МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра АТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання курсового проекту з дисциплін
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА І МІКРОПРОЦЕСОРИ»,
«МІКРОПРОЦЕСОРНІ СИСТЕМИ»

для студентів напрямів
6.050903 “Телекомунікаційні системи та мережі”
6.050201 “Системна інженерія”
усіх форм навчання

Розглянуто на засіданні кафедри
«Автоматика і телекомунікації»
протокол № 9 від 30.08.2010р.

Затверджені на засіданні
Навчально-видавничої ради ДонНТУ
Протокол № 4 від 07.10.2010
Р.№340

ДОНЕЦЬК – 2010

Методичні вказівки до виконання курсового проекту з дисциплін «Обчислювальна техніка і мікропроцесори» для студентів напряму 6.050903 “Телекомунікаційні системи та мережі” і «Мікропроцесорні системи» для студентів напряму 6.050201 “Системна інженерія” усіх форм навчання. /Укладачі Суков С.Ф, В.Я., Яремко І.М., Батир С. С. – Донецьк, ДонНТУ, 2010. - 18с.

Укладачі:

Суков С.Ф, В.Я., Яремко І.М., Батир С.С.

Відповідальний за випуск:
Зав. кафедрою «Автоматика і телекомунікації» к.т.н., доцент
Бессараб В.І.

Рецензент: к.т.н., доцент кафедри «Автоматизовані системи управління» П.О.Шатохін

Завдання на курсовий проект

В курсовому проекті пропонується за допомогою лабораторного стенда на основі мікроконтролера AT90S8515 скласти програмне забезпечення для реалізації (на вибір викладача):

1. Електронного годинника реального часу;

при подачі живлення на індикаторі відображується 00 годин 00 хвилин і годинник починає "іти". При цьому повинна блимати точка другого розряду індикатора з періодом 1с (0.5с горить, 0.5с не горить). При натисненні кнопки "*" включається режим введення часу при якому послідовно задаються годинник і хвилини і після введення останньої цифри починається хід годинника. При введенні часу повинна здійснюватися перевірка на некоректне введення (наприклад при введенні першої цифри можна ввести тільки "0", "1" або "2" інших кнопок повинні ігноруватися).

2. Таймера прямої ходи;

При подачі живлення включається режим введення часу при якому послідовно задаються години і хвилини і після введення останньої цифри починається хід таймера від нуля до вказаного часу. При цьому повинна блимати точка другого розряду індикатора з періодом 1с (0.5с горить, 0.5с не горить). При натисненні кнопки "*" також включається режим введення часу. При введенні часу повинна здійснюватися перевірка на некоректне введення (наприклад при введенні першої цифри можна ввести тільки "0", "1" або "2" інших кнопок повинні ігноруватися).

3. Таймера зворотної ходи.

При подачі живлення включається режим введення часу при якому послідовно задаються години і хвилини і після введення останньої цифри починається хід таймера від вказаного часу до нуля. При цьому повинна блимати точка другого розряду індикатора з періодом 1с (0.5с горить, 0.5с не горить). При натисненні кнопки "*" також включається режим введення часу. При введенні часу повинна здійснюватися перевірка на некоректне введення (наприклад при введенні першої цифри можна ввести тільки "0", "1" або "2" інших кнопок повинні ігноруватися).

Окрім цього викладач видає індивідуальні завдання для реалізації проекту. Методичні вказівки містять загальні питання з реалізації проекту, які розглядаються на прикладах і які можна використовувати в своїх розробках.

Завдання: розробити годинник реального часу на базі мікроконтролера AT90S8515. Час відображується за допомогою чотирьох семисегментних індикаторів, управління здійснюється за допомогою клавіатури (3x4 - 12 кнопок). Програмування і прошивка МК здійснюється за допомогою додатка Algorithm Builder.

Робота пристрою: при подачі живлення на індикаторі відображується 00 годин 00 хвилин і годинник починає "іти". При цьому повинна блимати точка другого розряду індикатора з періодом 1с (0.5с горить, 0.5с не горить). При натисненні кнопки "*" включається режим введення часу при якому послідовно задаються годинник і хвилини і після введення останньої цифри починається хід годинника. При введенні часу повинна здійснюватися перевірка на некоректне введення (наприклад при введенні першої цифри можна ввести тільки "0", "1" або "2" інших кнопок повинні ігноруватися).
Принципова електрична схема пристрою зображена на рис.1.

