AVR242: 8-бітний мікроконтролер динамічно керує світлодіодами і клавіатурою 4 x 4

Можливості:

- **Опитування стану 16 кнопок (панель), в матриці 4 x 4**
- **Керування 4-розрядним LED дисплеєм з відображення двох точок**
- **Індустріальний годинник/таймер реального часу**
- **Звукове підтвердження натискання кнопок за допомогою п'єзо-резонатора**
- **Мигання дисплею для індикації пониження напруги живлення**
- **Подвійна функція I/O виводів**
- **Мінімум зовнішніх компонент**
- **Ефективний код**
- **Вміщує повну програму для AT90S1200**
- **Підходить для будь-якого AVR MCU з 20 і більше виводами**

Вступ

Ця примітка по практичному застосуванню описує завершену систему, що забезпечує клавіатуру 4 x 4, як вхідну, для годинника/таймера реального часу з двома виводами. Ця система керує зовнішніми навантаженнями і чотирьох розрядним світлодіодним дисплеєм. Ця примітка розроблена для того, щоб показати універсальність конфігурації портів AVR і ефективність широкого набору інструкцій. Дана реалізація може працювати з будь-яким AVR з 20 і більше виводами, хоча потрібно буде звернути увагу на ініціалізацію стеку і різницю в таблицях регістрів. Структура програми оптимізована під трьохрівневий апаратний стек AT90S1200 і може бути покращена для інших AVR з програмним стеком.

Теорія роботи

Для підключення клавіатури 4 x 4, **п'єзо-резонатора, двох LED навантажень і чотирьохрозрядного дисплею звичайно потрібно двадцять три лінії вводу/виводу. Дана реалізація показує як можна, використавши деяку "хитрість", зменшити дане число до п'ятнадцяти, що дозволяє застосувати менший 20-вивідний AVR. Принципова схема повністю наведена на рис. 2, за виключення компонентів генерації тактової частоти, котрі опущені для більшої ясності.**

Чотири стовбці клавіатури під'єднані до нижнього півбайта порту B і чотири рядки клавіатури під'єднані до старшого півбайта. Ті самі вісім біт також безпосередньо керуються катодними сегментами чотирьохрозрядного LED дисплею з допомогою струмообмежуючих резисторів R13 – R20. Ці виводи таким чином виконують подвійну функцію, активуються як вихідні, коли керують LED дисплеєм, і вводу/виводу, коли сканують клавіатуру. Це робить довершеним використання програмного середовища і надає великі поточні можливості керування AVR портами з досягненням хорошого ефекту.

Більшість часу порти B споживають струм в 9 мА, напряму керуючи LED сегментами. Кожна цифра послідовно вмикається на проміжок часу в 5 мс, а динамічне керування дисплеєм відбувається з допомогою PNP транзисторів Q1 – Q4. Загальні аноди для розрядів LED дисплеїв керуються з допомогою PNP транзисторів, аноди споживають максимально можливий струм 72 мА (9 мА – 8 сегментів), що виходить за межі робочих можливостей портів.

8-бітний мікроконтролер динамічно керує світлодіодами і клавіатурою 4 x 4

Примітка по практичному використанню

Це може бути будь-який тип PNP, що може керувати струмом в 100 мА чи біля того (наприклад BC479). Можливо підключити паралельно два виводи портів для кожного аноду, щоб розподілити струм, але тоді загальне число I/O виводів, що використовується, зросте, що викличе потребу в застосуванні більшого МК.

Перед кожним циклом відображення дисплею, конфігурація портів змінюється для надання чотирьох входів з ввімкненими внутрішніми "підтягуючими" резисторами і чотирьох виходів "низького" рівня для сканування клавіатури. Якщо натиснута кнопка, то зчитується конфігурація півбайту для обрахування стану клавіатури зі збереженням номеру кнопки в змінній. Коротка пауза між кожною зміною стану портів дозволяє порту встигнути перестроїтись. Цей метод програмно більш ефективний для даної реалізації, чим традиційний метод порядкового сканування.

Загальні аноди вимкнуті впродовж цього часу для запобігання спотворенням відображаємої інформації. Тоді відновлюється конфігурація портів для динамічної індикації. Потім основні службові функції використовують змінну з номером натиснутої кнопки для виконання відповідних дій.

Годинник реального часу керується перериваннями, використовуючи Таймер 0, який працює на системній тактовій частоті з попереднім діленням на 256. Таймер попередньо програмується числом 176 і переривання по переповненню відбуваються кожні п'ять мілісекунд, гарантуючи високу точність, якщо використовується високоякісний кварцовий резонатор. Для високої точності застосовують кварцовий резонатор на 4,096 МГц. Програма може бути змінена для використання кристалу на 4,00 МГц з незначними модифікаціями.

Програма обслуговування переривань перезавантажує таймер і збільшує значення трьох змінних: змінну лічильника (tock), змінну, що відповідає за нейтралізацію "дренькоту" кнопки (bounce) і лічильник, що підраховує секунди (second). Остання змінна використовується основними службовими функціями для корекції значень хвилин і годин, котрі в свою чергу відображаються на дисплеї функціями обслуговування дисплею.

Службові функції контролюють дві лінії навантаження, як для часу ввімкнення, так і для вимкнення, і керують виходами, як відповідним старшим півбайтом порту D. В даній реалізації, навантаження імітуються червоним і зеленим світлодіодами, що керуються вхідним (активний стан – "низький") струмом. Це можна замінити керованим реле чи оптоізольованим тиристором, щоб підключити потужні навантаження.

Клавіатура забезпечує можливість встановлення (SET) значень часу і час ввімкнення/вимкнення для кожного навантаження та також дає можливість одразу вимикати (CLEAR) навантаження. А пьезо-резонатор під'єднаний до старшого біту порту D, забезпечує звукове підтвердження натискання кнопки.

Використання виводів порту B потребує деяких застережних заходів. Оскільки виводи використовуються для двох функцій, дуже важливим є те, щоб при натисканні кнопки, вони не шунтували вихід дисплею. Це досягається встановленням струмообмежуючих резисторів послідовно з кожною кнопкою. Коли виводи застосовуються як вхідні, то використання внутрішніх "підтягуючих" резисторів зменшує кількість зовнішніх компонент. Вибір значень резисторів (R1-8) такий, що незначно впливає на напругу утворену резистивним дільником. З вибраними значеннями резисторів і живленням 5 В, логічні рівні будуть складати 0,6 В для логічного "0" і 4,95 В для логічної "1". Резистори R21 і R22 – звичайні струмообмежуючі резистори для світлодіодів і їх значення може бути підібране для будь-якого живлення. Дана примітка була протестована з використання 330 Ом і 5 В живлення. Світлодіоди керуються режимом вхідного струму ("0" = ON) і живляться від прямого струму в 9 мА у відповідності з визначеними параметрами.

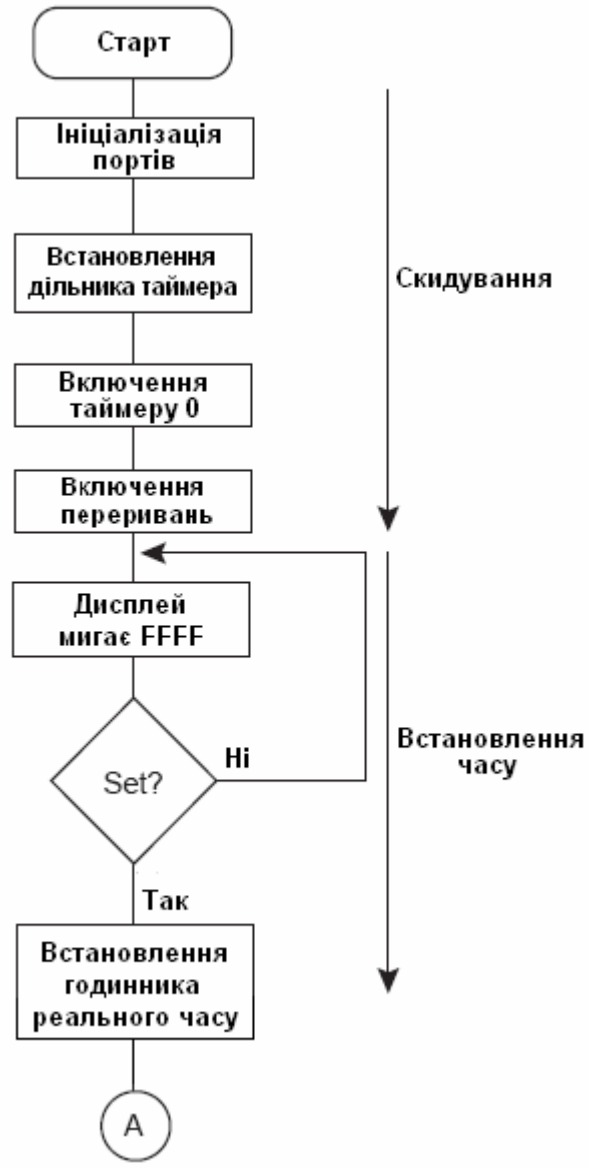
Реалізація

Програмне забезпечення включає дві основні частини: фонові функції, які керовані перериваннями і забезпечують точність часу і функції основного процесу. Вони складаються з трьох розділів: програма скидування, котра встановлює порти, таймер і переривання, програма встановлення часу та основні службові функції.

