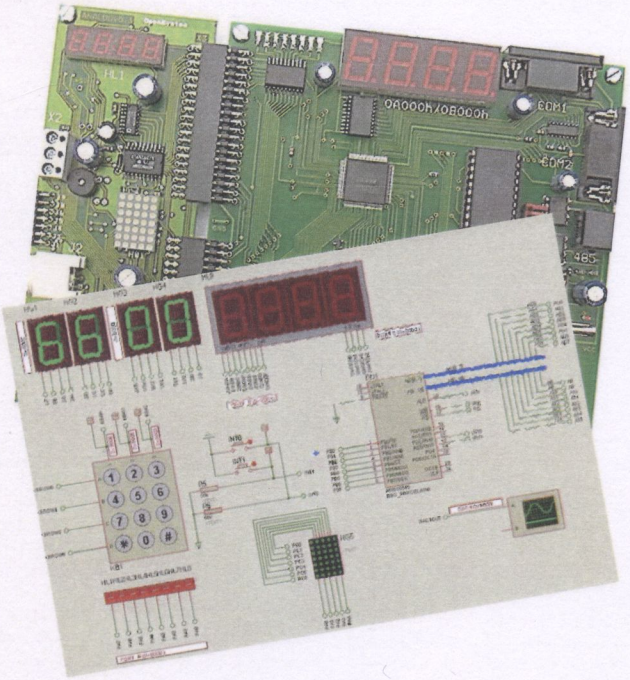
МІКРОПРОЦЕСОРНА ТЕХНІКА

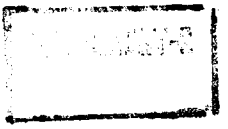


Міністерство освіти і науки України
Вінницький національний технічний університет

**С. М. Цирульник, О. Д. Азаров, Л. В. Крупельницький,
Т. І. Трояновська**

МІКРОПРОЦЕСОРНА ТЕХНІКА

Навчальний посібник



Вінниця
ВНТУ
2017

УДК 004.272.43(075)
ББК 32.971.32-043я73
М59

Автори:
Цирульник С. М., Азаров О. Д., Крупельницький Л. В., Трояновська Т. І.

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 14 від 23 червня 2016 р.)

Рецензенти:
С. І. Перевозніков, доктор технічних наук, професор
В. Ю. Кучерук, доктор технічних наук, професор
Л. Б. Ліщинська, доктор технічних наук, професор

Мікропроцесорна техніка : навч. посіб. / Цирульник С. М., М59 Азаров О. Д., Крупельницький Л. В., Трояновська Т. І. – Вінниця : ВНТУ, 2017. – 123 с.

У навчальному посібнику розглянуто особливості програмування та моделювання роботи електричних схем з використанням мікроконтролерів загального призначення виробництва Atmel, що об'єднані під загальною маркою AVR. Посібник призначений для студентів технічних спеціальностей усіх форм навчання при вивченні дисциплін «Архітектура комп'ютерів», «Комп'ютерна схемотехніка» спеціальності 123 – «Комп'ютерна інженерія», інженерів та вчених, а також для всіх, хто вже має певні основні знання про побудову й функціонування мікроконтролерів, бажає зрозуміти й вивчити мікроконтролери AVR і успішно запровадити в життя власні ідеї.

УДК 004.272.43(075)
ББК 32.971.32-043я73

ЗМІСТ

ВСТУП

Промисловий випуск мікропроцесорів (МП), імовірно, на кілька сотень порядків більший, ніж випуск персональних комп'ютерів. Багато функцій, а також технічні особливості МП такі, що більшість завдань, реалізованих на МП, просто неможливо реалізувати на звичайних комп'ютерах.

Відмінність програмування мікропроцесорних систем (МПС) від класичного полягає в тому, що крім математичних знань, знання мов програмування, необхідно знати апаратну частину об'єкта, його фізичні або електричні властивості.

Програмування МПС є не менш складною й цікавою галуззю знань, ніж написання програмного забезпечення для комп'ютера, а з урахуванням роботи з реальними процесами – можна вважати, що ця робота дуже наближена до мистецтва.

Лабораторні роботи є найбільш дорогим видом навчальних занять, організація якого на сучасному рівні виявляється практично недоступною з економічних причин для більшості навчальних закладів України. Проте перехід до фундаментальної освіти припускає відповідну зміну мети навчальних лабораторних досліджень. Замість завдань отримання навичок практичної роботи з конкретними об'єктами, які виявляються схильними до прискореного морального старіння, слід шукати підходи до освоєння загальних принципів функціонування, покладених в основу при створенні цілих класів об'єктів.

Фірмою «Open System» [1] випускається навчально-лабораторний стенд EV8031/AVR. Він дозволяє виконувати широкий спектр лабораторних робіт під час вивчення курсів «Комп'ютерна електроніка», «Комп'ютерна схемотехніка», «Архітектура комп'ютерів», «Моделювання компонентів комп'ютерних сиситем».

Модульна конструкція стенда максимально скорочує час на підготовку устаткування до лабораторних робіт і дозволяє, за необхідності, розширювати його функціональні можливості відповідно до специфіки дисциплін, які вивчаються. Стенд забезпечує максимальну доступність та наочність процесу обробки даних, що допомагає оволодіти сучасними методами програмування та налагодження програм, дослідити принципи роботи мікропроцесорних пристроїв (однокристальних ЕОМ серії MSC-51 та мікроконтролерів архітектури AVR) та особливості узгодження їх з периферійними пристроями.

Стенд побудований на сучасній елементній базі. Наявність системного та периферійного інтерфейсів дозволяє використовувати стенд для налагодження будь-яких систем.

Однак придбання та утримання навчальним закладом стенда EV8031/AVR для навчальних лабораторій у належній кількості є складною задачею, тому альтернативою мають стати віртуальні стенди. Цілеспрямоване застосування комп'ютерів як засобів навчання дозволяє кардинальним чином підвищити роль самостійної роботи в процесі отримання та експериментальної перевірки знань. У роботах [14–16] розглядаються особливості застосування пакета схемотехнічного аналізу Proteus VSM у навчальному процесі при вивченні курсу «Мікропроцесорна техніка».

Застосування комплексного підходу, який поєднує комп'ютерне моделювання роботи віртуального стенда для попереднього ознайомлення з лабораторним обладнанням та реального стенду EV8031/AVR, є золотою серединою, що дозволить у сучасних умовах існування навчальних закладів готувати високоякісних фахівців. Такий підхід підвищує ефективність проведення лабораторної роботи та дозволяє зменшити матеріальні витрати на ремонт лабораторного обладнання.

Стенд орієнтований на використання мікроконтролера серії MSC-51. На даний час дана архітектура є застарілою та малоефективною, тому в наш час не має застосування. Але є можливість використовувати мікроконтролер з архітектурою AVR – ATmega8515. Однак оригінальні методичні вказівки для роботи зі стендом [1] містять лише декілька рядків, що присвячені роботі з мікроконтролером з архітектурою AVR, тому самостійне опанування стенду як викладачем, так і студентом викликає певні труднощі. На чисельні звернення навчальних закладів, де використовується цей стендом, про доопрацювання методичних вказівок фірма «Open System» не реагує. Тому більшість навчальних закладів ВНЗ I-IV рівнів акредитації самотужки вирішує такі проблеми.

Метою роботи є розробка комплекту методичного забезпечення для стенда EV8031/AVR, і призначена вона для студентів технічних спеціальностей усіх форм навчання, інженерів і вчених, а також для всіх, хто вже має певні основні знання про побудову й функціонування мікроконтролерів, бажає зрозуміти й вивчити однокристальні мікроконтролери AVR і успішно запровадити в життя власні ідеї, щоб не відстати від «потяга», що рухається в технологічне майбутнє.

Робота ґрунтується на досвіді викладання дисциплін «Комп'ютерна електроніка», «Комп'ютерна схемотехніка», «Архітектура комп'ютерів», «Моделювання компонентів комп'ютерних сиситем» на факультеті інформаційних технологій та компютерної інженерії, пройшла апробацію у Вінницькому національному технічному університеті для студентів спеціальності «Комп'ютерні системи та мережі», протягом 2008–2016 років.

Мікропроцесорна техніка використовується широким спектром інформаційно-вимірювальних систем для обчислення результатів і коригування похибок аналого-цифрового та цифроаналогового перетворень.

1 ОСОБЛИВОСТІ ЗАСТОСУВАННЯ ЛАБОРАТОРНОГО МАКЕТА EV8031/AVR

1.1 Особливості застосування лабораторного макета EV8031/ AVR

Стенд являє собою мікропроцесорний контролер, оснащений пам'яттю програм, пам'яттю даних і різноманітними периферійними пристроями. Він дозволяє налагоджувати програми, написані мовами С та Асемблер.

Стенд дозволяє використовувати мікроконтролери x51 (AT89S52) та AVR (ATmega8515). Завантаження програми проводиться з персонального комп'ютера (ПК) паралельним портом (LPT). Для подачі на стенд напруги живлення (+5 В) його необхідно підключити до USB-порту ПК. Для під'єднання стенда до ПК останній повинен мати LPT-порт для програмування і USB-порт для живлення. За наявності USB-програматора можна обійтись лише USB-портом. Схему підключення стенда EV8031/AVR до ПК наведено на рис. 1.1.

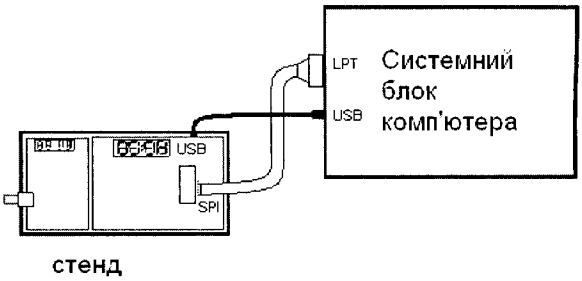


Рисунок 1.1 – Схема підключення стенда EV8031/ AVR до персонального комп'ютера

На рис. 1.2 наведено схему з'єднувача для програмування AVR мікроконтролера.

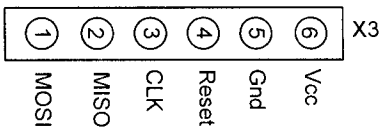


Рисунок 1.2 – З'єднувач для програмування мікроконтролера AVR

Стенд містить набір пристроїв, які дозволяють освоїти основні прийоми програмування. Структурну схему стенда наведено на рис. 1.3.

Схему розташування елементів стенда – на рис. 1.4. Структурна схема плати розширення наведена на рис. 1.5. Схема розташування елементів плати розширення – на рис. 1.6. Рекомендується також розглянути повну принципову схему стенда (додаток А).

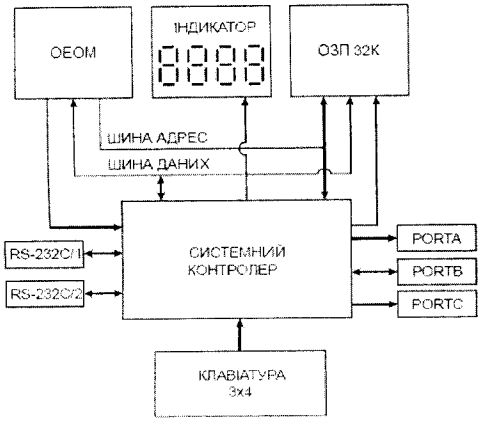


Рисунок 1.3 – Структурна схема стенда

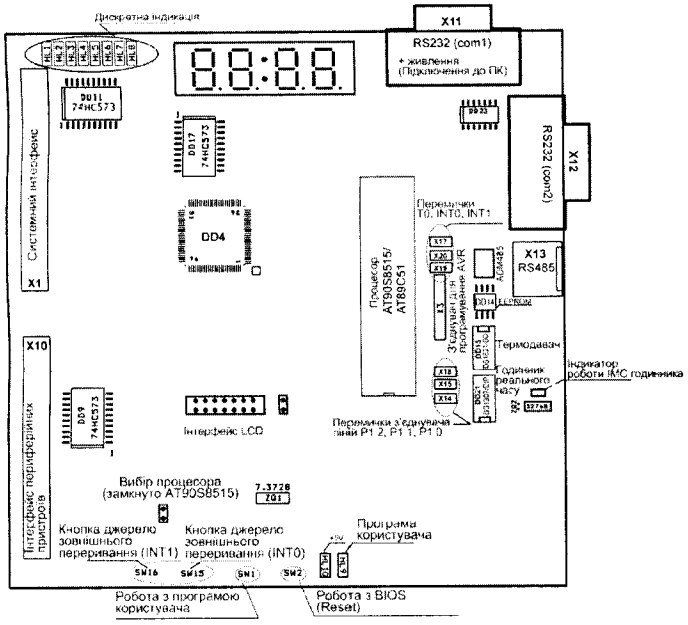


Рисунок 1.4 – Схема розміщення елементів лабораторного макета

Інтерфейси, що використовуються у схемі розміщення елементів лабораторного макета (див. рис. 1.4):

X1 – Системний інтерфейс з повним адресним простором;

X10 – Інтерфейс розширення для підключення зовнішніх пристроїв з використанням паралельного інтерфейсу;

X11 – Інтерфейс послідовного порта COM1 для зв'язку стенда з PC;

X12 – Інтерфейс послідовного порта COM2 для зв'язку стенда з іншими пристроями, які мають стандартний порт RS232C;

X3 – Інтерфейс програмування AVR;

X14, X15 – Перемичка підключення пристроїв шини I^2C до процесора.

Плата розширення EV8031/AN (рис. 1.5) містить:

- 8-розрядний інтегральний ЦАП (AD7801);
- 4-розрядну динамічну індикацію;
- світлодіодну матрицю 5×7;
- компаратор напруги;
- генератор із змінюваною частотою;
- резистор для зміни величини аналогового сигналу;
- динамік;
- BNC-роз'єм.

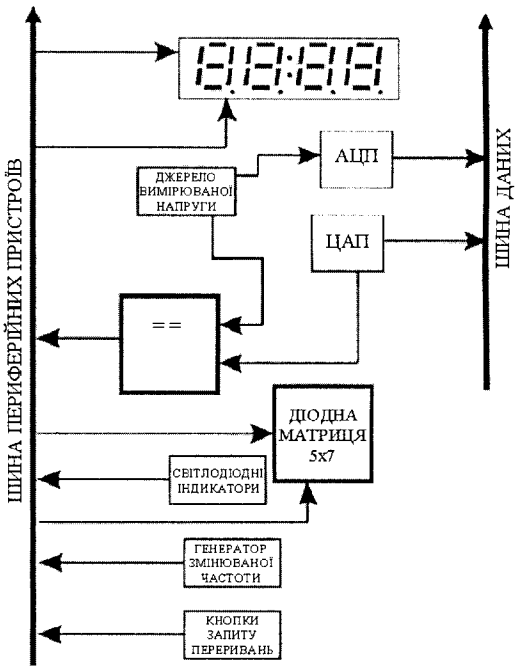


Рисунок 1.5 – Структурна схема плати розширення EV8031/AN

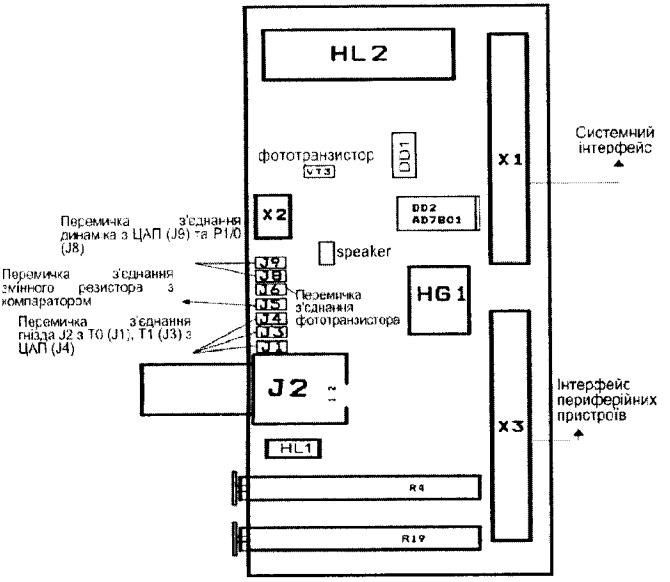


Рисунок 1.6 – Схема розміщення елементів плати розширення EV8031/AN

Перелік інтегральних схем, а також їх аналоги, що використовуються в стенді, наведено в табл. 1.1.

Таблиця 1.1 – Перелік комплектувальних мікросхем

Позначення на схемі	Позначення (зарубіжне)	Аналог (найближчий)	Короткий опис ІМС
DD1	AT90S8515	немає	Однокристальна ЕОМ
DD2, DD7, DD9, DD11, DD17	74HC573N	1533 ИР33	8-розрядний регістр
DD3	62256	немає (К537РУ17)	Статичний ОЗП 32К
DD4	EPM7128STC100	немає	Програмована логічна схема
DD6, DD18	74HC04	К1564ЛН2	6 КМОН інверторів
DD8, DD10	1489	К559ИП20	Перетворювач рівня RS-232C
DD12	ADM485	невідомий	Перетворювач рівня RS-485
DD14	AT29C02	немає	EEPROM 2k

Кожен вузол стенда (індикатори, клавіатура та ін.) має свій регістр пам'яті – встановлені на стенді інтегральні мікросхеми 74HC573N. У цьому регістрі зберігається інформація, що її мікроконтролер надсилає до певного вузла, або ж зчитує з нього. Операціями передачі даних між вузлами стенда і мікроконтролером керує системний контролер.

Тому звернення до індикаторів, клавіатури, світлодіодної матриці відбувається як звернення до комірок зовнішньої пам'яті. Потрібно пам'ятати, що для підтримки зовнішньої пам'яті необхідно записати «1» у біт дозволу SRE регістра керування мікроконтролером MCUCR. Мовою С це матиме такий вигляд: MCUCR = (1<<SRE). Після такої операції можна звертатись до усіх вузлів стенда та працювати з ними.

Звичайно, це не стосується кнопок SW15 і SW16, які напряму підключені до порту D мікроконтролера, а саме – PIND2 та PIND3. Також кнопки SW15 і SW16 можуть використовуватись для створення запитів на зовнішні переривання INT0 INT1. Для цього необхідно дозволити переривання INT0, INT1 за допомогою регістру керування зовнішніми перериваннями GICR, записавши «1» в його біти INT0, INT1.

Динамік (Buzzer), що знаходиться на платі розширення EV8031/AN, напряму підключений до виводу мікроконтролера PINB7. Також можна з'єднати динамік із виходом ЦАП, розімкнувши перемичку «J9» на платі розширення, яка з'єднує динамік з виводом мікроконтролера PINB7, та замкнувши перемичку «J8», що з'єднує його з виходом ЦАП.

Також для роботи з платою розширення використовуються віртуальні порти, з якими працюють як з комірками пам'яті: порт pA має адресу 0×8000, порт pB – адреса 0×8001 та порт pC – адреса 0×8002.

Таблиця 1.2 – Карта портів введення/виведення стенда

Адреса	Тип циклу	B7	B6	B5	B4	B3	B2	B1	B0	Назва
Порти периферійних пристроїв										
8××0	Запис	[Порт A]								PA_REG
8××1	Запис	[Порт B]								PB_REG
8××2	Запис	[Порт C]								PC_REG
8××3	Запис	×	×	×	×	×	TRISC	×	×	TRIS
LCD										
8××4	Запис	Регістр команд LCD індикатора								LCD_CMD
8××5	Запис	Регістр даних LCD індикатора								LCD_DATA
Послідовний порт										
9×××	Читання	CTS	DSR	DCD	RI	KL3	KL2	KL1	KL0	US_REG
C××0	Запис	×	×	×	×	DTR	RTS	CFG1	CFG0	UC_DATA
Індикатор та світлодіоди										
A××0	Запис	[Регістр індикатора 0]								DISPLAY[0]
A××1	Запис	[Регістр індикатора 1]								DISPLAY[1]
A××2	Запис	зарезервовано								DISPLAY[2]
A××3	Запис	зарезервовано								DISPLAY[3]
A××4	Запис	DP3	DP2	DP1	DP0	BL3	BL2	BL1	BL0	DC_REG
A××5	Запис	зарезервовано								EDC_REG
A××6	Запис	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0	LED_REG
Керування роботою										
A××7	Запис	×	×	×	×	×	×	×	RUN	SYS_CTL
Сумісні регістри										
B××0	Запис	[Регістр індикатора 1]								DISPLAYB

Програма завантажувач знаходиться у Flash-пам'яті мікроконтролера, вона проводить ініціалізацію послідовного прийомопередавача ОЕОМ (DD1), перевіряє наявність і місткість пам'яті даних.

При надходженні даних з послідовного порту персонального комп'ютера в послідовний порт (з'єднувач X11) стенда ОЕОМ записує їх до ОЗП, відведеного під пам'ять програм. Сигнали управління: PME, WR, RD, ALE, що формуються процесором і необхідні для звернення до пам'яті даних, надходять через системний контролер. Після прийняття останнього байта завантажувач формує сигнал запуску програми записом керувального коду в системний контролер.

Кнопка SW2 потрібна для формування сигналу скидання на вході RESET процесора, тобто переведення стенда в режим завантаження і очікування прийому даних з послідовного порту. Процесор готовий приймати дані в пам'ять даних.

Кнопка SW1 потрібна для перезапуску завантаженої з ПК програми що знаходиться в пам'яті програм (DD3). При натисканні кнопки SW1 спалахує світлодіод HL9. При цьому можливий новий запис програми у стенд з персонального комп'ютера. При передачі даних з персонального комп'ютера в стенд комп'ютер на лінії RI послідовного порту формує сигнал, який через системний контролер скидає процесор, як і кнопка SW2.

1.2 Робота зі стендом

У цьому підрозділі буде розглянуто один з варіантів програмування стенда на AVR-мікроконтролері.

1. Встановити програми AVR Studio, AVRreal, драйвер порту LINX_IO_NT_Port та драйвер на стенд. Усе програмне забезпечення поставляється на компакт-диску в комплекті зі стендом.

2. На персональному комп'ютері завантажити програму AVR Studio.

3. В стартовому діалоговому вікні натиснути кнопку «Новий проект». Якщо стартове діалогове вікно вимкнено, оберіть пункт меню «Проект» - «Новий проект».

4. Оберіть тип проекту Atmel AVR Assembler або AVR GCC залежно від того, якій мові програмування ви надаєте перевагу.

5. Введіть назву проекту у відповідному полі, оберіть розташування (бажано, щоб шлях містив тільки латинські літери). Якщо встановити прапорець «створити папку», то всі файли проекту міститимуться в окремій папці з іменем назви проекту. Натисніть «Далі».

6. Оберіть мікроконтролер ATmega8515 та натисніть «Фініш».

7. Наберіть програму у вікні редактора коду.

8. Натисніть кнопку «build» на панелі завдань або пункті меню «build», або кнопку F7 на клавіатурі.

9. Якщо є помилки, то інформація про них виведеться в вікні «build» внизу екрана. Їх необхідно виправити і скомпілювати проект заново. Коли проект успішно скомпілюється, то створиться двійковий hex-файл. Цей файл і потрібно запрограмувати в мікроконтролер стенда.

Виконуваний код завантажується у стенд за допомогою програми AVREAL32, яка запускається BAT-файлом «!send.bat». У файлі «!send.bat» вказано шлях до файлу «avreal32.exe», марку мікроконтролера, опції компіляції, а також ім'я HEX-файлу, що знаходиться в тому самому каталозі. Там знаходяться такі інструкції:

@c:\avr\avreal\avreal32 +MEGA8515 -p1 -e -w -v -o1000 -c main.hex.
@pause

10. Після того, як стенд підключено до LPT-порту комп'ютера за допомогою кабелю, запускається BAT-файл і код програми заноситься у пам'ять програм мікроконтролера.

Для тестування стенда виконати перших п'ять кроків, наведених вище. У вікні редактора коду в AVR Studio ввести такий код:

```
.include «m8515def.inc»      ;підключення файлу визначень
.def temp = R16              ;присвоєння імені регістру R16
ldi temp, 1<<SRE             ;дозвіл роботи з зовнішньою ;пам'яттю
out MCUCR, temp              ;вивести число 1234 на індикатор
ldi temp, $12
sts $A000, temp
ldi temp, $34
sts $A001, temp
ldi temp, $00                ;запалити всі розряди
sts $A004, temp
end: rjmp end                ;кінець
```

Виконати кроки 7–10, що наведені вище.

1.3 Віртуальний стенд EV8031/AVR

У програмному середовищі Proteus VSM віртуальний стенд реалізований у вигляді схеми, що складається з 5 аркушів [16]. На перший аркуш (рис. 1.7) винесені: мікроконтролер, статичні та динамічні індикатори, клавіатура, кнопки, 8 світлодіодів, світлодіодна матриця, вихід ЦАП. Це дозволяє оперативно аналізувати роботу мікропроцесорної системи та програмного забезпечення. Більш детально дізнатись про роботу з програмним середовищем Proteus VSM можна у [7–9].

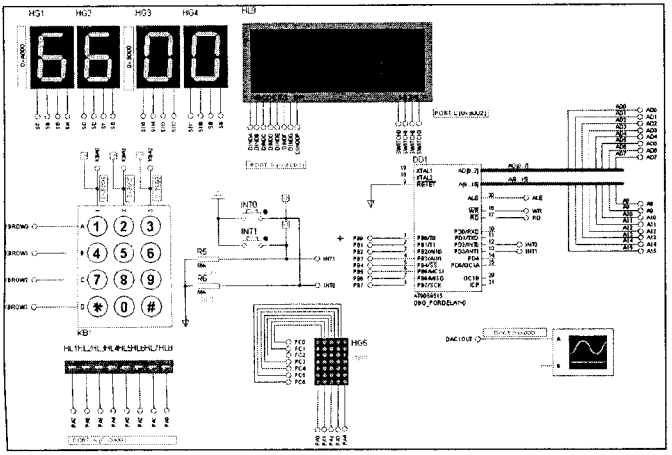


Рисунок 1.7 – Фрагмент віртуального стенда
у середовищі Proteus VSM

Статична індикація реалізована на 4-х статичних семисегментних індикаторах (HG1-HG4) з вбудованим дешифратором, розділених на дві частини. Ліва частина (HG1-HG2) має адресу 0×A000, права (HG3-HG4) – 0×B000. Для виведення інформації на індикатор необхідно записати потрібне значення за адресою 0×A000 або 0×B000.

Робота динамічної індикації реалізована через програмований периферійний інтерфейс 8255A. Мікросхема має 3 порти введення–виведення: A, B, C за адресами 0×8000, 0×8001, 0×80002 та регістр керувального слова за адресою 0×8003, що задає роботу мікросхеми. Для ініціалізації роботи виведення достатньо перед початком звертання до портів завантажити до регістра керувального слова значення 0×80.

Світлодіодна лінійка (HL1–HL8) під'єднана до порту A. Виведення інформації на семисегментний чотирирозрядний динамічний індикатор (HL9) забезпечується портами B і C. Порт B містить значення інформації, що виводиться, порт C – обирає необхідний розряд.

Світлодіодна матриця 7×5 (HG5) керується портами C та A. Логічна одиниця у відповідному розряді порту A вибирає стовпець матриці, а порт C – значення стовпця. Пікселі світяться при значенні логічного нуля.

Зчитування інформації з матричної клавіатури забезпечується зчитуванням інформації з відповідної комірки зовнішньої пам'яті (див. табл. 1.2). Ознака натиснення клавіші – логічний нуль у першій тетраді зчитаного байта.

Дана схема не має генераторів частоти, як на платі розширення EV8031/AN, але можна використати віртуальні генератори частоти.

Для завантаження HEX-файлу в мікроконтролер необхідно відкрити вікно властивостей мікроконтролера, де у полі Program File вказати шлях до файлу, як показано на рис. 1.8.

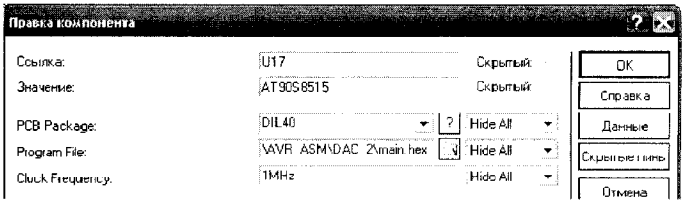


Рисунок 1.8 – Вікно властивостей мікроконтролера

Proteus ISIS має також вбудований компілятор мови Assembler та дозволяє створювати HEX-файли. Для цього необхідно додати до проекту файл з лістингом програми за допомогою меню «Исходник», де обрати опцію «Добавить/Удалить файлы исходника». При цьому відкриється вікно, показане на рис. 1.9, де необхідно вказати шлях до файлу, а також обрати необхідний компілятор, в даному разі – це AVRASM2.

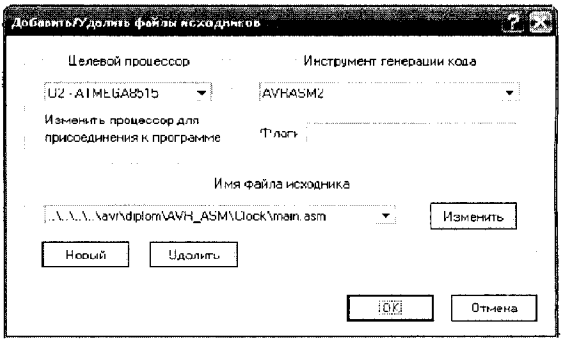


Рисунок 1.9 – Підключення лістингу

Після цього у тому ж меню обирається опція «Компоновать все» і створюється файл з розширенням .HEX, який тепер можна завантажити в мікроконтролер.

1.4 Архітектура мікроконтролера ATmega8515

Мікроконтролер (МК) ATmega8515 має AVR RISC-архітектуру, розроблену виробником – корпорацією Atmel, яка забезпечує йому високу продуктивність та низьку споживану потужність [5]. Дана архітектура передбачає виконання більшості команд за один такт тактової частоти, яка може досягати 16 МГц. Це означає, що при однаковій тактовій частоті мікроконтролери AVR у 12 разів швидші за мікроконтролери з CISC-архітектурою, наприклад, серії i8051 (AT89S52).

Блок-схема архітектури AVR показана на рис. 1.10.

