З М І С Т

В С Т У П

Розглянемо лінію 8-розрядних високопродуктивних RISC мікроконтролерів загального призначення виробництва Atmel Corp., об'єднаних загальною маркою AVR. Серійне виробництво AVR почалося в 1996 році, а в даний час у серійному виробництві в Atmel знаходяться три сімейства AVR - "tiny", "classic" і "mega".

Здавалося б, що ще нового можна було придумати в цій області і для чого? Адже до початку 1990-х широко розповсюджене сімейство мікроконтролерів MCS51, що випускаються багатьма фірмами-виробниками (Intel, Philips, Temic, OKI, Siemens і ін.), уже було де-факто промисловим стандартом для 8-розрядних систем і прекрасно підходило для використання в широкому класі задач, особливо якщо вибиралися кристали з додатковими убудованими периферійними пристроями і підвищеною тактовою частотою. Звичайно, був і зворотний бік медалі - значне питоме енергоспоживання цих мікроконтролерів.

Остаточний вибір розробником тієї чи іншої мікропроцесорної платформи (microprocessor platform) для реалізації своєї задачі залежить, природно, від великого числа різноманітних факторів, включаючи економічні. Але звичайно першорядною умовою залишається одержання максимально вигідного співвідношення "ціна - продуктивність - енергоспоживання", обумовленого складністю розв'язуваної задачі. Видимо, ця обставина і послужила поштовхом до розробки в середині 1990-х нового 8-розрядного мікроконтролера.

AVR, мабуть, один із самих цікавих напрямків, що розвиваються корпорацією Atmel. Вони являють собою могутній інструмент для створення сучасних високопродуктивних і економічних багатоцільових контролерів (bagatocilevikh controller). На даний час співвідношення "ціна - продуктивність - енергоспоживання" для AVR є одним із кращих на світовому ринку 8-розрядних мікроконтролерів. Обсяги продаж AVR у світі подвоюються щорічно. У геометричній прогресії росте число сторонніх фірм, що розробляють і випускають різноманітні програмні й апаратні засоби підтримки розробок для них. Можна вважати, що AVR поступово стає ще одним індустріальним стандартом серед 8-розрядних мікроконтролерів загального призначення.

Галузі застосування AVR мікроконтролерів багатогранні. Для сімейства "tiny" - це інтелектуальні автомобільні датчики різного призначення, іграшки, ігрові приставки, материнські плати персональних комп'ютерів, контролери захисту доступу в мобільних телефонах, зарядні пристрої, детектори диму і полум'я, побутова техніка, різноманітні інфрачервоні пульти дистанційного керування. Для сімейства "classic" - це модеми різних типів, сучасні зарядні пристрої, вироби класу Smart Cards і

пристрою читання для них, супутникові навігаційні системи для визначення місця розташування автомобілів на трасі, складна побутова техніка, пульти дистанційного керування, мережні карти, материнські плати комп'ютерів, стільникові телефони нового покоління а також різні і різноманітні промислові системи контролю і керування. Для "mega" AVR - це аналогові (NMT, ETACS, AMPS) і цифрові (GSM, CDMA) мобільні телефони, принтери і ключові контролери для них, контролери апаратів факсимільного зв'язку і ксероксів, контролери сучасних дискових нагромаджувачів, CD-ROM і т.д.

AVR - це молодий продукт корпорації Atmel. У цій лінії мікроконтролерів загального призначення постійно з'являються нові кристали, обновляються версії вже існуючих мікросхем, удосконалюється і розширюється програмне забезпечення підтримки. Так, перше офіційне видання - каталог Atmel, присвячений AVR - датований травнем 1997 року. У нього були включені всього чотири перших AVR - мікроконтролери сімейства AT90S "classic". Друге, істотно розширене видання каталогу вийшло в серпні 1999 року, і в нього вже були включені три сімейства AVR - "tiny", "classic" і "mega". І дотепер більш "свіжої" версії каталогу в друкованому виді не існує, постійно обновляються лише технічні дані в електронному виді (Data Sheet), що Atmel Corp. розміщає на своїй інформаційній сторінці в Інтернеті http://www.atmel.com/.

Зважаючи на те, що даний тип мікроконтролерів на сьогоднішній день широко використовуються в побутових приладах та вимірювальній техніці, необхідно впровадження і вивчення їх в програмі вищої школи, як в теоретичному, так і в практичному аспектах, особливо при підготовці бакалаврів і магістрів з метрології та вимірювальної техніки.

Перед виконанням лабораторних робіт потрібно ознайомитись з матеріалом, викладеним в даному лабораторному практикумі та в рекомендованій до відповідної роботи літературі. До виконання лабораторних робіт допускаються студенти, які виявили необхідну ступінь теоретичної підготовки. Захист проводиться на основі звітів до лабораторних робіт в обсязі, зазначеному в лабораторному практикумі та оформлених у відповідності з вимогами ДСТУ. Звіт з лабораторної роботи готується один на бригаду.

Цикл лабораторних робіт вибирається викладачем з урахуванням виділеного часу студентам на виконання лабораторного практикуму.

Лабораторна робота №1

АРХИТЕКТУРА МІКРОКОНТРОЛЕРА *AT90S2313*. ОРГАНІЗАЦІЯ ПАМ'ЯТІ. КОМАНДИ ПЕРЕСИЛАННЯ ДАНИХ

Мета роботи: вивчення структури мікроконтролера AT90S2313, організацію пам'яті та форматів його команд, а також придбання початкових навичків програмування в кодах мікроконтролера з використанням команд передачі даних.