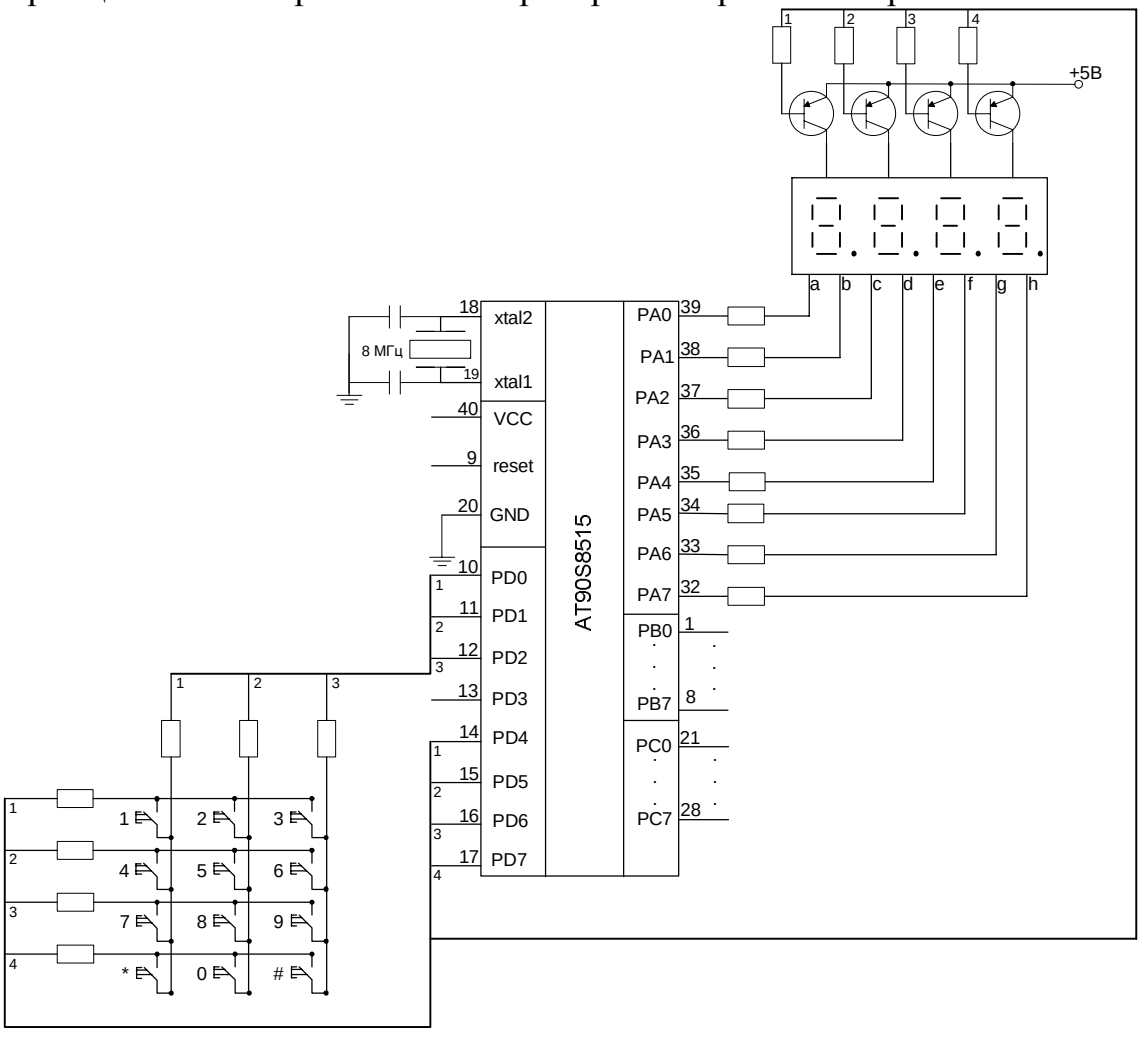

Рисунок 1 - Принципова електрична схема блоку клавіатури і індикації.

1. Опис принципової схеми

На рис.1 представлена принципова електрична схема годинника. Мікроконтролер є основною і єдиною мікросхемою, використовуваною в цій розробці. Для завдання тактової частоти контролера використовується кварцовий резонатор на 8 МГц. Як пристрій відображення використано чотири індикатори червоного кольору свічення із загальним анодом, кожен індикатор містить 8 сегментів.

Індикація поточного часу здійснюється динамічно, в кожний конкретний момент часу відображується лише одна цифра, що дозволяє значно знизити апаратні витрати. Аноди кожної з чотирьох цифр є роздільними, що дозволяє в даний момент часу підключити до джерела живлення тільки один анод і відобразити одну цифру. Для цього годинник має чотири транзисторні ключі. Ключі управляються мікроконтролером, причому відповідний ключ відкритий, якщо на виведенні контролера присутній логічний нуль. Однойменні сегменти усіх чотирьох цифр сполучені разом і через струмообмежувальні резистори підключені до виводів порту "A" (висновки PA.0 . PA.7). Програма, що управляє, один за іншим підключає розряди індикатора до джерела живлення і одночасно на відповідні виводи порту "A" виставляється код цифри, що відображується. Оскільки сканування індикатора відбувається дуже швидко, мерехтіння цифр стає непомітним. Як видно з схеми, лінії сканування клавіатури і індикації загальні, що дозволяє зменшити число використовуваних виводів. Живиться годинник від стабілізованого джерела живлення напругою 5В. Відразу після включення годинника програма дозволяє переривання, настроює порти контроллера відповідним чином і встановлює покажчик стека на старші адреси внутрішньої пам'яті даних. Далі програма переводить пристрій в режим годинника і запускає цикл сканування клавіатури, індикатора і цикл відліку часу. Основою програми є обробник переривань від таймера.

2. Опис можливого варіанту алгоритму програми

Годинник реального часу організований з використанням переривань по таймеру 0, який тактується системною частотою поділеною на 256. Таймер заздалегідь завантажується числом 100, що задає період генерації переривань по переповнюванню таймера кожні 5 мс, забезпечуючи високу точність ходу годинника, за умови використання якісного кварцового резонатора. При використанні кварцового резонатора 8 Мгц тривалість циклу інструкції дорівнює 0.125 мкс. З урахуванням цього, при записі числа n в регістр таймера 0 TCNT0 період його переповнювання визначається за виразом:

(256 - n)*256*0,125 мкс

Таким чином запис числа 100 забезпечить період переповнювання 5мс з високою для відліку реального часу точністю :

(256-100)*256*0,125*=4,992мс.

Усю програму можна розбити на декілька частин - це основна програма і підпрограми переривання за переповненням таймера/лічильника, відліки часу, виводу на індикацію, сканування клавіатури і режимів введення часу.
У основній програмі настроюється МК і очікується натиснення клавіші, якщо клавіша натиснута, то визначається її код і відбувається перехід до однієї з підпрограм введення, де аналізується яка цифра вводиться в даний момент і коректність введеної цифри. Окрім цього, кожні 5мс відбувається виклик підпрограми таймера/лічильника де нарощується лічильник спрацьовувань (коли його значення стане рівним 200, то це означає, що пройшла 1сек) і викликаються підпрограми відліку часу, виводу на індикацію і сканування клавіатури.

2.1. Використання ресурсів

Використання периферійних пристроїв :
Таймер/счетчик0 - лічильник імпульсів з періодом 5мс;
7 ліній порту D - підключення клавіатури 3х4 і управління анодами цифрових індикаторів;
8 ліній порту A - сегментні лінії індикаторів;

Використовувані регістри:
R0 - використовується при роботі з таблицею даних, знаходиться лічене значення з таблиці даних за адресою Z;
R1 - число секунд;
R2 - число хвилин;
R3 - число годинника;
R4 - виводиться на перший індикатор;
R5 - виводиться на другий індикатор;
R6 - виводиться на 3-й індикатор;
R7 - виводиться на 4-й індикатор;
R8 - номер індикатора на який в даний момент виводитиметься інформація;
R9 - лічильник спрацьовування таймера;

R10, R11, R12, R13, R14, R15 - використовуються в підпрограмі для сканування клавіатури;
R18, R19 - 16-розрядний регістр FLAGS 12 молодших бітів якого відповідають 12 клавішам клавіатури, при встановленні одного з бітів в "0" вважається, що відповідна клавіша натиснута.
R16, R17, R20 - використовуються як допоміжні регістри.

