Лабораторна робота №4

Тема: ПРОГРАМНА РЕАЛІЗАЦІЯ ПОСЛІДОВНОГО ПОРТУ ПРИЙНЯТТЯ ІНФОРМАЦІЇ

Мета: Визначити архітектуру програмного забезпечення для МК.

Теоретичні відомості

Для передачі даних в мікроконтролерній та комп'ютерній техніці використовують різноманітні протоколи та апаратно-програмні засоби. З метою вивчення можливостей передачі інформації послідовним однодротовим інтерфейсом, розглянемо гіпотетичну задачу:

ЗАДАЧА:

Вхідна інформація: Прийняти байт (значення байту знаходиться в межах 0..9), який передається послідовно від старшого до молодшого біту за протоколом вказаним на рисунку 1.

Вихідна інформація: Зобразити прийняте значення на світлодіодному індикаторі, рис. 2.

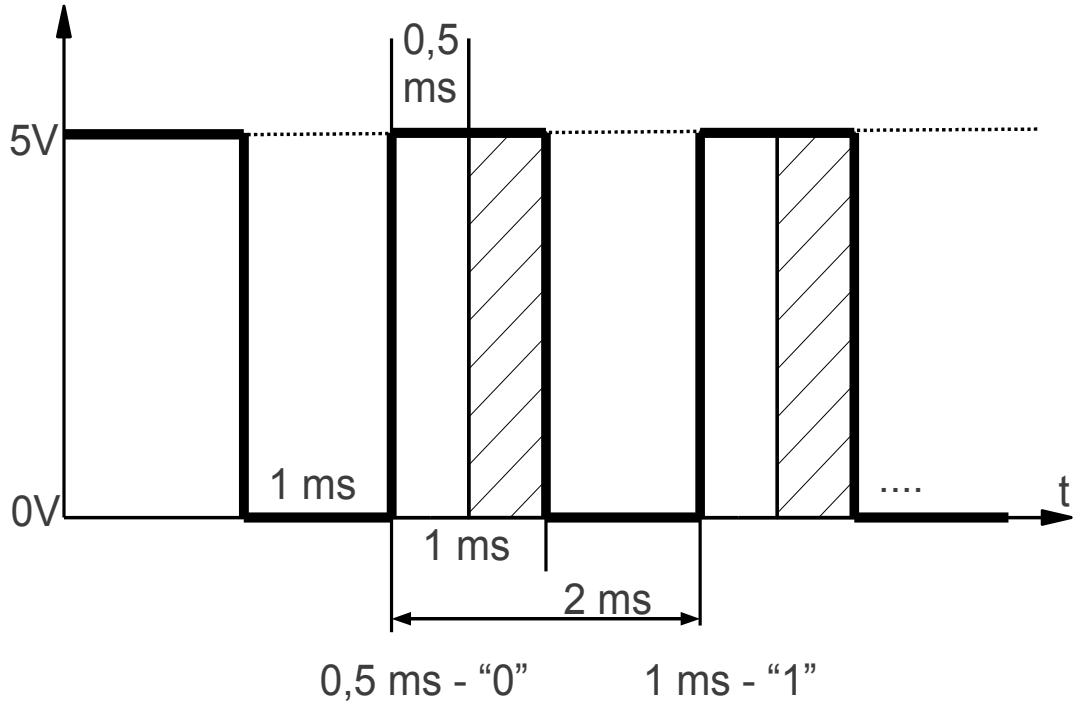

Рис. 1 – Протокол послідовної передачі байту

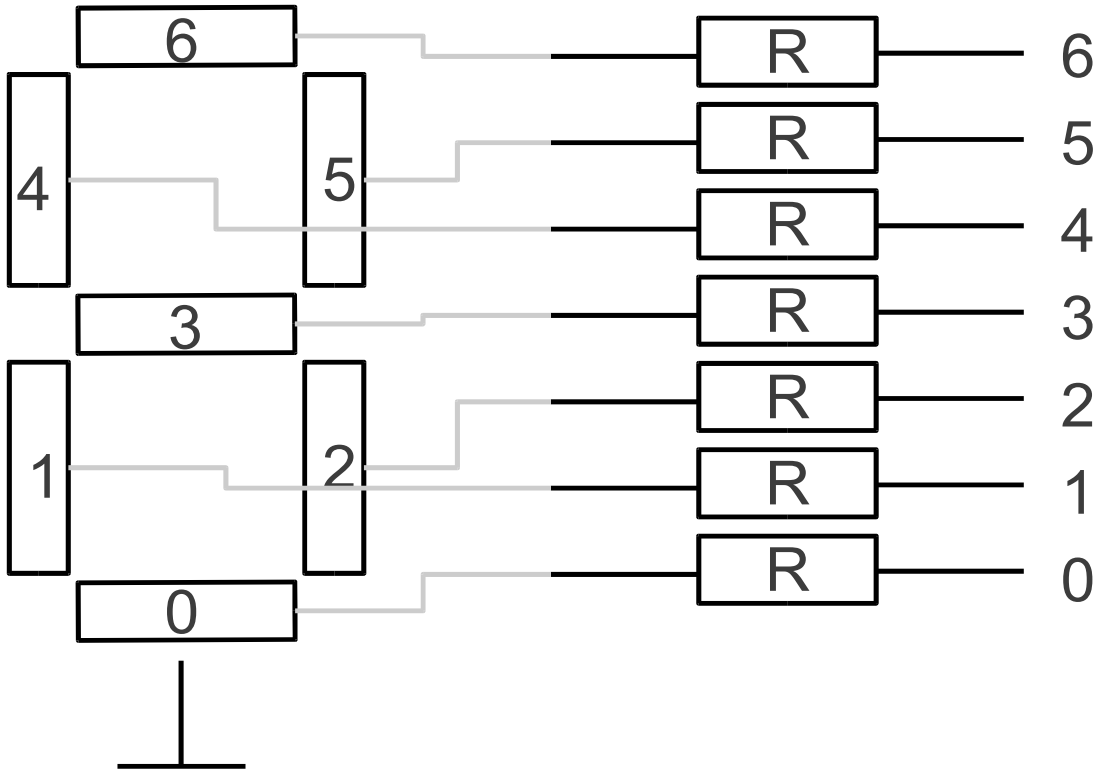

Рис. 2 – Пояснення що до підключення світлодіодів для зображення цифр.

Розв’язання

1) Для створення програмного забезпечення потрібно визначити його архітектуру (структуру). Структура програмного забезпечення визначається її складовими блоками. Ступінь деталізації структури може бути різною, і підвищуватися з розвитком розробки. Прикладом розробки структури програмного забезпечення може бути наступна ***структурна схема програмного забезпечення*** (рис. 3).

При створенні структурної схеми програмного забезпечення розглядають основні блоки, які потрібно реалізувати для повної функціональності системи. Фактично, на цьому етапі проходить процес розбиття основної задачі на ряд більш простих задач, які можна вирішувати окремо. Наприклад, окремою задачею є система, яка вимірюватиме довжину імпульсу в 1 чи 2 мілісекунди. Така система складатиметься з апаратною частиною відліку часу, системою синхронізації початку та кінця відліку. Окремо, як варіант вирішення задачі,

можна запропонувати системі вимірювати напругу через 1,5 мілісекунди після переходу напруги від низького до високого стану. В разі отримання довгого одиничного імпульсу, результатом вимірювання буде стан “1”, інакше “0”. Це спрощує загальний алгоритм, бо замість вимірювання часу між двома переходами, нам потрібно запустити вимір однотипової паузи, що є значно простішою задачею, ніж вимірювання часу між різнорідними подіями.

