятого байту передається біт підтвердження. Умови START і STOP розпізнаються як початок і кінець послідовної передачі. Розпізнавання адреси виконується апаратно після прийому адреси веденого і біта напряму передачі.

2 ПРОЕКТУВАННЯ ЕЛЕКТРИЧНОЇ ПРИНЦИПОВОЇ СХЕМИ

2.1 Розробка схеми електричної принципової мікроконтролера

У процесі проектування будь-якого пристрою, формується набір технічної документації по цьому пристрою. В технічній документації описується параметри, робота пристрою, необхідні для виготовлення компоненти та методи і засоби налагодження функціональних можливостей проектованого пристрою.

Одним із найважливіших розділів технічної документації являється подання електричної принципової схеми проектованого пристрою.

Електрична принципова схема – це графічне зображення будови пристрою на фізичному рівні. Дана схема відображає кількість та тип використовуваних у пристрої компонентів (радіотехнічних елементів, мікросхем, інтерфейсних роз‘ємів, тощо…) та їх взаємодію один з одним. Підключення компонентів схеми один до одного відображається лініями. Згідно цієї схеми відбувається розведення доріжок на платі у відповідності до ліній підключення, зображених на схемі, та монтаж і пайка елементів проектованого пристрою. Усі елементи, зображені на електричні принциповій схемі, підписуються згідно вимог ДСТУ.

Жоден електричний пристрій не може працювати без джерела напруги живлення. Пристрої, що побудовані на основі мікроконтролера, це цифрові пристрої. Діапазон напруги живлення для таких пристроїв складає від 1.5 до 5.5В, а струм - постійний. Для підключення цифрових пристроїв до мережі 220В використовуються трансформаторні, імпульсні, конденсаторні блоки живлення тощо... До складу найпростішої схеми входять: понижуючий трансформатор, діодний міст, електролітичні конденсатори та мікросхема стабілізації (див. рисунок 2.1).

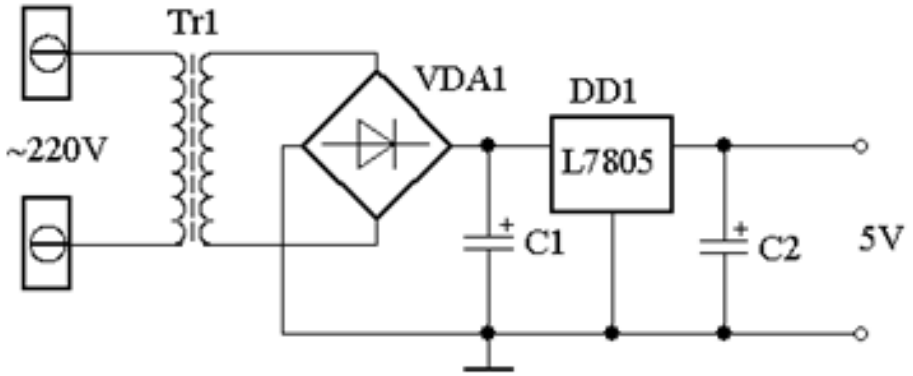

Рисунок 2.1- Схема стабілізації живлення для цифрових пристроїв на основі мікроконтролерів

Ключовим елементом в даній схемі є стабілізатор L7805. Існує два види стабілізатора 7805: з струмом навантаження до 1А і малопотужний 78L05 з струмом навантаження до 0,1А. Крім них є проміжний варіант 78M05 з струмом навантаження до 0,5А.

Ємність С1 на вході стабілізатора необхідна для згладження високочастотних завад при подачі вхідної напруги. Ємність С2 на виході стабілізатора задає стабільність напруги при різкій зміні струму навантаження, а також суттєво знижує амплітуду пульсацій. Для його нормальної роботи напруга на вході повинна бути у діапазоні 10-20 В.

Для формування напруги такого рівня використовується однофазний трансформатор та діодний міст. Для того, щоб сформувати необхідні умови для роботи МКС необхідно передбачити джерело імпульсів синхронізації.

В мікроконтролерах сімейства PIC16 передбачено декілька типів генераторів. Для PIC16 користувач може запрограмувати два конфігураційних біти (FOSC1 і FOSCO) для вибору одного із чотирьох режимів: RC, LP, XT, HS., де:

RC – генератор на RC-ланці;

XT – стандартний кварцовий генератор;

HS – високочастотний кварцовий генератор;

LP – низькочастотний малої потужності генератор.

В режимах XT, LP і HS до виводів OSC1/CLKIN і OSC2/CLKOUT підключається кварцовий або керамічний резонатор.

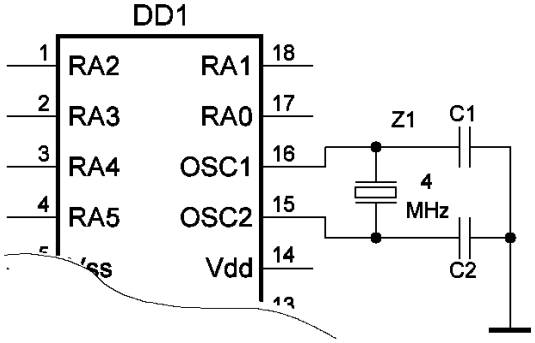

Рисунок 2.2 - Схема тактування мікроконтролера

Для підключення резонатора використовується типова схема виробника. Ємність конденсаторів повинна бути в інтервалі від 15 до 22 пФ, один вивід яких підключається до резонатора, а інший до землі.

У пристроїв, побудованих на основі мікроконтролерів передбачений механізм апаратного ресетування. У мікроконтролерів PIC16 вивід, що призначений для ручного перезапуску називається /MCLR. Активний сигнал для скидання – це сигнал логічного 0, що подається на вхід /MCLR при натисканні кнопки S1.

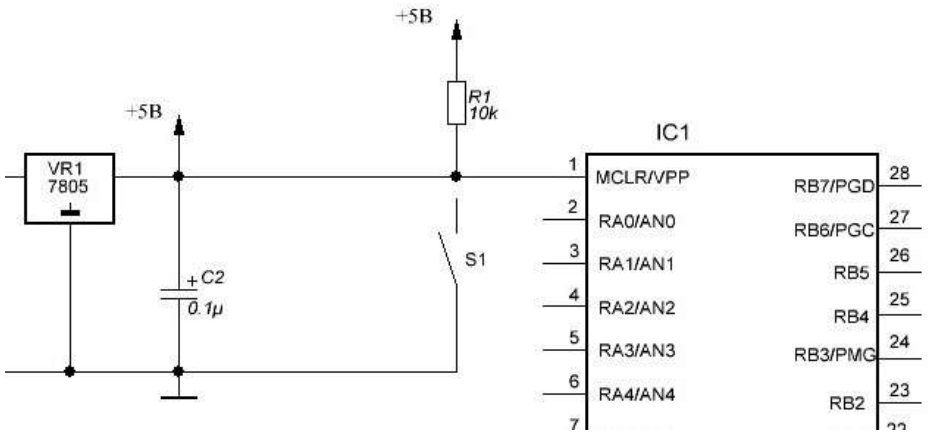

Рисунок 2.3 - Схема скидання мікроконтролера

Для зміни програмного забезпечення МК використовується роз‘єм ICSP. Даний роз‘єм, як і одноіменний протокол обміну, дозволяє змінювати робочі параметри мікроконтролера без вилучення його із пристрою. При прошиванні мікроконтролера на ніжку /MCLR подається сигнал рівнем 15В. Даний сигнал позначено на роз‘ємі як Vpp. Задіюється механізму доступу до пам‘яті мікроконтролера. Дані передаються по лінії Data на вивід PGD мікроконтролера, при цьому з лінії Clock роз‘єму ISCP поступає сигнал тактування. Даний сигнал приймається виводом PGC мікроконтролера, та відповідає за коректний запис програми у мікроконтролер. Сигнали Vss і Vdd забезпечують стандартну напругу живлення для мікроконтролера рівнем +5В.

