Міністерство освіти і науки України

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра радіотехнічних та вбудованих систем

**ЗВІТ**

про виконання розрахунково-графічної роботи

з дисципліни: «Програмування периферійних пристроїв»

Варіант №18

**Виконав:** студент групи Кіт-211

Оленченко Ілля Романович

**Викладач:** кандидат технічних наук,

доцент кафедри РТВС

Красножон Олексій Васильович

Чернігів 2022

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА №1**

**УПРАВЛІННЯ ПЕРФІРЕЙНИМІ ПРИСТРОЯМИ ЗА ДОПОМОГОЮ ІНКРЕМЕНТАЛЬНОГО ЕНКОДЕРА**

**Мета роботи:** Ознайомитися із функціональними можливостями і внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню організацію і принципи роботи інкрементального енкодера. Навчитися керувати різними периферійними пристроями стенду за допомогою інкрементального енкодера.

**1 Короткі теоретичні відомості**

Енкодер (або перетворювач кутових переміщень) – це пристрій, призначений для перетворення кута повороту об’єкта (вала), що обертається, в електричні сигнали, що дозволяє визначити величину кута повороту.

Енкодери дуже широко застосовуються в промисловості. Вони поділяються на інкрементальні та абсолютні (які можуть досягати дуже високої роздільної здатності).

Інкрементальний енкодер за один повний оберт формує фіксовану кількість імпульсів, абсолютні ж енкодери дозволяють в будь-який момент часу знати поточний кут повороту вісі, в тому числі і після зникнення та відновлення живлення. Багатооборотні абсолютні енкодери, крім того, також підраховують і запам’ятовують кількість повних обертів осі.

Енкодери можуть бути як оптичні, резистивні, так і магнітні, при цьому вони можуть працювати через шинні інтерфейси або промислову мережу.

Перетворювачі кутових переміщень на сьогоднішній день практично повністю витіснили застосування сельсинів.

**1.1 Інкрементальні енкодери**

Інкрементальні енкодери призначені для визначення кута повороту об’єктів, що обертаються. Вони генерують послідовний імпульсний цифровий код, що містить інформацію про кут повороту об'єкта. Якщо вал зупиняється, то припиняється і передача імпульсів. Основним робочим параметром інкрементального енкодера є кількість імпульсів за один оберт. Миттєве значення кута повороту об'єкта визначають шляхом підрахунку кількості імпульсів від початку. Для обчислення кутової швидкості об’єкта, обчислювальний елемент тахометра виконує диференціювання кількості імпульсів у часі, таким чином, показуючи відразу величину швидкості, тобто число обертів на хвилину.

Вихідні сигнали інкрементального енкодера формуються одночасно в двох каналах, в яких ідентичні послідовності імпульсів зсунуті на 90° один відносно одного (так звані парафазні імпульси), що дозволяє визначати напрямок обертання. Є також цифровий вихід нульової мітки, який дозволяє завжди розрахувати абсолютне положення вала.

Імпульсний (або інкрементальний, покроковий) енкодер відноситься до типу енкодерів, призначених для вказівки напряму руху і/або кутового переміщення зовнішнього механізму. Інкрементальний енкодер формує імпульси, кількість яких відповідає повороту вала на певний кут. Цей тип енкодерів, на відміну від абсолютних, не формує код положення вала, коли вал знаходиться в спокої. Енкодер пов’язаний із лічильним пристроєм, це необхідно для підрахунку імпульсів і перетворення їх в міру переміщення вала.

Покроковий оптичний енкодер складається з наступних компонентів: джерела світла, диска з мітками, фототранзисторної збірки і схеми обробки сигналу. Диск покрокового енкодера (дивись рисунок 1.1) поділено на точно спозиціоновані позначки. Кількість відміток визначає кількість імпульсів за один оберт. Наприклад, якщо диск поділено на 1000 міток, тоді за 250 імпульсів вал повинен повернутися на 90 градусів.