Основний процес

Основний процес протікає більшість часу, тільки переривається на 5,127 мкс (21 цикл) кожні 5 мс для обновлення значень реального часу годинника. Він складається з трьох частин: RESET, TIME SETTING і HOUSEKEEPING. Блок-схема показана на рис. 1.

Рисунок 1. Блок-схема основного процесу (Частина 1) Продовження на рис. 3.



Розділ Reset

При подачі живлення чи в стані скидування, програма скидування входить в режим ініціалізації потрібного апаратного забезпечення. Порти ініціалізуються з вибором напрямку їх роботи і всі виводи встановлюються у “високий” стан для відключення будь-якого навантаження. Всі виводи встановлюються як вихідні, що потребує завантаження 255 у регістр напрямку даних для обох портів. Напрямок роботи для порту B буде змінено через деякий час, коли буде виконуватись функція сканування клавіатури. Дільник таймера встановлюється на ділення тактової частоти на 256, щоб отримати переривання періодом 5 мс, що досягається записом 176 в таймер. Тоді переривання таймера по переповненню буде ввімкнене по наступному Глобальному перериванню.

Рівняння, що зв’язує період переривань з частотою 4,096 МГц, забезпечує час циклу виконання інструкції в 0,2441 мікросекунди. Число n, що записується в регістр TCNT0 таймеру 0, тоді буде таким:

(256-n)*256*0,2441 мікросекунди.

Значення 176 надає точні інтервали в 5 мс, гарантуючи високу точність годинника реального часу.

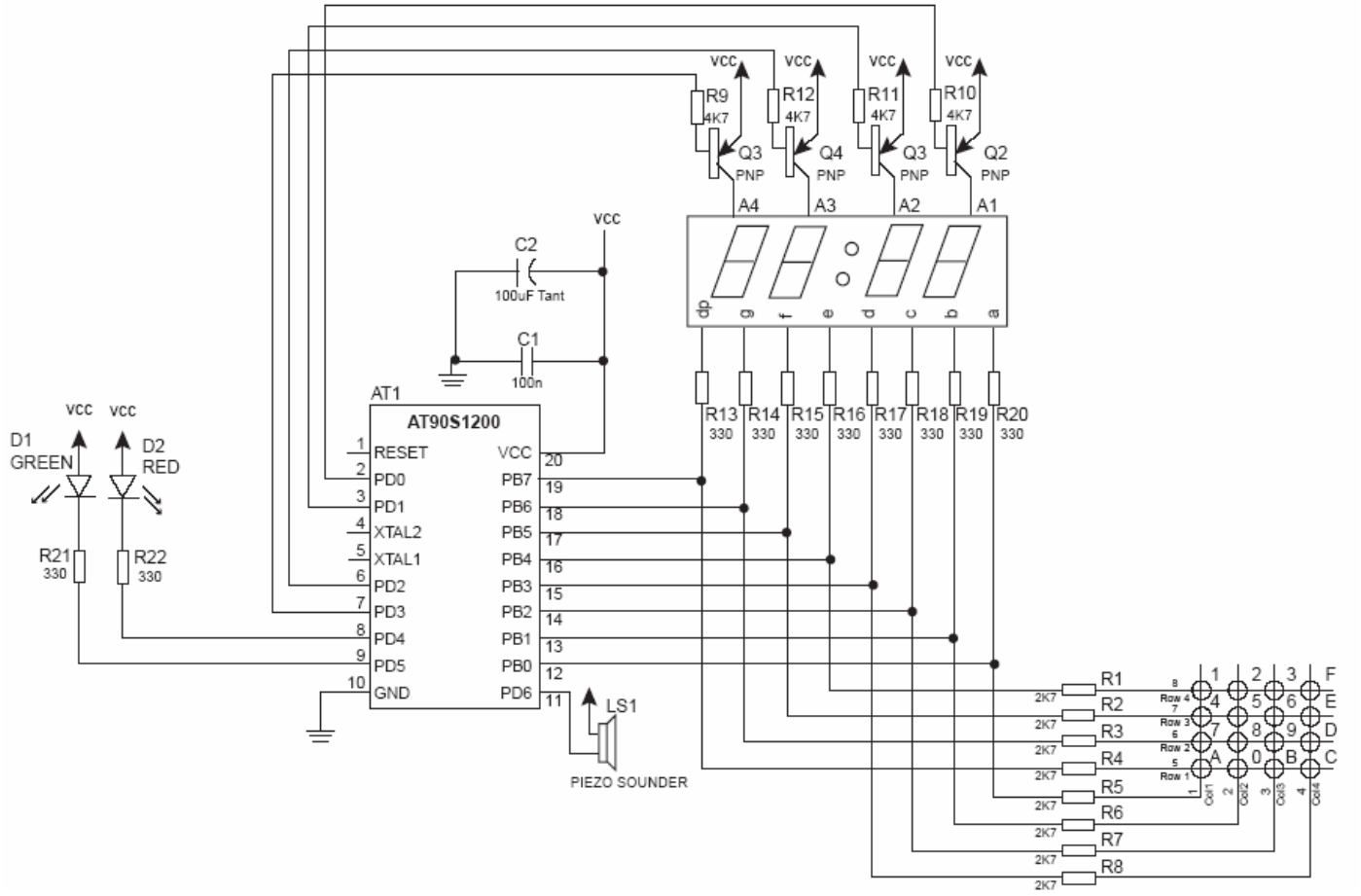
Встановлення часу (Time Setting)

Світлодіодний дисплей тепер буде мигати EEEE того, що час неправильний і потрібне перезавантаження часу. Це буде продовжуватись, поки на клавіатурі не буде нажата кнопка SET. Це викличе функцію “setrtc”, котра керує входами клавіатури і встановлює зворотній зв'язок з дисплеєм. Тільки після закінчення перезавантаження значень часу, основні службові функції обновлюються і керують дисплеєм з допомогою основної змінної “second”та сканують клавіатуру на команди.

Службові функції

Основні службові функції працюють на оновлення змінних часу, як функції неосновного процесу, і керують світлодіодним дисплеєм, коректуючи значення часу. Також сканується клавіатура на вхідні команди і перевіряється час ввімкнення/вимкнення навантажень. Блок-схема показана на рис. 3.

Рисунок 2. Принципова схема для кнопок/дисплею



Секунди, збільшуються програмою обслуговування переривання і порівнюються з 60. Якщо досягнуто 60 секунд, то змінна хвилин збільшується і секунди скидаються в нуль. Подібна процедура реалізована для годин, коли змінна хвилин порівнюється з 60 і змінна годин відповідно збільшується. Змінна годин в свою чергу порівнюється з 24, починаючи новий день, і все від годин до секунд обнуляється.

Для збереження з використанням RAM масиву, змінні хвилин і годин мають бути обмежені одним байтом кожна. Нижня тетрада міститиме цифру меншого розряду, старша тетрада – цифру старшого. Це означає, що ці байти мусять бути оброблені як BCD (двійково-десяткові) і мають властивість виявляти помилки, що забезпечує корекцію відліку часу. Тому байт хвилин чи годин повинен розбиватися на тетради і перевірятися на розмір при кожні зміні.

