МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Чернігівська політехніка»

УПРАВЛІННЯ ПЕРИФЕРІЙНИМИ ПРИСТРОЯМИ ЗА ДОПОМОГОЮ ІНКРЕМЕНТАЛЬНОГО ЕНКОДЕРА

МЕТОДИЧНІ ВКАЗІВКИ
до виконання розрахунково-графічної та самостійної роботи
для здобувачів першого (бакалаврського) рівня вищої освіти з дисципліни
«Програмування периферійних пристроїв»
для студентів спеціальності123 – "Комп'ютерна інженерія"

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних та комп'ютерних систем
протокол №7 від 06.06.24

Чернігів 2024

Управління периферійними пристроями за допомогою інкрементального енкодера. Методичні вказівки до виконання розрахунково-графічної та самостійної роботи для здобувачів першого (бакалаврського) рівня вищої освіти з дисципліни “Програмування периферійних пристроїв” для студентів денної форми навчання напряму підготовки 123 – “Комп'ютерна інженерія”. / Укл. Красножон О.В., Роговенко А.І., Красножон А.В. – Чернігів: НУ «Чернігівська політехніка», 2024 48ст.

Укладачі: Красножон Олексій Васильович, канд. техн. наук, доцент кафедри інформаційних та комп'ютерних систем;
Роговенко Андрій Іванович, канд. техн. наук, доцент кафедри інформаційних та комп'ютерних систем;
Красножон Андрій Васильович, канд. техн. наук, доцент, доцент кафедри електричної інженерії та інформаційно-вимірювальних технологій

Відповідальний за випуск: В.М. Базилевич, завідувач кафедрою інформаційних та комп'ютерних систем, к.е.н., доцент.

Рецензент: І.В. Білоус, канд. техн. наук, доцент, завідувач кафедри інформаційних технологій та програмної інженерії Національного університету “Чернігівська політехніка”

ЗМІСТ

ВСТУП

Останнім часом однокристальні мікроконтролери знаходять найрізноманітніше застосування у багатьох галузях людської діяльності, причому межі їх застосування розширюються з кожним днем. Від часу появи перших мікроконтролерів їх складність постійно зростає за рахунок розробки більш досконалих архітектур, нових рішень у сфері напівпровідникових технологій, додавання нових інструкцій. Для вирішення завдань із високим рівнем складності і застосовуються мікроконтролерні системи.

Значних успіхів у галузі створення 8-розрядних мікроконтролерних ядер досягла компанія ATMEL (яка наразі є підрозділом компанії Microchip). Завдяки високій продуктивності та вигідному співвідношенню ціна/якість ці мікроконтролери набули широкої популярності в різних галузях людської діяльності. Фактично, мікроконтролери з ядром AVR виділилися в окремий клас мікроконтролерів для додатків що мають різноманітну направленість.

Компанія ATMEL має власні потужності напівпровідникового виробництва, тому мікроконтролерні ядра, що розробляються нею в різних видах (описані мовами VHDL або Verilog, в електричних схемах), реалізуються надалі у вигляді готових мікросхем із високим ступенем інтеграції.

Згідно рекомендацій виробника, мікроконтролери з ядром AVR призначено для використання в автомобілебудуванні, медицині, промисловій та побутовій електроніці (у тому числі із акумуляторним живленням), комп'ютерних мережах і системах. Це можливо завдяки високим показникам надійності, відносно низькій потужності споживання та наявності декількох режимів функціонування.

Мікроконтролери з ядром AVR різних сімейств і версій підтримуються багатьма програмними продуктами – відлагоджувачі, компілятори мов C/C++, емулятори, операційні системи реального часу, драйвери низького рівня, високорівневі програми – різних виробників: Keil Software, Microchip, ATMEL та ін.

Виробник визначає ядро AVR як універсальне ядро 8-розрядного RISC-мікроконтролера із малим енергоспоживанням, призначене для використання в різноманітних замовних та спеціалізованих інтегральних схемах.

Ці методичні вказівки призначено для ознайомлення та початкового вивчення внутрішньої організації мікроконтролерів з ядром AVR виробництва компанії ATMEL (підрозділ компанії Microchip), особливостей їх функціонування, а також для виконання розрахунково-графічної роботи з дисципліни “Програмування периферійних пристроїв”. Вони не охоплюють усіх доступних можливостей та нюансів функціонування цього сімейства мікроконтролерів, тому за потреби слід звертатися до інших літературних джерел та ресурсів.

1 ІНКРЕМЕНТАЛЬНІ І АБСОЛЮТНІ ЕНКОДЕРИ

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

Енкодери дуже широко застосовуються в промисловості. В залежності від способу формування електричних сигналів вони поділяються на 2 види:

– інкрементальні;

– абсолютні.

Інкрементальний енкодер за один повний оберт формує фіксовану кількість імпульсів, на відміну від них абсолютні дозволяють в будь-який момент часу визначити поточний кут повороту вала, в тому числі і після зникнення та відновлення живлення. Багатообертові абсолютні енкодери, крім того, також підраховують і запам’ятовують кількість повних обертів вала.

В залежності від принципу дії виділяють наступні види енкодерів:

– механічні;

– оптичні;

– резистивні;

– магнітні.

Варто зазначити, що сучасні енкодери адаптовано для роботи як через шинні інтерфейси (наприклад, I^2C, SPI, UART), а також промислові мережі (ModBus, ProfiBus, CAN), саме тому на сьогоднішній день вони практично повністю витіснили застосування сельсинів.

1.1 Інкрементальні енкодери

Інкрементальні енкодери призначено для визначення кутів повороту об’єктів, що обертаються. Вони генерують послідовний імпульсний цифровий код, що містить інформацію про кут повороту об’єкта. Якщо вал зупиняється, то і генерація імпульсів припиняється. Основним робочим параметром інкрементального енкодера є кількість імпульсів за один повний оберт. Миттєве значення кута повороту об’єкта визначають шляхом підрахунку кількості імпульсів від нульової мітки. Для обчислення кутової швидкості об’єкта, обчислювальний елемент тахометра (пристрою, що вимірює частоту обертання) виконує диференціювання кількості імпульсів у часі, таким чином, показуючи відразу величину швидкості, тобто кількості обертів за хвилину.

Вихідні сигнали інкрементального енкодера формуються одночасно в двох каналах, тобто, 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.

Інкрементальний (імпульсний або кроковий) енкодер відноситься до типу енкодерів, призначених для визначення напряму обертання та/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. Такий енкодер завжди з'єднується із лічильним пристроєм, це необхідно для підрахунку кількості імпульсів і перетворення їх в міру переміщення вала.

Кроковий оптичний енкодер складається з наступних компонентів:

– джерела світла;

– диска з мітками;

– фототранзисторної збірки;

– схеми обробки сигналу.

Диск оптичного енкодера (рисунок 1.1) поділено на точно позиціоновані позначки. Кількість відміток визначає кількість імпульсів за один повний оберт. Наприклад, якщо диск поділено на 20 міток, тоді за 5 імпульсів вал повинен повернутися на 90 градусів.

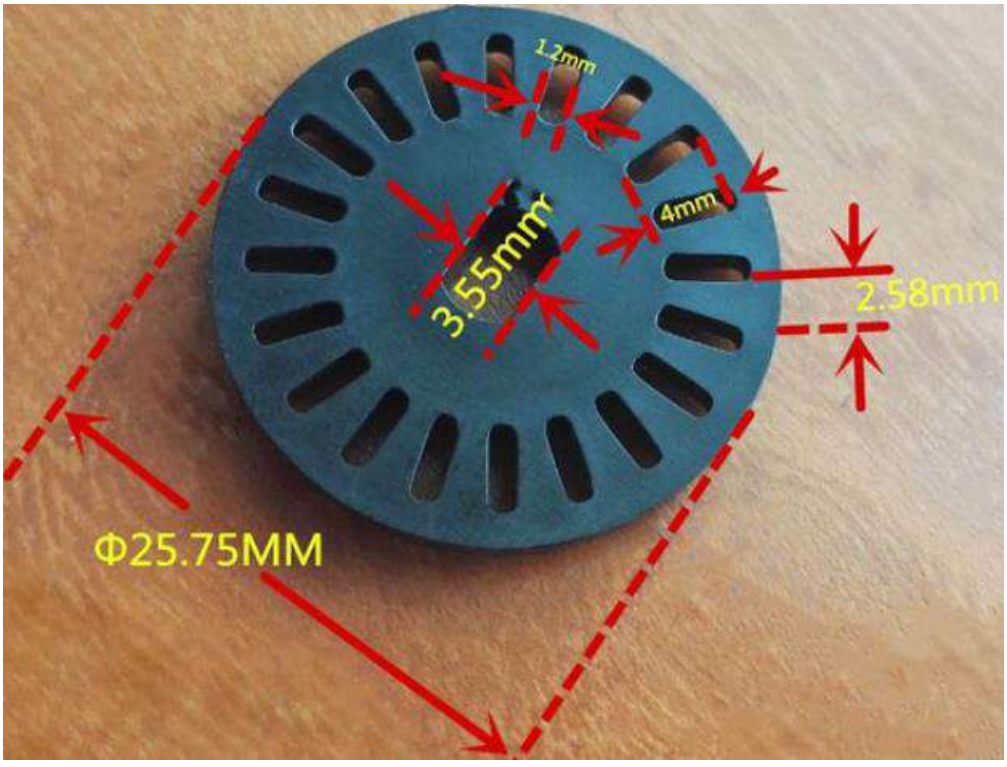

Рисунок 1.1 – Диск із мітками оптичного інкрементального енкодера

Для квадратури виходу енкодера використовуються два вихідних канали (A і B), що дозволяє визначити напрямок обертання валу (за годинниковою стрілкою чи проти неї). Таке визначення ґрунтується на зсуві фази 90°±0°, при цьому, прийнятний допуск для визначення зсуву фаз може досягати ±45°. Енкодер, що має лише один єдиний вихід (A) не може визначати напрямок обертання, тому використовується як тахометр.

Спосіб визначення напрямку обертання валу за допомогою аналізу кута зсуву фаз в каналах А і В представлено нижче, на рисунку 1.2.

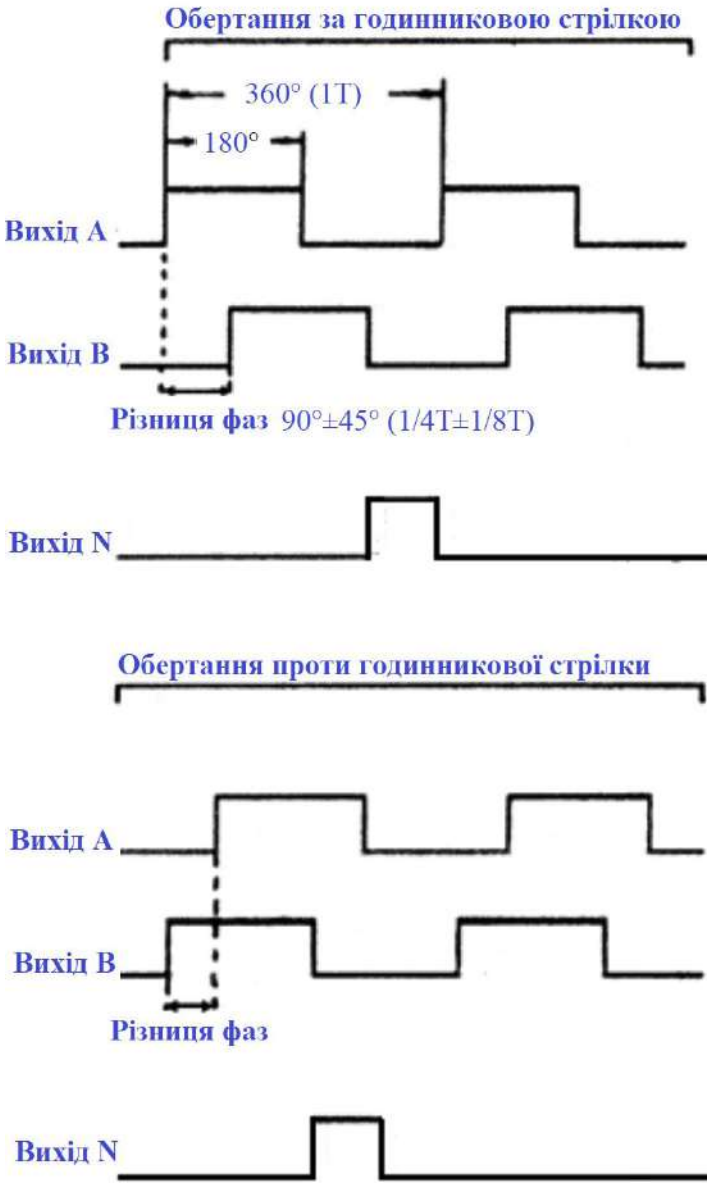

Рисунок 1.2 – Зсув фаз між імпульсами в каналах А і В при різних напрямках обертання інкрементального енкодера

Максимальною частотою відповіді (F_{max}) є частота, за якої енкодер, що обертається, ще може формувати електричні імпульси в каналах А і В, які можливо розрізнити по фазі. Ця частота нерозривно пов'язана із кількістю вихідних імпульсів, на які енкодер реагує за секунду. Отже, енкодер інкрементального типу повинен задовольняти наступному співвідношенню:

$$\frac{N}{60} \cdot k \leq F_{max}, \quad (1.1)$$

де N – кількість обертів інкрементального енкодера за хвилину;

k – роздільна здатність інкрементального енкодера.

Роздільна здатність інкрементального енкодера – це кількість вихідних імпульсів за один повний оберт вала.