Теоретичні відомості

1. Архітектура мікроконтролера

Мікроконтролери сімейства AVR мають єдину базову структуру. Узагальнена структурна схема мікроконтролера (МК) зображена на рис.1.1.

До складу мікроконтролера входять:

- генератор тактового сигналу (GCK);
- процесор (CPU);
- постійний запам'ятовуючий пристрій для збереження програми виконаний за технологією Flash, (FlashROM);
- оперативний запам'ятовуючий пристрій статичного типу для збереження даних (SRAM);
- постійний запам'ятовуючий пристрій для збереження даних, виконаний за технологією EEPROM, (EEPROM);
- набір периферійних пристроїв для вводу/виводу даних і керуючих сигналів, і виконання інших функцій.

До складу процесора (CPU) входять:

- лічильник команд (PC);
- арифметико-логічний пристрій (ALU);
- блок регістрів загального призначення (GPR, General Purpose Registers) і інші елементи.

Усі AVR мають також блок енергонезалежної пам'яті даних EEPROM, що електрично стирається. Цей тип пам'яті, доступний програмі мікроконтролера безпосередньо в ході її виконання, зручний для збереження проміжних даних, різних констант, таблиць перекодувань, каліброваних коефіцієнтів і т.п. EEPROM також може бути завантажена ззовні як через SPI інтерфейс, так і за допомогою звичайного программатора. Число циклів перезапису - не менш 100000. Два програмувальних біти таємності дозволяють захистити пам'ять програм і енергонезалежну пам'ять даних EEPROM від несанкціонованого зчитування. Внутрішня оперативна пам'ять SRAM мається у всіх AVR сімейств "classic" і "mega" і в одного нового кристала сімейства "tiny" - ATtiny26/L. Для деяких мікроконтролерів можлива організація

підключення зовнішньої пам'яті даних обсягом до 64Kb.

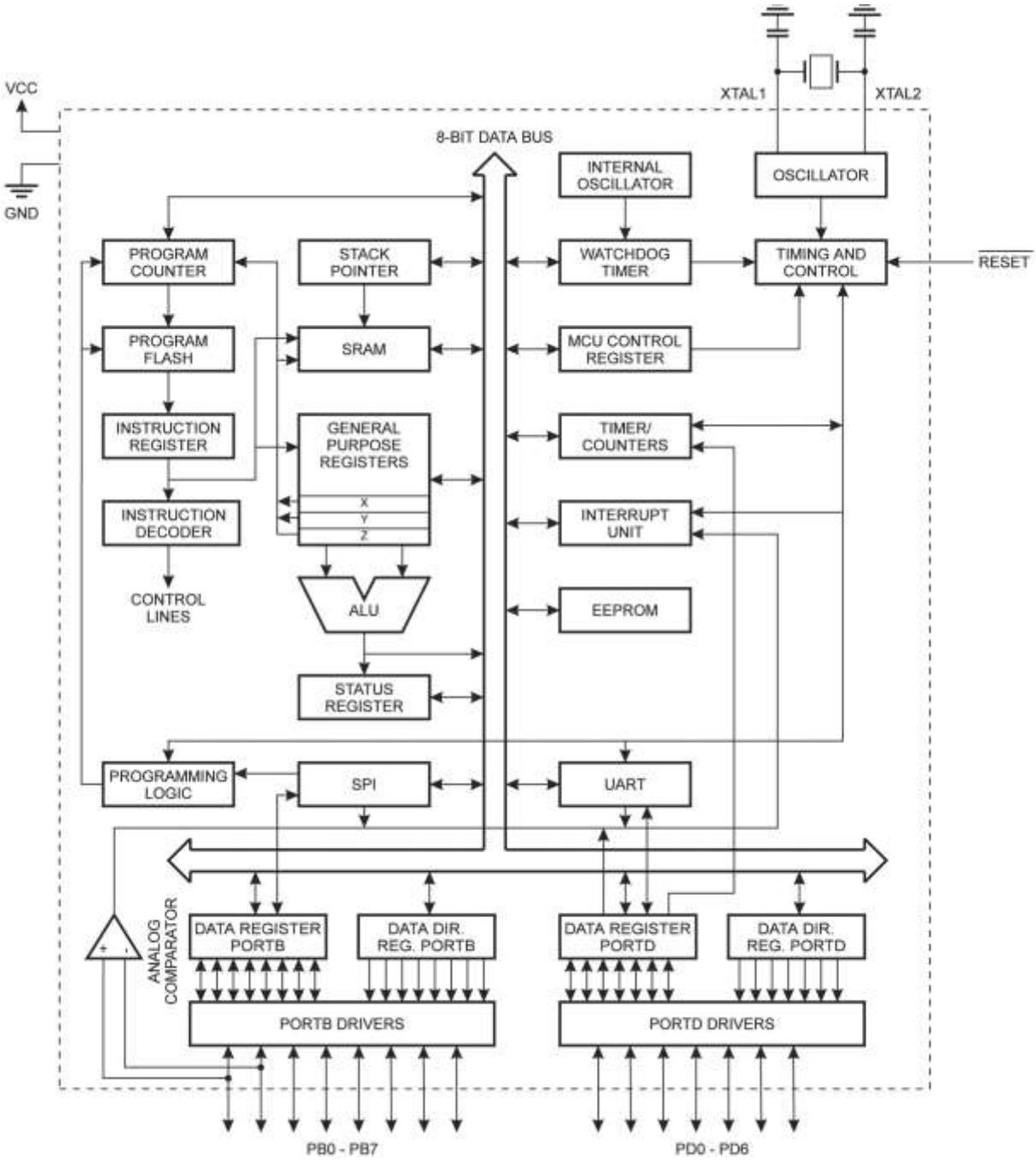


Рисунок 1.1 – Блок схема AVR мікроконтролерів

Структурна схема мікроконтролера AT90S2313 приведена на рис.1.2.

