Міністерство освіти і науки України

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра радіотехнічних та вбудованих систем

**ЗВІТ**

про виконання лабораторних робіт

з дисципліни: «Програмування периферійних пристроїв»

Варіант 8

**Виконав:** студент групи Кіт-211

Оленченко Ілля Романович

**Викладач:** кандидат технічних наук,

доцент кафедри РТВС

Красножон Олексій Васильович

Чернігів 2022

**ЗМІСТ**

Лабораторна робота №1 3

Лабораторна робота №2 8

Лабораторна робота №3 19

Лабораторна робота №4 32

Лабораторна робота №5 42

**ЛАБОРАТОРНА РОБОТА №1**

**Тема:** вивчення відлагоджувального стенду ev8031/avr, системи команд мікроконтролера atmega8515. напівпровідниковий світлодіодний індикатор.

**Мета роботи:** Ознайомитися із функціональними можливостями і внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню архітектуру і структуру мікроконтролера ATmega8515, його систему команд. Вивчити функціональні можливості IDE AVR Studio 4, навчитися створювати і компілювати, виконувати покрокове трасування програм, написаних мовами С або асемблер, програмувати вказаний мікроконтролер. Навчитися відображати інформацію за допомогою напівпровідникового світлодіодного індикатора.

**1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

* 1. **Загальні відомості**

Найпростішими приладами відображення інформації в цифрових пристроях є світлодіоди напівпровідникових світлодіодах використовується властивість р-n переходу випромінювати світло у видимій частині спектру при протіканні через нього прямого струму (Іпр = 5 – 20 мА, Uпр = 2 – 3 B). Схему реалізації індикатора на основі напівпровідникових світлодіодів представлено ​​вище на рисунку 1.1.

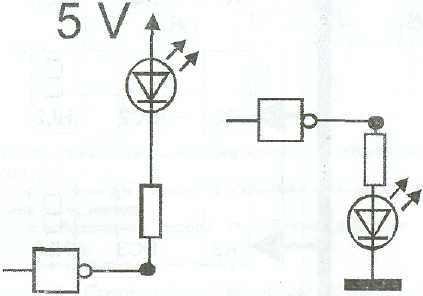


Рисунок 1.1 – Схема підключення діода, що випромінює світло

У навчально-відлагоджувальному стенді EV8031/AVR доступ до світлодіодного індикатора (лінійки світлодіодів HL1 – HL8) здійснюється як до комірки зовнішньої пам’яті, яку розташовано за адресою 0xA006 (A006h). Кожен із світлодіодів запалюється записом логічної “1” у відповідний розряд.

**1.2 Порядок виконання роботи**

1.2.1 Запустіть IDE AVR Studio 4.

1.2.2 Активуйте створення нового проекту за допомогою вибору пункту New Project з пункту Project головного меню AVR Studio 4.

1.2.3 Виберіть тип створюваного проекту: проект буде створюватися мовою асемблер (пункт Atmel AVR Assembler) або мовою С (пункт AVR GCC).

1.2.4 Нижче, в полі Location, вкажіть шлях, за яким буде розташовуватися проект. Для зручності подальших маніпуляцій із проектом, необхідно зберегти його в окремому каталозі (наполегливо рекомендується D:\AVR\_projects).

1.2.5 Вказати ім’я проекту (англійською мовою) в полі Project name.

1.2.6 Примусово (якщо це не зроблено самою AVR Studio 4) встановити прапорець Create initial file. Бажано також встановити прапорець Create folder, що дозволить створити окремий каталог (шлях до якого вказано у полі Location) для збереження всіх файлів проекту всередині нього. Після зазначених дій натиснути кнопку Next >>.

1.2.7 У вікні Debug platform вибрати пункт AVR Simulator. У вікні Device вибрати пункт ATmega8515. Вибір будь-яких пунктів, що відрізняються від наведених вище, у зазначених вікнах призведе до подальших помилок і неможливості виконання лабораторної роботи!

1.2.8 Натиснути кнопку Finish для завершення створення проекту.

1.2.9 У вікні, що з’явилося, написати програму мовою С або асемблер з урахуванням варіанту завдання, зазначеного у таблиці 1.1. Варіант №1.

**2 ХІД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ**

**2.1 Результати виконання завдання**

Згідно варіанту потрібно реалізувати алгоритм індикації, що зображений в таблиці 2.1. Тривалість індикації одного стану (Ti) – 2,25 с.

Таблиця 2.1 – алгоритм індикації

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Т1** | **Т2** | **Т3** | **Т4** | **Т5** | **Т6** | **Т7** | **Т8** | **Т9** | **Т10** | **Т11** | **Т12** | **Т13** | **Т14** | **Т15** |
| **VD1** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **VD2** | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **VD3** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **VD4** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| **VD5** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| **VD6** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **VD7** | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **VD8** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Примітка. Стану "1" відповідає запалення світлодіода | | | | | | | | | | | | | | | |

Для запалювання світлодіодів було визначено маску при запис одиниці у вказаний біт регістра викликає запалювання обраного світлодіода на рисунку 2.1 зображено фрагмент коду.

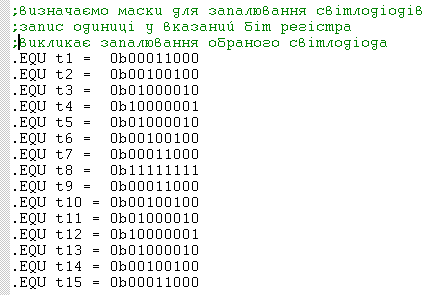


Рисунок 2.1 – Маска світлодіодів

Для запалювання світлодіодів був створений нескінчений цикл, на рисунку 2.2 зображено фрагмент коду.

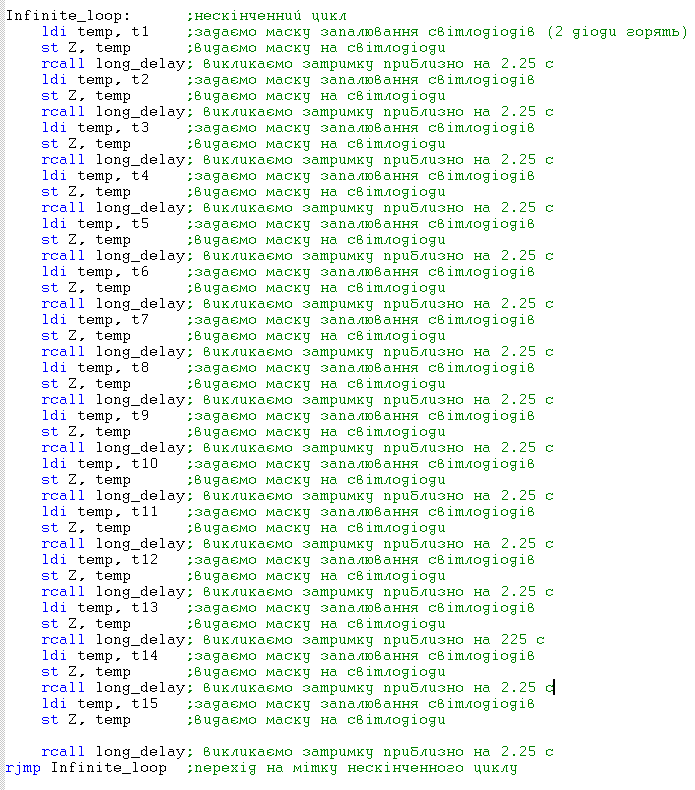


Рисунок 2.2 – Нескінчений цикл

Для затримки запалювання була використана підпрограма довгої затримки long\_delay з затримкою 2.25с. На рисунку 2.3 зображено фрагмент коду.



Рисунок 2.3 – Підпрограма довгої затримки long\_delay

**ВИСНОВКИ**

Ознайомилися із функціональними можливостями мікроконтролера ATmega8515, його систему команд. Ознайомилися з IDE AVR Studio 4, навчилися створювати і компілювати код, виконувати покрокове трасування програм, написаних мовами С або асемблер, програмувати вказаний мікроконтролер. Навчитися відображати інформацію за допомогою напівпровідникового світлодіодного індикатора.