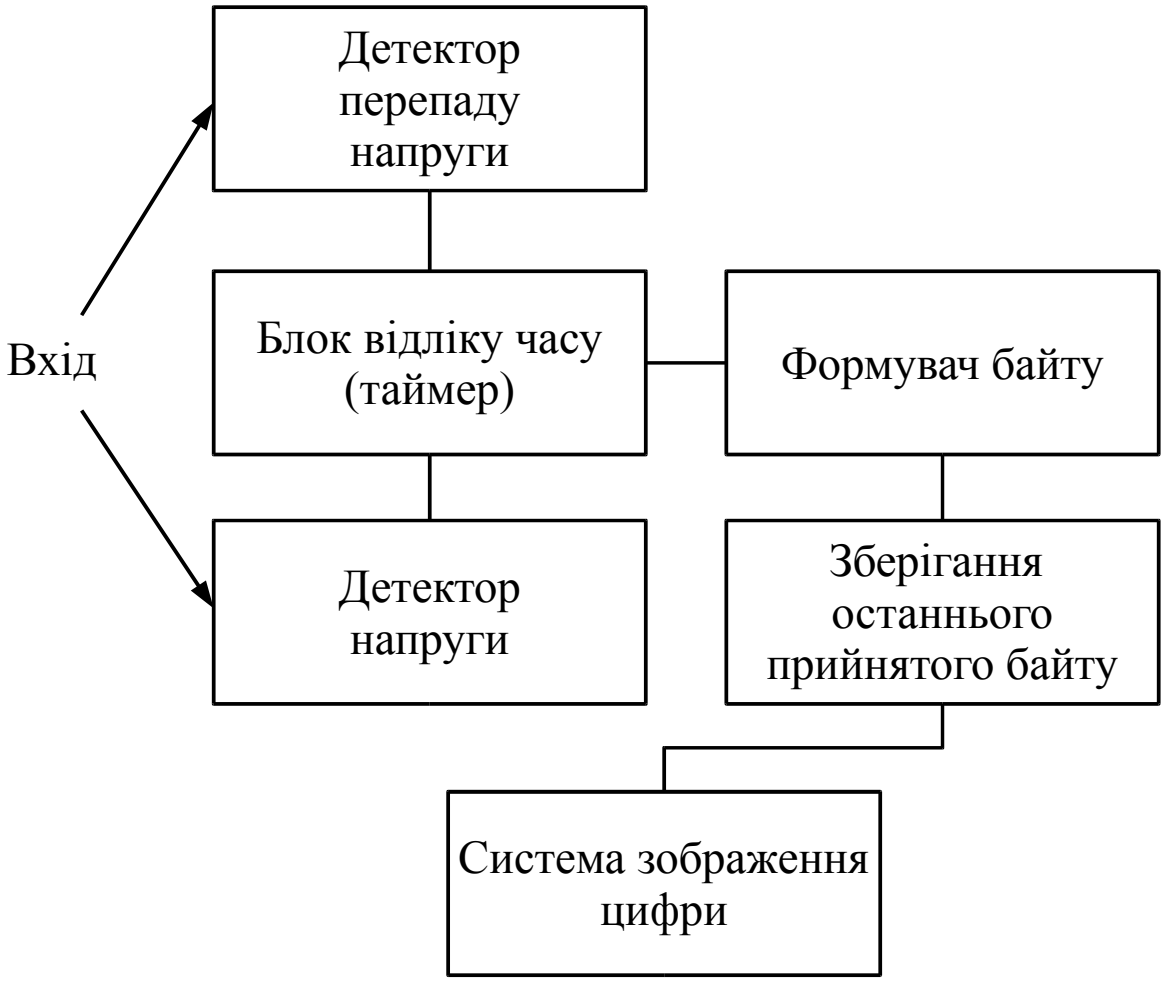

Рис. 3 – Структурна схема програмного забезпечення

Розробивши структурну схему програмного забезпечення та зробивши аналіз роботи її структурних одиниць, результати записують у вигляді функціональної схеми програмного забезпечення.

2) ***Розробка функціональної схеми*** розв’язує задачу визначення порядку роботи програму та функціональне призначення її структурних одиниць. Тому функціональна схема є продовженням структурної схеми, лише вона не

відповідає на питання “з чого складається програма”, а дає відповідь на питання “в якому порядку і що робить програма”. Прикладом функціональної схеми для вирішення поставленої задачі може бути схема, яка показана на рис. 4.