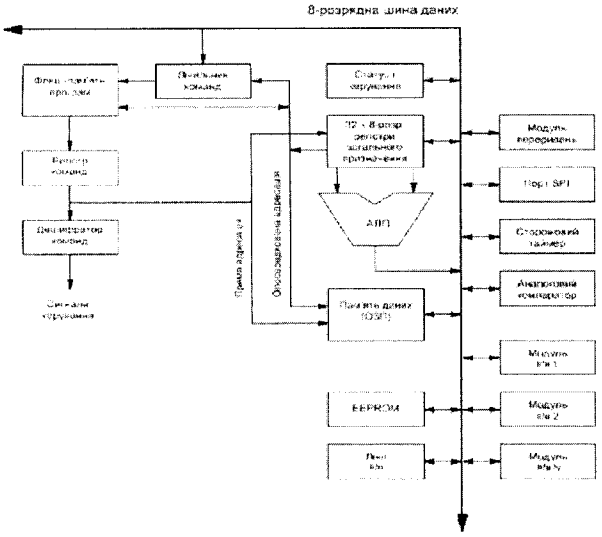


Рисунок 1.10 – Блок-схема архітектури AVR

Як видно, вона передбачає окрему пам'ять команд і даних. Іншою важливою особливістю даної архітектури є те, що вона розроблялась з орієнтацією на вико-ристання мов високого рівня при написанні програм для мікроконтролерів, зокрема мови С. Ці та багато інших переваг дозволяють застосовувати мікроконтролери AVR для побудови найрізноманітніших МПС, а невисока вартість дозволяє використовувати їх для розробки простих та дешевих пристроїв.

Що ж стосується МК ATmega8515, то це економічний 8-розрядний мікроконтролер. AVR-ядро об'єднує велику кількість інструкцій з 32 регістрами загального призначення, які безпосередньо підключені до арифметико-логічного пристрою (АЛП). Серед структурних особливостей – 35 ліній введення–виведення, наявність двох універсальних таймерів-лічильників, внутрішні та зовнішні запити на переривання, аналоговий компаратор.

AVR ядро об'єднує широкий набір інструкцій (130) з 32 робочими регістрами загального призначення. Усі 32 регістри безпосередньо підключені до АЛП, що дозволяє вказувати два регістри в одній інструкції і виконати її за один цикл. ATmega8515 має енергонезалежну пам'ять програм і даних. 8 кБайтів внутрішньої схемної програмованої флеш-пам'яті з можливістю самозапису та довговічністю 10000 циклів «запис–стирання». EEPROM має об'єм 512 байтів та довговічністю 100000 циклів «запис-стирання». Внутрішній статичний ОЗП має об'єм 512 байт. Є можливість організації зовнішньої області пам'яті розміром до 64 кБайтів та програмування бітів захисту програмного забезпечення.

Периферійні пристрої:

1. Один 8-розрядний таймер-лічильник з окремим переддільником і режимом компаратора.

2. Один 16-розрядний таймер-лічильник з окремим переддільником, режимом компаратора і режимом захоплення фронтів.

3. Три канали ШІМ (широтно-імпульсна модуляція).

4. Програмований послідовний пристрій синхронної або асинхронної прийомопередачі (ПСАПП).

5. Послідовний інтерфейс SPI з режимами: головний і підпорядкований.

6. Програмований сторожовий таймер з окремим вбудованим генератором.

7. Вбудований аналоговий компаратор.

Спеціальні функції мікроконтролера:

1. Скидання при подачі живлення і програмований супервізор живлення.

2. Вбудований RC-генератор, що калібрується.

3. Внутрішні і зовнішні джерела запитів на переривання.

4. Три режими управління енергоспоживанням: холостий хід (Idle), знижене споживання (Power-down) і черговий (Standby).

Напруга живлення: 2.7–5.5 В для ATmega8515L та 4.5–5.5 В для ATmega8515. Робоча частота: 0–8 МГц для ATmega8515L та 0–16 МГц для ATmega8515.

На рис. 1.11 наведена архітектура мікроконтролера ATmega8515.

Опис виводів схеми:

VCC – Напруга джерела живлення;

GND – Земля.

Port A (PA7...PA0) – Порт A – 8-розрядний двосторонній порт введення/виведення з внутрішніми резисторами підтягування (підібрані для кожного біта). Вихідні буфери порту A мають симетричні провідні характеристики як стоку, так і витоку. Коли контакти PA0–PA7 використовуються як входи, вони будуть джерелом струму, якщо внутрішні підтягувальні резистори включені. Виводи порту A мають 3 стани, коли умова скидання стає активною, навіть якщо немає синхроімпульсів. Порт A також обслуговує функції різних спеціальних особливостей ATmega8515.

Port B (PB7...PB0) – Порт B – 8-розрядний двосторонній порт введення/виведення з внутрішніми резисторами підтяжки (підібрані для кожного біта).

Port C (PC7...PC0) – Порт C – 8-розрядний двосторонній порт введення/виведення з внутрішніми резисторами підтягування (підібрані для кожного біта).

Port D (PD7.PD0) – Порт D – 8-розрядний двосторонній порт введення/виведення з внутрішніми резисторами підтягування (підібрані для кожного біта).

Port E(PE2.PE0) – Порт E – 3-розрядний двосторонній введення/виведення з внутрішніми резисторами підтягування (підібрані для кожного біта).

$\overline{\text{RESET}}$ – вхід скидання. Низький рівень на цьому виводі тривалістю більше, ніж мінімальна довжина імпульсу генеруватиме скидання, навіть якщо відсутній синхроімпульс.

XTAL1 – вхід інвертувального підсилювача осцилятора та вхід до внутрішньої схеми синхронізації.

XTAL2 – вихід інвертувального підсилювача.

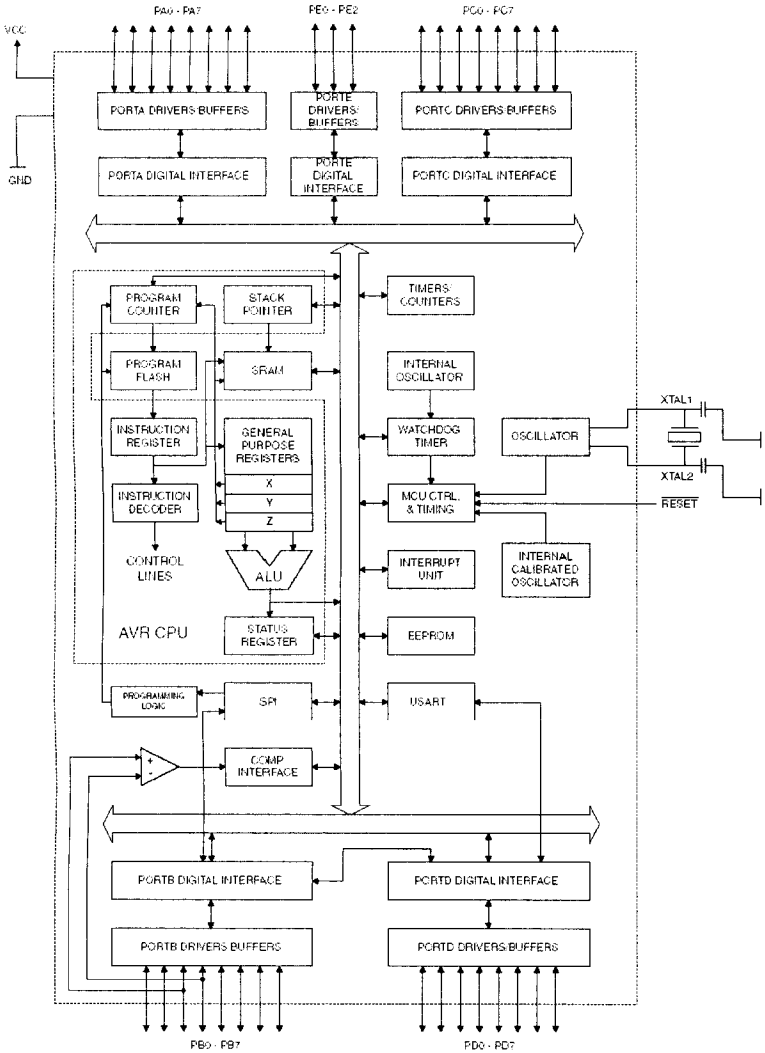


Рисунок 1.11 – Архітектура мікроконтролера ATmega8515

2 ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR

Процес написання програм для МК AVR, як і для будь-яких інших МК, складається з декількох етапів:

– підготовка вихідного тексту програми будь-якою мовою програмування;

– компіляція програми;

– налагодження й тестування програми;

– остаточне програмування й підготовка до серійного виробництва.

Мікропрограма пристрою повинна бути написана однією з мов програмування. На даний час для МК AVR існує декілька мов програмування, а також різних засобів підтримки розробки, що використовують одну мову, але різняться за функціональністю.

На кожному з етапів необхідне застосування спеціальних програмних й апаратних засобів. Варто відзначити, що базовий набір програмного забезпечення (компілятор асемблера, ПЗ для програмування) поширюється фірмою Atmel безкоштовно. Однак за досить довгий період часу, що пройшов з моменту появи цих МК, з'явилася велика кількість програмного забезпечення сторонніх виробників.

Вибір компілятора асемблера є найменш принциповим питанням, оскільки сам процес компіляції (перетворення вихідного тексту програми в машинний код) виконується досить однозначно. Власне, всі відмінності між асемблерами полягають у можливостях процесора, що обробляє макрокоманди, наявності текстового редактора або середовища розробки й типу операційної системи, що підтримується.

Досить вдалим вибором при розробці програмного забезпечення для МК сім'ї AVR може служити інтегроване середовище розробки AVR Studio фірми Atmel [2, 5, 11], що містить у собі текстовий редактор з підсвічуванням синтаксису, компілятор асемблера, симулятор, налагоджувач й інтерфейс із апаратними емуляторами.

Останнім часом усе популярнішим стає використання компіляторів мов високого рівня при написанні програм для мікропроцесорних систем. Найбільше поширення при цьому одержали компілятори мови С, оскільки в цій мові найбільш просто реалізуються всі необхідні можливості з керування апаратними засобами МК.

Для написання коду програми можна використовувати звичайний текстовий редактор операційної системи Windows – «Блокнот» чи інші. Також існують спеціально розроблені для програмування текстові редактори з додатковими функціями, наприклад, з підсвічуванням синтаксису, що полегшує роботу програмісту.

Звичайно, набагато зручніше розробляти програми у спеціальному середовищі розробки, яке, окрім зручного текстового редактора та компілятора, може містити також відлагоджувач (Debugger), за допомогою якого

можна виявити синтактичні помилки у програмі та помилки, які можуть виникнути під час виконання програми. В даному випадку мікроконтролери AVR мають ще одну перевагу, адже компанія Atmel безкоштовно розповсюджує середовище розробки AVR Studio, завантажити яке можна з офіційного сайту компанії.

Розглянемо поширені середовища розробки програм для МК AVR. Одним із розповсюджених середовищ є AVR Studio, що його підтримує більшість мікроконтролерів AVR, воно має вбудований відлагоджувач. Також програма підтримує багато моделей програматорів і дозволяє програмувати мікроконтролери AVR одразу після написання програми. Робоче вікно AVR Studio зображено на рис. 2.1.

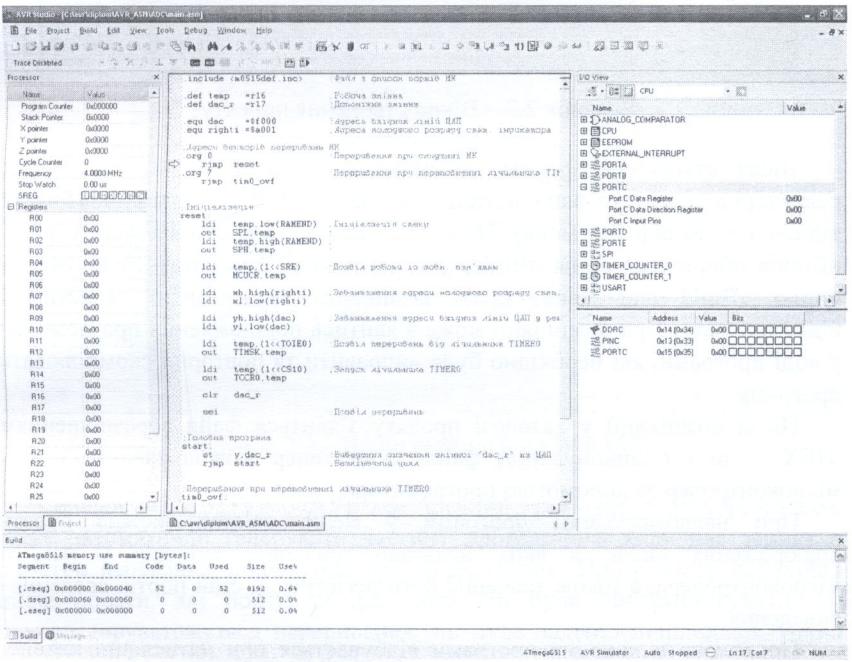


Рисунок 2.1 – Вікно середовища AVR Studio

AVR Studio дозволяє писати програми мовою Assembler, а також мовою C. Однак для цього необхідно встановити на комп'ютер бібліотеки середовища розробки WinAVR.

Для створення проекту у AVR Studio необхідно обрати меню Project=>New Profect. При цьому відкриється діалогове вікно створення проекту, яке показане на рис. 2.2, де можна обрати мову програмування та модель мікроконтролера.

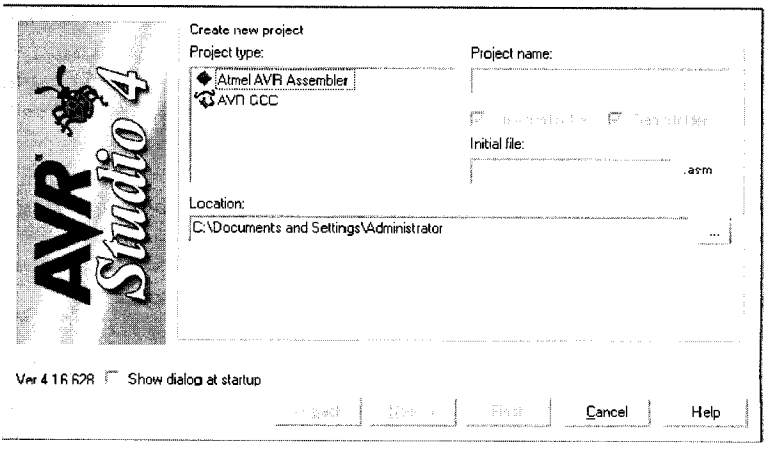


Рисунок 2.2 – Вікно створення проекту

Після створення проекту автоматично у вбудованому текстовому редакторі відкриється файл лістингу програми (з розширенням .ASM чи .C, відповідно до обраної мови). Після написання програмного коду у меню «Build» обирається опція «Build» для компіляції програми. Якщо обрати опцію «Build and Run», після компіляції автоматично запуститься відлагоджувач. При компіляції може з'явитись повідомлення про помилки у коді програми, які необхідно буде виправити та повторно скомпілювати програму.

Після компіляції у каталозі проекту з'явиться файл з розширенням .HEX – це і є виконуваний файл, який тепер можна завантажити у мікроконтролер за допомогою програматора.

При налагоджуванні програми у меню «View» можна обрати відображення вікон, в яких показано значення, записані у пам'яті мікроконтролера, а також значення його регістрів та стан портів введення–виведення.

Покрокове виконання програми відбувається при натисканні клавіші F10, також у меню «Debug» можна обрати автоматичне виконання.

Середовище WinAVR дозволяє створювати програми мовою С, для чого надає зручний текстовий редактор з підсвічуванням синтаксису та можливістю вставляння готових функціональних конструкцій, як показано на рис. 2.3. Для компіляції програми у меню «Tools» необхідно обрати опцію «Make All». За наявності підтримуваного програматора, за допомогою опції «Program» того ж меню можна запрограмувати мікроконтролер.

Вибір моделі мікроконтролера та опцій компіляції здійснюється редагуванням Make-файлу, який знаходиться у каталозі WinAVR\mfile.

Також можна запустити компілятор з командного рядка, вказавши опції компіляції та імена файлів. Звичайно, найпростіше створити виконуваний файл з розширенням .BAT, про що йтиметься далі.

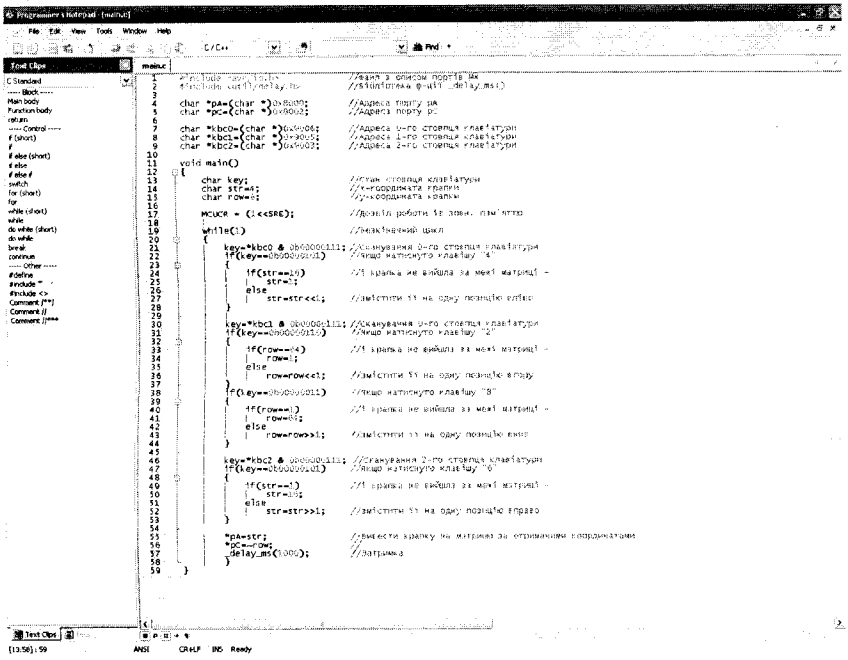


Рисунок 2.3 – Вікно середовища WinAVR

Середовище CodeVisionAVR має зручний текстовий редактор, а також може автоматично генерувати частину програмного коду для запису в регістри керування мікропроцесора налаштувань. Програмісту необхідно лише обрати, які порти та для чого він буде використовувати, чи використовуватимуться переривання, чи буде задіяно лічильники. Вікно CodeVisionAVR показано на рис. 2.4.

Проект створюється за допомогою меню File=>New=>Project. Після створення проекту до нього необхідно додати файл для тектсу програми: меню File=>New=>Sourse. Після цього необхідно обрати модель мікроконтролера, провести його налаштування, як показано на рис. 2.5. Після цього у відкритому вікні натискається кнопка (Generate program, save and exit), згенерований код автоматично вставляється у лістинг.

Для компіляції програми у меню «Project» обирається «Build». При цьому створюються всі необхідні файли проекту та HEX-файл, а також файл з розширенням .COFF, необхідний для налагоджування програми.

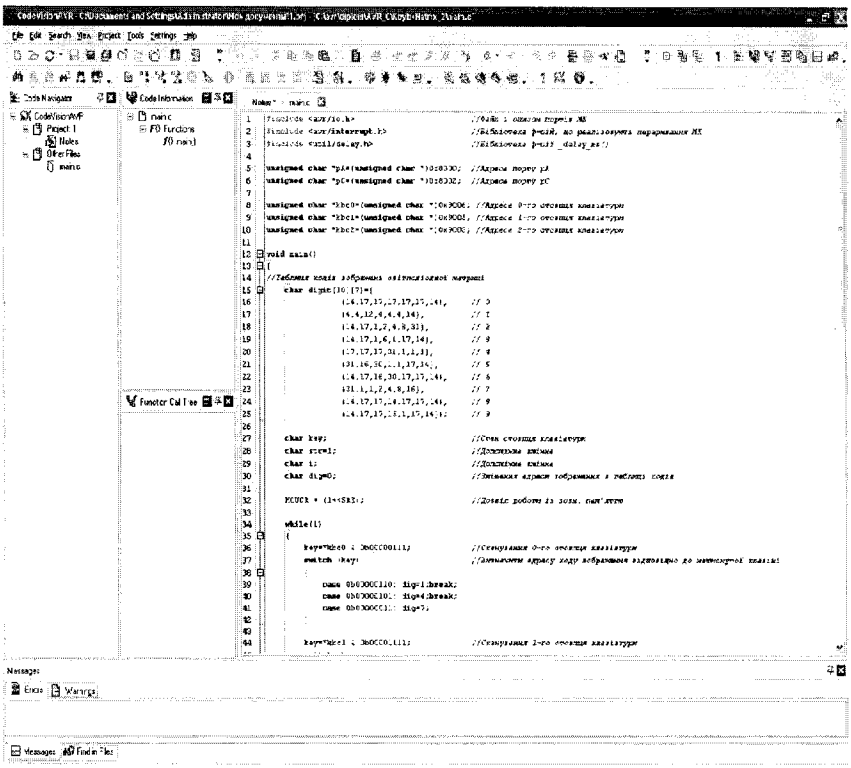


Рисунок 2.4 – Вікно середовища CodeVisionAVR

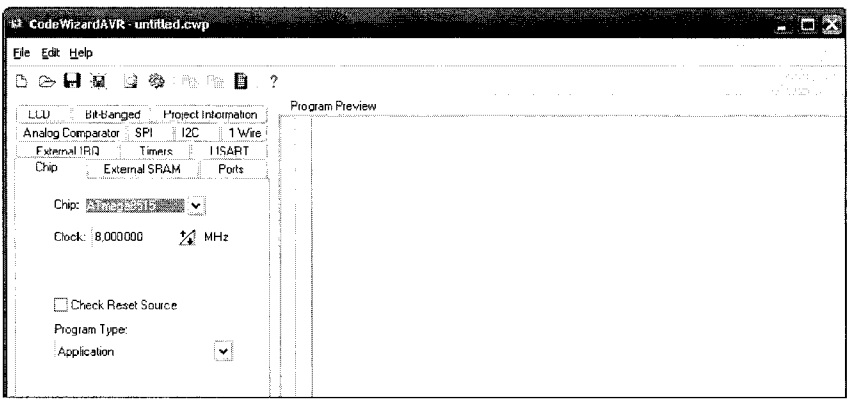


Рисунок 2.5 – Вікно налаштувань проекту

Недоліком CodeVisionAVR є відсутність вбудованого відладчика, але є можливість використати для цього AVR Studio. У меню Settings=>Debugger необхідно вказати шлях до файлу «AVRStudio.exe», якщо середовище AVR Studio встановлене не у каталог за замовчуванням. Потім у «Tools» обирається опція «Debugger», запускається AVR Studio, за допомогою якого необхідно відкрити COFF-файл, який знаходиться в каталозі проекту. Далі можна проводити налагодження програми за допомогою налагоджувача AVR Studio.

Ну і найпростіший спосіб створення HEX-файлу – за допомогою блокнота і компілятора, наприклад, Avrasm. Для цього лістинг програми на асемблері зберігається у файлі з розширенням .ASM, потім з командного рядка запускається компілятор Avrasm з необхідними опціями компіляції.

Найзручніше проводити розробку у середовищі AVR Studio, адже воно надає багато можливостей щодо дослідження роботи програми та має зручний інтерфейс, тому і буде використовуватись AVR Studio.

Програмний пакет AVR Studio розробляється компанією Atmel з 2004 року. Починаючи з версії 6.0 компанія змінила назву програми на Atmel Studio та додала можливість програмувати системи на базі ARM-архітектури. Раніше існував і фірмовий асемблер під ОС Windows (wavrasm.exe) від Atmel, що поєднував асемблер і редактор, проте, невдовзі після появи AVR Studio, відмовились від його подальшого розвитку.

Atmel Studio містить у собі такі інструмени, як: вбудований C/C++-компілятор, симулятор мікропроцесорної системи для налагодження програм, менеджер проектів, редактор коду, модуль внутрішньосхемного налагодження, а також інтерфейс командного рядка. Крім стандартних елементів, середовище підтримує низку таких інструментів: компілятор GCC та плагін AVR RTOS (операційної системи реального часу).

3 ЛАБОРАТОРНИЙ ПРАКТИКУМ

3.1 Вивчення стенда EV8031/ AVR, команд сім'ї AVR

Мета роботи: вивчення функціональних можливостей навчально-відлагоджувального стенда, архітектури мікроконтролера AT90S8515, внутрішньої структури та системи команд мікроконтролерів (МК) сім'ї AVR.

Навчальне завдання: вивчення архітектури мікроконтролера AT90S8515 та команд пересилання, арифметичних, логічних команд, команд переходів. Взаємодія внутрішніх вузлів мікроконтролера.

Порядок виконання лабораторної роботи

1) Вивчити структурну схему стенда, розподіл пам'яті, призначення вузлів. Вивчити структуру ЕОМ сім'ї AVR. Вивчити синтаксис команд пересилання, арифметичних команд, команд переходів.

2) Розробити алгоритм для виконання індивідуального завдання до початку лабораторного заняття (таблиця завдання 3.1).

3) Розробити програму для виконання індивідуального завдання до початку лабораторного заняття.

4) Ввести програму індивідуального завдання на ПК.

5) Вивчити програмно-налагоджувальні засоби (ПНЗ) для AVR.

6) За допомогою ПНЗ проаналізувати виконання індивідуальної програми.

7) Завантажити програму в стенд ЕОМ. Переконатися в правильному виконанні індивідуального завдання, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд ЕОМ.

8) Роздрукувати лістинг програми, що працює правильно.

9) Відповісти на контрольні питання викладача.

Контрольні питання

1. Час виконання команд (поняття такту, машинного циклу).
2. Типи команд. Формат команд. Команди прямої і непрямої адресації.
3. Команди зсуву. Арифметичні команди. Логічні команди.
4. Регістр ознак, команди викликають зміну регістра ознак.
5. Команди роботи зі стеком, послідовність дій (команд) під час роботи зі стеком.
6. Призначення внутрішніх вузлів ЕОМ.
7. Призначення і робота з внутрішньою пам'яттю даних ЕОМ.
8. Система переривання ОЕОМ. Призначення портів ЕОМ.
9. Фізичні характеристики вихідних сигналів ЕОМ.

Приклад виконання лабораторної роботи № 1

Записати в регістри загального призначення двійково-десяткове число у форматі 0X та X0. Суму чисел вивести на першому і другому знакомісцях статичної індикації.

```
.include «m8515def.inc»     ;підключення файлу заголовку
ldi R16, 1<<SRE             ;дозвіл роботи з зовнішньою
out MCUCR, R16              ;пам'яттю
ldi R16, $0C                ;запалити 1 і 2 знакомісця
sts $A004, R16              ;статичної індикації
ldi R20, $04                ;записати 04hв R20
ldi R21, $20                ;20h вR21
add R20, R21                ;додати
sts $A000, R20              ;вивести результат на індикатор
end: rjmp end               ;кінець
```

Таблиця 3.1 – Варіанти індивідуальних завдань

№ завдання	Текст індивідуального завдання
1	Занести в регістр R24 двійково-десяткове число 0X, в регістр R26 – двійково-десяткове число X0, суму чисел відобразити на першому і другому знакомісцях статичної індикації.
2	Занести в регістр R3 двійково-десяткове число XX відобразити його на першому і четвертому знакомісцях статичної індикації.
3	Занести в регістр R16 двійково-десяткове число, з частотою 2 Гц виводити це число на першому і другому знакомісцях статичної індикації.
4	Занести в R16 двійково-десяткове число XX, в регістр R25 – X0, число з R16 відобразити на першому і другому знакомісцях статичної індикації, число з R25 відобразити на третьому знакомісці статичної індикації.
5	Занести в регістр R21 двійково-десяткове число 0X, в регістр R25 – X0, суму чисел відобразити на другому і третьому знакомісцях статичної індикації.
6	Занести в комірку з адресою B0h оперативної пам'яті ОЕОМ двійково-десяткове число 0X, в регістр R23 – число X0, суму чисел відображати на другому і третьому знакомісцях статичної індикації з частотою 0,5 Гц.
7	Занести в регістр R20 двійково-десяткове число XX, поперемінно відображати молодшу і старшу тетраду на першому і четвертому знакомісцях статичної індикації з частотою 1 Гц.
8	Занести в R18 двійково-десяткове число X0, в регістр R1 – XX, число з R18 відображати на першому знакомісці статичної індикації з частотою 1 Гц, число з R1 відображати на третьому і четвертому знакомісцях статичної індикації з частотою 0,5 Гц.
9	Зчитати значення регістра SREG і відобразити його на третьому і четвертому знакомісцях статичної індикації.
10	Занести в регістр R24 двійково-десяткове число 0X, в регістр R23 – X0, суму чисел відобразити на другому і третьому знакомісцях статичної індикації з повільним (протягом 5 с) згасанням цього числа.
11	Занести в R0 двійково-десяткове число X0, в регістр R16 – 0X, суму чисел відобразити на першому і четвертому знакомісцях статичної індикації.
12	Занести в регістр R16 двійково-десяткове число 0X, в регістр R5 – X0, два розряди суми (десятки і одиниці) по черзі відображати на першому і другому знакомісцях статичної індикації.

Продовження табл. 3.1

№ завдання	Текст індивідуального завдання
13	Занести в регістр R1 двійково-десяткове число 0X, віднімаючи від числа одиницю, відображати на третьому знакомісці статичної індикації отримане значення до нуля з частотою 1 Гц.
14	Занести в регістр R23 двійково-десяткове число XX, в регістр R25 – XX, поперемінно відображати ці числа на першому і другому знакомісцях статичної індикації (R23) і на третьому і четвертому знакомісцях статичної індикації (R25).
15	Занести в регістр R0 двійково-десяткове число 0X, в регістр R21 – X0, число з R0 відобразити на четвертому знакомісці статичної індикації, число з регістра R21 відображати на другому знакомісці статичної індикації з частотою в 0,5 Гц.
*	Занести в регістр R0 число XX, вивести його на лівій парі знакомісць статичної індикації. Через 2 с вивести число XX, занесене в регістр R1. Через 2 с на правій парі знакомісць статичної індикації вивести різницю чисел, занесених в регістри R0 і R1.
*	Занести число 0X в регістр R0 і число XX – в регістр R1. Вивести значення R1 на правій парі знакомісць статичної індикації. Потім додавати значення R0 до значення R1 до тих пір, поки результат не досягне певного, заздалегідь заданого порогу. Здійснити перехід на початок програми. (Час затримки 0,5 с).
*	Занести число 0X в регістр R20 і число XX – в регістр R21. Суму вивести на правій парі знакомісць статичної індикації. Потім додавати значення R20 до значення R21 до тих пір, поки значення суми не досягне певного, заздалегідь заданого порогу. Потім від результату віднімати число, занесене в регістр R22, причому R22 не дорівнює R20. При досягненні нуля повернутися до додавання значення регістра R20.