На електричній принциповій схемі показано як взаємодіють різні вузли між собою та які компоненти використовуються для побудови проектованого пристрою. Згідно із цією схемою у подальшому формуватиметься принцип роботи усієї системи та алгоритм функціонування мікроконтролера, як основного вузла керування.

Пристрої введення/виведення – це різного роду периферійні модулі, які підключаються до функціонально завершеного пристрою для керування роботою цього пристрою та отримання інформації про виконувані ним функції. Наприклад, для ПК такими пристроями є клавіатура, мишка (пристрої введення) та монітор і принтер (пристрої виведення).

Для мікроконтролерних систем номенклатура пристроїв введення чи виведення дещо інша. Пристроями вводу крім клавіатури (клавіатурної матриці) можуть буди різні датчики – тепла, вологості, загазованості, ІЧ-датчики, тощо… В свою чергу пристрої виводу також не обмежуються монітором. Для МКС це, – як правило, – знакосимвольні або графічні РКІ, ССІ, акустичні елементи (біпери, динаміки), навідь звичайні світлодіоди.

Найпростіші із цих пристроїв введення/виведення інтегруються в пристрій та підключаються безпосередньо до портів мікроконтролера, проте є і такі, для підключення яких використовуються інтерфейсні роз‘єми.

Інтерфейс – у мікроконтролерній техніці – це набір провідників, які використовуються для підключення зовнішнього пристрою вводу/виводу та забезпечують реалізацію протоколу передачі даних. Як правило, такі групи провідників об‘єднюються в межах одного роз‘єму. Наприклад, для зовнішніх пристроїв, які працюють по USART/RS-232 протоколу, група провідників, яка забезпечує зв‘язок із мікроконтролером основного пристрою по цьому протоколу об‘єднується у роз‘єм типу D-SUB9.

Наведена електрична принципова схема працює наступним чином. Після подачі живлення DD1 покроково виконує програму, що закладена у його пам'ять програм. Першим етапом є ініціалізація портів мікроконтролера та його вбудованих модулів. Активується інтерфейс зв'язку та виконується обмін даними між мікроконтролерами.

Робота модуля IC2 керується кнопками S1(старт) та S2(стоп). Інформація з даного модуля переходить через МК DD1 та надходить до МК DD2 по протоколу SPI. IC2, після нажимання кнопки S1, почне відраховувати час, а при нажимані S2 - відрахування зупиниться.

З DD2 інформація надходить на пристрої 7Seg1 та біпер, що здійснюють виведення по-секундної інкрементації /декрементації часу.

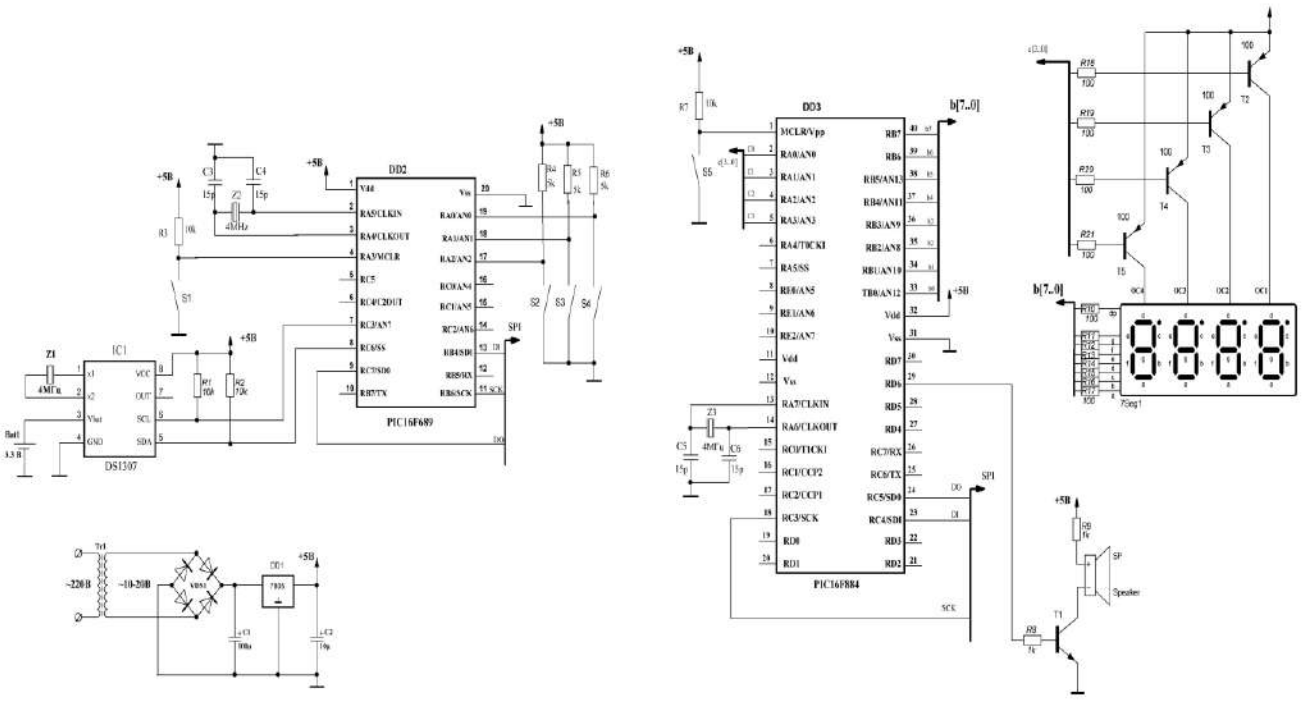

Рисунок 2.4 - Електрична принципова схема роботи Секундоміра-таймера

Таблиця 2.1 – Перелік елементів електричної принципової схеми

№ П/П	Назва елемента	Позначення	Номінал
1	Трансформатор	TR1	220В
2	Діодний міст	VDS1	D13A4
3	Регулятор напруги	DD1	7805
4	Кварцеві резонатори	Z1,Z2,Z3	4МГц
5	Конденсатор	C1	100мкФ
6	Конденсатор	C2	10мкФ
7	Конденсатори	C3, C4, C5, C6	15рФ
8	Резистори	R1,R2, R3, R7	10кОм
9	Резистори	R4, R5, R6	5кОм
10	Резистори	R8, R9	1кОм
11	Резистори	R10-R21	100Ом
12	Мікроконтролери	DD2,DD3	PIC16F689, PIC16F884
13	Транзистори	T1-T5	12В
14	4-розрядний ССІ	7Seg1	MPX4-CA
15	Кнопка-перемикач	S1 - S5	SK1-53
16	Біпер	Sp	Speaker
17	Модуль реального часу	IC1	DS1307

2.2 Створення проекту в середовищі PROTEUS VSM

Більшість радіоаматорів стикалися із ситуацією, коли за браком досвіду чи присутність помилок у схемі або ж за іншими обставинами, псували радіоелектронні компоненти.