2.2. Основна програма

При подачі живлення і виконанні умов скидання виконується процедура скидання (Reset) для ініціалізації системних пристроїв. Лінії портів налаштовуються на потрібні рівні. Порт А необхідно настроїти на вихід, старшу тетраду порту D на вихід і молодшу на вхід. Старша тетрада порту D використовується і для реалізації індикації. Тобто подаючи на один з цих виходів логічний "0" сканується клавіатура і одночасно запалюється потрібний індикатор. Далі обнуляються використовувані в програмі регістри або заносяться в них потрібні значення. Настроюється таймер і заносяться потрібні значення в регістри управління МК. Переривання по переповнюванню таймера стає активним після дозволу глобальних переривань. Далі програма чекає натиснення клавіші і якщо клавіша натиснута, то переходить до обробки підпрограми відповідної цьому натисненню. Реалізувати розпізнавання натиснення клавіші зручно за допомогою 16-розрядного регістра, оскільки клавіш 12, то використовуватиметься 12 його молодших розрядів. Наприклад, можна встановити усі розряди такого регістра в 1 і при натисненні кнопки обнуляти відповідний біт регістра. У цьому прикладі таким регістром виступає регістр FLAGS. Використовуючи такий спосіб легко визначити чи натиснута клавіша і яка саме. На рис.2 представлена блок-схема основної програми.

Ініціалізація стека - це запис старшої адреси ОЗУ ($25F) в покажчик стека SP.

Налаштування таймера : запис числа $04 в регістр управління таймером TCCR0, запис числа 100 ($64) в таймер TCNT0 щоб він переповнився через 5мс, запис числа $02 в регістр TIMSK (дозвіл переривання по переповнюванню таймера 0).

Далі заноситься 1 в біт I (дозвіл глобального переривання).

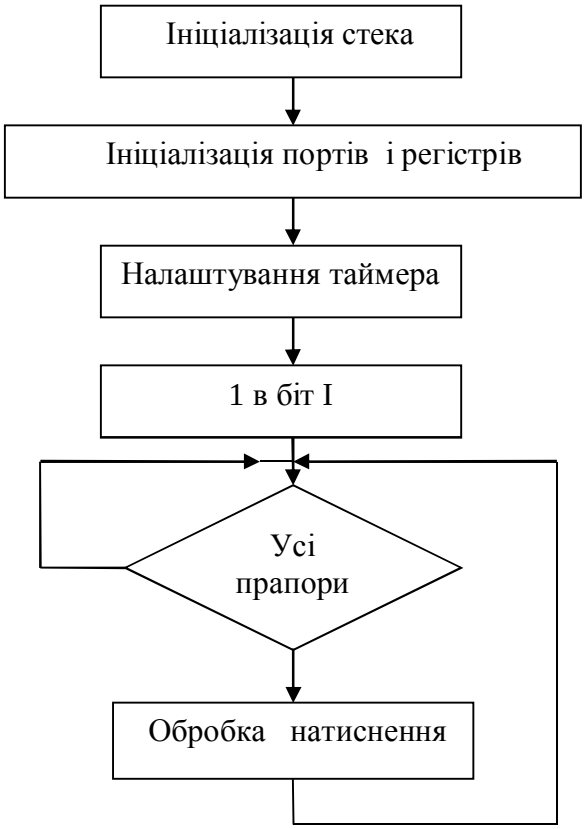

Рисунок 2 - Блок-схема основної програми

Потім перевіряються прапори натиснення і якщо який-небудь прапор = 0, то обробляється натиснення і здійснюється повернення на перевірку прапорів. Таким чином, основна програма в нескінченному циклі перевіряє прапори натиснення клавіш. Встановлювати ці прапори повинна підпрограма обробки клавіатури KLAV, яка викликається в підпрограмі обробки переривання таймера і тому виконується через кожні 5 мс. Яким чином підпрограма KLAV сканує клавіатуру і встановлює прапори натиснення клавіш буде розглянуто нижче.

На рис.3 наведений приклад реалізації основної програми.

В даному прикладі спочатку налаштовується МК, потім перевіряється регістр FLAGS на рівність одного з його бітів 0. Якщо один з бітів дорівнює 0, то за допомогою зрушення праворуч цього регістра визначається який саме біт дорівнює 0, після чого в регістрі R22 знаходитиметься код натиснутої клавіші.