* – завдання підвищеної складності. Видаються індивідуально викладачем.

3.2 Схеми відображення інформації

Мета роботи: вивчення схем динамічної та статичної індикації.

Навчальне завдання: розробка програм для мікроконтролерів сім'ї AVR для відображення цифрової інформації на пристроях динамічного і статичного типу та на одиничних індикаторах.

Порядок виконання лабораторної роботи

1) Вивчити принцип роботи різних методів відображення.

2) Розробити алгоритм для виконання індивідуального завдання до початку лабораторного заняття (табл. 3.2).

3) Розробити програму для виконання індивідуального завдання до початку лабораторного заняття з використанням підпрограм.

4) Ввести програму індивідуального завдання на персональному комп'ютері.

5) За допомогою ПНЗ проаналізувати виконання індивідуальної програми.

6) Завантажити програму в стенд. Переконатися в правильному виконанні індивідуального завдання, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд.

7) Роздрукувати лістинг програми, що правильно працює.

8) Відповісти на контрольні питання викладача.

Контрольні питання

1. Розрахунок часу регенерації для динамічного методу відображення.
2. Обґрунтування необхідності застосування різних методів відображення.
3. Схемотехнічні рішення для побудови схем відображення інформації.
4. Схеми включення одиничних індикаторів.
5. Включення рідкокристалічних індикаторів
6. Включення газорозрядних індикаторів.

Короткі теоретичні відомості

Практично кожен мікропроцесорний пристрій містить елементи індикації. Як індикатори в даний час найчастіше застосовуються світлодіоди. На ринку є величезний вибір світлодіодів найрізноманітніших видів і розмірів.

У мікропроцесорних пристроях світлові індикатори можуть служити для відображення різних режимів роботи: попередження про критичні ситуації, відображення ходу прийому сигналів керування тощо. Під'єднати окремий світлодіодний індикатор до МК дуже просто. На рис. 3.1 наведена схема під'єднання світлодіода безпосередньо до виводу порту МК.

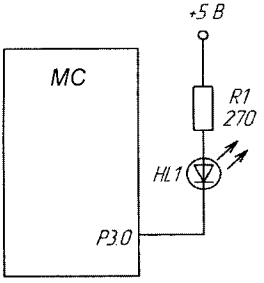


Рисунок 3.1 – Схема під'єднання світлодіодного індикатора

Усі вихідні каскади МК мають достатню навантажувальну здатність для того, щоб витримати під'єднання одного світлодіодного індикатора зі споживаним струмом у робочому режимі не більше 20 мА.

Для керування двома світлодіодами одним виводом у МК передбачено активні вихідні каскади, і для перемикання режиму роботи (введення або виведення) слугує спеціальний регістр. Таким чином, сигнал кожного виходу будь-якого порту може мати 3 значення – «0», «1» і високоімпедансний («Z») стан. Це дозволяє керувати двома світлодіодами за допомогою одного виводу (рис. 3.2).

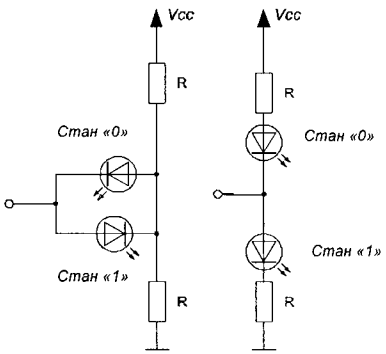


Рисунок 3.2 – Керування двома світлодіодами одним виводом МК

На стенді знаходиться дискретний індикатор – 8 світлодіодів, розміщених в ряд. Світлодіоди під'єднані до виходів регістра, як показано на схемі на рис. 3.3. Для виведення інформації на світлодіоди необхідно звертатись до комірки пам'яті за адресою 0×A006. При цьому двійкове число на індикаторі відображається у такому вигляді: 0-й розряд – на світлодіоді HL8, 7-й розряд – на світлодіоді HL1. Використання даного індикатора є дуже зручним при тестуванні програм на стенді, оскільки дає можливість відслідковувати значення певного регістра чи змінної.

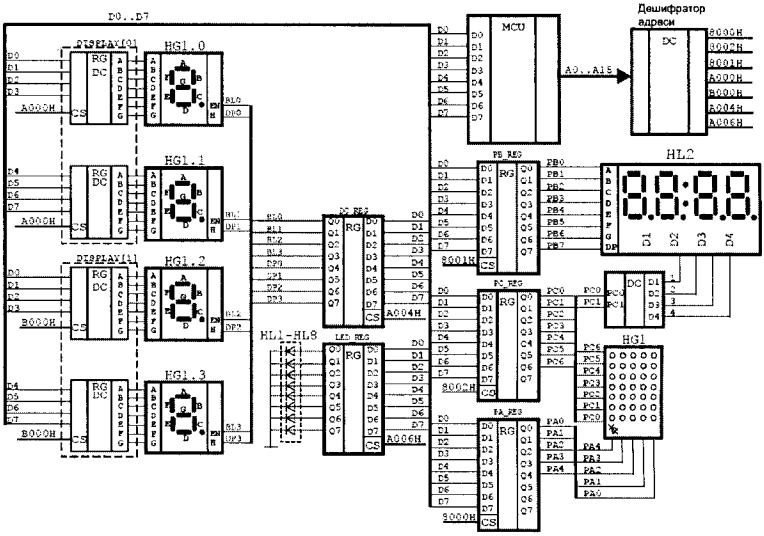


Рисунок 3.3 – Схема статичної та динамічної індикації стенда EV8031/AVR

На стенді присутній статичний 4-розрядний світлодіодний індикатор, який складається з двох окремих 2-розрядних індикаторів – відповідно молодшого і старшого розрядів статичного індикатора. Тобто, можна працювати окремо з лівим чи правим (старшим або молодшим) розрядами статичного індикатора, для доступу до яких необхідно звертатись до комірок пам'яті за адресами 0×A000 та 0×A001 відповідно.

Керування даним індикатором проводить системний контролер. Він організовує динамічну індикацію, тому програмісту необхідно просто вивести дані на необхідний розряд статичного індикатора – молодший або старший. Слід зауважити, що зображенням для розряду даного індикатора слугують шістнадцяткові числа від 00 до FF. Системний контролер автоматично перетворює дані, які виводяться на статичний індикатор, у шістнадцятковий код. Схема увімкнення індикатора показана на рис. 3.3.

Дуже часто МК використовується не тільки для керування роботою конструкції, але й для того, щоб повідомити що-небудь користувачеві і/або отримати від нього які-небудь вказівки про подальшу роботу. Наприклад, електронний годинник, крім власне відліків часу, повинен його ще відображати, а також дозволяти змінювати покази (встановлювати точний час). Якщо вся «інформація» зводиться до мигання парою світлодіодів, яких-небудь спеціальних зусиль з відображення інформації з боку розробника конструкції не вимагається, але якщо таких світлодіодів виявляється два-три десятки, тут вже потрібне застосування додаткових засобів (як апаратних, так і програмних). Як правило, в цьому випадку відображення інформації виконують у режимі динамічної індикації – це найбільш економний за кількістю використовуваних ліній спосіб.

Для організації динамічної індикації застосовується матриця, що складається з ліній рядків і ліній стовпців (рис. 3.4). На перетині стовпця і рядка матриці розташований індикаторний елемент – світлодіод. Для того, щоб запалити той або інший елемент, необхідно подати на матрицю не один, як у звичайних індикаторах, а два сигнали: логічна 1 на відповідному рядку і логічний 0 на відповідному стовпці матриці. Через односторонню провідність світлодіода кожна комбінація сигналів на входах рядків і стовпців однозначно вмикає рівно один індикаторний елемент.

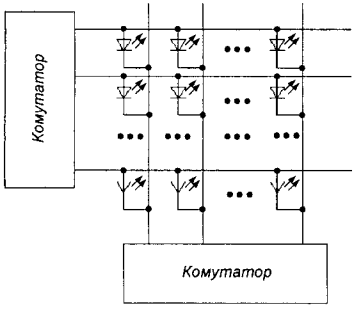


Рисунок 3.4 – Організація динамічної індикації

Головна перевага динамічної індикації – невелике число ліній керування: для матриці світлодіодів розміром N×N елементів потрібно всього 2N керівних сигналів. За таку економію, втім, доводиться платити – справа в тому, що при почерговому виведенні інформації на кожен світлодіод матриці його яскравість світіння буде в N^2 разів нижча, ніж при безпосередньому виведенні інформації на один світлодіод, що «окремо стоїть». Тому в пристроях, що використовують динамічну індикацію, виведення інформації здійснюється не на кожен світлодіод окремо, а на один рядок або на один стовпець повністю – у цьому випадку яскравість світіння світлодіодів падає тільки в N разів.

Плата розширення EV8031/AN містить динамічний 4-розрядний світлодіодній індикатор, схема під'єднання якого показана на рис. 3.3.

Доступ до даного індикатора здійснюється за допомогою віртуальних портів pB та pC. На порт pC виводиться номер розряду, який необхідно задіяти (0–3), а порт pB слугує для виведення зображення на розряд індикатора.

Матричні світлодіодні індикатори (МСІ) використовуються для відображення алфавітно-цифрової інформації. Кожен з таких МСІ, виконаний у вигляді інтегральної мікросхеми, є матрицею світлодіодів розмірністю $m \times n$, де n – число стовпців, m – число рядків матриці. Найбільшого поширення набули МСІ з розмірністю матриці 7×5 і 9×7 (рис. 3.5).

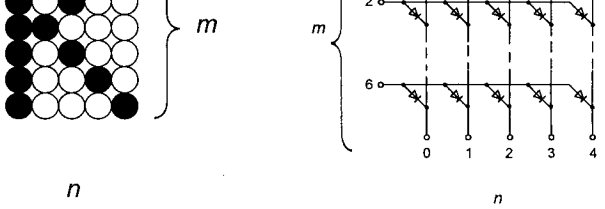


Рисунок 3.5 – Загальний вигляд та схема матричного індикатора

У кожному часовому такті збуджується строб-імпульс відповідного стовпця. У результаті відбувається відображення інформації у всіх елементах даного стовпця. Після кожного такту відбувається зсув інформації і в наступному часовому такті збуджується строб-імпульс у другому стовпці і так далі. За п'ять тактів відбувається передача повної інформації на матричний індикатор, після чого відбувається повторення передачі, якщо по шині введення даних не надійшла нова інформація. Часова діаграма формування букви М наведена на рис. 3.6.

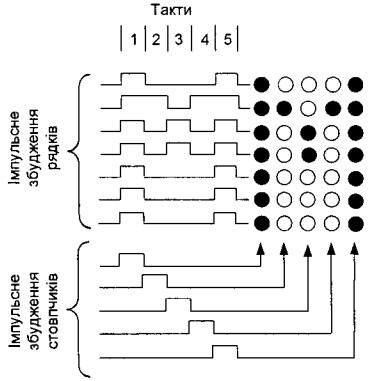


Рисунок 3.6 – Часова діаграма формування матричним індикатором літери М

На платі розширення EV8031/AN знаходиться світлодіодна матриця з розширенням 5×7 крапок-світлодіодів. Для роботи з даним індикатором необхідно використовувати віртуальний порт pA – для виведення значення рядка матриці – та порт pC – для виведення значення стовпця. Схема підключення матриці показана на рис. 3.3.

Приклад програми для статичної індикації
3 частотою 1 Гц відобразити на статичному індикаторі число 04

```
.include «m8515def.inc»          ;підключення файлу визначень
.def temp = R16                  ;присвоєння назв регістрам
.def del1 = R17                  ;temp для R16
.def del2 = R18                  ;del1 дляR17 – для затримки
.def del3 = R19                  ;del2 для R18 – для затримки
                                 ;del3 для R19 – для затримки
ldi temp, low(RAMEND)            ;ініціалізація стека
out SPL, temp
ldi temp, high(RAMEND)
out SPH, temp

ldi temp, 1<<SRE                 ;дозвіл роботи з зовнішнім
out MCUCR, temp                  ;адресним простором

ldi temp, $04                    ;вивести число 04h на індикатор
sts $B000, temp

ldi temp, $0F                    ;погасити всі розряди
sts $A004, temp                  статичного ;індикатора
loop:                            ;головний цикл
```

```
ldi temp, $03                ;запалити індикатор
sts $A004, temp
rcall delay                  ;затримка 0,5 с

ldi temp, $0F                ;погасити індикатор
sts $A004, temp
rcall delay                  ;затримка 0,5 с

rjmp loop                    ;на початок циклу

delay:                       ;підпрограма затримки
ldi del1, byte1(737280)      ;число для затримки
ldi del2, byte2(737280)      ;розраховується за формулою:
ldi del3, byte3(737280)      ;
del:                         ;N = FМК(Гц)*T(сек)/5 =
subi del1, 1                 ;= (7372800*1/2)/5 = 737280
sbci del2, 0                 ;
sbci del3, 0                 ;FМК – частота МК в герцах
brcc del                     ;T – тривалість затримки в
ret                          ;секундах
```

На світлодіодах HL1-HL8 запустити «вогник, що біжить», зменшуючи час затримки між засвічуванням світлодіодів до певного значення, після чого засвітити всі світлодіоди.

```
.include «m8515def.inc»      ;підключення файлу визначень
                             ;присвоєння назв регістрам
.def temp = R16              ;tempдля R16
.def del1 = R17              ;del1 дляR17 – для затримки
.def del2 = R18              ;del2 для R18 – для затримки
.def del3 = R19              ;del3 для R19 – для затримки
.def et_del = R20            ;максимальне значення затримки
.def line = R21              ;стан світлодіодів

ldi temp, low(RAMEND)        ;ініціалізація стека
out SPL, temp
ldi temp, high(RAMEND)
out SPH, temp

ldi temp, 1<<SRE             ;дозвіл роботи з зовнішнім
out MCUCR, temp              ;адресним простором

ldi temp, $0F                ;погасити всі розряди
sts $A004, temp              статичного ;індикатора
```

```
ldi line, 1                 ;стан світлодіодів – запалити
                            ;лівий світлодіод
ldi et_del, byte3(1474560)

loop:                       ;головний цикл
sts $A006, line             ;вивести line на світлодіоди
rcall delay                 ;затримка
dec et_del                  ;зменшити розмір затримки
breq end                    ;на вихід, якщо et_del = 0
lsl line                    ;зсув стану світлодіодів
brcc loop
ldi line, 1
rjmp loop                   ;на початок циклу

end:                        ;остання дія
ser line                    ;запалити всі світлодіоди
sts $A006, line
rjmp end                    ;кінець

delay:                      ;затримка
ldi del1, byte1(1474560)
ldi del2, byte2(1474560)
mov del3, et_del            ;старший байт (найзначиміший)
del:                        ;визначається числом, записаним
subi del1, 1                в ;et_del, що постійно
sbci del2, 0                зменшується
sbci del3, 0
brcc del
ret
```

Таблиця 3.2 – Варіанти індивідуальних завдань

№ завдання	Текст індивідуального завдання
1	Занести в регістр R1 число XX, віднімаючи від цього числа «1», відображати результат на динамічному індикаторі в молодшому розряді до нуля з частотою 0,5 Гц. Включаючи «вогник, що біжить» на HL1–HL8
2	Занести в R16 двійково-десяткове число X0, в R17 – XX, число з R16 відобразити на знакосинтезувальному індикаторі, число з R1 відображати на динамічному індикаторі в старшому розряді з частотою 0,5 Гц.
3	Включити в шаховому порядку світлодіоди HL1–HL8. Занести в регістр R19 в двійково-десяткове число 0X, в регістр R25 – X0, два розряди суми (десятки і одиниці) по черзі відображати на статичному індикаторі і на динамічному індикаторі з частотою 1 Гц.
4	Занести в R6 двійково-десяткове число XX, в R5 – двійково-десяткове число XX, в R0 – двійково-десяткове число XX, відображати ці числа: R5, R6 – на динамічному індикаторі, R0 – на статичному індикаторі.

Продовження табл. 3.2

№ завдання	Текст індивідуального завдання
5	Почергово відображати на знакоситезувальному індикаторі числа від 0 до 9 дублювати ці числа на динамічному індикаторі.
6	Занести в регістр R16 двійково-десяткове число 0X, в регістр R22 – X0, число з R16 відобразити на статичному індикаторі, число з регістра R22 відображати на динамічному індикаторі з частотою 0,6 Гц.
7	Занести в R0 двійково-десяткове число XX, в R1 – XX, молодші два розряди суми чисел відобразити на динамічному індикаторі, при цьому на знакосинтезувальному індикаторі здійснити плавне загоряння числа 5.
8	Занести в регістр R6 число XXH, перетворити його в двійково-десяткове число, відобразити його на динамічному індикаторі, відобразити на світлодіодах HL1–HL8 значення регістра R6 і його інверсний стан з частотою 1Гц.
9	Занести в регістр R17 двійково-десяткове число XX, в регістр R23 – XX, різницю чисел відобразити на динамічному індикаторі
10	Відобразити на знакосинтезувальному індикаторі букву У. Занести в R16 число XX, в R25 – X0, число з R16 відобразити на статичному індикаторі, число з R5 відобразити на динамічному індикаторі.
11	Занести в регістр R0 двійково-десяткове число XX, поперемінно відображати молодший і старший розряди на динамічному індикаторі з частотою 0,5 Гц.
12	Занести в регістр R22 двійково-десяткове число XX, в регістр R25 – XX, суму відобразити на динамічному індикаторі.
13	Занести в регістр R22 двійково-десяткове число, з частотою 2 Гц виводити це число на статичному індикаторі і одночасно на динамічному індикаторі.
14	Почергово включати світлодіоди HL1–HL8. Занести в клітинку з адресою 0080h оперативної пам'яті ОЕОМ двійково-десяткове число 0X, в регістр R3 – XXH, суму чисел відобразити в старшому розряді на динамічному індикаторі.
15	Занести в регістр R1 двійково-десяткове число 0X, в регістр R3 – XX, суму відобразити на динамічному індикаторі. Шістнадцяткове число відобразити на HL1–HL8.
*	По черзі засвічуючи світлодіоди HL1–HL8, на статичному індикаторі паралельно відображати кількість засвічених світлодіодів. Інтервал між засвічуванням 1 с.
*	Показати на динамічному індикаторі слово з чотирьох букв. Перша буква, з крайнього правого положення, пройшовши всі сегменти, залишається горіти в крайньому лівому положенні. Наступні букви, аналогічно пройшовши всі сегменти, залишаються за попередніми.

3.3 Система переривань. Опитування дискретних датчиків

Мета роботи: вивчення режимів роботи системи переривання мікроконтролерів AVR, програмна обробка дискретних сигналів.

Навчальне завдання: вивчення систем переривання, режимів введення дискретної інформації, розробка програм опитування сигналів від датчиків.

Порядок виконання лабораторної роботи

1. Вивчити систему переривання МК AVR, особливості опитування дискретних датчиків з механічними контактами.
2. Розробити алгоритм для виконання індивідуального завдання (табл. 3.2) до початку лабораторного заняття.
3. Розробити програму для виконання індивідуального завдання до початку лабораторного заняття.
4. Ввести програму індивідуального завдання на ПК.
5. За допомогою ПНЗ проаналізувати виконання індивідуальної програми.
6. Завантажити програму в стенд EV8031/AVR. Переконатися в правильному виконанні індивідуального завдання, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд EV8031/AVR.
7. Роздрукувати лістинг програми, що прцює правильно.
8. Відповісти на контрольні питання викладача.

Контрольні питання

1. Структура системи переривань мікроконтролерів AVR.
2. Призначення, приклади застосування системи переривань.
3. Регістри управління перериваннями в мікроконтролерах AVR.
4. Переривання від таймерів, послідовного приймача-передавача.
5. Апаратне усунення «тремтіння» контактів.
6. Програмне усунення «тремтіння» контактів.
7. Необхідність застосування апаратного або програмного усунення «тремтіння» контактів.

Короткі теоретичні відомості

Практично жоден мікропроцесорний пристрій не обходиться без кнопок і простих датчиків. За допомогою цих периферійних елементів в МПП надходить різна інформація, яка використовується для зміни алгоритму роботи програми. Схема підключення контактного датчика до МК наведена на рис. 3.7. У наведеному прикладі датчик підключений до лінії *PD0* порту *D* МК. Через цей вхід МК проводить зчитування стану датчика. Датчик можна підключити і до будь-якої іншої лінії будь-якого з портів МК.

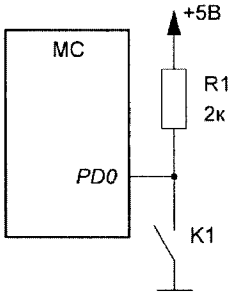


Рисунок 3.7 – Підключення контактного датчика до МК

У початковому стані контакти датчика розімкнені. На вхід МК через резистор R1 прикладається напруга від джерела живлення + 5 В. МК сприймає цю напругу як сигнал логічної одиниці. При спрацьовуванні датчика контакти замикаються і з'єднують вивід МК із загальною шиною. Тепер мікросхема сприймає вхідний рівень сигналу як логічний нуль. Резистор R1 при цьому служить струмообмежувальним елементом, запобігаючи короткому замиканню між шиною живлення і загальною. Деякі МК мають свої внутрішні резистори навантаження, які можуть замінити зовнішній резистор. Схема підключення декількох датчиків або кнопок до МК зображена на рис. 3.8.

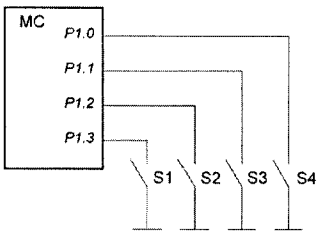


Рисунок 3.8 – Підключення кнопок або простих датчиків до МК

У схемі, що зображена на рис. 3.9, при натисканні однієї з клавіш змінюється постійна напруга на відповідному вході процесора, яка розпізнається процесором і дешифрується в певну команду. Ця напруга максимальна (приблизно 5 В), коли кнопки не натиснуті, і мінімальна (0 В) при натиснутій клавіші S1 [3, 10, 12, 13]. Блок-схема 12-клавішної клавіатури зі скануванням показана на рис. 3.10.

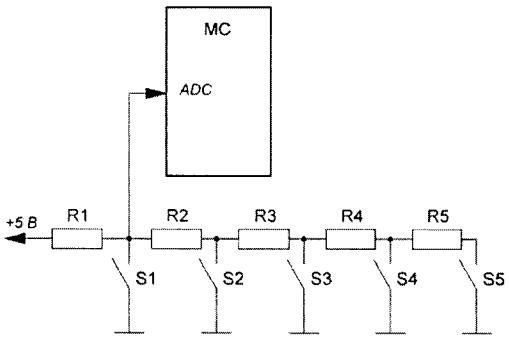


Рисунок 3.9 – Підключення кнопок зміною напруги
на аналоговому вході МК

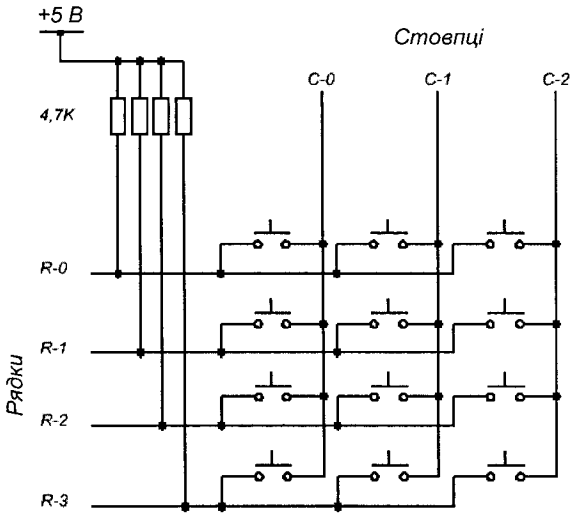


Рисунок 3.10 – Матрична клавіатура 4×3

Клавіші розташовані у вузлах матриці, у якої чотири лінії рядків і три лінії стовпців. На лінії стовпців по черзі подається від'ємний імпульс (логічний «0»). У цей момент перевіряється стан чотирьох ліній рядків. Якщо натиснутих клавіш немає, всі лінії рядків мають високий рівень (вони підключені до напруги +5 В через резистори). Якщо ж клавіша натискається, і на лінії стовпця, відповідного натиснутій клавіші, все ще нуль, то адекватна лінія рядка також дорівнює нулю. Знаючи номери стовпця і рядка, можна отримати позицію натиснутої клавіші.

Окрім розпізнавання положення натиснутої клавіші слід програмно захиститися від «тремтіння» контактів, тобто від впливу перехідних процесів (8–12 мс), а також від ситуацій, пов'язаних з одночасним натисканням декількох клавіш.

Для того, щоб запобігти протіканню небезпечних струмів при одночасному замиканні декількох клавіш у одному стовпці, в лініях R0...R3 встановлюють послідовно резистори або діоди. З цією ж метою можна використовувати й інший метод сканування, при якому всі неактивні горизонтальні шини, окрім шини нуля, що «біжить», програмно призначаються входами. Впливу перехідних процесів можна уникнути, якщо повторно зчитувати стан входів матриці сканування з затримкою в 12 мс. Для усунення «тремтіння» на виході контактної пари встановлюють спеціальні формувачі. Приклад такого формувача, заснованого на принципі безпосереднього встановлення RS-тригера, наведено на рис. 3.11.

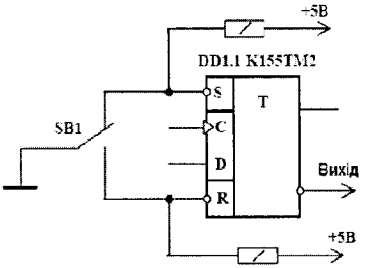


Рисунок 3.11 – Схема усунення «тремтіння» контактів RS-тригером

Обмін за перериваннями використовується тоді, коли необхідна реакція мікропроцесорної системи на якусь зовнішню подію, на появу зовнішнього сигналу. Організувати реакцію на зовнішню подію можна так:

– за допомогою постійного програмного контролю факту появи події (так званий метод опитування прапорця або polling);

– за допомогою переривання, тобто примусового переведення процесора з виконання поточної програми на виконання екстрено необхідної програми.

У мікроконтролерів AVR кожне переривання має стандартну адресу вектора. В конкрет них моделей ці адреси різні. У табл. 3.3 наведено список векторів переривань для МК ATmega8515.

Таблиця 3.3 – Список векторів переривань для МК ATmega8515

№ джерела	Адреса	Джерело	Опис
1	$000	RESET	При ввімкненні живлення, при скиданні
2	$001	INT0	Зовнішній запит переривання 0
3	$002	INT1	Зовнішній запит переривання 1
4	$003	TIMER1 CAPT	Захоплення значення T/C1
5	$004	TIMER1 COMPA	Збігання значення T/C1 з регістром порівняння A
6	$005	TIMER1 COMPB	Збігання значення T/C1 з регістром порівняння B
7	$006	TIMER1 OVF	Переповнення T/C1
8	$007	TIMER0 OVF	Переповнення T/C0
9	$008	SPI, STC	Передачу по SPI завершено
10	$009	USART, RXC	Прийом по USART завершено
11	$00A	USART, UDRE	Регістр даних USART пустий
12	$00B	USART, TXC	Передачу по USART завершено
13	$00C	ANA_COMP	Переривання від аналогового компаратора
14	$00D	INT2	Зовнішній запит переривання 2
15	$00E	TIMER0 COMP	Збігання значення T/C0 з регістром порівняння.
16	$00F	EE_RDY	Готовність EEPROM
17	$010	SPM_RDY	Готовність Flash до запису

При виниканні переривання поточне значення вказівника команд записується в стек і виконання програми продовжується з вектора переривання, в якому слід розмістити перехід на обробник переривання (команда *rjmp*).

Для дозволу переривань потрібно встановити в регістрі *SREG* спеціальний прапорець «*i*». Для кожного переривання (крім переривання за скидом), потрібно встановити дозвіл в спеціальному для нього регістрі. Для таймерів-лічильників це регістр TIMSK, для зовнішніх запитів – GICR.

Всі ці регістри, їх адреси та призначення бітів можна подивитись в офіційній документації мікроконтролера.

При виниканні переривання біт «*i*» регістра *SREG* автоматично скидається. Для завершення оброблення і повернення в основну програму слід використовувати команду *reti*. Вона відновлює значення вказівника команд зі стека і встановлює прапорець «*i*».

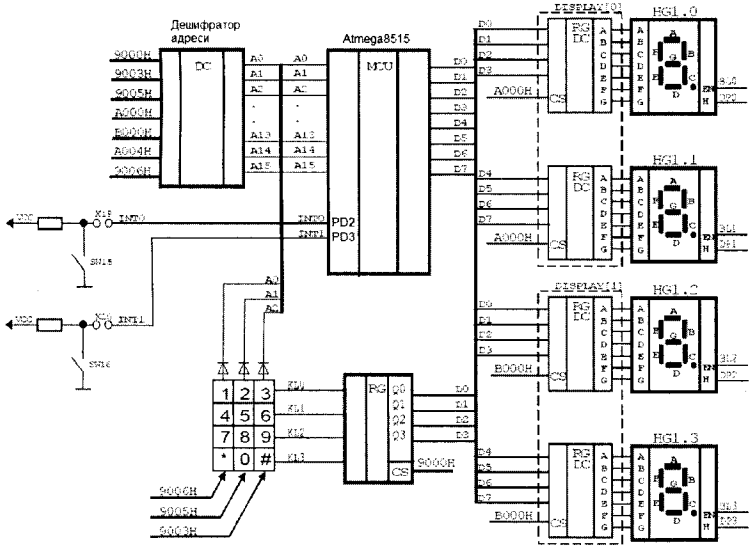


Рисунок 3.12 – Схема організації клавіатури та дискретних датчиків стенда EV8031/AVR

У складі стенда є дві окремі кнопки SW15, SW16 (рис. 3.12), які можуть перевірятись як програмно, так і за допомогою використання функцій переривань INT0, INT1 відповідно. У складі стенда є також матрична 3×4 клавіатура SW3–SW14. Клавіатура підключена до шини даних за допомогою мікросхеми буфера DD1 74245(АП6).