З'явилася величезна кількість програм симуляторів, що замінюють реальні радіодеталі й прилади, віртуальними моделями. Симулятори дозволяють, без збирання реального пристрою, налагодити роботу схеми, знайти помилки, отримані на стадії проектування, зняти необхідні характеристики й багато чого іншого. Однією з таких програм є PROTEUS VSM.

Proteus VSM складається із двох самостійних програм ISIS та ARES. ARES це трасирувальник друкованих плат з можливістю створення своїх бібліотек. Запуск програми відбувається з меню Пуск – Програми - Proteus 7 Professional - ISIS 7 Professional. При запуску програми з'являється основне вікно програми ISIS 7 Professional.

Найбільший простір відведений під вікно редагування EDIT WINDOW. Саме в ньому відбуваються всі основні процеси створення, редагування й налагодження схеми пристрою. Ліворуч угорі маленьке вікно попереднього перегляду Overview Window з його допомогою можна переміщуватися по вікну редагування (клацаючи лівою кнопкою миші по вікну попереднього перегляду, можна переміщувати вікно редагування за схемою, якщо звичайно схема не

вміщається у вікно). Переміщувати вікно редагування за схемою можна ще утримуючи натиснутої кнопку SHIFT та рухаючи курсором миші по вікну редагування. Наближувати й віддаляти схему у вікні можна кнопками F6 й F7 або ж колесом миші, F5 центрує схему у вікні, а натискання F8 підганяє розмір схеми під-вікно редагування.

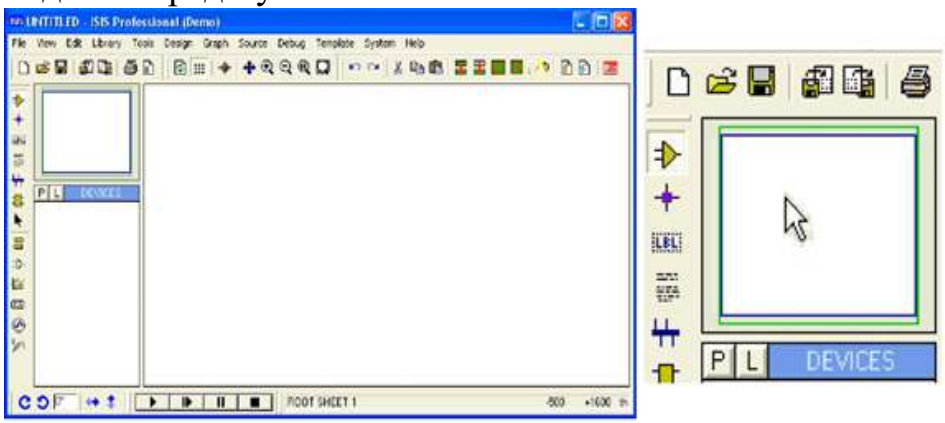

Рисунок 2.5 - Основне вікно програми ISIS 7 Professional і вікно попереднього перегляду.

Під вікном попереднього перегляду знаходиться Object Selector - список обраних у цей момент компонентів, символів й інших елементів. Виділений у списку об'єкт відображається у вікні попереднього перегляду.

Всі можливі функції та інструменти Proteus VSM доступні через головне меню, через піктограми які знаходяться під меню і у лівому куті основного вікна та через гарячі клавіші, які можуть перепризначуватися користувачем. Внизу основного вікна розташовані: кнопки обертання та розвороту об'єкта навколо своєї осі, панель керування інтерактивною симуляцією - виглядає як магнітофонна і функції такі ж: ПУСК , ПОКРОКОВИЙ РЕЖИМ, ПАУЗА,СТОП.

Рисунок 2.6 - Панель керування інтерактивною симуляцією.

Середовище PROTEUS має величезну бібліотеку електронних компонентів. Всі елементи перебувають у бібліотеці компонентів. Щоб до неї потрапити потрібно перейти в режим COMPONENT (компоненти), натиснувши відповідну піктограму P (Pick devices) або двічі клацнувши лівою кнопкою в полі вибору компонентів Object Selector.

Компоненти можна вибирати по категоріях, та у списку. Зображення компонента з'явитися у вікні попереднього перегляду. Щоб розмістити компонент на робочому полі потрібно навести курсор миші на потрібне місце і клацнути лівою кнопкою миші. Щоб з’єднати виводи компонентів провідником

потрібно підвести курсор миші до виводу компонента, після появи на виводі компонента хрестика потрібно клацнути лівою кнопкою миші і підвести курсор миші до виводу іншого компонента.

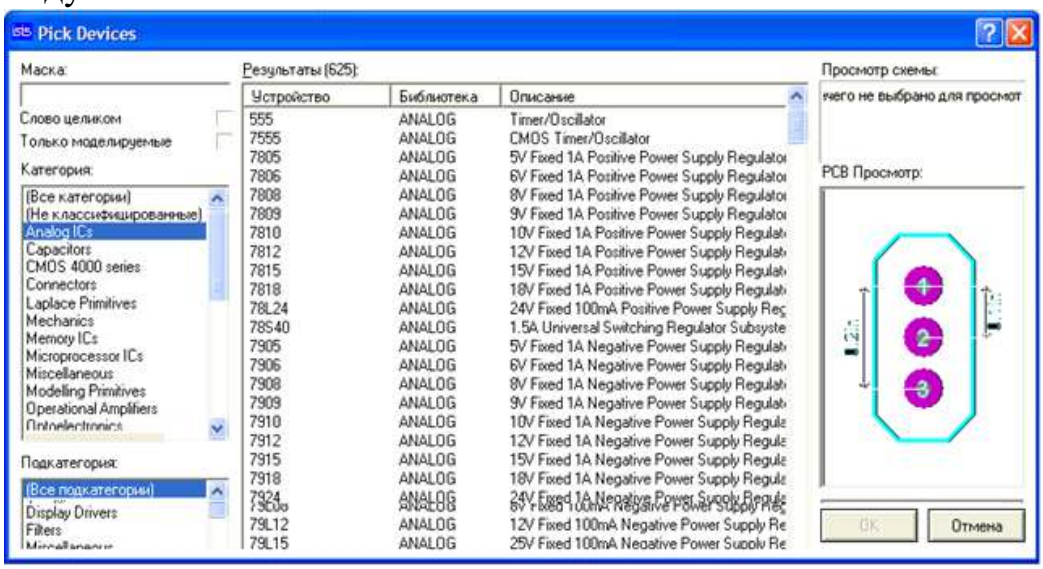

Рисунок 2.7 - Бібліотека електронних компонентів.

Для того, щоб переглянути роботу схеми, необхідний вихідний HEX-файл. Середовище PROTEUS підтримує багато засобів розробки, серед них і HI-TECH Сі компілятор і CROWHILL PIC BASIC і BASIC STAMP. Попередньо HEX-файл повинен бути створений з використанням середовищем програмування мікроконтролерів від фірми MICROCHIP, а саме MPLAB.