В інкрементальних енкодер, що мають вказівник нульової мітки або імпульс повного оберту (вихід N на рисунку 1.2), імпульс на цьому виході з'являється при кожному оберті вала. Функція вказівника нуля може використовуватися для скидання пов’язаного із енкодером зовнішнього лічильника або для фіксації початкової (нульової) позиції.

Для механічного з’єднання вала датчика із зовнішнім механізмом слід використовувати спеціальний перехідник гнучкого типу, призначений для компенсації можливого биття валів, як в радіальному, так і в осьовому напрямках. Це дозволяє істотно знизити імовірність передчасного зносу підшипників вала. Навіть незначний люфт, що виник в осьовому напрямку валу, може призвести до повної електричної відмови енкодера. Це пов’язано з тим, що для досягнення високої роздільної здатності, оптичний диск і матриця, яка зчитує, розташовуються в безпосередній близькості один від одного, і мінімальне осьове биття вала може призвести до їх механічного контакту, що в підсумку призведе до руйнування нанесених на диск міток.

1.2 Абсолютні енкодери

Основними робочими характеристиками абсолютних енкодерів, як оптичних, так і магнітних, є:

– число кроків, тобто унікальних кодових слів на оберт;

– кількість таких обертів (при цьому не потрібна первинна установка та ініціалізація датчика).

Найбільш поширені типи вихідного сигналу – це код Грея, паралельний код, інтерфейси Profibus-DP, CANopen, DeviceNet, SSI, LWL, через які також здійснюється програмування датчиків.

Абсолютний енкодер відноситься до типу енкодерів, який генерує унікальний код для кожної позиції вала. На відміну від інкрементального, лічильник імпульсів не потрібен, оскільки кут повороту завжди відомий. Абсолютний енкодер формує сигнал, як під час обертання, так і в режимі спокою. Диск абсолютного енкодера (рисунок 1.3) відрізняється від диска крокового, оскільки має кілька концентричних доріжок. Із допомогою кожної доріжки формується унікальний двійковий код для конкретної позиції вала.

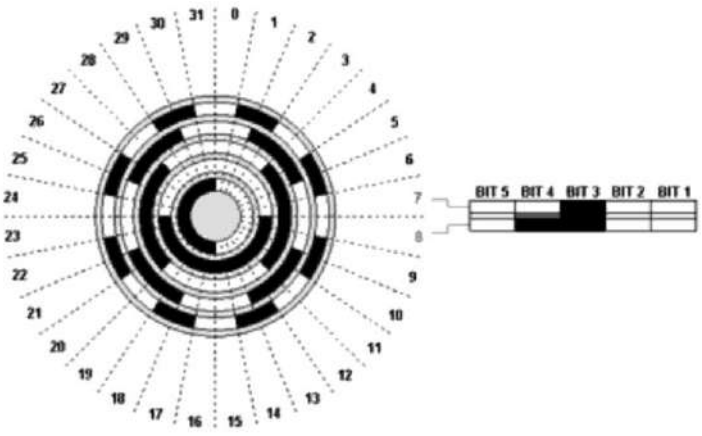

Рисунок 1.3 – Кодовий диск абсолютного енкодера

Абсолютний енкодер не втрачає свого значення при провалах живлення і не вимагає повернення в початкову позицію. Сигнал абсолютного енкодера майже не зазнає впливу завад, для нього не потрібна точна установка валу. Крім того, навіть якщо кодований сигнал не може бути прочитаний енкодером (наприклад, вал обертається надто швидко), правильний кут обертання буде зареєстрований, коли швидкість обертання зменшиться. Абсолютні енкодери є стійкішими до впливу вібрацій.

1.3 Способи кодування станів абсолютного енкодера

Як відомо, двійковий код – це найпоширеніший код, який може оброблятися безпосередньо мікроконтролером, і є основним кодом для обробки цифрових сигналів. Такий код складається тільки із цифр 0 і 1.

Максимальне число, яке може бути виражено двійковим кодом, залежить від кількості розрядів, що використовується, тобто від кількості бітів комбінації, що виражає число.

Переходячи від чисел до фізичних величин, можна сформулювати вищенаведене твердження в більш загальному вигляді: найбільша кількість значень m будь-якої величини (кута повороту, напруги, струму, тощо), яке може бути виражено двійковим кодом, залежить від числа розрядів n, що використовуються, як $m = 2^n$. Двійковий код є багатокроковим кодом. Це означає, що при переході від одного значення в сусіднє можуть змінюватися кілька бітів одночасно. Наприклад, $3_{10} = 011_2$, а $4_{10} = 100_2$. Видно, що при переході від 3 до 4, всі біти змінюють свій стан на протилежний одночасно. Зчитування такого коду із кодового диска призвело б до того, що (через неминучі відхилення в процесі виробництва диска) зміна інформації від кожної із доріжок окремо ніколи не буде одночасною. В свою чергу, це призведе до того, що при переході від одного числа до іншого короткочасно буде видано невірну інформацію. Так при переході від 3 до 4 дуже імовірна короткочасна поява 7, якщо, наприклад, старший біт змінив своє значення трохи раніше за інших. Тобто, використання звичайного двійкового коду може призвести до серйозних помилок, оскільки сусідні комбінації можуть відрізнятися не одним, а кількома розрядами. Щоб уникнути цього, застосовується т. зв. однокроковий код, наприклад, код Грея.

Код Грея краще звичайного двійкового, бо має властивість безперервності бінарної комбінації: зміна кодованого числа на 1 відповідає зміні кодової комбінації тільки в одному розряді (відстань Хеммінга завжди рівна 1). Він будується на базі двійкового за таким правилом:

1) старший розряд переноситься без змін;

2) кожний наступний розряд – інвертується, якщо попередній розряд вихідного двійкового коду дорівнює “1”.

Такий алгоритм побудови може бути формально представлено як результат складання по модулю 2 вихідної комбінації двійкового коду з такою ж комбінацією, але зсунутою на один розряд вправо. При цьому крайній правий розряд зсунутої комбінації відкидається.

Таким чином, похибка при зчитуванні інформації з механічного кодового диска при переході від одного числа до іншого призведе до того, що перехід від одного положення до іншого буде лише трохи зміщений у часі, проте видача абсолютно невірного значення кутового положення при переході від одного положення до іншого виключається повністю. Перевагою кода Грея є також його здатність до дзеркального відображення інформації, тобто, інвертуючи лише старший біт, можна змінювати напрям лічби і, таким чином, визначати фактичний напрямок обертання, оскільки значення, що видається, може бути зростаючим або спадаючим при одному і тому ж фізичному напрямку обертання.

Оскільки інформація, виражена кодом Грея, є лише закодованим представленням, що не несе реальної числової інформації, він повинен бути перетворений в стандартний двійковий код перед подальшою обробкою. Здійснюється це за допомогою перетворювача із коду Грея в двійковий код, який легко реалізується за допомогою кіл із логічних елементів “виключне або” (xor) як програмним, так і апаратним способом, який зображено нижче, на рисунку 1.4.

Нижче, в таблиці 1.1, представлено співвідношення між розрядами коду Грея і відповідним двійковим кодом. З аналізу цієї таблиці видно, що при переході від одного числа до іншого (сусіднього) лише один біт інформації змінює свій стан, якщо число представлено кодом Грея, в той час як в двійковому коді можуть змінити свій стан кілька бітів одночасно. Код Грея завжди формується на виході, отже, він ніколи не має помилку читання і застосовується в багатьох абсолютних енкодер.

Таблиця 1.1 – Співвідношення розрядів двійкового коду та коду Грея

Десятковий код	**Двійковий код**				**Код Грея**
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0 0 0 0
1	0	0	0	**1**	0 0 0 1
2	0	0	**1**	**0**	0 0 1 1
3	0	0	1	**1**	0 0 1 0
4	0	**1**	**0**	**0**	0 1 1 0
5	0	1	0	**1**	0 1 1 1
6	0	1	**1**	**0**	0 1 0 1
7	0	1	1	**1**	0 1 0 0
8	1	**0**	**0**	**0**	**1** 1 0 0
9	1	0	0	**1**	1 1 0 1
10	1	0	**1**	**0**	1 1 1 1
11	1	0	1	**1**	1 1 1 0
12	1	**1**	**0**	**0**	1 0 1 0
13	1	1	0	**1**	1 0 1 1
14	1	1	**1**	**0**	1 0 0 1
15	1	1	1	**1**	1 0 0 0

Для зручності сприйняття таблиці 1.1 біти, що змінюють свій стан при переході від одного числа до іншого, виділено жирним шрифтом.

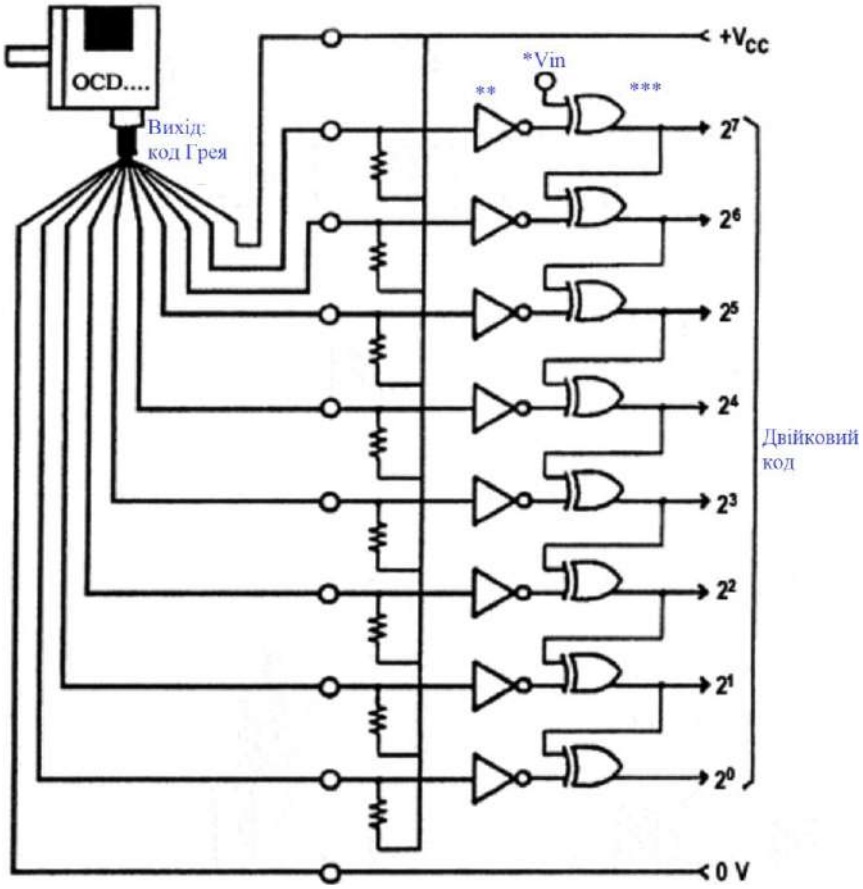

Рисунок 1.4 – Схема перетворювача із коду Грея в двійковий

Для комбінаційної схеми перетворювача кодів, яку зображено на рисунку 1.4, прийнято наступні позначення:

1 *Код Грея може логічно перетворюватися в двійковий код, коли на лінії живлення Vin утримується низький логічний рівень (0 В).

2 **Інвертор.

3 ***Виключне або.

Звичайний однокроковий код Грея підходить для роздільної здатності, яку може бути представлено у вигляді числа, зведеного до степеня двійки. У випадках, де треба реалізувати іншу роздільну здатність, зі звичайного коду Грея вирізається і використовується середня його ділянка. Таким чином, зберігається “однокроковість” коду. Однак числовий діапазон починається не з 0, а зміщується на певне значення. При обробці інформації від згенерованого сигналу віднімається половина різниці між початкової і редукованою роздільною здатністю. Наприклад, такі здатності як 360° для вираження кута часто реалізуються цим методом. Так 9-ти бітний код Грея, що має 512 кодових комбінацій, урізаний з обох боків на 76 кроків буде дорівнювати 360°.

Вимірювальна система абсолютного енкодера складається із поворотної вісі, монтованої на двох високо прецизійних підшипниках,

кодового диску, встановленого на вісь, а також оптоелектронної матриці зчитування і схеми обробки сигналів. В якості джерела світла виступає світлодіод, інфрачервоні промені якого просвічують кодовий диск і потрапляють на фототранзисторну матрицю, розташовану зі зворотного боку кодового диска. При кожному кроці кутового положення кодового диска темні ділянки диску запобігають потраплянню світла на ті чи інші фототранзистори матриці. Таким чином, темні і світлі ділянки кожної з доріжок будуть відображені на фототранзисторній матриці і перетворені в електричні сигнали (0 або 1). Електричні сигнали, в свою чергу, обробляються операційними підсилювачами і вихідними драйверами для видачі у вигляді n-бітного двійкового коду. Зміни інтенсивності джерела світлового потоку реєструються за допомогою додаткового сенсора і компенсуються електронною схемою.

1.4 Різновиди енкодерів

Однообертовими (або single-turn) енкодерами називаються такі датчики, які видають абсолютне значення в межах одного повного оберту (тобто на всі 360°). Після одного оберту код є повністю пройденим і починається знову зі свого початкового значення. Ці датчики застосовуються, переважно, для вимірювання кута повороту, наприклад, в антенних системах, ексцентричних колінчастих пресах і т.д. Внутрішню будову однооберотового енкодера зображено нижче, на рисунку 1.5.