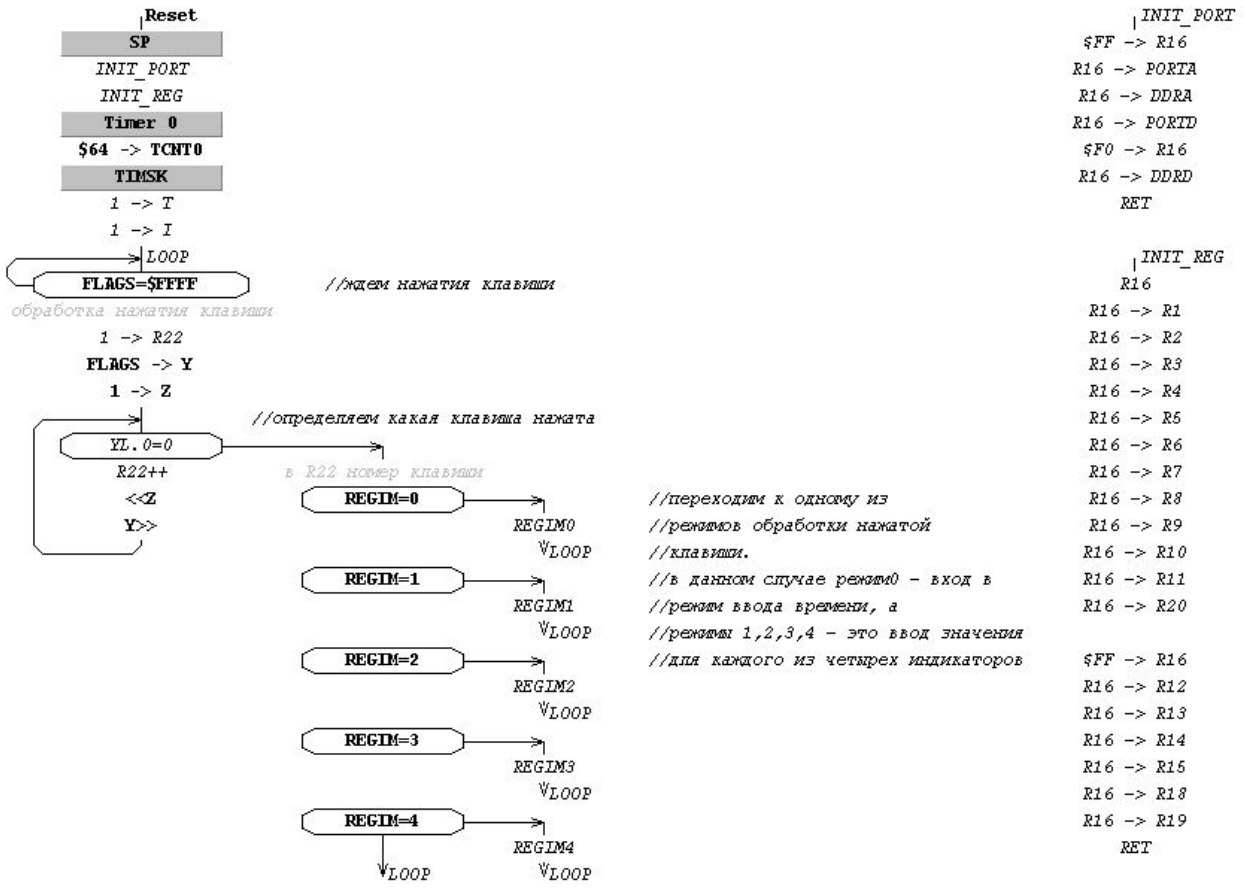

Рисунок 3 - Приклад основної частини програми складений за допомогою Algorithm Builder

2.3. Режими роботи пристрою

Роботу пристрою можна розділити на 5 режимів:

Режим 0 - звичайний хід годинника, усі розряди горять без миготіння.

Режим 1 - миготить перший розряд, очікується введення годин

Режим 2 - миготить другий розряд, очікується введення годин

Режим 3 - миготить третій розряд, очікується введення хвилин

Режим 1 - миготить четвертий розряд, очікується введення хвилин.

Таким чином реагувати на натиснення клавіш в різних режимах необхідно по-різному, тому для кожного режиму існує своя підпрограма обробки натиснення клавіші. Доцільно який-небудь регістр визначити як змінну REGIM, в якій зберігатиметься номер поточного режиму. Тепер якщо в режимі 0 виконується натиснення клавіші, то викликається підпрограма REGIM0, яка повинна перевірити коректність натиснення і перейти (якщо натиснута коректна кнопка) в наступний режим. Для режиму 1 як обробник натиснення клавіші використовується підпрограма REGIM1 і так далі

Як приклад нижче приведена блок-схема підпрограми REGIM0

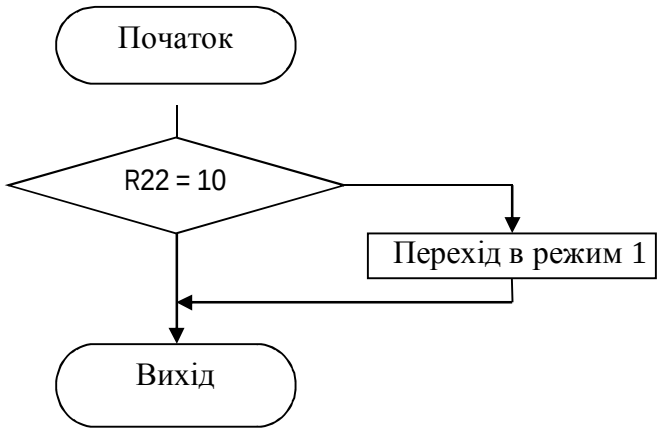

Рисунок 4 - Блок-схема підпрограми REGIM0

При виклику підпрограми REGIM0 в регістрі R22 зберігався номер натиснутої клавіші. Тому здійснюється перевірка R22=10 (натиснута "*") і тільки якщо натиснута кнопка "*" те переходимо в наступний режим.

Зрозуміло, що при натисненні кнопки в режимі 2 необхідно записати введене значення годинника в регістр R3 (поточне значення годинника), а при натисненні кнопки в режимі 4 треба виконати ті ж дії з хвилинами.

2.4. Підпрограма обробки переривання таймера

У Algorithm Builder це підпрограма з ім'ям ***Timer_0_Overflow***. Виклик цієї підпрограми здійснюється при переповнюванні таймера/лічильника 0, в нашому випадку кожні 5мс. При вході в цю підпрограму необхідно реалізувати збереження важливих змінних і регістра прапорів SREG в стеку, а при виході з неї відновити ці значення. Далі можна організувати лічильник (наприклад використовувати регістр R9), який фіксуватиме число входів в дану п/п. Коли цей лічильник стане рівним 200, то це означатиме, що пройшла 1с (200*5мс=1с) і при цьому необхідно збільшити поточне значення секунд на 1. Цей лічильник можна використовувати в п/п відліку часу. Потім необхідно реалізувати виклик підпрограм відліку часу, якщо ми знаходимося не в режимі введення, сканування клавіатури і виводу на індикацію. Зразковий алгоритм реалізації підпрограми таймера приведений на рис.5.