Опитування усієї клавіатури робиться за три рази (за один раз читається стан тільки одного стовпця клавіатури). Щоб зробити опитування стовпця

клавіатури (SW3, SW6, SW9, SW12; SW4, SW7, SW10, S13; чи SW5, SW8, SW11, SW14), необхідно виставити на відповідній лінії адреси (A0, A1, A2 для першого, другого і третього стовпця відповідно) рівень логічного нуля, а на інших лініях – рівень логічної одиниці і зчитати стан буфера клавіатури, підключеного до шини даних стенда як доступний для читання елемент пам'яті з адресою 9000h. Якщо кнопка натиснута, то відповідний біт у байті, що читається, дорівнюватиме нулю, якщо не натиснута – одиниці.

Таблиця 3.4 – Адресація клавіатури стенда EV8031/AVR

Стовпець (кнопки)	Адреса
1 (SW3,SW6,SW9,SW12)	9006h
2 (SW4,SW7,SW10,SW13)	9005h
3 (SW5,SW8,SW11,SW14)	9003h

Приклад програми з обробниками переривань

Вивести на статичний індикатор число 00H. Інкрементувати (збільшувати на 1) його за натисканням SW15 та декрементувати (зменшувати на 1) за натисканням SW16.

```
.include «m8515def.inc»          ;підключення файлу визначень
                                 ;присвоєння назв регістрам
.def temp = R16                  ;для тимчасових значень
.def number = R17                ;для зберігання числа для
виведення
.def del1 = R24                  ;del1 дляR17 – для затримки
.def del2 = R25                  ;del2 для R18 – для затримки


rjmp RESET                       ;вектори переривань:
rjmp INT_0                       ;вектор початкового скидання
rjmp INT_1                       ;вектор  INT0
.org $011                        ;вектор  INT1

INT_0:
rcall delay                      ;обробка переривання INT0
sbic PIND, 2                     ;затримка проти брязкоту
reti                             ;повторно зчитати кнопку
inc number                       ;вийти, якщо кнопку не
sts $A001, number                натиснуто
reti                             ;інакше – інкрементувати
                                 number
INT_1:                           ;і вивести на індикатор
rcall delay                      ;кінець обробки
sbic PIND, 3
reti                             ;обробник переривання INT1
```

```
dec number                    ;затримка проти брязкоту
sts $A001, number             ;повторно зчитати кнопку
reti                          ;вийти, якщо кнопку не
                              натиснуто
RESET:                        ;інакше – декрементувати
ldi temp, low(RAMEND)         number
out SPL, temp                 ;і вивести на індикатор
ldi temp, high(RAMEND)        ;кінець обробника
out SPH, temp

                              ;початок програми
ldi temp, $8A                 ;ініціалізація стека
out MCUCR, temp


ldi temp,(1<<INT1)|(1<<INT0)
out GICR, temp                ;дозвіл роботи з зовнішнім
                              адресним
ldi temp, $03                 ;простором та налаштування
sts $A004, temp               реакції
                              ;переривань INT0 iINT1 на спад
clr number
sts $A001, number             ;дозволити переривання INT0
                              maINT1
sts $8000, number
sts $8001, number
sts $A006, number             ;запалити молодший байт
                              статичного
sei                           ;індикатора

loop:                         ;обнулити number
rjmp loop                     ;і вивести на індикатор

delay:                        ;погасити всі інші індикатори
ser del1
ser del2
del:
sbiw del1, 1                  ;дозволити переривання
brcc del
ret                           ;зациклити


                              ;підпрограма затримки
```

Варіанти індивідуальних завдань наведено в табл. 3.5.

Таблиця 3.5 – Варіанти індивідуальних завдань

№ завдання	Текст індивідуального завдання
1	Підрахувати і відобразити на статичному індикаторі кількість натискань кнопки SW15.
2	Реалізувати опитування клавіатури. Номер клавіші відображати шляхом засвічування відповідної точки на знакосинтезувальному індикаторі.
3	Реалізувати опитування клавіатури. Номер клавіші послідовно відображати в кожному розряді динамічного індикатора.
4	За натисканням SW15 запускати «вогник, що біжить» на світлодіодах HL1–HL8, при відпусканні – плавне загоряння числа 3 на знакосинтезувальному індикаторі.
5	За натисканням SW16 включити секундомір з відображенням на статичному індикаторі значення секунд, при відпусканні запускати «бігучу тінь» на світлодіодах HL1–HL8.
6	Реалізувати опитування клавіатури. Номер клавіші відображати позиційним кодом на світлодіодах HL1–HL8 з відображенням значення кнопки на динамічному індикаторі.
7	Реалізувати програму введення чотиризначного числа з клавіатури, використовуючи статичний індикатор і дублюючи значення натиснутої кнопки на знакосинтезувальному індикаторі.
8	Реалізувати опитування клавіатури після двох натискань SW16. Номер клавіші відображати на динамічному індикаторі.
9	За натисканням SW15 запускати будь-яке бігуче значення на знакосинтезувальному індикаторі, а після натискання SW16 запалити всі точки в шаховому порядку.
10	Відображати значення секунд на статичному індикаторі. За перериванням INT0 зупинити секундомір і засвітити світлодіоди HLn (n–непарне).
11	Відображати число 7543 на динамічному індикаторі. За перериванням INT1 засвітити світлодіоди HLn (n–парне).
12	На статичному індикаторі відобразити число 5555. За перериванням INT0 відображати «шахову дошку» на знакосинтезувальному індикаторі, за перериванням INT1 відобразити на статичному індикаторі число 3333.
13	За натисканням SW15 реалізувати програму введення тризначного числа з клавіатури з відображенням на статичному індикаторі.
14	За натисканням SW16 запускати «тінь, що біжить» на знакосинтезувальному індикаторі, а при повторному натисканні SW16 загасити всі точки.
15	Реалізувати опитування клавіатури. Номер клавіші відображати двійковим кодом на світлодіодах HL1–HL8.

3.4 Цифроаналогове перетворення

Мета роботи: вивчення методів цифроаналогового перетворення.

Навчальне завдання: розробка програм для формування різних аналогових сигналів.

Порядок виконання лабораторної роботи

1. Вивчити структурну схему модуля цифроаналогового перетворення (ЦАП) на платі розширення стенда EV8031/AN.
2. Розробити алгоритм для виконання індивідуального завдання (табл. 3.6) до початку лабораторного заняття.
3. Розробити програму для виконання індивідуального завдання до початку лабораторного заняття.
4. Ввести програму індивідуального завдання на персональному комп'ютері.
5. За допомогою ПНЗ проаналізувати виконання індивідуальної програми.
6. Завантажити програму в стенд EV8031/AVR. Переконатися в правильному виконанні індивідуального завдання (формування заданої форми сигналу), використовуючи осцилограф, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд EV8031/AVR.
7. Роздрукувати лістинг програми, що правильно працює.
8. Відповісти на контрольні питання викладача.

Контрольні питання

1. Методи і типи ЦАП.
2. Статичні параметри ЦАП.
3. Поняття дискретності, квантування, роздільна здатність ЦАП.
4. Характеристика перетворення, нелінійність ЦАП.
5. Напруга зсуву нуля, допустима напруга на виході ЦАП.
6. Динамічні параметри ЦАП.
7. Фактори, що впливають на похибку ЦАП.
8. Способи апаратної реалізації ЦАП.
9. Приклади практичного застосування ЦАП.

Короткі теоретичні відомості

ЦАП призначений для перетворення числа, визначеного, як правило, у вигляді двійкових кодів, у напругу або струм пропорційно значенню цифрового коду.

Дуже часто ЦАП входить до складу мікропроцесорної системи. У цьому випадку, якщо не потрібна висока швидкодія, цифроаналогове перетворення може бути дуже просто здійснено за допомогою широтно-імпульсної модуляції (ШІМ). Схема ЦАП з ШІМ наведена на рис. 3.13.

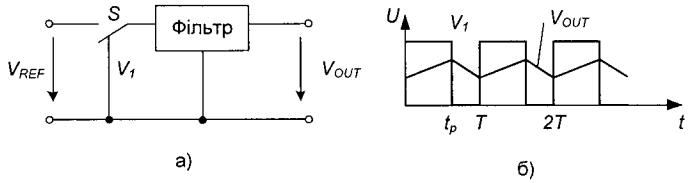


Рисунок 3.13 – ЦАП з широтно-імпульсною модуляцією:
а) структурна схема; б) часова діаграма

Найпростіше організовується цифроаналогове перетворення в тому випадку, якщо МК має вбудовану функцію широтно-імпульсного перетворення (наприклад, AT90S8515 фірми Atmel або 87C51GB фірми Intel) [5, 11, 12]. Вихід ШІМ керує ключем *S*. Залежно від заданої розрядності перетворення (для МК AT90S8515 можливі режими 8, 9 і 10 розрядів) контролер за допомогою власного таймера/лічильника формує послідовність імпульсів, відносна тривалість яких $\gamma = t_p / T$ визначається співвідношенням:

$$\gamma = \frac{D}{2^N},$$

де N – розрядність перетворення;

D – код перетворення.

Фільтр нижніх частот згладжує імпульси, виділяє середнє значення напруги. У результаті вихідна напруга перетворювача:

$$V_{OUT} = \gamma V_{REF} = \frac{DV_{REF}}{2^N}.$$

Розглянута схема забезпечує ідеальну лінійність перетворення, не містить прецизійних елементів (за винятком джерела опорної напруги). Основний її недолік – низька швидкодія.

Паралельні ЦАП мають більшу швидкодію і тому вони можуть застосовуватися для більш широкого кола задач. Більшість схем паралельних ЦАП реалізовано на додаванні струмів, що пропорційні вазі цифрових двійкових розрядів, причому повинні додаватись тільки струми тих розрядів, значення яких дорівнює 1.

Важливу частину ЦАП складає цифровий інтерфейс, тобто схеми, що забезпечують зв'язок входів керування ключів з джерелами цифрових сигналів. Структура цифрового інтерфейсу визначає спосіб підключення ЦАП до джерела вхідного коду. Властивості цифрового інтерфейсу безпосередньо впливають і на форму вихідної характеристики ЦАП. Так, неодночасність надходження розрядів вхідного слова на входи керування ключів перетворювача приводить до появи вузьких викидів («голок») у вихідному сигналі при зміні коду.

При керуванні ЦАП від цифрових пристроїв з жорсткою логікою входів керування ключів ЦАП можуть бути безпосередньо підключені до виходів цифрових пристроїв, тому в багатьох мікросхемах ЦАП, особливо ранніх (572ПА1, 594ПА1, 1108ПА1, AD565) [3, 4, 12], будь-яка істотна цифрова частина відсутня. Якщо ж ЦАП входить до складу МПС і отримує вхідний код від шини даних, то він повинен бути забезпечений схемами, що дозволяють приймати вхідне слово від шини даних, зберігати його до отримання іншого слова, комутувати відповідно до цього слова ключі ЦАП. Для керування процесом завантаження вхідного слова ЦАП повинен мати відповідні входи керування і схему керування. Залежно від способу завантаження вхідного слова в ЦАП розрізняють перетворювачі з послідовним і паралельним інтерфейсами вхідних даних.

ЦАП з послідовним інтерфейсом вхідних даних, крім власне ЦАП, містить на кристалі послідовний регістр завантаження, паралельний регістр зберігання та логіку керування (рис. 3.14, а). При активному рівні сигналу $\overline{CS}$ (в даному випадку низькому) вхідне слово довжиною N (що дорівнює розрядності ЦАП) завантажується по лінії DI в регістр зсуву під керуванням тактової послідовності CLK. Після закінчення завантаження, виставивши активний рівень на лінію $\overline{LD}$, вхідне слово записують в регістр зберігання, виходи якого безпосередньо керують ключами ЦАП. Для того, щоб мати можливість передавати по одній лінії вхідні коди на декілька ЦАП, останній розряд регістра зсуву з'єднується з виводом ІС. Цей вивід підключається до входу DI наступного ЦАП і так далі. Коди вхідних слів передаються, починаючи з коду найостаннішого перетворювача.

Як приклад, на рис. 3.14, б наведена часова діаграма, що відображає процес завантаження вхідного слова в ЦАП AD7233 [3, 4]. Мінімально допустимі значення інтервалів часу (порядку 50 нс), позначених на епюрах, вказуються в технічній документації на мікросхему.

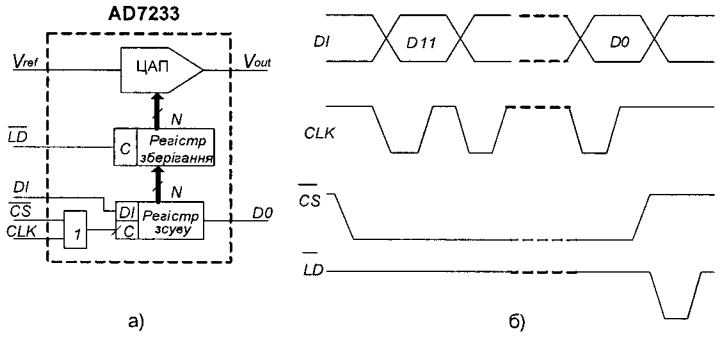


Рисунок 3.14 – ЦАП з послідовним інтерфейсом:
а) функціональна схема; б) часова діаграма

Після закінчення завантаження МК змінює рівень на вході $\overline{CS}$, як це показано на рис. 3.14, б). Виставивши активний рівень на вході $\overline{LD}$, ЦАП забезпечує пересилання вхідного коду з регістра зсуву ЦАП в регістр зберігання. Час завантаження залежить від тактової частоти МК і зазвичай складає одиниці мікросекунд. У випадку, якщо коливання вихідного сигналу ЦАП під час завантаження допустимі, вхід $\overline{LD}$ можна з'єднати з загальною точкою схеми.

На рис. 3.15 наведений варіант схеми підключення перетворювача з послідовним інтерфейсом до МК. На час завантаження вхідного слова в ЦАП через послідовний порт МК, до якого можуть бути також підключені і інші приймачі, на вхід $\overline{CS}$ (вибір кристала) подається активний рівень з однієї з ліній введення/виведення МК.

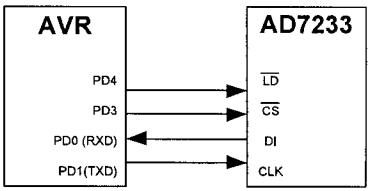


Рисунок 3.15 – Підключення ЦАП
з послідовним інтерфейсом до МК AVR

Стосовно ЦАП з паралельним інтерфейсом вхідних даних можна сказати, що частіше використовуються два варіанти. У першому варіанті на N входів даних N-розрядного ЦАП подається вхідне слово повністю. Інтерфейс такого ЦАП містить два регістри зберігання і схему керування (рис. 3.16, а) Два регістри зберігання потрібні, якщо пересилання вхідного коду до ЦАП і встановлення вихідного аналогового сигналу, відповідного цьому коду, повинні бути розділені в часі.

Подача на вхід асинхронного скидання $\overline{CLR}$ сигналу низького рівня приводить до скидання першого регістра і, відповідно, вихідної напруги ЦАП.

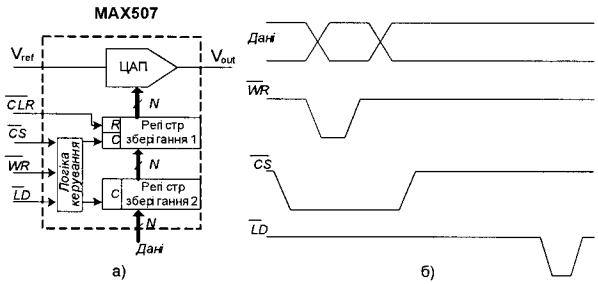


Рисунок 3.16 – ЦАП з паралельним інтерфейсом:
а) структурна схема; б) часова діаграма

Приклад блок-схеми підключення 12-розрядного ЦАП MAX507 до 16-розрядного МП наведений на рис. 3.17. Процесор посилає вхідний код в ЦАП як в елемент пам'яті даних. Спочатку з шини адреси/даних AD надходить адреса ЦАП, яка фіксується регістром за командою з виходу *ALE* МП і, після дешифрування, активізує вхід $\overline{CS}$ ЦАП. Зразу після цього МП подає на шину *AD* вхідний код ЦАП і потім сигнал запису на вхід $\overline{WR}$.

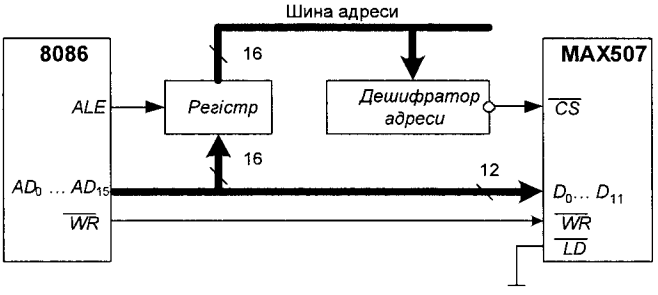


Рисунок 3.17 – Підключення ЦАП
з паралельним інтерфейсом до МП 8086

Для підключення багаторозрядних ЦАП до восьмирозрядних МП і МК використовується другий варіант паралельного інтерфейсу. Він передбачає наявність двох паралельних завантажувальних регістрів для прийому молодшого байта (МБ) вхідного слова і старшого байта (СБ) (рис. 3.18). Пересилання байтів вхідного слова до завантажувальних регістрів може відбуватися у будь-якій послідовності.

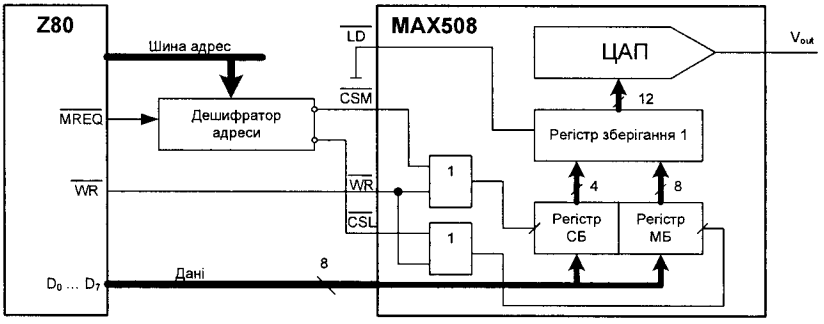


Рисунок 3.18 – Підключення ЦАП з паралельним інтерфейсом
другого типу до восьмирозрядного МП

На платі розширення стенда EV8031/AN є мікросхема ЦАП AD7801 (рис. 3.19).

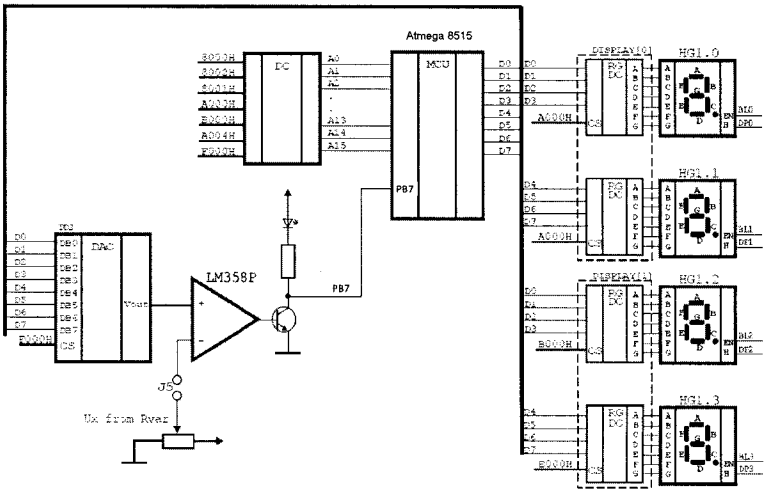


Рисунок 3.19 – Схема підключення мікросхеми ЦАП AD7801 на платі розширення стенда EV8031/AN

Вона являє собою восьмирозрядний ЦАП (з паралельним інтерфейсом) з розмахом вихідної напруги, що дорівнює напрузі живлення. Для спостереження вихідного сигналу з мікросхеми ЦАП необхідно підключити осцилограф до BNC гнізда і замкнути перемичку J5. Доступ до ЦАП здійснюється як до осередку зовнішнього ОЗП за адресою 0F000h.

Приклад виконання програми цифроаналогового перетворення
Сформувати пилкоподібний сигнал

```
.include «m8515def.inc»      ;підключення файлу визначень
                             ;присвоєння назв регістрам
.def temp = R16              ;для тимчасових значень
.def vout = R17              ;для зберігання значення напруги

                             ;вектори переривань:
rjmp RESET                   ;вектор початкового скиду
.org $007
rjmp TIM0_OVF                ;вектор  переповнення T/C0
.org $011

TIM0_OVF:                    ;обробник переривання T/C0
inc vout                     ;збільшити вихідну напругу
                             ;(при досягненні максимального
                             ;значення скинути в нуль)
```

```
    sts $F000, vout              ;вивести в ЦАП
    reti                         ;кінець обробника переривання

    RESET:                       ;початок основної програми
    ldi temp,                    ;ініціалізація стеку
low(RAMEND)
    out SPL, temp
    ldi temp,
high(RAMEND)
    out SPH, temp                ;дозвіл роботи з зовнішнім
                                 адресним
    ldi temp, 1<<SRE             ;простором
    out MCUCR, temp
                                 ;запуск таймера-лічильника 0
    ldi temp, $01                ;частотою Fm = 7 372 800 Гц
    out TCCR0, temp              ;частота вихідної напруги:
                                 ;Fвих = 7 372 800 /256 = 28 800 Гц

                                 ;дозвіл переривання за
    ldi temp, 1<<TOIE0           переповненням
    out TIMSK, temp               ;таймера-лічильника 0

    sei                          ;дозвіл глобальних переривань

    loop:                        ;зациклення
    rjmp loop
```

Варіанти індивідуальних завдань наведено в табл. 3.6.

Таблиця 3.6 – Варіанти індивідуальних завдань

№ варіанта	Текст індивідуального завдання
1	Сформувати пилкоподібну напругу з частотою 50 Гц. Відображати на статичному індикаторі число згенерованих імпульсів.
2	За натисканням SW3 сформувати трикутні імпульси, передній фронт – 20 мс, задній – 10 мс, кожен 10 імпульс відображати на динамічному індикаторі.
3	За натисканням SW4 сформувати трапецієподібні імпульси, передній фронт – 13 мс, задній – 15 мс; кожну секунду запалювати світлодіод HL1.
4	Сформувати синусоїду з частотою повторення 120 Гц.
5	Сформувати пилкоподібну напругу з частотою повторення 200 Гц і тривалістю переднього фронту 2 мс.
6	За натисканням SW5 сформувати синусоїду з частотою повторення 100 Гц.
7	Сформувати прямокутні імпульси з тривалістю 25 мс і скважністю 4.

Продовження табл. 3.6

№ варіанта	Текст індивідуального завдання
8	За натисканням SW16 сформувати трикутні імпульси, передній фронт – 25 мс, задній – 5 мс.
9	Сформувати синусоїду з частотою повторення 300 Гц. За натисканням SW15 змінити частоту на 100 Гц.
10	Сформувати два прямокутні імпульси, один з максимальною амплітудою і тривалістю і другий – 2/3 амплітуди і тривалості від максимальної з періодом повторення 40 Гц.
11	Сформувати прямокутні імпульси, з тривалістю 25 мс за натисканням SW6 на клавіатурі сформувати трикутні імпульси.
12	Сформувати синусоїду з частотою повторення 70 Гц, після натискання SW7 на клавіатурі прямокутні імпульси з тривалістю 25 мс і скважністю 2.
13	За натисканням SW15 сформувати трикутні імпульси, передній фронт – 15 мс, задній – 40 мс.
14	За натисканням SW9 на клавіатурі сформувати трапецієподібні імпульси, передній фронт – 20 с, задній – 20 мс.
15	Сформувати три прямокутних імпульси, один 1/3 максимальної амплітуди, 2-ий – 2/3 максимальної амплітуди, 3-й – максимальної амплітуди з періодом повторення 100 Гц.

3.5 Аналого-цифрове перетворення

Мета роботи: навчитися вимірювати аналогову величину.

Навчальне завдання: розробка програм вимірювання аналогових величин для різних методів вимірювання і типів АЦП.

Порядок виконання лабораторної роботи

1. Вивчити структурну схему модуля АЦП на платі розширення стенда.
2. Розробити алгоритм і програму для виконання індивідуального завдання (табл. 3.6) до початку лабораторного заняття.
3. Ввести програму індивідуального завдання на персональному комп'ютері.
4. За допомогою ПНЗ проаналізувати виконання індивідуальної програми.
5. Завантажити програму в стенд EV8031/AVR. Переконатися в правильному виконанні індивідуального завдання, змінити значення напруги, що подається на вхід АЦП, повторити перетворення, при негативному результаті здійснити зміна алгоритму або програми. Повторити завантаження програми в стенд EV8031/AVR.
6. Роздрукувати лістинг програми, що правильно працює.
7. Відповісти на контрольні питання викладача.

Контрольні питання

1. Методи і типи АЦП.
2. Статичні параметри АЦП.
3. Поняття дискретності, квантування, роздільна здатність.
4. Характеристика перетворення, диференціальна нелінійність АЦП, відхилення коефіцієнта перетворення.
5. Напруга зсуву нуля.

6. Динамічні параметри АЦП.

7. Час перетворення, час затримки запуску, час циклу перетворення, максимальна частота перетворення.

8. Поняття «апертурний час».

9. Фактори, що впливають на похибку АЦП.

10. Апаратні реалізації АЦП.

11. Приклади практичного застосування АЦП.

12. Побудова схем АЦП за допомогою мікросхем ЦАП.

Короткі теоретичні відомості

На рис. 3.20 показаний простий резистивний АЦП східчастого типу з трьома ключами. Резистори складені в *R*/2 *R* конфігурації. Номінали резисторів не важливі; опір може бути 10 кОм, 100 кОм і тому подібне. Кожен з ключів *S*0...*S*2 може підключати один вивід одного резистора номіналом 2*R* між землею і вхідною опорною напругою, V_{REF}. На рис. 3.20 показано, що відбувається, коли *S*2 замкнутий «ON» (з'єднаний із V_{REF}), а *S*0 і *S*1 розімкнені «OFF» (з'єднані з землею). У результаті спаду напруги на послідовно-паралельній резистивній ділянці остаточна вихідна напруга (V_0) стає рівною 0,5 V_{REF}. Можна так само обчислити V_0 для всіх інших комбінацій ключів (табл. 3.7).

Якщо положення (замкнутий-розімкнутий) трьох ключів уявити як цифрове слово з трьох бітів, тоді можемо переписати таблицю, використовуючи позначення ON = 1 (замкнутий), OFF = 0 (розімкнутий) (табл. 3.8).

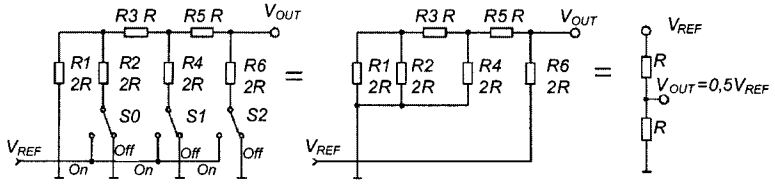


Рисунок 3.20 – Бітові АЦП

Таблиця 3.7 – Залежність вихідної напруги відносно стану ключів

Стан			Вихідна напруга V_{OUT}
S2	S1	S0	
OFF	OFF	OFF	0
OFF	OFF	ON	$0{,}125 \cdot V_{REF}(1/8\text{-}V_{REF})$
OFF	ON	OFF	$0{,}25 \cdot V_{REF}(2/8\text{-}V_{REF})$
OFF	ON	ON	$0{,}375 \cdot V_{REF}(3/8\text{-}V_{REF})$
ON	OFF	OFF	$0{,}5 \cdot V_{REF}(4/8\text{-}V_{REF})$
ON	OFF	ON	$0{,}625 \cdot V_{REF}(5/8\text{-}V_{REF})$
ON	ON	OFF	$0{,}75 \cdot V_{REF}(6/8\text{-}V_{REF})$
ON	ON	ON	$0{,}875 \cdot V_{REF}(7/8\text{-}V_{REF})$

Таблиця 3.8 – Змінена таблиця залежності вихідної напруги відносно стану ключів

Стан ключа			Логічний еквівалент			S0...S2 чисельний еквівалент
S2	S1	S0	S2	S1	S0	
OFF	OFF	OFF	0	0	0	0
OFF	OFF	ON	0	0	1	1
OFF	ON	OFF	0	1	0	2
OFF	ON	ON	0	1	1	3
ON	OFF	OFF	1	0	0	4
ON	OFF	ON	1	0	1	5
ON	ON	OFF	1	1	0	6
ON	ON	ON	1	1	1	7

Таким чином, вихідна напруга є поданням комбінації станів ключів. Кожен додатковий біт у таблиці додає $V_{REF}/8$ до загальної напруги. Або, іншими словами, вихідна напруга дорівнює двійковому числу комбінації S0...S2, помноженому на $V_{REF}/8$. Такий 3-бітовий АЦП має 8 можливих станів, і кожен крок напруги складає $V_{REF}/8$.

Якщо додати ще одну *R*/2 *R* пару і ще один ключ до схеми, отримаємо схему з чотирма ключами і шістнадцятьма кроками по $V_{REF}/16$ вольт кожна. Схема з вісьмома ключами має 256 рівнів кожний по $V_{REF}/256$ вольт. Замінивши механічні ключі в схемі на електронні, можна створити повноцінний інтегральний АЦП.