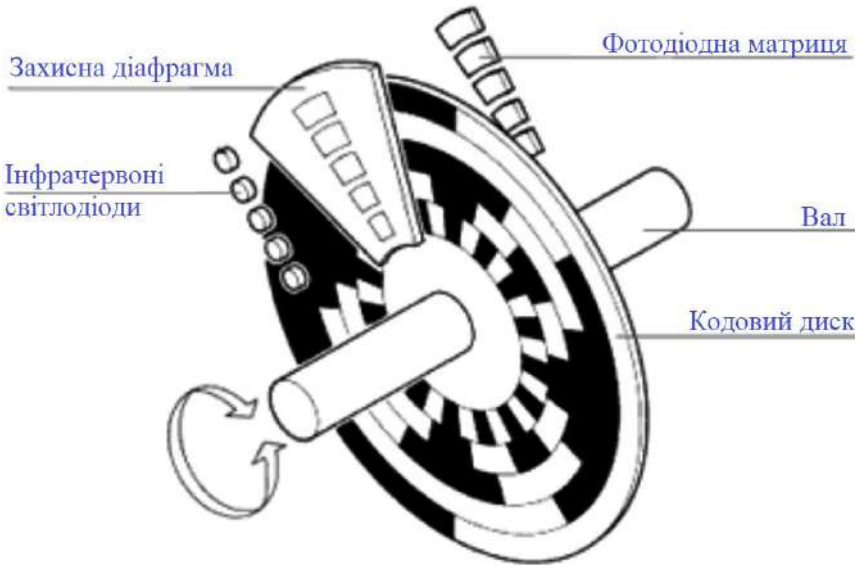

Рисунок 1.5 – Внутрішня будова однооберотового енкодера

Лінійні переміщення передбачають застосування вимірювальної системи з кількістю обертів n. Наприклад, в лінійних приводах або при вимірювання за допомогою зубчастої вимірювальної штанги, застосування однообертових датчиків є неприйнятним. В цьому випадку приходять на допомогу датчики, де додатково до вимірювання кута повороту в межах одного оберту також відбувається реєстрація кількості обертів за допомогою

додатково вбудованого передавального механізму, тобто свого роду редуктора з декількох кодових оптичних дисків, утворюючи, таким чином, багатообертовий енкодер (або multi-turn), внутрішню будову якого зображено нижче, на рисунку 1.6.

Рисунок 1.6 – Внутрішня будова багатообертового енкодера

Оптичні енкодери мають жорстко закріплений співвісно валу скляний диск із прецизійною оптичною шкалою. При обертанні вала, оптопара зчитує інформацію, а електронна схема перетворює її в послідовність дискретних електричних імпульсів. Абсолютні оптичні енкодери – це датчики кута повороту, де кожному положенню вала відповідає унікальний цифровий вихідний код, який поряд з числом обертів є основним робочим параметром датчика. Абсолютні оптичні енкодери, так само як і інкрементальні, зчитують і фіксують параметри обертання оптичного диска. Внутрішню будову оптичного енкодера зображено нижче, на рисунку 1.7.

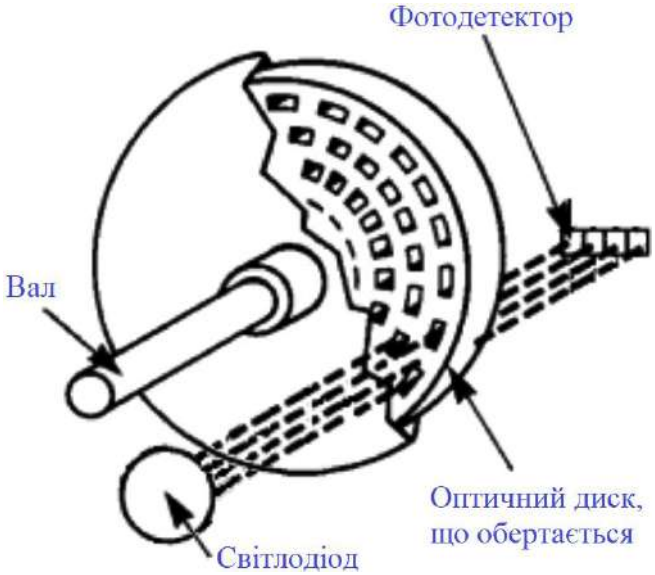

Рисунок 1.7 – Внутрішня будова оптичного енкодера

Магнітні енкодери з високою точністю реєструють проходження магнітних полюсів магнітного елемента безпосередньо поблизу чутливого елемента (за допомогою датчиків Холла), перетворюючи ці дані у відповідний цифровий код. Внутрішню будову магнітного енкодера зображено нижче, на рисунку 1.8.

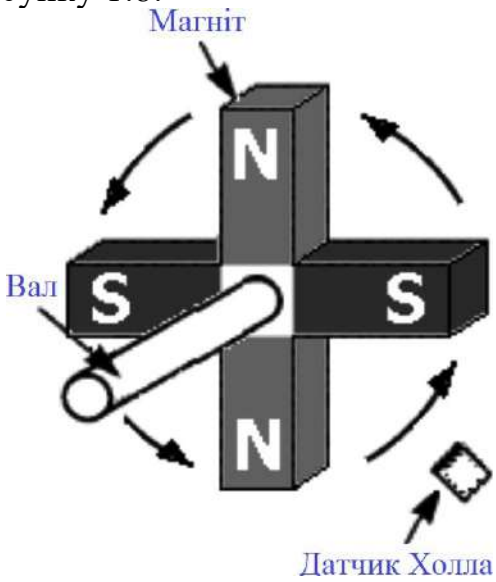

Рисунок 1.8 – Внутрішня будова магнітного енкодера

Магніторезистивний енкодер складається із котушки, яка закріплюється на валу, розташованої в магнітному полі постійного магніта. При обертанні котушки її витки будуть змінювати своє положення відносно ліній поля, вони будуть то паралельні полю, то перпендикулярні йому, відповідно, струм в котушці буде змінюватися. Таким чином, струми, що протікають через котушку, будуть змінюватися в залежності від кута повороту вала. Внутрішню будову магніторезистивного енкодера зображено нижче, на рисунку 1.9.

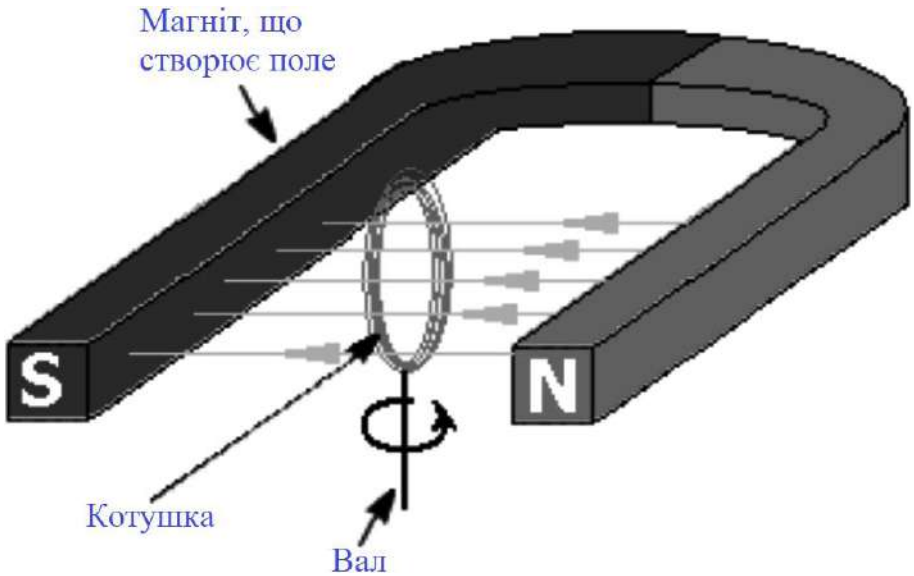

Рисунок 1.9 – Внутрішня будова магніторезистивного енкодера

Механічні та оптичні енкодери із послідовним виходом містять диск із діелектрика або скла із нанесеними опуклими ділянками, що пропускають струм/світло або ні. Зчитування абсолютного кута повороту диска проводиться лінійкою перемикачів або контактів у випадку механічної схеми, або лінійкою оптронів у випадку оптичної. Вихідні сигнали являють собою код Грея, що дозволяє позбутися неоднозначної інтерпретації сигналу.

З'єднання енкодера із обертовим об'єктом здійснюється за допомогою нормального або порожнистого вала, останній може бути як наскрізним, так і непрямим (тупиковим). Вал об'єкта, що обертається, і вал енкодера з'єднують механічно за допомогою гнучкої або жорсткої муфти. В якості альтернативи, енкодер монтують безпосередньо на вал об'єкта, якщо енкодер має порожнистий вал. У першому випадку імовірна неспіввісність і допустимі биття компенсуються деформацією гнучкої втулки, у другому – можлива фіксація енкодера за допомогою штифта.

1.5 Параметри енкодерів

При виборі енкодерів слід звертати увагу на наступні важливі параметри:

1. Число імпульсів на один оберт (число бітів для абсолютних енкодерів). Даний показник впливає на точність системи: чим більша кількість імпульсів – тим вище точність, оскільки кожному імпульсу буде відповідати менший кут повороту.

2. Вал, отвір під кріплення вала (а також діаметри цих отворів). Від цього залежить те, яким чином на енкодер передаватиметься обертання, або об'єкт буде з'єднуватися із отвором енкодера, або на вал енкодера буде передаватися обертання за допомогою, наприклад, зубчастої або ремінної передачі.

3. Тип вихідного сигналу енкодера (HTL, TTL, RS-422, двійковий код, код Грея та ін.). Даний параметр впливає на спосіб формування вихідного сигналу енкодера і подальшу його передачу.

4. Напруга живлення. Від цього параметра залежить робота системи і точність отримання сигналів.

5. Довжина кабелю або тип роз'єму впливають на можливості установки робочої системи.

6. Інші вимоги за кріпленням (необхідність муфти, монтажного фланця, кріпильної штанги та ін.). Даний параметр впливає на стійкість установки, тобто на точність системи.

7. Ступінь захисту енкодера від проникнення пилу і вологи.

1.6 Особливості реалізації інкрементального енкодера в навчально-відлагоджувальному стенді EV8031/AVR

У навчально-відлагоджувальному стенді EV8031/AVR інкрементальний енкодер реалізовано на другій платі розширення. Він являє собою

механічний (магнітний) енкодер, який не має виходу імпульсу повного оберту. Інкрементальний енкодер має безпосередній зв'язок із ATmega8515, для забезпечення якого використовуються 3 сигнальних лінії. Кожна з них підключається до відповідної ніжки порту вводу/виводу мікроконтролера (МК) AVR, системний контролер стенда (мікросхема зовнішньої пам'яті) в цьому випадку залишається незадіяною. Таким чином, робота із інкрементальним енкодером буде здійснюватися виключно шляхом опитування стану ліній вводу/виводу мікроконтролером напряму. Схему електричну принципову взаємодії інкрементального енкодера із МК ATmega8515, реалізовану у навчально-відлагоджувальному стенді, зображено нижче, на рисунку 1.10.

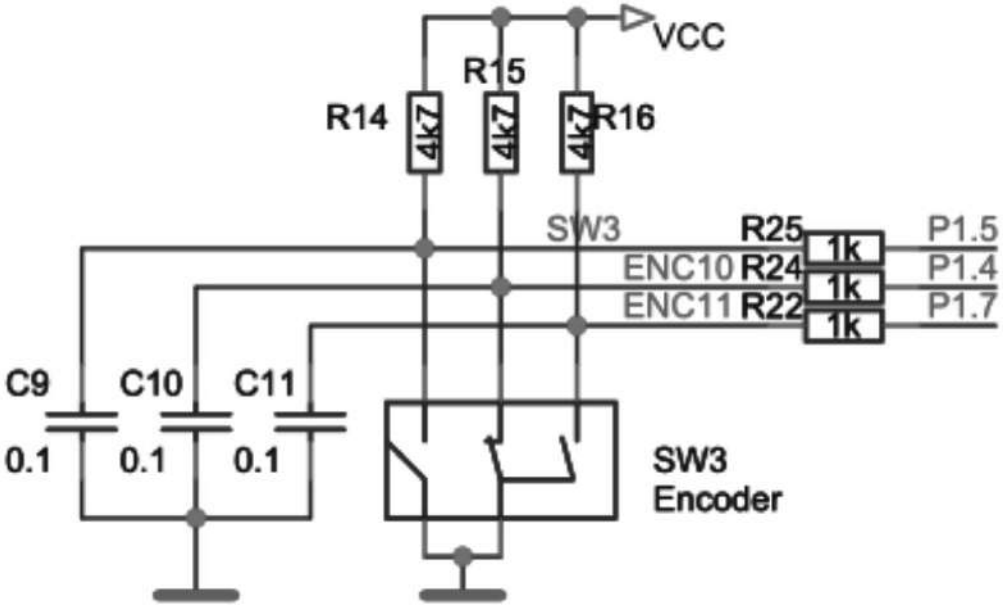

Рисунок 1.10 – Схема електрична принципова взаємодії інкрементального енкодера із МК ATmega8515

Вивод P1.5 інкрементального енкодера підключається до виводу PB5 МК AVR, сигнал на цьому виводі може бути сформований енкодером в тому випадку, якщо користувач натисне на інкрементальний енкодер як на кнопку.

Вивод P1.4 інкрементального енкодера підключається до виводу PB4 МК AVR. Даний вивод є виходом каналу B енкодера.

Вивод P1.7 інкрементального енкодера підключається до виводу PB7 МК AVR. Даний вивод є виходом каналу A енкодера.

Очевидно, що для опитування стану виводів інкрементального енкодера відповідні виводи порту B (4, 5 і 7) на боці МК повинні бути налаштовані як вхідні. Однак робити цю дію окремо не варто, оскільки за замовчуванням після подачі напруги живлення, виводи всіх портів налаштовуються як вхідні (Tri-state Hi-Z).

