Форма № Н-6.01

Вінницький національний технічний університет
Кафедра електромеханічних систем автоматизації в промисловості і на транспорті

КУРСОВА РОБОТА

з дисципліни «Обчислювальна техніка і програмування»
на тему «Розробка програми для мікропроцесорного присрою»

Студента 3 курсу ЕМмс-13б групи
напряму підготовки 7.050702
Електромеханіка
спеціальності Електромеханічні системи автоматизації та електропривод

Деревицький В. М.
(прізвище та ініціали)

Керівник ст. викл. каф. ЕМСАПТ
(посада, вчене звання)
Шевчук Ю. В.
(науковий ступінь, прізвище та ініціали)

Національна шкала
Кількість балів Оцінка: ECTS

Члени комісії

	ст. викл. Шевчук Ю.В.
(підпис)	(прізвище та ініціали)
(підпис)	(прізвище та ініціали)
(підпис)	(прізвище та ініціали)

м. Вінниця - 2013 рік

АНОТАЦІЯ

Деревицький В.М. «Розробка програми для мікропроцесорного пристрою». Курсова робота. – Вінниця.: ВНТУ. 2013.– 57 с. Бібліогр.:13. Іл.:13. Табл.:3.

Під час розробки програми для мікропроцесорного пристрою було розглянуто будову мікроконтролера AT90S2313. Структурну та принципова схеми, алгоритм роботи програми розроблено згідно порівняльного аналізу рішень поставленого завдання. Розробка програми була здійснена на двох мовах програмування: Assembler та C, здійснено моделювання в Proteus.

Ключові слова: мікроконтролер, семисегментний індикатор, кнопка, алгоритм роботи, програма.

АННОТАЦИЯ

Деревицкий В.М. «Разработка программы для микропроцессорного устройства». Курсовая работа. - Винница.: ВНТУ. 2013 - 57. с. Библ. 13. Ил. 13. Табл. 3.

В разработке программы для микропроцессорного устройства были рассмотрено строение микроконтроллера AT90S2313. Структурную и принципиальную схемы, алгоритм работы программы разработаны согласно сравнительного анализа решений поставленой задачи. Разработка программы была осуществлена на двух языках программирования: Assembler и C, осуществлено моделирование в Proteus.

Ключевые слова: микроконтроллер, семисегментный индикатор, кнопка, алгоритм работы, программа.

THE SUMMARY

Derevickiy V.M. "Development of programs for microprocessor." Coursework. - Vinnitsa.: VNTU. 2013. – 57 st. Refs: 13. Fig.: 13. Tabl.: 3.

Provide a brief description of the degree proektuvannya.Pid the development program for the microprocessor was considered structure microcontroller AT90S2313. Structural and schematic diagrams, the algorithm of the program developed by comparative analysis of solutions of this problem. Development of the program was carried out on two programming languages: Assembler and C by simulation in Proteus.

Keywords: microcontroller, seven-segment LED indicator, the algorithm of the program.

ЗМІСТ

ВСТУП

Найважливіша властивість мікропроцесорних систем — висока гнучкість, можливість швидкої перенастройки, а при необхідності навіть зміни алгоритмів управління. Як правило, перенастройка здійснюється програмним шляхом, без суттєвих виробничих затрат. Більше того, мікропроцесор дозволяє легко реалізовувати принцип відкритих систем, функціональні можливості яких можуть нарощуватись по мірі необхідності або по мірі виникнення нових технічних засобів.

Саме застосування мікропроцесорних засобів дозволяє реалізувати високонадійні системи управління, забезпечити їх гнучкість при адаптації до різних умов експлуатації, різних типів систем тощо [1, 2].

Для реалізації мікропроцесорної системи необхідно застосувати мікроконтролер [3] та в деяких випадках для розв’язання окремих задач зовнішню пам’ять, зовнішній АЦП, ПВЗ і комутатор сигналів.

Широке різноманіття мікрочіпів в сучасний період дозволяє в якості мікроконтролера використовувати розробки таких фірм як Intel, Atmel, Microchip, Motorola, Analog Devices, Texas Instruments тощо.

Враховуючи те, що в різних моделях мікроконтролерів, окрім внутрішньої постійної і оперативної пам’яті є також і АЦП з ПВЗ та комутатором, в багатьох випадках цих апаратних ресурсів вистачає для реалізації необхідних систем контролю. Розмір пам’яті визначається конкретною реалізацією проектованої системи. Застосування зовнішнього АЦП обґрунтовано лише необхідністю збільшення розрядності представлення сигналів та швидкодії [3].

Оскільки архітектура мікропроцесорної системи традиційна, то основну увагу треба зосередити на розробці алгоритмічного та програмного забезпечення.

1 БУДОВА МІКРОКОНТРОЛЕРА AT90S2313, ОГЛЯД АЛГОРИТМІВ, СХЕМНИХ ТА ПРОГРАМНИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1 Статична індикація

Виведення на семисегментний світлодіодний індикатор цифрової інформації в схемі з мікроконтролером зустрічається часто, і розробники таких схем виконують його кожен по-своєму. Нижче описаний модуль індикації, який цілком може стати універсальним рішенням, що значно спростить розробку нових пристроїв [4].

Нижче приведена принципова електрична схема модуля індикації на два знакомісця. Резистори R1.. R5 визначають стан вхідних сигналів при відключенні модуля від микроконтроллерной схеми і служать «навантаженням» ліній управління індикатором. Резистори, що гасять R6 R21 визначають струм кожного сегмента при його включенні, який в усякому разі не повинен перевищувати максимально допустимий для виходів мікросхеми SN74HC595D. Установка перемички J1 на замикання її середньої точки на загальний провід GND або провід живлення + Vcc дозволяє використовувати в схемі семисегментні світлодіодні індикатори як із загальним катодом, так і з загальним анодом. Конденсатори C1.. C3 запобігають збої від можливих зовнішніх перешкод з харчування.

На рис. 1.1 наведено зовнішній вигляд модуля індикації. Модуль розроблений під застосування світлодіодних семисегментних індикаторів виробництва фірми Kingbright серій SA05 - 11 (із загальним анодом) або SC05 - 11 (із загальним катодом) або їм аналогічних з висотою знака 12,7 мм. В результаті вийшов компактний модуль індикації, габаритні розміри якого не набагато більше розмірів самих індикаторів. Зробити це виявилося зовсім не складно, так як кожна мікросхема SN74HC595D в корпусі SOIC - 16 цілком уміщається під « своїм» індикатором з іншого боку двосторонній

друкованої плати. Відповідність виходів мікросхем і сегментів індикатора обрано виходячи з простоти трасування друкованих провідників на платі модуля. Також під поверхневий монтаж застосовані резистори і конденсатори типорозміром 0805. Габаритні розміри друкованої плати 43,2 х 22,9 мм. З одного боку плати встановлюються світлодіодні семисегментні індикатори L1 і L2, резистори R1.. R5 і конденсатор C1, а з іншого - всі інші елементи : мікросхеми D1 і D2, що гасять резистори R6.. R21, конденсатори C2 і C3, а також запаюється перемичка J1.

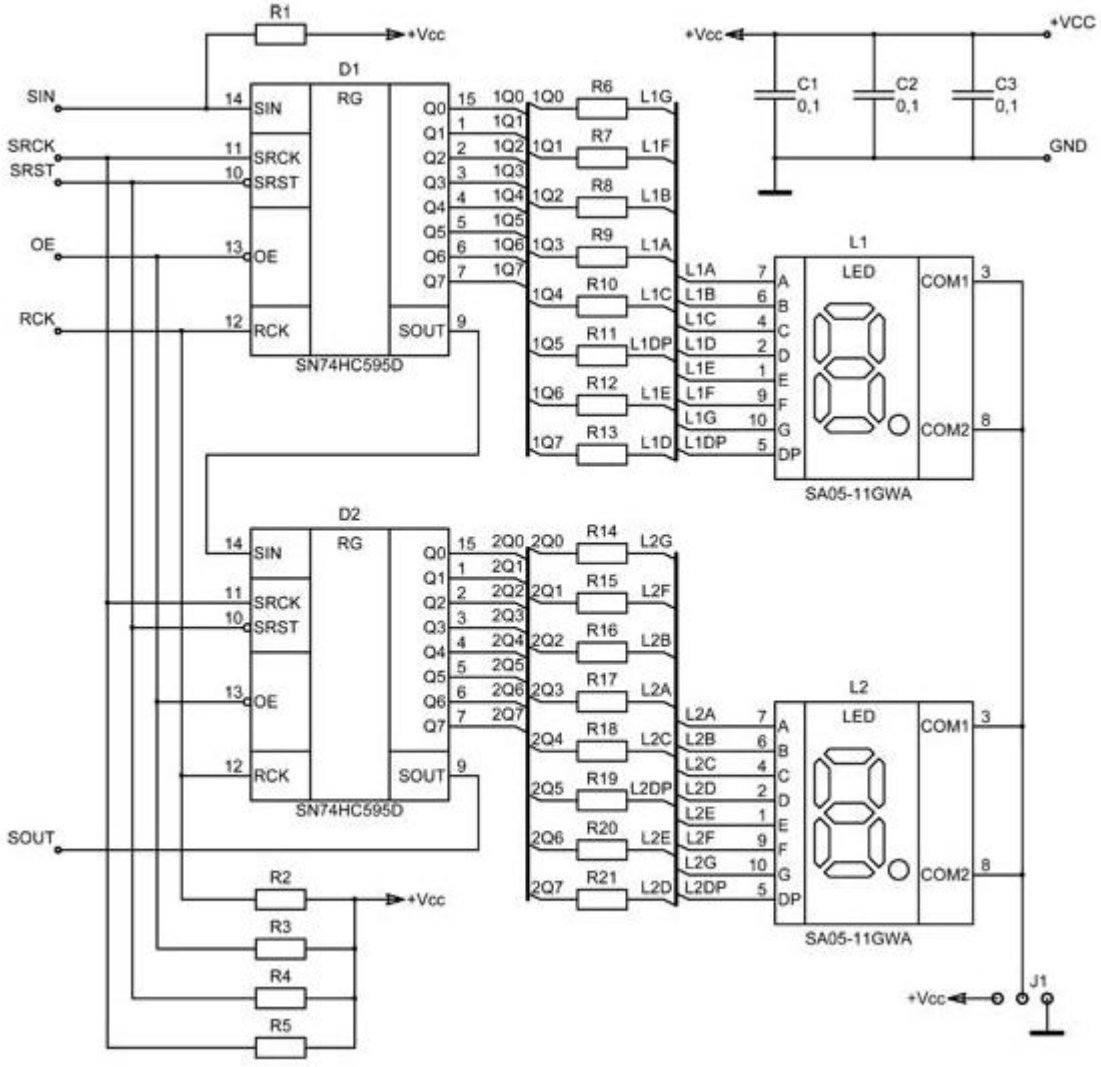

Рисунок 1.1 – Принципова електрична схема модуля індикації на два знакомісця

Модуль працює при напрузі живлення від +2,8 В до +5,5 В. Значення напруги живлення і тип індикаторів визначають і необхідний опір гасящих резисторів R6.. R21 виходячи з струму одного сегмента в межах 4... 4,5 мА.