Рисунок 1.1 – Диск із мітками інкрементального енкодера

Для квадратури виходу енкодера використовуються два вихідних канали (А і B), що дозволяє визначити, чи обертається вал за годинниковою стрілкою або проти неї. Визначення ґрунтується на зсуві фази 90°±0°, при цьому, прийнятний допуск для визначення зсуву фаз може досягати ±45°. Енкодер із єдиним виходом (A) більш відомий як тахометр.

**1.2 Порядок виконання роботи**

1.1 Запустити IDE AVR Studio 4.

1.2 Створити нові проекти в IDE AVR Studio 4.

1.3 У вікнах, що з’явилися, написати програми мовами С і асемблер з урахуванням варіанту завдання, зазначеного в таблиці 1.1. Номер варіанту співпадає із порядковим номером студента згідно списку у журналі академ. групи.

Таблиця 1.1 – Варіанти завдань

|  |  |
| --- | --- |
| № | Опис завдання |
| 18 | При кожному повороті енкодера за годинниковою стрілкою на 2 тіки, значення 8-розрядного реверсивного лічильника в коді Грея збільшується на 1, а при повороті енкодера проти годинникової стрілки на 4 тіки – зменшується на 1. При натисканні на інкрементальний енкодер як на кнопку, лічильник приймає значення коду Грея, що відповідає числу 15. Початковим значенням лічильника є код Грея для числа 255. Стани реверсивного лічильника повинні відображатися на напівпровідниковому світлодіодному індикаторі. |

**2 ХІД ВИКОНАННЯ РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

**2.1 Результати виконання завдання**

Згідно до варіанту розрахункової-графічної роботи на напівпровідниковому світлодіодному індикаторі було відображено стани реверсивного лічильника які мінялися відповідно до положення енкодера.

В лістингу 1.1 зображено код програми мовою С

/\*\*\* Програма до розрахунково-графічної роботи (стенд EV8031) \*\*\*\*\*

\*\*\* Працюємо з інкрементальним енкодером \*\*\*

\*\*\* При кожному повороті енкодера (один тік) за годинниковою стрілкою

\*\*\* виведене на лінійку світлодіодів значення збільшується на 1, а

\*\*\* при кожному повороті енкодера (один тік) проти годинникової стрілки

\*\*\* виведене на лінійку світлодіодів значення зменшується на 1

\*\*\* при натисканні на інкрементальний енкодер як на кнопку,

\*\*\* світлодіодний індикатор повністю гасне \*\*\*/

#define F\_CPU 7372800L //задаємо частоту кварцу (7,3728 МГц)

#include <avr/io.h>

#include <avr/iom8515.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define led\_line 0xA006 //адреса лінійки світлодіодів в стенді

//регістр стану напівпровідникового світлодіода

volatile unsigned char diode;

//регістр тимчасового зберігання

volatile unsigned char temp;

//лічильник кількості імпульсів, одержаних від енкодера

//volatile unsigned char counter;

int counter = 0;

//регістр, який зберігає попередній стан інкрементального енкодера

volatile unsigned char last\_state;

//регістр, який зберігає наступний стан інкрементального енкодера

volatile unsigned char next\_state;

volatile unsigned char clock = 0b10000000;

volatile unsigned char conterclock = 0b00010000;

volatile unsigned int data = 0;

volatile unsigned char timer = 0;

volatile unsigned int dot = 0b11111111;

ISR (TIMER1\_COMPA\_vect){

timer = 1;

}

int main(void) {

ACSR = 1 << ACD; //відключення живлення аналогового компаратора

MCUCR = 1 << SRE; //дозволяємо роботу із зовнішньою пам'яттю

TIMSK = 1 << OCIE1A;

OCR1A = 0x0399; //завантажуємо в регістрову пару OCR1AH:OCR1AL число 0x707F

//або 28799, тобто задаємо значення, при досягненні якого

//таймер/лічильник T1 згенерує переривання по співпадінню А

SFIOR = 1 << PSR10; //скидаємо переддільник,

//фактично в цей момент відбувається початок відліку

TCCR1A = 0x00;

TCCR1B = 1 << WGM12 | 1 << CS11;