При написанні програм управління різними периферійними пристроями та механізмами за допомогою енкодерів, гарним тоном є використання механізму переривань. Це позбавляє необхідності постійно опитувати стан виводів порту, оскільки аналіз сигналів, що формуються

енкодером, починається тільки після появи умов для відповідного переривання. На жаль, така можливість недоступна в даному навчальному стенді, оскільки функції входів зовнішніх переривань (INT0 – INT2) ATMega8515 виконують відповідні виводи портів D і E, а не порту B.

Залежно від напрямку обертання валу-ручки енкодера, черговість появи імпульсів в каналах A і B буде різною. Таким чином, якщо обертати енкодер проти годинникової стрілки, то спочатку з'являтиметься імпульс в каналі B (вивод PB4), а вже потім в каналі A (вивод PB7). При повороті енкодера за годинниковою стрілкою – спочатку в каналі A, а вже потім – в каналі B. Оскільки джерело напруги живлення і номінали струмообмежуючих резисторів підібрані таким чином, щоб в каналах формувалися логічні рівні напруги, можна показати спрощену часову діаграму сигналів на виходах енкодера. Дану діаграму зображено нижче, на рисунку 1.11. В ідеальному випадку імпульси в каналах A і B "відстають" один від одного на чверть періоду.

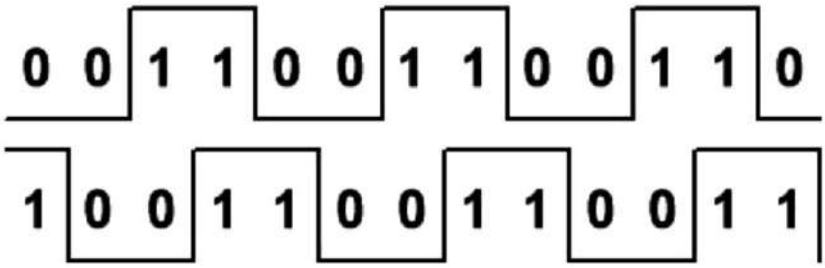

Рисунок 1.11 – Послідовність імпульсів в каналах енкодера

Аналіз діаграми, зображеної на рисунку 1.11, дозволяє побудувати алгоритм опитування та аналізу станів інкрементального енкодера, використовуючи елементи теорії цифрових автоматів, тобто необхідно програмно реалізувати певний кінцевий автомат. Для цього з певною дискретністю за часом слід опитувати канали енкодера і порівнювати поточний їх стан із знову отриманими, на підставі цього можна робити висновки про те, в який бік було повернуто вал-ручку енкодера.

Очевидно, що частота опитування станів енкодера повинна відповідати наступним вимогам:

– повинна бути такою, щоб повністю виключати можливість пропуску навіть одного імпульсу в будь-якому із трьох каналів;

– дозволяти МК виконувати команди основної програми.

Інкрементальний енкодер, який реалізовано в навчально-відлагоджувальному стенді формує 12 імпульсів на один повний оберт, тобто один імпульс на кожні 30º. Обертання енкодера здійснюється вручну, максимаьна частота імпульсів, що надходять від енкодера, не перевищує 1 КГц (її реальне значення – близько 833 Гц).

Таким чином, частота опитування каналів інкрементального енкодера повинна становити 1 КГц. Зменшення значення цієї частоти дозволить уникнути можливих помилкових спрацювань, які можуть з'являтися у зв'язку із появою "дріб'язку" контактів (цей дріб'язок може проявлятися в

процесі тривалої або інтенсивної експлуатації енкодера). Однак, як можна побачити із рисунка 1.10, для зниження впливу цього явища використовуються керамічні конденсатори С9 – С11, які включено паралельно до кожного із виводів енкодера. Зменшення впливу “дріб’язку” при такому способі пов'язано із властивістю конденсатора: напруга на конденсаторі не може змінюватися стрибкоподібно (другий закон комутації).

1.7 Приклади програм для МК AVR по роботі із інкрементальним енкодером

Нижче наведено приклад програми мовою асемблер, яка здійснює управління напівпровідниковим світлодіодним індикатором за допомогою інкрементального енкодера. При кожному повороті вала-ручки енкодера на один тік за годинниковою стрілкою, двійковий код, що відображається на світлодіодному індикаторі, збільшується на 1, а при кожному повороті на один тік проти годинникової стрілки – зменшується на 1. При натисканні на інкрементальний енкодер як на кнопку, здійснюється гасіння всіх світлодіодів індикатора. Затримка між послідовними опитуваннями стану інкрементального енкодера становить 1 мс і організовується програмно.

```
;*** 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 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 ***

Infinite_loop:; нескінченний цикл

      rcall long_delay  ;викликаємо затримку, щоб не читати стан енкодера
занадто часто

      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 B енкодера не було "0" і в каналі A 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 Decr_state         ;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_state         ;переходимо до збільшення стану енкодера

      rjmp Next_iterate ;якщо в жодному з каналів не стався фронт - йдемо далі

Incr_state:
      inc encoder_data  ;інкремент кількості імпульсів, отриманих від енкодера
      rjmp Next_iterate ;переходимо до подальших дій

Decr_state:
      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 diode, encoder_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 х 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
```

Аналогічну програму, написану мовою С, представлено нижче.

```
/*** 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>
#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 encoder_data;
//регістр, який зберігає попередній стан інкрементального енкодера
volatile unsigned char last_encoder_state;
//регістр, який зберігає наступний стан інкрементального енкодера
volatile unsigned char next_encoder_state;

int main (void) {
      //початкова ініціалізація контролера
      ACSR = 1 << ACD;  //відключення живлення аналогового компаратора
      MCUCR = 1 << SRE; //дозволяємо роботу із зовнішньою пам'яттю

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

      last_encoder_state = PINB; //читаємо поточний стан каналів енкодера
      last_encoder_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"

      //Переходимо в нескінченний цикл опитування стану енкодера
      while (1) {
```

```
            _delay_ms (1); //викликаємо затримку, щоб не читати стани енкодера
занадто часто
            next_encoder_state = PINB; //читаємо поточний стан каналів енкодера
            next_encoder_state & = 0b10010000; //отримуємо стан бітів порту, що
відповідають за поточний стан енкодера

            if (next_encoder_state!= last_encoder_state) { //якщо стан енкодера
змінився
                  if (last_encoder_state == 0x00) { //якщо в каналі B енкодера
не було "0" і в каналі A теж не було "0"
                        if (next_encoder_state == 0b00010000) { //якщо в каналіA
енкодера залишився "0", а в каналі B стала "1"
                              encoder_data--;  //зменшення значення лічильника
кількості імпульсів, одержуваних від енкодера
                        }
                        if (next_encoder_state == 0b10000000) { // якщо в каналі
A енкодера стала "1", а в каналі B залишився "0"
                              encoder_data++;  //збільшуємо значення лічильника
кількості імпульсів, одержуваних від енкодера
                        }
                  }
            }

            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
                  encoder_data = 0x00; //власне, скидання стану енкодера
            }

            diode = encoder_data; //оновлюємо вміст змінної стану світлодіодної
лінійки
            *p = diode; //оновлюємо стан індикатора
            last_encoder_state = next_encoder_state; //підготовка до наступної
ітерації опитування стану енкодера
            //присвоюємо попередньому стану енкодера його поточний стан
      }
      return 0;
}
```

2 ТАЙМЕРИ/ЛІЧИЛЬНИКИ В МІКРОКОНТРОЛЕРАХ AVR

2.1 Таймери у складі мікроконтролера ATMega8515

Мікроконтролер ATMega8515 має в складі своєї периферії 2 таймери/лічильники: T0 і T1.

Таймер/лічильник T0 має розрядність 8 біт. Він може використовуватися для відліку або вимірювання інтервалів часу, як лічильник зовнішніх подій, а також як генератор сигналів ШІМ (широтно-імпульсна модуляція).

Таймер/лічильник T1 має розрядність 16 біт. Він може використовуватися для відліку або вимірювання інтервалів часу, як лічильник зовнішніх подій, як генератор ШІМ-сигналів. Крім того, він може запам’ятовувати свій поточний стан при появі деякого зовнішнього сигналу (т. зв. режим захоплення).

Таймери/лічильники можуть бути джерелами генерації переривань. Для дозволу обробки переривань від таймерів в мікроконтролері ATMega8515 використовується регістр масок переривань TIMSK (Timer Interrupt Mask). Для дозволу обробки будь-якого переривання потрібно встановити в “1” відповідний біт цього регістра (а також не забути встановити прапорець “I” глобального дозволу переривань). Для індикації моменту настання переривань від таймерів/лічильників використовується регістр прапорців TIFR (Timer Interrupt Flag Register). При настанні будь-якої визначенної для таймерів події (переповнення, співпадіння) відповідний прапорець регістра TIFR встановлюється в “1”. При вході мікроконтролера в підпрограму-обробник відповідного переривання, такий прапорець апаратно скидається в “0” самим ядром мікроконтролера. Крім того, будь-який прапорець регістра TIFR може бути скинуто програмно: шляхом запису в нього логічної “1”.

2.2 Переддільник для таймерів/лічильників T1 і T0

Мікроконтролер ATMega8515 містить у своєму складі переддільник частоти, який служить для формування тактових імпульсів, що подаються на лічильні входи таймерів/лічильників. Структурну схему переддільника зображено нижче на рисунку 2.1.

З рисунка видно, що до складу переддільника входить, власне, сам 10-бітний переддільник частоти і 2 вихідних мультиплексори, кожен із яких формує тактовий сигнал для відповідного таймера/лічильника. На вхід 10-бітного переддільника надходить сигнал $clk_{I/O}$ від кварцевого резонатора (який має частоту 7,3728 МГц), а також сигнал скидання PSR10. Вихідні сигнали переддільника подаються на входи мультиплексорів симетрично. Джерело тактового сигналу вибирається за допомогою бітів CS12 – CS10 (Clock Selection) для таймера/лічильника T1 або бітів CS02 – CS00 для таймера/лічильника T0. Окрім цього, передбачено можливість їх тактування зовнішніми тактовими імпульсами, які повинні надходити на входи T0 або T1 відповідно.

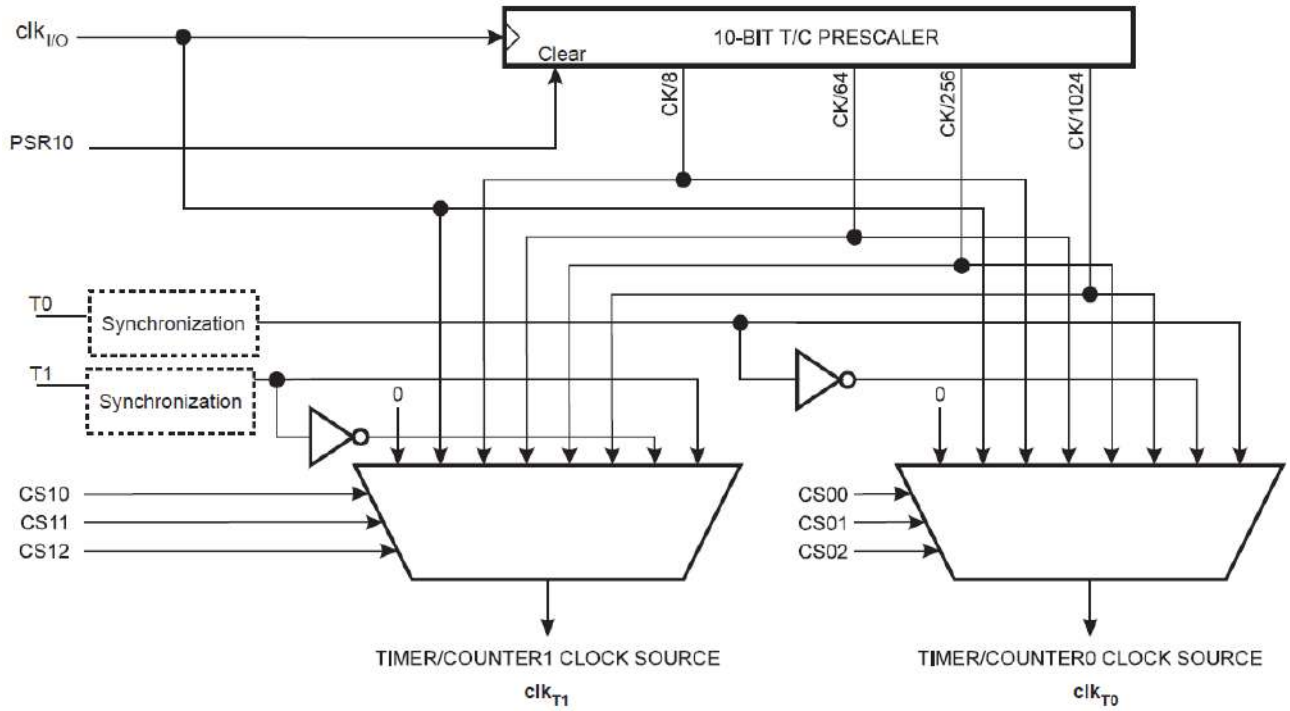

Рисунок 2.1 – Структурна схема переддільника частоти для таймерів/лічильників T1 і T0

Слід звернути особливу увагу на те, що 10-бітний переддільник працює безперервно, тому для встановлення моменту початку відліку часу рекомендується виконувати його скидання. У МК ATMega8515 для цієї потреби використовується один із бітів регістра спеціальних функцій вводу/виводу SFIOR, формат якого показано на рисунку 2.2.

Bit	7	6	5	4	3	2	1	0
	–	XMBK	XMM2	XMM1	XMM0	PUD	–	PSR10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.2 – Формат регістра SFIOR в МК ATMega8515

Результатом запису логічної “1” в біт PSR10 є скидання переддільника. У наступному після запису такті, цей біт апаратно скидається самим мікроконтролером, сигналізуючи про те, що скидання переддільника виконано.