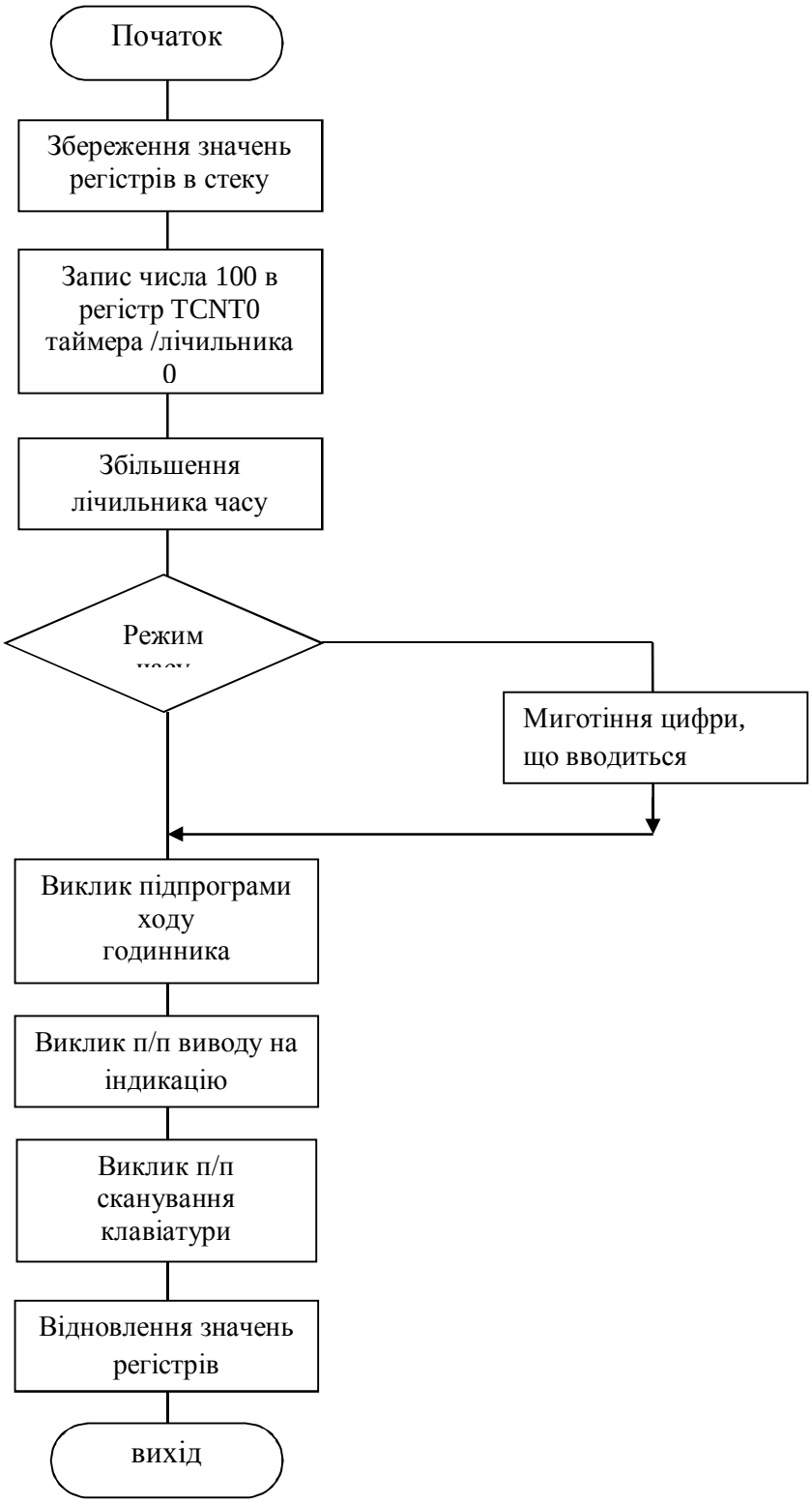

Рисунок 5 - Алгоритм підпрограми таймера/лічильника

2.5. Підпрограма відліку часу

У цій п/п можна використовувати лічильник, описаний в п/п таймера. Приклад представлений на рис. 6.

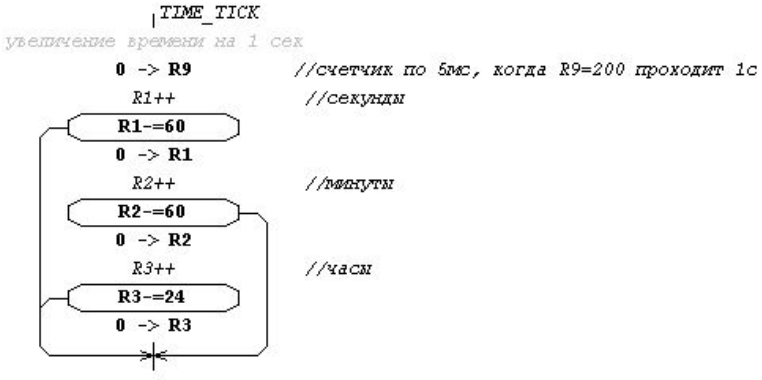

Рисунок 6 - Приклад п/п відліку часу

У цій п/п в регістрі R1 знаходиться число секунд, в R2 - число хвилин і в R3 - число годинника. R9 - це лічильник, який інкрементується кожного разу при виклику підпрограми обробки переривання за переповнюванням таймера 0, тобто R9 збільшується кожні 5 мс. Таким чином, коли R9 досягає 200 (проходить 1 сік) викликається TIME_TICK, яка обнуляє R9 і збільшує поточне число секунд на 1. Крім цього, ще виконується перевірка секунд на рівність 60 і, якщо необхідно, збільшуються хвилини і години.

При організації індикації існує наступна проблема: в R3 зберігається поточне число годинника в двійковому виді. А необхідно окремо виводити число десятків і число одиниць на перший і другий розряди індикатора. Наприклад, зараз 15 годин (R3=15), тоді на перший розряд індикатора необхідно вивести 1, а на другій - 5. Щоб розділити десятки і одиниці числа R3 треба виконати наступні дії, вказані на рис.7.

У цьому прикладі використовуються допоміжні регістри R16 і R17, в R17 знаходитимуться десятки, а в R16 - одиниці. У R3 початкове число годин, Аналогічну програму треба скласти і для хвилин.