//Оголошуємо допоміжний вказівник для звернення до світлодіодної лінійки

volatile unsigned char \*p = (unsigned char \*) led\_line; //встановлюємо вказівник p на адресу лінійки світлодіодів в стенді

diode = 0x00; //очищаємо регістр стану лінійки светодіов (гасимо все)

counter = 0x00; //очищаємо регістр лічби імпульсів, що надходять від енкодера

last\_state = PINB; //читаємо поточний стан каналів енкодера

last\_state &= 0b1001000; //визначаємо стан бітів порту, що відповідають за поточний стан енкодера

//вважаємо, що в початковому стані енкодер ніхто не чіпав, тобто

//в кожному з його каналів (канал B - PB4; канал A - PB7) утримується "0"

diode = dot ^ (dot >> 1);

\*p = diode;

sei();

//Переходимо в нескінченний цикл опитування стану енкодера

while (1) {

if (timer == 1) {

//\_delay\_ms(1); //викликаємо затримку, щоб не читати стани енкодера занадто часто

next\_state = PINB; //читаємо поточний стан каналів енкодера

next\_state &= 0b10010000; //отримуємо стан бітів порту, що відповідають за поточний стан енкодера

if (next\_state != last\_state) { //якщо стан енкодера змінився

if (last\_state == 0x00) { //якщо в каналі В енкодера не було "0" і в каналі А теж не було "0"

if (next\_state == clock) { // якщо в каналі A енкодера стала "1", а в каналі B залишився "0"

counter++; //збільшуємо значення лічильника кількості імпульсів, одержуваних від енкодера

if (counter >= 2) {

counter = 0;

data++;

}

} else if (next\_state == conterclock) { //якщо в каналіA енкодера залишився "0", а в каналі B стала "1"

counter--; //зменшення значення лічильника кількості імпульсів, одержуваних від енкодера

if (counter <= -4) {

counter = 0;

data--;

}

}

}

}

temp = PINB; //читаємо поточний стан каналів енкодера

temp &= 0b00100000; //отримуємо стан біту порту, що відповідає за натискання енкодера як кнопки

if (temp == 0x00) { //якщо енкодер натиснуто як кнопку, то обнуляємо стан діодів

counter = 0; //власне, скидання стану енкодера

data = 15;

}

diode = data ^ (data >> 1);

\*p = diode; //оновлюємо стан індикатора

last\_state = next\_state; //підготовка до наступної ітерації опитування стану енкодера

//присвоюємо попередньому стану енкодера його поточний стан

timer = 0;

}

}

return 0;