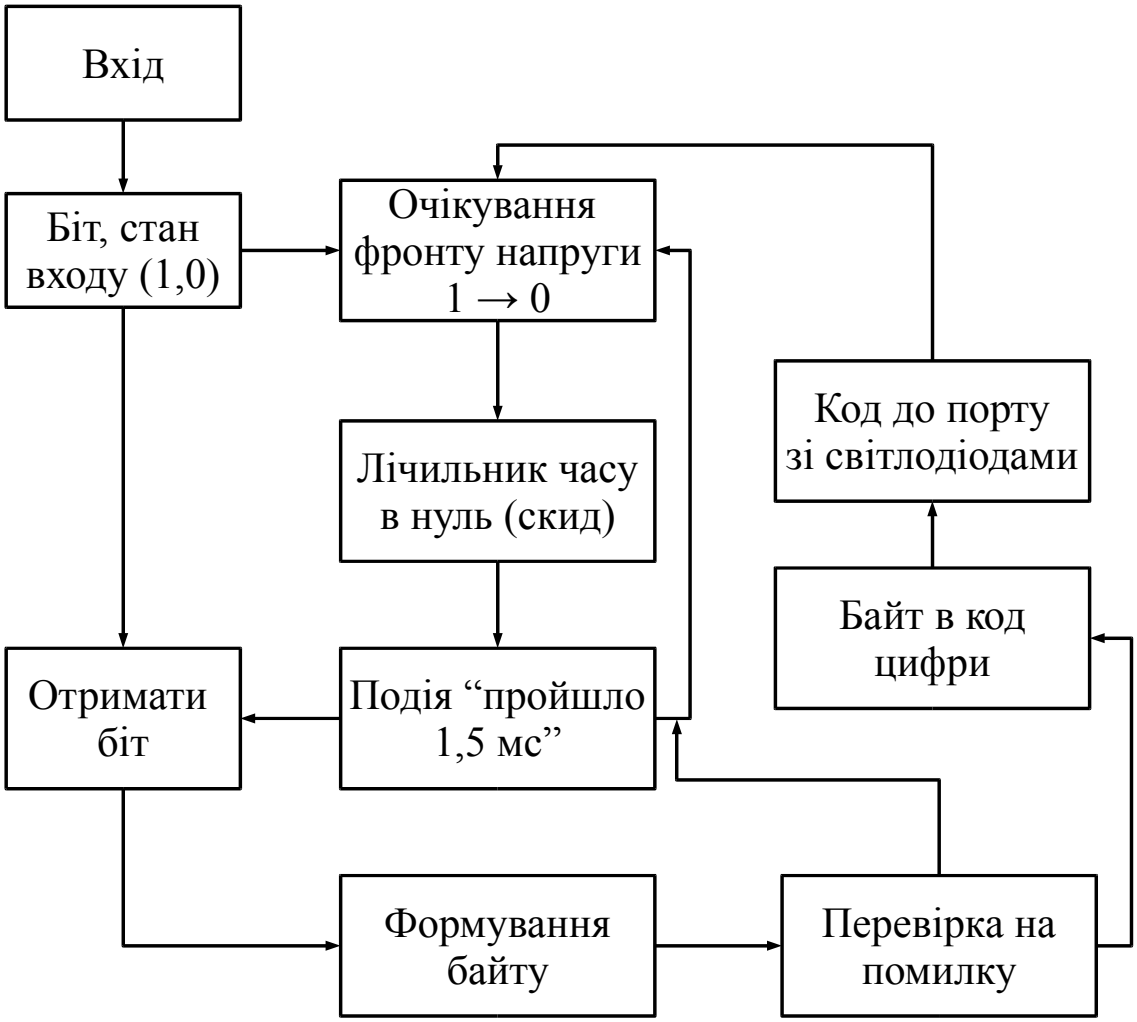

Рис. 4 – Функціональна схема програмного забезпечення

Фактично, така схема містить алгоритм роботи програмного забезпечення, і як для попередньої схеми, функціональна схема може містити узагальнюючи блоки та об’єднання блоків. Таке використання дозволить також використати розбиття виділених підзадач на ще більш мілкі та прості завдання. Логіка взаємодії таких складових елементів програмного забезпечення як раз і зображена на показаній схемі.

3) Після розробки структурної та функціональних схем приступають до розробки ***алгоритму роботи основної програми та допоміжних програм-функцій у вигляді блок-схем***, що розв’язують окремі підзадачі. Для складних

інформаційних систем розробці алгоритмів може передувати побудова діаграми руху інформації по вже створеним схемам. Така діаграма дозволить виявити помилки на попередніх етапах, виявити часові конфлікти, дублювання потоків та їх перетини по рівням обробки. При виявлені таких помилок потрібно повернутися до попередніх етапів для зміни розроблених схем та діаграм.

Блок схема основної програми повинна містити лише загальну логіку роботи. Фрагменти коду на мовах програмування використовувати в таких блок-схемах є недоречним, що може маскувати логічне призначення коду. В ідеальному випадку, блок-схема пояснює призначення та семантичний зміст дій та їх порядок, а фактична реалізація є вже мовно залежною з маскуванням сенсу використаних виразів. Ідею зрозумілості блок-схеми можна проілюструвати фрагментами блок-схем, які показано на рис. 5:

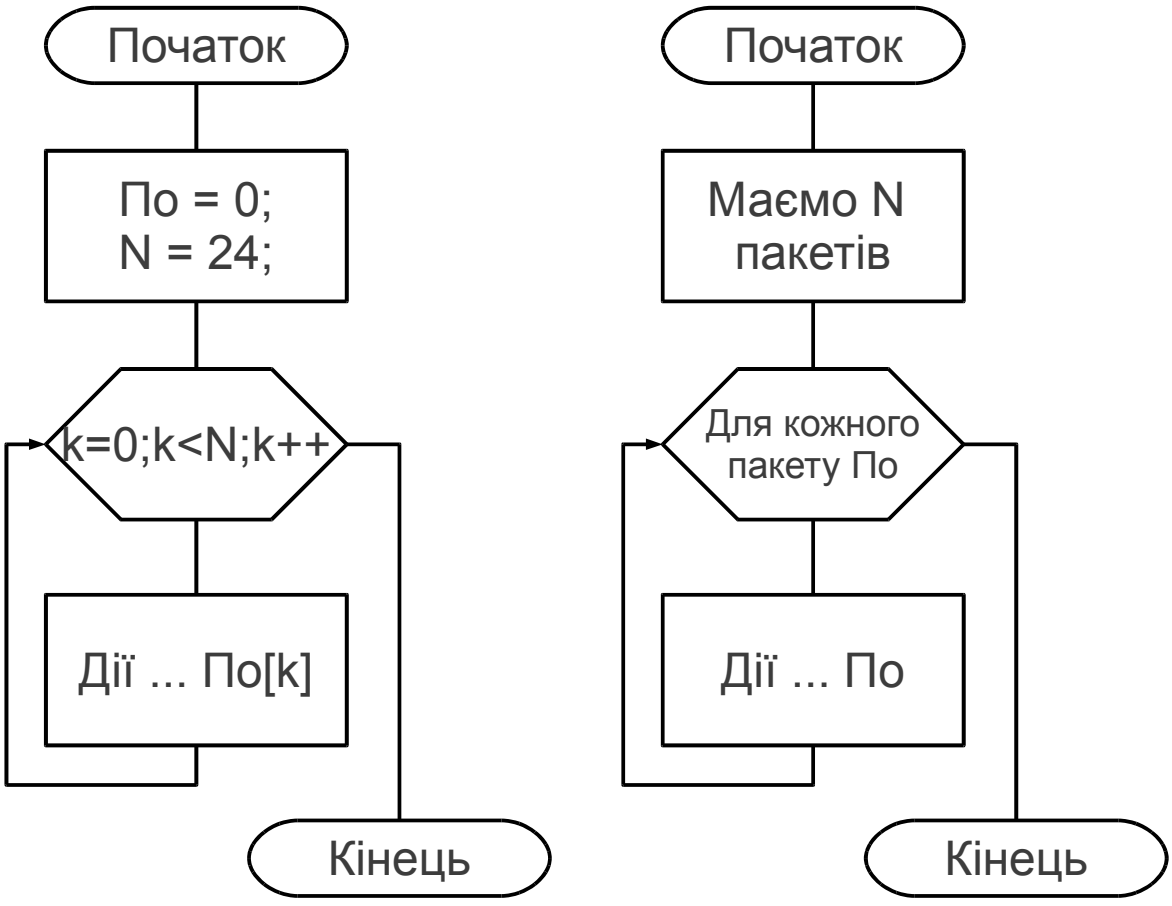

Рис. 5 – Приклад семантичної наповненості запису алгоритму

Програмне забезпечення для роботи на мікроконтролері отримує

керування зразу після подачі напруги, та виконується, доки пристрій не буде від'єднаний від живлення. Такий принцип роботи показано на наступній блок-схемі, яка є загальним алгоритмом поставленої задачі зображення прийнятої цифри на світлодіодному індикаторі (рис. 6):