2.6. Виведення часу на індикацію

Динамічна індикація здійснюється таким чином: спочатку в порт А виводиться код, який засвічує сегменти індикатора, при яких світиться

необхідна цифра (0, 1, 2 .) і відкривається перший транзисторний ключ, шляхом посилки 0 в PORTD.4. При цьому відображуватиметься тільки перший розряд

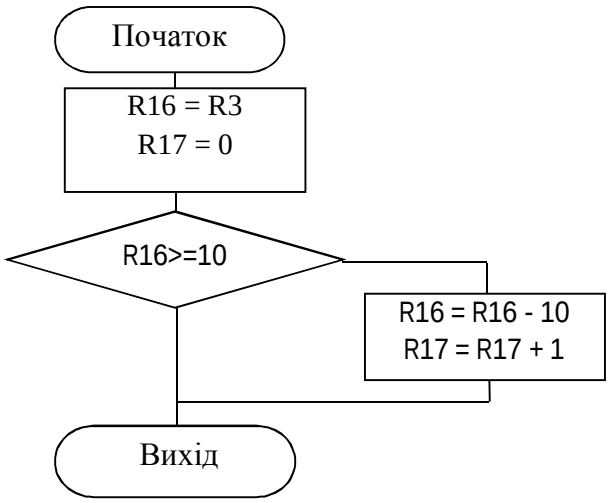

Рисунок 7 - Приклад розподілу числа годинника на десятки і одиниці

індикатора. Через 5 мс необхідно закрити перший транзисторний ключ, вивести в порт А код, який відповідає цифрі, що виводиться на другий розряд і відкрити другий транзисторний ключ. При цьому відображуватиметься тільки другий розряд індикатора. Далі ця процедура повторюється для кожного розряду і після відображення четвертого розряду знову виводиться перший розряд. Таким чином, час оновлення кожного розряду складає 5мс * 4 = 20 мс, при такій частоті оновлення людське око не помічає мерехтіння і сприймає індикацію як статичну.

Для виконання цих дій доцільно використовувати вже налагоджений раніше таймер на 5 мс. Тому в підпрограмі обробника переривання за переповненням таймера викликатимемося функція INDIKATOR, яка і виконуватиме усі вищеперелічені дії.

При виводі на індикацію послідовно повинні виводитися на кожний з чотирьох розрядів індикатора відповідні значення. Для перебору розрядів може бути використаний лічильник (наприклад на регістрі R8). Оскільки усього розрядів індикатора 4, то значення регістра R8 можуть мінятися в межах від 1 до 4. При вході в п/п INDIKATOR лічильник інкрементується і, відповідно до його значення, видається скануючий "0" на відповідну лінію порту D і оновлюються дані відповідного індикатора. Скануючий "0" далі використовується також і для опитування клавіатури. При досягненні лічильником максимального значення його слід обнуляти. Для полегшення виводу можна задіювати 4 регістри, наприклад R4, R5, R6 і R7. Підпрограма INDIKATOR відображуватиме значення регістра R4 на першому розряді індикатора, R5 - на другому, R6 - на третьому, R7 - на четвертому. Тепер в будь-

якому місці програми при зміні значень регістрів R4.R7 здійснюватиметься зміна інформації, що виводиться на індикатор.

Таким чином, підпрограма INDIKATOR повинна виводити на 1 розряд індикатора число, яке зберігається в R4. Нехай в R4 зберігається число від 0 до 9, тоді необхідно виводити в порт A код, який відповідає поточному значенню R4. Це зручно здійснити заздалегідь склавши таблицю цих кодів. Таблиця складається з 10 байт, якщо перший байт таблиці послати в порт A, те на індикаторі з'явиться цифра 0, якщо послати другий байт таблиці, то відображуватиметься цифра 1 і так далі. Ця таблиця кодів розташовується в пам'яті програм. Тепер щоб вивести на індикатор значення регістра R4, необхідно узяти адресу першого елементу таблиці кодів, додати до цієї адреси регістр R4, за отриманою адресою розрахувати байт з таблиці і вивести цей байт в порт A. Підпрограма, що виконує ці дії, показана на рис.8.

```
        |OUT_IND
      R0 ->
INDIK_COD * 2 -> Z
      Z+R19
       LPM
   R0 -> PORTA
      -> R0
       RET


        |INDIK_COD
DB: $C0,$F9,$A4,$B0,$99,$92,$82,$F8,$80,$90,$FF
```

Рисунок 8 - Приклад використання таблиці кодів

Algorithm Builder дозволяє помістити в тіло програми довільні дані: таблиці кодів, рядки повідомлень і ін. Для їх запису використовується елемент алгоритму "FIELD". Для визначення їх розташування в адресному просторі, перед ними слід поставити вершину або мітку.

Синтаксис запису даних :

DB: [Format] #, #, #.

Формат даних вказувати необов'язково, при цьому за умовчанням прийматиметься однобайтний формат.

При читанні даних за допомогою операції "LPM", в регістр Z необхідно завантажувати подвоєну адресу програми, оскільки ця операція припускає побайтову організацію пам'яті програми

В даному прикладі в R19 знаходиться значення цифри, що виводиться, а INDIK_COD - це таблиця даних зі значеннями відповідними кожній цифрі. Наприклад, якщо вивести в порт A значення $C0, то на індикаторі висвітиться 0, якщо $F9 - то 1 і так далі. Щоб з цієї таблиці вибрати потрібне значення застосовується команда LPM. Заздалегідь в регістр Z заноситься початкова адреса таблиці помножена на 2. Далі до цього регістра додається номер

потрібного значення (регістр R19) і після виконання команди LPM потрібне значення міститиметься в регістрі R0.

При складанні таблиці кодів враховувався той факт, що сегменти індикатора світяться при подачі на них логічного "0".

Блок-схема підпрограми INDIKATOR представлена на рис.9.