}

Лістинг 1.1 – код програми мовою С

В лістингу 1.2 зображено код програми мовою асемблер

;\*\*\* Програма до розрахунково-графічної роботи (стенд EV8031) \*\*\*\*\*

;\*\*\* Працюємо з інкрементального енкодером \*\*\*

;\*\*\* При кожному повороті енкодера (один тік) за годинниковою стрілкою

;\*\*\* виведене на лінійку світлодіодів значення збільшується на 1, а

;\*\*\* при кожному повороті енкодера (один тік) проти годинникової стрілки

;\*\*\* виведене на лінійку світлодіодів значення зменшується на 1

;\*\*\* при натисканні на інкрементальний енкодер як на кнопку,

;\*\*\* світлодіодний індикатор повністю гасне

.include "m8515def.inc" ;підключення модуля контролера ATmega8515

;\*\*\* Призначення символічних імен регістрів \*\*\*

.def diode = r16 ;регістр, який зберігає стан світлодіодним лінійки

.def temp = r17 ;регістр тимчасового зберігання

.def encoder\_data = r18 ;лічильник імпульсів енкодера

;Регістр, що зберігає попередній стан інкрементального енкодера

.def last\_encoder\_state = r19

;Регістр, що зберігає наступний стан інкрементального енкодера

.def next\_encoder\_state = r20

.def encoder\_counter\_r = r21

.def encoder\_counter\_l = r22

.def data = r15

.def counter = r23 ;лічильник циклу в підпрограмі генерації затримки

.def long\_delay\_low = r24 ;молодший байт лічильника довгої затримки

.def long\_delay\_high = r25 ;старший байт лічильника довгої затримки

;\*\*\* Призначення констант \*\*\*

.EQU led\_line = 0xA006; адреса лінійки світлодіодів в стенді

;\*\*\*\*\* Початок програми \*\*\*\*\*

.CSEG ;визначаємо початок сегмента коду

.ORG 0x0000 ;визначаємо адресу початку сегмента коду в пам'яті програм

;\*\*\* Таблиця векторів переривань контролера \*\*\*

rjmp Init; вектор переривання по скиданню

reti; rjmp EXT\_INT0; IRQ0 Handler

reti; rjmp EXT\_INT1; IRQ1 Handler

reti; rjmp TIM1\_CAPT; Timer1 Capture Handler

reti; rjmp TIM1\_COMPA; Timer1 Compare A Handler

reti; rjmp TIM1\_COMPB; Timer1 Compare B Handler

reti; rjmp TIM1\_OVF; Timer1 Overflow Handler

reti; rjmp TIM0\_OVF; Timer0 Overflow Handler

reti; rjmp SPI\_STC; SPI Transfer Complete Handler

reti; rjmp USART\_RXC; USART RX Complete Handler

reti; rjmp USART\_UDRE; UDR0 Empty Handler

reti; rjmp USART\_TXC; USART TX Complete Handler

reti; rjmp ANA\_COMP; Analog Comparator Handler

reti; rjmp EXT\_INT2; IRQ2 Handler

reti; rjmp TIM0\_COMP; Timer0 Compare Handler

reti; rjmp EE\_RDY; EEPROM Ready Handler

reti; rjmp SPM\_RDY; Store Program memory Ready

;\*\*\* Початкова ініціалізація контролера \*\*\*

Init:

ldi temp, low (RAMEND)

out SPL, temp

ldi temp, high (RAMEND)

out SPH, temp ;установка SP на останню адресу в SRAM

sbi ACSR, 7 ;відключення живлення аналогового компаратора

ldi temp, 0b10000000 ;дозволяємо роботу із зовнішньою пам'яттю

out MCUCR, temp

;Встановлюємо вказівник Z на адресу лінійки світлодіодів в стенді

ldi ZL, low (led\_line)

ldi ZH, high (led\_line)

clr diode ;очищаємо регістр стану світлодіодної лінійки (гасимо всі діоди)

clr encoder\_data ;очищаємо регістр лічби імпульсів, що надійшли з енкодера

in last\_encoder\_state, PINB ;читаємо поточний стан каналів енкодера

;Отримуємо стан бітів порту, що відповідають за поточний стан енкодера

andi last\_encoder\_state, 0b10010000

;Вважаємо, що в початковому стані енкодер ніхто не використовує, тобто

;в кожному з його каналів (канал B - PB4; канал A - PB7) утримується "0"

;\*\*\* Переходимо в нескінченний цикл опитування стану енкодера \*\*\*

;\*\*\* Налаштування переривань \*\*\*

ldi temp, (1 << OCIE1A) ;записуємо маску в регістр temp

out TIMSK, temp ;записуємо в регістр TIMSK (дозволяємо переривання

;за збігом A таймера/лічильника T1)

;\*\*\* Налаштування та пуск таймера/лічильника T1 \*\*\*

;Завантажуємо в пару регістрів OCR1AH: OCR1AL число 0x707F (28799d), тобто задаємо

;значення, при досягненні якого таймером/лічильником T1 буде

;згенеровано переривання по співпадінню A

ldi encoder\_counter\_r, 0b000000000

ldi encoder\_counter\_r, 0b000000000

ldi temp, 0x70

out OCR1AH, temp

ldi temp, 0x7F

out OCR1AL, temp

ldi temp, (1 << PSR10) ;скидаємо переддільник, записуючи 1 в біт PSR10,

out SFIOR, temp ;фактично в цей момент відбувається початок відліку

;Завантажуємо в пару регістрів TCCR1A: TCCR1B число 0x000C

ldi temp, 0b00000000

out TCCR1A, temp ;задаємо режим роботи CTC і власне запускаємо

ldi temp, (1 << WGM12) | (1 << CS12) ;таймер/лічильник T1, переддільник встановлено на 256

out TCCR1B, temp ;!у цьому місці саме відбувається пуск таймера, хоча переддільник вже працює!

