МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

Д. О. ЖУК, М. В. ДЖАНГИРОВ, І. Ю. ЖУК

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
"Основи проектування спеціалізованих мікроконтролерних та вбудованих комп'ютерних систем для комплексів суднової і промислової автоматизації"

Частина 1

Рекомендовано Методичною радою НУК

Електронне видання комбінованого
використання на DVD-ROM

МИКОЛАЇВ • НУК • 2011

УДК 004.031.6:681.5
ББК 32.965
О-75

Укладачі:
Д. О. Жук, канд. техн. наук, доцент, доцент кафедри ЕОС та ІБ, заступник директора ІАЕ;
М. В. Джангиров, викладач кафедри морського приладобудування ІАЕ;
І. Ю. Жук, старший викладач кафедри медичних приладів і систем ЧДУ ім. П. Могили

Рецензенти:
Д. В. Костенко, канд. техн. наук, доцент;
Г. В. Павлов, д-р техн. наук, професор

Д. О. Жук
О-75 Методичні вказівки до виконання лабораторних робіт "Основи проектування спеціалізованих мікроконтролерних та вбудованих комп'ютерних систем для комплексів суднової і промислової автоматизації" : Частина 1 / Д. О. Жук, М. В. Джангиров, І. Ю. Жук. – Миколаїв : Видавництво НУК, 2011. – 70 с.

Методичні вказівки призначені для студентів спеціальностей 7.05070202 "Електричні системи і комплекси транспортних засобів", 7.05100302 "Прилади точної механіки", 7.05010203 "Спеціалізовані комп'ютерні системи", 7.05100307 "Медичні прилади і системи".

УДК 004.031.6:681.5
ББК 32.965

Навчальне видання

ЖУК Дмитро Олександрович
ДЖАНГИРОВ Максим Володимирович
ЖУК Ірина Юріївна

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
"Основи проектування спеціалізованих мікроконтролерних та вбудованих комп'ютерних систем для комплексів суднової і промислової автоматизації"

Частина 1

Комп'ютерне верстання *В.Г. Мазанко*
Коректор *М.О. Паненко*

Формат 60×84/16. Ум. друк. арк. 4,1. Обсяг даних 2797 кб.
Тираж 16. Вид. № 28. Зам. № 509.

Видавець і виготівник Національний університет кораблебудування,
54025, м. Миколаїв, просп. Героїв Сталінграда, 9
E-mail : publishing@nuos.edu.ua

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції
ДК № 2506 від 25.05.2006 р.

ВСТУП

Однокристальні мікроконтролери (ОМК) знаходять широке застосування в самих різних сферах: від обчислювальних пристроїв, фото- та відеотехніки, медичного обладнання, принтерів, сканерів до автомобілебудування, мобільного зв'язку, побутової техніки. Перевага мікропроцесорних систем – їх гнучкість: систему, розроблену для виконання конкретного завдання, легко пристосувати для вирішення інших завдань зміною програмного забезпечення. Сучасні мікропроцесорні великі інтегральні схеми (ВІС), наприклад однокристальні мікроконтролери, містять усі складові ЕОМ – МП, пам'ять даних, пам'ять програм, інтерфейсні схеми – та ефективно використовуються в системах керування різноманітним обладнанням (промисловим, медичним, побутовим).

У вказівках наведені основні аспекти проектування систем на восьмирозрядних мікроконтролерах AVR. Розглянута архітектура мікроконтролера, система команд, робота з периферійними пристроями та галузі його використання.

Призначаються для студентів при вивченні профільних дисциплін студентами спеціальностей 7.090901 "Прилади точної механіки", 7.05010203 "Спеціалізовані комп'ютерні системи", "Медичні прилади і системи", 7.092201 "Електричні системи і комплекси транспортних засобів" і є практичним посібником з розробки цифрових систем на мікроконтролерах AVR. Можуть бути корисними фахівцям і викладачам у галузі проектування приладів та спеціалізованих вбудованих комп'ютерних систем.

1. ОСНОВНІ ВІДОМОСТІ ПРО МК AVR

Мікроконтролер сімейства AVR фірми Atmel представляє собою восьмирозрядну однокристальну мікро-ЕОМ зі спрощеною (скороченою) системою команд – RISC (Restricted (Reduced) Instruction Set Computer).

Ці мікроконтролери дозволяють вирішувати велику кількість задач вбудованих систем. Вони відрізняються від інших поширених мікроконтролерів великою швидкістю роботи та своєю універсальністю. Швидкодія даних мікроконтролерів дозволяє в ряді випадків застосовувати їх в пристроях, для реалізації яких раніше можна було застосовувати лише 16-розрядні мікроконтролери, що дозволяє значно здешевити готову систему. Крім того, мікроконтролери AVR дуже легко програмуються.

До складу сімейства AVR входять восьмирозрядні мікроконтролери двох серій – ATtiny та ATmega. Мікроконтролери семества Tiny мають невеликі обсяги пам'яті програм (1...2 кбайт) і обмежену переферію. Практично всі вони випускаються в 8-вивідних корпусах і призначені для "бюджетних" рішень, що приймаються в умовах жорстких фінансових обмежень. Галузь застосування цих мікроконтролерів – інтелектуальні датчики різного призначення (контрольні, пожежні, охоронні), іграшки, зарядні пристрої, різна побутова техніка та інші подібні пристрої. Мікроконтролери Mega навпаки мають найбільш розвинену периферію, найбільші серед усіх мікроконтролерів AVR обсяги пам'яті програм і даних. Вони призначені для використання в мобільних телефонах, контролерах різного переферійного обладнання (принтери, сканери, сучасні дискові накопичувачі, приводи CD-ROM/DVD-ROM), складної офісної техніки.

Мікроконтролери обох сімейств підтримують кілька режимів зниженого енергоспоживання, мають блок переривань, сторожовий таймер і допускають програмування безпосередньо в готовому пристрої.

Більшість команд, які входять в систему команд, обираються з пам'яті за один такт і виконуються за один такт роботи мікроконтролеру. При цьому, відсутнє внутрішнє ділення частоти, як, наприклад, в мікроконтролерах PIC. Тому, кількість команд, що виконуються за 1 с, співпадає з тактовою частотою роботи мікроконтролеру.

Мікроконтролери виготовляються по високоякісній КМОН (CMOS) технології, містять енергонезалежні запам'ятовуючі пристрої для зберігання програми та даних, виконані по Flash та EEPROM технологіям, і відрізняються низьким енергоспоживанням при високій тактовій частоті.

Запис програми і вихідних даних в пам'ять може виконуватись після встановлення мікроконтрлеру в апаратуру, де йому належить працювати.

Мікроконтролери сімейства AVR мають єдину базову структуру. Узагальнена структурна схема мікроконтролера зображена на рис.1.

До складу мікроконтролеру входять:

– генератор тактового сигналу (GCK);

– процесор (CPU);

– постійний запам'ятовуючий пристрій для зберігання програми, виконаний по технології Flash, (FlashROM);

– оперативний запамятовуючий пристрій статичного типу для зберігання даних (SRAM);

– постійний запам'ятовуючий пристрій для зберігання даних, виконаний по технології EEPROM (EEPROM);

– набір периферійних пристроїв для вводу та виводу даних, керуючих сигналів та виконання інших функцій.

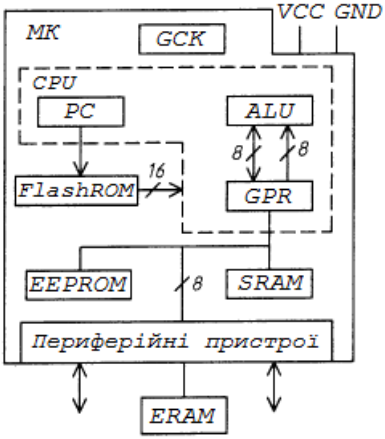


Рис. 1. Узагальнена структурна схема МК

Виводи VCC та GND призначені для підключення джерела напруги живлення МК. Рівень напруги всіх сигналів в МК відраховується відносно рівня на шині GND, який приймається за 0 В.

До складу процесора (CPU) входять:

– лічильник команд (PC);

– арифметико-логічний пристрій (ALU);

– блок регістрів загального призначення (GPR).

Окрім регістрів загального призначення в мікроконтролері існують регістри спеціальних функцій, які в сімействі AVR називаються регістрами вводу/виводу (I/O Registers, IOR). За участю цих регістрів здійснюється:

– керування роботою мікроконтролера та окремих його пристроїв;

– визначення стану мікроконтролера та окремих його пристроїв;

– ввод даних в мікроконтролер та окремі його пристрої і вивід даних і т. і.

Широка номенклатура AVR-МК дає користувачеві можливість оптимізувати співвідношення продуктивність – енергоспоживання – ціна. Високу продуктивність забезпечують:

– виконанням команд за один тактовий цикл;

– конвеєром команд, що забезпечують одночасно з виконанням поточної команди вибірку наступної;

– потужною системою команд єдиного 16-розрядного формату;

– вбудованими апаратними пристроями.

Мале енергоспоживання забезпечують:

– CMOS-технологією;

– цілком статичною роботою – від покрокового режиму до максимальної тактової частоти.

Малу вартість як на рівні вартості апаратного обладнання, так і на рівні вартості розробки і налагодження прикладних програм, забезпечують:

– Flash-пам'яттю програм, яку програмують на цільовій платі;

– можливістю вибору мікроконтролеру з достатньою і відповідною кількістю функцій і вбудованої периферії.

Нині співвідношення продуктивність – енергоспоживання – ціна для AVR – одне з найкращих на світовому ринку 8-розрядних мікроконтролерів.

2. ПРОГРАМНЕ ТА АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

AVR Studio – це інтегроване середовище розробки (IDE, Integrated Development Environment), яке дуже зручне для відлагодження AVR-додатків в операційних системах Windows 9x/Me/NT/2000/XP. Це середовище пропонує інтерфейс програмного імітатору та внутрішньо-схемного емулятору для восьмирозрядних мікроконтролерів AVR RISC. Окрім того, AVR Studio підтримує набір розробника STK500, який дозволяє програмувати AVR-пристрої, а також новий вбудований емулятор JTAG. Пакет AVRStudio безкоштовно розповсюджується фірмою Atmel.

Програма AVRStudio дозволяє писати програми на мові асемблер, під'єднувати зовнішній компілятор для мови С, відлагоджувати текст написаної програми, використовуючи вбудований програмний симулятор або під'єднувати внутрішньо-схемний емулятор.

В середовищі AVRStudio користувач має можливість контролювати стан кожного елементу за допомогою наступних вікон візуалізації (рис. 2):

– **Watch window** – відображає значення та адреси визначених змінних;

– **Register Window** – відображає стан регістрів загального призначення;

– **Memory window** – відображає зміст пам'яті програм, пам'яті даних, регістрів вводу/виводу та зміст енергонезалежної пам'яті EEPROM.

– **I/O window** – відображає зміст регістру стану, таймерів, EEPROM регістрів, портів вводу/виводу і т.і.

– **Processor window** – відображає важливу інформацію про виконання програми, в тому числі лічильник команд (Program Counter), покажчик стеку (Stack Pointer), прапори регістру стану (Flags), лічильник циклів (Cycle Counter) і т.і.

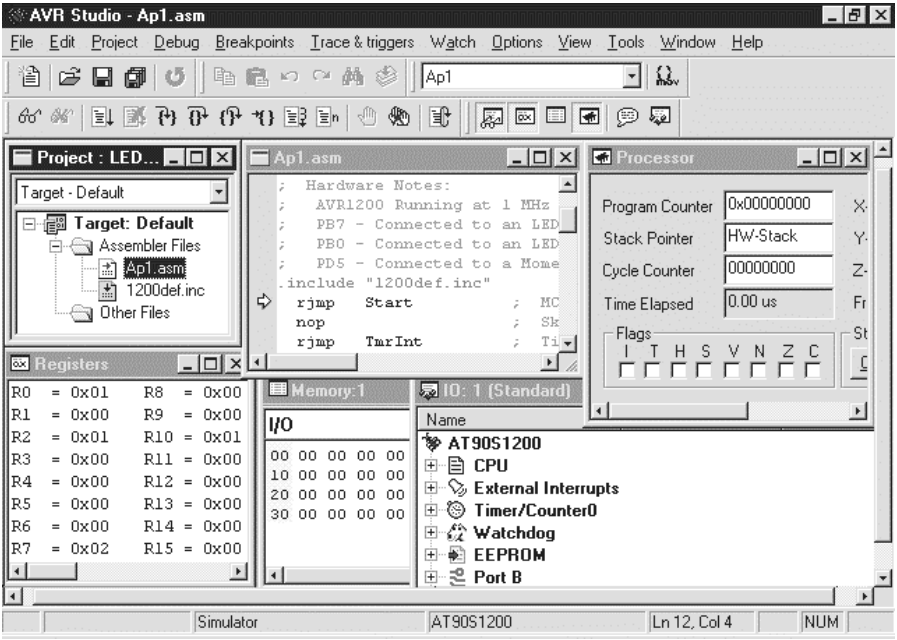


Рис. 2. AVR Studio в процесі відладки
вихідного коду програми

Під відладкою (відлагодженням) розуміють покрокове виконання з контролем вмісту регістрів мікроконтролеру (перевірка на низькому рівні) та змінних (перевірка на програмному рівні). Для переходу в режим відладки використовують наступні команди меню **Debug**:

– **Run, Auto Step** – перехід в режим відладки виконується, якщо зустрічається точка переривання.

– **Step Into** (клавіша <F11>) – виконує поточну команду з заходженням в підпрограми (всі вікна відновлюються). 

– **Step Over** (клавіша <F10>) – виконує поточну команду з виходом з підпрограми (всі вікна відновлюються). 

– **Step Out** (комбінація клавіш <Shift+F11>) – запускає програму ті виконує її до тих пір, поки не зустрінеться закінчення поточної підпрограми; якщо хід виконання знаходиться в області основної програми, то програма буде виконуватись до тих пір, поки не буде зупинена користувачем за допомогою команди Break або зустріне точку переривання. 

– **Run To Cursor** (комбінація клавіш <Ctrl+F10>) – запускає програму, яка виконується до тих пір, поки не зустрінеться курсор у вікні вихідного коду; якщо зустрічається точка зупинки, то виконання програми не зупиняється; якщо позиція курсору не досягається ніколи, то програма виконується до тих пір, поки не буде зупинена командою Break. 

Для того, щоб виконати програму не в режимі емуляції, потрібно дану програму записати в пам'ять цільового пристрою. Який би для цього програмний засіб не був би обраний, процес запису завжди проходить однаково:

1. Під'єднати мікроконтролер до програматору, який в свою чергу підключений до комп'ютеру.

2. Перевести мікроконтролер в режим програмування.

3. Записати дані в пам'ять (Flash або EEPROM) за допомогою спеціальних програмних засобів. Для виконання наведених лабораторних робіт рекомендується використовувати програму PonyProg2000.

Для виконання лабораторних робіт рекомендується використовувати відладочний модуль ATmega16 Evaluation Kit 2006 (рис. 3), побудований на базі мікропроцесору ATmega16 компанії Atmel. Це 8-розрядний однокристальний мікропроцесор на базі архітектури AVR-RISC Atmel, яка забезпечує достатньо високу швидкодію. Система команд містить 131 команду (повний список яких наведений в додатку 1), більшість з яких виконується за один такт. Схема має 32 лінії вводу/виводу та містить програмований послідовний універсальний асинхронний прийомопередавач, 10-розрядний 8-канальний АЦП, інтерфейс SPI, контрольний таймер та компаратор. Окрім того, ATmega16 містить три таймера/лічильника, один з яких 8-розрядний та 2 – 16-розрядні. Всі таймери

мають канали ШІМ, годинники реального часу та внутрішній осцилятор. Схема має функцію скидання при ввімкненні живлення та може працювати в двох режимах зниженого споживання потужності. Основні технічні характеристики ATmega16 наведені в табл.1.