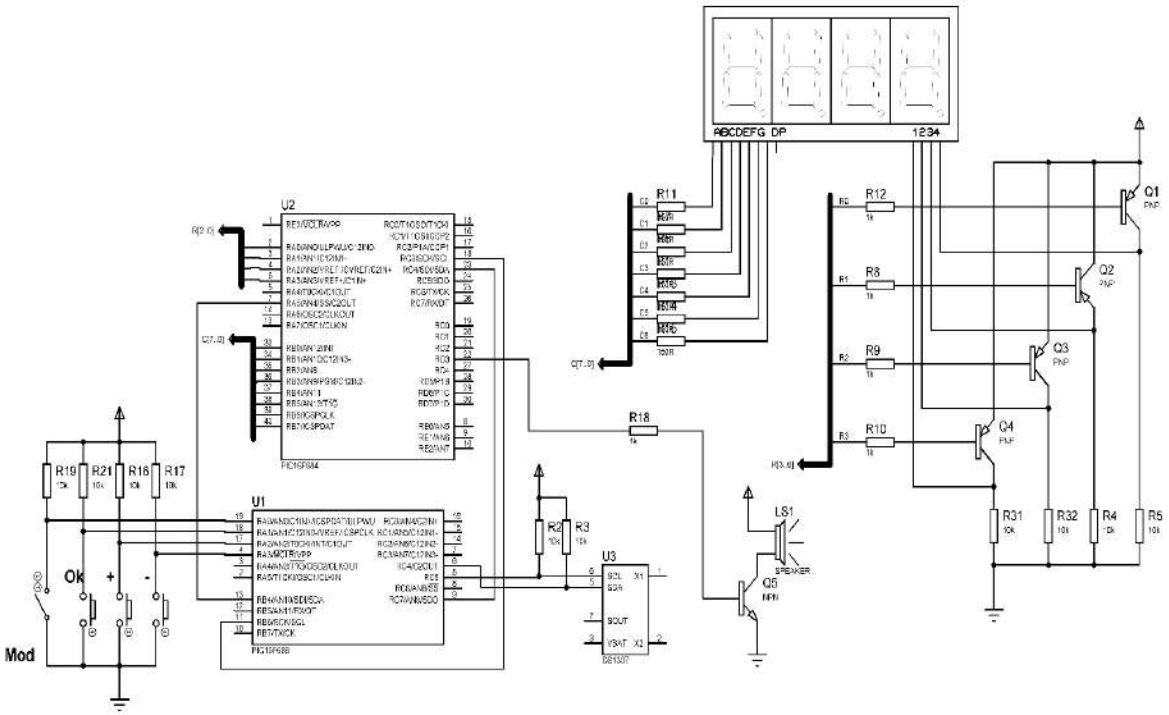

Рисунок 2.8 - Досліджувана схема

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МКС

3.1 Розробка алгоритму функціонування МКС

Алгоритм — це послідовність, система, набір систематизованих правил виконання обчислювального процесу, що обов'язково приводить до розв'язання певного класу задач після скінченного числа операцій. При написанні комп'ютерних програм алгоритм описує логічну послідовність операцій. Для візуального зображення алгоритмів використовують блок-схеми.

Блок-схема – схематичне зображення послідовності дій, спрямованих на досягнення спільної мети та їх особливостей. Розділ програми, у якому відбувається занесення значення змінних подається у вигляді паралелограма, в якому перераховані імена цих змінних. Розділ програми, у якому відбувається виконання певної операції подається у формі прямокутника, а розділ перевірки умови – у формі ромба. Крім того можуть використовуватись розділи, які повторюють певний набір команд для іншого аналогічного випадку. Такі розділи називають блоком ідентичних операцій та зображують у вигляді шестикутника. Ще один розділ, що є поширеним у програмах – це розділ виконання підпрограми. Зображується він у формі прямокутника із двома вертикальними смугами по краях фігури та записом у ньому назви підпрограми.

Кожен алгоритм є списком добре визначених інструкцій для розв'язання задачі. Починаючи з початкового стану, інструкції алгоритму описують процес обчислення, які відбуваються через послідовність станів, які, зрештою, завершуються кінцевим станом.

Принципи побудови програмного забезпечення точно такі ж, як і апаратної частини: створення кінцевого продукту з готових блоків - підпрограм. Якщо всю роботу програми розмістити у векторах переривань, то основний додаток ніяк не відчує виконання фрагментів сусідніх підпрограм. Головне визначити можливості апаратної частини.

Програмне забезпечення мікроконтролера необхідно будувати з максимальним використанням периферії. При цьому ядро не виконуватиме безлічі рутинних операцій.

Після включення живлення сервісна програма починає проводити ініціалізацію. На початку програми прикріплюється файл бібліотеки мікроконтролера, після чого оголошуються біти портів і регістри пам'яті, які використовуються в програмі. Мікроконтролер має вільні для використання комірки пам'яті і, щоб при кожному зверненні до певної комірки не описувати її адресу, та полегшити формування програми - присвоюється їй назва.

У поняття ініціалізації входить:

- присвоєння початкових значень всім змінним програми;
- налаштування всіх внутрішніх систем мікроконтролера;

- відновлення режимів роботи системи, встановлених у попередньому сеансі роботи шляхом зчитування відповідних параметрів з EEPROM-пам'яті;
- виконання підпрограм, призначених для налаштування всіх периферійних модулів у вихідний стан;
- виконання підпрограми запуску системних лічильників-таймерів і механізму переривань мікроконтролера.

Для використання у програмі констант і змінних необхідно їх попередньо описати та оголосити. Константа - це просто деяке число, що часто використовується в програмі, і має певний зміст.

Крім констант, які, відповідно до своєї назви, протягом усього часу роботи програми ніколи не міняються, у програмі існують змінні. Під змінною в програмі розуміють ім'я регістра, або адресу комірки пам'яті, де в процесі виконання програми буде тимчасово зберігатися заздалегідь невідома величина, або проміжний результат обчислень.

