**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).

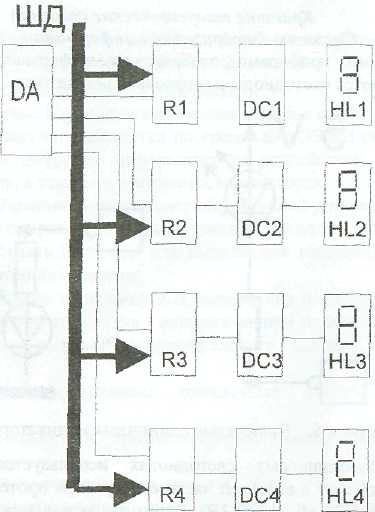


Рисунок 1.1 – Структурна схема організації статичної індикації

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.