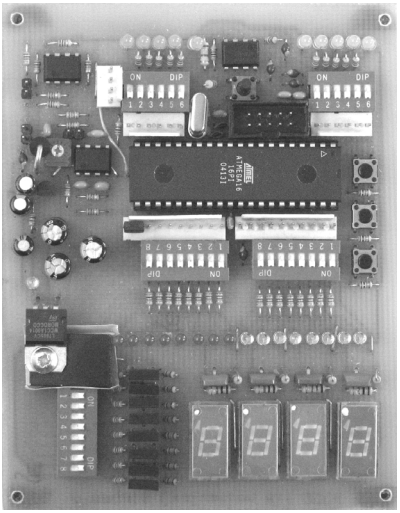
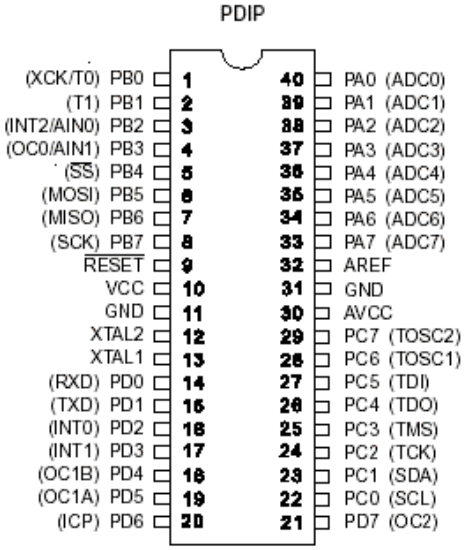


Рис. 3. Зовнішній вигляд модулю ATmega16 Evaluation Kit 2006

Таблиця 1. **Основні технічні характеристики**

Flash пам'ять	16 Кб
Напруга живлення	4,5 – 5,5 В
EEPROM пам'ять	0,5 Кб
SRAM пам'ять	1024 б
Максимальна частота	16 МГц
Максимальна кількість входів/виходів	32
Максимальна кількість переривань	20
Двопровідний інтерфейс TWI	+
Таймер 16 біт	1
Синхронний послідовний адаптер SPI	1
Кількість каналів АЦП (10 біт)	8
Внутрішньо-схемний програматор ISP	+
Універсальний асинхронний адаптер UART	1
Таймер 8 біт	2
Сторожовий таймер	+
Годинник реального часу RTS	+
Аналоговий компаратор	+
Схема скидання при зниженні живлення	+
Апаратне множення	+
Вбудований генератор	+
Режим самопрограмування	+

Розміщення виводів ATmega 16:



3. ЗАВДАННЯ ДО ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота №1

Тема: Ознайомлення із інтегрованим середовищем програмування та відладки МК AVR Studio та програмою-симулятором Proteus VSM.

Мета роботи: здійснити інсталяцію та ознайомитись з графічним інтерфейсом AVR Studio та Proteus VSM.

Порядок виконання роботи:

1. Здійснити установку AVR Studio 4.*;
2. Вивчити і пояснити основні пункти підменю AVR Studio;
3. Створити проект AVR Studio.

Рекомендації до процесу інсталяції AVR Studio:

AVR Studio – це інтегроване відладочне середовище (IDE) для мікро контролерів сімейства AVR фірми Atmel.

IDE AVR Studio містить:

– транслятор мови асемблер (Atmel AVR macroassembler);

– відгадчик (Debugger);

– програмне забезпечення верхнього рівня для підтримки внутрішньо-схемного програмування (In-system Programming, ISP).

Для інсталяції AVR Studio необхідно:

1. Запустити інсталяційний файл AVR Studio.
2. У відкритому вікні вибрати компоненти, які бажано інсталювати.
3. Виконати настройки, необхідні для інсталяції.

Основні пункти меню AVR Studio:

1. Меню File.
2. Меню Edit.
3. Меню Project.
4. Меню Debug.
5. Меню Breakpoint.
6. Меню Trace & triggers.
7. Меню Watch.
8. Меню Options.
9. Меню View.
10. Меню Tools.
11. Меню Window.
12. Меню Help.

Створення проекту AVR Studio:

1. Запустити AVR Studio.
2. В меню **Project** обрати команду **New**.
3. У вікні створення нового проекту необхідно вказати назву проекту (**Project name**) і його розміщення (**Location**). Новий проект зручно створювати в новій папці. Зверніть увагу, що назва проекту та його розміщення не повинні мати кириличні символи.
4. Далі обирається тип проекту:

– **AVR Assembler.** Використовує вбудований макроасемблер AVR Studio.

– **AVR GCC**. Використовує зовнішній компілятор С, що має інтерфейс командної строки.

Обрати тип **AVR Assembler**. Після цього у меню Debug Platform вибрати AVR Simulator, у меню Device – ATmega16. Натистути кнопку Finish.

5. У вікні, що відкрилось, набрати фрагмент програми, наведений нижче.

6. Далі необхідно відтранслювати програму, тобто перевірити її на правильне написання. Для цього потрібно вибрати пункт Build в меню Build. У нижньому вікні, з'явиться інформація про кількість слів коду та даних, про наявність помилок і т. і.

```
.include "m16def.inc"              ; Підключити файл опису імен регістрів
; вводу/виводу
; Ініціалізація
        ldi  r31, LOW(RAMEND); Ініціалізація покажчику стека, яка
        out  spl, r31                ; потрібна для використання підпрограм
        ldi  r31, HIGH(RAMEND)
        out  sph, r31
        ldi  r31, 0xff               ; Програмування порту B на вивід
        out ddrb, r31
; Безкінечний цикл
Loop:
   ldi  r16,0xff                     ; записати у регістр r16 всі одиниці
                                     ; тобто запалити всі світлодіоди
        out  portb, r16              ; вивести зміст r16 в порт B
        rcall  wait                  ; виклик підпрограми затримки
        ldi r16,0x00                 ; записати у регістр r16 всі нулі
                                     ; тобто загасити всі світлодіоди
        out  portb, r16              ; вивести зміст r16 в порт B
        rcall  wait                  ; виклик підпрограми затримки
        rjmp  Loop                   ; повернення на мітку Loop
; підпрограма затримки
wait:
              ldi r19,5              ; зовнішній цикл виконується 5 раз
label:  ldi r17, 255                 ; перший внутрішній цикл виконується
                                     ; 255 раз
label1: ldi r18, 255                 ; другий внутрішній цикл виконується
                                     ; 255 раз
label2: dec r18                      ; зменшити r18 на одиницю
              brne label2            ; поки r18 не буде дорівнювати нулю
              dec r17                ; зменшити r17 на одиницю
```

```
        brne label1         ; поки r17 не буде дорівнювати нулю
        dec r19             ; зменшити r19 на одиницю
        brne label          ; поки r19 не буде дорівнювати нулю
; підпрограма забезпечує затримку приблизно на 5×255×255 тактів
; контролеру
ret                         ; повернення з підпрограми
```

Рекомендації до процесу інсталяції Proteus VSM:

Proteus VSM – це програма-симулятор радіотехнічних схем та мікроконтролерних пристроїв, що підтримує мікроконтролери PIC, 8051, AVR, HC11, ARM7/LPC2000 та інші.

Proteus VSM складається з двох основних модулів:

– ISIS – графічний редактор принципових схем, який використовується для вводу розроблених проектів з подальшою імітацією і передачею для розробки друкованих плат в ARES. Після налагодження пристрою можна відразу розвести друковану плату в ARES, який підтримує авто розміщення і трасування за вже існуючою схемою.

– ARES – графічний редактор друкованих плат з вбудованим менеджером бібліотек і автотрасувальника ELECTRA, автоматичною розстановкою компонентів на друкованій платі.

Для інсталяції Proteus VSM необхідно:

1. Запустити інсталяційний файл.
2. У відкритому вікні вибрати компоненти, які необхідно інсталювати.
3. Виконати настройки, необхідні для інсталяції.

Основні пункти меню Proteus ISIS:

1. Меню File.
2. Меню View.
3. Меню Edit.
4. Меню Library.
5. Меню Tools.
6. Меню Design.
7. Меню Graph.
8. Меню Source.
9. Меню Debug.
10. Меню Template.
11. Меню System.
12. Меню Help.

Створення проекту Proteus ISIS:

1. Запустити **Proteus ISIS.** Найбільший простір відведено під вікно редагування EDIT WINDOW. В ньому відбуваються всі основні процеси створення, редагування та налагодження схеми пристрою. Зліва вгорі знаходиться вікно попереднього перегляду Overview Window за допомогою якого можна переміщуватись по вікну редагування. Під вікном попереднього перегляду знаходиться Object Selector – список обраних в даних момент компонентів, символів та інших елементів. Виділений в списку об'єкт відображається у вікні попереднього перегляду. всі можливі функції та інструменти Proteus ISIS доступні через меню, розташоване вгорі основного вікна програми, через піктограми, що знаходяться під меню і в лівому куті основного вікна та через "гарячі клавіші". Внизу головного вікна розташовані: панель управління інтерактивною симуляцією (кнопки ПУСК-ПОКРОКОВИЙ РЕЖИМ-ПАУЗА-СТОП), рядок статусу (у ньому відображаються помилки, підказки, поточний стан процесу симуляції і т. і.) і координати курсору, що відображаються в дюймах. Для маніпулювання об'єктами їх потрібно виділити за допомогою правої кнопки миші. Для виділення групи можна утримуючи CTRL послідовно натискати праву кнопку на всіх об'єктах або утримуючи праву кнопку протягнути область виділення по необхідним об'єктам. Повторне натискання правої кнопки миші по виділеному об'єкту видаляє його (видалити об'єкти можна ще натиснувши на кнопку DELETE).

2. Відкрийте бібліотеку компонентів. Для цього натисніть кнопку P на клавіатурі, по піктограмі Pick Devices, вибрати елемент Pick Devices в меню Library або двічі натиснувши ліву кнопку у полі вибору компонентів Object Selector. Компоненти можна вибирати за категоріями Category, підкатегоріями Sub-categories, за виробником Manufacturer або ж шукати за ключовими словами Keywords. Для підтвердження вибору компонента натисніть лівою кнопкою по рядку з назвою компонента. Натисніть кнопку OK і помістіть компонент на схему натиснувши два рази ліву кнопку.

3. Зберіть схему, що зображена на рис. 4. З'єднуйте компоненти за допомогою лівої кнопки миші. Якщо необхідно розгорніть їх за допомогою правої кнопки миші. Елементи типу "земля" вибираються в режимі INTER SHEET TERMINAL в лівій частині екрана.

4. Додати hex-файл з програмою в проект. Для цього натисніть правою кнопкою мишки на контролері. Оберіть пункт Edit Properties. У меню Program File виберіть файл з розширенням *.hex у папці, яка створюва-

лась для проекту в AVR Studio. У рядку PROCESSOR CLOCK FREQUENCY виставити тактову частоту процесора. Зберегти проект.

5. Натиснути кнопку ПУСК на панелі управління інтерактивною симуляцією. Подивиться на результат роботи схеми.

6. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

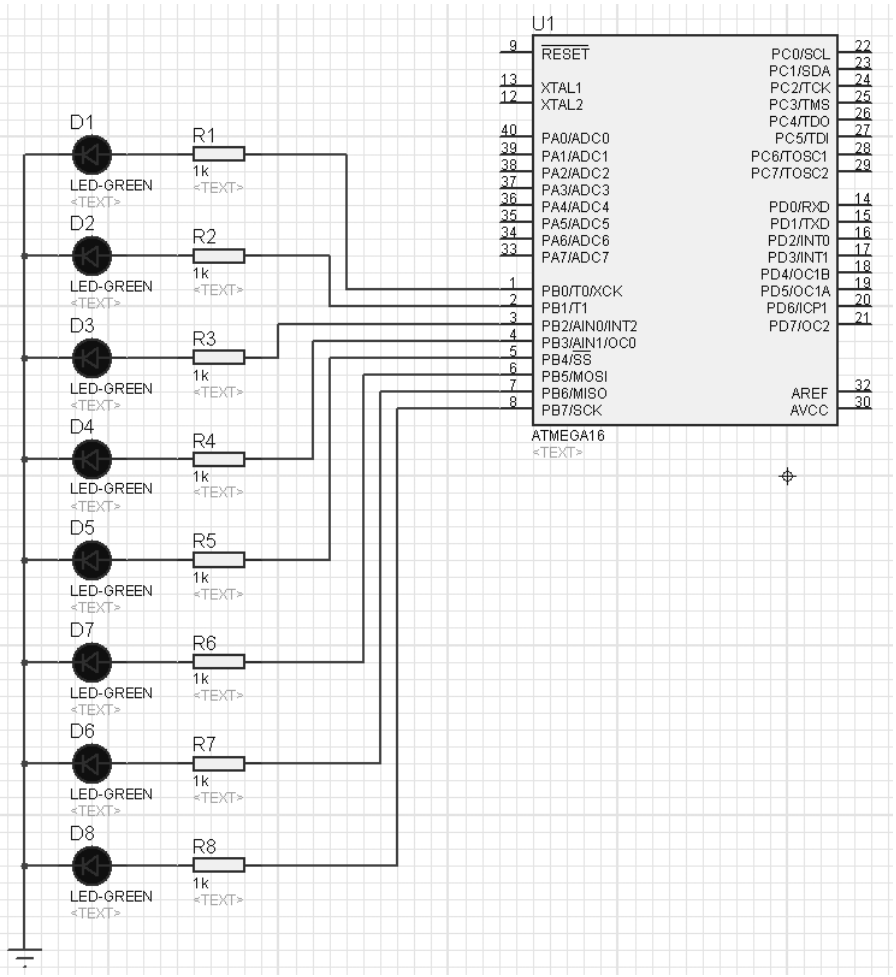


Рис. 4. Приклад створення схеми у програмі Proteus VSM

Контрольні питання:

1. Особливості інтегрованого середовища розробки AVR Studio.
2. Основні пункти меню AVR Studio.
3. Створення проекту в AVR Studio.
4. Створення проекту в Proteus ISIS.

Лабораторна робота №2

Тема: Організація затримок при виводі інформації на порти.

Мета роботи: вивчити команди асемблеру, навчитися організовувати цикли затримки при виводі інформації на порти та візуалізувати отримані результати на світлодіодах у програмі Proteus VSM та відладочному модулі ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

1. Скласти програму, яка із використанням циклу затримки, наведеного в лабораторній роботі №1, забезпечує перемикання світлодіодів відповідно з алгоритмом на порт з часом затримки, що вказані у таблиці.

№ за списком	Алгоритм роботи	Порт	Час затримки, с
1	Послідовно, починаючи з старшого розряду	A	(L+N+M)/2
2	Послідовно, починаючи з молодшого розряду	B	(L+N+M)/3
3	Послідовно, з країв до середини	C	(L+N+M)/4
4	Послідовно, з середини до країв	D	(L+N+M)/5
5	Послідовно, парні розряди порту	D	(L+N+M)/6
6	Послідовно, непарні розряди порту	C	(L+N+M)/7
7	Послідовно, 1, 2, 3, 6, 7, 8 розряд	B	(L+N+M)/8
8	Послідовно, 1, 4, 5, 6, 7, 8 розряд	A	(L+N+M)/9
9	Послідовно, 2, 3, 6, 7 розряд	A	(L+N+M)/10
10	Послідовно, 1, 2, 7, 8 розряд	B	(L+N+M)/11
11	Послідовно, 1, 4, 5, 8 розряд	C	(L+N+M)/12
12	Послідовно, 1+2, 3+4, 5+6, 7+8 розряд	D	(L+N+M)/2
13	Послідовно, 2+3+4, 5+6+7 розряд	D	(L+N+M)/3
14	Послідовно, 1+4, 5+8 розряд	C	(L+N+M)/4
15	Послідовно, 1+8, 4+5 розряд	B	(L+N+M)/5
16	Послідовно, 1+3, 6+8 розряд	A	(L+N+M)/6
17	Послідовно, 1+3, 4+5, 6+8 розряд	A	(L+N+M)/7
18	Послідовно, 1+2, 2+3, 3+4, 4+5, 5+6, 6+7, 7+8 розряд	B	(L+N+M)/8
19	Послідовно, 1+8, 1+7, 1+6, 1+5, 1+4, 1+3, 1+2, 1 розряд	C	(L+N+M)/9
20	Послідовно, 4+5, 2+7, 1+8	D	(L+N+M)/10

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

2. При написанні програми треба враховувати, що світлодіоди підключено до розрядів PB0...PB4 та PD2...PD6 портів B та D відповідно. Вмикаються розряди портів A, B, D при подачі на них сигналу низького рівня.