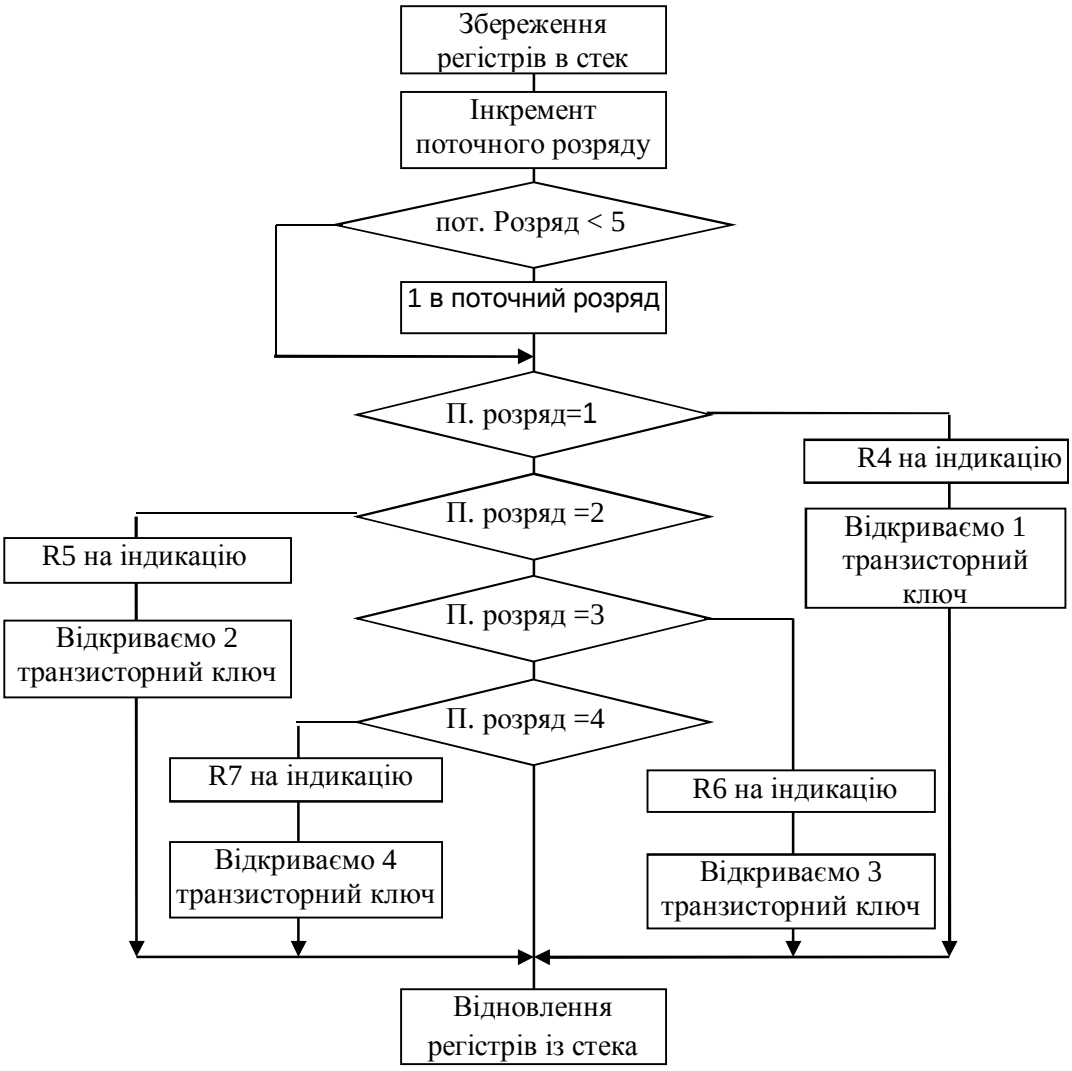

Рисунок 9 - Блок-схема підпрограми індикатор

На початку підпрограми виконується збереження важливих регістрів в стеку, далі інкрементується лічильник номера розряду індикатора і перевіряється на вихід за межі допустимих значень. Після цього, якщо в даний момент обробляється перший розряд індикатора, то R4 виводиться на індикатор і потім відкривається перший транзисторний ключ, який включає перший розряд індикатора.

2.7. Опитування клавіатури

Клавіатура сканується за допомогою логічного "0", який подається на відповідну лінію при виводі на індикацію. Далі вимагається опитати 3 молодші розряди порту D, і якщо один з них дорівнює "0", то це є ознакою натиснення відповідної кнопки. У цій підпрограмі також вимагається реалізувати процедуру антибрязкоту.

На рис.10 показаний брязкіт контактів при натисненні на кнопку. Як видно з малюнка в результаті брязкоту контактів кнопки відбувається імітація її багатократного натиснення. Для того, щоб уникнути неправильного декодування, читання скан-кода відбувається через деякий час після фіксації факту зміни стану.

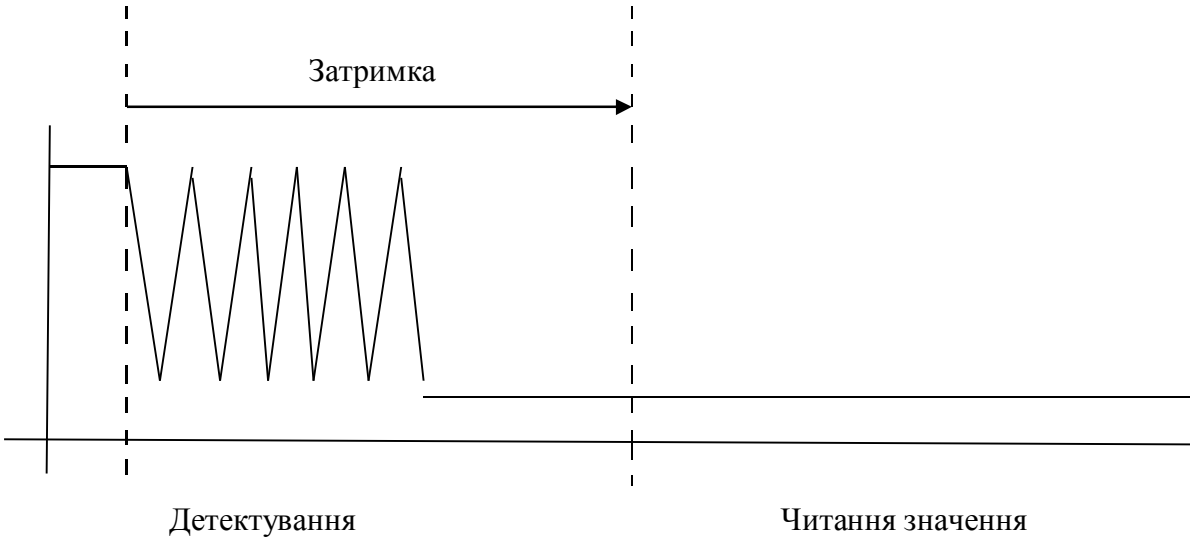

Рисунок 10 - Брязкіт контактів

Найчастіше використовується затримка тривалістю 20мс і після цього знову опитується розряд порту D на якому перед цим був присутній "0". Якщо стан не змінився, то вважається що кнопка натиснута.