Важливою перевагою описуваного модуля перед іншими схемами індикації є можливість каскадного з'єднання декількох таких модулів в ланцюжок, що можна використовувати, наприклад, для індикації відразу декількох цифрових значень в якому-небудь приладі при тому ж наборі ліній управління. А так як цих ліній трохи, і з'єднувальний кабель між модулем індикації і микроконтроллерной схемою містить всього кілька проводів, то модуль можна вільно розміщувати в будь-якому зручному місці корпусу приладу. Приклад підключення такого модуля індикації за допомогою кабелю завдовжки близько 70 см показаний на наведеному на фото нижче, опір « навантажувальних » резисторів R1.. R5 при цьому становило 10 кОм і збоїв у роботі не спостерігалося.

1.2 Динамічна індикація

Індикатори зазвичай розташовують у місцях, зручних для перегляду інформації, яка відображається на них. Решта цифрова схема може розташовуватися на інших друкованих платах. При збільшенні кількості індикаторів збільшується кількість провідників між платою індикаторів і цифровий платою. Це призводить до певних незручностей розробки конструкції та експлуатації апаратури. Ця ж причина призводить до збільшення її вартості [4].

Кількість з'єднувальних провідників можна зменшити, якщо змусити індикатори працювати в імпульсному режимі. Людське око володіє інерційністю і якщо змусити індикатори відображати інформацію по черзі з досить великою швидкістю, то людині буде здаватися, що всі індикатори відображають свою інформацію безперервно. В результаті можна по одним і тим же провідникам по черзі передавати отображаемую інформацію.

Зазвичай достатньо частоти оновлення інформації 50 Гц, але краще збільшити цю частоту до 100 Гц.

Давайте розглянемо структурну схему включення семисегментних світлодіодних індикаторів, наведену на рис. 1.2. Ця схема може забезпечити динамічну індикацію видаваної цифрової інформації.

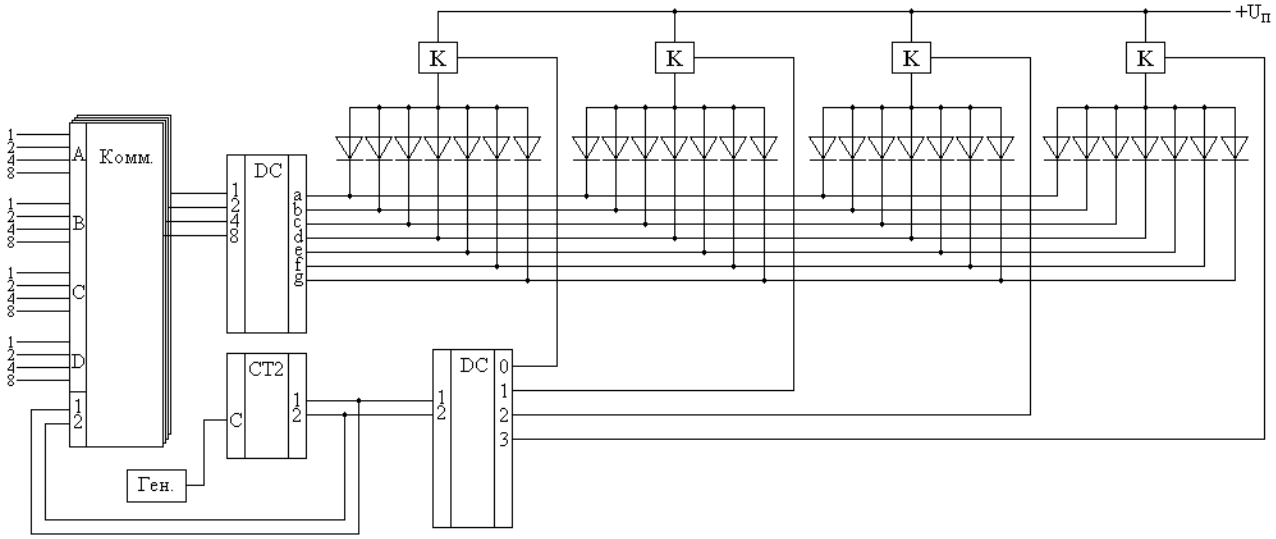

Рисунок 1.2 – Структурна схема включення семисегментних світлодіодних індикаторів.

У схемі, наведеній на малюнку 1, відображаються чотири цифрових розряду. Кожен розряд короткочасно підключається до свого входу комутатора. Генератор служить для завдання швидкості оновлення інформації на індикаторах. Двійковий лічильник послідовно формує чотири стану схеми, а дешифратор через ключі забезпечує почергову подачу живлення на семисегментні індикатори.

В результаті, коли комутатор подає двійковій- десятковий код з входу А на входи семисегментного дешифратора, то цей код відображається на індикаторі HL1. Коли комутатор подає на входи семисегментного дешифратора двійковій- десятковий код з входу В, то цей код відображається на індикаторі HL2, і так далі, по циклу.

Швидкість оновлення інформації в розглянутій схемі буде в чотири рази менше частоти генератора. Тобто для того, щоб отримати частоту мерехтіння індикаторів 100 Гц, потрібно частота генератора 400 Гц.

У скільки ж разів ми в результаті зменшили кількість з'єднувальних провідників ? Це залежить від того, де ми проведемо розтин схеми. Якщо ми на платі індикації залишимо тільки індикатори, то для їх роботи буде потрібно 7 інформаційних сигналів для сегментів і чотири комутуючих сигналу. Всього 11 провідників. У статичній схемі індикації нам було б потрібно 7 × 4 = 28 провідників. Як бачимо, виграш у наявності. При реалізації 8 -ми розрядного блоку індикації виграш буде ще більше.

Ще більший виграш буде, якщо перетин схеми провести по входах індикаторів. У цьому випадку для чотирирозрядний блоку індикації буде потрібно тільки шість сигнальних провідників і два провідники живлення схеми. Однак така точка перетину схеми динамічної індикації застосовується дуже рідко.

Тепер давайте розрахуємо струм, що протікає через кожен сегмент світлодіодного індикатора при його світінні. Для цього скористаємося еквівалентною схемою протікання струму по одному із сегментів індикатора. Дана схема наведена на рис. 1.3.

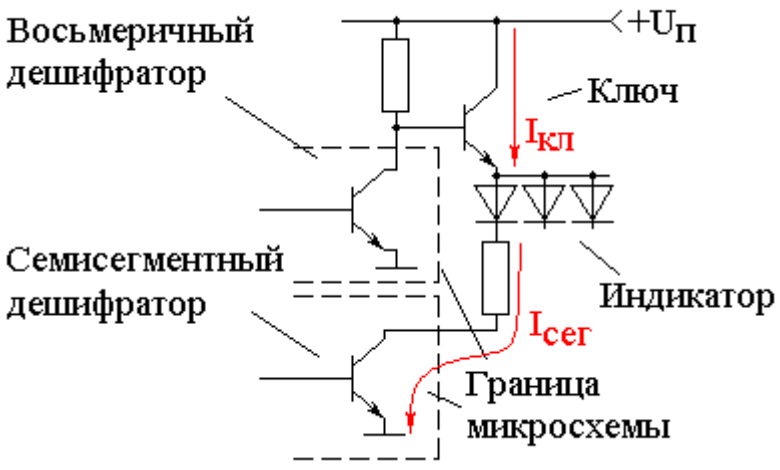

Рисунок 1.3 – Еквівалентна схема одного із сегментів індикатора.

Як уже згадувалося раніше, для нормальної роботи світлодіода потрібно струм від 3 до 10 мА. Задамося мінімальним струмом світлодіода 3 мА. Однак при імпульсному режимі роботи яскравість світіння індикатора падає в N раз, де коефіцієнт N дорівнює скважности імпульсів струму, що подаються на цей індикатор.

Якщо ми збираємося зберегти ту ж яскравість світіння, то потрібно збільшити величину імпульсного струму, що протікає через сегмент, в N разів. Для восьмирозрядного індикатора коефіцієнт N дорівнює восьми. Нехай спочатку ми вибрали статичний струм через світлодіод, рівний 3 мА. Тоді для збереження тієї ж яскравості світіння світлодіода в восьмирозрядному індикаторі буде потрібно імпульсний струм:

$$I_{seg\_din} = I_{seg\_stat} \cdot N = 3 \cdot 8 = 24(mA). \qquad (1.1)$$

Такий струм насилу зможуть забезпечити тільки деякі серії цифрових мікросхем. Для більшості ж серій мікросхем потрібні підсилювачі, виконані на транзисторних ключах.