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Зберіть схему, що зображена на рис. 4. Провести симуляцію роботи схеми. Переконатися, що система працює відповідно з алгоритмом, наведеним у таблиці.

6. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

7. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

8. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16AK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Регістри вводу/виводу.
2. Конфігурування портів вводу/виводу.

Лабораторна робота №3

Тема: Робота з регістрами та портами МК Atmel.

Мета роботи: вивчити команди асемблеру, навчитися виводити на порти A, B, C, D двійкові послідовності та візуалізувати отримані результати на світлодіодах та семисегментних індикаторах у програмі Proteus VSM та відладочному модулі ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

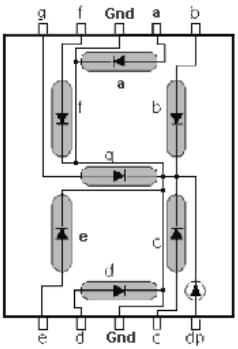
1. Скласти програму, яка виводить вказані значення на відповідні порти МК, навести блок-схему алгоритму до цієї програми.

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

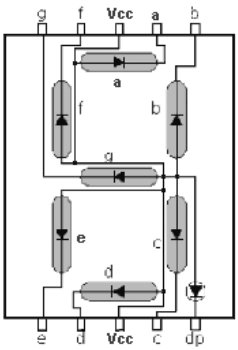
№ за списком	Порт A	Порт B	Порт C	Порт D
1	L	M	N	L
2	L	N	L	M
3	M	M	L	N
4	M	N	N	L
5	N	N	M	M
6	L	N	L	M
7	L	L	M	M
8	M	M	M	N
9	N	N	L	M
10	L	M	N	N
11	N	M	L	L
12	M	L	L	L
13	N	M	M	N
14	L	N	N	L
15	M	L	N	N
16	M	M	N	L
17	N	L	M	M
18	M	M	M	L
19	L	L	L	N
20	M	N	N	N

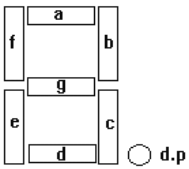
2. При написанні програми треба враховувати, що світлодіоди підключено до розрядів PB0...PB4 та PD2...PD6 портів B та D відповідно. Вмикаються розряди портів A, B, D при подачі на них сигналу низького рівня "0". Семисегментні індикатори вмикаються одиничними сигналами, які поступають на відповідні розряди порту C при ввімкнених розрядах PA0...PA3.

Загальний катод



Загальний анод





3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

6. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16AK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Регістри загального призначення.
2. Регістри вводу/виводу.
3. Регістр SREG.

Лабораторна робота №4

Тема: Організація затримок при виводі чисел на семисегментні індикатори.

Мета роботи: вивчити команди асемблеру, навчитися організовувати цикли затримки при виводі чисел на семисегментні індикатори та візуалізувати отримані результати у програмі Proteus VSM та відладочному модулі ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

1. Скласти програму, яка із використанням циклу затримки, наведеного в лабораторній роботі №1, забезпечує послідовний вивід вказаних значень на вказані індикатори із вказаним для відповідного варіанту часом затримки.

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

№ за списком	Значення	№ індикаторів	Час затримки, с
1	L, L, L	1, 2	(L+N+M)/10
2	L, L, M	2, 3	(L+N+M)/9
3	L, L, N	3, 4	(L+N+M)/8
4	L, M, L	1, 4	(L+N+M)/7
5	L, M, M	1, 3	(L+N+M)/6
6	L, M, N	2, 3	(L+N+M)/5
7	L, N, L	1, 2	(L+N+M)/4
8	L, N, N	1, 4	(L+N+M)/3
9	N, N, L	1, 3	(L+N+M)/2
10	N, N, N	3, 4	(L+N+M)/10
11	M, L, L	1, 4	(L+N+M)/9
12	M, N, N	1, 3	(L+N+M)/8
13	N, M, M	2, 3	(L+N+M)/7
14	M, N, L	3, 4	(L+N+M)/6
15	M, M, M	1, 2	(L+N+M)/5
16	N, M, N	2, 3	(L+N+M)/4
17	M, L, M	2, 4	(L+N+M)/3
18	M, L, N	1, 4	(L+N+M)/2
19	N, N, M	3, 4	(L+N+M)/11
20	N, L, M	1, 2	(L+N+M)/12

2. При написанні програми треба враховувати, що світлодіоди підключено до розрядів PB0...PB4 та PD2...PD6 портів B та D відповідно. Вмикаються розряди портів A, B, D при подачі на них сигналу низького рівня.

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

6. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16AK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Організація оператору розгалуження.

2. Організація оператору циклу.

Лабораторна робота №5.

Тема: Реалізація на модулі ATmega 16 Evaluation Kit 2006 режиму динамічної індикації.

Мета роботи: вивчити команди асемблеру, навчитися організовувати цикли затримки при виводі чисел на семисегментні індикатори в режимі динамічної індікації та візуалізувати отримані результати у програмі Proteus VSM та відладочному модулі ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

1. Скласти програму, в результаті виконання якої здійснюється вивід заданих чисел на семисегментні індикатори в режимі динамічної індикації.

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

№ за списком	Початковий код
1	L, M, N, round ((L+M+N)/5)
2	M, M, N, round ((L+M+N)/6)
3	N, M, L, round ((L+M+N)/7)
4	N, L, L, round ((L+M+N)/8)
5	M, L, L, round ((L+M+N)/9)
6	M, L, N, round ((L+M+N)/10)
7	N, N, N, round ((L+M+N)/9)
8	M, L, M, round ((L+M+N)/8)
9	N, L, N, round ((L+M+N)/7)
10	M, M, M, round ((L+M+N)/6)
11	L, L, L, round ((L+M+N)/5)
12	L, M, M, round ((L+M+N)/11)
13	L, N, L, round ((L+M+N)/12)
14	N, L, M, round ((L+M+N)/13)
15	N, N, M, round ((L+M+N)/14)
16	M, N, N, round ((L+M+N)/10)
17	M, M, L, round ((L+M+N)/9)
18	M, N, M, round ((L+M+N)/8)
19	L, N, N, round ((L+M+N)/7)
20	N, M, N, round ((L+M+N)/6)

Динамічне керування передбачає швидку комутацію розрядів, що індицируються (часове ущільнення). Динамічне керування економить порти мікроконтролеру. Сегменти індикаторів завжди керуються вісім'ю лініями паралельного порту, в той час як поєднані катоди по черзі комутуються транзисторами через інший порт.

Індикація цифр здійснюється по черзі, за рахунок перемикання відповідних розрядів порту А. Для нормального функціонування індикатору мікроконтролер повинен забезпечити достатню частоту перемикання цифр, яка не була б помітною для очей (не менше 40 Гц, тобто час циклу повинен бути не більше 25 мс).

2. При написанні програми треба враховувати, що світлодіоди підключено до розрядів PB0...PB4 та PD2...PD6 портів B та D відповідно. Вмикаються розряди портів A, B, D при подачі на них сигналу низького рівня "0".

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

6. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16AK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Особливості режиму динамічної індикації.
2. Організація оператору циклу.

Лабораторна робота №6

Тема: Вивчення системи переривань, роботи з портами, таймером та симуляція роботи з клавіатурою.

Мета роботи: вивчити групи команд асемблера AVR, навчитися програмувати порти мікроконтролера на необхідний режим роботи, за допомогою підпрограми здійснити опитування порту, до якого може бути підключена клавіатура.

Порядок виконання роботи:

1. Включити комп'ютер та запустити системне програмне забезпечення для симуляції роботи AVR-контролера (AVR-studio).

2. За допомогою клавіатури здійснити набір програми, що відповідає темі роботи.

3. Провести компіляцію та асемблювання програми, виправити синтаксичні похибки.

4. У режимі візуалізації вмісту регістрів, пам'яті, таймера-лічильника і т.і. виконати програму в покроковому та безперервному режимі. За допомогою даних у робочих вікнах AVR-studio розрахувати час виконання програми. Візуалізовані результати виконання програми скопіювати та додати до звіту з лабораторної роботи.

5. Генеровані файли лістінгу, похибок і т.і. роздрукувати та додати до звіту.

6. Розробити детальну блок-схему, яка має відповідати та зображувати основний зміст програми на асемблері.

7. Визначити які групи команд використовуються у програмі, пояснити їх синтаксис.

8. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

9. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

Код програми:

```
.include "m16def.inc"       ; Підключити файл опису імен
                            ; регістрів вводу/виводу
.def temp=r16               ; Завдання символічних імен регістрів
.def last_state=r18         ; Стан клавіатури при минулому
; опитуванні
.def present_state=r19      ; Стан клавіатури при поточному
                            ; опитуванні

; Встановлення вектора переривань таймера 0
.org 0
      rjmp start
.org OVF0addr               ; Адреса вектора переривань по
rjmp tim0                                ; переповненню таймера 0 (ім'я OVF0
; визначено у файлі "m16def.inc")
```

```
start:  ldi temp , LOW(ramend)          ; Ініціалізація покажчика стека
        out spl , temp
        ldi temp , HIGH(ramend)
        out sph , temp

        rcall port_init                 ; Ініціалізація портів
        rcall tim0_init                 ; Ініціалізація таймера 0
        sei                             ; Встановлення глобального біта
; дозволення переривань

Loop:
        rjmp Loop

; Підпрограма обробки переривань від таймера 0
tim0:   push temp                       ; Зберігання регістрів в стек
        in temp,sreg
        push temp

        com last_state                  ; Якщо минулого разу кнопка була
; натиснута ("0")
        in present_state,pina
        and last_state , present_state  ; і цього разу відпущена ("1"), то
        in temp , portb
        eor temp , last_state           ; інвертувати біт порту B
        out portb , temp
        mov last_state , present_state  ; Зберегти поточний стан клавіатури до
                                        ; наступного входу в підпрограму

        pop temp                        ; Відновлення регістрів з стеку
        out sreg , temp
        pop temp
        reti

; Підпрограма ініціалізації портів
port_init:
        ldi temp , $ff          ; Програмування порту B на вивід
        out ddrb , temp
```

```
        ldi temp , $00              ; Запалити всі світлодіоди
        out portb , temp
        out ddra , temp             ; Програмування порту A на введення
        ldi last_state , $ff        ; Вважаємо, що спочатку жодна кнопка не
ret                                 ; натиснута

; Підпрограма ініціалізації таймеру 0
tim0_init:
        ldi temp , $5
        out tccr0 , temp
        ldi temp , (1<<toie0)
        out timsk , temp
ret
```

Контрольні питання:

1. Таблиці векторів переривань.
2. Обробка переривань.
3. Зовнішні переривання.
4. Різниця між таймерами та лічильниками.
5. Переривання від таймерів/лічильників.

Лабораторна робота №7

Тема: Зміна коду, відображеного на індикаторах, при натисканні на кнопку.

Мета роботи: надбати навички у написанні програм на асемблері, вивчити команди асемблеру, навчитися організовувати цикли затримки при виводі інформації в режимі динамічної індикації, візуалізувати отримані результати на індікаторах відладочного модулю ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

1. Скласти програму, яка відповідає наведеному алгоритму та змінює числа, виведені в режимі динамічної індикації шляхом додавання до них величини *n*. Якщо отримане число більше 10, то зменшити його на 10.

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

№ за списком	Початковий код	*n*
1	N, M, L	round ((L+M+N)/6)
2	N, N, N	round ((L+M+N)/7)
3	L, L, L	round ((L+M+N)/8)
4	M, M, M	round ((L+M+N)/9)
5	L, N, N	round ((L+M+N)/6)
6	M, N, M	round ((L+M+N)/7)
7	M, L, L	round ((L+M+N)/8)
8	M, M, L	round ((L+M+N)/9)
9	M, N, N	round ((L+M+N)/5)
10	N, L, N	round ((L+M+N)/10)
11	M, L, N	round ((L+M+N)/9)
12	M, M, N	round ((L+M+N)/8)
13	M, N, M	round ((L+M+N)/7)
14	N, N, L	round ((L+M+N)/6)
15	N, M, M	round ((L+M+N)/9)
16	M, N, L	round ((L+M+N)/7)
17	M, N, M	round ((L+M+N)/6)
18	L, M, N	round ((L+M+N)/8)
19	L, L, N	round ((L+M+N)/6)
20	L, L, M	round ((L+M+N)/7)

2. При написанні програми треба враховувати, що вмикаються розряди порту А при подачі на них сигналу низького рівня.

Кнопка 1 підключена до PA.4, кнопка 2 – до PA.2, кнопка 3 – до PA.3.

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

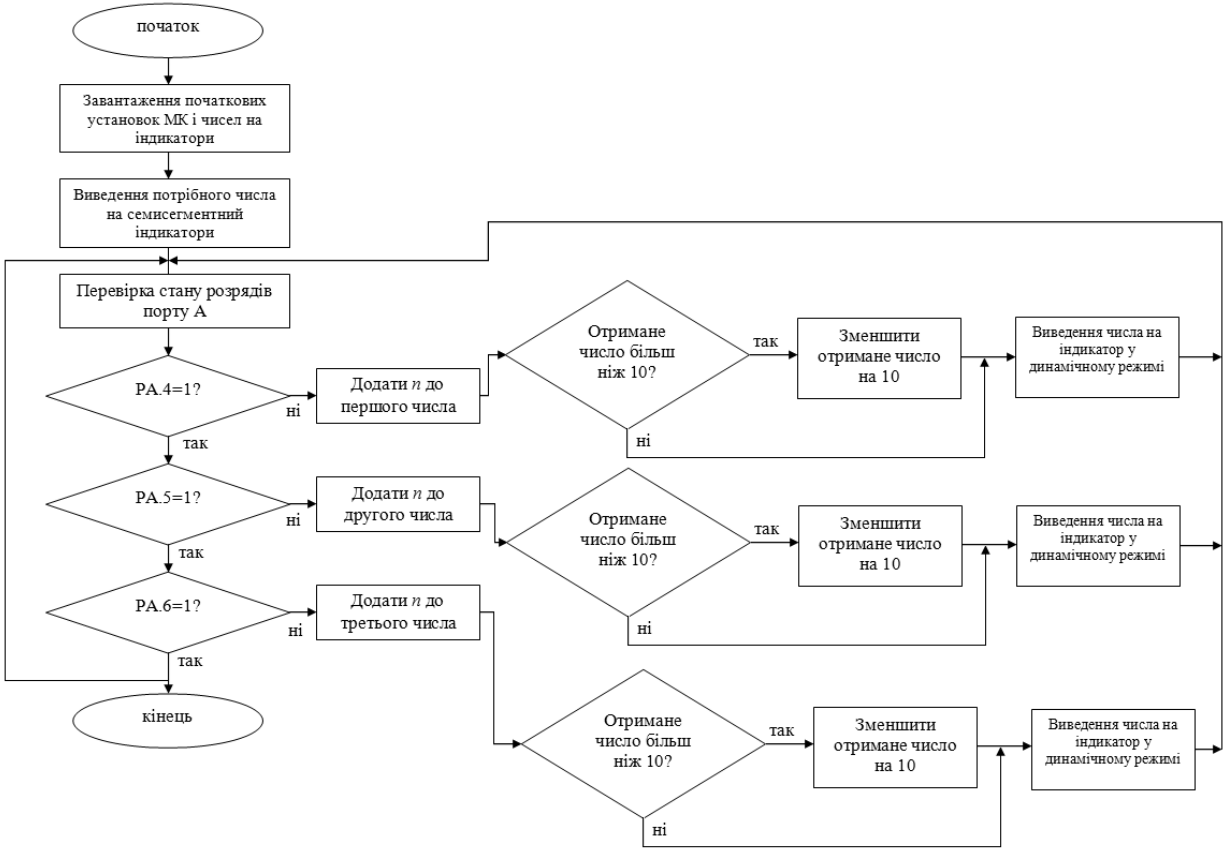
5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

6. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16EK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Особливості режиму динамічної індикації.
2. Організація оператору циклу.