**ЛАБОРАТОРНА РОБОТА №2**

**СТАТИЧНИЙ семисегментнИЙ СВІТЛОДІОДНИЙ ИНДИКАТОР**

**Мета роботи:** ознайомитися із функціональними можливостями і внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню організацію статичного семисегментного світлодіодного індикатора, навчитися відображати інформацію на ньому.

**1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

**1.1 Загальні відомості**

В якості пристроїв для відображення цифрової інформації найбільшого поширення набули семисегментні індикатори, в яких зображення символа складається із семи лінійних світлодіодних сегментів розташованих у вигляді цифри “8”.

На основі світлодіодів і семисегментних індикаторів будуються підсистеми відображення інформації. При побудові таких підсистем розрізняють два підходи: динамічну і статичну схеми організації індикації.

Сутність статичної індикації полягає в тому, що індикатори НL1 – HL4 неперервно і незалежно один від одного відображають символьну інформацію, отриману від єдиного джерела (дивись рисунок 1.1).

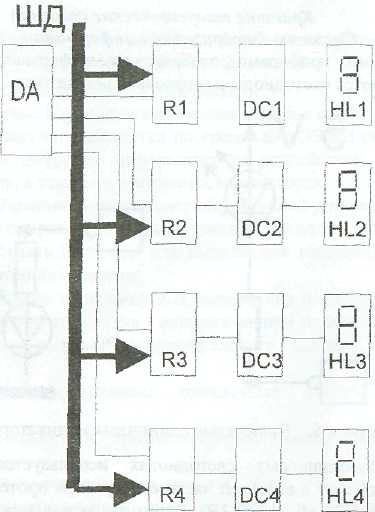


Рисунок 1.1 – Структурна схема організації статичної індикації

До основних структурних елементів схеми організації статичної індикації входять наступні:

1. DA – дешифратор адреси, необхідний для вибору і активації процесу запису даних до відповідного регістра (Ri).

2. R1 ... R4 – регістри, в яких тимчасово зберігається двійковий (або одразу семисегментний) код символу, що відображається на конкретному знакомісці.

3. DC1 ... DC4 – пристрої, що здійснюють перетворення вхідного двійкового коду в семисегментний.

4. HL1 ... HL4 – 4 знакомісця, які утворюются ceмисегментними свівтлодіодними індикаторами.

5. ШД – однонаправлена шина даних, по якій здійснюється їх передача для подальшого відображення.

При такій схемній організації кожен індикатор HLi підключається за допомогою власного перетворювача кодів DCi і регістра-засувки RGi до шини даних (ШД), а вибір конкретного регістра RGi здійснюється за допомогою дешифратора адреси DА. Апаратні витрати при такій організації складають n пар з'єднань регістрів і дешифраторів при n десяткових знакомісцях індикатора.

У навчально-відлагоджувальному стенді EV8031/AVR статична індикація реалізована за допомогою одного статичного семисегментного індикатора HG1, що містить 4 знакомісця. Звернення до них здійснюється, як до комірок зовнішньої пам'яті даних за адресами 0xА000 (ліва пара знакомісць) і 0xВ000 (права пара знакомісць). Старша і молодша тетради кожної з цих комірок зберігають двійкові коди символів, що виводяться на старше (ліве) і молодше (праве) знакомісця в кожній парі відповідно.

Керуванням процесами запалювання і гасіння точок (восьмий сегмент індикатора) і знакомісць статичного семисегментного індикатора здійснюється за допомогою запису певної комбінації бітів в комірку зовнішньої пам'яті, розташованої за адресою 0xA004 (A004h). Запис нулів в старшу тетраду цієї комірки призводить до гасіння точок на всіх знакомісцях (відповідно запис одиниць – до запалювання). Запис одиниць в молодшу тетраду комірки призводить до гасіння всіх семи сегментів на всіх знакомісцях (відповідно запис нулів – до запалювання).

**1.2 Порядок виконання роботи**

1.2.1 Запустіть IDE AVR Studio 4.

1.2.2 Активуйте створення нового проекту за допомогою вибору пункту New Project з пункту Project головного меню AVR Studio 4.

1.2.3 Виберіть тип створюваного проекту: проект буде створюватися мовою асемблер (пункт Atmel AVR Assembler) або С (пункт AVR GCC).

1.2.4 Нижче, в полі Location, вкажіть шлях по якому буде розташовуватися проект. Для зручності подальших маніпуляцій з проектом необхідно зберегти його в окремому каталозі (наполегливо рекомендується D:\AVR\_projects).

1.2.5 Вказати ім’я проекту (англійською мовою) в полі Project name.

1.2.6 Примусово (якщо це не зробила AVR Studio 4) встановити прапорець Create initial file. Бажано також встановити прапорець Create folder, що дозволить створити окремий каталог (шлях до якого вказано у полі Location) для збереження всіх файлів проекту всередині нього. Після зазначених дій натиснути кнопку Next >>.

1.2.7 У вікні Debug platform вибрати пункт AVR Simulator. У вікні Device вибрати пункт ATmega8515. Вибір будь-яких пунктів, що відрізняються від наведених вище, у зазначених вікнах призведе до подальших помилок і неможливості виконання лабораторної роботи!

1.2.8 Натиснути кнопку Finish для завершення створення проекту.

1.2.9 У вікні, що з’явилося, написати програму мовою С або асемблер з урахуванням варіанту завдання, зазначеного у таблиці 1.1.

1.2.10 Здійснити компіляцію проекту за допомогою вибору пункту Build з однойменного меню (або натиснути кнопку F7).

1.2.11 За наявності повідомлень про помилки або попередження повернутися до попереднього пункту і внести необхідні виправлення.

1.2.12 Обов’язково виконати покрокове відлагодження та трасування скомпільованої програми засобами меню Debug перед програмуванням мікроконтролера.

1.2.13 Перевірте наявність підключення USB-кабелю програматора до відповідного роз'єму системного блоку.

1.2.14 Для завантаження відкомпільованого проекту в мікроконтролер виберіть пункт меню Tools -> Program AVR -> Connect...

1.2.15 У вікні Platform, що з’явилося, виберіть пункт AVRISP mkII. У вікні Port виберіть пункт USB. Вибір будь-яких пунктів, що відрізняються від наведених вище, у зазначених вікнах призведе до подальших помилок і неможливості програмування мікроконтролера!

1.2.16 Натисніть на кнопку Connect ...

1.2.17 У вікні, що відкрилося, відразу перейдіть на вкладку Program. В полі Location вкажіть шлях до файлу з розширенням \*.hex. Він розташований в каталозі (або підкаталогах) проекту, а його ім'я співпадає із ім'ям проекту. Цей файл містить машинні коди, які будуть виконуватися мікроконтролером після завантаження в нього файлу \*.hex.

1.2.18 Натисніть кнопку Program. При цьому відбувається програмування (повне стирання і запис) Flash-пам'яті (пам'яті програм) мікроконтролера. Кількість гарантованих операцій стирання/запису цієї пам'яті достатньо мала, тому не слід натискати кнопку Program без потреби і бездумно перепрограмувати мікроконтролер!

1.2.19 Візуально оцініть правильність роботи алгоритму індикації для статичного семисегментного світлодіодного індикатора.

**2 ХІД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ**

**2.1 Результати виконання завдання**

Завдання для лабораторної роботи зображено в таблиці 1.1.

Таблиця 1.1 – Завдання

|  |  |
| --- | --- |
| № | Опис завдання |
| 8 | На правій парі знакомісць статичного семисегментного індикатора реалізувати відображення станів лічильника, що працює за алгоритмом: FF → FE → ... → 01 → 00 → 01 → ... → FE → FF (FF → 00 → FF). Затримка на відображення одного стану дорівнює 1,5 с і організовується програмно. |

Код програми зображено на в лістингу 1.1.