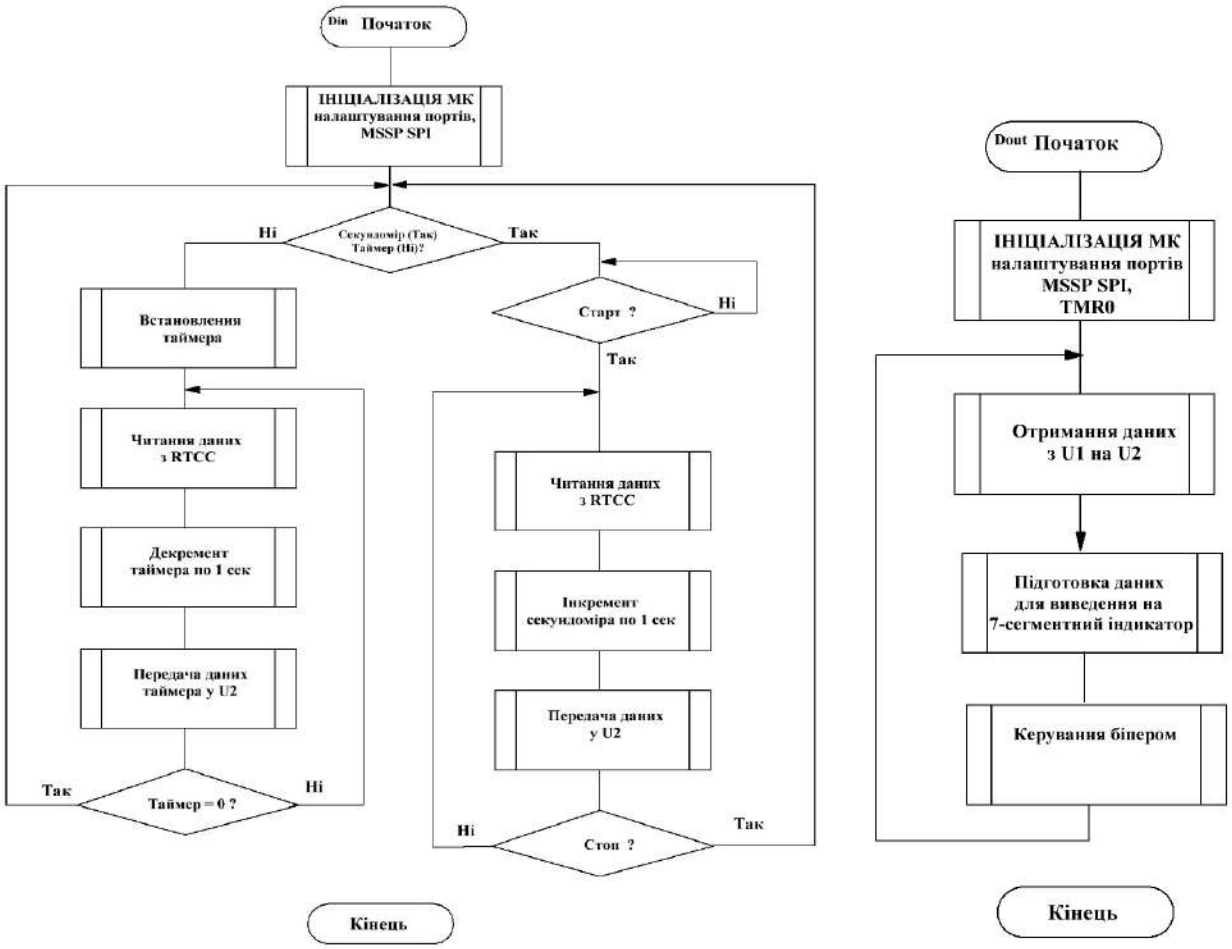

Рисунок 3.1 - Блок-схеми роботи Секундоміра-таймера

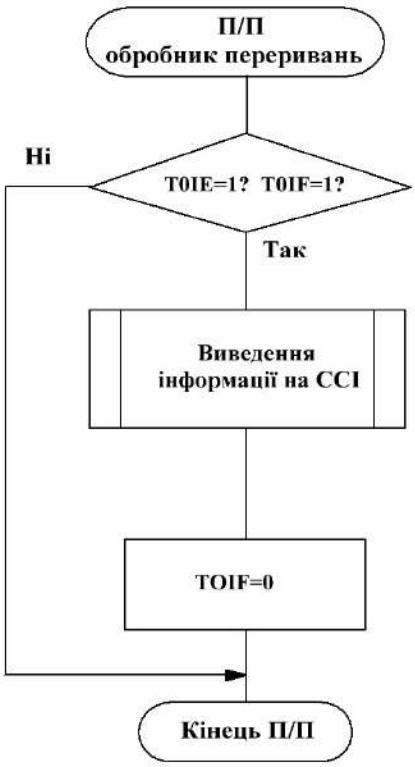

Рисунок 3.2 - Обробник переривання за TMR0

3.2 Конфігурування пристрою з використанням директиви __CONFIG

Сімейство мікроконтролерів PIC16 має набір спеціальних функцій, призначених для розширення можливостей системи, мінімізації вартості, виключення навісних компонентів, забезпечення мінімального енергоспоживання і захисту коду від зчитування. В PIC16 реалізовані наступні спеціальні функції:

- вибір типу генератора;
- скидання;
- схема скидання по включенню живлення ;
- таймер скидання (DRT);
- сторожовий таймер (WDT)
- режим пониженого енергоспоживання (SLEEP);
- захист коду від зчитування;
- біти ідентифікації.

В регістрі конфігурації описуються біти конфігурації, в які входять:

- біт захисту
- біт дозволу роботи таймера включення живлення

- біт дозволу роботи сторожового таймера
- вибір типу генератора

Біти конфігурації задаються за допомогою директиви __CONFIG.

Фрагмент заголовку при написання програми для мікроконтролера PIC16F689 з використанням мови С:

```
#include <pic.h>
__CONFIG(FOSC_HS&WDTE_OFF&PWRTE_OFF&MCLRE_OFFCP&CP_OFF
&CPD_OFF&BOREN_OFF&IESO_OFF&FCMEN_ON);
```

Фрагмент заголовку при написання програми для мікроконтролера PIC16F884 з використанням мови С:

```
#include <pic.h>
__CONFIG(FOSC_XT&WDTE_OFF&PWRTE_OFF&MCLRE_ON&CP_OFF&CP
D_OFF&BOREN_OFF&IESO_OFF&FCMEN_OFF&LVP_OFF&DEBUG_OFF);
__CONFIG (BOR4V_BOR4OV&WRT_OFF);
```

3.3 Написання керуючої програми для мікроконтролера

Після включення живлення сервісна програма починає проводити ініціалізацію даних. На початку програми прикріплюється файл бібліотеки мікроконтролера, після чого оголошуються біти портів і регістри пам'яті, які використовуються в програмі. Даний тип мікроконтролера має вільні для використання комірки пам'яті і, щоб при кожному зверненні до певної комірки не описувати її адресу, та полегшити формування програми - присвоюється їй назва.

У поняття ініціалізації входить:

- присвоєння початкових значень всім змінним програми;
- налаштування всіх внутрішніх систем мікроконтролера;
- відновлення режимів роботи системи, встановлених у попередньому сеансі роботи шляхом зчитування відповідних параметрів із флеш-пам'яті;
- виконання підпрограм, призначених для установки всіх периферійних пристроїв схеми позиціонера у вихідний стан (очищення індикаторів, підтвердження команди зупинки двигуна тощо...);
- виконання підпрограми запуску системних лічильників-таймерів і механізму переривань мікроконтролера.

Крім констант, які, відповідно до своєї назви, протягом усього часу роботи програми ніколи не міняються, у програмі існують змінні. Під змінною в програмі розуміється ім'я регістра, або адреса комірки пам'яті, де в процесі виконання програми буде тимчасово зберігатися заздалегідь невідома величина, або проміжний результат обчислень.

Після виконання модуля ініціалізації програма переходить до основного циклу. У цьому циклі програма перебуває увесь час, поки включене живлення.

Першою частиною нашого проекту було написання тексту програми для першого мікроконтролера, що виконує функцію одержання даних з пристрою DS1307, а також обробку та передачу даних на другий МК.