Якщо немає зміни у значеннях хвилин чи годин, під час будь-якої перевірки, то наступна частина програми пропускається і відображається час.

Годинник має 24 часовий формат і відповідно мусить починати новий день, коли час стає більший чим 23:59. Програма роботи з дисплеєм – це функція “display”, що викликається, яка також включає в себе програму роботи з кнопками. Ця функція буде розглянута пізніше.

При виході з функції дисплею перевіряється натискання кнопок, тому наступною дією буде ввімкнення/вимкнення часу навантаження і будь-яка відповідна інша дія перед повторенням службового циклу. Наприклад, якщо задати час роботи для першого навантаження, то воно одразу включиться, коли співпаде з часом на дисплеї.

Змінна “flag” використовується для збереження одиночних бітів, що показують виконання певної дії. Це використовується для пасивного контролю однієї функції іншою. Для цієї реалізації потрібно дев’ять прапорців (флагів), котрі мають на одне значення більше чим є в одному байті. Для збереження і невикористання інших регістрів для цього одного біту, буде застосований

прапорець “T” в статусному регістрі, як дев’ятий біт. Це можливо тому, що для перевірки цього прапорця застосовуються специфічні команди розгалуження (BRTC і BRTS), що легко програмується, а інструкції SBRS і SBRC використовуються для перевірки основного “flag” біта. Значення прапорців при високому рівні (одиниці) і їх розміщення показані нижче в таблиці 1, разом з їх функціями. Час, впродовж якого працює основний цикл, не впливає на точність RTC, оскільки годинником реального часу керують переривання, які можуть переривають основний цикл чотири рази за час його роботи.

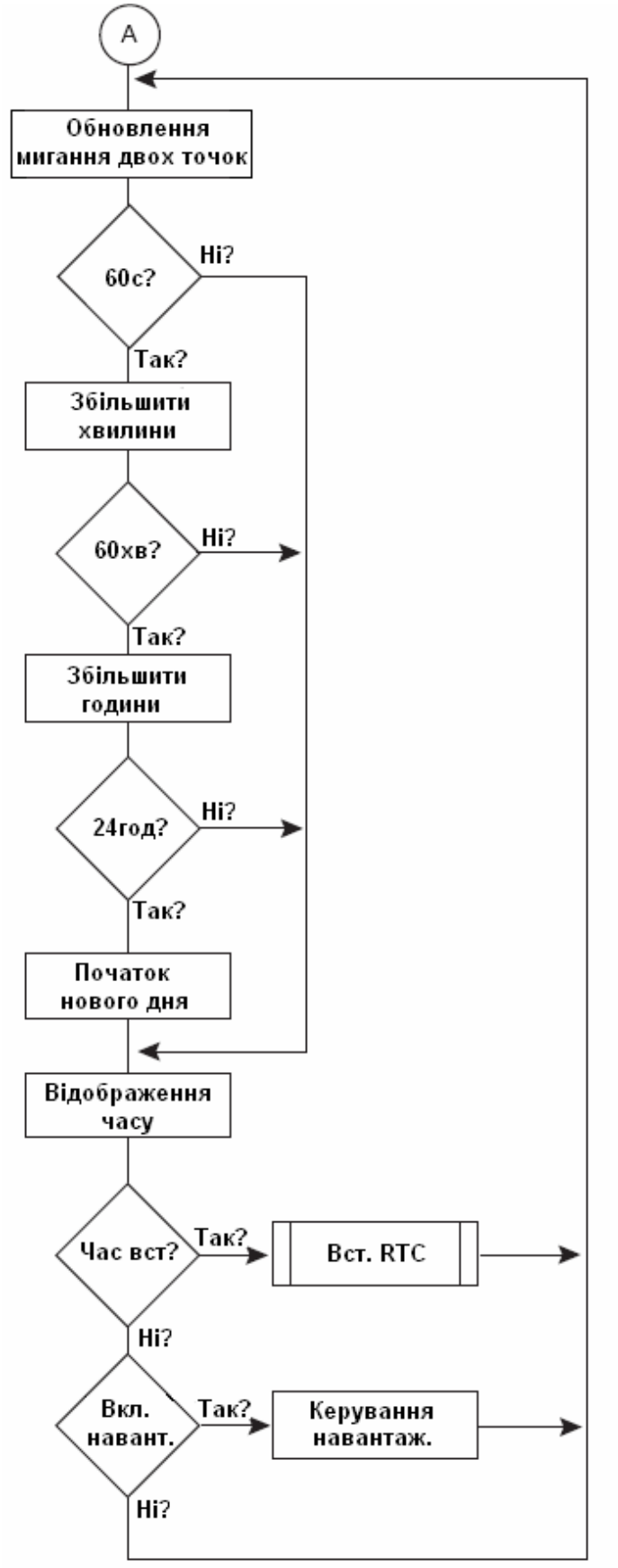
Таблиця 1. Значення використовуваних прапорців

Номер біту в “Flag”	Функція
0	Навант. 1 активне
1	Навант. 2 активне
2	Навант. 1 ввімкнути
3	Навант. 1 вимкнути
4	Навант. 2 ввімкнути
5	Навант. 2 вимкнути
6	Була нажата кнопка ОК (з усуненням “дребезга”)
7	Інтервал роботи 5 мс
Статус T прапорця	Установка часу

Центральні сегменти двох точок (dp) мигають з інтервалом в півсекунди, використовуючи змінну “blink”, яка інкрементується неосновним процесом переривань. Це використовується для переключення змінної “flash”, яка застосовується як маска для функції дисплею. Програма перевірки навантажень працює більш комплексно, чим показує одиничний блок на блок-схемі, тестуючи змінні контролюючі біти в байті “flag” і виконує відповідні дії. Включення цього в блок схему створило би труднощі в її поясненні в подальшому.

Якщо була вибрана команда “set load”, вона викликає функцію “setrtc”, яка встановлює час ввімкнення чи вимкнення навантаження відповідно з нажатою кнопкою. Той самий метод підтвердження миганням застосований тут, тільки тепер дисплей мигає тим сегментом, в якому вводиться цифра, і час спрацювання задається від вищого розряду до нижчого. Тоді користувач буде впевнений, яке число введено.

Рисунок 3. Блок-схема основного процесу



Команда CLEAR виключить обидва навантаження, напряму скасовуючи будь-які попередні команди ввімкнення/вимкнення. Цей процес не матиме ніякого впливу на годинник реального часу, оскільки корекція часу проходить у фоновому процесі. Значення RTC також може бути змінено, оновлено значення часу чи будь-яке інше значення, тим самим процесом.

Функції дисплею

Блок-схема показана на Рисунку 5. Ця функція викликається для мигання дисплеєм програмою скидування МК, функцією “setrtc” та службовими функціями, і вона обслуговує сканування кнопок та динамічне керування дисплеєм. Якщо застосувати більш потужний AVR, то він може за допомогою окремої функції керувати цифровими сегментами і викликати їх більше чотирьох разів. Це неможливо з використанням AT90S1200 тому, що він має стек глибиною всього в 3 рівні.

Перший блок (в блок-схемі) вимикає аноди дисплею і тоді сканує кнопки. Це досягається зміною конфігурації PORTB, де рядковий півбайт вмикається як вхідний і півбайт колонки як виходи. Внутрішні “підтягуючі” резистори також підключені до чотирьох входів. Всі чотири біти колонки мають низький вихідний рівень (“0”) і входи рядка зчитують значення з PINB. Це також генерує базове число, яке зберігається у змінній “key” значенням 0, 4, 8 чи 12 в залежності від натиснутої кнопки або число 0x10, якщо жодна кнопка не була нажата.

Далі конфігурація порту змінюється на протилежну, щоб рядковий півбайт став вихідним, а півбайт колонки почав працювати, як вхідний і біти рядку встановлюються в низький логічний рівень. Після короткого проміжку часу входи колонки зчитують значення (0, 1, 2 чи 3) з PINB і використовуються для сумування з базовим числом, в залежності від натиснутої кнопки на входах колонки. Кінцевий результат у вигляді числа зберігається в змінній “key”, яка використовується як індекс для пошуку значення потрібної кнопки в таблиці, що зберігається в EEPROM. Правильне значення кнопки записується назад у “key” і застосовується для виклику відповідної функції. Такий алгоритм необхідний тому, що кнопки не розміщені у логічній послідовності. Алгоритм також надає велику гнучкість у застосуванні для програміста. Розміщення кнопок і функцій показано на Рисунку 4.