Лістинг 1.1 – Імена регістрів

;\*\*\*\*\* Програма до лабораторної роботи №2 (стенд EV8031/AVR) \*\*\*\*\*

;\*\*\* Працюємо зі статичним семисегментним індикатором \*\*\*

;\*\*\* Відображаємо на статичному індикаторі інформацію за наступним алгоритмом

;(Стану ХХХХ відповідає повне гасіння індикатора) 0000, ХХХХ, 1111, ХХХХ, 2222, ХХХХ, ... EEEE, ХХХХ, FFFF, ХХХХ.

;Затримка на відображення одного стану дорівнює 2 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 dig = r17 ;регістр зберігання виведених на правий індикатор чисел (в hex-форматі)

.def counter = r18 ;лічильник циклу в підпрограмі генерації затримки

.def counter1 = r20 ;лічильник кількості пройдених станів алгоритму

.def const = r19 ;регістр, який зберігає константу, що додається до маски

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

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

.def dig0 = r21 ;регістр зберігання виведених на левий індикатор чисел 00

.def counter2 = r23 ;лічильник кількості пройдених станів алгоритму

;\*\*\* Призначення констант \*\*\*

;адреса пари лівих знакомісць статичного семисегментного індикатора

.EQU stat\_7seg\_left = 0xA000

;адреса пари правих знакомісць статичного семисегментного індикатора

.EQU stat\_7seg\_right = 0xB000

;адреса регістра керування запаленням/гасінням точок/знакомісць статичного семисегментний індикатора

.EQU stat\_7seg\_control = 0xA004

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

.CSEG ;визначаємо початок сегмента коду

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

; \*\*\* Вектор 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) ;ініціалізація вказівника стеку SP

out SPL, temp

ldi temp, high (RAMEND)

out SPH, temp ;встановлення SP на адресу останньої комірки Internal SRAM

sbi ACSR, 7 ;відключення живлення аналогового компаратора

;дозволяємо роботу із зовнішньою пам'яттю (звернення до системного контролеру)

ldi temp, 0b10000000

out MCUCR, temp

;Працюємо зі статичним семисегментним індикатором

;Встановлюємо вказівник X на адресу лівої пари знакомісць статичного індикатора

ldi XL, low (stat\_7seg\_left)

ldi XH, high (stat\_7seg\_left)

;Встановлюємо вказівник Y на адресу правої пари знакомісць статичного індикатора

ldi YL, low (stat\_7seg\_right)

ldi YH, high (stat\_7seg\_right)

;Встановлюємо вказівник Z на адресу регістра керування статичним індикатором

ldi ZL, low (stat\_7seg\_control)

ldi ZH, high (stat\_7seg\_control)

;\*\*\* Переходимо в нескінченний цикл \*\*\*

Infinite\_loop: ;нескінченний цикл

ldi dig, 0xFF ;заносимо початкове значення, яке виводиться на правий індикатор

ldi dig0, 0x00 ; заносимо значення для лівого індикатора

ldi const, 0x01 ;заносимо число 0х11, на яке буде зменшуватися початкове

ldi counter1, 0xFF ;ініціалізуємо значення лічильника алгоритму (FF - 255)

Loop1:

st X, dig0 ;видаємо маску на ліву пару знакомісць індикатора

st Y, dig ;видаємо маску на праву пару знакомісць індикатора

ldi temp, 0x03 ;вимикаємо всі точки і запалюємо всі знакомісця

st Z, temp ;записуємо цю маску в регістр керування індикатором

rcall long\_delay ;викликаємо затримку приблизно на 1.5 с

sub dig, const ; віднімаємо від початкового значення маски число 01h

dec counter1 ;зменшуємо значення лічильника кількості ітерацій на 1

breq loop ;перехід здійниться, якщо лічильник дорівнює 0

rjmp loop1 ;стрибаємо на виконання наступної ітерації алгоритму

Loop:

ldi counter2, 0xFF ;ініціалізуємо значення лічильника алгоритму

Loop2:

st X, dig0 ;видаємо маску на ліву пару знакомісць індикатора

st Y, dig ;видаємо маску на праву пару знакомісць індикатора

ldi temp, 0x03 ;вимикаємо всі точки і запалюємо всі знакомісця

st Z, temp ;записуємо цю маску в регістр керування індикатором

rcall long\_delay ;викликаємо затримку приблизно на 1.5 с

add dig, const ;додаємо до початкового значення маски число 01h

dec counter2 ;зменшуємо значення лічильника кількості ітерацій на 1

breq Infinite\_loop ;перехід здійниться, якщо лічильник дорівнює 0

rjmp Loop2 ;стрибаємо на виконання наступної ітерації алгоритму

;\*\*\* Підпрограма довгої затримки \*\*\*

long\_delay:

;\* Якщо в регістрову пару завантажити число 18432 (4800h), то затримка буде близько 2 секунд

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

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

ldi long\_delay\_low, 0x00;завантаження в регістрову пару коефіцієнта затримки

ldi long\_delay\_high, 0x36 ;(3600h), це буде затримка на 1.5 с

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

rcall short\_delay ;коротка затримка

sbiw long\_delay\_high: long\_delay\_low, 0b00000001 ;віднімання з 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 прапорцю нуля (зациклення)

ret ;повернення в основну програму

.EXIT ;кінець програми;

**ВИСНОВКИ**

На лабораторній роботі ми ознайомилися із функціональними можливостями і внутрішньою структурою стенду EV8031/AVR. Вивчили внутрішню організацію статичного семисегментного світлодіодного індикатора та принцип його роботи.

**ЛАБОРАТОРНА РОБОТА №3**

**ДИНАМІЧНИЙ семисегментний СВІТЛОДІОДНИЙ ИНДИКАТОР**

**Мета роботи:** Ознайомитися із функціональними можливостями і внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню організацію динамічного семисегментного світлодіодного індикатора, навчитися відображати інформацію на ньому.

**1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

**1.1 Загальні відомості**

При побудові підсистем відображення інформації розрізняють два підходи до її організації: динамічну і статичну схеми індикації.

Сутність динамічної індикації полягає в почерговому циклічному підключенні кожного знакомісця HLi індикатора до джерела інформації через спільну шину даних (дивись рисунок 1.1). Перебір знакомісць повинен відбуватися із досить високою швидкістю, оскільки лише в цьому випадку за рахунок інерції людського зору зображення буде здаватися цілісним і не блимати.

Вибір знакомісця індикатора здійснюється за допомогою дешифратора DA. У регістрі RD зберігається семисегментний (двійковий непозиційний) код, призначений для формування образу символа. У регістрі RA зберігається адреса (номер) знакомісця індикатора, задіяного у відображенні поточного символа.

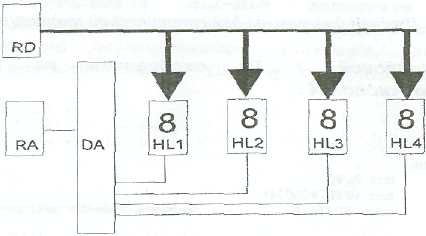


Рисунок 1.1 – Структурна схема динамічної індикації

До основних структурних елементів схеми організації динамічної індикації входять наступні:

RD – регістр даних, призначений для тимчасового зберігання семисегментного коду символа (цифри або букви), що відображається.

RA – регістр адреси, призначений для тимчасового зберігання номера знакомісця індикатора, на якому відображається символ.

DA – дешифратор, призначений для формування сигналу вибору знакомісця HLi індикатора, в залежності від заданого (в двійковому коді) номера на вході.

HL1 ... HL4 – 4 знакомісця, які утворюються семисегментними світлодіодними індикаторами.

При такій схемній організації значно зменшуються апаратні витрати. Але необхідно забезпечити достатній час світіння одного знакомісця, для того щоб не зменшувалася яскравість зображення. Як було зазначено вище, необхідно забезпечити таку частоту перебору знакомісця індикатора, щоб не було помітно блимання. Переваги такого способу відображення інформації помітні при кількості розрядів індикації більше, ніж 4.