2.3 Таймер/лічильник T0

Структурну схему таймера/ лічильника T0 показано на рисунку 2.3.

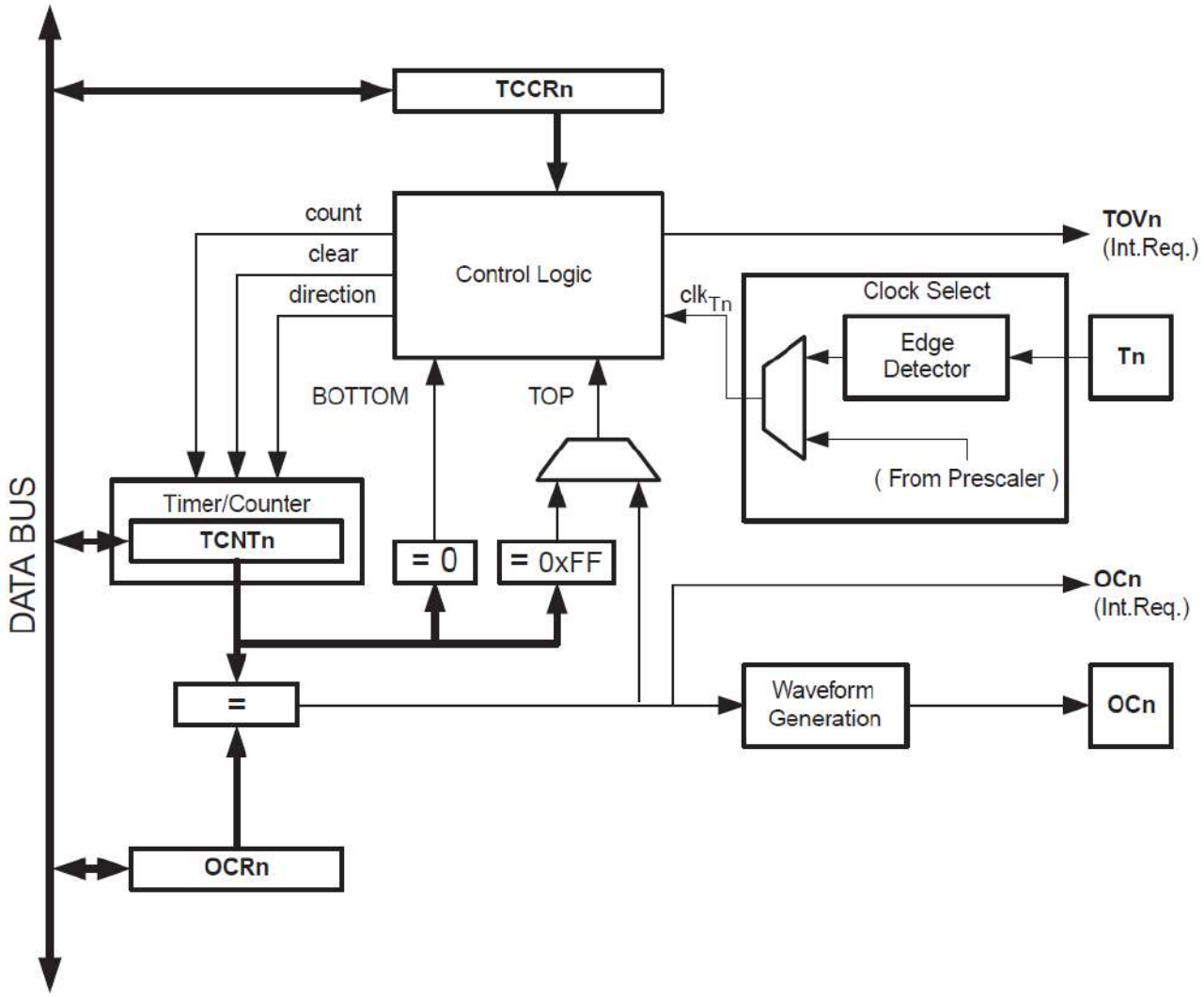

Рисунок 2.3 – Структурна схема таймера/лічильника T0

Для забезпечення керування перериваннями від таймера/лічильника T0, використовуються окремі біти регістра TIMSK, формат якого показано нижче, на рисунку 2.4.

Bit	7	6	5	4	3	2	1	0
	TOIE1	OCIE1A	OCIE1B	–	TICIE1	–	TOIE0	OCIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.4 – Формат регістра TIMSK в МК ATMega8515 для роботи із T0

Встановлення в логічну “1” біту TOIE0 (Timer Overflow Interrupt Enable) дозволяє обробку переривання по переповненню T0, а біту OCIE0 (Output Compare Interrupt Enable) – по співпадінню.

Формат регістра TIFR, що використовується для індикації моменту надходження переривань від таймера/лічильника T0, показано на рисунку 2.5.

Прапорець TOV0 (Timer Overflow) відображає переповнення таймера/лічильника T0, а OCF0 (Output Compare Flag) – співпадіння.

	TOV1	OCF1A	OCF1B	–	ICF1	–	TOV0	OCF0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.5 – Формат регістра TIFR в МК ATMega8515 для роботи із T0

Як видно з рисунка 2.3, до складу таймера/лічильника T0 входить 3 регістра вводу/виводу (кожен із яких має 8-бітну розрядність):

– TCCR0 – регістр управління;

– TCNT0 – лічильний регістр;

– OCR0 – регістр порівняння.

Лічильний регістр TCNT0 входить до складу модуля реверсивного лічильника. Залежно від режиму роботи, вміст лічильного регістра може скидатися в 0, інкрементуватися або декрементуватися по кожному імпульсу тактового сигналу clk_{T0} (який надходить від схеми вибору тактового сигналу). Регістр TCNT0 в будь-який момент часу є доступним як для читання, так і для запису. Після подачі живлення до МК або його скидання, цей регістр містить нульове значення. При досягненні таймером/лічильником T0 максимального значення (0xFF), встановлюється прапорець TOV0 в регістрі TIFR. Для того, щоб при цьому згенерувати переривання, повинен бути встановлений в “1” біт TOIE0 в регістрі TIMSK, а також прапорець I в регістрі SREG.

Регістр порівняння OCR0 входить до складу блоку порівняння таймера/лічильника T0. Під час роботи T0 безперервно (в кожному такті) проводиться порівняння вмісту регістрів OCR0 і TCNT0. При співпадінні цих значень встановлюється прапорець OCF0 в регістрі TIFR. Для того, щоб при цьому згенерувати переривання, повинен бути встановлений в "1" біт OCIE0 в регістрі TIMSK, а також прапорець I в регістрі SREG.

Слід пам’ятати, що будь-яка операція запису в лічильний регістр TCNT0 заблокує роботу блоку порівняння і формування сигналу про співпадіння на 1 період тактового сигналу. У зв’язку з цим, рекомендується перед зміною вмісту регістра OCR0 здійснювати зупинку таймера/лічильника і скидання його лічильного регістра TCNT0, потім таймер можна запустити знову.

Для управління роботою таймера/лічильника T0 призначено регістр TCCR0, формат якого показано на рисунку 2.6.

Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.6 – Формат регістра TCCR0 в МК ATMega8515

Розглянемо призначення лише основних бітів цього регістра.

Біти WGM01, WGM00 (Waveform Generation Mode) відповідають за вибір режиму роботи таймера/лічильника T0 відповідно до правил, показаних нижче, в таблиці 2.1.

Таблиця 2.1 – Налаштування режимів роботи таймера/лічильника T0

Mode	**WGM01 (CTC0)**	**WGM00 (PWM0)**	**Timer/Counter Mode of Operation**	**TOP**	**Update of OCR0 at**	**TOV0 Flag Set on**
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

Біти CS02 – CS00 відповідають за вибір джерела тактового сигналу для T0 відповідно до правил вибору, показаних в таблиці 2.2.

Таблиця 2.2 – Вибір тактового сигналу таймера/лічильника T0

CS02	**CS01**	**CS00**	**Description**
0	0	0	No clock source (Timer/counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Слід також пам’ятати, що запис нового значення у біти CS02–CS00 не тільки вибирає можливе джерело тактового сигналу, але і здійснює запуск/зупинку таймера/лічильника T0.

До режимів роботи таймера/лічильника T0, що використовуються найбільш часто, відносяться:

– Normal;

– CTC (Clear Timer on Compare).

Режим Normal є найпростішим режимом роботи T0. В ньому лічильний регістр TCNT0 функціонує як звичайний лічильник, що інкрементує своє значення по кожному імпульсу тактового сигналу clk_{T0}. Одночасно із переходом значення лічильного регістра від 0xFF до 0x00 (скидання лічильника) відбувається встановлення в “1” прапорця переривання TOV0. В результаті може бути згенеровано переривання по переповненню (якщо його дозволено). При співпадінні вмісту лічильного регістра TCNT0 і регістра порівняння OCR0 в режимі Normal, буде встановлюватися в “1” прапорець

переривання OCF0. В результаті може бути згенеровано переривання по співпадінню (якщо воно було дозволено).

У режимі CTC лічильний регістр TCNT0 також функціонує як звичайний лічильник, що додає. Однак лічба здійснюється від 0x00 не до 0xFF, а до значення, записаного в регістрі порівняння OCR0. При досягненні лічильним регістром цього значення, встановлюється прапорець OCF0. В результаті може бути згенеровано переривання по співпадінню (якщо воно було дозволено).

Оскільки початковим станом лічильного регістра TCNT0 є 0, то в регістр OCR0 слід записувати число, на 1 менше, ніж необхідний модуль лічби (наприклад, якщо необхідно лічити по модулю 15, то в OCR0 має бути записано 14).

2.4 Таймер/лічильник T1

Структурну схему таймера/лічильника T1 показано на рисунку 2.7.

Для забезпечення дозволу/заборони переривань від таймера/лічильника T1 використовуються окремі біти регістра TIMSK, формат якого показано нижче, на рисунку 2.8.

Встановлення в логічну “1” біту TOIE1 дозволяє переривання по переповненню T1, а біту OCIE1A – по співпадінню A.

Формат регістра TIFR, що використовується для індикації моменту надходження переривань від таймера/лічильника T1, показано на рисунку 2.9.

Прапорець TOV1 відображає переповнення таймера/лічильника T1, а OCF1A співпадіння A.

Як видно із рисунка 2.7, до складу таймера/лічильника T1 входить 4 основних регістра вводу/виводу (розрядністю як 8, так і 16 біт):

– TCCR1A, TCCR1B – регістри управління (кожен із них 8-розрядний);

– TCNT1 – лічильний регістр (16-розрядний);

– OCR1A – регістр порівняння (16-розрядний).

Необхідно пам’ятати, що кожен із 16-розрядних регістрів фізично розділений на два 8-розрядних регістра. Назви цих 8-розрядних регістрів утворюються шляхом додавання букви “H” (High, старший) і “L” (Low, молодший) до назви 16-розрядного регістра. Наприклад, 16-розрядний лічильний регістр TCNT1 фізично розміщується у парі 8-розрядних регістрів TCNT1H:TCNT1L.

Лічильний регістр TCNT1 входить до складу модуля реверсивного лічильника. Залежно від режиму роботи, вміст лічильного регістра може скидатися в 0, інкрементуватися або декрементуватися по кожному імпульсу тактового сигналу clk_{T1} (який надходить із схеми вибору тактового сигналу). Регістр TCNT1 в будь-який момент часу доступний як для читання, так і для запису. Після подачі живлення або скидання цей регістр містить нульове значення. При досягненні таймером/лічильником максимального значення (0xFFFF) встановлюється прапорець TOV1 в регістрі TIFR. Для того щоб при

цьому згенерувати переривання, повинен бути встановлений в “1” біт TOIE1 в регістрі TIMSK, а також прапорець “I” в регістрі SREG.