Крім регістрів загального призначення (general-purpose registers) в мікроконтролері маються регістри спеціальних функцій, що у сімействі AVR називаються регістрами вводу/виводу (I/O Registers,IOR). За участю цих регістрів здійснюються:

- керування роботою мікроконтролера і окремих його пристроїв;
- визначення стану мікроконтролера і окремих його пристроїв;

- ввід даних у мікроконтролер і окремі його пристрої і вивід даних і виконуються інші функції.

Для нумерації регістрів вводу/виводу використовуються номери від 0 до 63 (від $00 до S3F, де $ - показник шістнадцятирічного коду). Кожному регістру присвоєне ім'я, зв'язане з функцією, яку виконує цей регістр. Мікроконтролери різних типів мають різний склад регістрів вводу/виводу, при цьому регістри з однаковими номерами можуть мати різні імена.

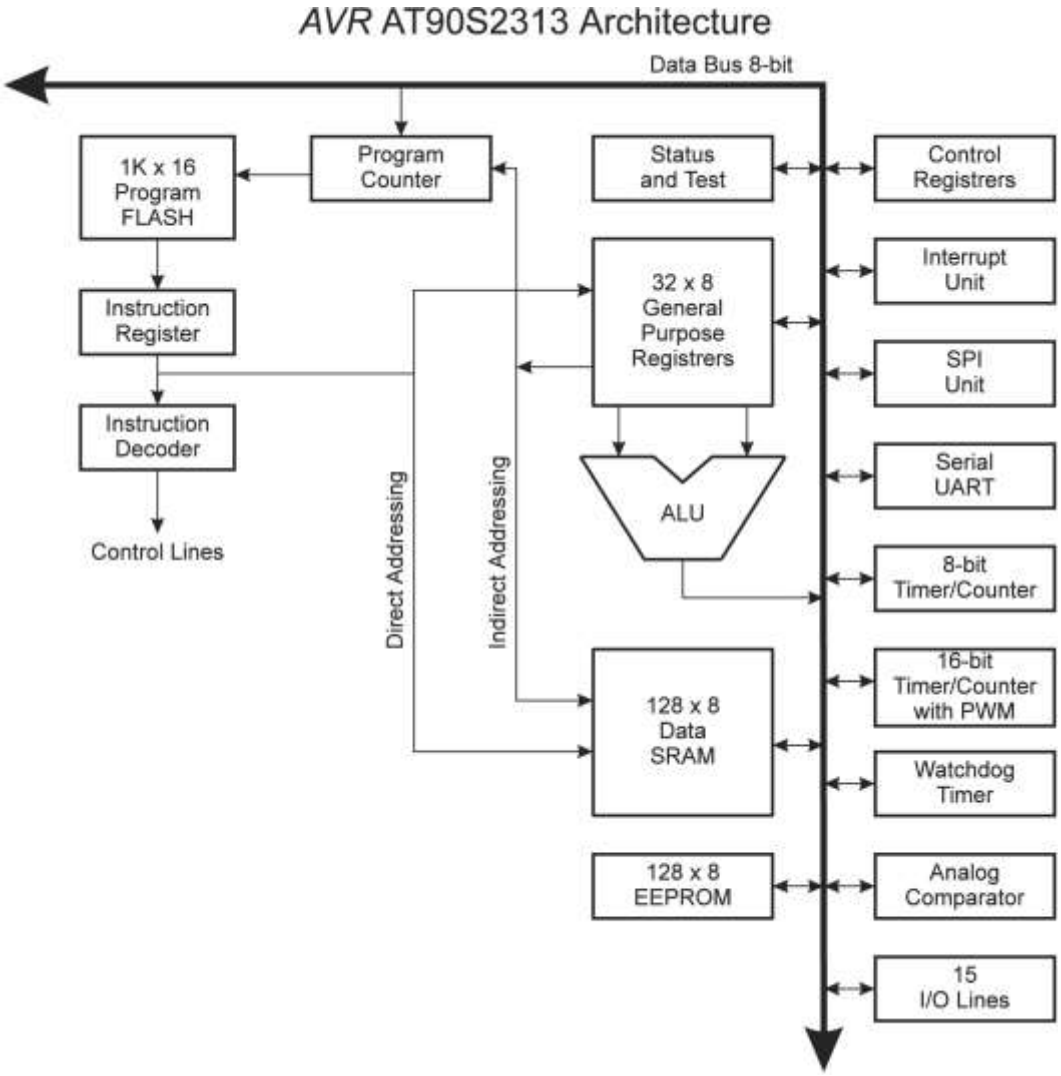


Рисунок 1.2 – Архітектура мікроконтролера 2313

1.1. Генератор тактового сигналу

Мікроконтролери сімейства AVR є пристроями синхронного типу. Дії, які виконуються в мікроконтролері, прив'язані до імпульсів тактового сигналу.

Як генератор тактового сигналу (GCK) використовуються:

- внутрішній генератор із зовнішнім кварцовим чи керамічним резонатором (XTAL);
- внутрішній RC-генератор (IRC);

- внутрішній генератор із зовнішнім RC-колом (ERC);
- зовнішній генератор (EXT).