Для передачі аналогового сигналу до МПС використовують АЦП. АЦП сприймає аналоговий сигнал, напругу або струм і перетворює його в цифрове слово, зрозуміле МП (рис. 3.21, а). На рис. 3.21, б) наводиться умовне зображення АЦП, який має вхід опорної напруги V_{REF} (reference) та вхід, куди подається сигнал. АЦП має один вихід – цифрове слово розмірністю 8 бітів, що являє в цифровій формі вхідну величину.

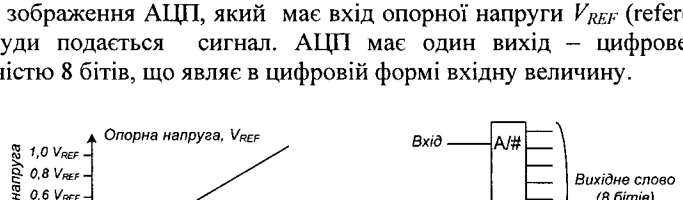


Рисунок 3.21 – До розгляду роботи АЦП

Опорна напруга, V_{REF} – максимальна величина напруги, яку АЦП може перетворити. АЦП, що зображений на рис. 3.21, може перетворити напругу з величинами від 0 до V_{REF}. Цей діапазон напруги поділений на 256 рівнів або кроків. Розмір одного кроку задається так:

$$\frac{V_{REF}}{256} = \frac{5B}{256} = 0{,}0195B = 19{,}5\ мВ$$

8-бітовий АЦП подає аналоговий сигнал цифровим словом. Старший значущий біт цього слова (біт 7) показує, чи вища вхідна напруга, ніж $V_{REF}/2$ (2,5 В при V_{REF} = 5 В). Кожен наступний біт (від біта 6 до біта 0, який буде молодшим значущим бітом – least significant bit, LSB) є половиною значення попереднього, як показано в табл. 3.9.

Таблиця 3.9 – Подання аналогового входу цифровим словом

Біт	Біт 7	Біт 6	Біт 5	Біт 4	Біт 3	Біт 2	Біт 1	Біт 0
Напруга [В]	2,5	1,25	0,625	0,3125	0,156	0,078	0,039	0,0195

Так, маючи цифрове слово 0010 1100, можна визначити величину напруги (табл. 3.10).

Таблиця 3.10 – Відповідність цифрового слова певним значенням

Біт	Біт 7	Біт 6	Біт 5	Біт 4	Біт 3	Біт 2	Біт 1	Біт 0
Напруга [В]	2,5	1,25	0,625	0,3125	0,156	0,078	0,039	0,0195
Вихідна величина	0	0	1	0	1	1	0	0

Додавши напругу, що відповідає кожному одиничному біту, отримуємо реальну напругу.

$$0{,}625 + 0{,}156 + 0{,}078 = 0{,}859B$$

Роздільна здатність АЦП визначається опорною напругою і розрядністю слова. Роздільна здатність – це найменша напруга, яка може бути виміряна АЦП і визначає точність перетворення. Роздільна здатність – це найменший крок, може бути визначена діленням V_{REF} на число можливих величин перетворення.

Наприклад, для 8-бітного АЦП з V_{REF} = 5 В роздільна здатність дорівнює 0,0195 В (19,5 мВ). Це означає, що будь-яка вхідна напруга, менша 19,5 мВ, дасть на виході 0. Вхідна напруга між 19,5 і 39 мВ приведе до 1 на виході. Між 39 і 58,6 мВ – до 2 і так далі. Роздільна здатність може бути покращена зменшенням V_{REF}. Зміна V_{REF} з 5 до 2,5 В дасть роздільну здатність 2,5/256 або 9,7 мВ. Проте максимальна напруга вимірювання тепер складе не 5, а 2,5 В.

Є лише один шлях збільшення роздільної здатності без зміни опорної напруги – використання АЦП з більшою розрядністю слова. Так 10-бітовий АЦП з опорною напругою 5 В має 2^{10}, або 1024 можливих вихідних двійкових кодів. Роздільна здатність складе 5 В/1024 або 4,88 мВ.

Існує декілька типів АЦП, що працюють з різними швидкостями, мають різні інтерфейси і забезпечують різну точність. Розглянемо їх особливості [3].

Слідкувальний АЦП складається з компаратора, реверсивного лічильника, керованого компаратором, і ЦАП. Компаратор порівнює вхідну напругу з вихідною напругою ЦАП і керує напрямком лічби реверсивного лічильника (+/-). Якщо вхідна напруга більша, ніж напруга ЦАП, лічильник під дією тактових імпульсів, що надходять на його вхід (рис. 3.22, а), рахує на додавання (+), якщо ж менше напруги ЦАП, лічильник рахує на віднімання (-).

Вхід ЦАП з'єднаний з виходом лічильника. Припустимо $V_{REF} = 5$ В. Це означає, що перетворювач зможе працювати в діапазоні вхідної напруги 0...5 В. Якщо старший значущий біт вхідного слова ЦАП дорівнює 1, то $V_{ЦАП} = 2{,}5$ В. Якщо наступний біт дорівнює 1, додається 1,25 В, в результаті встановлюється 3,75 В. Кожен наступний біт додає половину вихідної напруги, що відповідає попередньому біту. Таким чином, вхідні біти ЦАП відповідають певній напрузі (див. табл. 3.9, 3.10).

На рис. 3.22, а показано, як слідкувальний АЦП обробляє постійний сигнал з рівнем $V_{BX} = 0{,}37$ В. Лічильник стартує з нуля, при цьому вихід компаратора знаходиться у високому логічному рівні. Лічильник рахує на додавання з кожним тактовим імпульсом, ступінчасто піднімаючи $V_{ЦАП}$. Коли лічильник проходить двійкову величину, яка є V_{BX}, то компаратор перемикається, і лічильник починає рахувати на віднімання. У результаті, лічильник змінюється в околі величини, яка є V_{BX}.

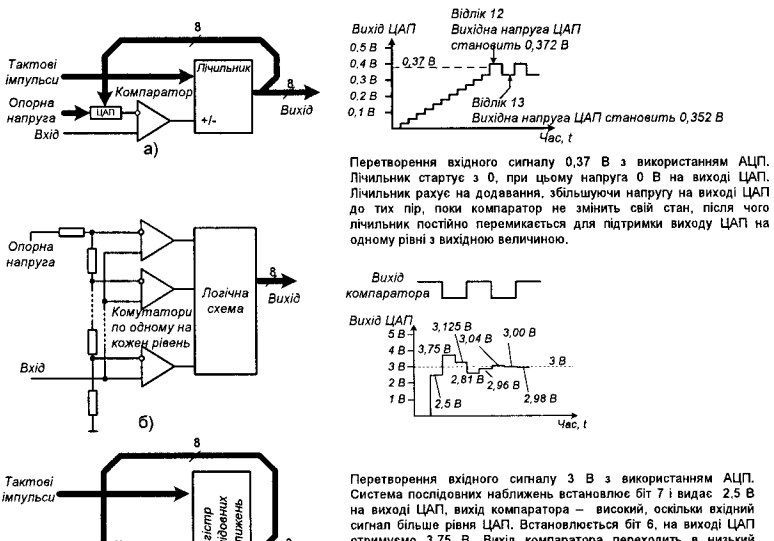


Рисунок 3.22 – Три типи АЦП:
а) слідкувальний; б) паралельний; в) послідовних наближень

Головний недолік слідкувальних АЦП – низька швидкість перетворення. Перетворення може зайняти до 256 тактів 8-бітового виходу, 1024 такти для 10-бітового значення і так далі. До того ж, швидкість перетворення змінюється залежно від $V_{ВХ}$. Якби напруга в даному прикладі складала 0,18 В, перетворення зайняло лише половину тактів у порівнянні з $V_{ВХ} = 0{,}37$ В.

Максимальна тактова частота АЦП слідкувального типу залежить від затримки поширення сигналу в ЦАП і компараторі. Після кожного тактового імпульсу вихід лічильника має бути перетворений ЦАП, а результат перетворення з виходу потрапити на інверсний вхід компаратора. Компаратор віднімає деяку частину часу для реакції на зміну $V_{ЦАП}$, створюючи новий сигнал керування на додавання чи віднімання для реверсивного лічильника. Всі затримки сигналу роблять слідкувальний АЦП найповільнішим з усіх презентованих типів АЦП. Слідкувальні АЦП широкого поширення не отримали; серед інтегральних схем, що випускаються такими фірмами, як: Analog Devices, Maxim, Burr-Brown, неможливо знайти слідкувальних АЦП. Треба відзначити, що АЦП послідовних наближень працює з великою швидкодією з розрядністю 8-бітового слова. Проте зустрічаються такі випадки, коли слідкувальний АЦП буде корисним. Наприклад, якщо сигнал змінюється повільно відносно частоти дискретизації, слідкувальний АЦП встановлюватиме вихідне число за менше число тактів, ніж АЦП послідовних наближень.

Паралельний АЦП (див. рис. 3.22, б) – найшвидший з тих, що випускаються виробниками. Паралельний АЦП містить по компаратору на кожен крок напруги. На один із входів кожного компаратора подається вимірювана напруга. Інші входи компараторів підключаються до лінійки резисторів. Якщо рухатися вгору по цій лінійці, то можна відмітити, що кожен наступний компаратор підключений до вищої напруги з постійним кроком приросту напруги. Залежно від зміни вхідної напруги будуть переключені у стан 1 лише ті компаратори, в яких було перевищено опорну напругу. Сигнали з виходів всіх компараторів надходять в логічний блок – шифратор, який і задає вихідне цифрове m-бітове слово АЦП залежно від того, які компаратори знаходяться в стані 0, а які – в стані 1. Число компараторів визначається числом кодових комбінацій 2^m–1, де m – роздільна здатність АЦП. Тоді для 3-бітового АЦП буде потрібно 7 компараторів, для 4-бітового – 15 компараторів і так далі. Роздільна здатність паралельного АЦП лежить в межах, як правило, від 6 до 12 бітів. Нескладно підрахувати, що для 6-бітового АЦП потрібно 63 вхідних компаратори.

Швидкість перетворення паралельного АЦП складається з часу затримки одного компаратора і логічного блока. Паралельні АЦП найбільш швидкодійні серед основних типів АЦП. Їх швидкодія складає десяті частки мікросекунд. До недоліків паралельних АЦП можна віднести різке збільшення вхідних компараторів разом зі збільшенням роздільної здатності.

З цієї причини для них недосяжний бар'єр у 12 бітів, навіть зі збільшенням міри інтегрування мікросхем. Такі величезні об'єми інтегральних елементів (компараторів і шифраторів) в одній мікросхемі реально впливають на робочі характеристики. Використання великого числа компараторів веде до збільшення споживаного струму. Так, наприклад, 10-бітовий паралельний АЦП може споживати струм до 0,5 А. У свою чергу, струм споживання визначає потужність розсіювання корпусом мікросхеми, тому 8–12-бітові АЦП повинні мати на платі ефективний тепловідвід від корпуса АЦП.

АЦП послідовного наближення (див. рис. 3.22, в) подібний до слідкувального АЦП в тому, що система ЦАП/лічильник формує напругу, яка надходить на один вхід компаратора, а вхідний сигнал надходить на інший вхід. Відмінність полягає в тому, що регістр послідовного наближення виконує двійковий пошук замість лічби вгору або вниз по одиниці. Візьмемо для прикладу, що початкова вхідна напруга складає 3 В, а опорна – 5 В (див. рис. 3.22). Регістр послідовного наближення виконає перетворення таким чином.

Встановити MSB, напруга на виході ЦАП дорівнює 2,5 В.

Вихід компаратора – високий логічний рівень, MSB залишається встановленим.

Результат: 1000 0000 Встановити біт 6, напруга на виході ЦАП дорівнює 3,75 В (2,5 + 1,25).

Вихід компаратора – низький логічний рівень, скидається біт 6.

Результат: 1000 0000. Встановити біт 5, напруга на виході ЦАП дорівнює 3,125 В (2,5 + 0,625).

Вихід компаратора – низький логічний рівень, скидається біт 5.

Результат: 1000 0000. Встановити біт 4, напруга на виході ЦАП дорівнює 2,8125 В (2,5 + 0,312 5).

Вихід компаратора – високий, залишається біт 4 встановленим.

Результат: 1001 0000 Встановити біт 3, напруга на виході ЦАП дорівнює 2,968 В (2,8125 + 0,15625).

Вихід компаратора – високий, залишається біт 3 встановленим.

Результат: 1001 1000. Встановити біт 2, напруга на виході ЦАП дорівнює 3,04 В (2,968 + 0,078125).

Вихід компаратора – низький, скидається біт 2.

Результат: 1001 1000. Встановити біт 1, напруга на виході ЦАП дорівнює 3,007 В (2,8125 + 0,039).

Вихід компаратора – низький, скидається біт 1.

Результат: 1001 1000. Встановити біт 0, напруга на виході ЦАП дорівнює 2.988 В (2.8125 + 0.0195).

Вихід компаратора – високий, залишається біт 0 встановленим.

Результат: 1001 1001.

При використанні 8-бітового ЦАП з вихідною напругою 0...5 В цьому результату відповідає напруга:

$$2,5 + 0,3125 + 0,15625 + 0,0195 = 2,988 \text{ В}.$$

Це не точно 3 В, але настільки близько, наскільки можна отримати з 8-бітовим перетворенням і опорною напругою 5 В.

8-бітовий АЦП послідовного наближення може завершити перетворення за 8 тактів, незалежно від вхідної напруги. Потрібно більше логічних кіл, ніж для слідкувального АЦП, але швидкість перетворення буде вища.

За допомогою ЦАП і аналогового компаратора можна реалізувати АЦП послідовного наближення. У стенді EV8031/AVR АЦП побудований на мікросхемах AD7801 (восьмирозрядний ЦАП) і LM358 (компаратор) (див. рис. 3.19). Компаратор своїм прямим входом підключений до виходу ЦАП, як показано на схемі на рис. 3.19. До виходу компаратора через ключ на транзисторі підключений світлодіод, який служить індикатором спрацьовування компаратора, а також з колектора транзистора сигнал надходить на вивід PB7 мікроконтролера. Доступ до ЦАП здійснюється як до осередка зовнішнього ОЗП за адресою 0F000h.

Напруга Ux, що вимірюється, формується змінним резистором R19 (перемичка J5) або джерело зовнішнього сигналу підключається до з'єднувача JP2 (перемичка J8).

У розширеній комплектації стенд поставляється з інтегральним десятибітовим АЦП з паралельним інтерфейсом AD7813. Доступ до АЦП AD7813 здійснюється як до осередку зовнішнього ОЗП за адресою 0E000h.

Приклад програми аналого-цифрового перетворення

Виміряти напругу з внутрішнього джерела опорної напруги і вивести на статичний індикатор.

```
.include «m8515def.inc»        ;підключення файлу визначень

.def temp = R16                ;для тимчасових значень
.def dac = R17                 ;цифрове значення напруги
.def res1 = R18                ;для переведення в 10-ву форму
.def res2 = R19                ;-//-
.def ind1 = R20                ;для виведення
.def ind2 = R21                ;-//-
.def div = R22                 ;для ділення, при переведенні в
                               ;10-ву систему числення
.def del1 = R24                ;для формування затримки
.def del2 = R25                ;-//-

RESET:
ldi          temp,             ;початок
```

```
low(RAMEND)                    ;ініціалізація стека
    out SPL, temp
    ldi                temp,
high(RAMEND)
    out SPH, temp

                               ;дозвіл роботи з зовнішнім
    ldi temp, 1<<SRE           адресним
    out MCUCR, temp            ;простором

    ser temp                   ;встановити резистори
    out PORTB, temp            «підтяжки» на ;порт B

    ldi temp, $20              ;вивести на статичний індикатор
    sts $A004, temp            ;десяткову точку

    clr dac
    out DDRB, dac              ;Порт B – на введення
    sts $8000, dac             ;погасити матрицю
    sts $8001, dac             ;-//- динамічний індикатор
    sts $A006, dac             ;-//- світлодіоди


    loop:                      ;цикл вимірювання
    rcall delay                ;затримка для спрацювання
                               ;ЦАП  і компаратора
    sbis PINB, 7               ;зчитати значення компаратора
    rjmp minus                 ;на minus, якщо вихідна напруга
                               ;перевищує опорну
    inc dac                    ;інакше – інкрементувати напругу
    sts $F000, dac             ;і вивести в ЦАП
    rjmp loop                  ;на початок циклу

    minus:
    dec dac                    ;декрементувати напругу
    sts $F000, dac             ;вивести в ЦАП
    rcall indication           ;вивести на індикатор
    rjmp loop                  ;на початок циклу

    indication:                ;підпрограма виведення на
    mov res1, dac              індикатор
    clr res2                   ;помножити цифрове значення
                               напруги ;на 2 (макс. 510 ~ 4,9 В)
```

```
lsl res1
rol res2
subi res1, 24          ;макс. 486
                       ;перевести в 10-ву систему
rcall div10            ;поділити на 10
mov ind1, res1         ;остача в молодший 16-вий розряд
mov res1, div
rcall div10            ;поділити на 10
swap res1              ;остача в середній 16-вий розряд
add ind1, res1
mov res1, div
rcall div10            ;поділити на 10
mov ind2, res1         ;остача в старший 16-вий розряд
sts $A000, ind2        ;вивести на індикатор
sts $A001, ind1        ;-//-
ret                    ;кінець підпрограми

div10:                 ;підпрограма ділення на 10
clr div                ;віднімати від діленого 10, поки
division:              ;ділене не менше 10.
cpi res2, 0            ;Кожне віднімання рахувати в
brne next              ;регістр div
cpi res1, 10
brlo end_div
next:
subi res1, 10
sbci res2, 0
inc div
rjmp division          ;результат ділення - в div
end_div:               ;остача - в res1
ret                    ;кінець

delay:                 ;затримка
ser del1
ldi del2, $3F
del:
sbiw del1, 1
brcc del
ret
```

УВАГА!!! Для роботи програми ISP – програматор (з'єднувач X3) повинен бути від'єднаний від плати стенда EV8031/AVR (див. відповідний рис.), адже вихід компаратора і лінія SLK під'єднані до однієї лінії МК.

Таблиця 3.11 – Варіанти індивідуальних завдань

№ варіанта	Текст індивідуального завдання
1	Метод послідовного наближення, значення відображати на статичному індикаторі.
2	Метод послідовного наближення, значення відображати на динамічному індикаторі.
3	Слідкувальний, значення відображати на статичному індикаторі.
4	Слідкувальний, значення відображати на динамічному індикаторі.
5	При натисканні кнопки SW15 запустити АЦП послідовного наближення, значення відображати на динамічному індикаторі.
6	При натисканні кнопки SW3 на клавіатурі запустити АЦП послідовного наближення, значення відображати на статичному індикаторі.
7	При натисканні кнопки SW4 на клавіатурі запустити АЦП слідкувального типу, значення відображати на динамічному індикаторі.
8	При натисканні кнопки SW5 на клавіатурі запустити АЦП слідкувального типу, значення відображати на статичному індикаторі.
9	При натисканні кнопки SW6 на клавіатурі запустити АЦП послідовного наближення, результат відобразити на динамічному індикаторі після натискання кнопки SW3.
10	За перериванням INT0 запустити АЦП послідовного наближення, результат відобразити на статичному індикаторі після натискання кнопки SW3.
11	При натисканні кнопки SW7 запустити АЦП слідкувального типу, результат відобразити після натискання кнопки SW8.
12	При натисканні кнопки SW8 запустити АЦП слідкувального типу, результат відобразити на динамічному індикаторі після натискання кнопки SW9.

3.6 Обробка частотних і часових сигналів

Мета роботи: вивчення методів частотного перетворення.

Навчальне завдання: навчитися виконувати вимірювання частоти, періоду, тривалості дискретних сигналів за допомогою таймерів-лічильників мікроконтролера AVR, а також з використанням зовнішнього еталонного генератора.

Порядок виконання лабораторної роботи

1. Вивчити методи частотного перетворення (апаратні і за допомогою внутрішніх таймерів-лічильників мікроконтролера).
2. Вивчити методи вимірювання часових інтервалів.
3. Розробити алгоритм для виконання індивідуального завдання (табл. 3.11) до початку лабораторного заняття.
4. Розробити програму для виконання індивідуального завдання до початку лабораторного заняття.
5. Ввести програму індивідуального завдання на персональному комп'ютері.
6. За допомогою ПВЗ проаналізувати виконання індивідуальної програми.

7. Завантажити програму в стенд EV8031/AVR. Переконатися в правильному виконанні індивідуального завдання, змінити значення напруги, що подається на вхід АЦП, повторити перетворення, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд EV8031/AVR.

8. Роздрукувати лістинг програми, що правильно працює.

9. Відповісти на контрольні питання викладача.

Контрольні питання

1. Методи та типи частотного і часового перетворення.
2. Параметри частотного перетворення.
3. Фактори, що впливають на похибку частотного перетворення.
4. Поняття роздільної здатності частотного перетворення.
5. Вимірювання періоду.
6. Характеристика перетворення, нелінійність частотного перетворення.
7. Апаратні реалізації частотного і часового перетворення.
8. Приклади практичного застосування частотного і часового перетворення.

Короткі теоретичні відомості

В системах автоматизованого управління часто доводиться вимірювати такі величини, як частота f, період T, тривалість τ, зсув фаз φ. Для цього застосовується перетворення частота – код. Залежно від того, який саме параметр необхідно виміряти, застосовують різний підхід.

При вимірюванні частоти fx проводиться підрахунок імпульсів вхідного сигналу, протягом фіксованого відрізку часу T_0 (рис. 3.23).

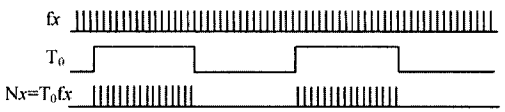


Рисунок 3.23 – Вимірювання частоти

Такий метод застосовують для вимірювання високих частот, понад 100 Гц. Верхня межа встановлюється швидкодією елементів схеми і розрядністю лічильників.

Вимірювання частот нижче 100 Гц замінюється вимірюванням періоду Tx. При цьому проводиться підрахунок імпульсів фіксованої частоти f_0 за інтервал, що дорівнює чи кратний періоду сигналу вимірювання (рис. 3.24).

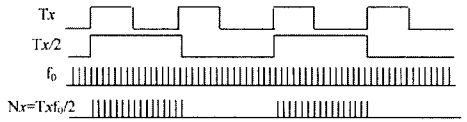


Рисунок 3.24 – Вимірювання періоду
методом підрахунку імпульсів відомої частоти

Таким же чином вимірюється і тривалість імпульсу τ_x, відмінність полягає лише в тому, що при вимірюванні тривалості імпульсу непотрібно ділити частоту на 2.

При вимірюванні зсуву фаз часовий інтервал τ_φ формують шляхом кон'юнкції сигналів f_1x та f_2x (рис. 3.25), отриманий інтервал вимірюють методом, описаним вище.

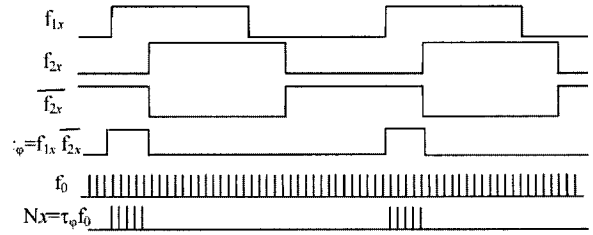


Рисунок 3.25 – Вимірювання зсуву фаз

У лабораторному стенді до входу T0 підключений генератор постійної частоти, а до входу T1 підключений генератор змінної частоти (за наявності плати розширення). Сигнали цих генераторів можна спостерігати на осцилографі, підключеному до BNC з'єднувача. Для спостерігання сигналу T0 потрібно замкнути перемичку J1, а для T1 – J3.

Таймери/лічильники (T/C) призначені для підрахунку зовнішніх подій, для отримання часових затримок, виконання функцій формування часових інтервалів.

МК ATmega8515 має два таймери-лічильники: таймер типу A (8-розрядний T/C0) та таймер типу D (16-розрядний T/C1).

Таймер-лічильник типу A формує запит переривання T/C0 OVF при переповненні восьмирозрядного базового лічильника TCNTO. Структурна схема таймера-лічильника типу A зображена на рисунку 3.26.

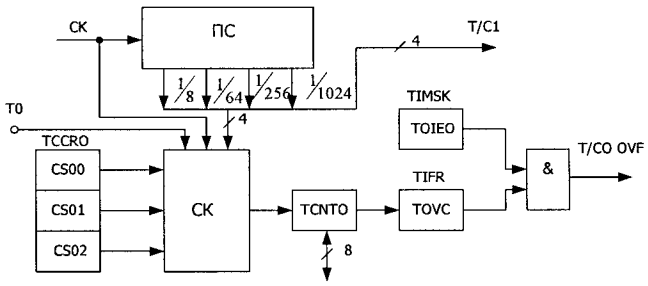


Рисунок 3.26 – Структурна схема таймера-лічильника типу A

Тактовий сигнал мікроконтролера *СК* надходить у перерахункову схему ПС, що являє собою десятирозрядний лічильник, де виконується розподіл частоти тактового сигналу на 8, 64, 256 і 1024. Сигнали з чотирьох виходів перерахункової схеми надходять у схему керування *СК* (мультиплексор).

У схему керування надходять також тактовий сигнал *СК* і сигнал із зовнішнього джерела. Як вхід *Т0* використовується вивід порту PB0 мікроконтролера ATmega8515.

Схема керування, залежно від комбінації станів розрядів CS00, CS01 і CS02 регістра керування TCCRO (табл. 3.12), передасть один із сигналів, що надходять, на рахунковий вхід базового лічильника TCNTO, що веде рахунок на додавання. При переповненні лічильника TCNTO встановлюється в одиничний стан розряд TOVC регістра TIFR і при одиничному стані розряду TOIEO регістра TIMSK у блок переривань надходить запит переривання T/C0 OVF.

Таблиця 3.12 – Регістр керування таймером/лічильником 0 – TCCR0 (The Timer/Counter 0 Control Register)

Біти	7	6	5	4	3	2	1	0	
$33 ($53)						**CS02**	**CS01**	**CS00**	**TCCR0**
Читання/Запис	R	R	R	R	R	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Попередній дільник таймерів/лічильників (табл. 3.13) містить чотири ступеня розподілу: СК/8, СК/64, СК/256 і СК/1024, де СК – вхідний тактовий сигнал. Крім того, як джерело тактових сигналів можуть бути використані сигнали від зовнішніх джерел, тактовий сигнал СК і нульовий тактовий сигнал (STOP).

Таблиця 3.13 – Керування таймером/лічильником 0 – TCCR0

CS02	CS01	CS00	Опис
0	0	0	Таймер-лічильник 0 зупинений
0	0	1	СК
0	1	0	СК/8
0	1	1	СК/64
1	0	0	СК/256
1	0	1	СК/1024
1	1	0	Зовнішній вхід Т0, спадний фронт
1	1	1	Зовнішній вхід Т0, наростаючий фронт

Налаштування TC0

```
LDI R16, $0F
OUT TCNT0, R16  ;Load the TCNTO register with $0F (00001111)
LDI R16, 1
OUT TCCR0, R16  ;Load the TCCRO register with $01 (00000001)
```

Таймер-лічильник типу D має ім'я T/C1. Він містить 16-розрядний базовий лічильник і виконує функції захоплення, порівняння та широтно-імпульсної модуляції (PWM). Структурна схема таймера-лічильника зображена на рис 3.27.

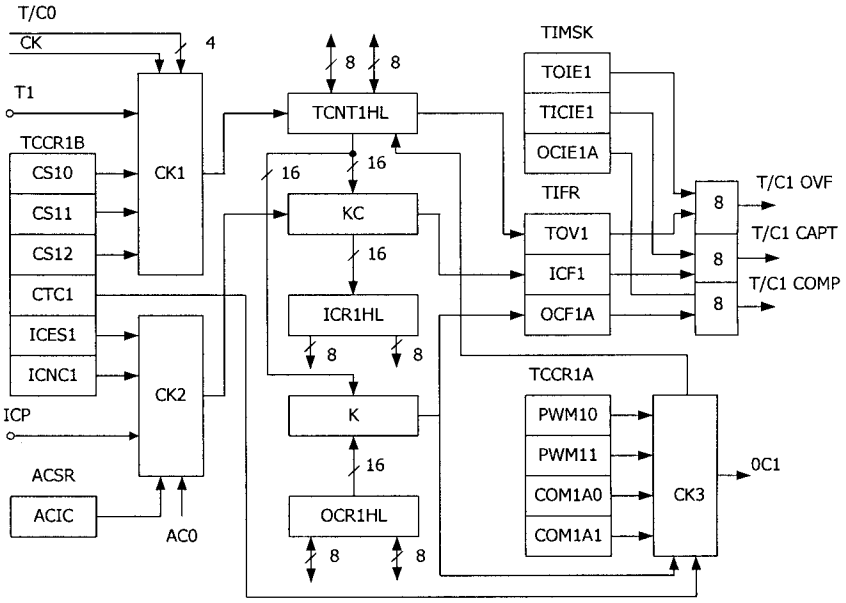


Рисунок 3.27 – Структурна схема таймера-лічильника типу D

На рахунковий вхід 16-розрядного базового лічильника TCNT1HL з виходу схеми керування CK1 може надходити тактовий сигнал мікроконтролера CK, чи один з чотирьох сигналів з перерахункової схеми T/C1, чи сигнал із зовнішнього джерела, що надходить на вхід T1. Як вхід T1 використовується вивід порту PB1. Вибір сигналу визначається комбінацією станів розрядів CS10, CS11 і CS12 регістра керування TCCR1B. При переповненні базового лічильника встановлюється в одиничний стан розряд TOV1 регістра TIFR і при одиничному стані розряду TOI1 регістра TIMSK у блок переривання надходить запит на переривання T/C1 OVF.