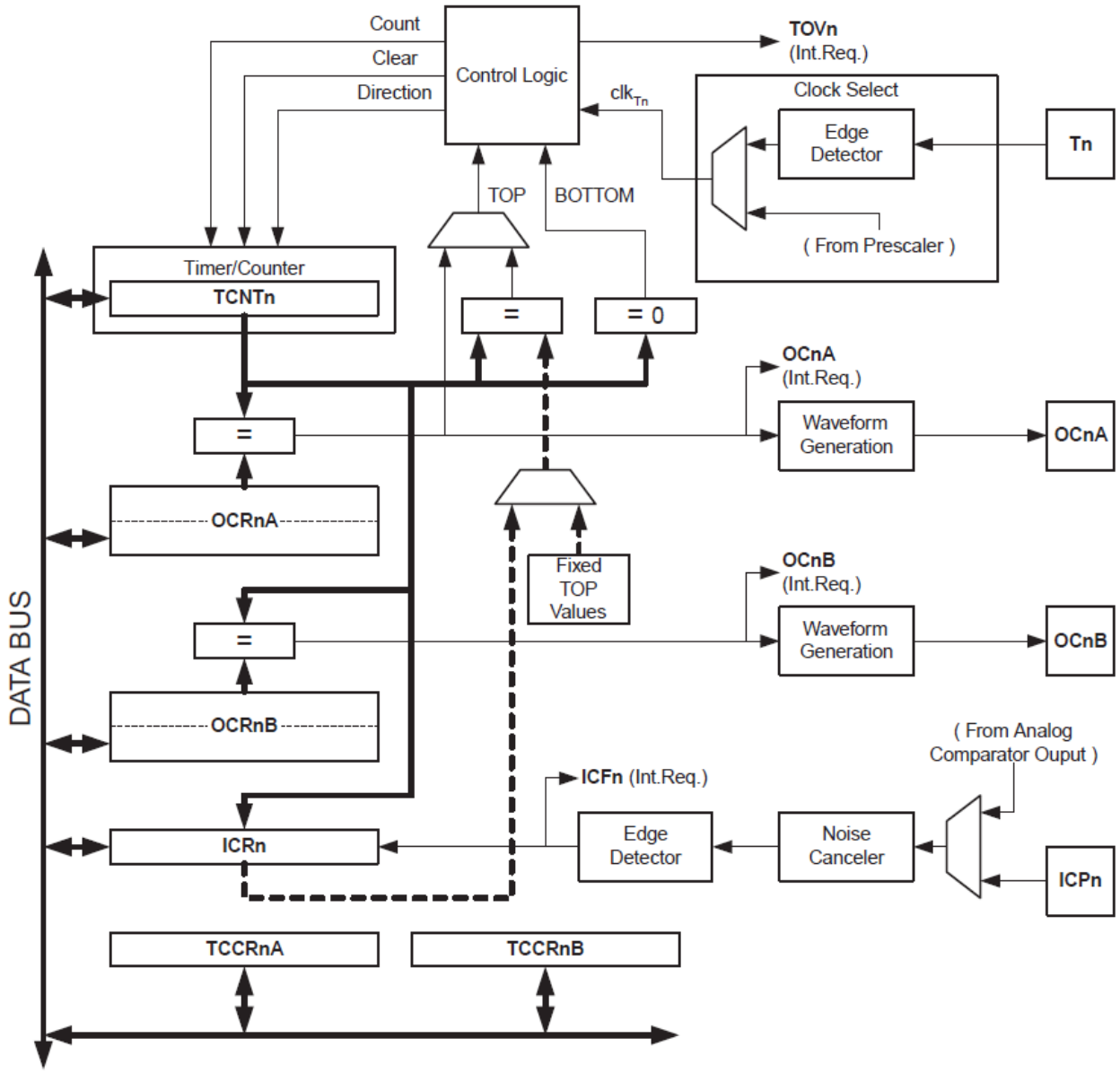

Рисунок 2.7 – Структурна схема таймера/лічильника T1

Bit	7	6	5	4	3	2	1	0
	TOIE1	OCIE1A	OCIE1B	OCIE2	TICIE1	TOIE2	TOIE0	OCIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.8 – Формат регістра TIMSK в МК ATMega8515 для T1

Bit	7	6	5	4	3	2	1	0
	TOV1	OCF1A	OC1FB	–	ICF1	–	TOV0	OCF0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.9 – Формат регістра TIFR в МК ATMega8515 для T1

Регістр порівняння OCR1A входить до складу блоку порівняння таймера/лічильника T1. Під час роботи T1 безперервно (в кожному такті) проводиться порівняння вмісту регістрів OCR1A і TCNT1. У разі співпадіння їх значень, встановлюється прапорець OCF1A в регістрі TIFR. Для того щоб при цьому згенерувати переривання, повинен бути встановлений в “1” біт OCIE1A в регістрі TIMSK, а також прапорець “I” в регістрі SREG.

Слід пам’ятати, що будь-яка операція запису в лічильний регістр TCNT1 заблокує роботу блоку порівняння і формування сигналу про співпадіння на 1 період тактового сигналу. У зв’язку з цим, рекомендується перед зміною вмісту регістра OCR1A здійснювати зупинку таймера/лічильника і скидання його лічильного регістра TCNT1, потім таймер можна запустити знову.

Для управління роботою таймера/лічильника T1 призначено регістри TCCR1A і TCCR1B, формати яких показано на рисунках 2.10 і 2.11 відповідно.

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.10 – Формат регістра TCCR1A в МК ATMega8515

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Рисунок 2.11 – Формат регістра TCCR1B в МК ATMega8515

Розглянемо призначення тільки основних бітів цих регістрів.

Біти WGM13–WGM10 відповідають за вибір режиму роботи таймера/лічильника T1 відповідно до правил, показаних нижче, в таблиці 2.3.

Біти CS12–CS10 відповідають за вибір джерела тактового сигналу для T1 відповідно до правил вибору, показаних в таблиці 2.4.

Слід також пам’ятати, що запис нового значення у біти CS12–CS10 не тільки вибирає можливе джерело тактового сигналу, але і здійснює пуск/зупинку таймера/лічильника T1.

До режимів роботи таймера/лічильника T1, що використовуються найбільш часто, слід віднести режим Normal і CTC.

Режим Normal є найпростішим режимом роботи таймера/лічильника T1. В ньому лічильний регістр TCNT1 функціонує як звичайний лічильник, що інкрементує своє значення по кожному імпульсу тактового сигналу clk_{T1}. Одночасно із переходом значення лічильного регістра від 0xFFFF до 0x0000 (скидання лічильника) відбувається встановлення в “1” прапорця переривання TOV1. В результаті може бути згенеровано переривання по переповненню (якщо воно дозволено). При співпадінні вмісту лічильного

регістра TCNT1 і регістра порівняння OCR1A в режимі Normal, буде встановлюватися в “1” прапорець переривання OCF1A. В результаті може бути згенеровано переривання по співпадінню А (якщо воно дозволено).

Таблиця 2.3 – Налаштування режимів роботи таймера/лічильника Т1

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	—		
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Таблиця 2.4 – Вибір тактового сигналу таймера/лічильника Т1

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge

У режимі CTC лічильний регістр TCNT1 також функціонує як звичайний лічильник, що додає. Однак лічба здійснюється від 0x0000 не до 0xFFFF, а до значення, записаного в регістрі порівняння OCR1A. При

досягненні лічильним регістром цього значення, встановлюється прапорець OCF1A. В результаті може бути згенеровано переривання по співпадінню А (якщо воно було дозволено).

Оскільки початковим станом лічильного регістра TCNT1 є 0, то в регістр OCR1A слід записувати число, на одиницю менше, ніж необхідний модуль лічби (наприклад, якщо нам необхідно лічити по модулю 1024, то в OCR1A має бути записано 1023).

2.5 Особливості оперування вмістом 16-розрядних регістрів

Кожен 16-розрядний регістр таймера/лічильника Т1 фізично розміщується у двох 8-розрядних регістрах. Відповідно, при зверненні до них потрібно виконати по 2 операції читання або запису. Для того щоб запис або читання обох байтів 16-розрядного регістра відбувалися одночасно, у складі кожного таймера/лічильника є спеціальний 8-розрядний регістр TEMP (використовується тільки ядром МК і програмно недоступний користувачу), призначений для зберігання значення старшого байту.

При виконанні циклу запису в 16-разрядний регістр, першим треба завантажувати старший байт значення, який поміщається в TEMP. При подальшому записі молодшого байта, він об’єднується із вмістом регістра TEMP, і обидва байти одночасно (в одному такті) записуються в 16-розрядний регістр. Якщо потрібно перезаписати кілька 16-розрядних регістрів таймера/лічильника, а старші байти всіх значень, що записуються, однакові, завантаження старшого байта досить виконати тільки 1 раз.

При виконанні циклу читання 16-розрядного регістру, першим повинен бути прочитаний молодший байт. При цьому, вміст старшого байта поміщається в TEMP. При подальшому читанні старшого байта повертається значення, збережене в регістрі TEMP.

Слід зауважити, що регістри порівняння OCR1A і OCR1B є винятковими, оскільки при читанні їх вмісту TEMP НЕ ЗАДІЮЄТЬСЯ!

Схему розміщення регістрів таймера/лічильника Т1 (включаючи регістр TEMP) показано на рисунку 2.12.

Слід звернути увагу, що при написанні програм, які включають роботу із таймерами/лічильниками мовою С, компілятор сам організовує правильний порядок звернення до кожного із 16-розрядних регістрів таймерів/лічильників в залежності від типу операції. Тобто, в програмах мовою С можна безпосередньо присвоювати значення 16-розрядному регістру, наприклад, TCNT1 = 0x707F.

При виконанні циклу звернення до 16-розрядного регістра таймера/лічильника, обробка переривань повинна бути заборонена! Інакше, якщо переривання відбудеться між двома командами звернення до 16-розрядного регістра, а в підпрограмі обробки цього переривання теж буде вироблено звернення до будь-якого із 16-розрядних регістрів того ж таймера/лічильника, вміст регістра TEMP буде змінено назавжди. Як наслідок, результат звернення до 16-розрядного регістра в основній програмі

буде невірним. Дане зауваження стосується програм, написаних як на асемблері, так і мовою С.