Рисунок 4. Розміщення кнопок і функції

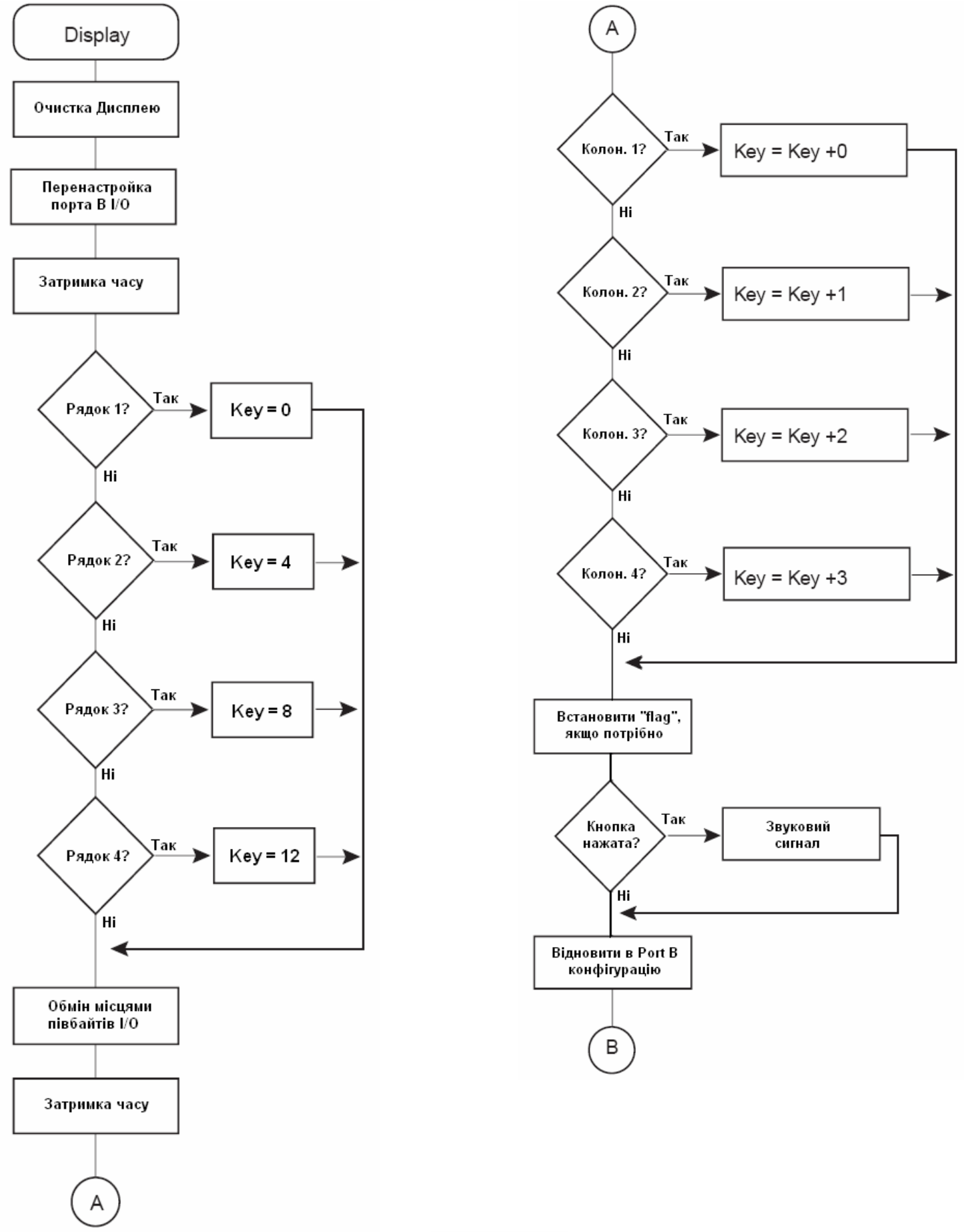
1 #1	2 #2	3 #3	F Load 1 ON
4 #4	5 #5	6 #6	E Load 1 OFF
7 #7	8 #8	9 #9	D Load 2 ON
A SetRTC	0 #0	B Clear	C Load 2 OFF

Коли значення кнопок більше 9, це відслідковується і застосовується для встановлення відповідних бітів у змінній “flag”, яка використовується для виклику функцій. Якщо значення кнопок визначене як 0x10, то жодна кнопка не була натиснута.

Якщо була нажата кнопка, звучить короткий сигнал (beep), за допомогою п'єзорезонатора підключеного до 6 біту PORTD, для встановлення оберненого зв’язку з користувачем.

Цифри, в свою чергу, тоді з допомогою динамічної індикації відображаються з інтервалом в 5 мс, який встановлюється фоновим процесом. Це забезпечує обновлення дисплею з частотою в 50 Гц, достатню яскравість та відсутність мерехтіння (за виключенням короткого проміжку часу на сканування кнопок).

Рисунок 5. Блок-схема сканування кнопок функції “display”

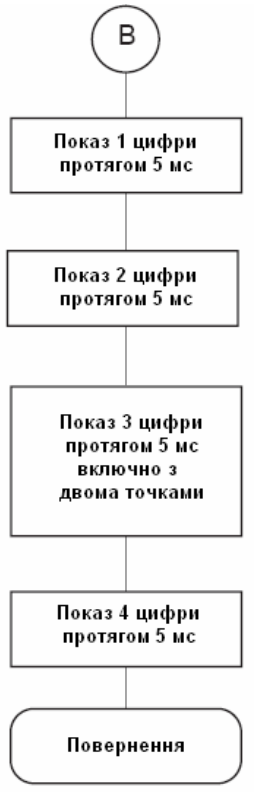


Для відображення (для декодування в код 7-сегментного індикатора) кожної цифри використовується пошук у таблиці, збереженій в EEPROM, збереження відповідного індексу в регістрі “temp” і застосування його, як доступу до потрібного, для запалення цього символу, байта. Декілька спеціальних символів використовується при роботі з кнопками, щоб показати якусь важливу інформацію. Для прикладу, літера “E” визначена для сигналізації про похибку відображення дисплею одразу після подачі живлення, літери “o”, “n” і “f” визначенні для показу станів ввімкнення/вимкнення навантажень. Якщо ви застосуєте більш потужний AVR для вашої схеми, то можете при бажання перенести таблицю значень в ROM і здійснювати адресну індексацію.

Далі йде блок, де мигання двох точок відбувається кожні півсекунди і це змінює маску “flash”, яка використовується в попередньому процесі роботи дисплею, це мигання двох точок по центру сигналізує про коректну роботу функції годинника.

Потім програма повертається до виклику функції, яка зберігає значення натиснутої кнопки в “key”.

Рисунок 5. Блок-схема, що відповідає за відображення на дисплеї функції “Display”



Функція Setrtc

Блок-схема показана на рисунку 7. Ця функція викликається всіма підпрограмами, які обслуговують вхідні дані з кнопок для виводу значень на дисплей. Це відбувається при подачі живлення чи перезавантаженні для введення реальних значень часу, або при натисканні кнопки SET для зміни значень часу, або при натисканні будь-якої з чотирьох кнопок для зміни установок навантажень. Це викликає функцію дисплею для визначення натиснутої кнопки і відображення відповідних цифр. Функція використовує лічильник “bounce”, який збільшується кожні 5 мс фоновою програмою переривання, для забезпечення прийнятного реагування на спрацювання кнопки (подавлення “дребезга” контактів).

Функція складається з чотирьох фаз, починається з вводу першої цифри і триває поки не будуть введені всі, дисплей у цей час відображає миганням “n” кожну цифру, значення якої мають ввести з допомогою клавіатури. Значення, які виходять за межі діапазону (годин, хвилин) ігноруються, і введення триває до тих пір, поки е буде введене значення в межах діапазону.

Коли всі чотири цифри будуть коректно введені, функція зберігає значення годин у змінній “hiset” і хвилини у змінній “loset”. Їх значення переписуються у відповідні змінні, з викликом функції “setrtc”, коли її використовують фонові службові функції.