У мікроконтролерів, що мають внутрішній генератор із зовнішнім резонатором (XTAL), резонатор підключається до виводів XTAL1 н XTAL2, що через конденсатори малої ємності (20-30 пф) з'єднуються із шиною GND. Тактова частота визначається робочою частотою резонатора. XTAL1 і XTAL2 є входом і виходом, відповідно, який інвертує підсилювач, що з використанням кварцового чи кристалокерамічного резонатора працює як вбудований генератор, як показано на рис. 1.3. При використанні зовнішнього джерела тактової частоти висновок XTAL2 повинний залишитися вільним, сигнал подається на висновок XTAL1, як показано на рис. 1.4.

Кварцовий кристал генератора таймера приєднується безпосередньо до виводів OSC1 і OSC2. Зовнішні конденсатори не вимагаються. Генератор оптимизований під часовий кварц із частотою 32,768 кГц. Зовнішній тактовий сигнал, який подається на ці виводи, надходить до підсилювача зі смугою пропускання 256 кГц. Частота зовнішнього сигналу повинна знаходитися в діапазоні від 0 до 256 Кгц.

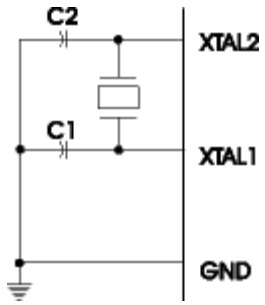


Рисунок 1.3 - Приєднання тактовог генератора

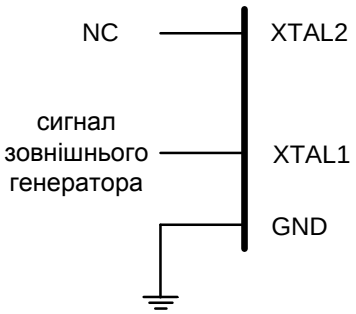


Рисунок 1.4 - Приєднання зовнішнього джерела тактового сигналу

1.2. CPU – процесор. ALU - Арифметико-логічний пристрій

Процесор (CPU) формує адреса чергової команди, вибирає команду з пам'яті й організовує її виконання. Код команди має формат "слово" (16 біт) чи "два слова". Система команд мікроконтролерів сімейства AVR розглядається в розділі 2.

До складу процесора крім лічильника команд (PC), арифметико-логічного пристрою (ALU) і блоку регістрів загального призначення (GPR),зображених на структурній схемі рис.1, входять:

- регістр стану мікроконтролера SREG;
- регістр-показник стека SP чи SPL і SPH.

Високопродуктивне AVR ALU з'єднано безпосередньо з усіма 32 швидкодіючими регістрами загального призначення. За один тактовий цикл ALU виконує операцію між регістрами цього реєстрового файлу.

Операції ALU підрозділяються на три основні категорії: арифметичної, логічні й операції над бітами.

1.3. Запам'ятовуючий пристрій SRAM. Файл регістрів загального призначення

Оперативний запам'ятовуючий пристрій статичного типу SRAM призначений для зберігання даних, одержуваних у процесі роботи мікроконтролера. При вимиканні напруги живлення мікроконтролера дані в SRAM губляться. Комірка пам'яті містить 8 розрядів.

Адреса байта при звертанні до SRAM може бути зазначена у коді команди зі звертанням до SRAM (пряма адресація) чи попередньо записана в пари регістрів X, Y чи Z (непряма адресація). Звертання до SRAM може виконуватись з використанням адреси, що зберігається в регістрі-показнику стека.

Байт для запису в SRAM надходить з регістра загального призначення. Байт, зчтаний з SRAM, надходить у регістр загального призначення.

В адресний простір SRAM крім адрес, по яких виконується звертання до комірок пам'яті SRAM, включені 32 адреси для звертання до регістрів загального призначення (адреси від $00 до $1F) і 64 адреси для звертання до регістрів вводу/виводу (адреси від $20 до $5F). На рис. 1.5 представлена структура 32 регістрів загального призначення.

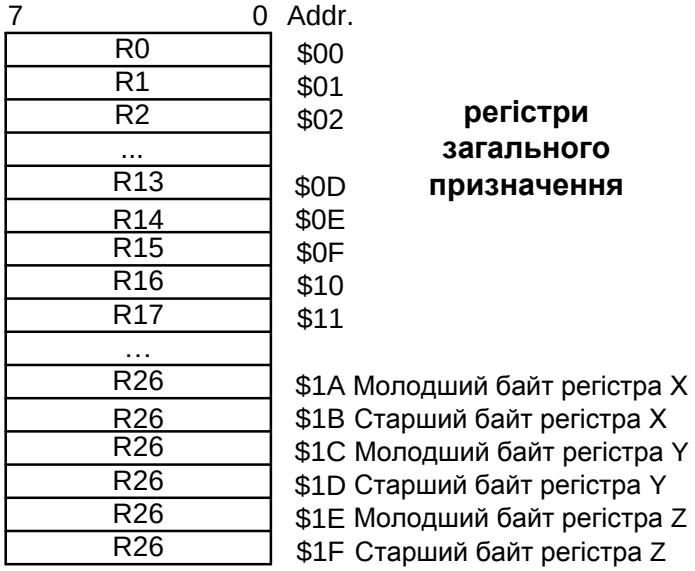


Рисунок 1.5 – Регістри загального призначення CPU мікроконтролерів AVR

Усі реєстрові команди звертаються безпосередньо до регістрів протягом одного тактового циклу. Виключенням є п'ять логічних і арифметичних операцій з константами (SBCI, SUBI, CPI і ANDI) і операція ORI між константою і вмістом регістра, і команда безпосереднього

завантаження константи LDI. Ці команди використовують другу половину регістрів реєстрового файлу - R16..R31.