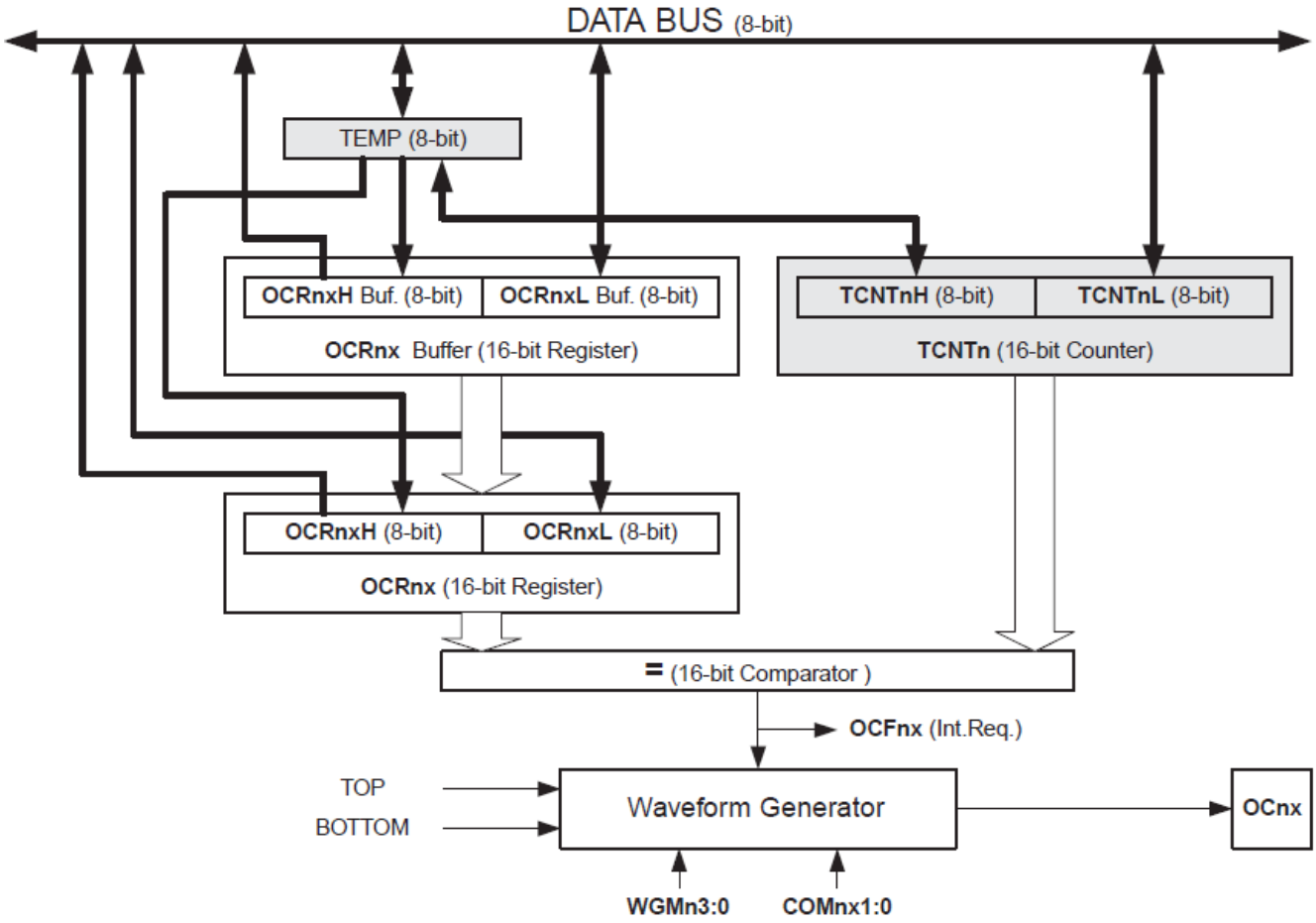

Рисунок 2.12 – Схема розміщення регістрів таймера/лічильника Т1 в МК ATMega8515

2.6 Приклади програм для МК AVR по роботі із таймерами/лічильниками

Нижче наведено приклад програми мовою асемблер, яка здійснює обробку переривання по співпадінню А таймера/лічильника Т1. Спрацювання переривання відбувається 1 раз на секунду. При цьому відбувається інкремент значення змінної, стан якої відображається на напівпровідниковому світлодіодному індикаторі.

```
;***** Програма до лабораторної роботи №8 (стенд EV8031/AVR) *****
;*** Працюємо з таймером/лічильником Т1 і світлодіодною лінійкою ***
;*** Виводимо на лінійку стан двійкового лічильника, що додає ***
;***  Збільшення стану лічильника відбувається із кожним перериванням від таймера/лічильника Т1 ***
;*** Т1 працює в режимі СТС, переривання генерується через проміжок часу, чітко рівний 1 с ***

;Підключення файлу, що містить опис регістрів і адрес для ATmega8515
.include "m8515def.inc"

;*** Призначення символічних імен регістрів ***
```

```
.def temp = r16   ;регістр тимчасового зберігання
.def diode = r19  ;регістр, який зберігає стан світлодіодів

;*** 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; Reset Handler (вектор переривання по скиданню)
      reti; rjmp EXT_INT0; IRQ0 Handler
      reti; rjmp EXT_INT1; IRQ1 Handler
      reti; rjmp TIM1_CAPT; Timer1 Capture Handler
      rjmp TIM1_COMPA; Timer1 Compare A Handler
;тобто,   адреса   переходу   на   обробник   переривання   по   співпадінню   А
таймера/лічильника Т1
      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)  ;ініціалізація вказівника стека SP
      out SPL, temp
      ldi temp, high (RAMEND)
      out SPH, temp           ;встановлення SP на останню адресу SRAM
      sbi ACSR, 7             ;відключення живлення аналогового компаратора
      ldi temp, 0b10000000    ;дозволяємо роботу із зовнішньою пам'яттю
      out MCUCR, temp

      clr diode   ;задаємо   початкове   значення   для   відображення   на   лінійці
світлодіодів (всі нулі)

;Встановлюємо вказівник Z на адресу лінійки світлодіодів
      ldi ZL, low (led_line)
      ldi ZH, high (led_line)

;*** Налаштування переривань ***

      ldi temp, (1 << OCIE1A) ;записуємо маску в регістр temp
      out TIMSK, temp         ;записуємо в регістр TIMSK (дозволяємо переривання
                              ;за збігом А таймера/лічильника Т1)

;*** Налаштування та пуск таймера/лічильника Т1 ***

;Завантажуємо   в   пару   регістрів   OCR1AH:   OCR1AL   число   0x707F   (28799d),   тобто
задаємо
;значення, при досягненні якого таймером/лічильником Т1 буде
```

```
;згенеровано переривання по співпадінню А
      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                                ;(скидання предделителя)

;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 !!!

;*** Дозволяємо роботу замаскованих переривань ***

      sei   ;встановлюємо прапорець глобального дозволу переривань
            ;працюватиме тільки переривання TIM1_COMPA),
            ;тобто переривання по співпадінню А таймера/лічильника T1

;*** Переходимо в нескінченний цикл, де очікуємо переривання від таймера
;(Раз на секунду) і змінюємо стан світлодіодів ***

Infinite_loop:; нескінченний цикл

      nop
      st Z, diode; виводимо значення лічильника на світлодіодну лінійку
      nop
      rjmp Infinite_loop; назад до початку нескінченного циклу

;*** Підпрограма обробки переривання по співпадінню А таймера/лічильника T1
TIM1_COMPA:       ;обробник зовнішнього переривання TIM1_COMPA (при вході I = 0)

      inc diode   ;збільшуємо значення лічильника, яке виводиться на світлодіоди

      reti  ;повернення із переривання
            ;при цьому автоматично встановлюється прапорець I
            ;з метою подальшого дозволу переривань

;***** Кінець програми *****
.EXIT
```

Аналогічну програму, написану мовою С, представлено нижче.

```
/*** Програма до лабораторної роботи №8 (стенд EV8031 / AVR) *****
**** Працюємо із таймером/лічильником T1 і світлодіодною лінійкою ***
**** Виводимо на лінійку стан двійкового лічильника, що додає ***
****  Збільшення стану лічильника відбувається із кожним перериванням від
таймера/лічильника T1 ***
```

```
**** T1 працює в режимі CTC, переривання генерується через проміжок часу, чітко рівний 1 с ***/

#define F_CPU 7372800L //задаємо частоту кварцу (7,3728 МГц)
#include <avr/io.h>
#include <avr/iom8515.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define led_line 0xA006 //визначаємо адресу лінійки світлодіодів в стенді

//Оголошуємо змінну, що відповідає за стан лінійки світлодіодів
volatile unsigned char diode = 0x00;

//Переривання по співпадінню А таймера/лічильника Т1
ISR (TIMER1_COMPA_vect) {
      diode ++; //збільшуємо значення лічильника, яке виводиться на світлодіоди
}

int main (void) {

      //встановлюємо вказівник на адресу лінійки світлодіодів
      volatile unsigned char* p = (unsigned char*) led_line;

      //початкова ініціалізація контролера
      ACSR = 1 << ACD; //відключення живлення аналогового компаратора
      MCUCR = 1 << SRE; //дозволяємо роботу із зовнішньою пам'яттю
      TIMSK = 1 << OCIE1A; //дозволяємо переривання по співпадінню А
таймера/лічильника Т1
      OCR1A = 0x707F;  //завантажуємо в регістрову пару OCR1AH:OCR1AL число
0x707F
                       //або 28799, тобто задаємо значення, при досягненні
якого
                       //таймер/лічильник Т1 згенерує переривання по
співпадінню А
      SFIOR = 1 << PSR10;     //скидаємо переддільник,
                              //фактично в цей момент відбувається початок
відліку
      TCCR1A = 0x00;
      TCCR1B = 1 << WGM12 | 1 << CS12;
            //завантажуємо в регістрову пару TCCR1A: TCCR1B число 0x000C,
            //тобто задаємо режим роботи CTC і, власне, запускаємо
            //таймер-лічильник Т1, переддільник встановлено на 256
            //в цьому місці, фактично, відбувається пуск таймера, хоча
переддільник вже працює

/* Оскільки переддільник встановлено на 256, то безпосередньо на лічильний вхід
модуль таймера/лічильника Т1
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) = 28799 або
0x707F (707Fh) !!!
Завжди слід записувати N-1, бо лічильник починає рахунок з нульового стану !!!*/

      sei();      //встановлюємо прапорець глобального дозволу переривань

      while (1) { //нескінченний цикл
            *p = diode; //виводимо значення змінної-лічильника на лінійку
світлодіодів

      }
      return 0;}
```

3 МЕТА І ПОРЯДОК ВИКОНАННЯ РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

3.1 Мета розрахунково-графічної роботи

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

3.2 Порядок виконання роботи

3.2.1 Запустити IDE AVR Studio 4.

3.2.2 Створити нові проекти в IDE AVR Studio 4.

3.2.3 Реалізувати програми мовами С і асемблер із урахуванням варіанту завдання згідно таблиці 3.1. Номер варіанту співпадає із порядковим номером студента у списку академічної групи.

Таблиця 3.1 – Варіанти завдань

№ варіанту	Опис завдання
1	При кожному повороті енкодера за годинниковою стрілкою на 3 тіки, значення 8-розрядного двійкового реверсивного лічильника інкрементується, а при повороті енкодера проти годинникової стрілки на 5 тіків – декрементується. При натисканні енкодера як кнопки, лічильник приймає значення 240. Початкове значення лічильника – 0. Стани лічильника відображати на напівпровідниковому світлодіодному індикаторі.
2	При кожному повороті енкодера за годинниковою стрілкою на 4 тіки, значення 8-розрядного реверсивного лічильника в коді Грея збільшується на 1, а при повороті енкодера проти годинникової стрілки на 2 тіки – зменшується на 1. При натисканні енкодера як кнопки, лічильник приймає значення коду Грея, що відповідає числу 240. Початкове значення лічильника – 0. Стани лічильника відображати на напівпровідниковому світлодіодному індикаторі.
3	При кожному повороті енкодера за годинниковою стрілкою на 5 тіків відбувається переміщення єдиного запаленого світлодіода на індикаторі в напрямку зліва направо на 1 розряд, а при повороті енкодера проти годинникової стрілки на 5 тіків – справа наліво на 1 розряд. Якщо світлодіод перетинає будь-який із країв індикатора, то він з’являється із протилежного краю. При натисканні енкодера як кнопки, запалений діод переміщується в крайнє ліве положення.

Продовження таблиці 3.1

№ варіанту	Опис завдання
4	При кожному повороті енкодера за годинниковою стрілкою на 7 тіків відбувається збільшення кількості запалених світлодіодів на світлодіодному індикаторі в напрямку справа наліво на 1, а при повороті енкодера проти годинникової стрілки на 7 тіків – зменшується на 1 (гасне крайній лівий). При перевищенні максимальної (всі горять) або мінімальної (всі погашені) кількості запалених світлодіодів, індикатор повертається в початковий стан (всі діоди погашено). При натисканні енкодера як кнопки, всі світлодіоди запалюються.
5	При кожному повороті енкодера за годинниковою стрілкою на 2 тіки здійснюється інкремент 2-розрядного реверсивного шістнадцяткового лічильника (із простором станів 00 → FF), стани якого відображаються на лівій парі знакомісць статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 2 тіки – декремент цього лічильника. При натисканні енкодера як кнопки, лічильник повертається в стан 00.
6	При кожному повороті енкодера за годинниковою стрілкою на 8 тіків здійснюється інкремент першого 2-розрядного шістнадцяткового лічильника (із простором станів 00 → FF), стани якого відображаються на лівій парі знакомісць статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 8 тіків – інкремент другого 2-розрядного шістнадцяткового лічильника (із простором станів 00 → FF), стани якого відображаються на правій парі знакомісць статичного семисегментного індикатора. При натисканні енкодера як кнопки, обидва лічильника повертаються в стан – 00.
7	При кожному повороті енкодера за годинниковою стрілкою на 7 тіків здійснюється інкремент 2-розрядного реверсивного десяткового лічильника (із простором станів 00 → 99), стани якого відображаються на правій парі знакомісць статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 7 тіків – декремент цього лічильника. При натисканні енкодера як кнопки, лічильник повертається в стан 00.
8	При кожному повороті енкодера за годинниковою стрілкою на 2 тіки здійснюється інкремент першого 1-розрядного 16-го лічильника (простір станів 0 → F), стани якого відображаються на крайньому лівому знакомісці статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 2 тіки – інкремент другого 1-розрядного 16-го лічильника (простір станів 0 → F), стани якого відображаються на крайньому правому знакомісці цього індикатора. При натисканні енкодера як кнопки, лічильники повертаються в стани 0 та F відповідно.

Продовження таблиці 3.1

№ варіанту	Опис завдання
9	При кожному повороті енкодера за годинниковою стрілкою на 5 тіків здійснюється вивід наступної великої літери англійського алфавіту на 5-му знакомісці першого рядка РКІ. При повороті енкодера проти годинникової стрілки на 5 тіків – вивід попередньої великої літери. При досягненні останньої або першої літери вивід повторюється заново. При натисканні енкодера як кнопки, відбувається очищення РКІ.
10	При кожному повороті енкодера за годинниковою стрілкою на 7 тіків здійснюється вивід наступної десяткової цифри на 11-му знакомісці першого рядка РКІ. При повороті енкодера проти годинникової стрілки на 7 тіків – вивід попередньої цифри. При досягненні останньої або першої цифри вивід повторюється заново. При натисканні енкодера як кнопки, відбувається очищення РКІ.
11	При кожному повороті енкодера за годинниковою стрілкою на 3 тіки здійснюється переміщення курсору в межах видимої області першого рядка РКІ в напрямку зліва направо на одне знакомісце. При повороті енкодера проти годинникової стрілки на 3 тіки – курсор зміщується справа наліво на одне знакомісце. Курсор не повинен виходити за межі видимої області і є завжди видимим. При натисканні енкодера як кнопки, відбувається повернення курсору на початок першого рядка.
12	При кожному повороті енкодера за годинниковою стрілкою на 10 тіків здійснюється зсув прізвища студента (попередньо виведене у перший рядок РКІ) в напрямку зліва направо на одне знакомісце. При повороті енкодера проти годинникової стрілки на 10 тіків – рядок зміщується справа наліво на одне знакомісце. При натисканні енкодера як кнопки, відбувається повернення до початкового стану (виводиться прізвище у перший рядок РКІ).
13	Попередньо створити і занести в графічне ОЗП РКІ відображення будь-яких чотирьох довільних символів користувача. Перший із символів одразу має відображатися у відповідній позиції РКІ. При кожному повороті енкодера за годинниковою стрілкою на 3 тіки здійснюється вивід наступного символу користувача на 5-му знакомісці першого рядка РКІ. При повороті енкодера проти годинникової стрілки на 3 тіки – вивід попереднього символа. При досягненні останнього або першого символа вивід припиняється. При натисканні енкодера як кнопки, відбувається повернення до початкового стану.

Продовження таблиці 3.1

№ варіанту	Опис завдання
14	Попередньо створити і занести в графічне ОЗП РКІ відображення будь-якого довільного символу користувача. Він одразу має відображатися на самому початку рядка 1 РКІ. При кожному повороті енкодера за годинниковою стрілкою на 2 тіки здійснюється вивід цього користувацького символу у наступному справа знакомісці першого рядка РКІ. При повороті енкодера проти годинникової стрілки на 2 тіки – символ у поточній позиції стирається і курсор переходить на попередній символ. При досягненні останнього або першого знакомісця рядка вивід припиняється. При натисканні енкодера як кнопки, відбувається повернення до початкового стану.
15	При кожному повороті енкодера за годинниковою стрілкою на 3 тіки здійснюється інкремент першого 1-розрядного десяткового лічильника (простір станів 0 → 9), стани якого відображаються на знакомісці 4 рядка 1 РКІ, а при повороті проти годинникової стрілки на 3 тіки – інкремент другого 1-розрядного десяткового лічильника (простір станів 0 → 9), стани якого відображаються на знакомісці 11 рядка 2 РКІ. При натисканні енкодера як кнопки, обидва лічильника повертаються в стан – 0.
16	При кожному повороті енкодера за годинниковою стрілкою на 2 тіки здійснюється переміщення літери “A” в напрямку зліва направо на статичному семисегментному індикаторі на одне знакомісце, а при повороті проти годинникової стрілки на 3 тіки – в напрямку справа наліво на одне знакомісце. При натисканні енкодера як кнопки, літера “A” замінюється на “E”, при повторному – знову повертається “A”.
17	При кожному повороті енкодера за годинниковою стрілкою на 5 тіків, значення 8-розрядного двійкового реверсивного лічильника інкрементується, а при повороті енкодера проти годинникової стрілки на 3 тіки – декрементується. При натисканні енкодера як кнопки, лічильник приймає значення 15. Початковим значенням лічильника є 255. Стани лічильника відображати на напівпровідниковому світлодіодному індикаторі.
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 енкодера як кнопки, лічильник приймає значення кода Грея, що відповідає числу 15. Початковим його значенням є код Грея числа 255. Стани лічильника відображати на напівпровідниковому світлодіодному індикаторі.