У навчально- відлагоджувальному стенді EV8031/AVR динамічна індикація реалізована на одній із двох плат розширення за допомогою одного 4-розрядного семисегментного індикатора HL2. Управління динамічною індикацією здійснюється за допомогою порту В мікросхеми системного контролера, сигнали вибору відповідного знакомісця індикатора надходять по лініях 0 і 1 порту С мікроконтролера (РС0 і РС1 відповідно) до дешифратор адреси DD3.

Нижче, на рисунку 1.2, зображено функціональну схему підключення динамічного семисегментного світлодіодного індикатора до мікроконтролера, яку реалізовано в навчально-відлагоджувальному стенді EV8031/AVR. З рисунку видно, що для звернення до шини даних динамічного семисегментного світлодіодного індикатора (фактично, до сегментів конкретного знакомісця) необхідно зробити запис у комірку зовнішньої пам’яті за адресою 0x8001 (8001h). При цьому нульовий біт відповідає за запалювання сегмента "A", перший – сегмента "B" і т.д. Сьомий (найстарший) біт відповідає за запалювання точок нульового, другого і третього (рахуючи зліва направо) знакомісць, для першого зліва знакомісця цей біт відповідає за запалювання двокрапки. Для запалювання відповідного сегмента знакомісця або точки/двокрапки відповідний біт повинен бути встановлений в “1”. Вибір конкретного знакомісця індикатора кодується двома молодшими бітами комірки пам’яті за адресою 0x8002 (8002h). Якщо ці молодші біти рівні “00”, то вибрано крайнє ліве (нульове) знакомісце, якщо “01” – наступне (друге зліва) і т.д.

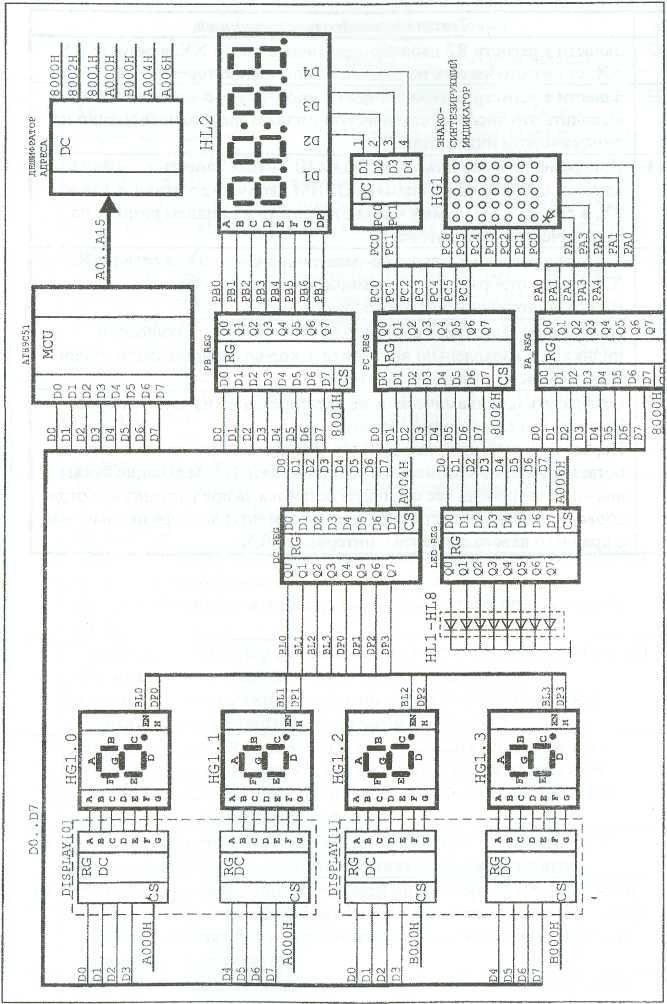


Рисунок 1.2 – Функціональна схема підключення динамічного семисегментного індикатора до мікроконтролера

**1.2 Порядок виконання роботи**

1.2.1 Запустіть IDE AVR Studio 4.

1.2.2 Активуйте створення нового проекту за допомогою вибору пункту New Project з пункту Project головного меню AVR Studio 4.

1.2.3 Виберіть тип створюваного проекту: проект буде створюватися мовою асемблер (пункт Atmel AVR Assembler) або С (пункт AVR GCC).

1.2.4 Нижче, в полі Location, вкажіть шлях по якому буде розташовуватися проект. Для зручності подальших маніпуляцій з проектом необхідно зберегти його в окремому каталозі (наполегливо рекомендується D:\AVR\_projects).

1.2.5 Вказати ім’я проекту (англійською мовою) в полі Project name.

1.2.6 Примусово (якщо це не зробила AVR Studio 4) встановити прапорець Create initial file. Бажано також встановити прапорець Create folder, що дозволить створити окремий каталог (шлях до якого вказано у полі Location) для збереження всіх файлів проекту всередині нього. Після зазначених дій натиснути кнопку Next >>.

1.2.7 У вікні Debug platform вибрати пункт AVR Simulator. У вікні Device вибрати пункт ATmega8515. Вибір будь-яких пунктів, що відрізняються від наведених вище, у зазначених вікнах призведе до подальших помилок і неможливості виконання лабораторної роботи!

1.2.8 Натиснути кнопку Finish для завершення створення проекту.

1.2.9 У вікні, що з’явилося, написати програму мовою С або асемблер з урахуванням варіанту завдання, зазначеного у таблиці 1.1.

1.2.10 Здійснити компіляцію проекту за допомогою вибору пункту Build з однойменного меню (або натиснути кнопку F7).

1.2.11 За наявності повідомлень про помилки або попередження повернутися до попереднього пункту і внести необхідні виправлення.

1.2.12 Обов’язково виконати покрокове відлагодження та трасування скомпільованої програми засобами меню Debug перед програмуванням мікроконтролера.

1.2.13 Перевірте наявність підключення USB-кабелю програматора до відповідного роз'єму системного блоку.

1.2.14 Для завантаження відкомпільованого проекту в мікроконтролер виберіть пункт меню Tools -> Program AVR -> Connect...

1.2.15 У вікні Platform, що з’явилося, виберіть пункт AVRISP mkII. У вікні Port виберіть пункт USB. Вибір будь-яких пунктів, що відрізняються від наведених вище, у зазначених вікнах призведе до подальших помилок і неможливості програмування мікроконтролера!

1.2.16 Натисніть на кнопку Connect ...

1.2.17 У вікні, що відкрилося, відразу перейдіть на вкладку Program. В полі Location вкажіть шлях до файлу з розширенням \*.hex. Він розташований в каталозі (або підкаталогах) проекту, а його ім'я співпадає із ім'ям проекту. Цей файл містить машинні коди, які будуть виконуватися мікроконтролером після завантаження в нього файлу \*.hex.

1.2.18 Натисніть кнопку Program. При цьому відбувається програмування (повне стирання і запис) Flash-пам'яті (пам'яті програм) мікроконтролера. Кількість гарантованих операцій стирання/запису цієї пам'яті достатньо мала, тому не слід натискати кнопку Program без потреби і бездумно перепрограмувати мікроконтролер!

1.2.19 Візуально оцініть правильність роботи алгоритму індикації для статичного семисегментного світлодіодного індикатора.

**2 ХІД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ**

**2.1 Результати виконання завдання**

Завдання для лабораторної роботи зображено в таблиці 1.1.

Таблиця 1.1 – Завдання

|  |  |
| --- | --- |
| № | Опис завдання |
| 8 | На динамічному семисегментному індикаторі реалізувати відображення станів лічильника, що працює за алгоритмом: 59:59 → 59:58 → ... → 00:01 → 00:00 → ... → 59:59 (59:59 → 00:00 → 59:59). Затримка на відображення одного стану дорівнює 1 с і організовується програмно. Семисегментні коди символів (або знакогенератор) задаються у вигляді констант (на початку програми), які необхідно зберігати в SRAM (ОЗП), починаючи із адреси 0090h. |

Фрагмент коду циклу виконання завдання на мові С зображено в лістингу 1.