початок
Завантаження початкових установок МК і чисел на індикатори
Виведення потрібного числа на семисегментний індикатори
Перевірка стану розрядів порту А
PA.4=1?
ні
так
Додати n до першого числа
Отримане число більш ніж 10?
так
ні
Зменшити отримане число на 10
Виведення числа на індикатор у динамічному режимі
PA.5=1?
ні
так
Додати n до другого числа
Отримане число більш ніж 10?
так
ні
Зменшити отримане число на 10
Виведення числа на індикатор у динамічному режимі
PA.6=1?
ні
так
Додати n до третього числа
Отримане число більш ніж 10?
так
ні
Зменшити отримане число на 10
Виведення числа на індикатор у динамічному режимі
кінець

Лабораторна робота №8

Тема: Вивід заданих чисел на індикатори в динамічному режимі.

Мета роботи: надбати навички у написанні програм на асемблері, вивчити команди асемблеру, навчитися організовувати цикли затримки при виводі інформації в режимі динамічної індикації, візуалізувати отримані результати на індікаторах відладочного модулю ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

1. Скласти програму, в результаті виконання якої здійсняється вивід заданих чисел на задані семисегментні індикатори в режимі динамічної індикації. Зміна чисел здійсняється через вказані проміжки часу.

Динамічне керування передбачає швидку комутацію розрядів, що індицируються (часове ущільнення). Динамічне керування економить порти мікроконтролеру. Сегменти індикаторів завжди керуються вісім'ю лініями паралельного порту, в той час як поєднані катоди по черзі комутуються транзисторами через інший порт.

Індикація цифр здійсняється по черзі, за рахунок перемикання відповідних розрядів порту А. Для нормального функціонування індикатору мікроконтролер повинен забезпечити достатню частоту перемикання цифр, яка не була б помітною для очей (не менше 40 Гц, тобто час циклу повинен бути не більше 25 мс).

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

№ за списком	Індикатори	Числа	Час затримки, с
1	1, 2	L, N, M	(L+M+N)/6
2	2, 3	L, L, L	(L+M+N)/7
3	3, 4	M, N, M	(L+M+N)/8
4	1, 4	N, L, M	(L+M+N)/9
5	1, 2	M, M, L	(L+M+N)/10
6	2, 3	N, L, L	(L+M+N)/11
7	3, 4	L, M, L	(L+M+N)/12
8	1, 4	M, L, N	(L+M+N)/6
9	1, 2	N, L, N	(L+M+N)/7
10	2, 3	M, M, M	(L+M+N)/8
11	3, 4	N, N, N	(L+M+N)/9

Продовж. таблиці

№ за списком	Індикатори	Числа	Час затримки, с
12	1, 4	N, N, M	(L+M+N)/10
13	1, 2	L, N, L	(L+M+N)/11
14	2, 3	N, N, L	(L+M+N)/12
15	3, 4	L, L, M	(L+M+N)/6
16	1, 4	M, M, L	(L+M+N)/7
17	1, 2	N, M, L	(L+M+N)/8
18	2, 3	M, M, L	(L+M+N)/9
19	3, 4	M, M, N	(L+M+N)/10
20	1, 4	N, M, M	(L+M+N)/11

2. При написанні програми треба враховувати, що світлодіоди підключено до розрядів PB0...PB4 та PD2...PD6 портів B та D відповідно. Вмикаються розряди портів A, B, D при подачі на них сигналу низького рівня "0".

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

6. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16EK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Особливості режиму динамічної індикації.
2. Організація оператору циклу.

Лабораторна робота №9

Тема: Реєстрація кількості натискання кнопки.

Мета роботи: вивчити команди асемблеру, навчитися організовувати цикли затримки при виводі чисел на семисегментні індикатори та візуалізувати отримані результати на модулі ATmega16 Evaluation Kit 2006.

Порядок виконання роботи:

1. Скласти програму, яка із використанням циклу затримки, забезпечує вивід на семисегментні індикатори кількість натискання кнопки. Вивід на індикатори здійснюється, якщо протягом вказаного часу не відбувається повторного натискання на кнопку. Програма повинна відповідати наведеному нижче алгоритму.

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

№ за списком	№ індикаторів	Час затримки, с
1	1, 2	(L+M+N)/9
2	2, 3	(L+M+N)/10
3	3, 4	(L+M+N)/11
4	1, 4	(L+M+N)/12
5	1, 3	(L+M+N)/6
6	1, 2	(L+M+N)/7
7	2, 3	(L+M+N)/8
8	3, 4	(L+M+N)/10
9	1, 4	(L+M+N)/9
10	1, 3	(L+M+N)/11
11	1, 2	(L+M+N)/12
12	2, 3	(L+M+N)/10
13	3, 4	(L+M+N)/11
14	1, 4	(L+M+N)/12
15	1, 3	(L+M+N)/10
16	1, 2	(L+M+N)/9
17	2, 3	(L+M+N)/8
18	3, 4	(L+M+N)/7
19	1, 4	(L+M+N)/6
20	1, 3	(L+M+N)/5

2. При написанні програми треба враховувати, що вмикаються розряди порту A при подачі на них сигналу низького рівня.

Кнопка 1 підключена до PA4, кнопка 2 – до PA5, кнопка 3 – до PA6.

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

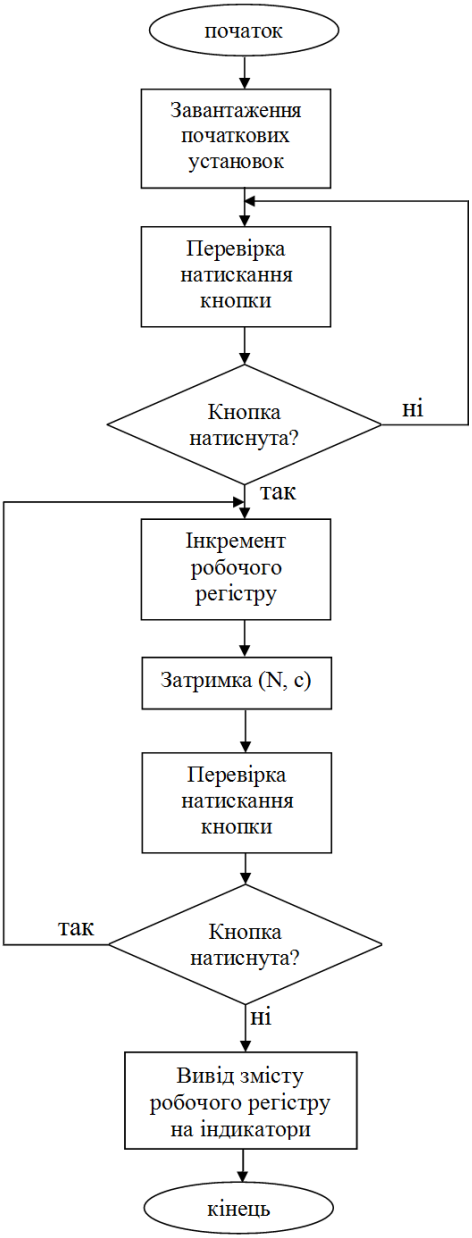
6. Пред'явити виконане завдання в програмі Proteus VSM в елек-тронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі А16ЕК_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Особливості режиму динамічної індикації.

2. Організація оператору циклу.



Лабораторна робота №10

Тема: Робота з символьним РКІ-дисплеєм.

Мета роботи: вивчити методи та засоби програмування алфавітно-цифрових РКІ-дисплеєв на основі контролеру HD44780 за допомогою ATMega16.

Порядок виконання роботи:

1. Включити комп'ютер та запустити системне програмне забезпечення для симуляції роботи AVR-контролера (AVR-studio).

2. Скласти програму, яка із використанням циклу затримки, забезпечує вивід прізвища, ім'я та по-батькові (латиницею), у режимі вказаному в таблиці, на символьний РКІ-індикатор, сумісний з контролером HD44780.

В таблиці L – кількість літер у прізвищі, M – кількість літер в імені, N – кількість літер у по батькові. Якщо L, M, N > 10, то скласти десятки та одиниці числа.

№ за списком	Шина даних	Напрям зсуву	Мерехтливе знакомісце	Підкреслення	Час затримки між словами, с
1	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/9
2	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/10
3	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/11
4	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/12
5	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/6
6	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/7
7	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/8
8	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/10
9	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/9
10	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/11
11	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/12
12	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/10
13	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/11
14	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/12
15	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/10
16	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/9
17	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/8
18	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/7
19	4-х розрядна	вліво	увімкнуте	вимкнуте	(L+M+N)/6
20	8-ми розрядна	вправо	вимкнуте	увімкнуте	(L+M+N)/5

Контролер HD44780 фірми Hitachi фактично є промисловим стандартом і широко застосовується при виробництві алфавітно-цифрових РКІ-модулів. Аналоги цього контролера або сумісні з ним по інтерфейсу і командній мові мікросхеми, випускають безліч фірм, серед яких: Epson, Toshiba, Sanyo, Samsung, Philips. Ще більша кількість фірм виробляють РКІ -модулі на базі даних контролерів. Ці модулі можна зустріти в найрізноманітніших пристроях: вимірювальних приладах, медичному обладнанні, промишленнном і технологічному обладнанні, офісної техніки – принтерах, телефонах, факсимільних і копіювальних апаратах.

Алфавітно-цифрові РК-модулі – недороге і зручне рішення, що дозволяє заощадити час і ресурси при розробці нових виробів, при цьому забезпечують відображення великого обсягу інформації при гарному розпізнаванні і низькому енергоспоживанні. Можливість оснащення РКІ-модулів заднім підсвічуванням дозволяє експлуатувати їх в умовах зі зниженою або нульовою освітленістю, а виконання з розширеним діапазоном температур (–20 °С…+70 °С) – в складних експлуатаційних умовах, у тому числі в переносних, польових і навіть, іноді, в бортових пристроях.

Для з'єднання РКІ-модуля з керуючою системою використовується паралельна синхронна шина, яка налічує 8 чи 4 (вибирається програмно) ліній даних DB0...DB7, лінію вибору операції R/W, лінію вибору регістра RS і лінію стробування/синхронізації E. Крім ліній керуючої шини є дві лінії для подачі напруги живлення 5 В – GND і VCC, і лінія для подачі напруги живлення драйвера РКІ – V_0. Плавна зміна напруги живлення драйвера РКІ призводить до зміни кута повороту рідких кристалів. Таким чином, можна відрегулювати фактичну контрастність дисплея при деякому переважному куті спостереження (знизу-вверх або зверху-вниз).

Для з'єднання модуля з керуючою системою можна вибрати один з двох варіантів: по 8-ми або 4-х розрядній шині. У першому випадку потрібно 11 сигнальних ліній, у другому – лише 7.

На рис. 5,***а*** наведена схема підключення РКІ-модуля з 8-ми розрядною шиною до деякої абстрактного мікроконтролера. Цей мікроконтролер містить два порти: 8-ми розрядний двонаправлений PA0 ... PA7, до якого підключена шина DB0 ... DB7 РКІ-модуля і 3-х розрядний PB0 ... ... PB2, до якого підключені лінії керуючих сигналів: E, RS, R/W. На рис. 5,***б*** можна побачити схему підключення РКІ-модуля до цього ж МК в 4-х розрядному режимі. Зверніть увагу, що для обміну в 4-х розрядному режимі використовується старша тетрада шини даних – DB4 ... DB7.

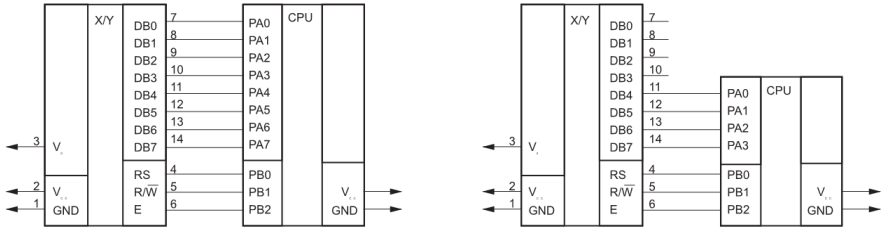


Рис. 5

У початковому стані сигнал E = 0, сигнал R/W = 0, значення сигналу RS – довільне, шина даних DB0 ... DB7 в стані високого імпедансу (HI). Такий стан керуючих сигналів (E і R/W) повинен підтримуватися весь час в проміжках між операціями обміну з РКІ-модулем. Шина даних в ці моменти вільна і може використовуватися в мультиплексному режимі для будь-яких інших цілей.

Послідовності дій, які необхідно виконувати керуючої системі при здійсненні операцій запису і читання для 8-ми і 4-х розрядної шини наведені відповідно у додатку 3.

Спрощена структурна схема контролера приведена на рис. 6. Можна відразу виділити основні елементи з якими доводиться взаємодіяти при програмному управлінні: регістр даних (DR), реєстр команд (IR), відеопам'ять (DDRAM), ОЗУ знакогенератора (CGRAM), лічильник адреси пам'яті (AC), прапор зайнятості контролера.

Управління контролером ведеться за допомогою інтерфейсу керуючої системи. Основними об'єктами взаємодії є регістри DR і IR. Вибір адресується регістра виробляється лінією RS, якщо RS = 0 – адресується регістр команд (IR), якщо RS = 1 – регістр даних (DR).

Дані через регістр DR, залежно від поточного режиму, можуть поміщатися (або прочитуватися) у відеопам'ять (DDRAM) або в ОЗУ знакогенератора (CGRAM) за поточною адресою, вказаною лічильником адреси (AC). Інформація, яка потрапляє в регістр IR, інтерпретується пристроєм виконання команд як керуюча послідовність. Прочитання регістра IR повертає в 7-ми молодших розрядах поточне значення лічильника AC, а в старшому розряді прапор зайнятості (BF).

Відеопам'ять, що має загальний об'єм 80 байтів, призначена для зберігання кодів символів, що відображаються на РК. Відеопам'ять органі-

зована у два рядки по 40 символів в кожній. Ця прив'язка є жорсткою і не підлягає зміні. Іншими словами, незалежно від того, скільки реальних рядків матиме кожен конкретний РКІ-модуль, адресація відеопам'яті завжди проводиться як до двох рядках по 40 символів.

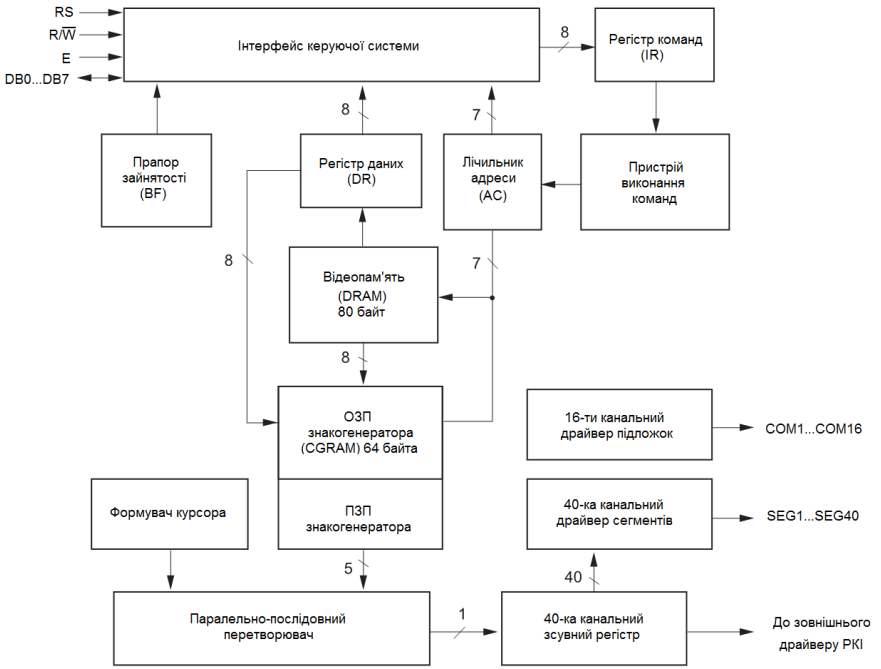


Рис. 6

Оскільки РКІ є пристроєм з динамічною індикацією, контролер циклічно виробляє оновлення інформації на РКІ. Сам РКІ організований як матриця, що складається в залежності від режиму роботи з 8-ми (один рядок символів точок), 11-ти (один рядок символів точок) або 16-ти (два рядки символів точок) рядків по 200 сегментів (коли рядок налічує 40 символів) у кожній.