Продовження таблиці 3.1

№ варіанту	Опис завдання
19	При кожному повороті енкодера за годинниковою стрілкою на 7 тіків відбувається переміщення єдиного запаленого світлодіода на індикаторі в напрямку справа наліво на 1 розряд, а при повороті енкодера проти годинникової стрілки на 7 тіків – зліва направо на 1 розряд. Якщо світлодіод перетинає один із країв індикатора, то з'являється на протилежному. При натисканні енкодера як кнопки, запалений діод стає в крайнє праве положення.
20	При кожному повороті енкодера за годинниковою стрілкою на 5 тіків відбувається збільшення кількості запалених світлодіодів на світлодіодному індикаторі в напрямку зліва направо на 1, а при повороті енкодера проти годинникової стрілки на 7 тіків – зменшується на 1 (гасне крайній правий). При перевищенні максимальної (всі горять) або мінімальної (всі погашені) кількості запалених світлодіодів, індикатор повертається в початковий стан (всі горять). При натисканні енкодера як кнопки, всі світлодіоди індикатора згасають.
21	При кожному повороті енкодера за годинниковою стрілкою на 4 тіки здійснюється декремент 2-розрядного реверсивного шістнадцяткового лічильника (простір станів FF → 00), стани якого відображаються на правій парі знакомісць статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 4 тіки – інкремент цього лічильника. При натисканні енкодера як кнопки, лічильник повертається в стан FF.
22	При кожному повороті енкодера за годинниковою стрілкою на 6 тіків здійснюється декремент першого 2-розрядного шістнадцяткового лічильника (простір станів FF → 00), стани якого відображаються на лівій парі знакомісць статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 6 тіків – декремент другого 2-розрядного шістнадцяткового лічильника (простір станів FF → 00), стани якого відображаються на правій парі знакомісць. При натисканні енкодера як кнопки, обидва лічильника повертаються в стан FF.
23	При кожному повороті енкодера за годинниковою стрілкою на 3 тіки здійснюється декремент 2-розрядного реверсивного десяткового лічильника (простір станів 99 → 00), стани якого відображаються на лівій парі знакомісць статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 3 тіки – інкремент цього лічильника. При натисканні енкодера як кнопки, лічильник повертається в стан 99.

Продовження таблиці 3.1

№ варіанту	Опис завдання
24	При кожному повороті енкодера за годинниковою стрілкою на 4 тіки здійснюється декремент першого 1-розрядного шістнадцяткового лічильника (простір станів F → 0), стани якого відображаються на другому зліва знакомісці статичного семисегментного індикатора, а при повороті проти годинникової стрілки на 4 тіки – декремент другого 1-розрядного шістнадцяткового лічильника (простір станів F → 0), стани якого відображаються на другому справа знакомісці. При натисканні енкодера як кнопки, обидва лічильника повертаються в стани F та 0 відповідно.
25	При кожному повороті енкодера за годинниковою стрілкою на 2 тіки здійснюється вивод наступної малої літери англійського алфавіту на 9-му знакомісці другого рядка РКІ. При повороті енкодера проти годинникової стрілки на 2 тіки – вивід попередньої малої літери. При досягненні останньої або першої літери вивод повторюється заново. При натисканні енкодера як кнопки, відбувається очищення РКІ.
26	При кожному повороті енкодера за годинниковою стрілкою на 5 тіків здійснюється вивод наступної десяткової цифри на 3-му знакомісці другого рядка РКІ. При повороті енкодера проти годинникової стрілки на 5 тіків – вивід попередньої цифри. При досягненні останньої або першої цифри вивід повторюється заново. При натисканні енкодера як кнопки, відбувається очищення РКІ.
27	При кожному повороті енкодера за годинниковою стрілкою на 4 тіки здійснюється переміщення курсору в межах видимої області другого рядка РКІ в напрямку справа наліво на одне знакомісце. При повороті енкодера проти годинникової стрілки на 4 тіки – курсор зміщується зліва направо на одне знакомісце. Курсор не повинен виходити за межі видимої області і є завжди видимим. При натисканні енкодера як кнопки, відбувається повернення курсору в кінець другого рядка.
28	При кожному повороті енкодера за годинниковою стрілкою на 6 тіків здійснюється зсув імені студента (попередньо виведене у другий рядок РКІ) в напрямку справа наліво на одне знакомісце. При повороті енкодера проти годинникової стрілки на 6 тіків – рядок зсувається зліва направо на одне знакомісце. При натисканні енкодера як кнопки, відбувається повернення до початкового стану (виводиться ім’я у другий рядок РКІ).

Продовження таблиці 3.1

№ варіанту	Опис завдання
29	Попередньо створити і занести в графічне ОЗП РКІ відображення будь-яких чотирьох довільних символів користувача. Останній із символів одразу має відображатися у відповідній позиції РКІ. При кожному повороті енкодера за годинниковою стрілкою на 7 тіків здійснюється вивід наступного користувацького символу на 14-му знакомісці другого рядка РКІ. При повороті енкодера проти годинникової стрілки на 7 тіків – вивід попереднього символа. При досягненні останнього або першого символа вивід припиняється. При натисканні енкодера як кнопки, відбувається повернення до початкового стану.
30	Попередньо створити і занести в графічне ОЗП РКІ відображення будь-якого довільного символу користувача. Він одразу має відображатися в самому кінці рядка 2 РКІ. При кожному повороті енкодера за годинниковою стрілкою на 4 тіки здійснюється вивід цього користувацького символу у наступному зліва знакомісці другого рядка РКІ. При повороті енкодера проти годинникової стрілки на 4 тіки – символ у поточній позиції стирається і курсор переходить на наступний символ. При досягненні останнього або першого знакомісця рядка вивод припиняється. При натисканні енкодера як кнопки, відбувається повернення до початкового стану.
31	При кожному повороті енкодера за годинниковою стрілкою на 4 тіки здійснюється декремент першого 1-розрядного десяткового лічильника (простір станів 9 → 0), стани якого відображаються на знакомісті 11 рядка 1 РКІ, а при повороті проти годинникової стрілки на 4 тіки – декремент другого 1-розрядного десяткового лічильника (простір станів 0 → 9), стани якого відображаються на знакомісті 4 рядка 2 РКІ. При натисканні енкодера як кнопки, обидва лічильника повертаються в стан 9.

Примітки.

1. У всіх варіантах РГР необхідно реалізувати опитування стану каналів інкрементального енкодера із частотою якомога ближчою до 1000 Гц.

2. Якщо номер варіанту є непарним числом, генерацію часових проміжків для опитування стану каналів інкрементального енкодера реалізувати за допомогою таймера/лічильника Т0, що працює в режимі СТС.

3. Якщо номер варіанту є парними числами, генерацію часових проміжків для опитування стану каналів інкрементального енкодера реалізувати за допомогою таймера/лічильника Т1, що працює в режимі СТС (генерація переривань по співпадінню А або В).

4. Використання іншогог незадіяного таймера/лічильника у всіх варіантах є опціональним.

5. Генерацію інших часових затримок виконувати програмно.

3.2.4 Здійснити компіляцію проектів.

3.2.5 За наявності повідомлень про помилки або попередження повернутися до попереднього пункту і внести необхідні виправлення. У разі некоректної роботи програми виконати її налагодження засобами меню Debug.

3.2.6 Перевірити підключення USB-кабелю програматора до однойменного гнізда системного блоку.

3.2.7 Завантажити виконувані файли проектів в мікроконтролер.

3.2.8 Візуально оцінити правильність роботи написаної програми.

4 ВИМОГИ ДО ЗМІСТУ І ОФОРМЛЕННЯ ЗВІТУ

4.1 Вимоги до змісту звіту

Звіт про виконання розрахунково-графічної роботи повинен містити:
– номер і назву роботи;
– мету роботи;
– короткі теоретичні відомості;
– порядок виконання роботи;
– результати виконання роботи у наступному вигляді:
  – тексти програм, розроблених відповідно до варіанта завдання, які містять всі необхідні коментарі і пояснення;
  – схему алгоритму управління, розробленого відповідно до варіанту завдання;
  – особливості функціонування IDE AVR Studio 4;
– висновки.

4.2 Вимоги до оформлення звіту

Викладені нижче вимоги щодо оформлення звіту про виконання розрахунково-графічної роботи є обов'язковими до виконання.

1. Для оформлення звіту використовуються стандартні листи формату А4 (210х297 мм), що мають наступні поля: ліве – не менше 25 мм; верхнє – не менше 15 мм; нижнє – не менше 15 мм; праве – не менше 10 мм.

2. Текст звіту (включаючи заголовки, назви розділів пунктів і підпунктів, підписи рисунків і таблиць) оформлюється шрифтом Times New Roman, 14 pt, одинарний міжрядковий інтервал без додаткових виступів до або після. Для представлення у звіті коду програми (будь-якою мовою), коду скрипта, тощо, використовується шрифт Courier New, 10 pt, всі інші вимоги – аналогічні.

3. Порядковий номер розрахунково-графічної роботи, її тема, назви всіх розділів верхнього рівня, підписи рисунків, тіла рисунків і таблиць вирівнюються по центру, без абзацного відступу. Виключення складають лише підписи таблиць, оскільки вони вирівнюються по ширині із абзацного відступу.

4. Підпис рисунка і його тіло є нерозривними, тому категорично забороняється розривати їх (переносити на різні сторінки) або розділяти будь-якими пропусками.

5. При переносі таблиці на іншу сторінку вгорі обов'язково ставиться напис "Продовження таблиці Х.У", а також дублюється її шапка.

6. Нумерація таблиць, рисунків і формул виконується наскрізно в межах кожного розділу звіту і складається із двох цифр: перша – номер розділу, до якого відноситься елемент, друга – порядковий номер елемента в межах розділу. Наприклад, напис "Таблиця 2.4" означає, що ця таблиця відноситься до розділу 2 і є четвертою по порядку слідування в межах розділу.

7. При нумерації пунктів і підпунктів розділу категорично не рекомендується використовувати рівень вкладеності глибший, ніж 3.

8. Після тексту перед назвою розділу/підрозділу, тілом рисунку або підписом таблиці, а також після назви розділу/підрозділу, підпису рисунка або тіла таблиці перед наступним текстом робиться пропуск в 1 рядок. Категорично забороняється робити пропуски рядків у суцільному тексті.

9. Рисунки і таблиці, що мають різний порядковий номер, не можуть мати ідентичні пояснюючі підписи. Забороняється залишати тільки нумерацію без пояснюючих підписів.

5 КОНТРОЛЬНІ ПИТАННЯ

5.1 Поясніть фізичні принципи роботи перетворювача кутових переміщень.

5.2 Дайте коротку характеристику кожному з існуючих типів енкодерів.

5.3 Поясніть в чому полягає перевага застосування коду Грея перед двійковим кодом при кодуванні станів енкодерів.

5.4 Поясніть особливості формування коду Грея.

5.5 Які операції булевої логіки необхідно використовувати для побудови перетворювача з двійкового коду в код Грея і зворотного перетворювача?

5.6 Поясніть, в чому полягають відмінності в принципах роботи абсолютних і інкрементальних енкодерів?

5.7 Наведіть і поясніть, які фактори необхідно враховувати при виборі частоти опитування станів інкрементального енкодера.

5.8 Поясніть як здійснюється взаємозв'язок між інкрементальним енкодером і мікроконтролером в навчально-відлагоджувальному стенді EV8031/AVR.

5.9 Поясніть, яким чином здійснюється опитування станів інкрементального енкодера за допомогою механізму переривань.

5.10 Вкажіть, в чому полягають переваги опитування станів інкрементального енкодера за допомогою механізму переривань в порівнянні з безпосереднім опитуванням виводів порту мікроконтролера.

5.11 Яким чином здійснювалося визначення напрямку обертання ручки інкрементального енкодера в даній розрахунково-графічної роботи?

5.12 Яке явище покладено в основу роботи магніторезистивного енкодера? Яким законом описується це явище?

5.13 Перелічіть, які основні параметри слід враховувати при виборі енкодерів?

5.14 Поясніть призначення переривань.

5.15 Охарактеризуйте основні режими роботи кожного з таймерів/лічильників, що входять до складу мікроконтролера ATMega8515.

5.16 Поясніть режим роботи таймера/лічильника T1 (або T0) по співпадінню (CTC).

5.17 Поясніть механізм взаємодії мікроконтролера ATmega8515 із зовнішньою пам'яттю, реалізований в навчально-відлагоджувальному стенді EV8031/AVR.

5.18 Поясніть механізм конфігурації переривань таймера/лічильника T1.

5.19 Вкажіть переваги і недоліки застосування переривань від таймера.

5.20 Вкажіть основні особливості, які необхідно враховувати при написанні програм, що містять обробку переривань від таймера.

5.21 Поясніть механізм обробки переривання (від моменту його виникнення і до моменту повернення в основну програму).

5.22 На прикладі мікроконтролера ATmega8515 розподіліть всі існуючі переривання (згідно таблиці векторів його переривань) на 2 групи: внутрішні і зовнішні.

5.23 На прикладі мікроконтролера ATmega8515 перерахуйте всі події, за яких може відбуватися генерація переривань від таймера/лічильника T1, вкажіть необхідні значення відповідних бітів конфігурації.

5.24 Наведіть таблицю пріоритетів переривань на прикладі мікроконтролера ATmega8515. Які існують механізми для переналаштування пріоритетів переривань?

5.25 Яким чином можливо організувати генерацію програмних (НЕ апаратних!) переривань в мікроконтролерах AVR?

5.26 Яким чином можливо організувати обробку вкладених переривань в мікроконтролерах AVR?

5.27 Поясніть режим роботи таймера/лічильника T1 (або T0) по переповненню (Normal).