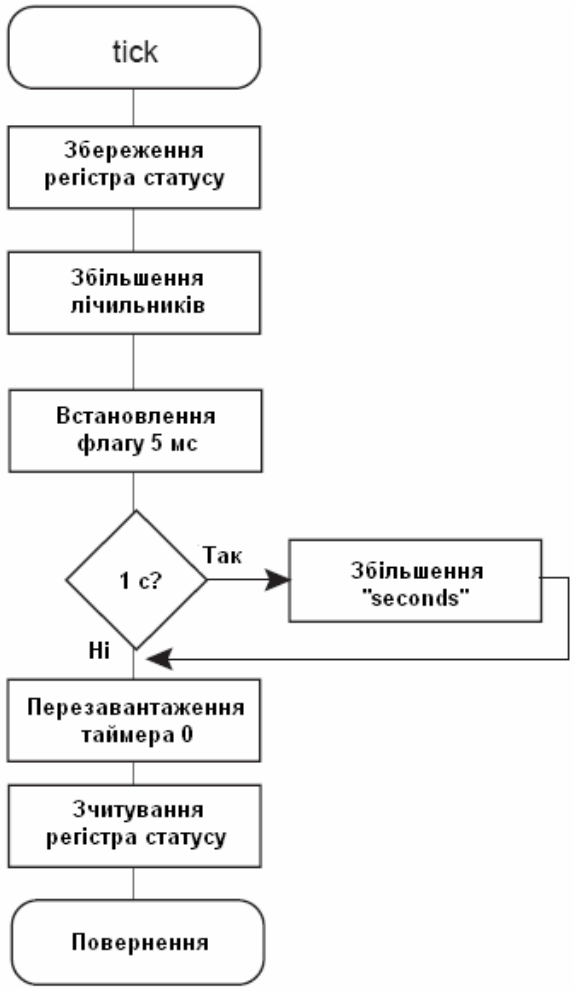
Рисунок 7. Блок-схема для “setrtc” функції

Фонові функції (час тактування)

Функції запускаються кожні 5 мс по переповненню таймера 0 і переривання стає основною функцією в будь-якій точці циклу. Програма послідовно зберігає статусний регістр при вході в переривання і зчитує його при виході, як основні установки роботи, для запобігання порушенню роботи фонового процесу. Використання регістру “temp” теж не застосовується по тій самій причині.

Функція дуже проста і послідовно збільшує три регістри при кожному перериванні, встановлює флаг 5 мс інтервалу, що використовується програмою обслуговування дисплею, перезавантажує таймер 0 та інкрементує регістр секунд RTC, якщо потрібно. Блок-схема показана на Рисунку 8.

Рисунок8. Блок-схема “тактування” фоновими функціями



Використані ресурси

Таблиця 2. Задіяння CPU і пам'яті

Функція	Розмір коду	Цикли	Використання регістрів	Переривання	Опис
Reset	17 слів	17 цикл	R16, R31	-	Ініціалізація
Timesetting	9 слів	14 цикл	R1, R2, R18, R19, R24, R25	-	Встановлення значень RTC
Housekeeping	97 слів	52 типових	R1, R2, R16, R17, R18, R19, R20, R21, R24, R25, R28	-	Основні фонові функції підтримують відображення часу на дисплеї, реагують на кнопки та керують навантаженнями.
Display	158 слів	150 типових	R16, R17, R20, R21, R23, R24, R25, R26, R28	-	Сканування кнопок і функції дисплею
Setrtc	47 слів	45 типових	R1, R2, R16, R20, R22, R24, R25, R26, R28	-	Функція введення часу і установок навантажень за допомогою кнопок
tick	15 слів	21 цикл	R0, R31	Таймер 0	Фонова функція обслуговування переривання для забезпечення інтервалу в 5 мс і 1 с відповідно
Total	343 слова	-	R0, R1, R2, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R28, R31	Таймер 0	