У контролера HD44780 існує набір внутрішніх прапорів, що визначають режими роботи різних елементів контролера (Додаток 4). У додатку 4 також наведені значення керуючих прапорів безпосередньо після подачі на РКІ-модуль напруги живлення. Перевизначення значень пра-

порів проводиться спеціальними командами, що записуються в регістр IR, при цьому комбінації старших бітів визначають групу прапорів або команду, а молодші містять власне прапори.

Список керуючих комбінацій бітів регістра IR і виконувані ними команди наведені в додатку 4. На момент включення РКІ-модуль нічого не відображає (прапор D = 0), тому, щоб вивести який-небудь текст необхідно, як мінімум, включити відображення, встановивши прапор D = 1. Ось приклад широко поширеної послідовності для ініціалізації РКІ-модуля: 38h, 0Ch, 06h (знак "h" – вказує на 16-річну систему числення). 38h – встановлює режим відображення – два рядка з матрицею точок 5×8 і роботу з 8-ми розрядною шиною даних; 0Ch – включає відображення на екрані РКІ-модуля, без відображення курсорів; 06h – встановлює режим автоматичного переміщення курсору зліва-направо після виведення кожного символу.

Контролер HD44780 підтримує як операції запису так і операції читання. Читання регістра DR призводить до завантаження вмісту DDRAM або CGRAM, залежно від поточного режиму, при цьому курсор зміщується на одну позицію, як і при записі. Читання регістра IR повертає 8 значущих розрядів, причому в 7-ми молодших міститься поточне значення лічильника AC (7 розрядів, якщо адресується DDRAM, і 6 – якщо CGRAM), а у старшому – прапор зайнятості BF. Цей прапор має значення 1 коли контролер зайнятий і 0 – коли вільний. Необхідно враховувати, що більшість операцій, які виконуються контролером, займають значний час, близько 40 мкс, а час виконання деяких доходить до одиниць мілісекунд, тому цикл очікування зняття прапора BF повинен обов'язково бути присутнім у програмах драйвера РКІ-модуля і передувати здійсненню будь-якої операції (природно, крім операції перевірки прапора BF). тому він дуже зручний для вітчизняних застосувань.

Виведення на екран символу проводиться записом його коду в регістр DR. При цьому символ розміщується в DDRAM за поточним адресою, вказаною AC, а значення AC збільшується або зменшується на 1. Щоб зробити переустановку курсору на потрібну позицію, потрібно призначити AC відповідне значення. Коли проводиться послідовний запис символів і в результаті заповнюється весь рядок, курсор автоматично переходить на другий рядок, але якщо необхідно примусово встановити курсор, скажімо, на початок другого рядка, то буде невірним привласнити AC здавалося б логічне значення 28h, правильним є значення 40h. Значення

адрес DDRAM в діапазоні 28h ... 3Fh (а також і 68h ... 7Fh) є невизначеними і результати роботи з ними можуть бути непередбачуваними. Необхідно враховувати, що контролери, що встановлюються на РКІ-модулі, можуть мати різні набори символів, причому це може залежати як від виробника контролера, так і від модифікації даної конкретної моделі.

З допустимих для розміщення в DDRAM кодів символи з кодами 00h...07h (і їх дублікат з кодами 08h...0Fh) мають спеціальне призначення – це символи, що можуть перевизначатися, графічне зображення яких може призначити сам споживач, розмістивши відповідну інформацію в області CGRAM. Для програмування доступно 8 символів, що перевизначаються, в режимі з матрицею 5×7 точок і 4 з матрицею 5×10 (в режимі 5×10 символи, що перевизначаються, адресуються кодами DDRAM через один: 00h, 02h, 04h, 06h) (додаток 5).

Виробник контролера рекомендує виконувати наступну послідовність дій для ініціалізації. Витримати паузу не менше 15 мс між встановленням робочої напруги живлення (> 4,5 В) і виконанням будь-яких операцій з контролером. Першою операцією виконати команду, вибирає розрядність шини (це повинна бути команда 30h незалежно від того, який розрядності інтерфейс ви збираєтеся використовувати в подальшому), причому перед виконанням цієї операції не перевіряти значення прапора BF. Далі знову витримати паузу не менше 4,1 мс і повторити команду вибору розрядності шини, причому перед подачею команди знову не проводити перевірку прапора BF. Наступним кроком необхідно знову витримати паузу, на цей раз 100 мкс, і втретє повторити команду встановлення розрядності шини, знову без перевірки BF. Ці три операції – ініціалізація, що покликана вивести контролер у початковий режим роботи (тобто перевести в режим роботи з 8-ми розрядною шиною) з будь-якого стану. Слідом за ними нормальним порядком (без витримування пауз, але з перевіркою прапора BF) виконується ініціалізація режимів роботи з видачею послідовності, що ініціалізує, аналогічній зазначеної в додатку 4 (що містить у тому числі команду вибору необхідної розрядності шини).

При переході у режим з 4-х розрядною шиною, тобто при команді $20, з 8-ми розрядного режиму, який встановлюється автоматично після подачі напруги живлення, неможливо адекватно оголосити необхідне значення прапорів N і F, що розташовуються в молодшій тетраді команди установки розрядності шини. Тому команду необхідно повторити у вже сталому 4-х розрядному режимі шляхом послідовної передачі двох тетрад, тобто відповідним чином для 4-х розрядного режиму.

3. Створити AVR проект, підключити файл з написаною програмою до проекту.

4. Провести компіляцію та асемблювання програми, виправити синтаксичні помилки.

5. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

6. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

7. Виконати "прошивання" МК за допомогою програми PonyProg2000 та перевірити вірність виконання роботи на модулі A16EK_2006, побудованому на основі ATmega16.

Контрольні питання:

1. Яким чином здійснюється організація роботи з РКІ?

2. Чим відрізняються між собою режими читання і запису для 8-ми і 4-х розрядної шини даних?

3. Призначення регістра IR і DR?

4. Призначення бітів регістра IR?

Лабораторна робота №11

Тема: Способи організації роботи послідовного асинхронного адаптера UART AVR.

Мета роботи: вивчити групи команд асемблера AVR, надбати навички по програмному тестуванню апаратних вузлів послідовного інтерфейсу.

Порядок виконання роботи:

1. Включити комп'ютер та запустити системне програмне забезпечення для симуляції роботи AVR-контролера (AVR-studio).

2. За допомогою клавіатури здійснити набір програми, що відповідає темі та меті роботи.

Параметри асинхронної передачі даних:

– швидкість – 9600 біт/с

– кількість біт даних – 8

– контроль парності – відсутній

– кількість стоп бітів – 1.

3. Провести компіляцію та асемблювання програми, виправити синтаксичні похибки.

4. У режимі візуалізації вмісту регістрів, пам'яті, таймера-лічильника і т. і. виконати програму в покроковому та безперервному режимі. За допомогою даних у робочих вікнах AVR-studio розрахувати час виконання програми.

Візуалізовані результати виконання програми скопіювати та додати до звіту з лабораторної роботи.

5. Генеровані файли лістінгу, похибок і т.і. роздрукувати та додати до звіту.

6. Розробити детальну блок-схему, яка має відповідати та зображувати основний зміст і особливості програми на асемблері.

7. Визначити які групи команд використовуються у програмі, пояснити синтаксис запису директив асемблера.

8. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

9. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

```
.include "8515def.inc"          ; Підключити файл опису імен
                                ; регістрів вводу/виводу
.def temp=r16                   ; Завдання символьних імен
; регістрів
.def last_state=r18             ; Стан клавіатури при минулому
; опитуванні
.def present_state=r19          ; Стан клавіатури при поточному
; опитуванні
.def count=r20                  ; Лічильник циклу
.def serial=r21                 ; Дані для передачі в послідовний
; канал

; Встановлення векторів переривання
.org 0
        rjmp start
.org OVF0addr                   ; Адреса вектора переривань по
rjmp tim0                       ; переповненню таймера 0 (ім'я OVF0
; визначено у файлі "8515def.inc")
```

```
; Ініціалізація
start : ldi temp , LOW(ramend)          ; Ініціалізація покажчика стека
        out spl , temp
        ldi temp , HIGH(ramend)
        out sph , temp

        rcall port_init                 ; Ініціалізація портів
        rcall tim0_init                 ; Ініціалізація таймера 0
        rcall uart_init                 ; Ініціалізація UART
        sei                             ; Встановлення глобального біта
; дозволення переривань
        ; Нескінченний цикл
Loop:
        rjmp Loop

; Підпрограма обробки переривань від таймера 0
; викликається приблизно 1 раз на 70 мс
tim0:   push temp                       ; Зберігання регістрів в стек
        in temp , sreg
        push temp

        com last_state                  ; Якщо минулого разу кнопка була
; натиснута ("0")
        in present_state , pina
        and last_state , present_state  ; і цього разу відпущена ("1"), то
        in temp , portb
        eor temp , last_state           ; інвертувати стан біту порту B
        out portb , temp
        mov last_state , present_state  ; Зберегти поточний стан клавіатури
;для наступного входження в підпрограму

; Для кожного розряду порту B надсилаємо в послідовний канал
; ASCII-код "0" ($30) або "1" ($31) в залежності від стану розряду
        ldi count , 8                   ; Ініціалізувати лічильник розрядів
next_bit:
        rol temp                        ;Зсунути в біт C старший біт регі-
стра temp
```

```
        clr serial              ; Очистити регістр
        rol serial              ; Помістити в нього біт C
        subi serial,-$30        ; Отримати ASCII-код
        rcall Trans             ; Передати його в буфер послідовного
; каналу
        dec count               ; Зменшити на 1 лічильник кількості біт
        brne next_bit           ; Якщо біти ще лишились, повернутися
; назад
        ldi serial , $0a        ; Передати в послідовний канал керуючий
        rcall trans             ; ASCII-код "переведення рядку"
        ldi serial , $0d        ; Передати в послідовний канал керуючий
        rcall trans             ; ASCII-код "повернення каретки"

        rcall receive

        pop temp                ; Відновлення регістрів з стеку
        out sreg , temp
        pop temp
        reti

; Підпрограма передачі байту в послідовний канал
trans:  sbis USR , UDRE         ; Якщо передавач ще не передав
; попередній байт,
        rjmp trans              ; то повернутись до trans і чекати.
        out UDR , serial        ; В іншому випадку вивести вміст serial в
; UART
        ret                     ; Вийти з підпрограми

; Підпрограма опитування приймача послідовного каналу
receive:
        sbic USR , RXC          ; Якщо флаг RXC в USR скинутий, тобто
; в приймачі нема нових байт, то
; пропустити наступну команду
        rjmp input              ; В іншому випадку перейти до
; зчитування байта з приймача
        ret                     ; Вийти з підпрограми

input:  in temp , UDR           ; Зчитати байт з приймача
        out portb , temp        ; та вивести його в порт B
        ret
```

```
; Підпрограма ініціалізації UART
uart_init:
        ldi temp , 0b00011000      ; Ініціалізація UART
        out UCR , temp             ; TXEN=1, RXEN=1
        ldi temp , 23              ; 9600 біт/с при  fclk=3,68МГц
        out UBRR , temp
        ret
```

Підпрограми "port_init" та "tim0_init" наведені в лабораторній роботі №3.

Контрольні питання:

1. Використання модулю USART.
2. Призначення регістрів UCSRA, UCSRB, UCSRC.
3. Формат кадру.

Лабораторна робота №12

Тема: Передавання даних в послідовний канал.

Мета роботи: навчитися виконувати передавання даних в послідовний канал без опитування флага готовності із використанням переривання передавача по готовності та проміжний кільцевий буфер FIFO.

В даній роботі драйвер послідовного каналу на передавання містить дві програми – "MOVSBUF" і "TRANSINT" (рис. 7).

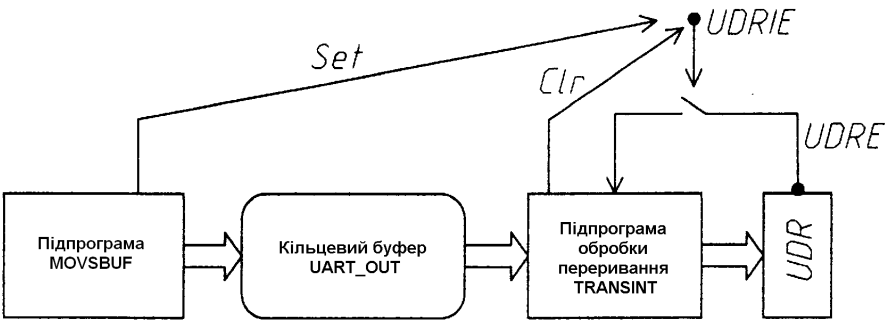


Рис. 7 Організація програми передавання даних
в послідовний канал із використанням кільцевого буфера

Кільцевий буфер UART_OUT організується у внутрішньої оперативній пам'яті AVR (рис. 8).

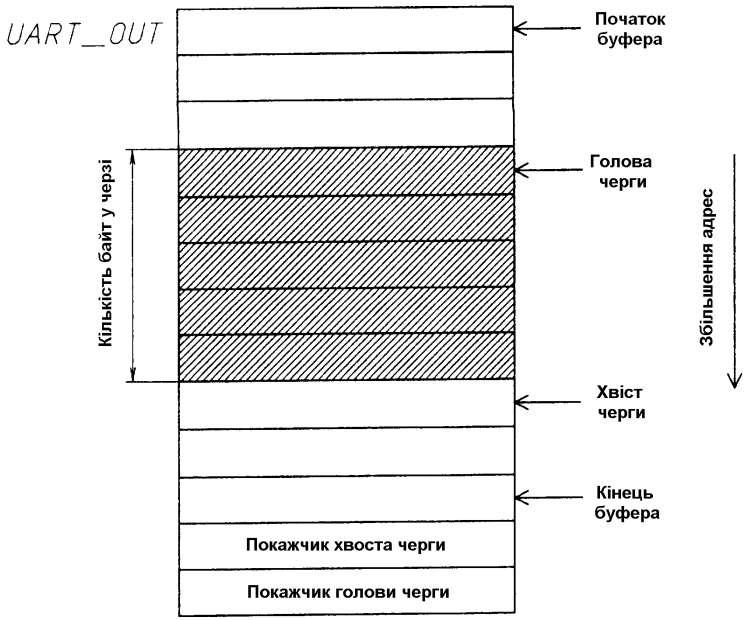


Рис. 8 Організація буфера FIFO в пам'яті

В пам'яті також містяться дві змінних покажчика голови черги (HEAD) та її хвіст (TAIL). Якщо буфер пустий, обидва покажчика є рівними.

Виклик підпрограми "MOVSBUF" виконується із основної програми. Підпрограма без очікування готовності передавача, заносить в буфер UART_OUT необхідну кількість байт. Підпрограма TRANSINT викликається по перериванню при установці флага UDRE, коли передавач UART готовий прийняти в UDR новий байт. Підпрограма TRANSINT переносить один байт із черги UART_OUT в регістр даних передавача UART (UDR). Обидві підпрограми потребують від процесора мінімальних витрат часу.

Підпрограма MOVSBUF переносить один байт в буфер за адресою, на яку вказує покажчик хвоста черги, та збільшує значення цього покажчика. Чим більше байт в буфері, тим більшим є розрив між покажчиками.

Програма зчитування із буфера TRANSINT зчитує байт із комірки пам'яті, на яку вказує покажчик голови черги, та збільшує значення цього покажчика. Таким чином покажчики переміщуються по кільцю прагнучи дістати один одного. Коли буфер звільняється, покажчик голови дістає покажчика хвоста і навпаки.

Підпрограма TRANSINT після передавання із черги в UART чергового байта перевіряє, чи є вільним буфер. Якщо усі дані передані, вона скидає біт UDRE та забороняє всі запити на переривання. Встановлення передавання даних виконує програма MOVSBUF, після виконання нового запису в буфер. За переповненням буфера стежить підпрограма MOVSBUF. Вона ігнорує черговий байт, який головна програма прагне занести до черги, якщо буфер заповнено.