Тепер визначимо струм, який буде протікати через ключ, коммутирующий харчування на окремі розряди восьмирозрядного блоку індикації. Як це видно зі схеми, наведеної на малюнку 2, через ключ може протікати струм будь-якого сегменту індикатора. При відображенні цифри 8 буде потрібно запалити всі сім сегментів індикатора, значить імпульсний струм, що протікає в цей момент через ключ, можна визначити наступним чином:

$$I_{кл} = I_{seg\_din} \cdot N_{seg} = 24 \cdot 7 = 168(mA). \qquad (1.1)$$

У радіоаматорських схемах зустрічаються рішення, де комутуючий струм береться безпосередньо з виходу дешифратора, який не може видати струм більше 20 мА.

Тепер розглянемо принципову схему отриманого блоку індикації. Вона наведена на рис.1.4.

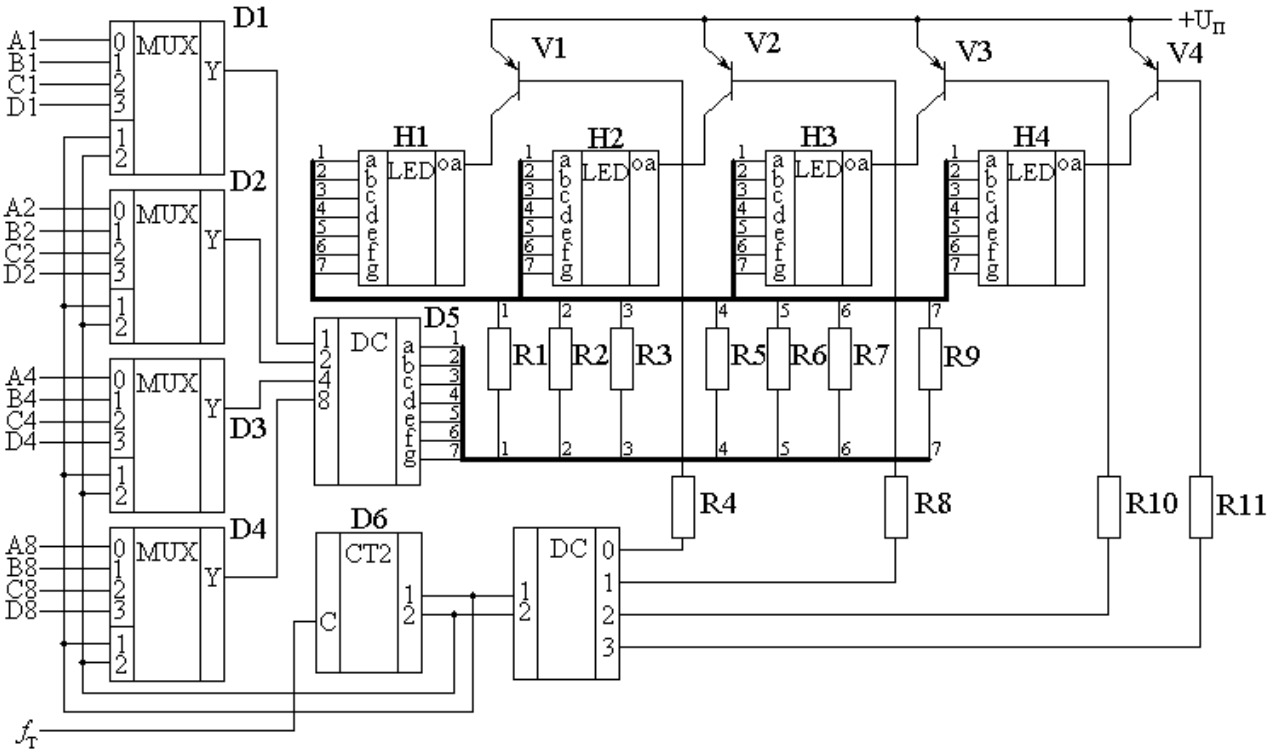

Рисунок 1.4 – Принципова схема блоку індикації.

Тепер, після того, як ми отримали схему динамічної індикації, можна обговорити її переваги і недоліки. Безсумнівним достоїнством динамічної індикації є мала кількість з'єднувальних проводів, що робить її незамінною в деяких випадках, таких як робота з матричними індикаторами.

Як недолік слід привести наявність великих імпульсних струмів, а так як будь провідник є антеною, то динамічна індикація служить потужним джерелом перешкод. Ще одним шляхом поширення перешкод є джерело живлення.

Звернемо увагу, що фронту у комутуючих імпульсів дуже короткі, тому їх гармонійні складові перекривають діапазон радіочастот аж до ультракоротких хвиль.

Отже, застосування динамічної індикації дозволяє мінімізувати кількість з'єднувальних проводів між цифровим пристроєм і індикатором, але є при цьому потужним джерелом перешкод, тому її застосування в радіоприймальних пристроях небажано.

Якщо з якихось причин, наприклад, необхідність застосування матричних індикаторів, доводиться використовувати динамічну індикацію, то потрібно вжити всіх заходів з придушення перешкод.

Як заходи з придушення перешкод від динамічної індикації можна назвати екранування блоку, з'єднувального кабелю і плат. Використання мінімальної довжини з'єднувальних проводів, застосування фільтрів по живленню. При екрануванні блоку, можливо, буде потрібно екранувати і самі індикатори. При цьому зазвичай використовується металева сітка. Ця сітка одночасно може збільшити контрастність відображуваних символів.

1.3 Опис мікроконтролера

ATMega -8 - 8 - розрядний КМОП мікроконтролер, заснований на архітектурі Atmel AVR. Контролер виконує більшість інструкцій за 1 такт, тому обчислювальна потужність контролера дорівнює 1MIPS на 1 Мгц. Блок - схема процесора відображено [5].

Мікроконтролер має RISC - архітектуру, але формат команди двохоперандних, за один такт може бути звернення тільки до двох регістрів. Контролер містить 32 регістра, які можуть рівноправно використовуватися в арифметичних операціях.

Основні апаратні характеристики мікроконтролера :

- 8 Кбт флеш -пам'яті команд ;
- 512 байт електрично програмованої пам'яті;
- 1 Кбайт статичної пам'яті ;
- 23 лінії введення/виводу загального призначення;
- 32 Рона ;
- Три багатоцільових таймер - лічильника з режимом порівняння;
- Підтримка внутрішніх і зовнішніх переривань ;
- Універсальний асинхронний адаптер;

- Байт -орієнтований двухпроводной послідовний інтерфейс;

- 6/8 канальний АЦП з точністю 8 і 10 двійкових розрядів ;

- Сторожовий таймер ;

- Послідовний порт SPI;

- Розширені режими управління енергоспоживанням.

Ядро мікроконтролера займається виконанням команд програми. Ядро включає в себе елементи гарвардської архітектури з незалежними шинами, тому вибірка інструкцій проводиться незалежно від операцій в АЛП.

АЛУ виробляє арифметико - логічні операції між регістрами (без обмежень) і регістром і константою. Кожна арифметико - логічна операція встановлює прапори під флаговая регістрі.

- I - прапор дозволу переривання ;

- T - прапор - хранитель біта - встановлюється і аналізується лише спецінструкціямі ;

- H - прапор додаткового перенесення з 3- го розряду в 4 -й ;

- S - прапор знака результату ;

- V - прапор переповнення ;

- N - прапор негативного результату операції ;

- Z - прапор нуля ;

- C - прапор переносу.

Як видно, 6 старших регістрів утворюють реєстрові пари - індексні регістри. Ядро процесора за допомогою цих регістрів допускають Автоінкрементний, автодекрементную адресацію і адресацію з малим зміщенням.

Команда контролера займає 16 або 32 біта, тому флеш - пам'ять програм організована у вигляді масиву 4К 16 -ти бітових слів. Флеш- пам'ять розділена на дві секції - область додатків і область завантаження (що знаходиться в старших адресах). Флеш -пам'ять витримує до 10.000 циклів перезапису.

Молодші 1120 адрес адресують регістри АЛУ, регістри введення / виводу і собствено оперативну пам'ять. Перші 96 адрес ставляться до сторонам і регістрів введення / виводу, решта 1024 байти - пам'ять.

Додатково сушествует електрично програмована пам'ять, що містить 512 байт і що витримує 100.000 циклів перезапису. Доступ до пам'яті здійснюється через спеціальні регістри введення / виводу.

0 -й таймер загального призначення має 8 - бітовий лічильник, з 10-бітовим додатковим дільником частоти. Таймер може генерувати переривання по переповнення, або по досягненню значення.

1 -й таймер має 16 - бітовий лічильник. Він може бути використаний для генерації сигналів із змінною шпаруватістю (широтно модульовані імпульси, генерації частоти і визначення часу надходження зовнішніх подій.

2 регістра порівняння значення таймера можуть використовуватися для генерації імпульсів із змінною шпаруватістю. Вхідний регістр використовується для завантаження значення таймера в момент надходження зовнішньої події.

Таймера можуть тактіроваться різними сигналами.

2 -й таймер є 8 -ми бітним, і може генерувати частоту і сигнали із змінною шпаруватістю, генерувати переривання по переповнення і досягненню значення.

Послідовний периферійний інтерфейс.

Особливості інтерфейсу :

- Дуплексная передача, використовується трехпроводная зв'язок ;

- Операції в режимах задатчика / виконавця ;

- Програмовані режими передачі: перший або старший або молодший біт ;

- 7 швидкостей передачі ;

- Генерація переривань.

1.3.1 Порти введеня-виведення

Більшість виводів AVR-мікроконтролерів мають також альтернативні призначення. Це аналогові лінії компаратора і АЦП, цифрові модулів SPI, TWI, USART і т.д. Схема їх розташування виводів показана на рис 1.5, 1.6 залежно від типу корпуса [6-10].