Лістинг 1 – Виконання завдання

/\*\*Програма до лабораторної роботи №3 (стенд EV8031/AVR) \*\*\*

\*\*\*Працюємо з динамічним семисегментним індикатором \*\*\*

\*\*\*Відображаємо на динамічному семисегментному індикаторі число 1234 \*\*\*/

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

#include <avr/io.h>

#include <avr/iom8515.h>

#include <util/delay.h>

//адреса регістра даних для знакомісця динамічного семисегментний індикатора

#define dyn\_7seg\_data 0x8001

//адреса регістра вибору рядків знакосинтезуючого індикатора і управління

//запалюванням знакомісць динамічного семисегментного індикатора

#define synthes\_5x7\_row\_dyn\_7seg\_control 0x8002

//визначаємо тривалість затримки (як параметр функції затримки),

//тобто час індикації даних на одному знакомісці індикатора

#define waiting 5

int main (void) {

//початкова ініціалізація контролера

//дозволяємо роботу із зовнішньою пам'яттю (звернення до системного контролеру стенду)

MCUCR = 1 << SRE;

//відключення живлення аналогового компаратора

ACSR = 1 << ACD;

volatile unsigned char\* A = (unsigned char \*) 0x0090;

volatile unsigned char\* B = (unsigned char \*) 0x0100;

volatile unsigned char\* D = (unsigned char \*) 0x0110;

volatile unsigned char\* C = (unsigned char \*) 0x0120;

\*A = 0xBF;//(10111111) 0

\*(A+1) = 0x86; //(10000110)1

\*(A+2) = 0xDB; //(11011011)2

\*(A+3) = 0xCF; //(11001111)3

\*(A+4) = 0xE6; //(11100110)4

\*(A+5) = 0xED; //(11101101)5

\*(A+6) = 0xFD; //(11111101)6

\*(A+7) = 0x87; //(10000111)7

\*(A+8) = 0xFF; //(11111111)8

\*(A+9) = 0xEF; //(11101111)9

\*B = 0xBF; //(10111111) 0

\*(B+1) = 0x86; //(10000110)1

\*(B+2) = 0xDB; //(11011011)2

\*(B+3) = 0xCF; //(11001111)3

\*(B+4) = 0xE6; //(11100110)4

\*(B+5) = 0xED; //(11101101)5

\*(B+6) = 0xFD; //(11111101)6

\*(B+7) = 0x87; //(10000111)7

\*(B+8) = 0xFF; //(11111111)8

\*(B+9) = 0xEF; //(11101111)9

\*C = 0xBF; //(10111111) 0

\*(D+1) = 0x86; //(10000110)1

\*(D+2) = 0xDB; //(11011011)2

\*(D+3) = 0xCF; //(11001111)3

\*(D+4) = 0xE6; //(11100110)4

\*(D+5) = 0xED; //(11101101)5

\*(D+6) = 0xFD; //(11111101)6

\*(D+7) = 0x87; //(10000111)7

\*(D+8) = 0xFF; //(11111111)8

\*(D+9) = 0xEF; //(11101111)9

\*D = 0xBF; //(10111111) 0

\*(C+1) = 0x86; //(10000110)1

\*(C+2) = 0xDB; //(11011011)2

\*(C+3) = 0xCF; //(11001111)3

\*(C+4) = 0xE6; //(11100110)4

\*(C+5) = 0xED; //(11101101)5

\*(C+6) = 0xFD; //(11111101)6

\*(C+7) = 0x87; //(10000111)7

\*(C+8) = 0xFF; //(11111111)8

\*(C+9) = 0xEF; //(11101111)9

unsigned int count1;

unsigned int count2 = 0;

//встановлюємо вказівник на адресу регістра даних динамічного індикатора

volatile unsigned char\* pd = (unsigned char \*) dyn\_7seg\_data;

// встановлюємо покажчик на адресу регістра управління динамічним індикатором

volatile unsigned char\* pc = (unsigned char\*) synthes\_5x7\_row\_dyn\_7seg\_control;

while (1) { //нескінченний цикл

if (count2 = 1){

for (count1 = 50; count1 <=50; count1++){ // 1 cекунда

\*pd = (\*D+5);

\*pc = 0x00; //запалюємо крайнє ліве знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

\*pd = (\*C+9);

\*pc = 0x01; //запалюємо друге зліва знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

\*pd = (\*B+5);

\*pc = 0x02; //запалюємо друге справа знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

\*pd = (\*A+9);

\*pc = 0x03; //запалюємо крайнє праве знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

}

\*A=\*A-1;

if (\*A){

\*A=\*A+9;}

if (\*A+9){

\*B=\*B-1;}

if (\*B)

{\*B=\*B+6;}

if (\*B+6)

{\*D=\*D-1;}

if (\*C)

{\*C=\*C+9;}

if (\*C+9)

{\*D=\*D-1;}

if (\*D)

{\*D=\*D+6;

count1 --;}

}

}

if (count2 = 0){

for (count1 = 50; count1 <=50; count1++){ // 1 cекунда

\*pd = \*D;

\*pc = 0x00; //запалюємо крайнє ліве знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

\*pd = \*C;

\*pc = 0x01; //запалюємо друге зліва знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

\*pd = \*B;

\*pc = 0x02; //запалюємо друге справа знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

\*pd = \*A;

\*pc = 0x03; //запалюємо крайнє праве знакомісце

\_delay\_ms (waiting); //викликаємо затримку на 5 мс

}

\*A=\*A+1;

if (\*A+10){

\*A=\*A-10;}

if (\*A-10){

\*B=\*B+1;}

if (\*B+6)

{\*B=\*B-6;}

if (\*B-6)

{\*D=\*D+1;}

if (\*C+10)

{\*C=\*C-10;}

if (\*C-10)

{\*D=\*D+1;}

if (\*D+6)

{\*D=\*D-6;

count1 ++;}

}

return 0;

}

**ВИСНОВКИ**

На лабораторній роботі ми ознайомилися із функціональними можливостями і внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчили внутрішню організацію динамічного семисегментного світлодіодного індикатора та навчилися відображати інформацію на ньому.

**4 Лабораторна робота №4**

**СТАТИЧНИЙ семисегментнИЙ СВІТЛОДІОДНИЙ ИНДИКАТОР**

**Мета роботи**

Ознайомитися із функціональними можливостями і внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню організацію рідкокристалічного індикатора, навчитися відображати інформацію на ньому.

**4.1 Короткі теоретичні відомості**

В даний час для відображення інформації широко використовуються алфавітно-цифрові РКІ-модулі. Такий модуль являє собою один або кілька рядків, кожен з яких складається із декількох знакомісць. Кожне знакомісце, в свою чергу, представляє собою масив точок (як правило, 5х7). У даній лабораторній роботі використовується РКІ-модуль, який містить 2 рядки по 16 символів (знакомісць).

Алфавітно-цифровий РКІ-модуль має наступні зовнішні контакти:

– Vdd – живлення модуля;

– Vss – спільний контакт (земля);

– V0 – керування контрастом зображення;

– RS – вибір регістру команд або даних;

– E – строб запису/читання;

– R/!W – сигнал читання/запису;

– DB7 ... DB 0 – двонаправлені лінії обміну даними (команди/символи);

– A – вивод аноду підсвітки модуля;

– K – вивод катоду підсвітки модуля.

Модуль, що використовується в лабораторній роботі, має світлодіодну підсвітку. Вивод R/!W наглухо заземлений, тому модуль доступний тільки для операцій запису команд/даних, читати з нього не можна.

Часову діаграму процедури запису команд/даних в РКІ-модуль показано на рисунку 4.1.