Порядок виконання роботи:

1. Включити комп'ютер та запустити системне програмне забезпечення для симуляції роботи AVR-контролера (AVR-studio).

2. За допомогою клавіатури здійснити набір програми, що відповідає темі та меті роботи.

3. Провести компіляцію та асемблювання програми, виправити синтаксичні похибки.

4. У режимі візуалізації вмісту регістрів, пам'яті, таймера-лічильника і т. і. виконати програму в покроковому та безперервному режимі. За допомогою даних у робочих вікнах AVR-studio розрахувати час виконання програми.

Візуалізовані результати виконання програми скопіювати та додати до звіту з лабораторної роботи.

5. Порівняти час виконання програми TIM0 в даній та попередній лабораторних роботах.

6. Генеровані файли лістінгу, похибок і т.і. роздрукувати та додати до звіту.

7. Розробити детальну блок-схему, яка має відповідати та зображувати основний зміст і особливості програми на асемблері.

8. Визначити які групи команд використовуються у програмі, пояснити синтаксис запису директив асемблера.

9. Оформити звіт з лабораторної роботи згідно з ЄСКД. Висновки до роботи мають стисло та зрозуміло пояснювати результати отримані на кожному кроці виконання роботи, згідно з наведеним вище порядком.

10. Пред'явити виконане завдання в програмі Proteus VSM в електронному вигляді на прохання викладача.

Код програми:

```
.include "8515def.inc"
.def temp=r16
.def last_state=r18
.def present_state=r19
.def count=r20                  ; Лічильник циклу
.def serial=r21                 ; Дані для передачі в послідовний
; канал
; Визначення символічних імен для констант
.equ UART_OUT=$c0               ; Адрес буфера UART_OUT в RAM
.equ UART_OUT_T=$be             ; Адрес покажчика на хвіст черги
; UART_OUT
.equ UART_OUT_H=$bf             ; Адрес покажчика на хвіст черги
; UART_OUT
; Встановлення векторів переривання
.org 0
        rjmp start
.org OVF0addr
        rjmp tim0
.org UDREaddr                   ; Адрес вектора переривання по флагу UDRE
        rjmp TransInt           ; готовності передавача UART

; Ініціалізація
start:  ldi temp,LOW(ramend)    ; Ініціалізація покажчика стека
        out spl,temp
        ldi temp,HIGH(ramend)
        out sph,temp

        rcall port_init         ; Ініціалізація портів
        rcall tim0_init         ; Ініціалізація таймера 0
        rcall uart_init         ; Ініціалізація UART
        sei

Loop:
        rjmp Loop

; Підпрограма обробки переривань від таймера 0
; викликається приблизно 1 раз в 70 мс.
    tim0:   push temp
        in temp,sreg
        push temp
```

```
        com  last_state
        in present_state,pina
        and  last_state,present_state
        in temp,portb
        eor  temp,last_state
        out portb,temp
        mov  last_state,present_state

; Для кожного розряду порту B посилаємо в послідовний канал
; ASCII-код "0" ($30) або "1" ($31) залежно від стану розряду
        ldi count,8
next_bit:
        rol temp
        clr serial
        rol serial
        subi serial,-$30
        rcall MovsBuf
        dec count
        brne next_bit
        ldi serial,$0a
        rcall MovsBuf
        ldi serial,$0d
        rcall trans

        rcall receive

        pop temp
        out sreg,temp
        pop temp
        reti

; Підпрограма передачі байта в буфер UART_OUT з регістру serial
MovsBuf:
        push zl                 ; Зберігання вмісту регістрів в стек
        push zh
        push temp
```

```
        lds zl,UART_OUT_T      ; Зчитати покажчик хвоста черги
        lds temp,UART_OUT_H; Зчитати покажчик голови черги
        cp zl,temp             ; Порівняти адреси голови та хвоста черги
        brne m1          ; Якщо покажчики на голову та хвіст черги рівні
        sbic ucr,udrie                  ; та UDRIE = "1"
        rjmp endm        ; то буфер переповнений, вихід з підпрограми

m1:     clr zh           ; Додати до покажчика хвоста черги
        subi zl,-UART_OUT      ; адрес початку буфера, тобто обчислити
                               ; фактичний адрес
        st z,serial            ; і записати в нього дані з регістру serial

        lds zl,UART_OUT_T      ; Зчитати покажчик хвоста черги
        inc zl                 ; Збільшити його на 1
        andi zl,$3f            ; Якщо величина покажчика більше 15, то
                               ; повернутися до початку буфера (так як
; в буфері 16 байт)
        sts UART_OUT_T,zl      ; Зберегти нове значення покажчика
        sbi ucr , udrie        ; Дозволити переривання UDRE

endm: pop temp                 ; Відновити зі стеку значення регістрів
        pop zh
        pop zl
        ret

; Підпрограма обробки переривань від UDRE
TransInt:                      ; Зберегти вміст регістрів в стек
        push zl
        push zh
        push temp
        in temp,sreg
        push temp

        lds zl,UART_OUT_H      ; Зчитати покажчик голови черги
        clr zh                 ; Обчислити абсолютний адрес
        subi zl , -UART_OUT    ; першого в черзі байта
        ld temp , z            ; Зчитати з буферу цей байт
```

```
    out UDR , temp                  ; Вивести його в UART

    lds zl,UART_OUT_H               ; Перевірка: буфер пустий?
    inc zl
    andi zl , $3f
    lds zh , UART_OUT_T
    cp zl , zh
    brne t1                         ; Якщо ще ні, перейти на t1

    cbi UCR,UDRIE   ; Якщо буфер пустий, заборонити подальші
  ; переривання по флагу UDRE (передавач
  ; пустий)

  t1:       sts UART_OUT_H , zl   ; Зберегти нове значення покажчика
                                  ; голови черги
    pop temp
    out sreg , temp
    pop temp                      ; Відновити зі стеку значення регістрів
    pop zh
    pop zl
    reti
```

Підпрограми "receive", "port_init", "tim0_init", "uart_init" наведені в лабораторних роботах №6 та №11.

Контрольні питання:

1. Передача даних.
2. Прийом даних.

Додаток 1

Система команд мікроконтролеру ATmega16.

Арифметичні та логічні інструкції

Мнемоника	Операнди	Опис	Операція	Цикли
ADD	Rd, Rr	Складання без переносу	Rd = Rd + Rr	1
ADC	Rd, Rr	Складання з переносом	Rd = Rd + Rr + C	1
SUB	Rd, Rr	Віднімання без переносу	Rd = Rd – Rr	1
SUBI	Rd, K8	Віднімання константи	Rd = Rd – K8	1
SBC	Rd, Rr	Віднімання з переносом	Rd = Rd – Rr – C	1
SBCI	Rd, K8	Віднімання константи з переносом	Rd = Rd – K8 – C	1
AND	Rd, Rr	Логічне І	Rd = Rd Λ Rr	1
ANDI	Rd, K8	Логічне І з константою	Rd = Rd Λ K8	1
OR	Rd, Rr	Логічне АБО	Rd = Rd V Rr	1
ORI	Rd, K8	Логічне АБО з константою	Rd = Rd V K8	1
EOR	Rd,Rr	Логічне виключне АБО	Rd = Rd EOR Rr	1
COM	Rd	Побітна інверсія	Rd = $FF – Rd	1
NEG	Rd	Зміна знаку (додатковий код)	Rd = $00 – Rd	1
SBR	Rd,K8	Встановити біт (біти) в регістрі	Rd = Rd V K8	1
CBR	Rd,K8	Скинути біт (біти) в регістрі	Rd = Rd Λ ($FF – K8)	1
INC	Rd	Інкрементувати значення регістру	Rd = Rd + 1	1
DEC	Rd	Декрементувати значення регістру	Rd = Rd – 1	1
TST	Rd	Перевірка на нуль або від’ємність	Rd = Rd Λ Rd	1

Мнемоника	Операнди	Опис	Операція	Цикли
CLR	Rd	Очистити регістр	Rd = 0	1
SER	Rd	Встановити регістр	Rd = $FF	1
ADIW	Rdl, K6	Скласти константу та слово	Rdh:Rdl = Rdh:Rdl + K6	2
SBIW	Rdl, K6	Відняти константу від слова	Rdh:Rdl = Rdh:Rdl – K6	2
MUL	Rd,Rr	Множення чисел без знаку	R1:R0 = Rd * Rr	2
MULS	Rd,Rr	Множення чисел зі знаком	R1:R0 = Rd * Rr	2
MULSU	Rd,Rr	Множення числа зі знаком з числом без знаку	R1:R0 = Rd * Rr	2
FMUL	Rd,Rr	Множення дробних чисел без знаку	R1:R0 = (Rd * Rr) << 1	2
FMULS	Rd,Rr	Множення дробних чисел зі знаком	R1:R0 = (Rd *Rr) << 1	2
FMULSU	Rd,Rr	Множення дробного числа зі знаком з числом без знаку	R1:R0 = (Rd * Rr) << 1	2

Команди розгалуження:

Мнемоника	Операнди	Опис	Операція	Цикли
RJMP	k	Відносний перехід	PC = PC + k +1	2
IJMP	Немає	Непрямий перехід на (**Z**)	PC = Z	2
EIJMP	Немає	Розширений непрямий перехід на (**Z**)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	2
JMP	k	Перехід	PC = k	3
RCALL	k	Відносний виклик підпрограми	STACK = PC+1, PC = PC + k + 1	3/4
ICALL	Немає	Непрямий виклик (**Z**)	STACK = PC+1, PC = Z	3/4
EICALL	Немає	Розширений нерямий виклик (**Z**)	STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND	4

Мнемоника	Операнди	Опис	Операція	Цикли
CALL	k	Виклик підпрограми	STACK = PC+2, PC = k	4/5
RET	Немає	Повернення з підпрограми	PC = STACK	4/5
RETI	Немає	Повернення з переривання	PC = STACK	4/5
CPSE	Rd,Rr	Порівняти, порівняти якщо дорівнюють	if (Rd ==Rr) PC = PC 2 or 3	1/2/3
CP	Rd,Rr	Порівняти	Rd –Rr	1
CPC	Rd,Rr	Порівняти з переносом	Rd – Rr – C	1
CPI	Rd,K8	Порівняти з константою	Rd – K	1
SBRC	Rr,b	Пропустити якщо біт в регістрі очищений	if(Rr(b)==0) PC = PC + 2 or 3	1/2/3
SBRS	Rr,b	Пропустити якщо біт в регістрі встановлений	if(Rr(b)==1) PC = PC + 2 or 3	1/2/3
SBIC	P,b	Пропустити якщо біт порту очищений	if(I/O(P,b)==0) PC = PC + 2 or 3	1/2/3
SBIS	P,b	Пропустити якщо біт порту встановлений	if(I/O(P,b)==1) PC = PC + 2 or 3	1/2/3
BRBC	s,k	Перейти якщо прапор в SREG очищений	if(SREG(s)==0) PC = PC + k + 1	1/2
BRBS	s,k	Перейти якщо прапор в SREG встановлений	if(SREG(s)==1) PC = PC + k + 1	1/2
BREQ	k	Перейти якщо дорівнює	if(Z==1) PC = PC + k + 1	1/2
BRNE	k	Перейти якщо не дорівнює	if(Z==0) PC = PC + k + 1	1/2
BRCS	k	Перейти якщо перенос встановлений	if(C==1) PC = PC + k + 1	1/2
BRCC	k	Перейти якщо перенос очищений	if(C==0) PC = PC + k + 1	1/2
BRSH	k	Перейти якщо дорівнює або більше	if(C==0) PC = PC + k + 1	1/2
BRLO	k	Перейти якщо менше	if(C==1) PC = PC + k + 1	1/2
BRMI	k	Перейти якщо мінус	if(N==1) PC = PC + k + 1	1/2

Мнемоника	Операнди	Опис	Операція	Цикли
BRPL	k	Перейти якщо плюс	if(N==0) PC = PC + k + 1	1/2
BRGE	k	Перейти якщо більше або дорівнює (зі знаком)	if(S==0) PC = PC + k + 1	1/2
BRLT	k	Перейти якщо менше (зі знаком)	if(S==1) PC = PC + k + 1	1/2
BRHS	k	Перейти якщо прапор внутрішнього переносу встановлений	if(H==1) PC = PC + k + 1	1/2
BRHC	k	Перейти якщо прапор внутрішнього переносу очищений	if(H==0) PC = PC + k + 1	1/2
BRTS	k	Перейти якщо прапор T встановлений	if(T==1) PC = PC + k + 1	1/2
BRTC	k	Перейти якщо прапор T очищений	if(T==0) PC = PC + k + 1	1/2
BRVS	k	Перейти якщо прапор переповнення встановлений	if(V==1) PC = PC + k + 1	1/2
BRVC	k	Перейти якщо прапор переповнення очищений	if(V==0) PC = PC + k + 1	1/2
BRIE	k	Перейти якщо переривання дозволені	if(I==1) PC = PC + k + 1	1/2
BRID	k	Перейти якщо переривання заборонені	if(I==0) PC = PC + k + 1	1/2

Інструкції передачі даних:

Мнемоника	Операнди	Опис	Операція	Цикли
MOV	Rd,Rr	Скопіювати регістр	Rd = Rr	1
MOVW	Rd,Rr	Скопіювати пару регістрів	Rd+1:Rd = Rr+1:Rr, r,d even	1
LDI	Rd,K8	Завантажити константу	Rd = K	1
LDS	Rd,k	Пряме завантаження	Rd = (k)	2
LD	Rd,X	Непряме завантаження	Rd = (X)	2
LD	Rd,X+	Непряме завантаження з постінкрементом	Rd = (X), X=X+1	2

Мнемоника	Операнди	Опис	Операція	Цикли
LD	Rd,–X	Непряме завантаження з предекрементом	X=X – 1, Rd = (X)	2
LD	Rd,Y	Непряме завантаження	Rd = (Y)	2
LD	Rd, Y+	Непряме завантаження з постінкрементом	Rd = (Y), Y=Y+1	2
LD	Rd, –Y	Непряме завантаження з предекрементом	Y=Y – 1, Rd = (Y)	2
LDD	Rd, Y+q	Непряме завантаження з заміщенням	Rd = (Y+q)	2
LD	Rd,Z	Непряме завантаження	Rd = (Z)	2
LD	Rd, Z+	Непряме завантаження з постінкрементом	Rd = (Z), Z=Z+1	2
LD	Rd, –Z	Непряме завантаження з предекрементом	Z=Z – 1, Rd = (Z)	2
LDD	Rd,Z+q	Непряме завантаження з заміщенням	Rd = (Z+q)	2
STS	k, Rr	Пряме збереження	(k) = Rr	2
ST	X, Rr	Непряме збереження	(X) = Rr	2
ST	X+, Rr	Непряме збереження з постінкрементом	(X) = Rr, X=X+1	2
ST	–X, Rr	Непряме збереження з предекрементом	X=X – 1, (X)=Rr	2
ST	Y, Rr	Непряме збереження	(Y) = Rr	2
ST	Y+, Rr	Непряме збереження з постінкрементом	(Y) = Rr, Y=Y+1	2
ST	–Y, Rr	Непряме збереження з предекрементом	Y=Y – 1, (Y) = Rr	2
ST	Y+q, Rr	Непряме збереження з заміщенням	(Y+q) = Rr	2
ST	Z, Rr	Непряме збереження	(Z) = Rr	2
ST	Z+, Rr	Непряме збереження з постінкрементом	(Z) = Rr, Z=Z+1	2
ST	–Z, Rr	Непряме збереження з предекрементом	Z=Z – 1, (Z) = Rr	2
ST	Z+q, Rr	Непряме збереження із заміщенням	(Z+q) = Rr	2
LPM	Нет	Завантаження з програмної пам’яті	R0 = (**Z**)	3