Самі загальні команди SBC, SUB, CP, AND і OR і всі інші операції між двома регістрами чи регістрами з одним регістром використовують для запису результату в реєстровий файл. Як показано на рис. 1.5, кожному регістру відповідає адреса пам'яті даних, що відображає їхній у перших 32 осередках користувальницького простору даних. Хоча вони не використовуються як фізичні осередки SRAM, така організація пам'яті забезпечує гнучке звертання до регістрів, оскільки X,Y і Z регістри можуть бути використані для індексації будь-якого регістра у файлі.

SRAM даних 2313 має обсяг 128x8 байт і займає адресний простір від $60 до $DF. Шість регістрів (з R26 по R31) реєстрового файлу, крім звичайної для інших регістрів функцій, виконують функцію 16-розрядних регістрів покажчиків адреси при непрямій адресації SRAM. Ці три регістри непрямої адресації визначаються як регістри X,Y і Z (рис. 1.6).

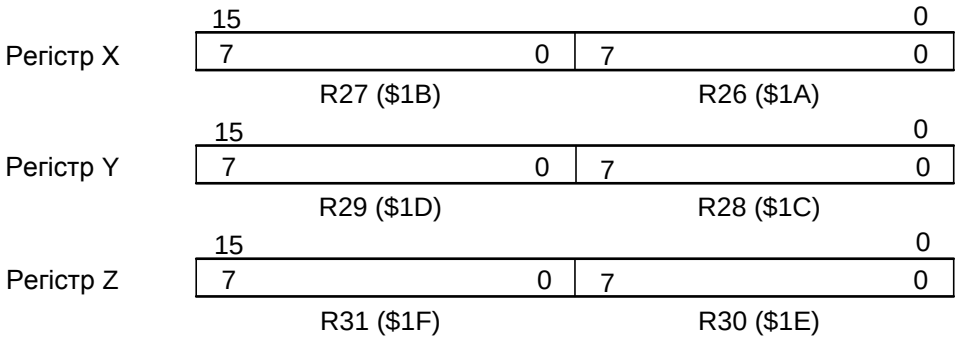


Рисунок 1.6 – Регістри X, Y і Z.

У різних режимах адресації ці регістри виконують функції фіксованого зсуву, автоматичного інкремента і декремента.

1.4. Запам'ятовуючий пристрій FlashROM

Постійний запам'ятовуючий пристрій FlashROM призначений для збереження кодів команд програми і констант. Комірка пам'яті містить 16 розрядів. У ній можуть зберігатися код команди формату "слово", половина коду команди формату "два слова" або коди двох констант.

При читанні кодів команд адреса в FlashROM надходить з лічильника команд. При читанні констант адреса надходить з пари Z регістрів загального призначення.

Запис кодів у FlashROM виконується в процесі програмування побайтно. У мікроконтролерах з великим числом виводів (20 і більше) байт може вводитися чи паралельно послідовно. У мікроконтролерах з малим числом виводів (8) байт вводиться послідовно.

1.5. Запам'ятовуючий пристрій EEPROM

Постійний запам'ятовуючий пристрій EEPROM призначений для

збереження даних, записаних при програмуванні мікроконтролера й одержуваних у процесі виконання програми. При вимиканні напруги живлення дані зберігаються. Комірка пам'яті містить 8 розрядів.

EEPROM має відособлений адресний простір. При звертанні до EEPROM адреса записується в регістр адреси EEAR (№ $1E). Байт, призначений для запису, заноситься в регістр даних EEDR (№ $1D). Байт, одержуваний при читанні, надходить у цей самий регістр. Для керування процедурами запису і читання використовується регістр керування EECR (№ $1C).

Для запису байта в EEPROM необхідно:

1) записати адреса в регістр адреси;

2) записати байт у регістр даних;

3) установити в одиничний стан розряд EEMWE регістра EECR,

4) при EEMWE = 1 встановити в одиничний стан розряд EEWE регістра EECR.

Процедура запису виконується в залежності від величини напруги живлення за 2,5-4 мс. При завершенні запису розряд EEWE регістра EECR апаратно скидається в нульовий стан.

Розряд EEMWE зберігає одиничний стан протягом 4-х тактів після установки і апаратно скидається в нульовий стан.

Для читання байта з EEPROM необхідно:

1) записати адреса в регістр адреси;

2) установити в одиничний стан розряд EERE регістра EECR.

Прочитаний байт надходить у регістр даних. Розряд EERE регістра EECR апаратно скидається в нульовий стан.

1.6. Конфігурація пам'яті AVR 2313

Пам'ять програм має обсяг 1до X 16, займає адресний простір $000-$3FF. Пам'ять даних складається з 32 регістрів загального призначення ($00-$1F), 64 регістрів введення/виводу ($20-$5F) і оперативної пам'яті 128 x 8 ($60-$DF). При адресації пам'яті даних використовуються п'ять режимів адресації: безпосередня адресація, непряма зі зсувом, непряма, непряма з преддекрементом і непряма з постдекрементом. Регістри з R26 по R31 реєстрового файлу працюють як X, Y і Z регістри показники непрямої адресації.

Непрямій адресації зі зсувом доступні 63 адреси щодо базових адрес, що знаходяться в регістрах Y чи Z. При використанні непрямої адресації з автоматичним преддекрементом і постдекрементом автоматично декрементуются і інкрементуются адреси записані в регістри

X, Y і Z. Усіма цими режимами перекривається весь адресний простір даних, включаючи 32 регістра загального призначення і 64 регістра I/O.