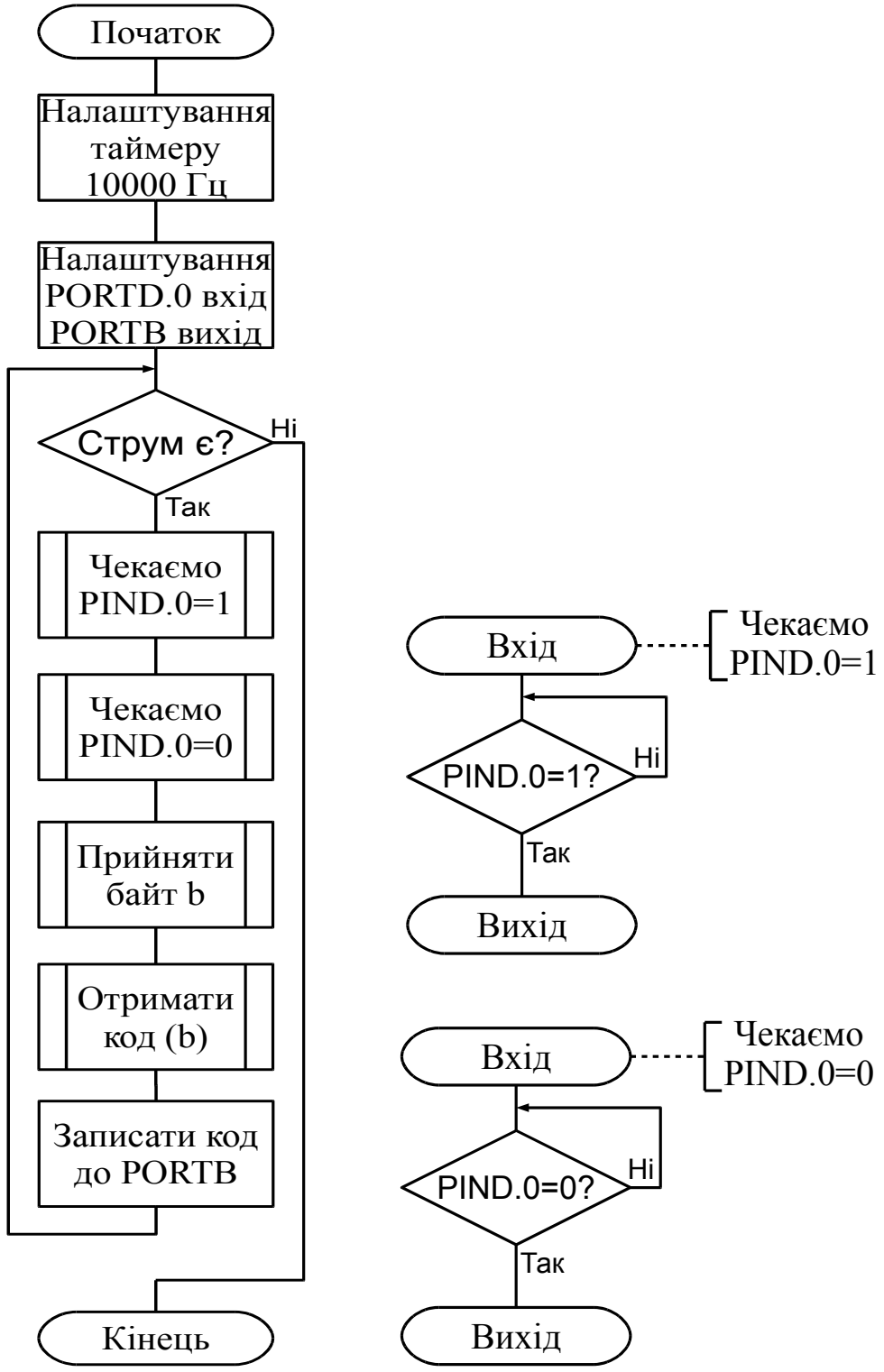

Рис. 6 – Блок-схеми алгоритмів програмного забезпечення

Також визначимо дії для прийняття байту блок-схеми алгоритмів прийняття байту та переводу значення цифри в код для її зображення:

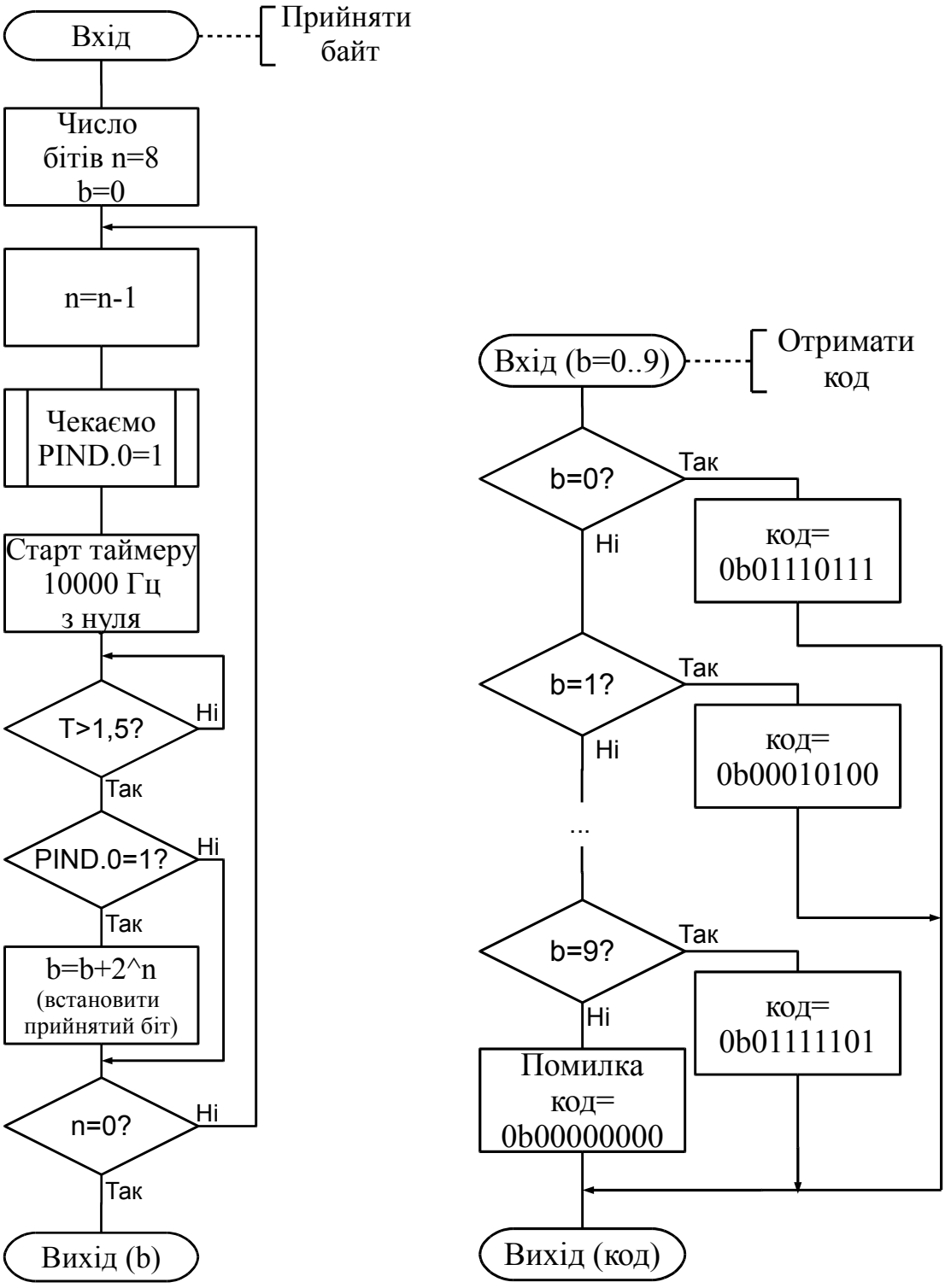

Рис. 7 – Блок-схеми алгоритмів програмного забезпечення

Після визначення роботи програми, можна записувати алгоритми довільною мовою програмування. Деякі моменти залежать від командної архітектури середовища виконання програмного коду, їх враховують саме на цьому етапі.

З блок-схем видно, що потрібно вирішити деякі апаратно залежні проблеми. Наприклад, для рахування часу нам потрібно налаштувати таймер, для якого зміна лічильника відбувалася 10000 раз за 1 секунду. Для налаштування частоти рахування таймера використовують регістр TCCR1B, в якому біти 2,1,0 - CS12, CS11, CS10 задають вибір тактування. Ці біти визначають джерело імпульсів для рахування для таймеру-лічильника 1.

Таблиця 1.

Вибір джерела імпульсів для рахування

CS12	**CS11**	**CS10**	**Опис**
0	0	0	Таймер/лічильник зупинено
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Зовнішній T1, 0 → 1
1	1	1	Зовнішній T1, 1 → 0

Звісно, для нашої мети повна відповідність та точність не потрібні, адже є потреба відрізнити імпульси в 1 та 2 мілісекунди, тому по факту замір часу в 1,2-1,8 мілісекунди дозволить розв’язати основну задачу.

Нехай маємо зовнішній кварцовий резонатор на 4МГц. Тоді при встановленні дільника імпульсів в 256, матимемо для 16-ти розрядного

лічильника частоту 15625 Гц з періодом переповнення 4,13 секунди. Тоді для 1,5 мілісекунди лічильник змінить показники на 23,4375 одиниці. Лічильник таймерів є цілочисловим, тому ми можемо реагувати на показник 23 або 24, що дає інтервал 1,5 мілісекунди з більш ніж 5% точністю. Для вирішення поставленої задачі ці показники є достатніми і не потребують уточнення.