Мнемоника	Операнди	Опис	Операція	Цикли
LPM	Rd, Z	Завантаження з програмної пам'яті	Rd = (**Z**)	3
LPM	Rd, Z+	Завантаження з програмної пам'яті з постінкрементом	Rd = (**Z**), Z=Z+1	3
ELPM	Нет	Розширене завантаження із програмної пам'яті	R0 = (RAMPZ:**Z**)	3
ELPM	Rd, Z	Розширене завантаження із програмної пам'яті	Rd = (RAMPZ:**Z**)	3
ELPM	Rd, Z+	Розширене завантаження із програмної пам'яті з постінкрементом	Rd = (RAMPZ:**Z**), Z = Z+1	3
SPM	Нет	Збереження в програмній пам'яті	(**Z**) = R1:R0	–
ESPM	Нет	Розширене збереження в програмній пам'яті	(RAMPZ:**Z**) = R1:R0	–
IN	Rd, P	Читання порту	Rd = P	1
OUT	P, Rr	Запис у порт	P = Rr	1
PUSH	Rr	Занесення регістру в стек	STACK = Rr	2
POP	Rd	Вилучення регістра із стеку	Rd = STACK	2

Інструкції роботи з бітами

Мнемоника	Операнди	Опис	Операція	Цикли
LSL	Rd	Логічний зсув ліворуч	Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)	1
LSR	Rd	Логічний зсув праворуч	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	1
ROL	Rd	Циклічний зсув ліворуч через C	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	1
ROR	Rd	Циклічний зсув праворуч через C	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	1

Мнемоника	Операнди	Опис	Операція	Цикли
ASR	Rd	Арифметичний зсув праворуч	Rd(n)=Rd(n+1), n=0,...,6	1
SWAP	Rd	Перестановлення тетрад	Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0)	1
BSET	s	Встановлення прапору	SREG(s) = 1	1
BCLR	s	Очищення прапору	SREG(s) = 0	1
SBI	P, b	Встановити біт порту	I/O(P,b) = 1	2
CBI	P, b	Очистити біт порту	I/O(P,b) = 0	2
BST	Rr, b	Зберегти біт з регистру в T	T = Rr(b)	1
BLD	Rd, b	Загрузити біт із T в регістр	Rd(b) = T	1
SEC	Немає	Встановити прапор переносу	C =1	1
CLC	Немає	Очистити прапор переносу	C = 0	1
SEN	Немає	Встановити прапор від'ємного числа	N = 1	1
CLN	Немає	Очистити прапор від'ємного числа	N = 0	1
SEZ	Немає	Встановити прапор нуля	Z = 1	1
CLZ	Немає	Очистити прапор нуля	Z = 0	1
SEI	Немає	Встановити прапор переривань	I = 1	1
CLI	Немає	Очистити прапор переривань	I = 0	1
SES	Немає	Встановити прапор числа зі знаком	S = 1	1
CLN	Немає	Очистити прапор числа зі знаком	S = 0	1
SEV	Немає	Встановити прапор переповнення	V = 1	1
CLV	Немає	Очистити прапор переповнення	V = 0	1
SET	Немає	Встановити прапор T	T = 1	1
CLT	Немає	Очистити прапор T	T = 0	1
SEH	Немає	Встановити прапор внутрішнього переносу	H = 1	1

Мнемоника	Операнди	Опис	Операція	Цикли
CLH	Немає	Очистити прапор внутрішнього переносу	H = 0	1
NOP	Немає	Нема операції		1
SLEEP	Немає	Спати (знизити енергоспоживання)		1
WDR	Немає	Скидання сторожового таймеру		1

Використані позначення:

Rd: Результуючий (та вихідний) регістр в регістровому файлі

Rr: Вихідний регістр в регістровому файлі

b: Константа (3 біта)

s: Константа (3 біта)

P: Константа (5-6 біт)

K6: Константа (6 біт)

K8: Константа (8 біт)

k: Константа (розмір залежить від інструкції)

q: Константа (6 біт)

Rdl: R24, R26, R28, R30. Для інструкцій ADIW та SBIW

X,Y,Z: Регістри непрямої адресації (X=R27:R26, Y=R29:R28, Z=R31:R30)

Додаток 2

Директиви асемблеру

Компілятор підтримує ряд директив. Директиви не транслюються безпосередньо в код. Вони використовуються для ініціалізації пам'яті, визначення макросів, підключення файлів і т.і. Перелік директив наводиться в наступній таблиці:

Директива	Опис
BYTE	Зарезервувати байти в ОЗП
CSEG	Програмний сегмент
DB	Визначити байти у Flash або EEPROM
DEF	Призначити регістру символічне ім’я
DEVICE	Визначити пристрій для якого компілюється програма
DSEG	Сегмент даних
DW	Визначити слова у Flash або EEPROM
ENDM, ENDMACRO	Кінець макросу
EQU	Встановити постійний вираз
ESEG	Сегмент EEPROM
EXIT	Вийти з файлу
INCLUDE	Вложити інший файл
LIST	Включити генерацію лістингу
LISTMAC	Включити розверчення макросів в лістингу
MACRO	Початок макросу
NOLIST	Вимкнути генерацію лістингу
ORG	Встановити положення в сегменті
SET	Встановити змінний символічний еквівалент виразу

Додаток 3

Робота з РКІ-дисплеєм

Операції запису для 8-ми розрядної шини

1. Встановити значення лінії RS
2. Вивести значення байта даних на лінії шини DB0 ... DB7
3. Встановити лінію E = 1
4. Встановити лінію E = 0
5. Встановити лінії шини DB0...DB7 = 1

Операції читання для 8-ми розрядної шини

1. Встановити значення лінії RS
2. Встановити лінію R/W = 1
3. Встановити лінію E = 1
4. Зчитати значення байта даних з лінії шини DB0 ... DB7
5. Встановити лінію E = 0
6. Встановити лінію R/W = 0

Операції запису для 4-ох розрядної шини

1. Встановити значення лінії RS
2. Вивести значення старшої тетради байта даних на лінії шини DB4...DB7
3. Встановити лінію E = 1
4. Встановити лінію E = 0
5. Вивести значення молодшої тетради байта даних на лінії шини DB4...DB7
6. Встановити лінію E = 1
7. Встановити лінію E = 0
8. Встановити лінії шини DB0...DB7 = 1

Операції читання для 4-ох розрядної шини

1. Встановити значення лінії RS
2. Встановити лінію R / W = 1
3. Встановити лінію E = 1
4. Зчитати значення старшої тетради байта даних на лінії шини DB4...DB7
5. Встановити лінію E = 0
6. Встановити лінію E = 1
7. Зчитати значення молодшої тетради байта даних на лінії шини DB4...DB7
8. Встановити лінію E = 0
9. Встановити лінію R/W = 0

Наведені операції повинні здійснюватися таким чином, щоб час виконання кожного кроку становив не менше 250 нс.

Додаток 4

Прапори, що керують роботою контролера HD44780

I/D:	режим зміщення лічильника адреси AC, 0 – зменшення, 1 – збільшення.
S:	прапор режиму зсуву змісту екрана. 0 – зсув екрану не проводиться, 1 – після запису в DDRAM коду екран зрушується в напрямку, який визначається прапором I/D: 0 – вправо, 1 – вліво. При зсуві не проводиться зміна змісту DDRAM. Змінюються тільки внутрішні покажчики розташування видимого початку рядка в DDRAM.
S/C:	прапор-команда, яка виробляє разом з прапором R/L операцію зсуву змісту екрана (так само, як і в попередньому випадку, без змін до DDRAM) або курсору. Визначає об'єкт зміщення: 0 – зсувається курсор, 1 – зсувається екран.
R/L:	прапор-команда, яка здійснює разом з прапором S/C операцію зсуву екрану або курсору. Уточнює напрям зсуву: 0 – вліво, 1 – вправо.
D/L:	прапор, який визначає ширину шини даних: 0 – 4 розряду, 1 – 8 розрядів.
N:	режим розгортки зображення на РКІ: 0 – один рядок, 1 – два рядки.
F:	розмір матриці символів: 0 – 5×8 точок, 1 – 5×10 точок.
D:	наявність зображення: 0 – вимкнено, 1 – включено.
C:	курсор у вигляді підкреслення: 0 – вимкнено, 1 – увімкнено.
B:	курсор у вигляді мерехтливого знакомісця: 0 – вимкнено, 1 – увімкнено.

Значення керуючих прапорів після подачі живлення

I/D=1:	режим збільшення лічильника на 1.
S=0:	без зсуву зображення.
D/L=1:	8-ми розрядна шина даних.
N=0:	режим одного рядка.
F=0:	символи з матрицею 5×8 точок.
D=0:	відображення вимкнено.
C=0:	курсор у вигляді підкреслення вимкнений.
B=0:	курсор у вигляді мерехтливого знакомісця вимкнений.

Керуючі комбінації бітів регістра IR

D7	D6	D5	D4	D3	D2	D1	D0	Призначення
0	0	0	0	0	0	0	1	Очищення екрану, AC = 0, адресація AC на DDRAM
0	0	0	0	0	0	1	–	AC = 0, адресація на DDRAM, скинуті зсуви, початок рядка адресується на початок DDRAM
0	0	0	0	0	1	I/D	S	Вибір напрямку зсуву курсору або екрана
0	0	0	0	1	D	C	B	Вибір режиму відображення
0	0	0	1	S/C	R/L	–	–	Команда зсуву курсору/екрану
0	0	1	DL	N	F	–	–	Визначення параметрів розгортки і ширини шини даних
0	1	AG	AG	AG	AG	AG	AG	Присвоєння лічильнику AC адреси в області CGRAM
1	AD	AD	AD	AD	AD	AD	AD	Присвоєння лічильнику AC адреси в області DDRAM

Додаток 5

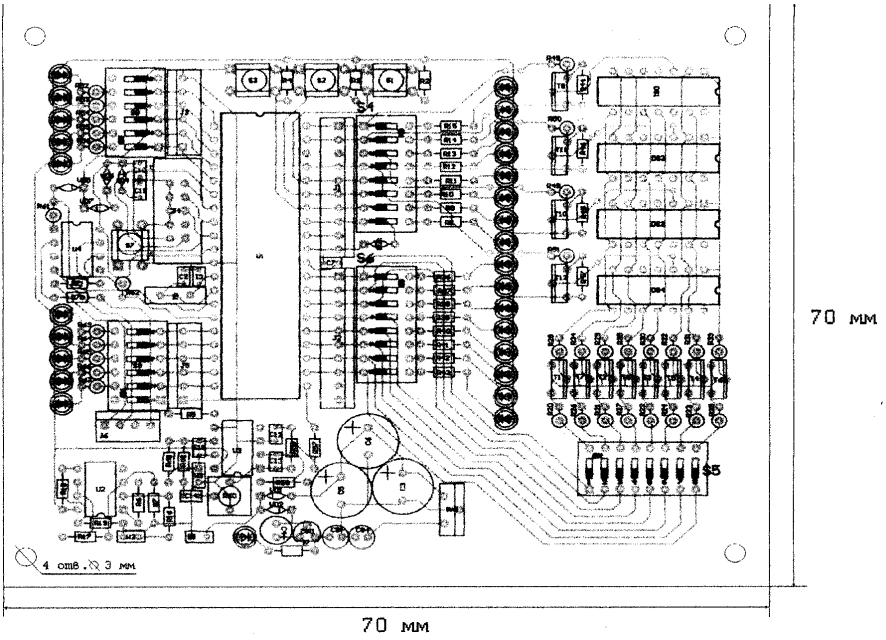
Кодові комбінації символів контролеру HD44780

Старші 4 біта / Молодші 4 біта	0	1	2	3	4	5	6	7
0	CGRAM (1)			0	@	P	`	p
1	CGRAM (2)		!	1	A	Q	a	q
2	CGRAM (3)		?	2	B	R	b	r
3	CGRAM (4)		#	3	C	S	c	s
4	CGRAM (5)		$	4	D	T	d	t
5	CGRAM (6)		%	5	E	U	e	u
6	CGRAM (7)		&	6	F	V	f	v
7	CGRAM (8)		‘	7	G	W	g	w
8	CGRAM (1)		(	8	H	X	h	x
9	CGRAM (2)		)	9	I	Y	i	y
A	CGRAM (3)		*	:	J	Z	j	z
B	CGRAM (4)		+	;	K	[	k	{
C	CGRAM (5)		,	<	L	?	l	|
D	CGRAM (6)		–	=	M	]	m	}
E	CGRAM (7)		.	>	N	^	n	?
F	CGRAM (8)		/	?	O	_	o	?