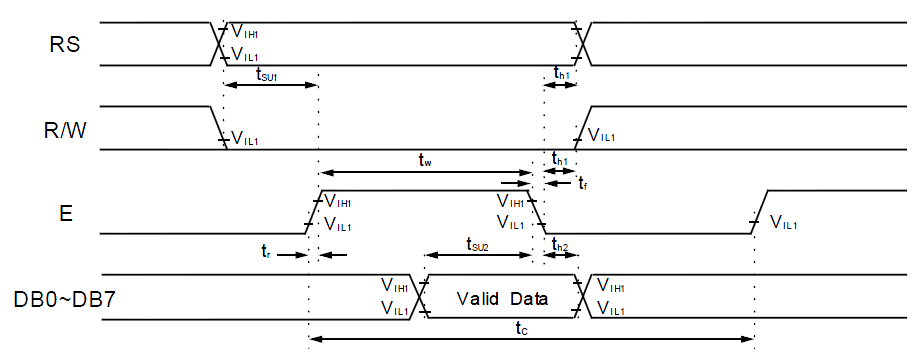


Рисунок 4.1 – Часова діаграма запису інформації в РКІ-модуль

У лабораторному стенді задача формування наведеної часової діаграми виконується системним контролером автоматично. Для підтримки інтерфейсу з РКІ-модулем в системному контролері передбачено 2 комірки із адресами 0x8004 і 0x8005. При записі байта в комірку із адресою 0x8004 він надійде в регістр команд РКІ-модуля, а з адресою 0x8005 – в регістр даних.

Спрощену внутрішню структуру РКІ-модуля показано на рисунку 4.2.

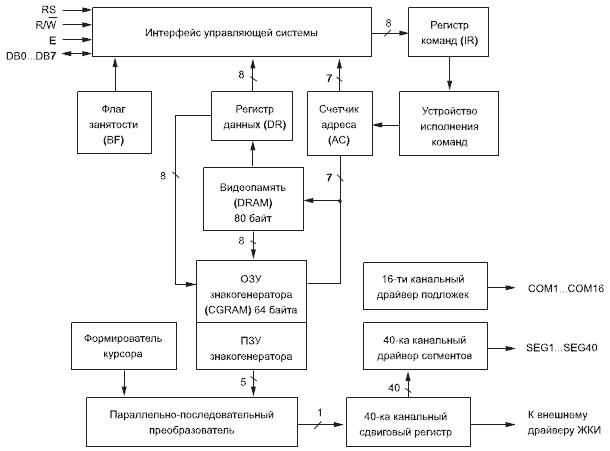


Рисунок 4.2 – Спрощена внутрішня структура РКІ-модуля

РКІ-модуль, що використовується в стенді, має 80 комірок відеопам’яті DDRAM (для зберігання символьних повідомлень). Для першого рядка модуля зарезервовано 40 комірок пам’яті DDRAM із адресами від 00h до 27h. Для другого рядка модуля також зарезервовано 40 комірок із адресами від 40h до 67h (таким чином, в адресах комірок пам’яті між першим і другим рядком є “дірка”). При послідовному запису інформації в DDRAM внутрішній контролер РКІ-модуля автоматично переходить із останньої комірки першого рядка (27h) на першу комірку другого рядка (40h), якщо ввімкнено відповідний режим автоматичного збільшення адрес. Слід пам’ятати, що на екрані РКІ-модуля відображається вміст лише 16-ти комірок для першого рядка і 16-ти комірок для другого рядка. При виконанні команди зсуву екрану вміст комірок DDRAM не змінюється!, змінюються лише вказівники початку видимих частин рядків. Наприклад, після ініціалізації модуля, ми бачимо в першому його рядку комірки DDRAM із адресами від 00h до 0Fh, а в другому – із адресами від 40h до 4Fh (рисунок 4.3).

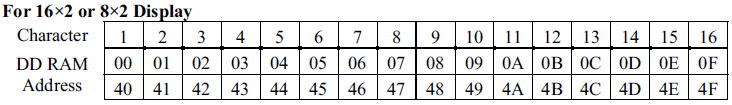


Рисунок 4.3 – Взаємна відповідність знакомісць і комірок пам’яті DDRAM для РКІ-модуля після скидання

Після виконання команди зсуву екрану вліво в першому рядку буде відображатися вміст комірок з адресами від 01h до 10h, а в другому – від 41h до 50h. Аналогічні дії виконуються і при зсуві екрану вправо.

РКІ-модуль також містить ПЗП знакогенератора (CGROM) і ОЗП знакогенератора (CGRAM). ПЗУ містить наперед встановлені виробником символи для відображення модулем. Для виведення такого наперед встановленого символу достатньо записати в відповідну комірку відеопам’яті DDRAM його код (або адресу в пам’яті CGROM). Кодову таблицю РКІ-модуля, що містить кириличні символи, показано на рисунку 4.4.

ОЗП знакогенератора (CGRAM) може використовуватися для створення символів користувача.

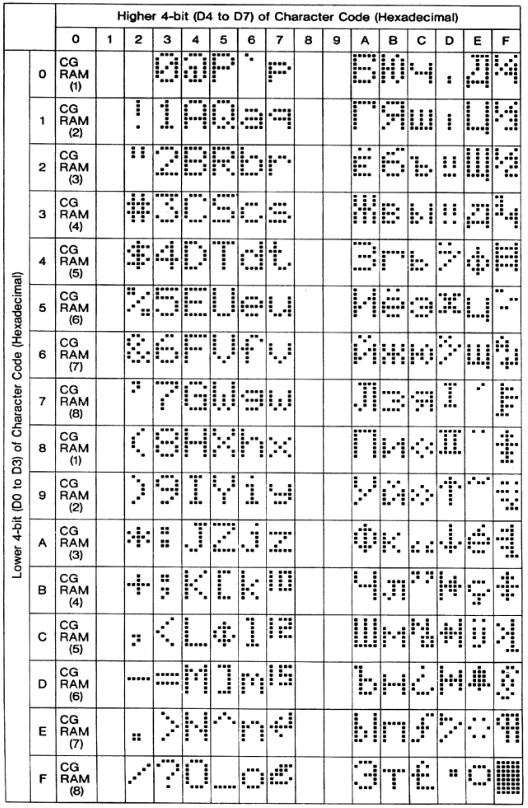


Рисунок 4.4 – Кодова таблиця РКІ-модуля

Для управління РКІ-модулем і його ініціалізації використовуються спеціальні команди. Таблицю таких команд показано на рисунку 4.5.