Наведений нижче модуль програми здійснює ініціалізацію портів мікроконтролера PIC16F689 та модуля SPI:

```
#include "MK_init.h"
//---- опис функцій ----------------------
void MK_init(void)
{
//--Налаштування портів--//
TRISA=1;//порт керування кнопками
TRISB4=1;//керуючий вивід протоколу SPI
TRISB6=1;//вивід SCK протоколу SPI
TRISC=0b00110000;//5 та 6 виводи приймають дані з DS1307, а 8 вивід - вивід SD0 протоколу SPI

//----------- MSSP SPI -----------
//--- SSPSTAT
SMP=0;// опитування входу в середині періоду виведення даних
CKE=0;// вибір фронту тактового сигналу, дані передаються по задньому фронту сигналу на вивід SCK
//--- SSPCON1
CKP=0;// дані передаються по задньому фронту сигналу на вивід SCK
// Режим роботи модуля MSSP:
// ведучий режим SPI, тактовий сигнал = Fosc/4
        CKP=0;
        SSPM3=0;
        SSPM2=0;
        SSPM1=0;
        SSPM0=0;
        SSPEN=1;
        GIE=0;
}
```

Після ініціалізації ми виконуємо написання керуючої програми у файлі main.c :

```
//---- Оголошуємо функцію для передачі даних на другий МК ----------------
//---- оголошення функції --------
void spi_trans(unsigned char dat);

//---- опис функцій ----------------------
void spi_trans(unsigned char dat)
{
RB4=0;
SSPBUF=dat;
while(BF==0){}
```

```
        BF=0;
        RB4=1;
        }
//Надаємо кнопкам імена, для зручності роботи
        #define Mod RA0 //перемикач  для вибору режиму роботи
        #define Ok  RA1 //кнопка Запуск-Старт/Стоп
        #define Inc RA2 //кнопка збільшення значення
        #define Dec RA3 //кнопка зменшення значення

        //---------------------------------------
        //---- ОСНОВНА ПРОГРАМА ------------------
        //---------------------------------------
        void main(void)
        {
        MK_init(); //функція ініціалізації МК

        //прирівнення до нуля змінних, для подальшої роботи
                tmr=0;
                flg=0;
                flg1=0;
                flg2=0;

        //---- РОБОЧИЙ ЦИКЛ -------------------
        while(1)
        {

        //-------- Робота з кнопками ----------------
        if(Mod==0)    // Timer
        {       flg2=0;
        if(flg1==0)//початкове значення таймера
        {
        spi_trans(0xFF);
        spi_trans(0x01);        // індикація режиму ТАЙМЕР
        spi_trans(0x03);
        spi_trans(0x00);
        spi_trans(0x00);
        flg1=1;
        }

        if(Inc==0)//встановлення таймера в режим інкрементації часу
        {
        while(Inc==0){}
        tmr++;
        spi_trans(0xFF);
        spi_trans(0x01);        // індикація режиму ТАЙМЕР
        spi_trans(0x03);
        spi_trans(0x00);
        spi_trans(tmr);
```

```
        }

        if(Dec==0) //встановлення таймера в режим декрементації часу
        {
        while(Dec==0){}
        tmr--;
        spi_trans(0xFF);
        spi_trans(0x01);        // індикація режиму ТАЙМЕР
        spi_trans(0x03);
        spi_trans(0x00);
        spi_trans(tmr);

        }

        if((Ok==0)&&(tmr>0))//перевірка підтвердження режиму таймера
        {
        while(Ok==0){}//очікування відтискання кнопки

        // запустити таймер по секундам з 1307
        //--------- виведення часу ----------------
        i2c_start_p();
        i2c_trans_p(0b11010000);//0xD0
        i2c_trans_p(0x00);
        i2c_stop_p();

        i2c_start_p();
        i2c_trans_p(0b11010001);//0xD1
        sec=i2c_recieve_nack_p();
        i2c_stop_p();
        sec_old=sec;
        sec_t=sec;

        while(tmr>0)//поки значення таймера буде більше “0”
        {
//буде здійснюватись :
        i2c_start_p();//старт програмної реалізації протоколу I2C
        i2c_trans_p(0b11010000);//0xD0//режим запису у пам’ять приладу
        i2c_trans_p(0x00);//запис самого значення
        i2c_stop_p(); //зупинка програмної реалізації протоколу I2C

        i2c_start_p(); //старт програмної реалізації протоколу I2C
        i2c_trans_p(0b11010001);//0xD1//режим читання з пам’яті приладу
        sec=i2c_recieve_nack_p();//запис зчитаних даних у змінну
        i2c_stop_p(); //зупинка програмної реалізації протоколу I2C

        if(sec_t!=sec) //перевірка на рівність між утвореним значенням та зчитаним
        {
```

```
//якщо умова справджується, то наша МКС виконує по-секундну інкрементацію часу:
        tmr--;

        spi_trans(0xFF);
        spi_trans(0x01);// індикація режиму ТАЙМЕР
        spi_trans(0x03);
        spi_trans(0x00);
        spi_trans(tmr);
        sec_t=sec;
        }

        __delay_ms(500);
        }
        }

        }

//Якщо кнопка Mod=1, то в нас включається режим секундоміра
        else            // Secundomir
        {
        flg1=0;
        tmr=0;
        if(flg2==0)
        {
        spi_trans(0xAA);
        spi_trans(0x01);        // індикація режиму СЕКУНДОМІР
        spi_trans(0x02);
        spi_trans(0x00);
        spi_trans(0x00);
        flg2=1;
        }

        if(Ok==0) //перевірка підтвердження режиму
        {
        while(Ok==0){} // очікування відтискання кнопки “ОК”
        flg=1;  // ознака, що кнопка Ок була один раз натиснута

        // ЗАПУСК відліку секунд по 1307
        i2c_start_p();//старт програмної реалізації протоколу I2C
        i2c_trans_p(0b11010000);//0xD0//режим запису у пам’ять приладу
        i2c_trans_p(0x00);//запис самого значення
        i2c_stop_p(); //зупинка програмної реалізації протоколу I2C

        i2c_start_p(); //старт програмної реалізації протоколу I2C
        i2c_trans_p(0b11010001);//0xD1 //режим читання з пам’яті приладу
        sec=i2c_recieve_nack_p(); //запис зчитаних даних у змінну
        sec_t=sec;//надання змінній зчитаного значення
        i2c_stop_p(); //зупинка програмної реалізації протоколу I2C
```

```
        while(flg==1) //в момент часу поки кнопка ОК одни раз, в нас в програму відбувається:
        {
        i2c_start_p(); //старт програмної реалізації протоколу I2C
        i2c_trans_p(0b11010000);//0xD0 //режим запису у пам'ять приладу
        i2c_trans_p(0x00);  //запис самого значення
        i2c_stop_p(); //зупинка програмної реалізації протоколу I2C

        i2c_start_p(); //старт програмної реалізації протоколу I2C
        i2c_trans_p(0b11010001);//0xD1 //режим читання з пам'яті приладу
        sec=i2c_recieve_nack_p(); //запис зчитаних даних у змінну
        i2c_stop_p(); //зупинка програмної реалізації протоколу I2C

        if(sec_t!=sec) //перевірка на рівність між утвореним значенням та зчитаним
        {
//якщо умова справджується, то наша МКС виконує по-секундну декрементацію часу:
        tmr++;
        spi_trans(0xAA);
        spi_trans(0x01);        // індикація режиму
                                //СЕКУНДОМІРА
        spi_trans(0x02);
        spi_trans(0x00);
        spi_trans(tmr);
        sec_t=sec;
        }
        __delay_ms(300);

        // ЗУПИНКА відліку секунд по 1307
        if((Ok==0)&&(flg==1))
        {
        while(Ok==0){tmr=0;flg=0;}
        break;
        }
        }
        }
        }
        }
        }
```

Другою частиною нашого проекту було написання тексту програми для другого мікроконтролера, що виконує функцію одержання даних з першого МК та виводить отримані дані на 1-рядковий РКІ.