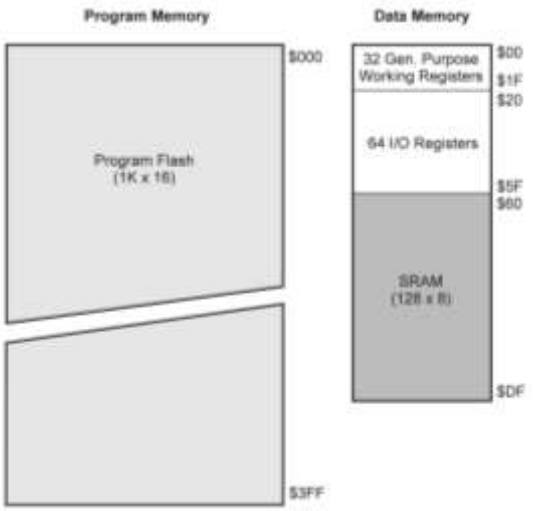


Рисунок 1.7 – Карта пам'яті

1.7. Периферійні пристрої

У групу периферійних пристроїв входять:

- паралельні порти вводу/виводу;
- послідовний порт SPI;
- послідовний порт UART;
- послідовний порт TWSI (I2C);
- таймери-лічильники загального призначення (timers-meters of the general setting);
- сторожовий таймер (watch timer);
- аналого-цифровий перетворювач (analog-digital converter);
- аналоговий компаратор (analog comparatorsorter);
- програмуємий апаратний модулятор (programuemiy vehicle keyer);
- блок переривань (block of breaking).

Паралельний порт вводу/виводу (Port, P) призначений для вводу і виводу даних. Мікроконтролери сімейства AVR мають від одного до шести портів. Порт може мати від трьох до восьми виводів. Вивід порту може працювати в режимі входу чи в режимі виходу. Напрямок передачі біта встановлюється для кожного виводу окремо.

Деякі виводи портів крім введення і виводу бітів даних можуть використовуватися для виконання альтернативних функцій при роботі інших пристроїв.

Послідовний порт вводу/виводу SPI *(Senal Peripheral Interface)* призначений для вводу і виводу байтів при обміні даними з іншими

пристроями, що мають порт SPI. Обмін виконується під керуванням тактового сигналу порту. Пристрій, ініціалізуючи обмін і виробляючи тактовий сигнал, є ведучим (master). Пристрій, що виконує обмін при надходженні тактового сигналу, є відомим (slave). У процесі обміну обидва пристрої послідовно біт за бітом одночасно видають і приймають байт. Обмін виконується з використанням трьох шин.

Максимальна швидкість прийому/передачі (у бітах за секунду) дорівнює 1/4 частоти тактового сигналу мікроконтролера (у МК типу ml63 - 1/2 тактової частоти).

До одного ведучого пристрою можуть бути підключені декілька відомих. Функції ведучого і відомого можуть мінятися в процесі роботи системи.

Послідовний порт вводу/виводу UART *(Universal Asynchronous Receiver-Transmitter)* призначений для передачі і прийому байтів даних по двухпровідних лініях зв'язку (наприклад, по інтерфейсі RS-232C чи "струмова петля"). Прийом і передача можуть виконуватись одночасно. При передачі байта формується послідовність з десяти чи одинадцяти бітів (кадр), що містить стартовий біт, що має нульове значення, вісім бітів байта (DO, D1, , D7) і столовий біт, що має одиничне значення. Між старшим бітом байта (D7) і стоповим бітом може міститися додатковий біт.

Послідовний порт вводу/виводу TWSI *(Two-Wire Senal Inter-face)* призначений для обміну байтами даних з іншими пристроями по двухпровідній шині I^2C (Integrated Circuit). До шини можуть під’єднуватись до 127 пристроїв.

Пристрій, підключений до шини I^2C, може працювати в якості ведучого (master) чи відомого (slave) Ведуче пристрій при звільненні шини від обміну між іншими пристроями посилає в шину сигнал початку обміну і займає шину, потім посилає адресний байт для вибору одного з ведучих пристроїв і завдання напрямку обміну, чи передає приймає байти даних і посилає сигнал закінчення обміну.

Відомий пристрій, пізнавши свою адресу, у залежності від зазначеного напрямку обміну приймає чи передає дані.

Пристрій, що приймає байти даних, після прийому кожного байта посилає в шину сигнал підтвердження прийому. Адресний байт може містити загальні адреса для передачі даних одночасно в усі відомі пристрої.

Таймер-лічильник загального призначення *(General Purpose*

Timer/Counter) призначений для формування запиту переривання при витіканні заданого інтервалу часу (режим таймера) чи здійсненні заданого числа подій (режим лічильника). Мікроконтролери сімейства AVR можуть мати від одного до трьох таймерів-лічильників загального призначення T/CX (X - номер таймера-лічильника).

Основним елементом таймера-лічильника є базовий лічильник, що веде лічбу на додавання. При його переповненні формується запит переривання T/CX OVF.

Таймер-лічильник загального призначення може виконувати додаткові функції:

- функцію захоплення (function of fascination);
- функцію порівняння (function of comparison);
- функцію широтно-імпульсного модулятора (function of latitudinal quantizer);
- функцію рахунка реального часу (function of account of the real time).

Функція захвату *(capture)* полягає в запам'ятовуванні коду, сформованого в базовому лічильнику, у спеціальному регістрі захоплення при зміні значення визначеного зовнішним чи внутрішним сигналом. При цьому формується запит переривання T/CX CAPT.

Функція порівняння *(compare)* полягає в зміні значення сигналу на визначеному виході мікроконтролера при збігу коду, зформованого в базовому лічильнику, з кодом у спеціальному регістрі порівняння. При цьому формується запит переривання T/CX COMP.

Функція широтно-імпульсного модулятора (PWM) полягає у формуванні на визначеному виході мікроконтролера імпульсної послідовності з заданими періодом повторення і тривалістю імпульсів.