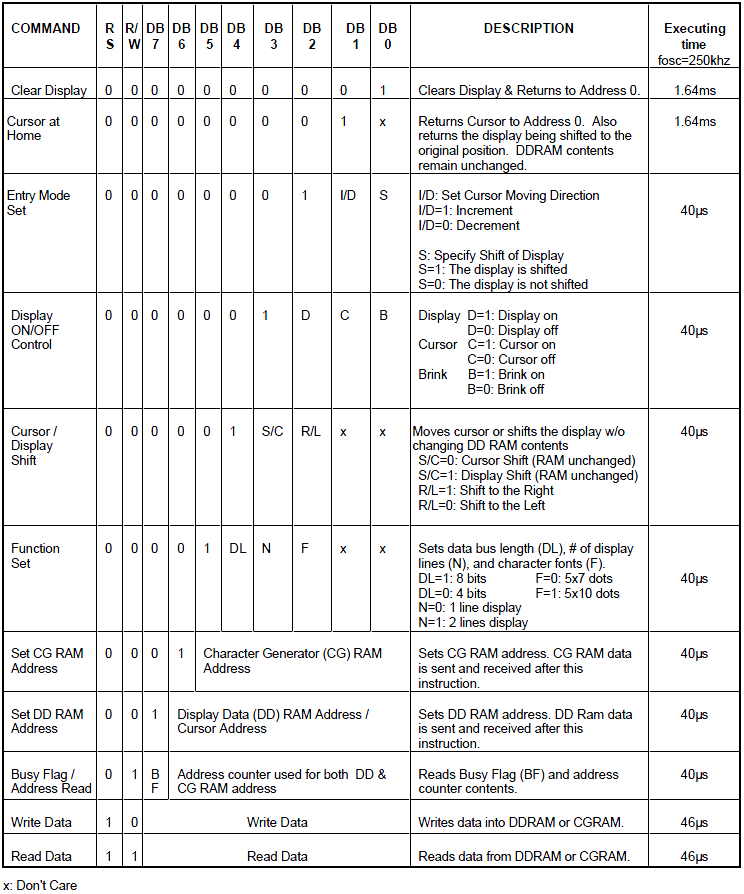


Рисунок 4.5 – Таблиця команд РКІ-модуля

Слід звернути увагу, що після подачі чергової команди або байту даних необхідно витримати паузу не менше, ніж зазначено в таблиці на рисунку 4.5. Протягом цього часу контролер РКІ-модуля обробляє попередній прийнятий байт і може некоректно відреагувати на байт, який знову надійшов. Час виконання команд в таблиці на рисунку 4.5 зазначено для випадку, коли внутрішня тактова частота роботи контролера РКІ-модуля становить 250 кГц. На практиці зустрічаються РКІ-модулі, що працюють на тактових частотах або 250 кГц, або 270 кГц. При частоті роботи в 270 кГц затримки на виконання команд/обробку даних, зазначені в таблиці на рисунку 4.5, будуть трохи меншими. На практиці, з метою забезпечення сумісності з будь-якими модулями, рекомендується використовувати затримки, які дещо перевищують ті, які наведено в таблиці на рисунку 4.5.

При ввімкненні живлення програмісту необхідно здійснити ініціалізацію РКІ-модуля. У лабораторному стенді використовується 8-бітний протокол обміну між мікроконтролером і РКІ-модулем. Ініціалізація РКІ-модуля в цьому випадку здійснюється згідно з алгоритмом, показаним на рисунку 4.6. Часові затримки на виконання алгоритму ініціалізації (на рисунку 4.6) вказано для РКІ-модуля із внутрішньою тактовою частотою у 270 кГц. При використанні більш “повільного” РКІ-модуля, наприклад, з частотою 250 кГц, всі зазначені затримки слід множити на коефіцієнт 270/250.

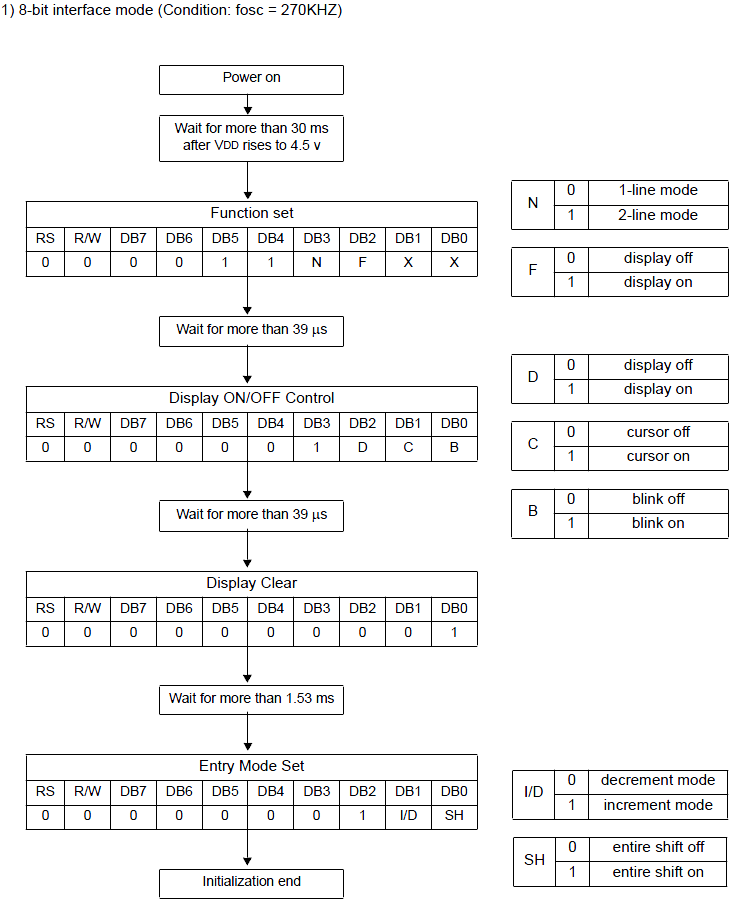


Рисунок 4.6 – Алгоритм ініціалізації РКІ-модуля

Нижче наведено приклад програми мовою асемблер, яка здійснює відображення інформаційного повідомлення на всіх знакомісцях першого і другого рядка рідкокристалічного індикатора (час виведення одного символу становить 200 мс). Після виведення повідомлення виробляється циклічний зсув вліво вмісту індикатора із періодом 1 с. Часові затримки організовуються програмним способом.

**4.3 Результати виконання завдання**

Завдання для лабораторної роботи зображено в таблиці 4.3.1

Таблиця 4.3.1 – Завдання

|  |  |
| --- | --- |
| № | Опис завдання |
| 8 | На рідкокристалічному індикаторі (в довільній позиції) реалізувати відображення секундоміра в форматі 59 → 00. Часові затримки організовуються програмно. |

Код програми наведено в лістингу 4.3.2

Лістинг 4.3.2

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

#include <avr/io.h>

#include <avr/iom8515.h>

#include <util/delay.h>

//задаємо константу - адресу регістра команд в РКІ

#define lcd\_comm 0x8004

//задаємо константу - адресу регістра даних в РКІ

#define lcd\_data 0x8005

int main (void) {

//початкова ініціалізація міконтролера

//дозволяємо роботу із зовнішньою пам'яттю (звернення до системного контролеру)

MCUCR = 1 << SRE;

//відключення живлення аналогового компаратора

ACSR = 1 << ACD;

//встановлюємо вказівник на адресу регістра команд РКІ

volatile unsigned char\* c = (unsigned char \*) lcd\_comm;

//встановлюємо вказівник на адресу регістра даних РКІ

volatile unsigned char\* d = (unsigned char \*) lcd\_data;

volatile unsigned char\* A = (unsigned char \*) 0x0090;

\_delay\_ms (1000); //встановлюємо затримку по ввімкненню живлення на 1 с

unsigned int counter = 0x59;

//початкова ініціалізація РКІ

\*c = 0x38; //Function Set

\_delay\_ms (10); //викликаємо затримку на 10 мс

\*c = 0x0E; //Display ON/OFF Control

\_delay\_ms (10); //викликаємо затримку на 10 мс

\*c = 0x01; //Display clear

\_delay\_ms (10); //викликаємо затримку на 10 мс

\*c = 0x06; //Entry Mode Set

\_delay\_ms (10); //викликаємо затримку на 10 мс

//Виводимо перший (верхній) рядок повідомлення

\*A = 0x30;//0

\*(A+1) = 0x31;//1

\*(A+2) = 0x32;//2

\*(A+3) = 0x33;//3

\*(A+4) = 0x34;//4

\*(A+5) = 0x35;//5

\*(A+6) = 0x36;//6

\*(A+7) = 0x37;//7

\*(A+8) = 0x38;//8

\*(A+9) = 0x39;//9

while (1) { //нескінченний цикл

\*d = \*(A+(counter / 10)); //код символу

\_delay\_ms (100); //викликаємо затримку на 200 мс

\*d = \*(A+(counter % 10)); //код символу

\_delay\_ms (100); //викликаємо затримку на 200 мс

\_delay\_ms (790); //викликаємо затримку на 790 мс

\*c = 0x01; //Display clear

\_delay\_ms (10); //викликаємо затримку на 10 мс

if (counter !=00){

counter = counter - 0x01;}

}

return 0;

}

**ВИСНОВКИ**

Ознайомилися з функціональними можливостями, внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчили внутрішню організацію рідкокристалічного індикатора, навчилися відображати інформацію на ньому.