Наведений нижче модуль програми здійснює ініціалізацію портів мікроконтролера PIC16F884 та модуля UART:

```
        #include "MK_init.h"
        //---- опис функцій ----------------------
        void MK_init(void)
```

```
{
//--Налаштування портів--//
ANSEL=0;//виключаємо аналогово-цифровий перетворювач
ANSELH=0;
TRISA=0b00100000;//перші 3 виводи - це виводи керування динамічною індикацією, а
6 вивід - керуючий за протоколом SPI
TRISB=0;//порт засвічення діодів кожного розряду
TRISC=1;//порт отримання даних з першого МК

//---------- MSSP SPI --------------------
//--- SSPSTAT
SMP=0; // опитування входу в середині періоду виведення даних
CKE=0; //вибір фронту тактового сигналу, дані передаються по задньому фронту
сигналу на вивід SCK

//--- SSPCON1
CKP=0; // дані передаються по задньому фронту сигналу на вивід SCK
// Режим роботи модуля MSSP:
// ведений режим SPI, тактовий сигнал з вивода SCK. Вивід -SS
SSPM3=0;
SSPM2=1;  //slave
SSPM1=0;
SSPM0=0;
SSPEN=1;

//---------- TMR0 -------------------------
TMR0=0;//скидання TMR0
INTCON=0; //виклик переривань і скидання T0IF
T0CS=0; //тактування внутрішнього джерела
PSA=0; //з попереднім дільником
//встановлення коефіцієнта розподілу 1:64
PS2=1;
PS1=0;
PS0=1;
//налаштування INTCON
INTCON=0;
T0IF=0;
T0IE=1;
GIE=1;
}
```

Після ініціалізації ми здійснуємо написання керуючої програми у файлі main.c :

```
//оголошення функції для отримання даниз з першого МК:
        unsigned char spi_receive(void);
//---- опис функцій ----------------------
        unsigned char spi_receive(void)
        {
```

```
        while(BF==0){}
        return SSPBUF;
        }

//---------------------------------------
//---- ОСНОВНА ПРОГРАМА ------------------
//---------------------------------------
        void main(void)
        {
        MK_init(); //функція ініціалізації МК
        cnt=0; //змінна для організації динамічної індикації
        R0=1;//змінна першого розряду
        R1=1; //змінна другого розряду
        R2=1; //змінна третього розряду
        R3=1; //змінна четвертого розряду

 //фрагмент,що  переводить символи для CCI зі CA
        for(char i=0;i<13;i++)
        {
        tmp=MAS1[i];
        tmp=~tmp;
        MAS[i]=tmp;
        }

//---------- РОБОЧИЙ ЦИКЛ ------------------
        while(1)
        {
        tmp=spi_receive(); //отримання та запис у змінну числа від першого МК для ідентифікації режиму роботи МКС

        if(tmp==0xFF)//якщо зчитана змінна дорівнює 0xFF, то в нас включається режим таймера:
        {
        //читання хвилин:
        h1t=spi_receive();
        h2t=spi_receive();
        //читання секунд:
        m1t=spi_receive();
        m2t=spi_receive();

//Та обробка отриманих даних для виведення на CCI
        h2=h2t-1;
        m1=(m2t/10)%10;
        m2=m2t%10;
        if((m1==m2t)&&(m2==m2t)){ h1=h1t; h2=h2t;  }

//включення та виключення біпера із затримкою 0,5 с:
        RC7=1;//
```

```
        __delay_ms(500);
        RC7=0;
        }

        else //інакше включається режим секундоміра
        {
//Після чого здійснюється те саме зчитування та обробка даних, що і в режимі таймера:
        h1t=spi_receive();
        h2t=spi_receive();
        m1t=spi_receive();
        m2t=spi_receive();
        h1=h1t;
        h2=h2t;
        if(m2t<=60){
        m1=(m2t/10)%10;
        m2=m2t%10;
        }
//запуск біпера
        RC7=1;
        __delay_ms(500);
        RC7=0;
        }
        }
        }

//Окрема функція, яка реалізовує постійну динамічну індикацію:
        void interrupt DYN(void)
        {
        if(T0IE&&T0IF)
        {
        switch(cnt)
        {
        case 0: R3=1;PORTB=MAS[m2];R0=0;R1=R2=1;break;
        case 1: R0=1;PORTB=MAS[m1];R1=0;R2=R3=1;break;
        case 2: R1=1;PORTB=MAS[h2];R2=0;R0=R3=1;break;
        case 3: R2=1;PORTB=MAS[h1];R3=0;R1=R0=1;break;
        }
        cnt++;
        if(cnt==4){cnt=0;}
        TMR0=0;
        T0IF=0;//очищщеня прапорця T0IF
        }
        }
```