Функції порівняння і PWM реалізуються з використанням того самого устаткування. Вибір потрібної функції виконується програмними засобами.

Функція підрахунку реального часу *(Real Time Clock)* реалізується в таймері-лічильнику при використанні додаткового внутрішнього генератора з зовнішнім кварцовим резонатором з частотою 32768 Гц.

У залежності від розрядності лічильника і виконуваних додаткових функцій можуть бути виділені п'ять типів таймерів-лічильників загального призначення, що входять у групу периферійних пристроїв мікро-контролерів сімейства AVR.

Тип A. Восьмирозрядний таймер-лічильник без додаткових функцій.

Тип B. Восьмирозрядний таймер-лічильник з функцією порівняння/PWM.

Тип C. Восьмирозрядний таймер-лічильник з функцією порівняння/PWM і функцією рахунка реального часу.

Тип D. Шістнадцятирозрядний таймер-лічильник з функціями захоплення і порівняння/PWM.

Тип E. Шістнадцятирозрядний таймер-лічильник з функцією захоплення і двома каналами для виконання функцій порівняння/PWM.

Сторожовий таймер *(Watchdog Timer, WDT)* призначений для ліквідування наслідків збою в ході програми шляхом перезапуску мікроконтролера при виявленні збою. Сторожовий таймер мається в мікроконтролерів усіх типів.

Аналого-цифровий перетворювач *(Analog-to-Digital Converter)* формує десятирозрядний двійковий код числа, пропорційного величині напруги аналогового сигналу на вході мікроконтролера. У мікроконтролерах AVR до перетворювача можуть підключатися від чотирьох до восьми входів мікроконтролера.

Аналого-цифровий компаратор *(Analog Comparator,* AC) порівнює по величині аналогові сигнали, що надходять на два входи мікроконтролера, і (формує запит переривання ANA COMP, коли різниця їхніх значень змінює знак. При цьому також може бути виданий сигнал для виконання функції захоплення в таймері-лічильнику загального призначення.

Програмувальний апаратний модулятор *(Programmable Hardware Modulator,* PHM) призначений для формування імпульсного сигналу на виводі PA2 для живленя світлодіодних індикаторів. Тривалість імпульсу і шпаруватість сигналу задаються програмними засобами. Струм навантаження може мати величину до 25 мА при напрузі живлення 1,8 В.

Блок переривань *(Interrupt Unit,* IU) організує перехід до виконання програми, що перериває, при надходженні запиту переривання, якщо переривання по даному запиті дозволене і він має більш високий пріоритет, чим інші запити, що надійшли одночасно з ним.

Переривання дозволене, якщо розряд I регістра SREG знаходиться в одиничному стані й в одиничному стані знаходиться розряд, що дозволяє/забороняє переривання по даному запиту, розташований в одному з регістрів вводу/виводу. Пріоритетність запитів задана апаратно.

При переході до виконання програми, що переривається, розряд I регістра SREG скидається в нульовий стан і зберігається в цьому стані до завершення програми, що переривається. Розряд I може бути переведений в одиничний стан по команді в програмі, що переривається.

Запити в блок переривань надходять із зовнішніх джерел і джерел, розташованих у внутрішніх пристроях мікроконтролера.

2. Режими адресації пам'яті програм і даних

При звертанні до Flash пам'яті програм і пам'яті даних (SRAM, реєстровому файлу і пам'яті I/O) AVR Enhanced RISC мікроконтролерами використовуються потужні й ефективні режими адресації.

Безпосередня адресація, одиночний регістр Rd зображено на рис.1.8.

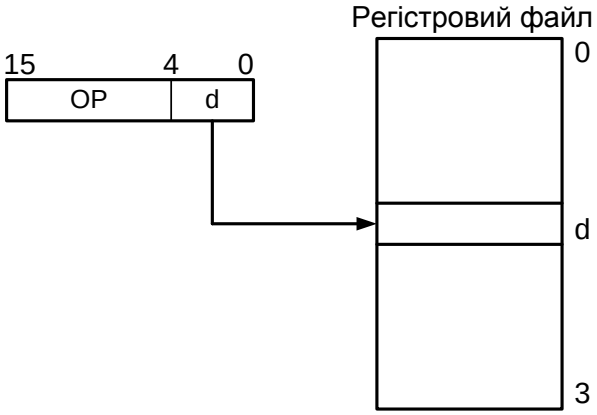


Рисунок 1.8 – Безпосередня адресація одного регістра

Операнд міститься в регістрі d (Rd).

Безпосередня адресація, два регістри Rd і Rr (рис. 1.9).

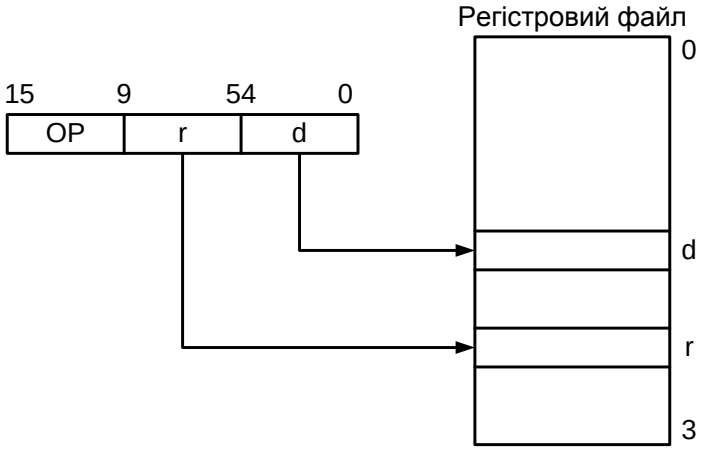


Рисунок 1.9 –Безпосередня регістрова адресація двох регістрів

Операнди містяться в регістрах r (Rr) і d (Rd). Результат зберігається в регістрі d (Rd).