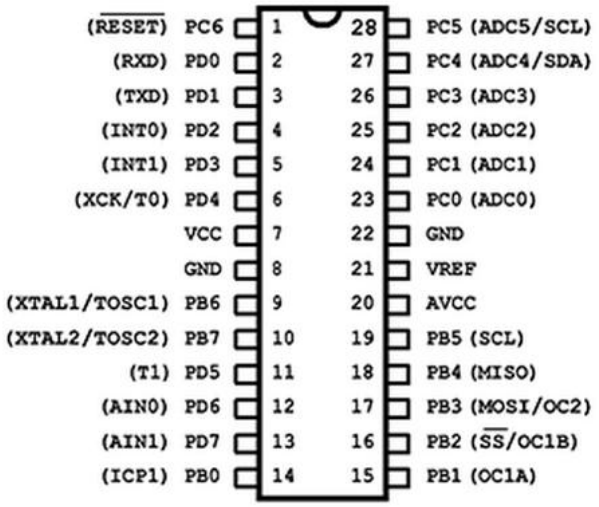

Рисунок 1.5 – Розташування виводів DIP корпус.

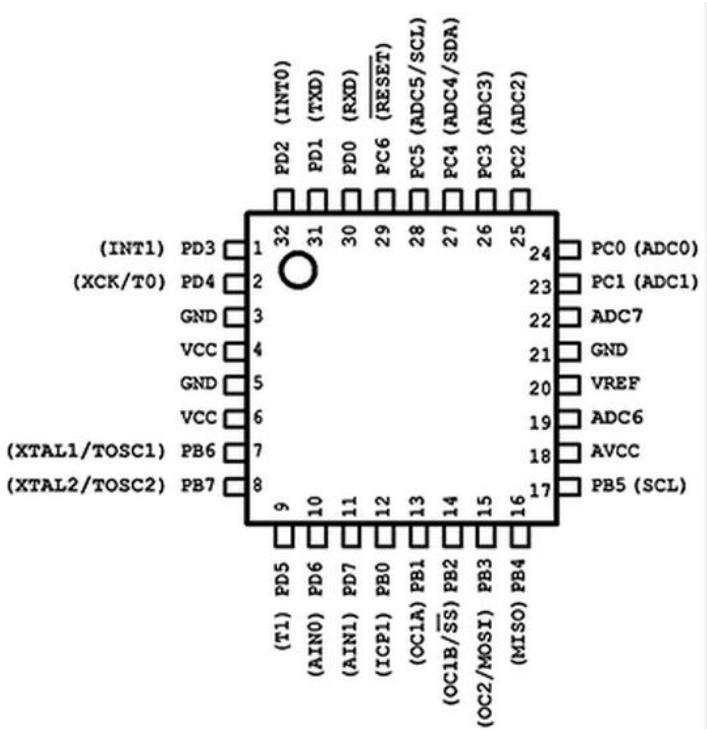

Рисунок 1.6 – Розташування виводів SMD корпус.

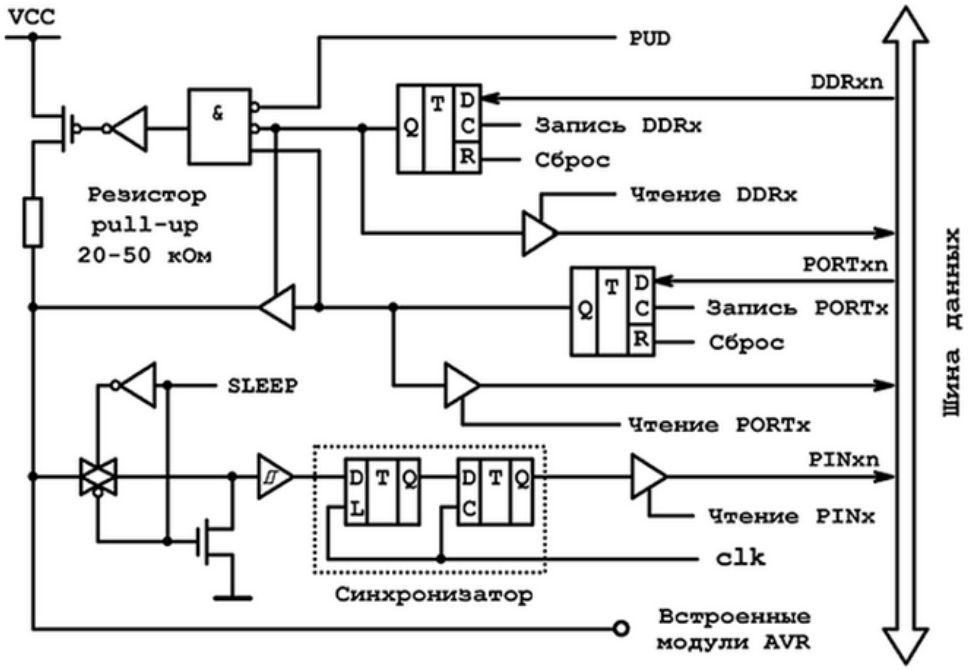

Рисунок 1.6 – Внутрішня схема лінії порту введення-виведення

Спрощена внутрішня схема лінії порту введення-виведення наведена на рис. 1.6. Кожним портом мікроконтролера управляють 3 РВВ. Це DDRx, PORTx і PINx (x - назва порта : A, B, C і т.д.). Регістри портів A, B, C, D всіх моделей знаходяться в першій половині адресного простору вводу-виводу і тому допускають прямі операції над окремими їх битами. У цифрових портах AVR реалізована дійсна функціональність "читання - модифікація - запис ".

Регістр DDRx відповідає за напрямок передачі даних. Запис лог.1 в розряди DDRx налаштовує лінії на висновок, а запис лог.0, відповідно, на введення (рис 1.7):

```
1  ldi  R16,0b11100011 ;настроить на ввод линии 2,3,4 и
2  out  DDRB,R16       ;на вывод линии 0,1,5,6,7 порта B
3
4  sbi  DDRB,0  ;настроить на вывод линию 0 порта B
5  cbi  DDRB,1  ;настроить на ввод линию 1 порта B
```

Рисунок 1.7 – Скріншот програм налаштування лінії порту введення-виведення

Регістр PORTx має подвійне призначення. Якщо лінії порту налаштовані на висновок, то вміст PORTx визначає логічне стан виводів

порту. Для ліній налаштованих на введення значення PORTx визначає стан внутрішнього подтягивающего (pull-up) резистора до шини харчування. У цьому випадку при рівні лог.1 в розрядах PORTx резистор підключений, а при рівні лог.0 - відключений і лінії переведені в високоімпедансное стан (z-стан) рис. 1.8.

```
ldi  R16,0b11110000  ;настроить на вывод линии 4…7
out  DDRC,R16        ;и на ввод линии 0…3 порта C
ldi  R16,0b11111111  ;выставить на линиях 4…7 порта C уровень лог.1
out  PORTC,R16       ;и подключить к линиям 0…3 порта C pull-up резисторы

sbi  DDRC,0   ;настроить на вывод линию 0 порта C
cbi  PORTC,0  ;выставить на линии 0 порта C уровень лог.0

cbi  DDRC,1   ;настроить на ввод линию 1 порта C
cbi  PORTC,1  ;перевести линию 1 порта C в z-состояние
```

Рисунок 1.8 – Скріншот програм налаштування лінії порту введення-виведення

Регістр PINx призначений для зчитування рівнів сигналу з висновків мікроконтролера. Природно, що необхідність в цьому існує тільки для ліній налаштованих на введення (для ліній налаштованих на висновок вміст PINx повторює стан вихідного регістра PORTx). Розряди PINx не роблять ні якого впливу на стан висновків, а сам регістр доступний тільки для читання. Логічний рівень вхідної лінії фіксується в тригері - клямці в кожному циклі тактової частоти. Таким чином, реальне значення сигналу при зчитуванні може мати відставання порядку 0.5... 1.5 машинних циклів. Приклад використання на рис. 1.9.

```
cbi  PORTD,0  ;настроить на ввод линию 0 порта D
sbi  PORTD,0  ;подключить к линии 0 порта D pull-up резистор
nop           ;задержка 1 цикл для установки режима
sbis PIND,0   ;считать состояние линии 0 порта D
rjmp ulo      ;если лог.0, то перейти на метку ulo
rjmp uhi      ;если лог.1, то перейти на метку uhi
```

Рисунок 1.9 – Скріншот програм налаштування лінії порту введення-виведення

У мікроконтролерів AVR є можливість управляти pull-up резисторами відразу на всіх його висновках. За цю властивість відповідає біт PUD з PBB SFIOR або MCUCR. При установці розряду PUD забороняється підключення всіх резисторів. При PUD = 0 (значення після скидання) стан внутрішніх резисторів визначається станом регістра PORTx.

1.3.2 Система команд

Мікроконтролери AVR мають систему скороченого набору команд RISC, хоча цілком і не повністю потрапляють під це визначення [12].

Система RISC увазі повну симетрію між ресурсами пам'яті різного типу. Це, зокрема, дозволяє звертатися до регістрів, портам і пам'яті даних одними і тими ж командами, що й обумовлює їх невелику кількість. Однак, не дивлячись на те, що адресний простір пам'яті AVR дійсно безперервно, все ж три різних його області використовуються тільки для своїх специфічних цілей. РОН - переважно для математичних операцій, PBB - для управління процесором, ОЗУ - тільки як сховище інформації. У зв'язку з цим існують групи команд як для роботи з кожним видом пам'яті окремо, так і для пересилання даних з однієї області пам'яті в іншу. Тому й кількість команд AVR досить велике. У фірмовій документації, де багато говориться про ортогональность ядра, в першу чергу мається на увазі повна рівноправність саме РОН.

Система команд лінійки ATtiny є підмножиною системи команд старшого сімейства ATmega. У ряді старих моделях ATtiny можуть бути відсутні деякі апаратні вузли (індексні регістри X, Y, програмний стек, пам'ять ОЗУ тощо) і, відповідно, відсутні пов'язані з ними команди. Система ж команд ATtiny пізнішого часу випуску практично аналогічна сімейства ATmega. Головна відмінність обох сімейств - відсутність вбудованого помножувача у ATtiny (відсутність групи команд множення).