;!!!Оскільки переддільник встановлено на 256, то безпосередньо на лічильний вхід модуля таймера/лічильника T1

;надходить частота 7.3728 МГц / 256 = 28800 Гц. Очевидно, що модуль лічби становить N = 28800

;Звідси легко бачити, що для генерації часового проміжку в 1 с необхідно в 16-розрядний регістр

;порівняння OCR1AH: OCR1AL записати значення (N-1) = 28799d або 0x707F (707Fh)!!!

;!!!Завжди слід записувати N-1, тому що лічильник починає рахувати із нульового стану !!!

ldi diode, 0x80

;\*\*\* Дозволяємо роботу замаскованих переривань \*\*\*

sei ;встановлюємо прапорець глобального дозволу переривань

;працюватиме тільки переривання TIM1\_COMPA),

;тобто переривання по співпадінню A таймера/лічильника Т1

;\*\*\* Переходимо в нескінченний цикл, де очікуємо переривання від таймера

;(Раз на секунду) і змінюємо стан світлодіодів \*\*\*

Infinite\_loop:; нескінченний цикл

rcall long\_delay ;викликаємо затримку, щоб не читати стан енкодера занадто часто

rjmp Infinite\_loop

;\*\*\* Підпрограма обробки переривання по співпадінню A таймера/лічильника T1

TIM1\_COMPA: ;обробник зовнішнього переривання TIM1\_COMPA (при вході I = 0)

in next\_encoder\_state, PINB ;читаємо поточний стан каналів енкодера

;Отримуємо стан бітів порту, що відповідають за поточний стан енкодера

andi next\_encoder\_state, 0b10010000

cp next\_encoder\_state, last\_encoder\_state

breq Next\_iterate ;якщо стан енкодера не змінився - продовжуємо його читати

cpi last\_encoder\_state, 0b00000000 ;якщо в каналі В енкодера не було "0" і в каналі А теж не було "0"

brne Next\_iterate; переходимо до подальших дій

cpi next\_encoder\_state, 0b00010000 ;якщо в каналі A енкодера залишився "0", а в каналі B з'явилася "1", то енкодер повернули проти годинниковою стрілкою

breq Dec\_enc\_counter ;переходимо до зменшення стану енкодера

cpi next\_encoder\_state, 0b10000000 ;якщо в каналі A енкодера з'явилася "1", а в каналі B залишився "0", то енкодер повернули за годинниковою стрілкою

breq Incr\_enc\_counter ;переходимо до збільшення стану енкодера

cpi next\_encoder\_state, 0b00100000

breq Encoder\_as\_button

rjmp Next\_iterate ;якщо в жодному з каналів не стався фронт - йдемо далі

reti ;повернення із переривання

;при цьому автоматично встановлюється прапорець I

;з метою подальшого дозволу переривань

;\*\*\*\*\* Кінець програми \*\*\*\*\*

Incr\_enc\_counter:

inc encoder\_counter\_r

ldi encoder\_counter\_l, 0b00000000

cpi encoder\_counter\_r, 0b00000010 ;якщо в каналі A енкодера з'явилася "1", а в каналі B залишився "0", то енкодер повернули за годинниковою стрілкою

BRGE Incr\_state

Dec\_enc\_counter:

inc encoder\_counter\_l

ldi encoder\_counter\_r, 0b00000000

cpi encoder\_counter\_l, 0b00000100 ;якщо в каналі A енкодера залишився "0", а в каналі B з'явилася "1", то енкодер повернули проти годинниковою стрілкою