Налаштування T/C1 на режим лічби

```
TIM_OVF1:
        LDI     R16, $FF
        OUT     TCNT1H, R16
        LDI     R16, $E5
        OUT     TCNT1L ,R16
        LDI     R16,0
        OUT     TCCR1A, R16
        LDI     R16,1
        OUT     TCCR1B, R16
        LDI     R16, $80
        OUT     TIMSK, R16
        LDI     R16, $DF
        OUT     SPL, R16
        SEI
```

Схема керування СК2 керує виконанням функції захоплення, що полягає в передачі коду, сформованого в базовому лічильнику, через ключову схему КС у 16-розрядний регістр захоплення ICR1H, L. При цьому встановлюється в одиничний стан розряд ICF1 регістра TIFR і при одиничному стані розряду TICI1 регістра TIMSK у блок переривань надходить запит переривання T/C1 CAPT.

Захоплення виконується при зміні значення зовнішнього сигналу, що надходить на вхід ICP, чи внутрішнього сигналу ACO, що надходить з аналогового компаратора. Вибір сигналу визначається станом розряду ACI регістра ACSR, що входить до складу аналогового компаратора. При ACI = 0 використовується зовнішній сигнал, при ACI = 1 – внутрішній. Вид зміни сигналу, при якому виконується захоплення, визначається станом розряду ICES1 регістра TCCR1B. При ICES1 = 0 захоплення виконується з появою негативного фронту сигналу, а при ICES1 = 1 – позитивного фронту.

Розряд ICNC1 регістра TCCR1B керує роботою схеми заглушування завад. При ICNC1 = 0 захоплення виконується з кожною появою фронту заданої полярності. При ICNC1 = 1 захоплення відбувається, якщо перед появою фронту протягом чотирьох тактів сигнал зберігає постійне значення.

Налаштування T/C1 на режим захоплення

```
TIM_CAPT1:
    LDI R16, 0
    OUT TCNT1H, R16
    OUT TCNT1L, R16
    OUT ICR1H, R16
    OUT ICR1L, R16
    OUT ACSR, R16
    OUT TIMSK, R16
    OUT TCCR1A, R16
    LDI R16,$01
    OUT TCCR1B, R16
    SEI
```

[TIMSK]

TOIE1	OCIE1A			TICIE1A		TOIE0	
0	0	-	-	0	-	0	-

[TIFR]

TOV1	OCF1A			ICF1		TOV0	
0	0	-	-	0	-	0	-

[TCCR1A]

COM1A1	COM1A0					PWM11	PWM10
0	0	-	-	-	-	0	0

[TCCR1B]

ICNC1	ICES1			CTC1	CS12	CS11	CS10
0	0	-	-	0	0	0	0

Схема керування СКЗ займається виконанням функції порівняння/ PWM. Функція порівняння полягає у видачі певного значення сигналу на виході OC1 при збігу кодів у базовому лічильнику та 16-розрядному регістрі порівняння OCR1H, L, яке виявляється за допомогою компаратора К. При цьому також встановлюється в одиничний стан розряд OCF1A регістра TIFR, і при одиничному стані розряду OCI1A регістра TIMSK у блок переривань надходить запит переривання T/C1 COMP.

Налаштування T/C1 на режим порівняння

```
TIM_C0MP1:
    LDI R16,0
    OUT TCNT1H,  R16
    LDI R16, $30
    OUT TCNT1L, R16
    LDI R16, $40
    OUT TCCR1A, R16
    LDI R16,1
    OUT TCCR1B, R16
    LDI R16, 0
    OUT OCR1AH, R16
    LDI R16, $47
    OUT OCR1AL, R16
    LDI  R16 ,$40
    OUT TIMSK,  R16
    SEI
```

Функція PWM полягає у видачі на вихід OC1 імпульсного сигналу з заданим періодом повторення і заданою тривалістю імпульсу. При цьому також періодично формується запит переривання T/C1 COMP.

Робота схеми СКЗ визначається комбінацією станів розрядів PWM10, PWM11, COM1A0 і COM1A1 регістра керування TCCR1A. При нульовому стані всіх чотирьох розрядів функція порівняння/PWM не виконується і вихід OC1 відключений від виводу порту.

При PWM10=0, PWM11=0 і інших комбінаціях станів розрядів COM1A0 і COM1A1 виконується функція порівняння. При виконанні функції порівняння режим роботи базового лічильника залежить від стану розряду CTC1 регістра керування TCCR1B. При CTC1=1 базовий лічильник при збігу кодів скидається в нульовий стан і продовжує рахунок, починаючи з 0.

При CTC1 = 0 він продовжує рахунок до переповнення і далі веде рахунок, починаючи з 0. При одиничному стані хоча б одного з розрядів PWM10 і PWM11 і одиничному стані розряду COM1A1 виконується функція PWM. У цьому випадку базовий лічильник веде підрахунок на додавання до одержання числа 255 або 511, або 1023, переходить у режим рахунку на віднімання, веде лічбу на вирахування до одержання числа 0 і знову повертається в режим рахунку на додавання. Вибір максимального

числа (Nmax), до якого ведеться рахунок на додавання, визначається комбінацією станів розрядів PWM11 і PWM10 регістра керування TCCR1A.

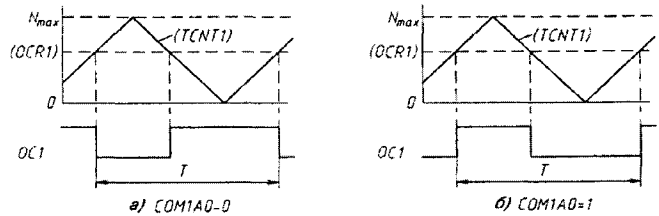


Рисунок 3.28 – Графіки зміни числа в базовому лічильнику (TCNT1) і часові діаграми сигналу PWM при різних станах розряду COM1AT

Сигнал PWM формується шляхом зміни значення сигналу на виході OC1 при збігу кодів у базовому лічильнику і регістрі OCR1 у процесі рахунка на додавання і на віднімання. Вигляд зміни сигналу залежить від стану розряду COM1A0. На рис. 3.28 зображені графіки зміни числа в базовому лічильнику (TCNT1) і часові діаграми сигналу PWM при різних станах розряду COM1A0.

Період сигналу PWM (T) залежить від максимального числа, до якого виконується рахунок на додавання. При Nmax = 255 період у 510 разів більше періоду проходження імпульсів на рахунковому вході базового лічильника. При Nmax=511 і 1023 це відношення дорівнює 1022 і 20461 відповідно.

Розряд CTC1 регістра TCCR1B у режимі PWM не використовується. Запит переривання T/C1 OVF формується при переході базового лічильника від числа 0 до числа 1. При записі коду в регістр OCR1 код запам'ятовується в регістрі тимчасового зберігання. Перезаписування коду в регістр OCR1 виконується з появою в базовому лічильнику максимального числа, що запобігає появі в сигналі PWM імпульсу з випадковою тривалістю.

Вивід порту PD5 використовується як вивід OC1 у мікроконтролера типу Atmega8515.

Налаштування T/C1 на режим PWM

```
PWM:
        LDI   R16,$C1
        OUT TCCR1A, R16
        LDI   R16,1
        OUT TCCR1B, R16
        LDI   R16, 0
        OUT OCR1AH, R16
        LDI   R16, $47
        OUT OCR1AL, R16
LOOP: RJMP LOOP
```

До програмно-доступних регістрів таймерів-лічильників ATmega8515 входять:

– регістри керування таймерами (TCCR0 для T/C0; TCCR1A, TCCR1B для T/C1);

– регістр маскування переривань (TIMSK);

– регістр ознак переривань (TIFR);

– лічильні регістри (TCNT0 для T/C0 та TCNT1L, TCNT1H для T/C1);

– регістри порівняння (OCR0 для T/C0, OCR1AL, OCR1AH, OCR1BL, OCR1BH для T/C1);

– регістри захоплення T/C1 (ICR1L, ICR1H).

Приклад програми вимірювання частоти генератора

Вимірять частоту генератора звукової частоти і вивести значення на статичний індикатор в кГц

```
.include «m8515def.inc»        ;підключення файлу визначень

.def store = R0                ;регістр для зберігання SREG
.def temp = R16                ;для тимчасових значень
.def f1 = R17                  ;для зберігання одиниць частоти
.def f2 = R18                  ;для зберігання десятків частоти
.def f3 = R19                  ;для зберігання сотень частоти
.def f4 = R20                  ;для зберігання тисяч частоти
.def f5 = R21                  ;для десятків тисяч частоти
.def ind1 = R22                ;число для виведення на індикатор
.def ind2 = R23                ;-//-


rjmp RESET                     ;на початок програми
.org $006                      ;вектор переривання за
rjmp TIM1_OVF                  переповненням ;таймера

TIM1_OVF:                      ;обробник переповнення таймера
in store, SREG                 ;зберегти SREG
ldi temp, high(36736)          ;відкалібрувати таймер на
out TCNT1H, temp               ;переривання раз в секунду
ldi temp, low(36736)
out TCNT1L, temp

mov ind1, f1                   ;переписати частоту в ind1 і ind2
swap f2
add ind1, f2
mov ind2, f3
```

```
    swap f4
    add ind2, f4
    ldi temp, $10
    cpi f5, 0
    breq indication          ;якщо частота менше 10 КГц
                             ;то вивести з точністю до
    ldi temp, 4              тисячних
    shift:
    lsr f5
    ror ind2                 ;інакше – з точністю до сотих.
    ror ind1
    dec temp
    brne shift
    ldi temp, $20

    indication:
    clr f1
    clr f2
    clr f3                   ;очистити регістри із значенням
    clr f4                   ;частоти
    clr f5
    sts $A004, temp
    sts $A000, ind2
    sts $A001, ind1          ;вивести на індикатор частоту

    out SREG, store
    reti
                             ;відновити SREG
    RESET:                   ;кінець обробника переривання
    ldi temp,
low(RAMEND)                  ;початок програми
    out SPL, temp            ;ініціалізація стека
    ldi temp,
high(RAMEND)
    out SPH, temp

    ldi temp, 1<<SRE         ;дозвіл роботи з зовнішнім
    out MCUCR, temp          адресним ;простором

                             ;запуск таймера з частотою
    ldi temp, $04            ;переривань в 1 с
    out TCCR1B, temp
```

```
ldi temp, (1<<TOIE1)
out TIMSK, temp

                        ;налаштувати порти
clr temp
out PORTD, temp
out DDRD, temp          ;та індикатори
sts $A004, temp
sts $8000, temp
sts $8001, temp
                        ;цикл опитування генератора
loop:                   ;дозволити переривання
sei                     ;якщо на виході генератора «1»,
sbic PIND, 5        то
clt                     ;скинути біт t (в SREG) і
sbic PIND, 5            ;повернутись на початок циклу
rjmp loop
brts loop               ;якщо «0», перевірити біт t. Якщо
set                     ;t=1 – на початок циклу
cli                     ;встановити t
inc f1                  ;заборонити переривання
cpi f1, 10              ;таким чином забезпечується
brlo loop           реакція
clr f1                  ;на фронт, що спадає
inc f2                  ;інкрементувати значення
cpi f2, 10          частоти в
brlo loop               ;десятковій формі
clr f2
inc f3
cpi f3, 10
brlo loop
clr f3
inc f4
cpi f4, 10
brlo loop
clr f4
inc f5
rjmp loop
```

УВАГА!!! Сигнал з генератора звукової частоти (ГЗЧ) надходить на PD5, з генератора фіксованої частоти (ГФЧ) – на PD4.

Таблиця 3.14 – Варіанти індивідуальних завдань

№ варіанта	Текст індивідуального завдання
1	Виміряти частоту ГФЧ, вивести її значення на статичний індикатор.
2	Виміряти частоту ГЗЧ відносно ГФЧ, вивести на статичний індикатор.
3	Виміряти частоту ГЗЧ відносно частоти роботи МК, вивести на статичний індикатор.
4	Виміряти різницю частот ГЗЧ і ГФЧ. Вивести на статичний індикатор.
5	З використанням внутрішніх Т/Л забезпечити плавне загоряння числа 55 на статичному індикаторі.
6	Почергово відображати 48 на другому і третьому знакомісцях статичної індикації і на першому і четвертому знакомісцях статичної індикації з інтервалом 1 с. Часовий інтервал сформувати за допомогою ГФЧ.
7	Підрахувати і вивести на динамічний індикатор кількість натискань SW3 за 10 с. Часовий інтервал сформувати за допомогою Т/С1 мікроконтролера.
8	Відображати на другому знакомісці статичної індикації числа від 1 до 9. Кожну цифру відображати протягом 1 сек., пауза між відображеннями – 1 сек. Часові інтервали формувати за допомогою Т/С1.
9	За натисканням кнопки SW4 включити секундомір з відображенням на динамічному індикаторі.
10	Відображати числа, що мигають, на чотирьох розрядах статичної індикації з різним часом відображення і паузи (0.25, 0.5, 0.75, 1 с. відповідно до кожного розряду). Часові інтервали задавати за допомогою Т/С1.
11	Підрахувати кількість натискань кнопки SW6 за 5 с. Часовий інтервал задати за допомогою Т/С1.
12	Виміряти час між натисканнями кнопок SW7 і SW8 в секундах та відобразити на статичному індикаторі.
13	При натисканні кнопки SW9 відобразити на статичному індикаторі в секундах час її натискання.
14	Виміряти період ГФЧ та відобразити на статичному індикаторі.
15	Виміряти період ГЗЧ та відобразити на статичному індикаторі.

У додатках Б–П показані практичні приклади застосування наведених у посібнику теоретичних даних.

ВИСНОВКИ

Підготовка технічних спеціалістів неможлива без організації й проведення лабораторних практикумів з базових навчальних дисциплін природничо-наукової, загальної професійної й спеціальної підготовки. У той же час добре відомо, що лабораторні роботи є найбільш дорогим видом навчальних занять, організація якого на сучасному рівні виявляється практично недоступної з економічних причин для більшості навчальних закладів в Україні.

Застосування стенда EV8031/AVR для проведення лабораторних робіт з дисциплін «Комп'ютерна електроніка», «Комп'ютерна схемотехніка», «Архітектура комп'ютерів», «Моделювання компонентів комп'ютерних сиситем» дасть змогу студентам на практиці набути навички в програмуванні мікроконтролерів, а використання розробленого програмного забезпечення пришвидшить цей процес.

Звичайно, наведені програми не є еталонними. Програміст самостійно приймає рішення, яким способом краще реалізувати той чи інший задум, а подані програми стануть у пригоді як приклади.

При потребі будь-яку програму можна модернізувати відповідно до потреб.

Запропонований підхід поєднання реальних і віртуальних аналогів стендів та вимірювального обладнання дозволяє покращити якість практико-орієнтованої підготовки фахівців за спеціальністю «Комп'ютерна інженерія» у технічних коледжах та університетах України.

ЛІТЕРАТУРА

1. Open System [Електронный ресурс] / Учебно-отладочный стенд EV8031/AVR – Режим доступа : http://opensys.com.ua/Stend/Ev8031, вільний. – Загл. з екрана. – Мова рос.

2. Белов А. В. Самоучитель разработчика устройств на микроконтроллерах AVR / Белов А. В. – СПб. : Наука и Техника, 2008. – 544 с. – ISBN 978-5-94387-363-8.

3. Богатырев Е. А. Энциклопедия электронных компонентов. Большие интегральные схемы / Е. А. Богатырев, В. Ю. Ларин, А. Е. Лякин ; под ред. А. Н. Еркина. – Т. 1. – М. : ООО «МАКРО ТИМ», 2006. – 224 с. – ISBN 5-9900833-1-9 (978-5-9900833-1-8).

4. Болл С. Р. Аналоговые интерфейсы микроконтроллеров / С. Р. Болл – М. : Издательский дом «Додэка-XXI», 2007. – 360 с. – ISBN 978-5-94120-142-6.

5. Евстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL» / А. В. Евстифеев. – М. : Издательский дом «Додэка-XXI», 2004. – 560 с. – ISBN 5-94120-081-1.

6. Корнєв В. VIII Всеукраїнська відкрита студентська олімпіада з прикладного програмування для мікропроцесорних систем/ В. Корнєв, В. Собченко // CHIP NEWS Украина / Инженерная микроэлектроника. – 2010. – № 3. – С. 72–74.

7. Краткий учебный курс PROTEUS [Електронний ресурс] / Русское руководство для начинающих. – Режим доступу : http://proteus123.narod.ru, вільний. – Загл. з екрана. – Мова рос.

8. Максимов А. Моделирование устройств на микроконтроллерах с помощью программы ISIS из пакета PROTEUS VSM / А. Максимов // Радио. – 2005. – № 4, 5, 6. – С. 30–33, 31–34, 30–32.

9. Радиокот [Електронний ресурс] / Proteus – первое знакомство. – Режим доступу : http://radiokot.ru/start/soft/proteus/01, вільний. – Загл. з екрана. – Мова рос.

10. Схемотехніка електронних систем : у 3 кн. / Кн. 3. Мікропроцесори та мікроконтролери : підручник / [В. І. Бойко, А. М. Гуржій, В. Я. Жуйков та ін.]. – К. : Вища шк., 2004. – 399 с. – ISBN 966-642-193-3.

11. Трамперт В. AVR-RISC микроконтроллеры / В. Трамперт ; пер. с нем. – К. : МК-Прес, 2006. – 464 с. – ISBN 966-8806-07-7, 3-7723-5476-9.

12. Трамперт В. Измерение, управление и регулировка с помощью AVR микроконтроллеров / В. Трамперт ; пер. с нем. – К. : МК-Пресс, 2006. – 208 с. – ISBN 966-8806-14-X.

13. Хартов В. Я. Микроконтроллеры AVR. Практикум для начинающих / В. Я. Хартов. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2007. – 240 с.

14. Цирульник С. М. Автоматизація проектування мікропроцесорних систем контролю доступу та охорони / С. М. Цирульник, С. І. Перевозніков, В. С. Озеранський //Вісник Вінницького політехнічного інституту. – 2009. – № 1.– С. 10–15.

15. Цирульник С. М. Застосування програми ISIS пакета Proteus VSM при вивченні курсу «Мікропроцесорна техніка» // С. М. Цирульник, В. К. Задорожний : матеріали XIII міжнародної конференції з автоматики управління (Автоматика 2006). – Вінниця : Універсум-Вінниця, 2007. – С. 526–530. – ISBN 978-966-641-210-5.

16. Цирульник С. М. Комп'ютеризований лабораторний віртуальний стенд / С. М. Цирульник, В. І. Роптанов / Вісник Вінницького політехнічного інституту. – 2010. – № 4.– С. 94–98

17. Теоретичні основи комп'ютерних напівпровідникових електронних компонентів : навч. посіб. / Азаров О. Д., Гарнага В. А., Сапсай Т. Г., Тарасенко В. П. – Вінниця : ВНТУ, 2015. – 134 с.

18. Комп'ютерні мережі : навч. посіб. / О. Д. Азаров, С. М. Захарченко, О. В. Кадук та ін. – Вінниця : ВНТУ, 2013. – 500 с.

19. Багаторозрядні АЦП і ЦАП із ваговою надлишковістю, стійкі до параметричних відмов : монографія / О. Д. Азаров, О. В. Кадук. – Вінниця : ВНТУ, 2010. – 150 с.

20. Двотактні підсилювачі постійного струму для багаторозрядних перетворювачів форми інформації, що самокалібруються : монографія / О. Д. Азаров, В. А. Гарнага. – Вінниця : ВНТУ, 2011. – 156 с.

21. Багатоканальні ІВС опрацювання стрибкоподібних сигналів на базі АЦП із ваговою надлишковістю : монографія / Азаров О. Д., Снігур А. В. – Вінниця : УНІВЕРСУМ-Вінниця, 2008. – 138 с.

22. Інформаційна технологія доставки контенту у системі комп'ютеризованої підготовки спеціалістів / О. І. Гороховський, О. Д. Азаров, Т. І. Трояновська. – Вінниця : ВНТУ, 2016. – 160 с.

23. Азаров О. Д. Аналого-цифрові пристрої систем, що самокоригуються, для вимірювань і обробляння низькочастотних сигналів : монографія / О. Д. Азаров, Л. В. Крупельницький. – Вінниця : УНІВЕРСУМ-Вінниця, 2005. – 167 с.

24. Азаров О. Д. Самокалібровані АЦП із накопиченням заряду на основі надлишкових позиційних систем числення / Азаров О. Д., Захарченко С. М., Харьков О. М. – Вінниця : УНІВЕРСУМ-Вінниця, 2005. – 235 с.

25. Азаров О. Д. Обчислювальні АЦП і ЦАП, що самокалібруються, для систем цифрового обробляння аналогових сигналів : монографія / О. Д. Азаров, О. О. Коваленко. – Вінниця : УНІВЕРСУМ-Вінниця, 2006. – 146 с.

26. Азаров О. Д. Аналого-цифрове порозрядне перетворення на основі надлишкових систем числення з ваговою надлишковістю : монографія / Азаров О. Д. – Вінниця : ВНТУ, 2010. – 232 с.

27. Азаров О. Д. Методи та засоби високоточного слідкувального аналого-цифрового перетворення з ваговою надлишковістю. : монографія / О. Д. Азаров, О. В. Дудник. – Вінниця : ВНТУ, 2014. – 120 с.

Додаток А

Лабораторний макет EV8031. Схема електрична принципова

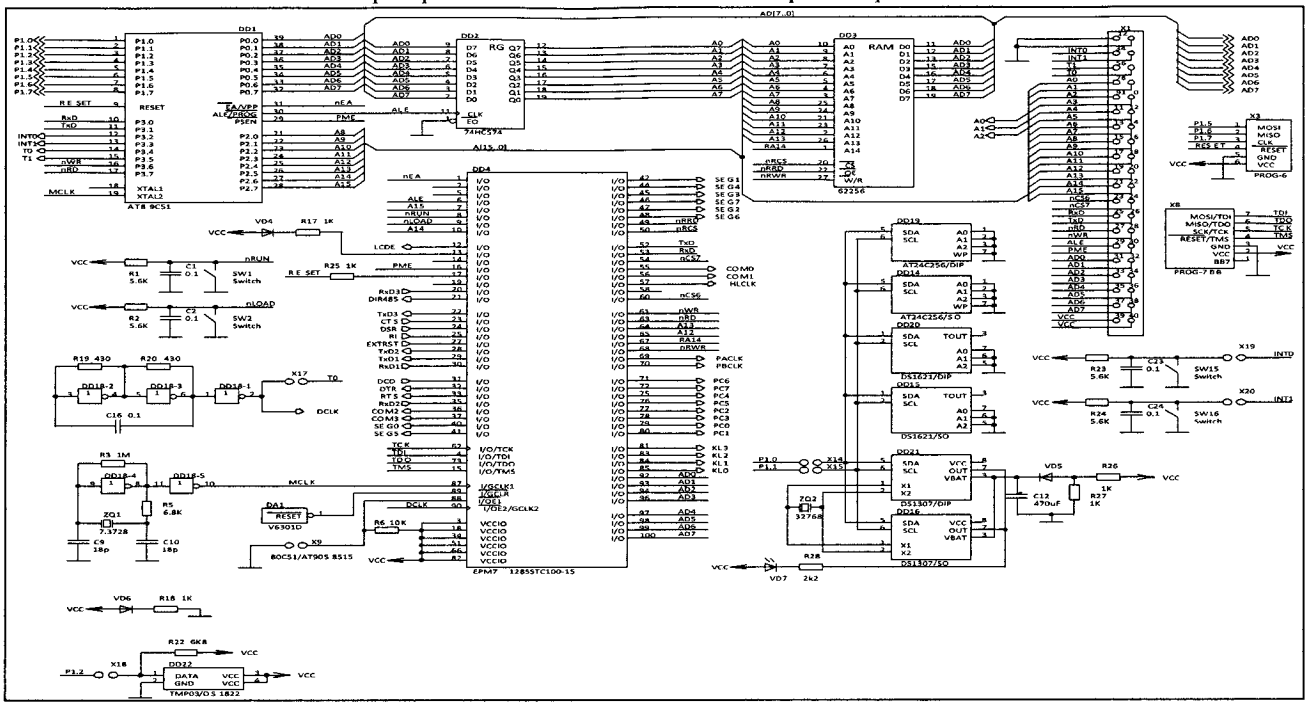


Рисунок А.1 – Лабораторний макет EV8031/AVR. Плата основна

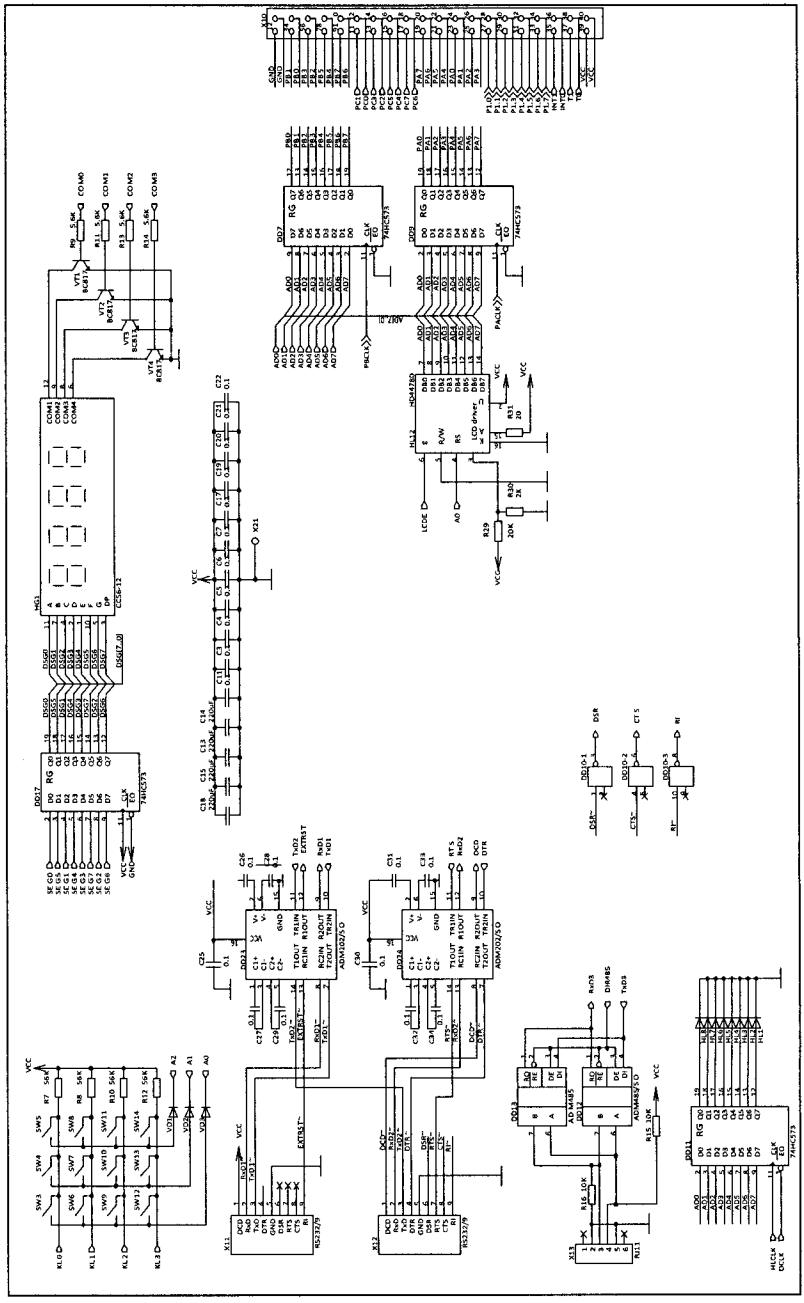


Рисунок А.2 – Лабораторний макет EV8031/AVR. Плата розширення EV8031/AN

Додаток Б

Програма «Вогник, що біжить»

Програма демонструє використання дискретного індикатора. Алгоритм програми простий – на індикатор виводиться число і зсувається на один розряд вліво, потім процедура повторюється. Візуально це спостерігається як почергове засвічування кожного світлодіода, або ж «вогник, що біжить».

Лістинг програми «Вогник, що біжить» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

char *LED_REG=(char *)0xa006;          //Адреса дискретного індикатора

void main()
{
char i=1;                              //Робоча змінна
  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн. пам'яттю
  while(1)                             //Нескінченний цикл
  {
        if(i==0)
              i=1;
        *LED_REG=i;                    //Виведення числа з «i» на індикатор
        i=i<<1;                        //Логічний зсув значення «i» вліво
        _delay_ms(1000);               //Затримка
  }
}
```

Лістинг програми «Вогник, що біжить» мовою Assembler

```
.include <m8515def.inc>                ;Файл з описом портів МК
.def temp       = r16                  ;Робоча змінна
.equ LED_REG = $a006                   ;Виразу «LED_REG» присвоєно
                                     адресу індикатора
                                       ;Ініціалізація
reset:
  ldi    temp,low(RAMEND)              ;Ініціалізація стека
  out    SPL,temp
  ldi    temp,high(RAMEND)
  out    SPH,temp
```

```
  ldi   temp,(1<<SRE)          ;Дозвіл роботи із зовн. пам'яттю
  out   MCUCR,temp

  ldi   zh,high(LED_REG)       ;Завантаження адреси в регістр z
  ldi   zl,low(LED_REG)

  ldi   temp,1

                               ;Головна програма
start:
  tst   temp                   ;Якщо значення «temp»==0,
                               «temp» =1
  brne  m1
  ldi   temp,1
m1:
  st    z,temp                 ;Запис числа з «temp» за адресою,
                               що знаходиться в z
  lsl   temp                   ;Логічний зсув значення в «temp»
  rcall delay                  ;Виклик підпрограми затримки
  rjmp  start                  ;Нескінченний цикл

delay:                         ;Підпрограма затримки
  ser   r17
  ser   r18
  ldi   r19,5
m2:
  dec   r17
  brne  m2
  dec   r18
  brne  m2
  dec   r19
  brne  m2

  ret
```

Додаток В

Програма «Лічильник»

Програма «Лічильник» використовує молодший розряд статичного індикатора і кнопки SW15, SW16. Спочатку на індикатор виводиться число 0. При натисканні кнопки SW15 значення на індикаторі збільшується на 1, а при натисканні кнопки SW16 – зменшується на 1.

Лістинг програми «Лічильник» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

#define SW15 (PIND & 4) == 0           //«SW15» означає стан кнопки SW15
                                       «Натиснута»
#define SW16 (PIND & 8) == 0           //«SW16» означає стан кнопки SW16
                                       «Натиснута»

char *righti=(char *)0xa001;           //Адреса молодшого розряду стат.
                                       індикатора

void main()
{
  char i=0;                            //Робоча змінна
  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн. пам'яттю

  while(1)
  {
       if (SW15)                       //Якщо натиснуто кнопку SW15,
            i++;                       //збільшити на 1 значення змінної «i»
            if (SW16)                  //Якщо натиснуто кнопку SW16,
                       i--;            //зменшити на 1 значення змінної «i»

       *righti=i;                      //Вивести на індикатор значення змінної «i»
       _delay_ms(1300);                //Затримка
  }
}
```

Лістинг програми «Лічильник» мовою Assembler

```
.include <m8515def.inc>                ;Файл з описом портів МК
.def temp      =r16                    ;Робоча змінна
```

```
.equ righti     =$a001          ;Адреса молодшого розряду стат. індикатора

.equ SW15       =2              ;Виразу «SW15» присвоєно значення 2
.equ SW16       =3              ;Виразу «SW16» присвоєно значення 3

                                ;Ініціалізація
reset:
   ldi    temp,(1<<SRE)         ;Дозвіл роботи із зовн. пам'яттю
   out    MCUCR,temp

   ldi    temp,0b11110011       ;Конфігурація порту D МК
   out    DDRD,temp

   ldi    xh,high(righti)       ;Завантаження адреси розряду індикатора в регістр x

   ldi    xl,low(righti)

   clr    temp                  ;Початкове значення

                                ;Головна програма
start:
   sbic   PIND,SW16             ;Якщо натиснуто кнопку SW16,
   inc    temp                  ;інкремент змінної «temp»

   sbic   PIND,SW15             ;Якщо натиснуто кнопку SW16,
   dec    temp                  ;декремент змінної «temp»

   st     x,temp                ;Вивести значення змінної «temp» на індикатор

   ser    r17                   ;Затримка
   ser    r18
   ldi    r19,7
m1:
   dec    r17
   brne   m1

   dec    r18
   brne   m1

   dec    r19
   brne   m1
   rjmp   start                 ;Нескінченний цикл
```

Додаток Г

Програма «Змійка»

Програма «Змійка» демонструє використання динамічної індикації. По черзі на кожен розряд динамічного індикатора виводиться зображення – по контуру індикатора по черзі засвічуються сегменти. При цьому створюється ефект «змійки».