Різні моделі ATtiny можуть мати (90... 120) команд. ATmega підтримують (130... 135) інструкцій. Так заявлено в специфікаціях Atmel. Але фактичне число, насправді, значно менше.

Це пов'язано з тим, що в асемблері AVR вбудований ряд макровизначень, еквівалентних реальним командам, але мають інший символічний вигляд. Так, наприклад, у команди ori Rd, K існує команда двійник sbr Rd, K, яка виконує теж дія (Rd = Rd OR K) і має такий же код операції. Аналогічні псевдокоманди існують і для різних випадків застосування bset s, bclr s, brbs s, k, brbc s, k, і мн. ін.

1.3.3 Способи адресації

Більшість команд асемблера використовують різні комірки пам'яті мікроконтролера і, відповідно, містять крім коду операції (КОП) також їх адреси. Залежно від того в якому вигляді в команді зберігається адреса розрізняють два способи адресації : пряму і непряму. У першому випадку адресу комірки заданий явно, а по-другому він знаходиться в одному з регістрів - покажчиків (у AVR це 16 - розрядні регістри X, Y, Z). У мікропроцесорів різного типу, обидва способи адресації можуть мати безліч варіацій. Нижче наведено характерні тільки для мікроконтролерів AVR [5].

У команді присутня адреса регістра приймача небудь джерела результату. Прикладами команд можуть служити inc Rd, dec Rd, lsl Rd, lsr Rd і т.д. Адресацію, де в команді крім адреси регістра знаходиться ще і константа (ldi Rd, K, ori Rd, K і т.д.), називають також безпосередній, а в разі збереження / відновлення даних в стеку (push Rr і pop Rd) - стековой. Адреса РОН знаходиться в межах 0... 31 (у командах з безпосередньою адресацією 0... 15).

Команди даного типу містять адреси двох РОН, один з яких є джерелом, а другий приймачем результату в арифметичних операціях, а також операціях пересилання. Приклади команд : mov Rd, Rr, add Rd, Rr, sub Rd, Rr, and Rd, Rr і т.д. Адреси обох регістрів лежать в межах 0... 31, але в

деяких командах множення можуть використовуватися тільки Рони 16... 31 (muls Rd, Rr і fmuls Rd, Rr) або 16... 23 (mulsu Rd, Rr і fmulsu Rd, Rr).

Пряму адресацію регістра введення -виведення у AVR використовують команди двох типів. Це копіювання РВВ в РОН in Rd, P і пересилання в протилежному напрямку out P, Rr. В обох випадках можуть бути використані будь регістри загального призначення 0... 31 і регістри введення -виведення 0... 63. Пряма адресація ОЗУ зустрічається в командах lds Rd, k і sts k, Rr. Перша інструкція пересилає байта з SRAM мікроконтролера в один з РОНов, другий копіює вміст Рона в осередок SRAM. В обох командах під поле адреси комірки пам'яті відводиться 16 бітів, а значить, є можливість безпосередньо звертатися до будь-якого адресою SRAM з діапазону 0... 65535. Інструкції працюють з усіма сторонами 0... 31 і мають розмір в 2 слова (4 байти). Проста непряма адресація застосовується для копіювання даних з SRAM в РОН однієї з команд ld Rd, X / Y / Z, а також для пересилання в зворотному напрямку st X / Y / Z, Rr. У 2 - байтових регістрах - покажчиках X, Y, Z міститься адресу осередки приймача небудь джерела в діапазоні 0... 65535. Цей вид адресації подібний простий непрямої адресації за винятком одного відрізняючи. Перед виконанням операцій пересилки значення індексних регістрів X, Y, Z апаратно зменшується на одиницю, що і символізує знак "-" в командах ld Rd, -X/-Y/-Z і st -X/-Y/-Z, Rr.

При непряма адресації з постинкрементом значення покажчиків X, Y, Z апаратно збільшується на одиницю (знак " + " перед покажчиками) після пересилання байта командами ld Rd, X + / Y + / Z + і st X + / Y + / Z +, Rr.

Відносна непряма адресація також використовується для пересилання даних між РОН і SRAM. Однак адресу комірки пам'яті визначається тут як сума вмісту покажчиків Y, Z і фіксованого зсуву q. Для пересилання байта з SRAM в РОН застосовуються команди ldd Rd, Y + q / Z + q, а для пересилки в зворотному напрямку st Y + q / Z + q, Rr. Величина q може лежати в межах 0... 63.

2 РОЗРОБКА СХЕМ СТРУКТУРНОЇ, ПРИНЦИПОВОЇ ТА АЛГОРИТМУ РОБОТИ ПРОГРАМИ

Для забезпечення алгоритму роботи мікропроцесорного пристрою сигналізації семисегментного індикатора була розроблена структурна схема (рисунок 2.1).

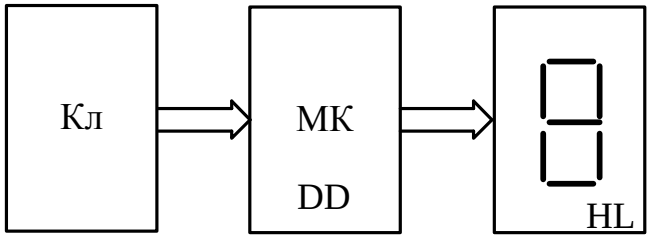

Рисунок 2.1 – Структурна схема мікропроцесорного пристрою світлової індикації

Згідно структурної схеми було розроблена принципова схема мікропроцесорного пристрою сигналізації семисегментного індикатора (рисунок 2.2) та алгоритм роботи мікропроцесорного пристрою (рисунок 2.2).

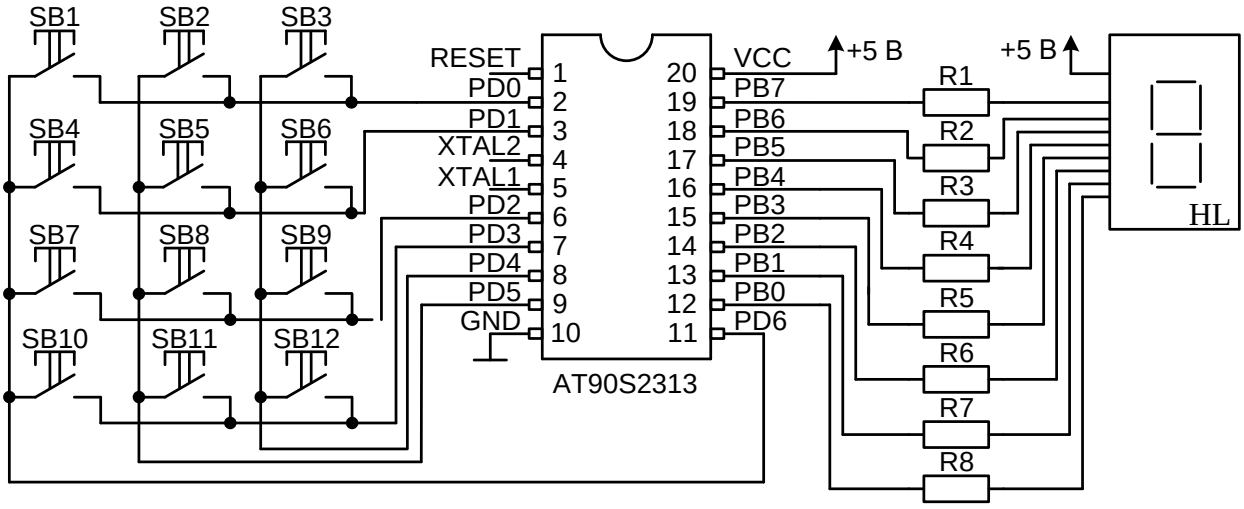

Рисунок 2.2 – Схема електрична принципова мікропроцесорного пристрою світлової сигналізації

Умовні позначення на рисунку 2.2: SB1 – кнока клавіатури «0»; SB2 – кнока клавіатури «1»; SB3 – кнока клавіатури «2»; SB4 – кнока клавіатури «3»; SB5 – кнока клавіатури «4»; SB6 – кнока клавіатури «5»; SB7 – кнока клавіатури «6»; SB8 – кнока клавіатури «7»; SB9 – кнока клавіатури «8»; SB10 – кнока клавіатури «9»; SB11 – кнока клавіатури «A»; SB12 – кнока клавіатури «B»; R1-R8 – активні опори; HL – семисигментний індикатор.