Тоді для налаштування таймеру 1, потрібно до регістру TCCR1B занести константу 4. Значення відліків часу можна прочитати з парного регістру лічильника TCNT1L, TCNT1H. Щоб читання та запис двох байтів проходило одночасно, для роботи з ними використовується додатковий апаратний регістр (TEMP):

- Запис до таймеру-лічильника 1: При запису старшого байту в TCNT1H, значення передається в регістр TEMP. Потім, при запису молодшого байту, значення одночасно з TEMP передається до таймеру-лічильника 1. Тобто, при запису 16-розрядного значення, першим повинен записуватися байт в TCNT1H.

- Читання таймера-лічильника 1: При зчитуванні молодшого байту з TCNT1L, він попадає до процесора, але значення з TCNT1H зберігається у TEMP, тобто одночасно зчитуються всі 16-розрядів. При наступному зчитуванні регістру TCNT1H, значення береться з регістру TEMP.

Тепер можна перейти до написання власне програмного коду на обраній мові програмування.

```
#include <avr/io.h>

void initPINS(void);
void initTIMER1(void);
void wait0(void);
void wait1(void);
void wait15(void);
char toKod(char b);
char getByte(void);
```

```
int main(){
    initPINS();
    initTIMER1();
    PORTB = 0;
    char b,k;
    wait1();
    while(1){
        wait0();
        b = getByte();
        k = toKod(b);
        PORTB = k;
        wait1();
    }
    return 0;
}

void initPINS(void){
    DDRD = 0b00000000;
    DDRB = 0b11111111;
    PORTB = 0xFF;
}
void initTIMER1(void){
    TCCR1B = 4;
}
void wait0(void){
    char b=1;
    while(b==1){
        b = PIND & 0b00000001;
    }
}
```

```
void wait1(void){
    char b=0;
    while(b==0){
        b = PIND & 0b00000001;
    }
}
void wait15(void){
    TCNT1H = 0;
    TCNT1L = 0;
    char b = 0;
    char t = 0;
    while(b<24){
        b = TCNT1L;
        t = TCNT1H;
    }
}
char toKod(char b){
    if(b==0) return 0b01110111;
    if(b==1) ...
    ...
    if(b==9) return 0b01111101;
    return 0;
}
char getByte(void){
    char n=8;
    char b=0;
    while(n>0){
        n--;
        wait1();
        wait15();
        if( (PIND & 1)==1 ) b |= (1<<n);
```

```
            wait0();
        }
        return b;
    }
```

Завдання:

№	Задача (вхід/вихід обрати довільно)
0	Подавати/знімати напругу до світлодіоду, якщо до входу подано одиничний імпульс довший за 1 сек.
1	Засвітити світлодіод на 1 сек. при одночасному надходженні одиничного стану на два входи.
2	Засвітити світлодіод PORTD.0, якщо була правильна послідовність одиничних імпульсів на PORTB.0..9
3	Порахувати кількість імпульсів на вході за 4 сек.
4	Визначити час між двома імпульсами з точністю 1 мс.
5	Зробити мигалку зі світлодіодом на 3 спалахи за 1 с.
6	Визначити час утримання одиничного сигналу на вході з точністю 1 мс.
7	Зробити почергове спалахування світлодіодів з частотою 10Гц.
8	Визначити факт наявності імпульсу довшому за 0,5 с.
9	Зробити дільник вхідної частоти на 5. (Одне перемикання на 5 вхідних)

1. Розробити структурну схему програмного забезпечення.
2. Розробити функціональну схему програмного забезпечення.
3. Розробити блок-схеми алгоритмів роботи програмного забезпечення.
4. Запишіть з поясненнями налаштування портів вводу:

_____

_____

_____

_____

5. Запишіть з поясненнями налаштування таймеру:

_____

_____

_____

_____

6. Дайте відповіді на контрольні питання.

Контрольні питання:

1. Які параметри потрібно задати, щоб таймер рахував з частотою 100000Гц:

_____

_____

_____

_____

_____

2. Який регістр використовується для порівняння з лічильником таймеру?

_____

_____

_____

3. Навіщо міряти час подачі імпульсів напруги (вигадати приклад)?

_____

_____

_____

_____

_____

4. Як можна підключати світлодіод до МК?

_____

_____

_____

_____

_____

5. Як можна підключати кнопку до МК?

_____

_____

_____

_____

_____