Лістинг програми «Змійка» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()
char *pB=(char *)0x8001;               //Адреса порту pB
char *pC=(char *)0x8002;               //Адреса порту pC

char digit[12]={1,1,1,1,2,4,8,8,8,8,16,32};    //Таблиця кодів зображення
                                               дин. індикатора
char row[12]=  {0,1,2,3,3,3,3,2,1,0,0,0};      //Таблиця номерів розрядів
                                               дин. індикатора

void main()
{
  char i;                              //Робоча змінна
  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн. пам'яттю

  while(1)                             //Нескінченний цикл
  {
        for(i=0;i<12;i++)
        {
              *pC=row[i];              //Вивести номер розряду на порт pC
              *pB=digit[i];            //Вивести значення розряду на порт pB
              _delay_ms(400);          //Затримка
        }
  }
}
```

Лістинг програми «Змійка» мовою Assembler

```
.include <m8515def.inc>          ;Файл з описом портів МК

.def temp =r16                   ;Робоча змінна
.def cnt1 =r18                   ;Допоміжна змінна
.def cnt2 =r19                   ;Допоміжна змінна
.def cnt3 =r20                   ;Допоміжна змінна
```

```
.equ HL1d =$8001                ;Адреса порту pB
.equ HL1r =$8002                ;Адреса порту pC

                                ;Таблиця кодів зображення, збережена
                                у FLASH-пам'яті МК
segm:
  .db  1,0,1,1,1,2,1,3,2,3,4,3,8,3,8,2,8,1,8,0,16,0,32,0

                                ;Ініціалізація
reset:
  ldi   temp,(1<<SRE)           ;Дозвіл роботи із зовн. пам'яттю
  out   MCUCR,temp

  ldi   xh,high(HL1r)           ;Завантаження адреси порту pC в
                                регістр y
  ldi   xl,low(HL1r)

  ldi   yh,high(HL1d)           ;Завантаження адреси порту pB в
                                регістр z
  ldi   yl,low(HL1d)

                                ;Головна програма
start:
  ldi   zh,high(segm*2)         ;Завантаження адреси таблиці кодів в
                                регістр z
  ldi   zl,low(segm*2)
m1:
  lpm   temp,z+                 ;Завантаження коду зображення з
                                комірки FLASH
  st    y,temp                  ;Виведення коду зображення на порт pB

  lpm   temp,z+                 ;Завантаження номера розряду з
                                FLASH-пам'яті
  st    x,temp                  ;Виведення номеру розряду на порт pC
  ldi   cnt3,3
m2:
  dec   cnt1                    ;Затримка
  brne  m2
  dec   cnt2
  brne  m2
  dec   cnt3
  brne  m2
  cpi   zl,segm*2+24            ;Якщо виведено зображення на
                                останній розряд,
  brne  m1                      ;повторити спочатку
rjmp    start                   ;Нескінченний цикл
```

Додаток Д

Програма «Сканер»

Програма передбачає використання світлодіодної матриці. Засвічуються стовпці матриці справа-наліво, потім – рядки згори-донизу, створюючи ефект роботи сканера.

Лістинг програми «Сканер» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

char *pA=(char *)0x8000;               //Адреса порту pA
char *pC=(char *)0x8002;               //Адреса порту pC

void main()
{
  char i;                              //Робоча змінна
  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн. пам'яттю

  while(1)
  {
        i=1;
        *pC=0;                         //Вивести на матрицю зображення
                                       стовпця
        while(i<32)
        {
              *pA=i;
            i=i<<1;                    //Змістити зображення по горизонталі
              _delay_ms(500);          //Затримка
        }

        i=1;
        *pA=255;                       //Вивести на матрицю зображення
                                       рядка
        while(i<128)
        {
              *pC=~i;                  //Змістити зображення по вертикалі
              i=i<<1;
              _delay_ms(500);
        }
  }
}
```

```
.include <m8515def.inc>          ;Файл з описом портів МК
.def temp =r16                   ;Робоча змінна
.def cnt =r20                    ;Допоміжна змінна
.equ strM =$8000                 ;Адреса порту pA
.equ rowM =$8002                 ;Адреса порту pC

                                 ;Ініціалізація
reset:
  ldi   temp,low(RAMEND)         ;Ініціалізація стека
  out   SPL,temp        ;
  ldi   temp,high(RAMEND);
  out   SPH,temp        ;

  ldi   temp,(1<<SRE)            ;Дозвіл роботи із зовн. пам'яттю
  out   MCUCR,temp

  ldi   xh,high(rowM)            ;Завантаження адреси порту pC в
                                 регістр x
  ldi    xl,low(rowM)

  ldi   yh,high(strM)            ;Завантаження адреси порту pA в
                                 регістр y
  ldi   yl,low(strM)

                                 ;Головна програма
start:
  clr   temp                     ;Вивести на матрицю зображення
                                 стовпця
  st    y,temp
  ldi   cnt,1
m1:
  st    x,cnt                    ;Номер наступного стовпця
  rcall delay                    ;Викликати підпрограму затримки

  cpi   cnt,32                   ;Якщо не досягнуто кінця
                                 матриці, повторити
  brne  m1

  ser   temp
  st    x,temp                   ;Вивести на матрицю зображення
                                 рядка
  ldi   cnt,1
```

```
m2:
   mov   temp,cnt
   com   temp
   st    y,temp          ;Номер наступного рядка
   rcall delay           ;Викликати підпрограму затримки

   cpi   cnt,128         ;Якщо не досягнуто кінця
                         матриці, повторити
   brne  m2
   rjmp  start           ;Нескінченний цикл

                         ;Підпрограма затримки
delay:
   lsl   cnt             ;Логічний зсув значення змінної
                         cnt вліво
   ser   r17
   ser   r18
   ldi   r19,$04
m3:
   dec   r17
   brne  m3
   dec   r18
   brne  m3
   dec   r19
   brne  m3
   ret
```

Додаток Е

Програма «Друкарська машинка»

Програма передбачає використання світлодіодної матриці. Засвічується крапка, потім її положення зміщується зліва-направо та згори-донизу.

Лістинг програми «Друкарська машинка» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

char *pA=(char *)0x8000;               //Адреса порту pA
char *pC=(char *)0x8002;               //Адреса порту pC

void main()
{
  char i;                              //y-координата крапки
  char j;                              //x-координата крапки

  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн. пам'яттю

  while(1)
  {
      for(i=64; i>0; i=i>>1)           //Виведення крапки по вертикалі
      {
          *pC=~i;
          for(j=32; j>0; j=j>>1)       //Виведення крапки по
                                       горизонталі
          {
              *pA=j;
              _delay_ms(400);
          }
      }
  }
}
```

Лістинг програми «Друкарська машинка» мовою Assembler

```
.include <m8515def.inc>                ;Файл з описом портів МК

.def temp =r16                         ;Робоча змінна
.def strR =r20                         ;x-координата крапки
```

```
.def rowR =r21                    ;y-координата крапки

.equ strM =$8000                  ;Адреса порту pA
.equ rowM =$8002                  ;Адреса порту pC

                                  ;Ініціалізація
reset:
  ldi   temp,low(RAMEND)          ;Ініціалізація стека
  out   SPL,temp                  ;
  ldi   temp,high(RAMEND)         ;
  out   SPH,temp                  ;

  ldi   temp,(1<<SRE)             ;Дозвіл роботи із зовн. пам'яттю
  out   MCUCR,temp

  ldi   xh,high(strM)             ;Завантаження адреси порту pC
                                  в регістр x
  ldi   xl,low(strM)

  ldi   yh,high(rowM)             ;Завантаження адреси порту pA
                                  в регістр y
  ldi   yl,low(rowM)

                                  ;Головна програма
start:
  ldi   strR,64                   ;Початкова координата крапки
m1:
  tst   strR
  breq  start

  mov   temp,strR
  com   temp
  st    y,temp                    ;Вивести дані про зображення
                                  на порт pA
  lsr   strR                      ;Зсунути крапку вправо
                                  (наступний стовпець)

  ldi   rowR,16
m2:
  tst   rowR
  breq  m1

  st    x,rowR                    ;Вивести дані про зображення
                                  на порт pC
```

```
  lsr   rowR                    ;Зсунути крапку вниз
                                (наступний рядок)
  rcall delay

  rjmp  m2

                                ;Підпрограма затримки
delay:
  ser   r17
  ser   r18
  ldi   r19,3
m3:
  dec   r17
  brne  m3
  dec   r18
  brne  m3
  dec   r19
  brne  m3

  ret
```

Додаток Ж

Програма «Рулетка»

Програма передбачає використання світлодіодної матриці, засвічується крапка та «бігає» по контуру матриці.

Лістинг програми «Рулетка» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

char *pA=(char *)0x8000;               //Адреса порту pA
char *pC=(char *)0x8002;               //Адреса порту pC

void main()
{
  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн. пам'яттю

  while(1)
  {
        *pC=63;                        //Початкові координати крапки
        for(j=16; j>0; j=j>>1)         //Зміщення зображення крапки
                                       зліва-
                                       //направо
        {
              *pA=j;
              _delay_ms(500);
        }

        for(i=64; i>0; i=i>>1)         //Зміщення зображення крапки згори-
                                       //вниз
        {
              *pC=~i;
              _delay_ms(500);
        }
        for(j=1; j<32; j=j<<1)         //Зміщення зображення крапки справа-
                                       //наліво
        {
              *pA=j;
              _delay_ms(500);
        }
        for(i=1; i<64; i=i<<1)         //Зміщення зображення крапки знизу-
                                       //вгору
        {
              *pC=~i;
              _delay_ms(500);
        }
  }
}
```

Лістинг програми «Рулетка» мовою Assembler

```
.include <m8515def.inc>                ;Файл з описом портів МК

.def temp =r16                         ;Робоча змінна
.def cnt  =r20                         ;Допоміжна змінна

.equ strM =$8000                       ;Адреса порту pA
.equ rowM =$8002                       ;Адреса порту pC

                                       ;Ініціалізація
reset:
   ldi    temp,low(RAMEND)             ;Ініціалізація стека
   out    SPL,temp          ;
   ldi    temp,high(RAMEND);
   out    SPH,temp          ;

   ldi    temp,(1<<SRE)                ;Дозвіл роботи із зовн. пам'яттю
   out    MCUCR,temp

   ldi    xh,high(strM)                ;Завантаження адреси порту pA
                                       в регістр x
   ldi    xl,low(strM)

   ldi    yh,high(rowM)                ;Завантаження адреси порту pC
                                       в регістр y
   ldi    yl,low(rowM)

   ldi    cnt,32

                                       ;Головна програма
start:
   ldi    temp,63                      ;Початкові координати
   st     y,temp
right:                                 ;Виведення зображення крапки
                                       зліва-направо
   lsr    cnt
   rcall  horiz
   sbrs   cnt,0
   rjmp   right
   ldi    cnt,64
down:                                  ;Виведення зображення крапки
                                       згори-вниз
   lsr    cnt
```

```
   rcall  vert
   sbrs   cnt,0
   rjmp   down
   ldi    cnt,1
left:                          ;Виведення зображення крапки
                               справа-наліво
   lsl    cnt
   rcall  horiz
   sbrs   cnt,4
   rjmp   left
   ldi    cnt,1
up:                            ;Виведення зображення крапки
                               знизу-вгору
   lsl    cnt
   rcall  vert
   sbrs   cnt,5
   rjmp   up
   rjmp   start                ;Нескінченний цикл
horiz:                         ;Підпрограма для виведення
                               зображення рядка
   st     x,cnt
   rcall  delay
   ret
vert:                          ;Підпрограма для виведення
                               зображення стовпця
   mov    temp,cnt
   com    temp
   st     y,temp

                               ;Підпрограма затримки
delay:
   ser    r17
   ser    r18
   ldi    r19,3
m3:
   dec    r17
   brne   m3
   dec    r18
   brne   m3
   dec    r19
   brne   m3

   ret
```

Додаток И

Програма «Сирена»

На вхід ЦАП надходять числа, що зростають від 0 до 255, потім зменшується в зворотній послідовності. На виході ЦАП створюється пилкоподібна напруга, яка відворюється динаміком.

Лістинг програми «Сирена» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <avr/interrupt.h>             //Бібліотека функцій, що
                                       реалізовують переривання МК

char *pDAC = (char *)0xF000;           //Адреса вхідних ліній ЦАП
char *righti = (char *)0xA001;         //Адреса молодшого розряду
                                       статичного індикатора

char y;                                //Робоча змінна
char i=255;                            //Значення тональності звуку
char z=0;                              //Напрямок зміни тональності
                                       (зростання–спадання)

                                       //Переривання при переповненні
                                       лічильника TIMER1
                                       //Залежно від напрямку, змінити
                                       тональність
SIGNAL(SIG_OVERFLOW1)
{
  switch(z)
  {
        case 0:                        //Якщо z == 0 – зменшити
                                       тональність
        {
              i--;
              if(i==210) z=1;          //Якщо досягнуто межі, змінити
                                       напрямок
        }break;
        case 1:                        //Якщо z == 1 – збільшити
                                       тональність
        {
              i++;
              if(i==255) z=0;          //Якщо досягнуто межі, змінити
                                       напрямок
        }
  }
}
```

```
void main ()
{
  MCUCR = (1<<SRE);                    //Дозвіл роботи із зовн.
                                       пам'яттю
  TIMSK = (1<<TOIE1);                  //Дозвіл переривань від
                                       лічильника TIMER1
  TCCR1B = (1<<CS10);                  //Запуск лічильника TIMER1

  while(1)
  {
       cli();                          //Заборона переривань
       for(y=0; y<i; y++)
       {
            *pDAC = y;                 //Виведення значення у на ЦАП
            *righti=i;                 //Виведення значення
                                       тональності (i) на стат.
                                       //індикатор
       }
       sei();                          //Дозвіл переривань
  }
}
```

Лістинг програми «Сирена» мовою Assembler

```
.include <m8515def.inc>                ;Файл з описом портів МК

.def temp =r16                         ;Робоча змінна
.def dac_r =r17                        ;Регістр даних для запису в
                                       ЦАП
.def i =r18                            ;Допоміжна змінна

.equ dac =$f000                        ;Адреса вхідних ліній ЦАП
.equ righti =$a001                     ;Адреса молодшого розряду
                                       статичного індикатора


                                       ;Адреси векторів переривань
                                       МК
.org 0                                 ;Переривання при скиданні
                                       МК
  rjmp  reset
.org 6                                 ;Переривання при переповненні
                                       лічильника TIMER1
  rjmp  tim1_ovf
```

```
                                        ;Ініціалізація
reset:
    ldi   temp,low(RAMEND)              ;Ініціалізація стека
    out   SPL,temp          ;
    ldi   temp,high(RAMEND);
    out   SPH,temp          ;

    ldi   temp,(1<<SRE)                 ;Дозвіл роботи із зовн.
                                        пам'яттю
    out   MCUCR,temp

    ldi   temp,(1<<TOIE1)               ;Дозвіл переривань від
                                        лічильника TIMER1
    out   TIMSK,temp

    ldi   temp,(1<<CS11)                ;Запуск лічильника TIMER1
    out   TCCR1B,temp

    ldi   xh,high(righti)               ;Завантаження адреси
                                        молодшого розряду в
                                        ;регістр x
    ldi   xl,low(righti)

    ldi   yh,high(dac)                  ;Завантаження адреси
                                        вхідних ліній ЦАП в
                                        ;регістр y
    ldi   yl,low(dac)

    ser   i

                                        ;Головна програма
start:
    cli                                 ;Заборона переривань
    clr   dac_r
m1:
    st    y,dac_r                       ;Запис значення dac_r в
ЦАП
    st    x,i                           ;Запис значення
                                        тональності (i) за адресою x
    inc   dac_r                         ;Змінити значення dac_r
(від 0 - i)
    cp    dac_r,i                       ;Поки dac_r <i
    brlo  m1                            ;Повторити вищезазначені
                                        операції
```

```
  sei                          ;Дозволити переривання
  rjmp  start                  ;Нескінченний цикл

                               ;Переривання при
                               переповненні лічильника
                               TIMER1
                               ;Залежно від напрямку
                               змінити тональність
                               ;Напрямок зміни
                               визначається станом біта «t»
                               регістра SREG
tim1_ovf:
  in    temp,SREG              ;Зчитати значення біта
                               «t» регістра SREG

  sbrc  temp,6
  rjmp  m2

  dec   i                      ;Якщо біт «t» = 1,
                               збільшити «i» на 1
  cpi   i,220
  brge  m3

  set
  rjmp  m3
m2:
  inc   i                      ;Інакше зменшити «i»
                               на 1
  cpi   i,255
  brlo  m3
  clt
m3:
reti
```

Додаток К

Програма «Вольтметр»

Інверсний вхід компаратора перемичкою «J5» підключено до потенціометра R19, з якого знімається вимірювана напруга і надходить на інверсний вхід компаратора. Якщо ж замкнути перемичку «J4», напруга буде подаватись на вхід компаратора через «J2». За допомогою мікроконтролера реалізується програмний лічильник, з виходу якого двійкове число надходить на вхід ЦАП. На виході ЦАП встановлюється його аналоговий еквівалент. Коли напруга на прямому вході компаратора зрівняється з вимірюваною напругою на його прямому вході, компаратор спрацює, на його виході встановиться низький рівень сигналу. При спрацьовуванні компаратора встановлюється низький рівень сигналу на виводі PINB7 мікроконтролера. Візуально це можна спостерігати при засвічуванні світлодіода. Стан виводу PINB7 перевіряється у перериваннях від лічильника мікроконтролера T/C0, також при цьому відбувається збільшення значення програмного лічильника. Як тільки на виводі PINB7 встановиться низький рівень сигналу, лічба припиняється, значення програмного лічильника виводиться на статичний індикатор у вигляді 16-го числа. Потім програмний лічильник обнуляється, і процес повторюється спочатку. На індикаторі відображається не значення поданої напруги, а просто її цифровий еквівалент.

Лістинг програми «Вольтметр» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <avr/interrupt.h>             //Бібліотека функцій, що
                                     реалізовують переривання МК

char *pDAC = (char *)0xf000;           //Адреса вхідних ліній ЦАП
char *righti = (char *)0xa001;         //Адреса молодшого розряду
                                     статичного індикатора

char y=0;                              //Робоча змінна

                                       //Переривання при переповненні
                                     лічильника TIMER1
                                       //Сканування виводу PINB7,
                                     інкремент робочої змінної
SIGNAL(SIG_OVERFLOW0)
{
  if(!(PINB & 0b10000000))             //Якщо PINB7 == «0»,
  {
```

```
        *righti=y;                  //вивести значення у на індикатор,
        y=0;                        //обнулити у
    }

    y++;                            //Інкремент робочої змінної
}

void main ()
{
    MCUCR = (1<<SRE);               //Дозвіл роботи із зовн. пам'яттю
    TIMSK = (1<<TOIE0);             //Дозвіл переривань від лічильника
                                    TIMER0
    TCCR0 = (1<<CS10);              //Запуск лічильника TIMER1

    sei();                          //Дозвіл переривань

    while(1)                        //Нескінченний цикл
        *pDAC = y;                  //Виведення значення у на ЦАП
}
```

Лістинг програми «Вольтметр» мовою Assembler

```
.include <m8515def.inc>             ;Файл з описом портів МК

.def temp   =r16                    ;Робоча змінна
.def dac_r  =r17                    ;Допоміжна змінна

.equ dac    =$f000                  ;Адреса вхідних ліній ЦАП
.equ righti =$a001                  ;Адреса молодшого розряду стат.
                                    індикатора

                                    ;Адреси векторів переривань МК
.org 0                              ;Переривання при скиданні МК
    rjmp  reset
.org 7                              ;Переривання при переповненні
                                    лічильника TIMER0
    rjmp  tim0_ovf

                                    ;Ініціалізація
reset:
    ldi   temp,low(RAMEND)          ;Ініціалізація стека
    out   SPL,temp          ;
    ldi   temp,high(RAMEND);
    out   PH,temp                   ;
```

```
  ldi   emp,(1<<SRE)          ;Дозвіл роботи з зовн. пам'яттю
  out   CUCR,temp
  ldi   xh,high(righti)       ;Завантаження адреси молодшого
                              розряду в
                              ;регістр x
  ldi   l,low(righti)
  ldi   h,high(dac)           ;Завантаження адреси ЦАП у
                              регістр y
  ldi   l,low(dac)
  ldi   emp,(1<<TOIE0)        ;Дозвіл переривань від лічильника
                              TIMER0
  out   IMSK,temp
  ldi   temp,(1<<CS10)        ;Запуск лічильника TIMER0
  out   TCCR0,temp
  clr   dac_r
  sei                         ;Дозвіл переривань

                              ;Головна програма
start:
  st    y,dac_r               ;Виведення значення змінної «dac_r»
                              на ЦАП
  rjmp  start                 ;Нескінченний цикл

                              ;Переривання при переповненні
                              лічильника TIMER0
tim0_ovf:
  sbic  PINB,7                ;Якщо PINB7 == «0»,
  rjmp  m1
  st    x,dac_r               ;записати значення «dac_r» за
                              адресою, що в регістрі x,
  ldi   dac_r,255             ;очистити змінну «dac_r»
m1:
  inc   dac_r                 ;Інкремент змінної «dac_r»
reti
```

Додаток Л

Програма «Маніпулятор»

У центрі світлодіодної матриці засвічується крапка. За допомогою клавіш цифрової клавіатури «4», «6», «2» та «8» можна змінювати положення крапки відповідно вліво, вправо, вгору та вниз відповідно.

Лістинг програми «Маніпулятор» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

char *pA=(char *)0x8000;               //Адреса порту pF
char *pC=(char *)0x8002;               //Адреса порту pC

char *kbc0=(char *)0x9006;             //Адреса стовпця A0 клавіатури
char *kbc1=(char *)0x9005;             //Адреса стовпця A1 клавіатури
char *kbc2=(char *)0x9003;             //Адреса стовпця A2 клавіатури

void main()
{
  char key;                            //Стан стовпця клавіатури
  char str=4;                          //y-координата крапки
  char row=8;                          //x-координата крапки

  MCUCR = (1<<SRE);                    //Дозвіл роботи з зовн. пам'яттю

  while(1)                             //Нескінченний цикл
      {
          key=*kbc0 & 37               //Сканування стовпця A0
                                       клавіатури
          if(key==0b00000101)          //Якщо натиснуто клавішу «4»
          {
              if(str==16)              //і крапка не вийшла за межі
                                       матриці
                  str=1;
              else
                  str=str<<1;  //змістити її на одну позицію вліво
          }

          key=*kbc1 & 7;               //Сканування стовпця A1 клавіатури
      if(key==0b00000110)              //Якщо натиснуто клавішу «2»
          {
          if(row==64)                  //і крапка не вийшла за межі
                                       матриці –
```

```
                    row=1;
            else
                    row=row<<1;     //змістити її на одну позицію вгору
            }
        if(key==0b00000011)         //Якщо натиснуто клавішу «8»
            {
            if(row==1)              //і крапка не вийшла за межі
                                    матриці –
                    row=64;
            else
                    row=row>>1;     //змістити її на одну позицію вниз
            }

        key=*kbc2 & 7;              //Сканування стовпця А2 клавіатури
        if(key==0b00000101)         //Якщо натиснуто клавішу «6»
            {
            if(str==1)              //і крапка не вийшла за межі матриці
                    str=16;
            else
                    str=str>>1;     //змістити її на одну позицію вправо
            }

            *pA=str;                //Вивести крапку на матрицю
            *pC=~row;               // за отриманими координатами
            _delay_ms(1000);        //Затримка опитування клавіатури
        }
}
```

Лістинг програми «Маніпулятор» мовою Assembler

```
.include <m8515def.inc>             ;Файл з описом портів МК

.def temp =r16                      ;Робоча змінна
.def keyR =r20                      ;Стан стовпця клавіатури
.def rowR =r21                      ;у-координата крапки
.def strR =r22                      ;х-координата крапки

.equ strM =$8000                    ;Адреса порту pA
.equ rowM =$8002                    ;Адреса порту pC

.equ kbc0 =$9006                    ;Адреса 0-го стовпця клавіатури
.equ kbc1 =$9005                    ;Адреса 1-го стовпця клавіатури
.equ kbc2 =$9003                    ;Адреса 2-го стовпця клавіатури

                                    ;Ініціалізація
```

```
reset:
   ldi    temp,low(RAMEND)          ;Ініціалізація стека
   out    SPL,temp          ;
   ldi    temp,high(RAMEND);
   out    SPH,temp          ;

   ldi    temp,(1<<SRE)             ;Дозвіл роботи із зовн. пам'яттю
   out    MCUCR,temp

   ldi    rowR,8                    ;Початкові координати крапки
   ldi    strR,4            ;

   ldi    xh,high(kbc0)             ;Завантаження адреси стовпця A0
                                    ;клавіатури в регістр x
   ldi    xl,low(kbc0)

                                    ;Головна програма
start:
   ldi    yh,high(rowM)             ;Завантаження адреси порту pC в
                                   регістр y
   ldi    yl,low(rowM)
   ldi    zh,high(strM)             ;Завантаження адреси порту pA в
                                   регістр z
   ldi    zl,low(strM)

   mov    temp,rowR                 ;Вивести зображення крапки на
                                   матрицю:
   com    temp                      ;
   st     y,temp                    ;y-координата
   st     z,strR                    ;x-координата

   ldi    yh,high(kbc1)             ;Завантаження адреси стовпця A1
                                   клавіатури
                                    ;в регістр y
   ldi    yl,low(kbc1)
   ldi    zh,high(kbc2)             ;Завантаження адреси стовпця A2
                                   клавіатури
                                    ;в регістр z
   ldi    zl,low(kbc2)
                                    ;Сканування клавіатури
kbscan:

                                    ;Сканування 0-го стовпця
                                    клавіатури;
   ld     keyR,x                    ;Якщо натиснуто клавішу «4»
   andi   keyR,0b00000010
   sbrc   keyR,1                    ;і крапка не вийшла за межі
                                    матриці –
   rjmp   m1
```

```
  lsl   strR                    ;змістити її на одну позицію вліво
  sbrc  strR,5
  ldi   strR,1

                                ;Сканування 1-го стовпця
                                клавіатури
m1:
  ld    keyR,y                  ;Якщо натиснуто клавішу «2»
  andi  keyR,0b00000101
  sbrc  keyR,0
  brne  m2
  lsl   rowR                    ;і крапка не вийшла за межі
                                матриці –
  sbrc  rowR,7                  ;змістити її на одну позицію вгору
  ldi   rowR,1
m2:
  sbrc  keyR,2                  ;Якщо натиснуто клавішу «8»
  rjmp  m3
  sbrc  rowR,0                  ;і крапка не вийшла за межі
                                матриці –
  ldi   rowR,0b10000000
  lsr   rowR                    ;змістити її на одну позицію вниз

                                ;Сканування 2-го стовпця
                                клавіатури
m3:
  ld    keyR,z                  ;Якщо натиснуто клавішу «6»
  andi  keyR,0b00000010 ;       ;і крапка не вийшла за межі
                                матриці –
  sbrc  keyR,1
  rjmp  kbscan
  sbrc  strR,0
  ldi   strR,0b00100000
  lsr   strR                    ;змістити її на одну позицію вгору

delay:                          ;Затримка
  ser   r17
  ser   r18
  ldi   r19,8
m4:
  dec   r17
  brne  m4
  dec   r18
  brne  m4
  dec   r19
  brne  m4
  rjmp  start
```

Додаток М

Програма «Табло»

Програма дозволяє виводити на матрицю числа від 0 до 9, залежно від натиснутої клавіші цифрової клавіатури.

Лістинг програми «Табло» мовою С