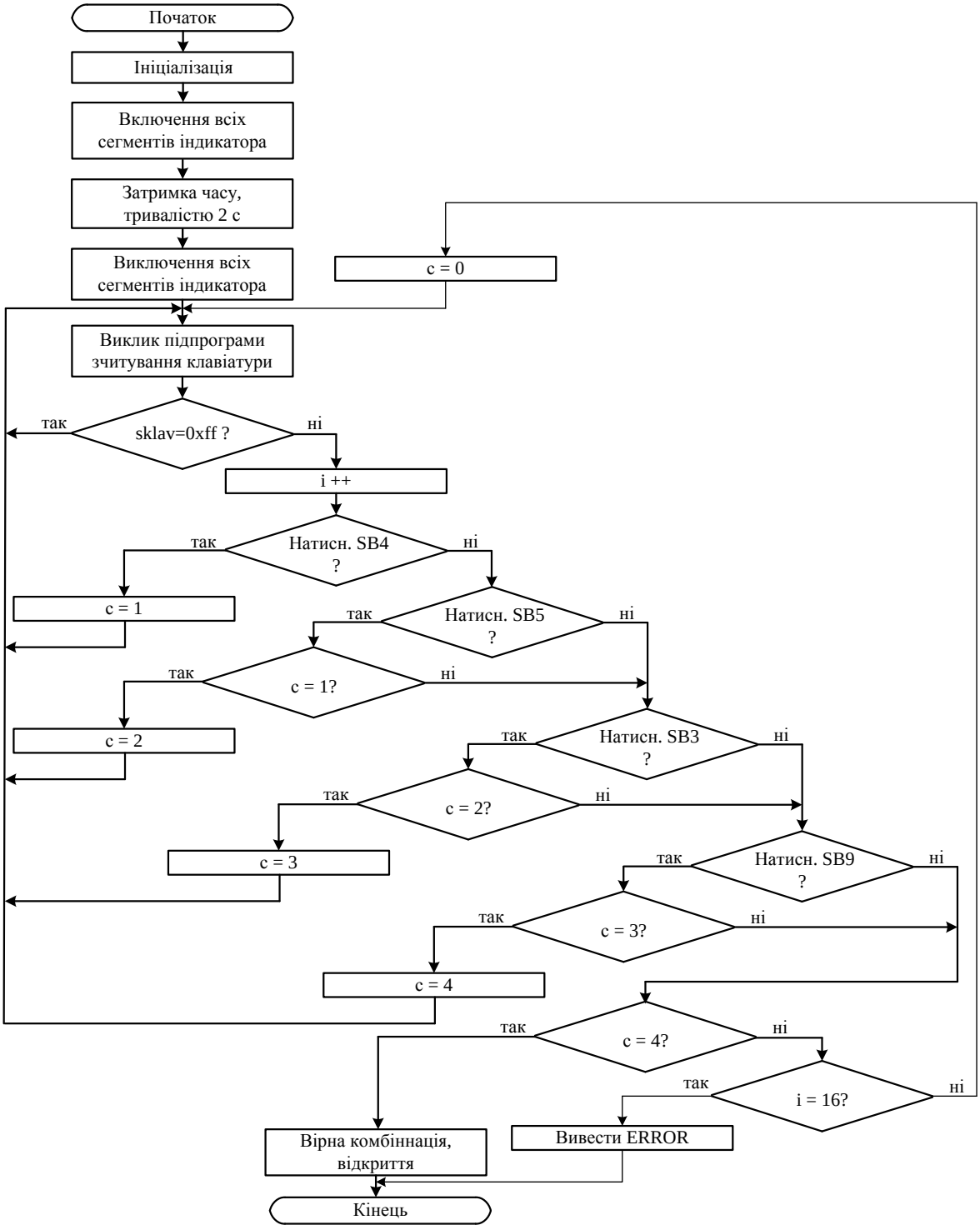

Рисунок 2.3 – Алгоритм роботи мікропроцесорного пристрою

Алгоритм роботи підпрограми зчитування матричної клавіатури приведений на рисунку 2.4.

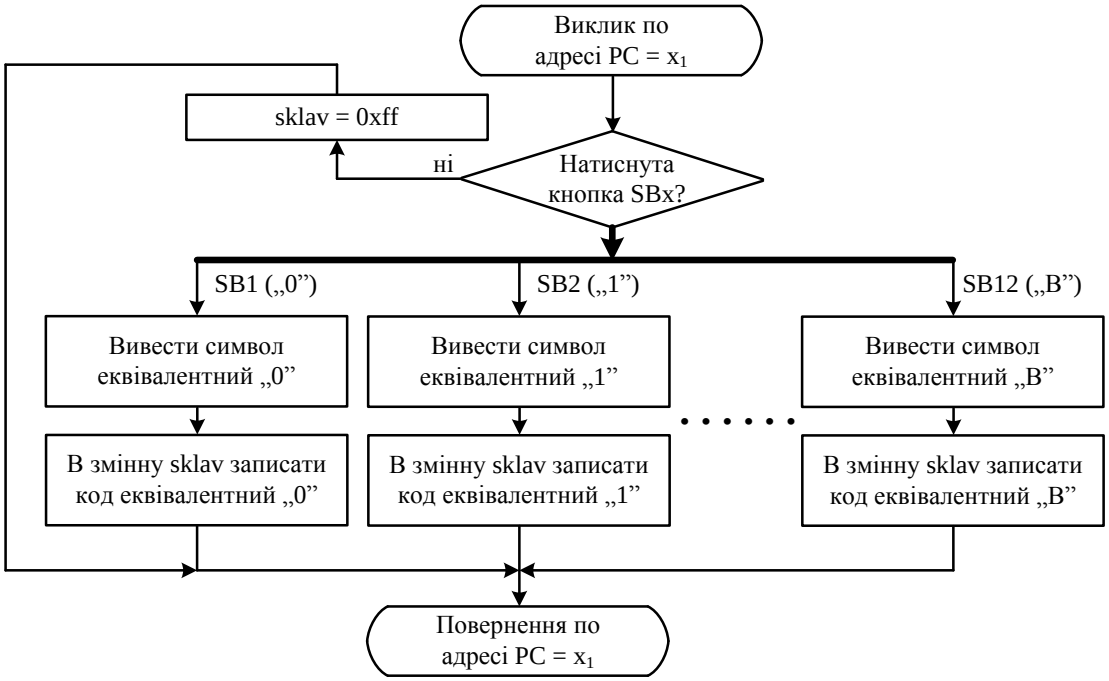

Рисунок 2.4 – Алгоритм роботи підпрограми зчитування клавіатури

3 РОЗРОБКА ПРОГРАМИ НА МОВІ ПРОГРАМУВАННЯ ASSEMBLER

Згідно розробленого алгоритму, було розроблено програму мікропроцесорного пристрою світлової індикації, повний варіант якої представлений в додатку А [13 - 15].

Спочатку відбувається підключення бібліотек та налаштування стеку.

```
.include "2313def.inc"
.def temp = r16
.def sklav = r17
.def const = r18
.def klav = r19
ldi r31,low(ramend) // налаштування стеку
out spl,r31
```

Фрагмент програми, який приведений нижче відповідає за формування кодів вивведення на семисегментний індикатор.

```
ldi temp, 0b00001111
out ddrd, temp
ldi temp, 0xff
out ddrb, temp
```

Даний фрагмент програми відповідає за почергове включення всіх можливих цифр на семисегмантному індикаторі.

```
ldi temp, 0b10000000 //включення всых сегментів індикатора на 2с
out PortB, temp
rcall wait2c
ldi temp, 0b11111111 //виключення всіх сегментів індикатора на 2с
out PortB, temp
```

```
ldi temp, 0
rcall wait2c
rcall wait
```

Оскільки клавіатура складається з чотирьох стрічок по три стовпці, то відповідно розроблена програма, яка почергово здійснюватиме аналіз по стрічкам.

Приведений фрагмент програми опрацьовує першу стрічку і три стовпці, відповідно кнопку SB1, SB2 та SB3.

```
ldi klav,0b00000001
out PortD,klav
sbic PinD,4
ldi sklav,0b10010000 //0
sbic PinD,5
ldi sklav,0b11111001 //1
sbic PinD,6
ldi sklav,0b10100100 //2
```

Приведений фрагмент програми опрацьовує другу стрічку і три стовпці, відповідно кнопку SB4, SB6 та SB6.

```
ldi klav,0b00000010
out PortD,klav
sbic PinD,4
ldi sklav,0b11000000 //3
sbic PinD,5
ldi sklav,0b10011001 //4
sbic PinD,6
ldi sklav,0b10010010 //5
```

Приведений фрагмент програми опрацьовує третю стрічку і три стовпці, відповідно кнопку SB7, SB8 та SB9.

```
ldi klav,0b00000100
out PortD,klav
sbic PinD,4
ldi sklav,0b10110000 //6
sbic PinD,5
ldi sklav,0b11111000 //7
sbic PinD,6
ldi sklav,0b10000000 //8
```

Приведений фрагмент програми опитує четверту стрічку і три стовпці, відповідно кнопку SB10, SB11 та SB12.

```
ldi klav,0b00001000
out PortD,klav
sbic PinD,4
ldi sklav,0b10000010 //9
sbic PinD,5
ldi sklav,0b10001000 //a
sbic PinD,6
ldi sklav,0b10000110 //e
```

Для врахування можливого дребезгу контактів, реалізована затримка часу тривалістю 80 мс після натиску кнопки.

```
drebezg:
        ldi r22, 50
    d:
    dec r22
    brne d
ret
```

До порта B підключенні індикатор, тому визивається програма. Якщо не наживається жодна кнопка, то в sklav записується значення ff, що відповідає вимкненню всіх індикаторів, оскільки семи сегментний індикатор зі спільним анодом, тобто керується нулями.

```
ldi sklav, 0xff
out PortB, sklav
rcall wait
out PortB, const
rcall wait
```

Якщо в результаті порівння стану порту B із константою виявиться що натиснута жодна з кнопок, то відбувається перехід до підпрограми лічильника, який інкрементує своє значення і викликає підпрограму затримки:

```
cpi sklav, 0xff
brne lich
```

Якщо в результаті виклику лічильника, його значення рівне 5 то відбувається виклик підпрограми що сигналізує про помилку:

```
ll:     cpi temp, 5
        breq error
error:
        ldi sklav, 0b10000110
        out portb, sklav
        rcall wait2c
        rjmp start
```

Даний фрагмент відповідає за розпізнавання натиснутої першої кнопки і виклику підпрограми kod1:

```
cpi sklav,0b11111001 //4
```

breq kod1 //перехід якщо дорівнює

sbrs r9,0

rjmp start

Якщо натиснута 6 кнопка відбувається виклик підпрограми kod2:

cpi sklav,0b10110000 //5

breq kod2

sbrs r9,1

rjmp start

Якщо натиснута 5 кнопка відбувається виклик підпрограми kod3:

cpi sklav,0b10010010 //3

breq kod3

sbrs r9,2

rjmp start

Якщо натиснута 9 кнопка відбувається виклик підпрограми kod4:

cpi sklav,0b10000010 //9

breq kod4

mov const, r8

cpi const, 0b10000010

brne start // перехід якщо не дорівнює

Підпрограма завантаження кодів виведення цифрового значення кнопки на індикатор:

l:

ldi sklav, 0b10110110

out portb, sklav

rcall wait2c

ldi sklav, 0b11111111

```
out portb, sklav
rcall wait2c
ldi sklav, 0b10110110
out portb, sklav
rcall wait2c
ldi sklav, 0b11111111
out portb, sklav
rcall wait2c
ldi temp, 0
mov r6, temp
mov r7, temp
mov r8, temp
rjmp start
```