BRGE Decr\_state ;переходимо до зменшення стану енкодера

Encoder\_as\_button:

ldi encoder\_counter\_r, 0b00000000

ldi encoder\_counter\_l, 0b00000000

ldi encoder\_data, 0b00001111

rjmp Next\_iterate ;якщо в жодному з каналів не стався фронт - йдемо далі

Incr\_state:

ldi encoder\_counter\_r, 0b00000000

inc encoder\_data ;інкремент кількості імпульсів, отриманих від енкодера

rjmp Next\_iterate ;переходимо до подальших дій

Decr\_state:

ldi encoder\_counter\_l, 0b00000000

dec encoder\_data ;декремент кількості імпульсів, отриманих від енкодера

Next\_iterate: ;перехід на цю мітку здійсниться в будь-якому випадку!

in temp, PINB ;читаємо поточний стан каналів енкодера

;Отримуємо стан біта порту, що відповідає за натискання енкодера як кнопки

andi temp, 0b00100000

tst temp ;якщо енкодер натиснуто як кнопку, то обнуляємо лічильник імпульсів, отриманих від нього

brne Led\_action ;а якщо ні - йдемо далі

clr encoder\_data ;власне, скидання стану енкодера

Led\_action:

mov data, encoder\_data

mov temp, encoder\_data

lsr temp

eor data, temp

mov diode, data ;оновлюємо вміст регістра стану лінійки світлодіодів

st Z, diode ;оновлюємо стан індикатора

mov last\_encoder\_state, next\_encoder\_state ;підготовка до наступної ітерації опитування стану енкодера

;Присвоюємо попереднього стану енкодера його поточний стан

rjmp Infinite\_loop

;\*\*\* Підпрограма довгою затримки \*\*\*

long\_delay:

;\* Якщо в регістрову пару завантажити число 9 (0009h), то затримка буде близько 976,5 мкс

;\* Приблизна формула розрахунку коефіцієнта при кварці в 7.3728 МГц така:

;\* 800 x коефіцієнт затримки/(7.3728 \* 1 000 000) = необхідний час в с

ldi long\_delay\_low, 0x09; завантаження в регістрову пару коефіцієнта затримки

ldi long\_delay\_high, 0x00 ;0009h - це буде затримка на 976,5 мкс

long\_loop: ;тіло циклу займає 796 + 2 + 2 = 800 тактів

rcall short\_delay ;коротка затримка

sbiw long\_delay\_high: long\_delay\_low, 0b00000001 ;віднімання із пари числа 1 (декремент довгого лічильника)

brne long\_loop ;якщо не 0, повторити цикл

ret ;повернення в основну програму

;\*\*\* Підпрограма короткої затримки (потрібна для генерації довгих затримок) \*\*\*

short\_delay: ;вся підпрограма займає рівно 796 тактів разом з rcall і ret

nop

ldi counter, 0xC5 ;лічильник циклу

short\_loop:

nop

dec counter

brne short\_loop ;команда розгалуження по прапору нуля (зациклення)