Таблиця 3. Задіяна периферія

Периферія	Опис	Переривання
Таймер 0	Лічильник тактування 5 мс	Переповнення таймера 0 з передподільником на 256
16 байт EEPROM	Значення кнопки декодується в відображення семисегментного індикатора	-
8 виводів вводу/виводу порту B	Підключення клавіатури 4x4 і керування світлодіодними сегментами (подвійна функція)	-
3 виводів вводу/виводу порту D	1 і 2 навантаження та п'єзорезонатор	-
4 виводів вводу/виводу порту D	Керування анодами 4-цифрового світлодіодного дисплею	-

```
;**** Примітка по практичному використанню AVR242 ***********************
;*
;* Назва: Просте динамічне керування світлодіодним дисплеєм і клавіатурою 4x4
;* Версія: 1.0
;* Останні зміни: 98.07.24
;* Об’єкт: Всі AVR пристрої
;*
;* Технічна підтримка E-mail:avr@atmel.com
;*
;* ОПИС
;* Дана примітка по практичному використанню розповідає про програму, що забезпечує функціонування 24-годинного
;* індустріального таймеру чи годинника реального часу, використовуючи виводи I/O з подвійною функцією.
;* Разом з вхідною матрицею 4 x 4 клавіатури, виходи динамічно
;* керують чотирьохцифровим LED дисплеєм і ввімкненням/вимкненням виходів для керування навантаженнями з
;* застосуванням додаткової електричної схеми зовнішнього інтерфейсу. Для прикладу, в якості навантажень
;* світлодіоди, але можуть бути включені будь-які навантаження з використанням відповідних компонентів.
;* Зворотній зв'язок при натискання кожної з кнопок забезпечується з допомогою п’єзорезонатора, котрий подає
;* сигнал при натисканні кнопки.
;* Складається з основної програми, що дозволяє задавати значення часу з допомогою клавіатури
;* і по одному часу ввімкнення/вимкнення впродовж 24 годин для кожного навантаження, функцій для
;* годинника реального часу, сканування кнопок та оброблення переривань. Для прикладу використано
;* AT90S1200, щоб продемонструвати як можна застосувати обмежене число виводів I/O, але можна
;* використати будь-який AVR, з відповідними змінами у векторах переривань, EEPROM і покажчика стеку.
;* Тактування здійснюється 4.096 MHz кристалом (4 MHz кристал буде породжувати
;* похибку в -0.16%, якщо 178 буде записано замість 176 при ініціалізації таймера, але це
;* може бути програмно усунено при регулярних інтервалах тактування). Використані таблиці
;* записані в EEPROM для декодування даних для виводу на дисплей, з додатковими символами, що застосовуються
;* для відображення значень часу і значень ON/OFF навантаження на дисплеї та конвертації значень матриці кнопок.
;* Якщо EEPROM потрібне для інших цілей, таблиці можуть бути переміщені
;* в ROM для більших AVR пристроїв.
;***************************************************************************
;***** Регістри, що використовуються всіма програмами
;******Глобальні змінні, що використовуються підпрограмами
.def loset     =r1                      ;для збереження значення хвилин
.def hiset     =r2                      ;для збереження значення годин
.def ld1minon  =r3                      ;для збереження часу ввімкнення і вимкнення навантажень
.def ld1hron   =r4                      ;установка входів клавіатури
.def ld1minoff =r5                      ;і тестування в основних фонових функціях
.def ld1hroff  =r6                      ;і збереження часу ввімкнення і вимкнення навантажень
.def ld2minon  =r7
.def ld2hron   =r8
.def ld2minoff =r9
.def ld2hroff =r10
.def temp      =r16                     ;основна тимчасова змінна
.def second    =r17                     ;збереження лічильника для RTC секунд
.def minute    =r18                     ;збереження лічильника для RTC хвилин
.def hour      =r19                     ;збереження лічильника для RTC годин
.def mask      =r20                     ;flash маска для відображення цифр (мигання)
.def blink     =r21                     ;лічильник швидкості мигання колонки
.def bounce    =r22                     ;лічильник для клавіатури (подавлення дребезгу)
.def flash     =r23                     ;лічильник затримки мигання
.def lobyte    =r24                     ;збереження цифр хвилин для функції дисплею
.def hibyte    =r25                     ;збереження цифр годин для функції дисплею
.def key =r26                           ;число кнопки для сканування
;***'key' значення, що зчитується 'keyscan'**************************
;VALUE  0 1 2   3   4 5 6    7   8 9 10 11    12 13  14     15 16
;KEY    1 2 3   F   5 6 E    7   8 9 D   A    0   B   C    NONE
;FUNC   1 2 3 LD1ON 4 5 6 LD1OFF 7 8 9 LD2ON SET  0 CLEAR LD2OFF
.deftock=r27                            ;5 мс тікер
.defflags=r28                           ;байт флагу для команди кнопки з клавіатури
                        ;  7    6      5     4     3      2    1   0
                        ; 5ms keyok ld2off ld2on ld1off ld1on ld2 ld1
                        ; tick 0 = off, 1 = on
.equ ms5    =7                          ;тактування в 5 мс інтервали для часу оновлення дисплею
.equ keyok  =6                          ;встановлюємо коли кнопка нажата, змінна повинна бути знову очищена
.equ ld2off =5                          ;встановлюємо для навантаження стан ON/OFF від нажатої кнопки і флаги
.equ ld2on  =4                          ;якщо потрібно, то вмикаємо
.equ ld1off =3                          ;в фонових службових функціях
.equ ld1on  =2
.equ ld2    =1                          ;коли встановили, то фонова службова програма
.equ ld1    =0                          ;перевіряє час on/off навантажень
;*** T флаг в статусному регістрі використовується як SET флаг встановлення часу
.equ clear =0                           ;RTC завдання потрібних флагів
    ;Конфігурація виводів порту B
.equ col1 =0                            ;LED a сегмент/кнопка 1 колонки клавіатури
.equ col2 =1                            ;LED b сегмент/кнопка 2 колонки клавіатури
.equ col3 =2                            ;LED c сегмент/кнопка 3 колонки клавіатури
.equ col4 =3                            ;LED d сегмент/кнопка 4 колонки клавіатури
.equ row1 =4                            ;LED e сегмент/кнопка 1 рядка клавіатури
.equ row2 =5                            ;LED f сегмент/кнопка 2 рядка клавіатури
.equ row3 =6                            ;LED g сегмент/кнопка 3 рядка клавіатури
.equ row4 =7                            ;LED десяткова точка/кнопка 4 рядка клавіатури
    ;Конфігурація виводів порту D
.equ A1 =0                              ;керування загальним анодом (активний рівень - низький)
.equ A2 =1 ;
.equ A3 =2 ;
```

```
.equ A4 =3 ;
.equ LOAD1 =4                           ;навантаження 1 на вихід (активний рівень - низький)
.equ LOAD2 =5                           ;навантаження 2 на вихід (активний рівень - низький)
.equ PZ =6                              ;п'єзорезонатор на вихід (активний рівень - низький)
.include "1200def.inc"

;***** Регістри, які використовуються програмою обслуговування переривання по переповненню таймера

.def timer =r31                         ;вільна змінна для завантаження таймера
.def status =r0                         ;перший регістр для збереження статусного регістру

;***** Таблиця значень для декодування LED дисплею **********************

.eseg ;EEPROM сегмент
.org 0

table1:
.db      0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
;digit     0    1    2   3     4    5    6    7    8    9

.db      0x86,0x8E,0xA3,0xAB,0XFF,0XFF
;digit     E    f    o   n        Пропуск   спеціальні символи

;**** Таблиця значень для перетворення значень кнопок в використовувані числа****

  ;key1 2 3 F 4 5 6 E 7 8 9 D A 0 B C
table2:
  .db       1, 2, 3, 15, 4, 5, 6, 14, 7, 8, 9, 13, 10, 0, 11, 12
  ;значення 0  1  2   3  4  5  6   7  8  9  10 11  12  13 14  15

;****Початковий код*****************************************************
.cseg                                   ;CODE сегмент
.org 0
      rjmp reset                        ;Reset інструкція
      nop                               ;незадіяне зовнішнє переривання
      rjmp tick                         ;лічильник переповнення таймера (5 мс)
      nop                               ;незадіяне аналогове переривання

;*** Reset інструкція ************************************************
;*** для забезпечення ініціалізації порту, таймера і встановлення переривань

reset:
   ser temp                             ;
   out DDRB,temp                        ;ініціалізація порту B як всі виходи
   out DDRD,temp                        ;ініціалізація порту D як всі виходи
   out PORTB,temp                       ;колонка клавіатури на високий рівень/світлодіоди вимкнені
   out PORTD,temp                       ;вимикаємо світлодіоди і навантаження
   ldi temp,0x04                        ;дільник таймера/256
   out TCCR0,temp
   ldi timer,176                        ;завантажуємо таймер на 5 мс
   out TCNT0,timer                      ;(256 - n)*256*0.2441 мкс
   ldi temp,0x02                        ;дозволяємо переривання таймеру
   out TIMSK,temp
   clr flags                            ;очищуємо контрольні флаги
   clr tock                             ;очищуємо 5 мс тікер
   clr bounce                           ;очищуємо лічильник натискання кнопок (дребезгу)
   clr flash
   clr blink
   sei                                  ;глобальний дозвіл переривань

;****Мигаємо EEEE на LEDS як тест і сигнал, що було подане живлення**************
;****повторюємо доки кнопка SET не буде натиснута на клавіатурі

timesetting:
   ldi hibyte,0xaa                      ;показуємо "EEEE" на LED
   ldi lobyte,0xaa                      ;дисплеї і