Приведений фрагмент програми відповідає за виведення 1 на семисегментний індикатор:

```
kod1:
    mov r5, sklav
    ldi sklav, 0b00000001
    mov r9, sklav
    rjmp start
```

Приведений фрагмент програми відповідає за виведення 6 на семисегментний індикатор:

```
kod2:
    mov r6, sklav
    ldi sklav, 0b00000011
    mov r9, sklav
    rjmp start
```

Приведений фрагмент програми відповідає за виведення 5 на семисегментний індикатор:

```
kod3:
    mov r7, sklav
    ldi sklav, 0b00000111
    mov r9, sklav
    rjmp start
```

Приведений фрагмент відповідає за висвітлення всіх сегментів на індикаторі:

```
kod4:
    mov r8, sklav
    rjmp start
```

Після того як було здійснено відображення всіх сегментів індикатора, відбувається їх вимкнення після затримки часу, яка реалізується з допомогою програми wait.

```
wait2c:
    ldi r22,3
    labell:
    ldi r21, 255
    labell1:
    ldi r20, 255
    labell2:
    dec r20
    brne labell2
    dec r21
    brne labell1
    dec r22
    brne labell
```

4 РОЗРОБКА ПРОГРАМИ НА МОВІ ПРОГРАМУВАННЯ С ТА МОДЕЛЮВАННЯ В PROTEUS VSM

Згідно розробленого алгоритму, було розроблено на мові С програму мікропроцесорного пристрою світлової індикації, повний варіант якої представлений в додатку Б [16 - 19].

Фрагмент програми, який представлений нижче описує підключення бібліотек.

```
#include <90s2313.h>
#include <delay.h>
```

Мова С дозволяє описати функцію, яка буде визиватись в головні програмі. Вона повертає цілі числа і не має аргументів.

```
int klava()
```

Сама функція sklav опрацьовує стрічки і стовпці клавіатури.

Приведений фрагмент програми опитує першу стрічку і три стовпці, відповідно кнопку SB1, SB2 та SB3.

```
PORTD = 0b00000001;
if (PIND.4 == 1){sklav = 0b10010000;}
if (PIND.5 == 1){sklav = 0b11111001;}
if (PIND.6 == 1){sklav = 0b10100100;}
```

Фрагмент програми (нижче по тексту) опитує другу стрічку і три стовпці, відповідно кнопку SB4, SB6 та SB6.

```
PORTD = 0b00000010;
if (PIND.4 == 1){sklav = 0b11000000;}
if (PIND.5 == 1){sklav = 0b10011001;}
if (PIND.6 == 1){sklav = 0b10010010;}
```

Фрагмент програми (нижче по тексту) опитує третю стрічку і три стовпці, відповідно кнопку SB7, SB8 та SB9.

```
PORTD = 0b00000100;
if (PIND.4 == 1){sklav = 0b10110000;}
if (PIND.5 == 1){sklav = 0b11111000;}
if (PIND.6 == 1){sklav = 0b10000000;}
```

Фрагмент програми (нижче по тексту) опитує четверту стрічку і три стовпці, відповідно кнопку SB10, SB11 та SB12.

```
PORTD = 0b00001000;
if (PIND.4 == 1){sklav = 0b10000010;}
if (PIND.5 == 1){sklav = 0b10001000;}
if (PIND.6 == 1){sklav = 0b10000110;}
```

В даному фрагменті програми обявляються змінні.

```
char mas[4]={0,0,0,0};
int i=0;
int c=0;
int k=0;
int l=0;
int a=0xff;
```

В головній програмі визивається функція klava(), яка передає значення на порт B, а також реалізовується затримка у 2 секунди. Дана порграма забезпечує при подачі живлення на мікроконтролер загоряються всі сегменти семисегментного індикатора та гаснуть через 2с.

```
while (1)
{   a=klava();
 if (a!=0xff)
 {
```

```
PORTB=a;
i++;
delay_ms(100);
```

Перевірка послідовного натиску кнопок 1-6-5-9 (номер кнопки відображається на індикаторі), після чого сегменти зову загоряються і гаснуть через 2 с. Результат переноситься в масив.

```
if (a==0b11111001) //4
{
  delay_ms(200);
  PORTB=0b11110111;
  mas[0]=a;
  c=1;
}
if (a==0b10100100 && c==1) //5
{
  delay_ms(200);
  PORTB=0b10110111;
  mas[1]=a;
  k=1;
}
if (a==0b10011001 && k==1) //3
{
  delay_ms(200);
  PORTB=0b11110111;
  mas[2]=a;
  c=2;
}
if (a==0b10010010 && c==2) //9
{
```

```
    delay_ms(200);
    PORTB=0b10110111;
    mas[3]=a;
        if (mas[0]==0b11111001 && mas[1]==0b10110000 &&
mas[2]==0b10010010 && mas[3]==0b10000010 && i==4)
        {
         PORTB=0b10101010;
         delay_ms(3000);
         PORTB=0b11111111;
         delay_ms(3000);
         i=0;
        }
}
```

Фрагмент програми, який відповідає за обнулення масиву.

```
if (i>4)
{
mas[0]=0;
mas[1]=0;
mas[2]=0;
mas[3]=0;
c=0;
k=0;
PORTB=0b10000110;
delay_ms(2000);
i=0;
l++;
if(l>3){while(1){}}
}
```

Для перевірки роботи розроблених програм було здійснене моделювання в середовищі Proreus VSM.

Proteus - середовище для проектування і налагодження електронних пристроїв, в т.ч. виконаних на основі мікроконтролерів різних сімейств. Надає можливості введення схеми в графічному редакторі, моделювання її роботи і розробки друкованої плати, включаючи тривимірну візуалізацію її збірки. Унікальною рисою середовища Proteus є можливість ефективного моделювання роботи різноманітних мікроконтролерів (PIC, 8051, AVR, HC11, ARM7/LPC2000 та ін) і налаштування мікропрограмного забезпечення. середовище PROTEUS має величезну бібліотеку електронних компонентів, а відсутні - можна зробити самостійно. Передбачена підтримка SPICE-моделей, які часто надаються виробниками електронних компонентів.

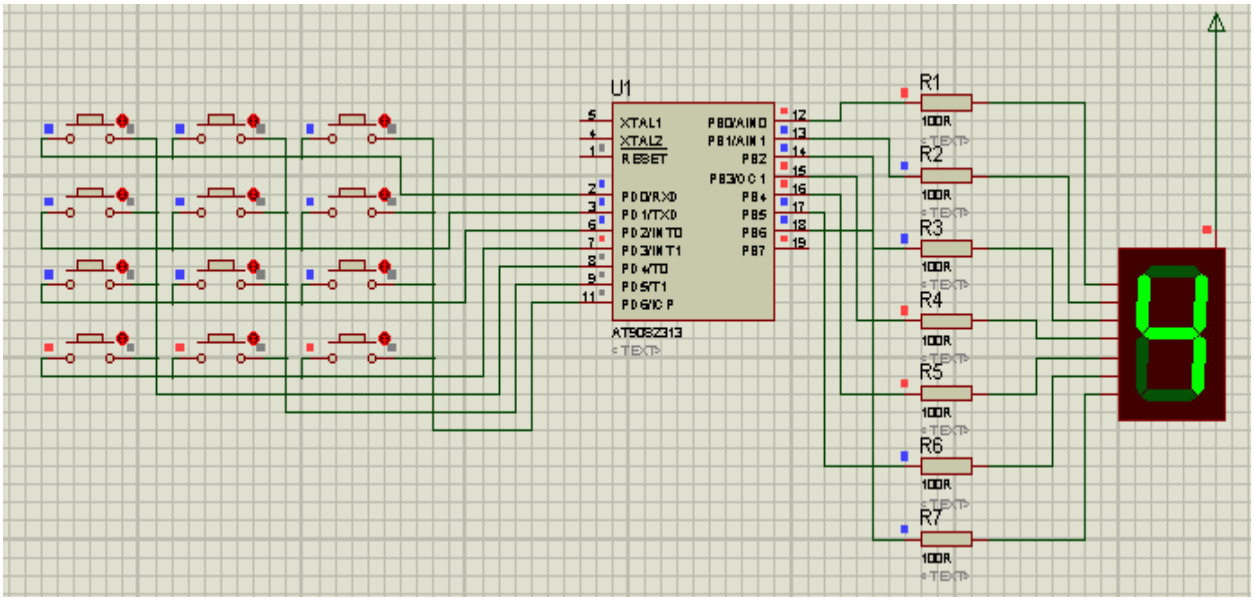

Рисунок 4.1 – Результат моделювання мікропроцесорного пристрою світлової індикаці

ВИСНОВКИ

В даній курсовій роботі було розроблено програми керування мікроконтролерним пристроєм світлової індикації на мові С та Asembler. Розроблені алгоритм роботи, структурна та електрична принципова схема. Здійснено комп’ютерне моделювання для перевірки розроблених програм. Моделювання засвідчило адекватність модельованих результатів.

Виконання даної курсової роботи довзолило закріпити отримані теоретичні знання з дисципліни «Обчислювальна техніка і програмування».

ПЕРЕЛІК ПОСИЛАНЬ

1. Цифровые и аналоговые интегральные микросхемы: справочник / Якубовский С. В., Ниссельсон Л. И., Кулешова В. И. [и др.] ; за ред. Якубовского С. В. – М.: Радио и связь, 1990. – 496 с.

2. Безуглов Дмитрий Анатолиевич. Цифровые устройства и микропроцессоры / Д. А. Безуглов, И. В. Калиенко. – [2–е изд.].– Ростов н/Д: Феникс, 2008. – 468 с. – ISBN 978-5-222-13917-2.