**5 ЛАБОРАТОРНА РОБОТА №5**

**СИСТЕМА ПЕРЕРИВАНЬ МІКРОКОНТРОЛЕРА ATMEGA8515. ЗОВНІШНІ ПЕРЕРИВАННЯ**

**Мета роботи**

Ознайомитися із функціональними можливостями та внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню організацію системи переривань мікроконтролерів AVR на прикладі ATMega8515. Навчитися конфігурувати та обробляти зовнішні переривання.

**5.1 Короткі теоретичні відомості**

5.1.1 Опитування дискретних сигналів

Для введення інформації широко застосовуються кнопочні перемикачі та контактні клавіатури. Сигнал таких перемикачів формується шляхом замикання (розмикання) електричного кола. Сигнал, що формується контактною парою, супроводжується дріб’язком, тривалість якого становить приблизно 8 – 15 мс (приклад діаграми сигналу при комутації ключа наведено на рисунку 5.1).

|  |
| --- |
|  |
| Рисунок 5.1 – Часова діаграма сигналу при замиканні і розмиканні контактів |

Для усунення дріб’язку в отриманому сигналі на виході контактної пари встановлюють спеціальні пристрої формування. Прикладом такого пристрою формування, заснованого на принципі безпосереднього встановлення RS-тригера, наведено на рисунку 5.2.

|  |
| --- |
|  |
| Рисунок 5.2 – Схема усунення дріб’язку за допомогою RS-тригера |

Для зменшення апаратних витрат дуже часто застосовують програмне усування дріб’язку. Воно полягає в повторному опитуванні контактної пари із затримкою в 15 мс (після визначення моменту натискання). Співпадіння результатів опитування означає, що кнопку було натиснуто, розбіжність – дріб’язок контактів. Слід зауважити, що час дріб’язку залежить від стану контактів та інтенсивності їх експлуатації.

У лабораторному стенді для емуляції зовнішніх переривань використовуються кнопки SW15 та SW16. Для того, щоб ці кнопки служили джерелами переривань, у лабораторному стенді повинні бути встановлені джампери (перемички) X19 та X20. У відпущеному стані будь-якої із кнопок на відповідний зовнішній контакт МК надходитиме рівень логічної “1”, а у натиснутому – логічний “0”. Для апаратного усунення дріб’язку контактів паралельно кнопкам на платі стенду встановлені керамічні конденсатори C23 та С24 ємністю 0.1 мкФ. Усування дріб’язку в даному випадку здійснюється за рахунок того, що напруга на конденсаторі, як відомо з курсу теоретичних основ електротехніки, змінюється не стрибком, а плавно.

**5.2 Порядок виконання роботи**

5.2.1 Запустіть IDE AVR Studio 4.

5.2.2 Створіть новий проект у IDE AVR Studio 4.

5.2.3 У вікні, що з’явилося, написати програму мовою С або асемблер з урахуванням варіанта завдання, зазначеного в таблиці 5.1. У всіх випадках мається на увазі робота із зовнішніми перериваннями INT0, INT1.

5.2.4 Здійснити компіляцію проекту.

5.2.5 За наявності повідомлень про помилки або попередження повернутися до попереднього пункту і внести необхідні виправлення. Обов’язково виконати покрокове відлагодження та трасування скомпільованої програми засобами меню Debug перед програмуванням мікроконтролера.

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

5.2.7 Завантажити виконуваний файл проекту в мікроконтролер.

5.2.8 Візуально оцінити правильність роботи написаної програми.

**5.3 Результати виконання завдання**

Завдання для лабораторної роботи зображено в таблиці 4.3.1

Таблиця 5.3.1 – Завдання

|  |  |
| --- | --- |
| № | Опис завдання |
| 8 | При появі заднього фронту на вході INT0 запускати секундомір, а при появі низького рівня на вході INT1 – зупиняти його. Виводити значення секундоміра на праву пару знакомісць статичного семисегментного індикатора. Організацію затримок виконувати програмно. |

Код програми наведено в лістингу 5.3.2

Лістинг 5.3.2

/\*\*\*\*\* Програма до лабораторної роботи №5 (стенд EV8031/AVR) \*\*\*\*\*

\*\*\* Робота із зовнішніми перериваннями INT0 та INT1 (кнопки SW15 та SW16) \*\*\*

\*\*\* При натисканні на будь-яку із кнопок (SW15 або SW16) на відповідному виводі мікроконтролера з'являється логічний "0" \*\*\*

\*\*\* Переривання INT0 спрацьовує по низькому рівню, а INT1 - по зрізу \*\*\*

\*\*\* По перериванню INT0 збільшуємо стан змінної-лічильника, який виводиться на лінійку світлодіодів \*\*\*

\*\*\* За переривання INT1 зменшуємо стан змінної-лічильника, який виводиться на лінійку світлодіодів \*\*\*

\*\*\* Затримка на оновлення стану світлодіодної лінійки складає 0.5 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 у стенді

//адреса правої пари знакомісць статичного семисегментного індикатора

#define stat\_7seg\_right 0xB000

//адреса регістра керування запаленням/гасінням точок/знакомісць статичного семисегментний індикатора

#define stat\_7seg\_control 0xA004

//Декларуємо допоміжну змінну - ознаку того, що нам потрібно робити:

//якщо її значення дорівнює 0xFF, то лічильник працює на додавання

//якщо її значення дорівнює 0x00, то лічильник працює на віднімання

//Значення цієї ознаки змінюється в обробнику відповідного переривання

volatile unsigned char attribute = 0xFF;

//Переривання IRQ0 (по входу INT0)

ISR(INT0\_vect){

attribute = 0xFF; //лічильник повинен лічити вгору

}

//Переривання IRQ1 (по входу INT1)

ISR(INT1\_vect){

attribute = 0x00; //лічильник повинен зупинитися

}

int main(void) {

//встановлюємо вказівник на адресу лінійки світлодіодів

// volatile unsigned char \*p= (unsigned char\*) led\_line;

//Змінна-лічильник, стан якої і відображається на світлодіодах

volatile unsigned char dig;

//встановлюємо вказівник на адресу правої пари знакомісць статичного індикатора

volatile unsigned char\* pr = (unsigned char\*) stat\_7seg\_right;

//встановлюємо вказівник на адресу регістра керування статичним індикатором

volatile unsigned char\* pc = (unsigned char\*) stat\_7seg\_control;

//Початкова ініціалізація контролера

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

//дозволяємо роботу із зовнішньою пам'яттю і налаштовуємо зовнішні переривання

MCUCR = 1 << SRE | 0<< ISC11 | 0<< ISC10 | 1<<ISC01 | 0<<ISC00;

//переривання: INT1 викликається по низькому рівню, а INT0 - по задньому фронту

//на відповідних виводах (біти 0, 1 (ISC00, ISC01) та 2, 3 (ISC10, ISC11) скинуто)

GICR = 1 << INT0 | 1<<INT1; //дозволяємо зовнішні переривання INT0 та INT1

//встановлюємо прапорець глобального дозволу переривань

//у цьому випадку працюватимуть тільки переривання INT0 та INT1

sei();

while(1) {//нескінченний цикл

//перевірка значення змінної-ознаки та аналіз подальших дій із лічильником

if(attribute== 0xFF) {

\*pl= dig; //видаємо маску на індикатор

//вимикаємо всі крапки і запалюємо ліві знакомісця, видаємо маску на індикатор

\*pc= 0x0C;

//викликаємо затримку на секунду для відображення станів лічильника

\_delay\_ms(1000);

dig++;

if ((dig == 0x0A) || (dig == 0x1A) || (dig == 0x2A) || (dig == 0x3A) || (dig == 0x4A) || (dig == 0x5A)) {

dig += 0x06;

}

if (dig == 0x60) {

dig = 0x00;

}

}

}

return 0;

}

**ВИСНОВКИ**

На лабораторній роботі ми ознайомилися із функціональними можливостями та внутрішньою структурою відлагоджувального стенду EV8031/AVR. Вивчити внутрішню організацію системи переривань мікроконтролерів AVR на прикладі ATMega8515. Навчилися конфігурувати та обробляти зовнішні переривання.