Доцільно використовувати вже налагоджений таймер на переповнювання через кожні 5 мс. Для реалізації цього слід створити підпрограму KLAV, яка виконуватиме опитування клавіатури, і викликатися в підпрограмі обробки переривання по переповнюванню таймера. Таким чином, підпрограма KLAV виконуватиметься через кожні 5 мс.

Оскільки сканування клавіатури і індикації виконується за допомогою одних лінії, то опитування клавіатури буде пов'язано з індикацією. Наприклад, якщо в даний момент на індикаторі відображується перший розряд (присутній 0 на лінії PORTD.4), то зараз подається 0 на перший рядок клавіатури, і можна рахувати стан тільки перших трьох кнопок цього рядка. Таким чином, номер

розряду індикатора, який відображується в даний момент, буде номером рядка клавіатури, яку можна сканувати в даний момент.

У підпрограмі INDIKATOR вже була введена змінна, в якій зберігався номер поточного розряду індикатора. Цю змінну і треба використовувати при опитуванні клавіатури.

Блок-схема підпрограми KLAV представлена на рис. 11.

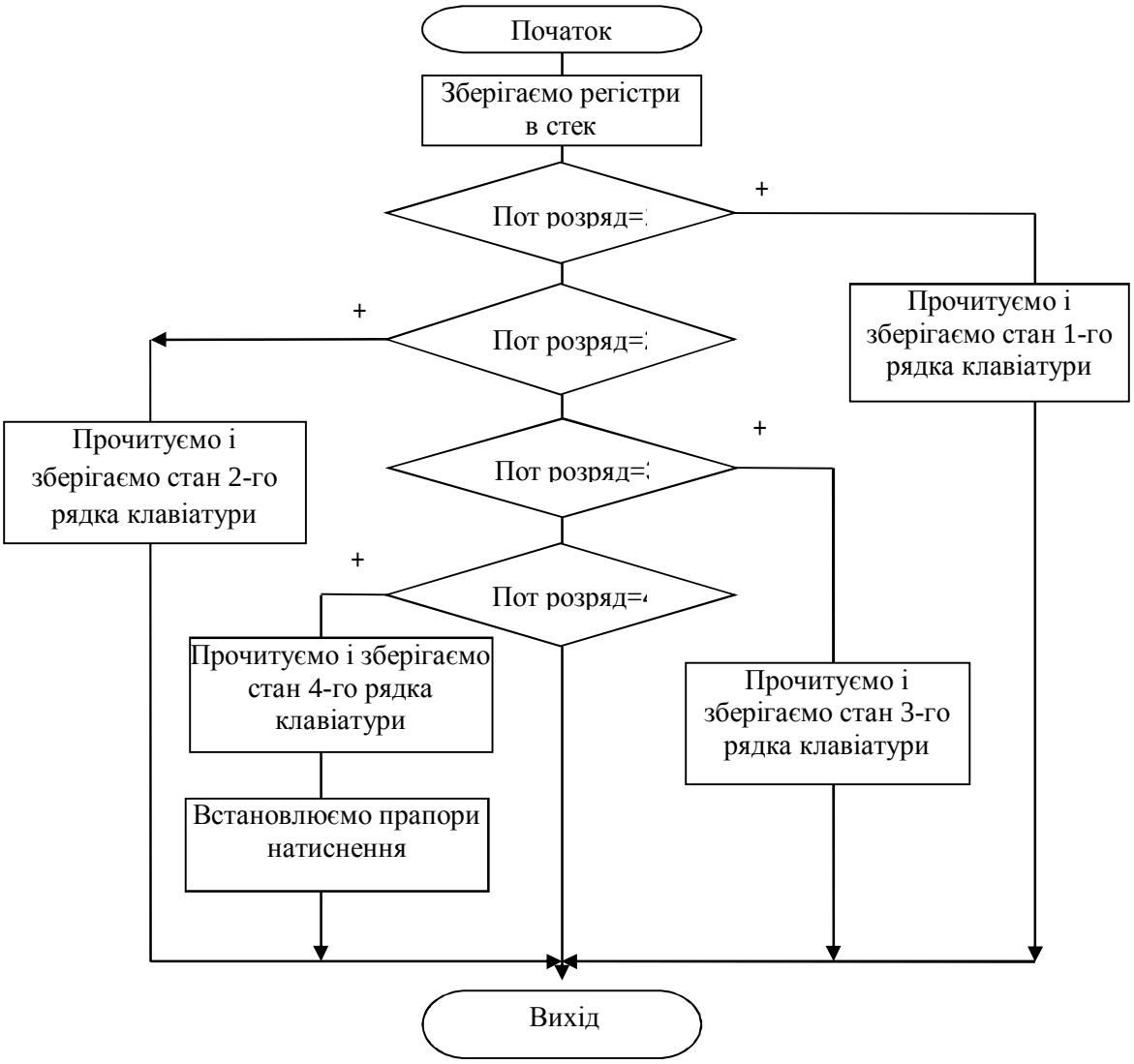

Рисунок 11 - Блок-схема підпрограми KLAV

Для боротьби з брязкотом контактів встановлювати прапори натиснення необхідно таким чином: якщо два останні прочитування показали, що кнопка натиснута (лічені нулі), а до цього кнопка була ненатиснута, то виставляється прапор натиснення кнопки. Таку перевірку необхідно зробити для кожної кнопки.

Навчальне видання

Методичні вказівки до виконання лабораторних робіт з дисципліни

«Обчислювальна техніка і мікропроцесори»

для студентів напряму 6.050903 “Телекомунікаційні системи та мережі”

і «Мікропроцесорні системи»

для студентів напряму 6.050201 “Системна інженерія”

(усіх форм навчання)

Укладачі:

Суков Сергій Феліксович, к.т.н., доцент

Яремко Ігор Миколайович, ст.викладач

Батир Семен Сергійович, асистент

Затверджені на засіданні Навчально-видавничої ради ДонНТУ
Протокол № 4 від 07.10.2010 Р.№340 1,12 друк.арк