3. Чумаченко Ігор Володимирович. Мікроконтролерні прилади: структура і використання: навчальний посібник / І. В. Чумаченко , М. Д. Кошовий, В. В Лопатин. – Харків: Нац. аерокосмічний ун–т “ХАІ”, 2001. – 277 с.

4. Статична, динамічна індикація [Електронний ресурс]: Web-сайт. — Режим доступу: http://digteh.ru/digital/DinInd.php – Назва з екрану.

5. Мортон Дж. Микроконтроллеры AVR. Вводный курс / Дж. Мортон./Пер. С англ. – М.: Издательский дом «Додэка-XXI», 2006.–272 с. – ISBN 5-94120-096-X

6. Форум програмистов [Електронний ресурс]: Web-сайт. – Режим доступу: http://www.radioparty.ru/index.php/2010-12-03-15-00-02/44-runningled-at90s 1200 Xa – Назва з екрану.http://www.ru-coding.com/talk/subj.php?id=6843

7. Белов А.В. Создаем устройства на микроконтроллерах / А.В. Белов. – СПб.: Наука и Техника, 2007. – 304 с.: ил. ISBN 978-5-94387-363 – 3

8. Белов А.В. Самоучитель разработчика устройств на микроконтроллерах AVR / А.В. Белов. – СПб.: Наука и Техника, 2008. – 544 с.: ил. ISBN 978-5-94387-363 – 8

9. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / А.В. Евстифеев — М.: Издательский дом «Додэка-XXI», 2007. — 592 с: ил. (Серия «Программируемые системы»). ISBN 978-5-94120-090-0 mm

10. Гребнев В.В. Микроконтроллеры семейства AVR фипмы Atmel / В.В. Гребнев – М.: ИП Радиософт, 2002 – 176 с.: ил. ISBN 5-93037-091-5

11. Ревич Ю.В. «Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера» - Видання «БХВ-Петербург» 2011 р. - мова російська – 354 ст.

12. Система команд [Електронний ресурс]: Web-сайт. – Режим доступу: http://cxem.net/mc/book13.php – Назва з екрану.

13. Рудольф Марек «Ассемблер на примерах» (Ucime se Programovat v Iazyce Assembler pro PC) - Видання «Наука и техника» 2005 р. - російська – 240 ст.

14. Александр Крупник «Ассемблер. Самоучитель» - Видання «Питер» 2005 р. - мова російська – 240 ст.

15. Зубков С. В. «Assembler. Язык неограниченных возможностей» - Видання «ДМК» 1999 р. - мова російська – 640 ст

16. Шпак Ю.А. «Программирование на языке С для AVR и PIC микроконтроллеров. 1-е издание» - Видання «МК-Пресс» 2006 р. - мова російська – 487 ст.

17. Шпак Ю.А. «Программирование на языке С для AVR и PIC микроконтроллеров. 2-е издание» - Видання «Корона-Век, МК-Пресс» 2011 р. - мова російська – 544 ст.

18. Лебедев М.Б. CodeVision AVR. Пособие для начинающих / М.Б. Лебедев – М.: Додэка XXI, 2010 – 450с. ISBN 978-5-94120-248-5

19. Система автоматизированого проэктирования [Електронний ресурс]: Web-сайт. – Режим доступу: http://ru.wikipedia.org/wiki/Proteus – Назва з екрану.

Додаток А

Програма на мові Assembler

```
.include "2313def.inc"

.def temp = r16
.def sklav = r17
.def const = r18
.def klav = r19

ldi r31,low(ramend)
out spl,r31
ldi const, 0xff

ldi temp, 0b00001111
out ddrd, temp
ldi temp, 0xff
out ddrb, temp

virno:

ldi temp, 0b10000000 //zapaluvannja vsih segmentiv na 2c
out PortB, temp
rcall wait2c
ldi temp, 0b11111111 //погашення vsih segmentiv na 2c
out PortB, temp
ldi temp, 0
rcall wait2c

start:
```

```
ldi sklav, 0xff

ldi klav,0b00000001
out PortD,klav
sbic PinD,4
ldi sklav,0b10010000 //0
sbic PinD,5
ldi sklav,0b11111001 //1
sbic PinD,6
ldi sklav,0b10100100 //2

ldi klav,0b00000010
out PortD,klav
sbic PinD,4
ldi sklav,0b11000000 //3
sbic PinD,5
ldi sklav,0b10011001 //4
sbic PinD,6
ldi sklav,0b10010010 //5

ldi klav,0b00000100
out PortD,klav
sbic PinD,4
ldi sklav,0b10110000 //6
sbic PinD,5
ldi sklav,0b11111000 //7
sbic PinD,6
ldi sklav,0b10000000 //8
```

```
ldi klav,0b00001000
out PortD,klav
sbic PinD,4
ldi sklav,0b10000010 //9
sbic PinD,5
ldi sklav,0b10001000 //a
sbic PinD,6
ldi sklav,0b10000110 //e

rcall drebezg
out PortB, sklav

        cpi sklav, 0xff
        brne lich
ll:     cpi temp, 5
        breq error

        cpi sklav,0b11111001 //4
        breq kod1

        sbrs r9,0
        rjmp start

        cpi sklav,0b10110000 //5
        breq kod2

        sbrs r9,1
        rjmp start

        cpi sklav,0b10010010 //3
```

```
breq kod3

sbrs r9,2
rjmp start

cpi sklav,0b10000010 //9
breq kod4

mov const, r8
cpi const, 0b10000010 //9
brne start

l:
ldi sklav, 0b10110110
out portb, sklav
rcall wait2c
ldi sklav, 0b11111111
out portb, sklav
rcall wait2c
ldi sklav, 0b10110110
out portb, sklav
rcall wait2c
ldi sklav, 0b11111111
out portb, sklav
rcall wait2c
ldi temp, 0
mov r6, temp
mov r7, temp
mov r8, temp
rjmp start
```

```
kod1:
        mov r5, sklav
        ldi sklav, 0b00000001
        mov r9, sklav
        rjmp start

kod2:
        mov r6, sklav
        ldi sklav, 0b00000011
        mov r9, sklav
        rjmp start

kod3:
        mov r7, sklav
        ldi sklav, 0b00000111
        mov r9, sklav
        rjmp start

kod4:
        mov r8, sklav
        rjmp start

error:
        ldi sklav, 0b10000110
        out portb, sklav
        rcall wait2c
        rjmp start
 lich:
        inc temp
```

```
        rcall wait2c

rjmp ll

 wait2c:
        ldi r22,3
        labell:
        ldi r21, 255
        labell1:
        ldi r20, 255
        labell2:
        dec r20
        brne labell2
        dec r21
        brne labell1
        dec r22
        brne labell
ret
 drebezg:

        ldi r22, 50
        d:
        dec r22
        brne d

ret
```

Додаток Б

Програма на мові С

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.04.8b Evaluation
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 08.11.2012
Author  : Freeware, for evaluation and non-commercial use only
Company :
Comments:


Chip type               : AT90S2313
AVR Core Clock frequency: 1,000000 MHz
Memory model            : Tiny
External RAM size       : 0
Data Stack size         : 32
*****************************************************/

#include <90s2313.h>
#include <delay.h>

// Declare your global variables here
```

```
 int klava()
{   int sklav=0xff;

   PORTD = 0b00000001;
   if (PIND.4 == 1){sklav = 0b10010000;}
   if (PIND.5 == 1){sklav = 0b11111001;}
   if (PIND.6 == 1){sklav = 0b10100100;}

   PORTD = 0b00000010;
   if (PIND.4 == 1){sklav = 0b11000000;}
   if (PIND.5 == 1){sklav = 0b10011001;}
   if (PIND.6 == 1){sklav = 0b10010010;}

   PORTD = 0b00000100;
   if (PIND.4 == 1){sklav = 0b10110000;}
   if (PIND.5 == 1){sklav = 0b11111000;}
   if (PIND.6 == 1){sklav = 0b10000000;}

   PORTD = 0b00001000;
   if (PIND.4 == 1){sklav = 0b10000010;}
   if (PIND.5 == 1){sklav = 0b10001000;}
   if (PIND.6 == 1){sklav = 0b10000110;}

   return sklav;
}

void main(void)
{
char mas[4]={0,0,0,0};
int i=0;
```

```
int c=0;
int k=0;
int l=0;
int a=0xff;

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xff;

// Port D initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=15;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=FFFFh
```

```
// OC1 output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1H=0x00;
OCR1L=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;

while (1)
{    a=klava();
```

```
if (a!=0xff)
{
 PORTB=a;
 i++;
 delay_ms(100);

  if (a==0b11111001) //4
  {
    delay_ms(200);
    PORTB=0b11110111;
    mas[0]=a;
    c=1;
  }
  if (a==0b10100100 && c==1) //5
  {
    delay_ms(200);
    PORTB=0b10110111;
    mas[1]=a;
    k=1;
  }
  if (a==0b10011001 && k==1) //3
  {
    delay_ms(200);
    PORTB=0b11110111;
    mas[2]=a;
    c=2;
  }
  if (a==0b10010010 && c==2) //9
  {
    delay_ms(200);
```

```
            PORTB=0b10110111;
            mas[3]=a;
                  if (mas[0]==0b11111001 && mas[1]==0b10110000 &&
mas[2]==0b10010010 && mas[3]==0b10000010 && i==4)
                  {
                   PORTB=0b10101010;
                   delay_ms(3000);
                   PORTB=0b11111111;
                   delay_ms(3000);
                   i=0;
                  }
            }
            if (i>4)
            {
            mas[0]=0;
            mas[1]=0;
            mas[2]=0;
            mas[3]=0;
            c=0;
            k=0;
            PORTB=0b10000110;
            delay_ms(2000);
            i=0;
            l++;
            if(l>3){while(1){}}
            }
          }
        }
        }
```