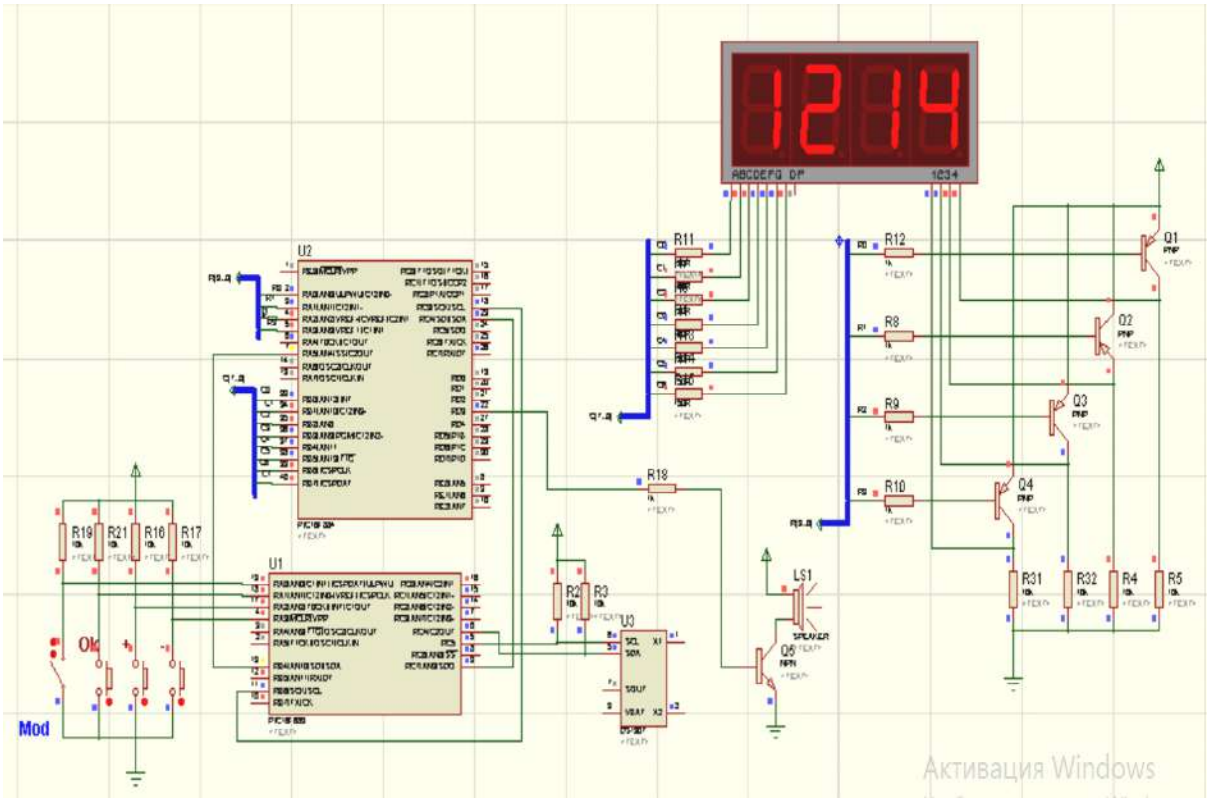

Рисунок 3.3 - Робоча схема у PROTEUS в режимі секундоміра

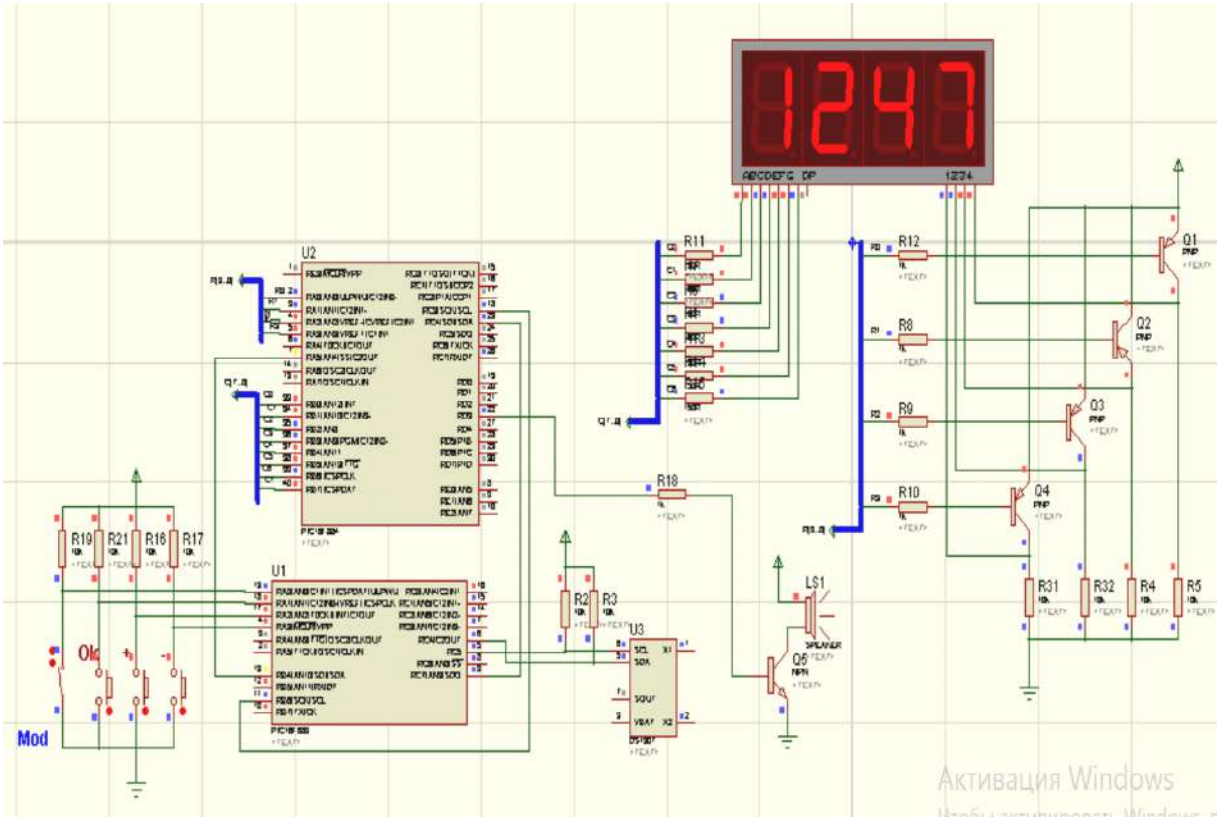

Рисунок 3.4 - Робоча схема у PROTEUS в режимі таймера

ВИСНОВКИ

В процесі курсового проєктування розглянуто принципи проектування пристроїв і систем, що побудовані з використанням однокристальних мікроконтролерів фірми Microchip. Під час виконання проектування здійснено закріплення теоретичних знань та практичних навичок по реалізації етапів проектування. Узагальнено технічний матеріал для формування основної ідеї проектування. Здійснено аналіз вхідних даних, обґрунтування і синтез структурної схеми. Детально розглянуто складові апаратної частини їх призначення і особливості функціонування. Проаналізовано документацію, що описує роботу складових вузлів МКС. На основі цих даних описано та спроектовано принципову схему та алгоритм роботи.

Для побудови блок-схем, алгоритмів та принципових схем закріплено навички роботи з програмою sPlan. Для написання і влагодження програмного забезпечення використано середовище розробки MPLAB 8.92. У цьому середовищі створено проект, написано текст програми з використанням мови С, і здійснено компіляцію у hex-файл. Середовище розробки може використовуватись разом з програматорами для запису готового вихідного коду прошивки у пам'ять програми вибраного мікроконтролера.

При написанні тексту програми сформовано значення регістра конфігурації за даними документації виробника, описано регістри спеціальних функцій та загального призначення.

Проведено ініціалізацію мікроконтролера, його вузлів та модулів.

Для забезпечення роботи периферійних пристроїв здійснено керування, прийом та передача даних з пристроїв введення та до пристроїв виведення згідно варіанту.

У робочій частині забезпечено функціонування МКС згідно варіанту.

Курсовий проєкт може бути основою розробки керуючої системи збору даних чи системи керування та контролю за виконанням завдань.

ПЕРЕЛІК ПОСИЛАНЬ

1. Войтович І. В., Сидоренко В. М. Мікропроцесорні системи та мікроконтролери. — Київ: КНТ, 2020.
2. Мазур О. М. Мікроконтролери AVR: програмування мовою С. — Львів: Львівська політехніка, 2018.
3. Погорілий Ю. С. Вбудовані системи на базі мікроконтролерів. — Київ: КПІ ім. І. Сікорського, 2021.
4. Barrett S. F., Pack D. J. Atmel AVR Microcontroller Primer: Programming and Interfacing. — Morgan & Claypool Publishers, 2019.
5. Han-Way Huang. The Atmel AVR Microcontroller: MEGA and XMEGA in Assembly and C. — Cengage Learning, 2017.
6. Jack Ganssle. The Art of Designing Embedded Systems. — Newnes, 2020.
7. Barnett R. H., Cox S., O’Cull L. Embedded C Programming and the Atmel AVR. — Cengage Learning, 2018.
8. Martin Bates. PIC Microcontrollers: An Introduction to Microelectronics. — Newnes, 2022.