```
#include <avr/io.h>                    //Файл з описом портів МК
#include <avr/interrupt.h>             //Бібліотека ф-цій, що
реалізовують переривання МК
#include <util/delay.h>                //Бібліотека ф-ції _delay_ms()

char *pA=(char *)0x8000;               //Адреса порту pA
char *pC=(char *)0x8002;               //Адреса порту pC

char *kbc0=(char *)0x9006;             //Адреса 0-го стовпця клавіатури
char *kbc1=(char *)0x9005;             //Адреса 1-го стовпця клавіатури
char *kbc2=(char *)0x9003;             //Адреса 2-го стовпця клавіатури

void main()
{
                                       //Таблиця кодів зображень
                                       світлодіодної матриці
  char digit[10][7]={
                    {14,17,17,17,17,17,14},     // 0
                    {4,4,12,4,4,4,14},          // 1
                    {14,17,1,2,4,8,31},         // 2
                    {14,17,1,6,1,17,14},        // 3
                    {17,17,17,31,1,1,1},        // 4
                    {31,16,30,1,1,17,14},       // 5
                    {14,17,16,30,17,17,14},     // 6
                    {31,1,1,2,4,8,16},          // 7
                    {14,17,17,14,17,17,14},     // 8
                    {14,17,17,15,1,17,14}};     // 9

  char key;                //Стан стовпця клавіатури
  char str=1;              //Допоміжна змінна
  char i;                  //Допоміжна змінна
  char dig=0;              //Зміщення адреси зображення в таблиці
                           кодів

  MCUCR = (1<<SRE);    //Дозвіл роботи з зовн. пам'яттю

  while(1)
  {
```

```
        key=*kbc0 & 7;          //Сканування стовпця A0 клавіатури
        switch (key)            //Визначити адресу коду зображення
                                відповідно до натиснутої клавіші
        {
                case 0b00000110: dig=1;break;
                case 0b00000101: dig=4;break;
                case 0b00000011: dig=7;
        }

  key=*kbc1 & 0b00001111;       //Сканування стовпця A1 клавіатури
        switch (key)
        {
                case 0b00001110: dig=2;break;
                case 0b00001101: dig=5;break;
                case 0b00001011: dig=8;break;
                case 0b00000111: dig=0;
        }

  key=*kbc2 & 0b00000111;       //Сканування стовпця A2 клавіатури
        switch (key)
        {
                case 0b00000110: dig=3;break;
                case 0b00000101: dig=6;break;
                case 0b00000011: dig=9;
        }

        for(i=0,str=64; i<7; i++)   //Вивести зображення на матрицю
        {
                *pA=digit[dig][i];
                *pC=~str;
                str=str>>1
                _delay_ms(15);
        }
  }
}
```

Лістинг програми «Табло» мовою Assembler

```
.include <m8515def.inc>         ;Файл з описом портів МК

.def temp =r16                  ;Робоча змінна
.def keyR =r20                  ;Стан стовпця клавіатури
.def str  =r21                  ;
.def dig  =r22                  ;Адреса зображення в таблиці кодів

.equ strM =$8000                ;Адреса порту pA
.equ rowM =$8002                ;Адреса порту pC

.equ kbc0 =$9006                ;Адреса стовпця A0 клавіатури
```

```
.equ kbc1 =$9005                ;Адреса стовпця A1 клавіатури
.equ kbc2 =$9003                ;Адреса стовпця A2 клавіатури

                                ;Ініціалізація
reset:

                                ;Таблиця кодів зображень світлодіодної
                                матриці
digit:
.db     14,17,17,17,17,17,14        ;0
.db     4,4,12,4,4,4,14             ;1
.db     14,17,1,2,4,8,31            ;2
.db     14,17,1,6,1,17,14           ;3
.db     17,17,17,31,1,1,1           ;4
.db     31,16,30,1,1,17,14          ;5
.db     14,17,16,30,17,17,14        ;6
.db     31,1,1,2,4,8,16             ;7
.db     14,17,17,14,17,17,14        ;8
.db     14,17,17,15,1,17,14         ;9

  ldi   temp,low(RAMEND)            ;Ініціалізація стека
  out   SPL,temp                    ;
  ldi   temp,high(RAMEND)           ;
  out   SPH,temp                    ;

  ldi   temp,(1<<SRE)               ;Дозвіл роботи з зовн. пам'яттю
  out   MCUCR,temp

  ldi   yh,high(rowM)               ;Завантаження адреси порту pC в
                                    регістр y
  ldi   yl,low(rowM)

  ldi   zh,high(digit*2)            ;Завантаження адреси таблиці
                                    кодів регістр zh

  clr   dig

                                ;Головна програма
start:
  ldi   xh,high(kbc0)               ;Сканування 0-го стовпця клавіатури
  ldi   xl,low(kbc0)

  ld    keyR,x
  andi  keyR,0b00000111

  sbrc  keyR,0                      ;Якщо натиснуто клавішу «1»
  rjmp  m1
        ldi     dig,8               ;Завантажити в «dig» зміщення
                                    до комірки з кодом
                                    ;зображення цифри «1»
```

```
m1:
  sbrc  keyR,1                ;Якщо натиснуто клавішу «4»
  rjmp  m2
  ldi   dig,32
m2:
  sbrc  keyR,2                ;Якщо натиснуто клавішу «7»
  rjmp  m3
  ldi   dig,56
m3:
  ldi   xl,low(kbc1)          ;Сканування 1-го стовпця клавіатури

  ld    keyR,x
  andi  keyR,0b00001111

  sbrc  keyR,0                ;Якщо натиснуто клавішу «2»
  rjmp  m4
  ldi   dig,16
m4:
  sbrc  keyR,1                ;Якщо натиснуто клавішу «5»
  rjmp  m5
  ldi   dig,40
m5:
  sbrc  keyR,2                ;Якщо натиснуто клавішу «8»
  rjmp  m6
  ldi   dig,64
m6:
  sbrc  keyR,3                ;Якщо натиснуто клавішу «0»
  rjmp  m7
  clr   dig
m7:
  ldi   xl,low(kbc2)          ;Сканування 2-го стовпця клавіатури

  ld    keyR,x
  andi  keyR,0b00000111

  sbrc  keyR,0                ;Якщо натиснуто клавішу «3»
  rjmp  m8
  ldi   dig,24
m8:
  sbrc  keyR,1                ;Якщо натиснуто клавішу «6»
  rjmp  m9
  ldi   dig,48
m9:
  sbrc  keyR,2                ;Якщо натиснуто клавішу «9»
```

```
   rjmp  m10
   ldi   dig,72
m10:
   ldi   xh,high(strM)          ;Завантажити адресу порту pA в
                                регістр x
   ldi   xl,low(strM)

   ldi     zl,low(digit*2)      ;Завантажити адресу таблиці
                                кодів в регістр
                                ;zl
   add   zl,dig                 ;Визначити адресу необхідної
                                комірки із
                                ;кодом

   ldi   str,64
m11:
   rcall delay

   lpm   temp,z+                ;Завантажити код зображення з
                                таблиці
   st    x,temp                 ;Вивести код зображення на
                                матрицю

   mov   temp,str               ;
   com   temp                   ;
   st    y,temp                 ;

   lsr   str                    ;
   brne  m11

   rjmp  start                  ;Нескінченний цикл

                                ;Підпрограма затримки
delay:
   ser   r17
   ldi   r18,20
m15:
   dec   r17
   brne  m15
   dec   r18
   brne  m15
ret
```

Додаток Н

Програма «Секундомір»

Програма дозволяє вимірювати проміжки часу з точністю до десятих секунди. Максимальне значення виміряного часу – 999 секунд. Значення часу відображається на статичному індикаторі. При натисканні кнопки SW16 секундомір запускається. Повторне натискання кнопки призупиняє/продовжує відлік часу. При натисканні кнопки SW15 значення секундоміра обнуляється і очікується повторний запуск секундоміра. Для відліку часу використовується переривання лічильника T/C1, яке відбувається з частотою 10 Гц. Сканування кнопок відбувається під час обробки переривання від лічильника T/C0.

Лістинг програми «Секундомір» мовою С

```
#include <avr/io.h>                  //Файл з описом портів МК
#include <avr/interrupt.h>           //Бібліотека функцій, що реалізовують
                                     переривання МК
#include <util/delay.h>              //Бібліотека ф-ції _delay_ms()

#define SW15 (PIND & 4) == 0  //«SW15» означає стан кнопки SW15
                              «Натиснута»
#define SW16 (PIND & 8) == 0  //«SW16» означає стан кнопки SW16
                              «Натиснута»

char *lefti=(char *)0xa000;          //Адреса старшого розряду стат.
                                     індикатора
char *righti=(char *)0xa001;         //Адреса молодшого розряду стат.
                                     індикатора
char *DC_REG=(char *)0xa004; //Регістр управління стат. індикатором

char timeR;                          //Лічильник секунд
char timeL;                          //Лічильник десятків секунд

                                   //Переривання при переповненні
                                   лічильника TIMER0
                                     //Сканування кнопок
SIGNAL(SIG_OVERFLOW0)
{
  if(SW16)                           //Якщо натиснуто кнопку SW16 –
                                     пауза/запуск секундоміра
  {
```

```
        while(SW16);                    //Захист від багаторазового
                                        спрацьовування
        TCCR1B = ~TCCR1B & 3;  //Стоп/запуск лічильника TIMER0
    }

    if(SW15)                            //Якщо натиснуто кнопку SW15,
                                        обнулити секундомір
    {
        TCCR1B = 0;                     //Зупинити лічильник TIMER1
        TCNT1 = 0xd2f0;
        *DC_REG = 0b01000011;  //Погасити старший розряд стат.
                                        індикатора
        timeR=0;                        //Обнулити значення змінних-
                                        лічильників
        timeL=0;                        //
    }
}

                                        //Переривання при переповненні
                                        лічильника TIMER1
                                        //Підрахунок часу
SIGNAL(SIG_OVERFLOW1)
{
    TCNT1 = 0xd2f0;
    timeR++;                            //Збільшити значення лічильника
                                        секунд
    if(timeR>99)                        //Якщо відбулось переповнення
                                        лічильника секунд,
    {
        timeR=0;                        //обнулити лічильник секунд,
        timeL++;                        //збільшити на 1 лічильник десятків
                                        секунд
        *DC_REG = 0b01000001;           //Новий формат відображення
        if(timeL>9)
            *DC_REG = 0b01000000;  //Новий формат відображення
            if(timeL>99)                //Якщо значення лічильника
                                        >99 – обнулити секундомір
        {
            timeL=0;
            *DC_REG = 0b01000011;  //Погасити старший розряд
                                        індикатора
        }
    }
}
```

```
                                    //Переривання при переповненні
                                  лічильника TIMER0
                                    //Сканування кнопок
SIGNAL(SIG_OVERFLOW0)
{
  if(SW16)                          //Якщо натиснуто кнопку SW16 –
                                  пауза/запуск секундоміра
  {
        while(SW16);                //Захист від багаторазового
                                  спрацьовування
        TCCR1B = ~TCCR1B & 3;       //Стоп/запуск лічильника
                                  TIMER0
  }

  if(SW15)                          //Якщо натиснуто кнопку SW15,
                                  обнулити секундомір
  {
        TCCR1B = 0;                 //Зупинити лічильник TIMER1
        TCNT1 = 0xd2f0;
        *DC_REG = 0b01000011;       //Погасити старший розряд стат.
                                  індикатора
        timeR=0;                    //Обнулити значення змінних-
                                  лічильників
        timeL=0;                    //
  }
}

void main()
{
  MCUCR = (1<<SRE);                 //Дозвіл роботи із зовн. пам'яттю
  TIMSK = (1<<TOIE1)|(1<<TOIE0); //Дозвіл переривань від
                                  лічильників
  TCCR0 = (1<<CS02)|(1<<CS00);      //Запуск лічильника TIMER0
  TCNT1 = 0xd300;

  sei();                            //Дозвіл переривань
  while(!SW16);                     //Запуск секундоміра натиском
                                  кнопки SW16

  *DC_REG = 0b01000011;             //Задіяний молодший розряд
                                  індикатора і сегмент крапки

  while(1)
  {
```

```
    *righti=timeR/10*16+timeR%10;
                                    //Формування та виведення
                                зображення на молодший розряд стат.
                                індикатора
    *lefti=timeL/10*16+timeL%10;
                                    //Формування та виведення
                                зображення на старший розряд стат.
                                індикатора
      }
    }
```

Лістинг програми «Секундомір» мовою Assembler

```
    .include <m8515def.inc>   ;Файл з описом портів МК

    .def cnt     =r0          ;Допоміжна змінна
    .def temp    =r16         ;Робоча змінна
    .def dataR   =r17         ;Змінні для збереження зображення стат.
індикатора
    .def dataL   =r18         ;
    .def timeR   =r19         ;Лічильник секунд
    .def timeL   =r20         ;Лічильник десятків секунд
    .def multip  =r21         ;Змінна, необхідна для обрахунків
    .def tim1h   =r22         ;Початкове значення регістра TCNT1H
    .def tim1l   =r23         ;Початкове значення регістра TCNT1L
    .def DCreg   =r24         ;Регістр керування розрядами стат.
                            індикатора
    .def tim1st  =r25         ;Змінна керування станом лічильника
                            TIMER1

    .equ lefti    =$a000      ;Адреса старшого розряду стат. індикатора
    .equ righti   =$a001      ;Адреса молодшого розряду стат. індикатора
    .equ DC_REG =$a004        ;Регістр керування розрядами стат.
                            індикатора

    .equ SW15  =2             ;Виразу «SW15» присвоєно значення 2
    .equ SW16  =3             ;Виразу «SW16» присвоєно значення 3

                            ;Адреси векторів переривань МК
    .org 0                    ;Переривання при скиданні МК
      rjmp  reset
    .org 6                    ;Переривання при переповненні лічильника
                            TIMER1
      rjmp  tim1_ovf
```

```
.org 7                          ;Переривання при переповненні
                                лічильника TIMER0
  rjmp  tim0_ovf

                                ;Ініціалізація
reset:
  ldi   temp,low(RAMEND)        ;Ініціалізація стека
  out   SPL,temp                ;
  ldi   temp,high(RAMEND);
  out   SPH,temp                ;

  ldi   temp,(1<<SRE)           ;Дозвіл роботи із зовн. пам'яттю
  out   MCUCR,temp

  cbi   DDRD,2                  ;Конфігурація порту D (для роботи із
                                кнопками)
  cbi   DDRD,3

  ldi   temp,(1<<TOIE1)|(1<<TOIE0)
  out   TIMSK,temp              ;Дозвіл переривань від лічильників
                                TIMER1, TIMER0

  ldi   temp,(1<<CS02)|(1<<CS00)
  out   TCCR0,temp              ;Запуск лічильника TIMER0
  ldi   xh,high(righti)         ;Завантаження адреси молодшого
                                розряду в регістр x
  ldi   xl,low(righti)

  ldi   yh,high(lefti)          ;Завантаження адреси старшого
                                розряду в регістр y
  ldi   yl,low(lefti)

  ldi   zh,high(DC_REG)         ;Завантаження адреси регістра
                                керування в регістр z
  ldi   zl,low(DC_REG)

  clr   timeL                   ;Ініціалізація змінних
  clr   timeR                   ;
  clr   dataR                   ;
  clr   dataL                   ;
  ldi   multip,16               ;
  ldi   tim1h,$d3               ;
  ldi   tim1l,$00               ;
  ldi   tim1st,3                ;
```

```
   ldi    DCreg,0b01000011      ;Задіяний молодший розряд індикатора
                                і сегмент крапки
   st     z,DCreg

   sei                          ;Дозвіл переривань

                                ;Головна програма
begin:
   sbic   PIND,SW16             ;Запуск таймера
   rjmp   begin

start:
   mov    temp,timeL            ;Кількість секунд
   rcall  conv                  ;Сформувати код зображення
   mov    dataL,temp            ;Завантажити код в регістр зображення
                                молодшого розряду
   mov    temp,timeR            ;Кількість десятків секунд
   rcall  conv                  ;Сформувати код зображення
   mov    dataR,temp            ;Завантажити код в регістр зображення
                                старшого розряду

   st     x,dataR               ;Вивести зображення на стат. індикатор
   st     y,dataL               ;

   clt                          ;Очистити біт «t» регістра SREG

wait:                           ;Очікування переривання від
                                лічильника TIMER1
   in     temp,SREG
   sbrs   temp,6
   rjmp   wait

   rjmp   start


                                ;Підпрограма для формування
                                зображення стат. індикатора
                                ;Перетворити 16-ве число в 10-ве для
                                відображення на стат. індикаторі
conv:
   clr    cnt
m1:
   cpi    temp,10
   brlo   m2
```

```
  subi  temp,10
  inc   cnt
  rjmp  m1
m2:
  mul   cnt,multip
  add   temp,cnt

  ret

                              ;Переривання при переповненні
                              лічильника TIMER1
tim1_ovf:
  out     TCNT1H,tim1h        ;Записати значення лічильного
                              регістра лічильника
  out     TCNT1L,tim1l

  inc     timeR               ;Збільшити значення лічильника
                              секунд
m3:
  cpi     timeR,100           ;Якщо значення лічильника секунд
                              >100,
  brlo    m6
  clr     timeR               ;обнулити лічильник секунд,
  inc     timeL               ;збільшити на 1 лічильник десятків
                              секунд

  ldi     DCreg,0b01000001    ;Новий формат відображення
  st      z,DCreg
m4:
  cpi     timeL,10
  brlo    m5

  ldi     DCreg,0b01000000    ;Новий формат відображення
  st      z,DCreg
m5:
  cpi     timeL,100           ;Якщо значення лічильника >100,
  brlo  m6
  clr     timeL               ;Обнулити лічильник десятків
                              секунд

  ldi     DCreg,0b01000111    ;Погасити старший розряд стат.
                              індикатора
  st      z,DCreg
```

```
m6:
  set                          ;Відбулось переривання

  reti


                               ;Переривання при переповненні
                               лічильника TIMER0
tim0_ovf:
  sbic  PIND,SW16              ;Якщо натиснуто кнопку SW16 –
                               пауза/запуск таймера
  rjmp  m8
m7:                            ;Захист від багаторазового
                               спрацьовування
  sbis  PIND,SW16
  rjmp  m7

  in    tim1st,TCCR1B
  com   tim1st
  andi  tim1st,0b00000011
  out   TCCR1B,tim1st          ;Стоп/запуск лічильника TIMER0

  out   TCNT1H,tim1h
  out   TCNT1L,tim1l
m8:
  sbic  PIND,SW15              ;Якщо натиснуто кнопку SW15,
                               обнулити таймер
  rjmp  m9
  clr   timeR                  ;Обнулити змінні
  clr   timeL                  ;
  clr   dataR                  ;
  clr   dataL                  ;
  out   TCCR1B,timeL           ;Зупинити лічильник TIMER1

  ldi   DCreg,0b01000011       ;Погасити старший розряд стат.
                               індикатора
  st    z,DCreg

  set                          ;Відбулось переривання від
                               лічильника TIMER1
m9:
  reti
```

Додаток П

Програма «Годинник»

Програма демонструє реалізацію цифрового годинника, який може вимірювати час до 100 хвилин. Після сотої хвилини відбувається скидання годинника. Значення часу у форматі «хвилини:секунди» виводиться на динамічний індикатор. Також щосекунди відбувається засвічування сегмента «:» індикатора. Запуск годинника відбувається натисканням кнопки SW16. Для відліку часу використовується переривання лічильника T/C1 з частотою 2 Гц, для динамічної індикації – переривання лічильника T/C0. Також програмно реалізовано дешифратор, за допомогою якого і створюється зображення динамічного індикатора. Таблиця з кодами дешифратора зберігається у пам'яті програм мікроконтролера.

Лістинг програми «Годинник» мовою С

```
#include <avr/io.h>                  //Файл з описом портів МК
#include <avr/interrupt.h>           //Бібліотека функцій, що реалізовують
                                     переривання МК

#define SW16 (PIND & 8) == 0  //Вираз «SW16» означає стан кнопки
                               SW16 «Натиснута»

char *pB=(char *)0x8001;             //Адреса порту pB
char *pC=(char *)0x8002;             //Адреса порту pC

                                      //Таблиця кодів зображень динамічного
                                     індикатора:
                                      //«0», «1», «2», «3», «4», «5», «6», «7», «8»,
                                      «9», «-»
char digit[]={63,6,91,79,102,109,125,7,127,111,64};

                                      //Лічильники часу/значення розрядів
                                     дин. індикатора:
                                      //»row[0]» – десятки хвилин, «row[1]» –
                                     хвилини, «row[2]» – десятки секунд,
                                     «row[3]» – секунди
char row[4]={10,10,10,10};     //Початкове значення розрядів (знак «-»
                               в таблиці кодів)

char rowP;                           //Номер розряду дин. індикатора
char dot=0;                          //Стан сегмента «:» на дин. індикаторі
                                    (поч. значення – незадіяний)
```

```
                                //Переривання при переповненні
                              лічильника TIMER0
                                //Виведення зображення на дин.
                              індикатор
SIGNAL(SIG_OVERFLOW0)
{
  if(rowP==1 && dot)          //Вивести число з елемента таблиці
                              зображень з індексом «row[rowP]»
        *pB=digit[row[rowP]] | 0b10000000;   //на порт «pB», якщо
                                             необхідно, задіяти сегмент
                                             «:» дин. індикатора
  else
        *pB=digit[row[rowP]];

  *pC=rowP;                   //Задіяти розряд дин. індикатора, номер
                              якого в «rowP»
  rowP++;                     //Номер наступного розряду

  if(rowP>3)
        rowP=0;
}

                                //Переривання при переповненні
                              лічильника TIMER1
                                //Зміна стану сегмента «:» дин.
                              індикатора
                                //Лічба часу (відбувається при кожному
                              2-му перериванні таймера: F=1Hz)
SIGNAL(SIG_OVERFLOW1)
{
  TCNT1 = 0xf1a8;
  dot=~dot;                   //Змінити стан сегмента «:» дин.
                              індикатора
  if(dot)                     //Виконати при кожному 2-му
                              перериванні
  {
        row[3]++;             //Збільшити значення лічильника
                              «row[3]» на 1
        if(row[3]>9)          //Якщо кількість секунд >9,
        {
              row[3]=0;       //обнулити лічильник «row[3]»,
              row[2]++;       //збільшити лічильник «row[2]» на 1

              if(row[2]>5)    //Якщо кількість секунд >59,
              {
```

```
                    row[2]=0;     //обнулити лічильник «row[2]»,
                    row[1]++;     //збільшити лічильник «row[1]» на 1
                    if(row[1]>9)  //Якщо кількість хвилин >9,
                    {
                        row[1]=0;     //обнулити лічильник «row[1]»,
                        row[0]++;     //збільшити лічильник «row[0]» на
                            if(row[0]>9)  //Якщо кількість десятків
                                          хвилин >9,
                        row[0]=0;     //обнулити лічильник «row[0]»
                    }
                }
            }
        }
    }

void main()
{
    MCUCR = (1<<SRE);                   //Дозвіл роботи із зовн. пам'яттю
    TIMSK = (1<<TOIE1)|(1<<TOIE0);      //Дозвіл переривань від
                                        лічильників TIMER1, TIMER0
    TCCR0 = (1<<CS01)|(1<<CS00);        //Запуск лічильника TIMER0
    TCNT1 = 0xf1a8;

    sei();                              //Дозвіл переривань
    while(!SW16);                       //Очікування натискання кнопки
                                        SW16
    TCCR1B = (1<<CS10)|(1<<CS12);       //Запуск лічильника TIMER1
    while(1);                           //Нескінченний цикл
}
```

Лістинг програми «Годинник» мовою Assembler

```
.include <m8515def.inc>   ;Файл з описом портів МК

.def temp  =r16           ;Робоча змінна
.def rowP  =r17           ;Номер розряду дин. індикатора
.def tim1h =r18           ;Початкове значення регістра TCNT1H
.def tim1l =r19           ;Початкове значення регістра TCNT1L
.def data  =r20           ;Значення розряду дин. індикатора
.def row0  =r21           ;Лічильник десятків хвилин
.def row1  =r22           ;Лічильник хвилин
.def row2  =r23           ;Лічильник десятків секунд
.def row3  =r24           ;Лічильник секунд
```

```
.def dot  =r25              ;Регістр, для зміни стану сегмента «:» дин.
                            індикатора.
Стан сегменту «:» визначається станом біта «t» регістра SREG

.equ HL1d  =$8001           ;Адреса порту pB
.equ HL1r  =$8002           ;Адреса порту pC

.equ SW16  =3               ;Виразу «SW16» присвоєно значення 3

                            ;Адреси векторів переривань МК
.org 0                      ;Переривання при скиданні МК
  rjmp  reset
.org 6                      ;Переривання при переповненні лічильника
                            TIMER1
  rjmp  tim1_ovf
.org 7                      ;Переривання при переповненні лічильника
                            TIMER0
  rjmp  tim0_ovf

                            ;Таблиця кодів зображень динамічного
                            індикатора, збережена у FLASH-пам'яті МК
digit:                      ;«0», «1», «2», «3», «4», «5», «6», «7», «8», «9»
  .db     63,6,91,79,102,109,125,7,127,111

                                  ;Ініціалізація
reset:
  ldi     temp,low(RAMEND)    ;Ініціалізація стека
  out     SPL,temp            ;
  ldi     temp,high(RAMEND)   ;
  out     SPH,temp            ;

  ldi     temp,(1<<SRE)       ;Дозвіл роботи із зовн. пам'яттю
  out     MCUCR,temp

  cbi     DDRD,3              ;Налаштування портів

  ldi     temp,(1<<TOIE1)|(1<<TOIE0)
  out     TIMSK,temp          ;Дозвіл переривань від лічильників
                              TIMER1, TIMER0
  ldi     temp,(1<<CS01)|(1<<CS00)
  out     TCCR0,temp          ;Запуск лічильника TIMER0
  ldi     xh,high(HL1d)       ;Завантаження адреси порту «pB» в
                              регістр x
  ldi     xl,low(HL1d)
```

```
   ldi    yh,high(HL1r)          ;Завантаження адреси порту «pC» в
                                 регістр z
   ldi    yl,low(HL1r)

   ldi    tim1h,$f1              ;Завантаження значення в регістри
   ldi    tim1l,$a8

   out    TCNT1H,tim1h           ;Завантаження значення в регістр
                                 TCNT1
   out    TCNT1L,tim1l           ;(корекція частоти переривань)

   clr    row0                   ;Ініціалізація змінних
   clr    row1                   ;
   clr    row2                   ;
   clr    row3                   ;
   clr    rowP                   ;

   set                           ;Сегмент «:» задіяний
   sei                           ;Дозвіл переривань

                                 ;Головна програма
start:
   sbic   PIND,SW16              ;Очікування натискання кнопки SW16
   rjmp   start

   ldi    temp,(1<<CS10)|(1<<CS12)
   out    TCCR1B,temp            ;Запуск лічильника TIMER1
wait:
   rjmp   wait                   ;Нескінченний цикл

                                 ;Переривання при переповненні
                                 лічильника TIMER1
                                 ;Зміна стану сегмента «:» дин.
                                 індикатора
                                 ;Лічба часу (відбувається при кожному
                                 2-му перериванні таймера)
tim1_ovf:
   out    TCNT1H,tim1h           ;Запис значення в лічильний регістр
                                 TIMER1
   out    TCNT1L,tim1l
   com    dot                    ;Змінити стан сегмента «:» дин.
                                 індикатора
   andi   dot,0b01000000
   out    SREG,dot
```

```
 sbrs   dot,6                ;Виконати при кожному 2-му
                            перериванні
 rjmp   m1
 inc    row3                 ;Збільшити значення лічильника
                            «row3» на 1
 cpi    row3,10              ;Якщо кількість секунд >9,
 brlo   m1
 clr    row3                 ;обнулити лічильник row3,
 inc    row2                 ;збільшити лічильник row2 на 1
 cpi    row2,6               ;Якщо кількість секунд >59,
 brlo   m1
 clr    row2                 ;обнулити лічильник row2,
 inc    row1                 ;збільшити лічильник row1 на 1
 cpi    row1,10              ;Якщо кількість хвилин >9,
 brlo   m1
 clr    row1                 ;обнулити лічильник row1,
 inc    row0                 ;збільшити лічильник row0 на 1
 cpi    row0,10              ;Якщо кількість десятків хвилин >99,
 brlo   m1
 clr    row0                 ;обнулити лічильник row0
m1:
  reti


                            ;Переривання при переповненні
                            лічильника TIMER0
                            ;Виведення зображення на дин.
                            індикатор
                            ;Завантаження коду зображення з
                            таблиці кодів зображень за адресою
                            zl+row[i],
                            ;де «i» – лічильник відповідного
                            розряду в регістр «data»
tim0_ovf:
   ldi    zh,high(digit*2)  ;Завантаження адреси таблиці кодів
                            зображення
   ldi    zl,low(digit*2)   ;в регістр z
   cpi    rowP,0            ;Номер розряду
   brne   m2
   add    zl,row1           ;Формування адреси із зображенням із
                            таблиці кодів
   lpm    data,z            ;Завантаження коду зображення в
                            «data»
   in     temp,SREG         ;Якщо необхідно, задіяти сегмент «:»
                            дин. індикатора
```

```
   sbrc   temp,6                ;
   ori    data,0b10000000       ;
   rjmp   m5
m2:
   cpi    rowP,1
   brne   m3
   add    zl,row2
   lpm    data,z
   rjmp   m5
m3:
   cpi    rowP,2
   brne   m4
   add    zl,row3
   lpm    data,z
   rjmp   m5
m4:
   cpi    rowP,3
   brne   m5
   add    zl,row0
   lpm    data,z
m5:
   inc    rowP
   cpi    rowP,4
   brlo   m6
   clr    rowP
m6:
   st     x,data          ;Запис коду зображення з «data» за
                          адресою «pB»
   st     y,rowP          ;Запис номера розряду з «rowP» за
                          адресою «pC»
   reti
```

Навчальне видання

Цирульник Сергій Михайлович
Азаров Олексій Дмитрович
Крупельницький Леонід Віталійович
Трояновська Тетяна Іванівна

МІКРОПРОЦЕСОРНА ТЕХНІКА

Навчальний посібник

Редактор В. Дружиніна

Оригінал-макет підготовлено С. Цирульником

Підписано до друку 24.01.2017 р.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 8,6.
Наклад 50 пр. Зам. № 2017-018.

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-87-38.
publish.vntu.edu.ua; email: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.