   ser mask                             ;встановлюємо мигання дисплею
notyet:
   rcall display                        ;дисплей мигатиме весь час
   brtc notyet                          ;доки не буде натиснута кнопка SET
   rcall setrtc                         ;і перезавантаження часу
   mov hour,hiset                       ;і введення значень годин
   mov minute,loset                     ;і хвилин
   clt                                  ;очищаємо T флаг

;*****Основні службові функції тактування*****************************

do:
   clr mask                             ;виконуємо службову функцію
   cpi blink,100                        ;якщо було досягнуто півсекунди
   brne nohalf
   clr blink
   com flash                            ;інвертуємо flash
nohalf:
   cpi second,60                        ;пройшла одна хвилина?
   brne nochange                        ;ні
   clr second                           ;так, очищаємо секунди і
   inc minute                           ;додаємо одну хвилину
```

```
   mov temp,minute
   andi temp,0x0f                    ;маска десяткового розряду хвилин
   cpi temp,10                       ;пройшло десять хвилин?
   brne nochange                     ;ні
   andi minute,0xf0                  ;так, очищаємо нижчий розряд хвилин
   ldi temp,0x10
   add minute,temp                   ;збільшуємо старший (десятковий) розряд хвилин
   cpi minute,0x60                   ;пройшло 60 хвилин?
   brne nochange                     ;ні
   clr minute                        ;так, очищаємо хвилини і
   inc hour                          ;додаємо одну годину
   mov temp,hour
   andi temp,0x0f                    ;маска старшого розряду годин
   cpi temp,10                       ;пройшло 10 годин?
   brne nochange                     ;ні
   andi hour,0xf0                    ;так, збільшуємо
   ldi temp,0x10
   add hour,temp                     ;десятковий розряд значення годин

nochange:
   cpi hour,0x24                     ;пройшло 24 години?
   brne sameday                      ;ні,
   clr hour                          ;так, очищаємо змінні часу
   clr minute                        ;для початку нового дня
   clr second
sameday:                             ;обновлення значень часу
   mov lobyte,minute
   mov hibyte,hour
   rcall display                     ;показуємо час протягом 20 мс
   brtc case1                        ;якщо не було натиснуто SET
   rcall setrtc                      ;інакше скидуємо час
   mov hour,hiset                    ;перезавантажуємо години
   mov minute,loset                  ;і хвилини
   clt                               ;очищуємо T флаг
case1:sbrc flags,ld1                 ; 1 навантаження активне?
   rjmp chkload1                     ;так, перевіряємо 1 навантаження
case2:sbrc flags,ld2                 ; 2 навантаження активне?
   rjmp chkload2                     ;так, перевіряємо 2 навантаження
case3:
   sbrc flags,ld1on                  ;встановлений час включення навантаження 1?
   rjmp setld1on                     ;так, перевіряємо значення часу включення
case4:
   sbrc flags,ld1off                 ;встановлений час виключення навантаження 1?
   rjmp setld1off                    ;так, перевіряємо значення часу виключення
case5:
   sbrc flags,ld2on                  ;встановлений час включення навантаження 2?
   rjmp setld2on                     ;так, перевіряємо значення часу включення
case6:
   sbrc flags,ld2off                 ;встановлений час виключення навантаження 2?
   rjmp setld2off                    ;так, перевіряємо значення часу виключення
case7:
   rjmp do                           ;повторюємо службову функцію

;****підпрограми для випадку обслуговування часу спрацювання навантажень і натискання кнопок********

chkload1:
   cp hour,ld1hroff                  ;досягнуто час вимкнення 1 навантаження?
   brne onload1
   cp minute,ld1minoff
   brne onload1
   sbi PORTD,LOAD1                   ;так, вимикаємо 1 навантаження
onload1:
   cp hour,ld1hron                   ;досягнуто час ввімкнення 1 навантаження?
   brne case2
   cp minute,ld1minon
   brne case2
   cbi PORTD,LOAD1                   ; так, вмикаємо 1 навантаження
   rjmp case2                        ;повторюємо з включеним навантаженням
chkload2:
   cp hour,ld2hroff                  ;досягнуто час вимикання 2 навантаження?
   brne onload2
   cp minute,ld2minoff
   brne onload2
   sbi PORTD,LOAD2                   ;так, вимикаємо 2 навантаження
onload2:
   cp hour,ld2hron                   ;досягнуто час вмикання 2 навантаження?
   brne case3
   cp minute,ld2minon
   brne case3
   cbi PORTD,LOAD2                  ;так, вмикаємо 2 навантаження
   rjmp case3                       ;повторюємо з включеним навантаженням
setld1on:
   sbr flags,0x01                   ;активуємо включення 1 навантаження
   rcall setrtc                     ;задаємо новий час включення
   mov ld1hron,hiset                ;і зберігаємо
   mov ld1minon,loset
   cbr flags,0x04                   ;очищуємо ld1on флаг
   rjmp case4
```

```
setld1off:
   rcall setrtc                        ;задаємо новий час вимкнення
   mov ld1hroff,hiset                  ;і зберігаємо
   mov ld1minoff,loset
   cbr flags,0x08                      ;очищуємо ld1off флаг
   rjmp case5
setld2on:
   sbr flags,0x02                      ;активуємо включення 2 навантаження
   rcall setrtc                        ;задаємо новий час включення
   mov ld2hron,hiset                   ;і зберігаємо
   mov ld2minon,loset
   cbr flags,0x10                      ;очищуємо ld2on флаг
   rjmp case6
setld2off:
   rcall setrtc                        ;задаємо новий час вимкнення
   mov ld2hroff,hiset                  ;і зберігаємо
   mov ld2minoff,loset
   cbr flags,0x20                      ;очищуємо ld2off флаг
   rjmp case7

;****Програма динамічного керування дисплеєм і сканування клавіатури*****
;****неосновні програми,що використовуються всі для відображення вищих
;**** і нижчих розрядів для кожної цифри, обновлюючи кожні 5 мс

display:
   ser temp                            ;очищуємо дисплей
   out PORTB,temp

;****Програма сканування клавіатури для обновлення флагів кнопок******************

keyscan:
   cbr flags,0x40                      ;очищуємо keyok флаг
   ldi key,0x10                        ;встановлюємо значення, що ні одна кнопка не нажата
   ser temp                            ;встановлюємо високий рівень на порту клавіатури
   out PORTB,temp                      ;перезапускаємо порт
   in temp,PORTD                       ;виключаємо LEDs і живлення навантажень
   ori temp,0x0f                       ;незалежно від попереднього
   out PORTD,temp                      ;сканування клавіатури
   ldi temp,0x0f                       ;встановлюємо на вихід колонку клавіатури
   out DDRB,temp                       ;рядок вхідний з “підтягуючими” резисторами
   ldi temp,0xf0                       ;всю колонку встановлюємо
   out PORTB,temp                      ;в низький рівень для сканування
   ldi temp,20                         ;короткий час для установки портів
tagain1:
   dec temp
   brne tagain1
   sbis PINB,ROW1                      ;шукаємо натискання кнопки в рядку
   ldi key,0                           ;і встановлюємо ROW покажчик (номер кнопки)
   sbis PINB,ROW2
   ldi key,4
   sbis PINB,ROW3
   ldi key,8
   sbis PINB,ROW4
   ldi key,12
   ldi temp,0xF0                       ;перезапускаємо порт B I/O для
   out DDRB,temp                       ;пошуку натискання кнопки в колонці
   ldi temp,0x0F                       ;вмикаємо “підтягуючі” резистори і
   out PORTB,temp                      ;встановлюємо нулі в рядку
   ldi temp,20                         ;короткий час для установки портів
tagain2:
   dec temp
   brne tagain2                        ;дозволений час для установки значень портів
   clr temp
   sbis PINB,COL1                      ;шукаємо натискання кнопки в колонці
   ldi temp,0                          ;і встановлюємо COL покажчик
   sbis PINB,COL2
   ldi temp,1
   sbis PINB,COL3
   ldi temp,2
   sbis PINB,COL4
   ldi temp,3
   add key,temp                        ;додаємо ROW і COL покажчики
   cpi key,0x10                        ;якщо кнопка не була натиснута
   breq nokey                          ;виходимо з програми, інакше
   ldi temp,0x10
   add key,temp                        ;керуємось таблицею 2
   out EEAR,key                        ;відправляємо адрес в EEPROM (0 - 15)
   sbi EECR,EERE                       ;чекаємо спрацювання EEPROM
   in key,EEDR                         ;читаємо декодований номер для натиснутої кнопки
convert:
   cpi key,10                          ;це була кнопка SET?
   brne notset                         ;ні, перевіряємо наступну кнопку
   set                                 ;так, встановлюємо T флаг в статусному регістрі
notset:
   cpi key,11                          ;натиснута CLEAR?
   brne notclear                       ;ні, перевіряємо наступну кнопку
   sbi PORTD,LOAD1                     ;так, вимикаємо усі навантаження
   sbi PORTD,LOAD2
```

```
   cbr flags,0x03                   ;деактивуємо обидва навантаження
notclear:
   cpi key,15                       ;натиснута LD1ON?
   brne notld1on                    ;ні, перевіряємо наступну кнопку
   sbr flags,0x04                   ;так, встановлюємо LD1ON флаг
notld1on:
   cpi key,14                       ;натиснута LD1OFF?
   brne notld1off                   ;ні, перевіряємо наступну кнопку