ret ;повернення в основну програму

.EXIT; кінець програми

Лістинг 1.2 – код програми мовою ассемблер

На рисунку 2.1 зображено схему алгоритму

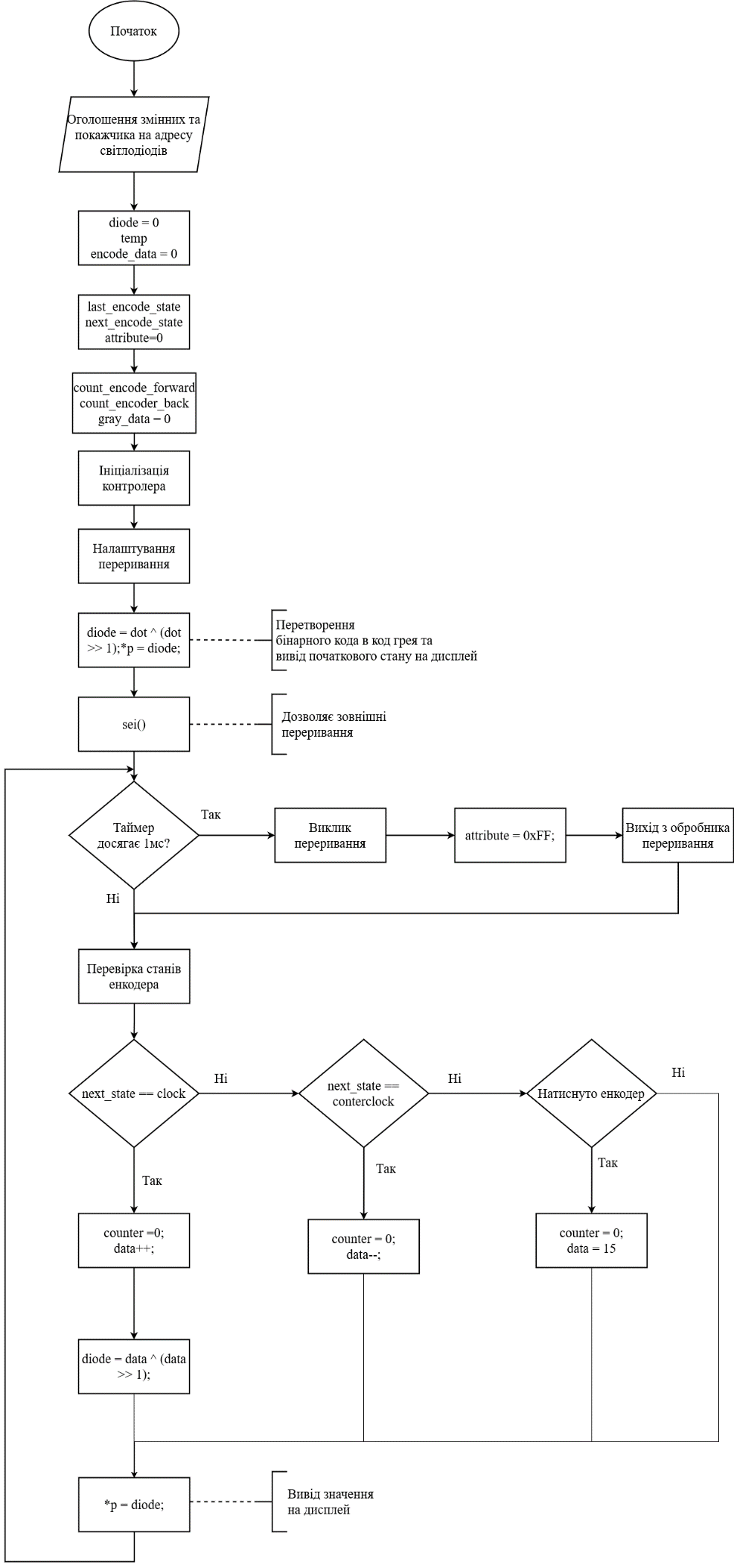


Рисунок 2.1 – схема алгоритму

**3 ОСОБЛИВОСТІ IDE AVR STUDIO 4**

З особливостей у IDE AVR STUDIO 4 э інтегроване налагоджувальне середовище. Для зручнішого налагоджування в avr studio 4 є багато вікон в яких можна відстежувати зміни які відбуваються у стенді.

IDE AVR Studio містить:

1. Засоби створення та управління проектом
2. Редактор коду мовою асемблер
3. Транслятор мови асемблера (Atmel AVR macroassembler)
4. Відладчик
5. Програмне забезпечення верхнього рівня для підтримки внутрішньосхемного програмування (In-System Programming, ISP) з використанням стандартних налагоджувальних засобів Atmel AVR
6. Драйвер для підтримки компілятора WinAVR - компілятора мови С

**ВИСНОВОК**

Після виконання розрахункової-графічної роботи ми ознайомилися та навчилися працювати з енкодером розташованим на мікроконтролері AT mega 8515. Навчився писати код на C та ассемблері для енкодера. Ознайомилися з IDE AVR Studio 4, навчилися створювати і компілювати код, виконувати покрокове трасування програм, написаних мовою C, програмувати вказаний мікроконтролер.