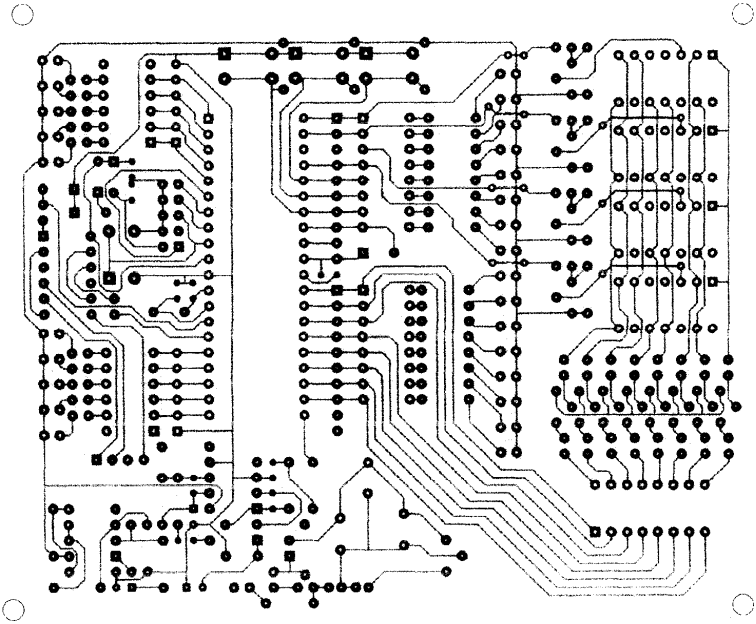
Додаток 6

Друкована плата відладочного модуля ATmega16 Evaluation Kit 2006

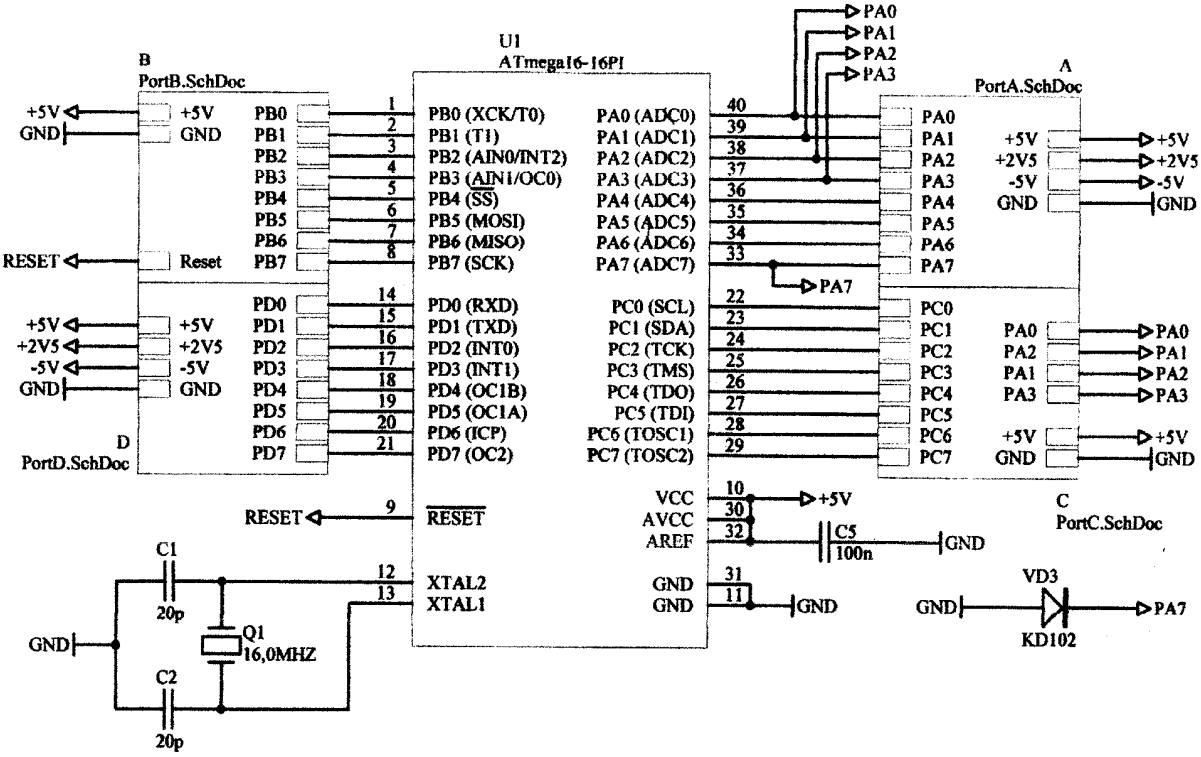
Вид з боку деталей

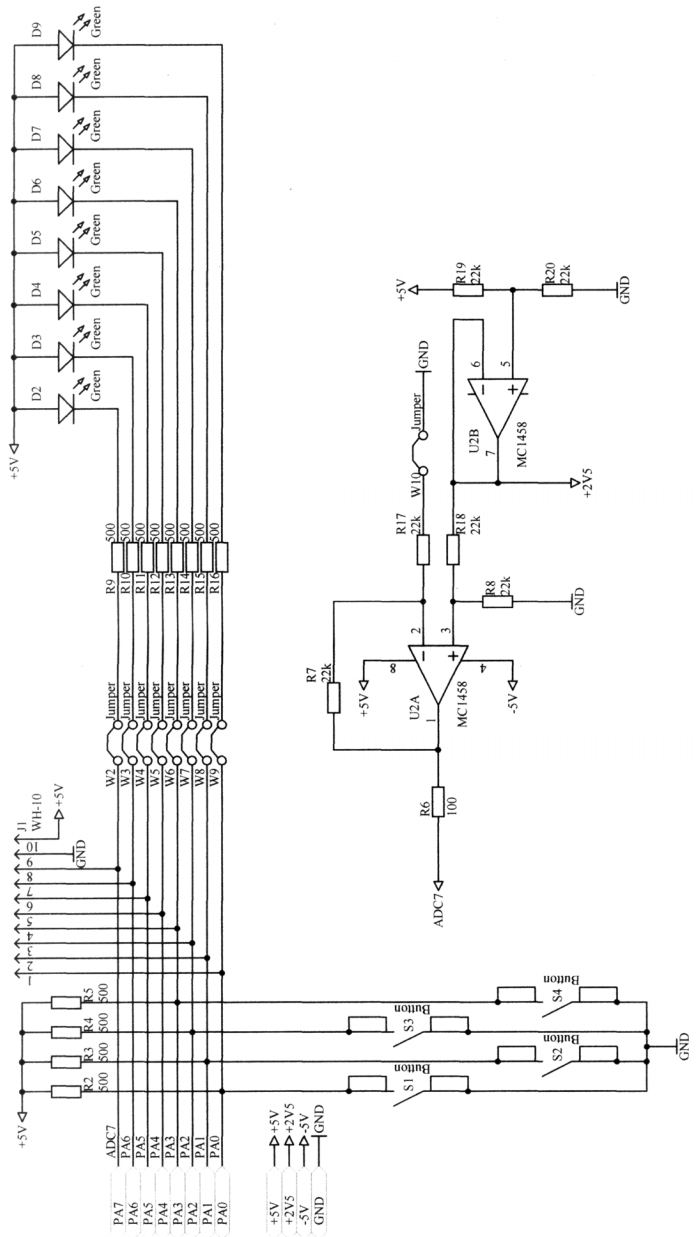


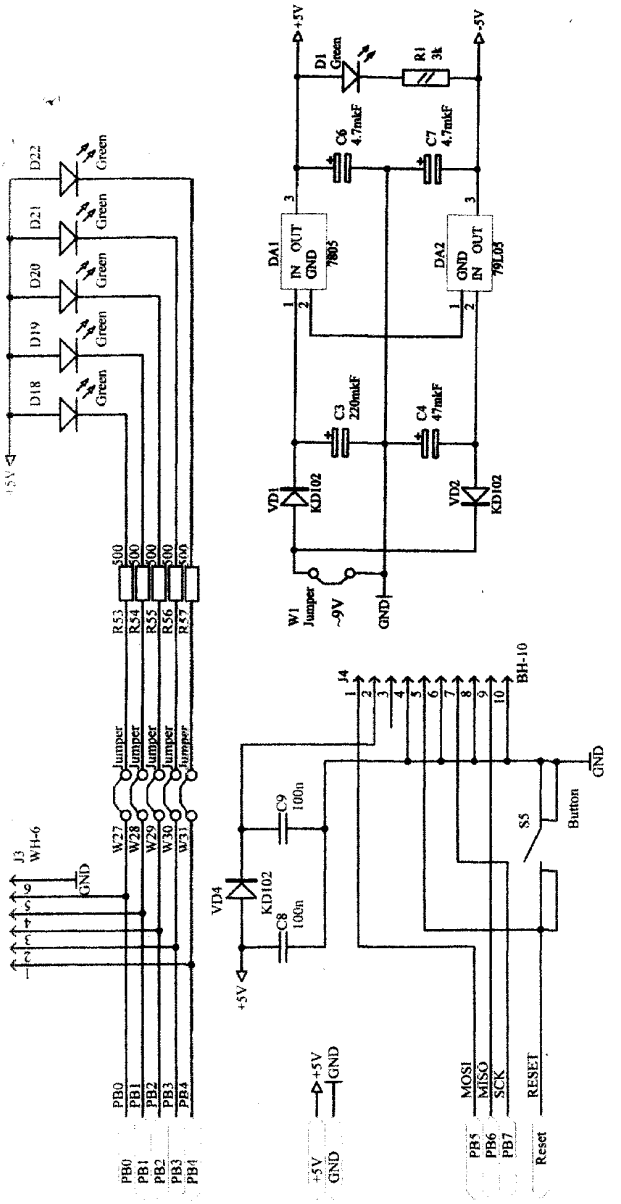
Вид з боку пайки

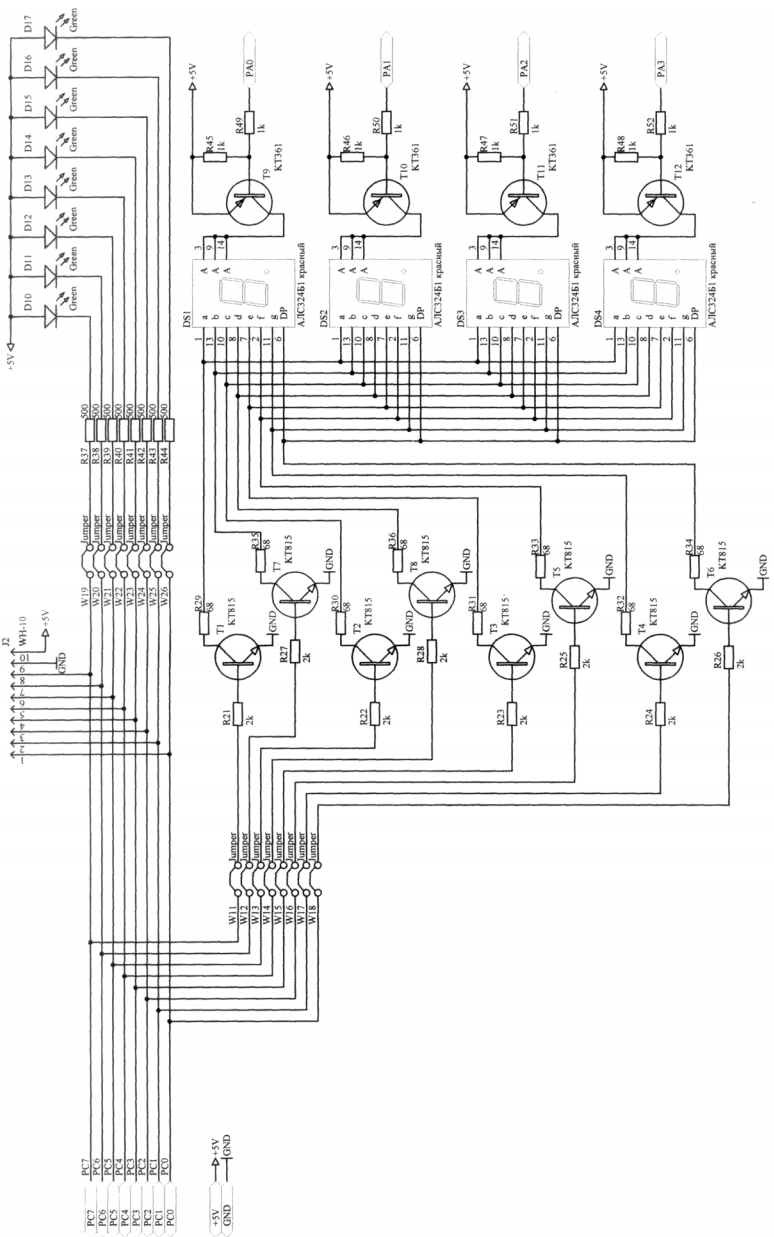


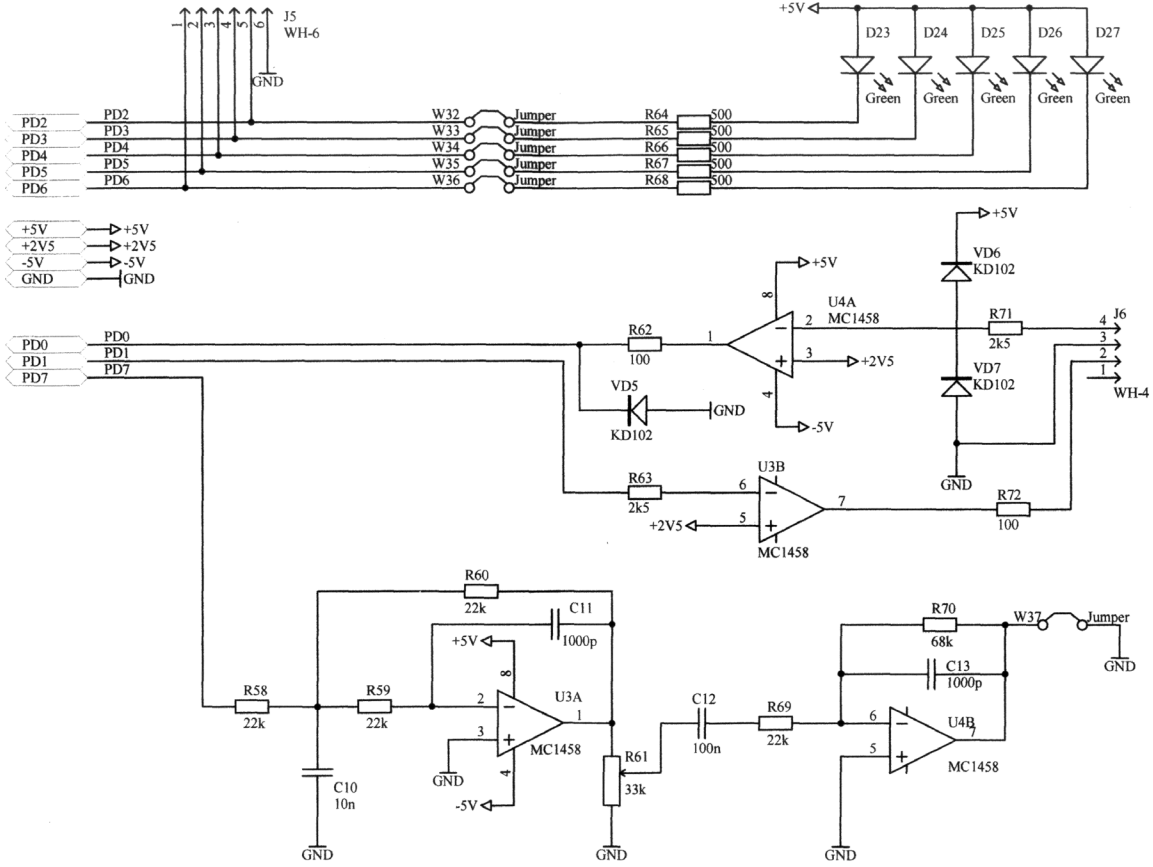
Електрична принципова схема відладочного модуля ATmega16 Evaluation Kit 2006




D2 D3 D4 D5 D6 D7 D8 D9
Green
+5V
R2 R3 R4 R5 R9 R10 R11 R12 R13 R14 R15 R16
500
ADC7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
J1 WH-10
W2 W3 W4 W5 W6 W7 W8 W9
Jumper
GND
S1 S2 S3 S4
Button
+5V +2V5 -5V GND
R6 100
R7 22k
R8 R17 R18 R19 R20 22k
W10
U2A U2B MC1458
ADC7
-5V
+2V5


D18
D19
D20
D21
D22
Green
+5V
R53
R54
R55
R56
R57
500
Jumper
W27
W28
W29
W30
W31
J3
WH-6
GND
PB0
PB1
PB2
PB3
PB4
VD4
KD102
C8
100n
C9
100n
+5V
GND
J4
BH-10
S5
Button
MOSI
MISO
SCK
RESET
PB5
PB6
PB7
Reset
DA1
7805
DA2
79L05
IN
OUT
C3
220mkF
C4
47mkF
C6
4.7mkF
C7
4.7mkF
D1
R1
3k
VD1
VD2
KD102
W1
Jumper
-9V


D10 D11 D12 D13 D14 D15 D16 D17
Green
+5V
R37 R38 R39 R40 R41 R42 R43 R44
500
Jumper
W19 W20 W21 W22 W23 W24 W25 W26
J2
WH-10
GND
W11 W12 W13 W14 W15 W16 W17 W18
PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0
DS1 DS2 DS3 DS4
АЛС324Б1 красный
R45 R46 R47 R48
R49 R50 R51 R52
1k
KT361
T9 T10 T11 T12
PA0 PA1 PA2 PA3
R29 R30 R31 R32
R35 R36 R33 R34
68
KT815
T7 T8 T5 T6
T1 T2 T3 T4
R27 R28 R25 R26
R21 R22 R23 R24
2k


J5
WH-6
GND
+5V
D23
D24
D25
D26
D27
Green
PD2
PD3
PD4
PD5
PD6
W32
W33
W34
W35
W36
Jumper
R64
R65
R66
R67
R68
500
+5V
+2V5
-5V
GND
PD0
PD1
PD7
R62
100
U4A
MC1458
VD6
KD102
VD7
KD102
VD5
KD102
R71
2k5
J6
WH-4
R63
2k5
U3B
MC1458
R72
100
R60
22k
C11
1000p
R58
22k
R59
22k
U3A
MC1458
C10
10n
R61
33k
C12
100n
R69
22k
R70
68k
C13
1000p
W37
U4B
MC1458

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. **Гребнев, В. В.** Микроконтроллеры семейства AVR фирмы Atmel [Текст] / В. В. Гребнев. – М. : ИП Радио Софт, 2002.

2. **Голубцов, М. С.** Микроконтроллеры AVR: от простого к сложному [Текст] / М. С. Голубцов, А. В. Кириченкова. – Изд. 2-е, испр. и доп. – М. : СОЛОН-Пресс, 2004.

3. **Евстифеев, А. В.** Микроконтроллері AVR семейства Tiny и Mega фирмы "ATMEL" [Текст] / А. В. Евстифеев. – М. : Издательский дом "Додэка-XXI", 2004. – 560 с.

4. Мікропроцесорна техніка [Текст] : підручник / Ю. І. Якименко, Т. О. Терещенко, Є. І. Сокол, В. Я. Жуйков, Ю. С. Петергеря; За ред. Т. О. Терещенко. – 2-ге вид., переробл. та доповн. – К. : ІВЦ Видавництво "Політехніка"; "Кондор", 2004.

5. **Ярочкина, Г. В.** Радиоэлектронная аппаратура и приборы : монтаж и регулировка [Текст] : учебник для нач. проф. образования / Г. В. Ярочкина. – М. : ИРПО; ПрофОбрИздат, 2002.

6. Проектирование цифровых устройств на однокристальных микроконтроллерах [Текст] / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева. – М. : Энергоатомиздат, 1990.

7. Сайт фірми ATMEL, www.atmel.com.

ЗМІСТ