   sbr flags,0x08                   ;так, встановлюємо LD1OFF флаг
notld1off:
   cpi key,13                       ;натиснута LD2ON?
   brne notld2on                    ;ні, перевіряємо наступну кнопку
   sbr flags,0x10                   ;так, встановлюємо LD2ON флаг
notld2on:
   cpi key,12                       ;натиснута LD2OFF?
   brne notld2off                   ;ні, перевіряємо наступну кнопку
   sbr flags,0x20                   ;так, встановлюємо LD2OFF флаг
notld2off:

;***Програма генерації звуку для зворотного зв'язку з користувачем*****************
;***забезпечує тональність 4 Кгц для п'єзорезонатора кожні 5 мс*****

tactile:
   cbr flags,0x80
   cbi PORTD,PZ                     ;вмикаємо п'єзорезонатор
   ldi temp,125                     ;на короткий час
t1again:
   dec temp
   brne t1again on piezo
   sbi PORTD,PZ                     ;вмикаємо п'єзорезонатор
   ldi temp,125                     ;на короткий час
t2again:
   dec temp
   brne t2again
   sbrs flags,ms5                   ;повторюємо через 5 мс
   rjmp tactile
notok:
   cpi bounce,40
   brlo nokey
   sbr flags,0x40                   ;встановлюємо bounce флаг

nokey:
   ser temp
   out DDRB,temp                    ;перезапускаємо порт B як весь на виходи
   out PORTB,temp                   ;і очищаємо LEDs

;***Програма дисплею для динамічного керування всіма LED цифрами****************

   cbi PORTD,A1                     ;вмикаємо 1 цифру
   mov temp,lobyte                  ;знаходимо молодший розряд хвилин
digit1:
   cbr flags,0x80                   ;очищаємо 5 мс флаг тактування
   andi temp,0x0f                   ;маска старшого півбайту для цифри
   out EEAR,temp                    ;відправляємо адресу в EEPROM (0 - 15)
   sbi EECR,EERE                    ;чекаємо спрацювання EEPROM
   in temp,EEDR                     ;зчитуємо декодоване число
   sbrs flash,clear                 ;мигаємо кожні 1/2 секунди
   or temp,mask                     ;мигаємо цифрою, якщо потрібно
   out PORTB,temp                   ;заносимо в LED для 5 мс
led1:
   sbrs flags,ms5                   ;5 мс закінчились?
   rjmp led1                        ;ні, очікуємо далі
   sbi PORTD,A1                     ;вимикаємо 1 цифру
   ser temp                         ;очищуємо дисплей
   out PORTB,temp
   cbi PORTD,A2                     ;
   mov temp,lobyte                  ;знаходимо старший розряд хвилин
   swap temp
digit2:
   cbr flags,0x80                   ;очищаємо 5 мс флаг тактування
   andi temp,0x0f                   ;маска старшого півбайту для цифри
   out EEAR,temp                    ;відправляємо адресу в EEPROM (0 - 15)
   sbi EECR,EERE                    ;чекаємо спрацювання EEPROM
   in temp,EEDR                     ;зчитуємо декодоване число
   sbrs flash,clear                 ;мигаємо кожні 1/2 секунди
   or temp,mask                     ;мигаємо цифрою, якщо потрібно
   out PORTB,temp                   ;заносимо в LED для 5 мс
led2:
   sbrs flags,ms5                   ;5 мс закінчились?
   rjmp led2                        ;ні, очікуємо далі
   sbi PORTD,A2                     ;
   ser temp                         ;очищуємо дисплей
   out PORTB,temp
   cbi PORTD,A3                     ;
   mov temp,hibyte
digit3:
   cbr flags,0x80                   ;очищаємо 5 мс флаг тактування
   andi temp,0x0f                   ;маска старшого півбайту для цифри
```

```
   out EEAR,temp                    ;відправляємо адресу в EEPROM (0 - 15)
   sbi EECR,EERE                    ;чекаємо спрацювання EEPROM
   in temp,EEDR                     ;зчитуємо декодоване число
   sbrs second,clear                ;мигаємо дома точками
   andi temp,0x7f
   sbrs flash,clear                 ;мигаємо кожні 1/2 секунди
   or temp,mask                     ;мигаємо цифрою, якщо потрібно
   out PORTB,temp                   ;заносимо в LED для 5 мс
led3:
   sbrs flags,ms5                   ;5 мс закінчились?
   rjmp led3                        ;ні, очікуємо далі
   sbi PORTD,A3
   ser temp                         ;очищуємо дисплей
   out PORTB,temp
   cbi PORTD,A4;
   mov temp,hibyte
   swap temp
   andi temp,0x0f                   ;старше значення годин рівне нулю?
   brne digit4
   ldi temp,0xff                    ;так, пропускаємо старше значення годин
digit4:
   cbr flags,0x80                   ;очищаємо 5 мс флаг тактування
   andi temp,0x0f                   ;маска старшого півбайту для цифри
   out EEAR,temp                    ;відправляємо адресу в EEPROM (0 - 15)
   sbi EECR,EERE                    ;чекаємо спрацювання EEPROM
   in temp,EEDR                     ;зчитуємо декодоване число
   sbrs flash,clear                 ;мигаємо кожні 1/2 секунди
   or temp,mask                     ;мигаємо цифрою, якщо потрібно
   out PORTB,temp                   ;заносимо в LED для 5 мс
led4:
   sbrs flags,ms5                   ;5 мс закінчились?
   rjmp led4                        ;ні, очікуємо далі
   sbi PORTD,A4
   ser temp                         ;очищуємо дисплей
   out PORTB,temp
   tst mask                         ;відображення закінчилось?
   breq outled                      ;так, вихід
   cpi blink,50                     ;час мигання вийшов?
   brlo outled                      ;ні, вихід
   clr blink                        ;так, очищаємо лічильник програми мигання
   com flash                        ;і інвертуємо байт мигання
outled:
   ret

;****Функція для введення RTC/on-off годин і хвилин з клавіатури
;****записує хвилини в 'loset' в години в 'hiset'

setrtc:
   ser mask                         ;задаємо мигання дисплею
   ldi hibyte,0xdf                  ;відображаємо 'n' в старшому значення годин
   ser lobyte                       ;і пропускаємо нижній розряд годин і хвилини
hihrus:
   clr bounce

bounce1:
   rcall display                    ;дисплей і перевіряємо клавіатуру
   sbrs flags,keyok
   rjmp bounce1
   cbr flags,0x40                   ;очищуємо keyok флаг
   cpi key,0x03                     ;якщо старший розряд годин > 2
   brsh hihrus                      ;так, зчитуємо далі кнопки
hihrok:                             ;ні, правильне значення
   swap key                         ;записуємо hihour в старший півбайт
   mov hiset,key                    ;і зберігаємо в годинах
   ldi hibyte,0x0d                  ;відображаємо 'n' в нижньому розряді годин
   add hibyte,hiset                 ;об’єднуємо hihour і 'n'
lohrus:
   clr bounce
bounce2:
   rcall display                    ;дисплей і перевіряємо клавіатуру
   sbrs flags,keyok                 ;кнопка залишається нажатою?
   rjmp bounce2                     ;ні, продовжуємо далі
   cbr flags,0x40                   ;так, очищаємо keyok флаг
   mov temp,hibyte                  ;перевіряємо тоді години в загальному
   andi temp,0xf0                   ;чи не > 24
   add temp,key
   cpi temp,0x24                    ;години більше>24?
   brsh lohrus                      ;так, зчитуємо кнопку далі
   add hiset,key                    ;ні, об’єднуємо малі і старші години
lohrok:
   mov hibyte,hiset                 ;години на дисплеї встановлені
   ldi lobyte,0xdf                  ;відображаємо 'n' в старшому розряді хвилин
himinus:
   clr bounce
bounce3:
   rcall display                    ;дисплей і перевіряємо клавіатуру
   sbrs flags,keyok
   rjmp bounce3
```

```
   cbr flags,0x40                   ;очищаємо keyok флаг
   cpi key,6                        ;якщо старші хвилини >5
   brsh himinus                     ;ні, зчитуємо кнопку далі
lominok:
   swap key                         ;записуємо himin в старший півбайт
   mov loset,key                    ;і зберігаємо в хвилинах
   ldi lobyte,0x0d                  ;відображаємо 'n' в нижніх хвилинах
   add lobyte,loset                 ;об'єднуємо з старшими хвилинами
lominus:
   clr bounce
bounce4:
   rcall display                    ;дисплей і перевіряємо клавіатуру
   sbrs flags,keyok
   rjmp bounce4
   cbr flags,0x40                   ; очищаємо keyok флаг
   cpi key,10                       ;якщо кнопка >9
   brsh lominus                     ;ні, зчитуємо кнопку далі
   add loset,key                    ;так, об'єднуємо малі і старші хвилини
   clr mask                         ;очищуємо мигання цифр
   ret                              ;і виходимо з установки значень часу

;****Підпрограма обслуговування переривання по переповненню таймера*****************************
;****Обновлюється в 5 мс, відображає і зменшує лічильник для забезпечення  відповідних значень RTC

tick:
   in status,SREG                   ;зберігаємо статусний регістр
   inc tock                         ;додаємо одиницю до 5 мс 'tock' лічильника
   inc blink                        ;і до лічильника швидкості відображення
   inc bounce                       ;і зменшення тривалості паузи
   sbr flags,0x80                   ;встановлюємо 5 мс флаг для часу відображення дисплею
   cpi tock,200                     ;пройшла одна секунда?
   breq onesec                      ;так, додаємо одиницю до секунд
   nop                              ;час для балансування тривалості переривання
   rjmp nosecond                    ;ні, вихід
onesec:
   inc second                       ;додаємо один до секунд
   clr tock                         ;очищаємо 5 мс лічильник
nosecond:
   ldi timer,176                    ;перезавантажуємо таймер
   out TCNT0,timer
   out SREG,status                  ;зчитуємо статусний регістр
   reti                             ;повертаємось в основну програму
```