Безпосередня адресація I/O (рис. 1.10).

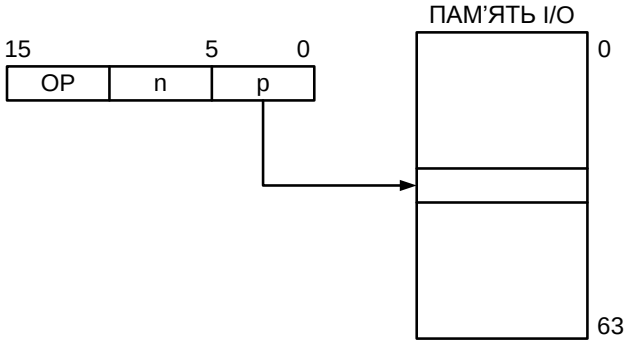


Рисунок 1.10 - Безпосередня адресація I/O

Адреса операнда міститься в 6 бітах слова команди. Величина n визначає адресу регістра чи джерела регістра призначення.

Безпосередня адресація даних представлена на рис. 1.11.

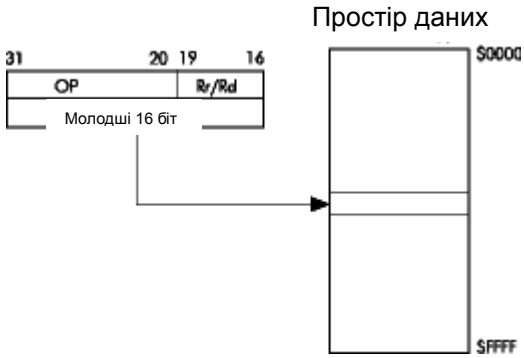


Рисунок 1.11 – Безпосередня адресація даних

16-розрядна адреса даних міститься в 16 молодших розрядах 32-розрядної команди. Rd/Rr визначають регістр чи джерело регістр призначення.

Непряма адресація даних зі зсувом (рис. 1.12).

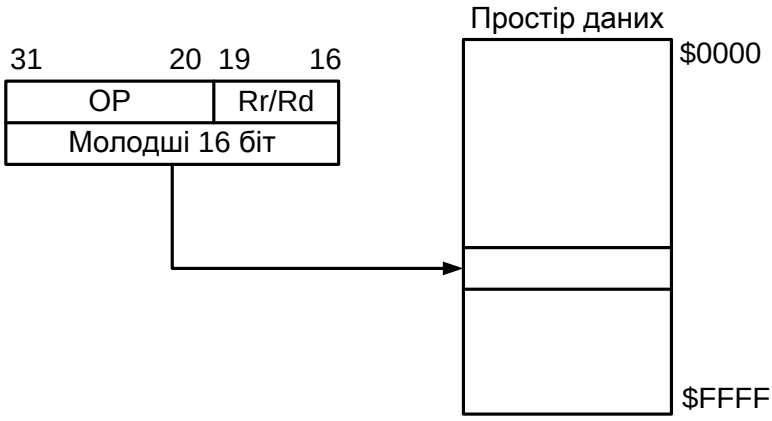


Рисунок 1.12 – Непряма адресація даних зі зсувом

Адреса операнда обчислюється підсумовуванням вмісту регістра Y чи Z з 6 бітами адреси, що містяться в слові команди.

Непряма адресація даних (рис. 1.13).

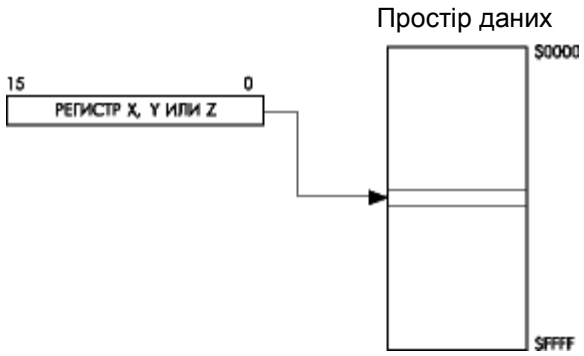


Рисунок 1.13 – Непряма адресація даних

Адреса операнда міститься в регістрі X, Y чи Z.

Непряма адресація даних із преддекрементом (рис. 1.14).

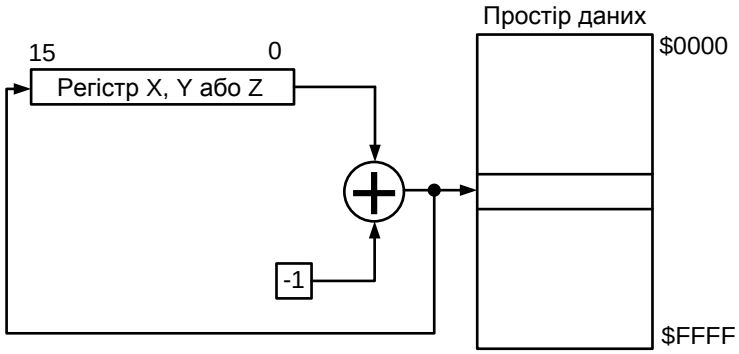


Рисунок 1.14 – Непряма адресація даних із преддекрементом

Перед виконанням операції регістр X, Y чи Z декрементується. Декрементований вміст регістра X, Y чи Z є адресою операнда.

Непряма адресація даних з постінкрементом (рис. 1.15).

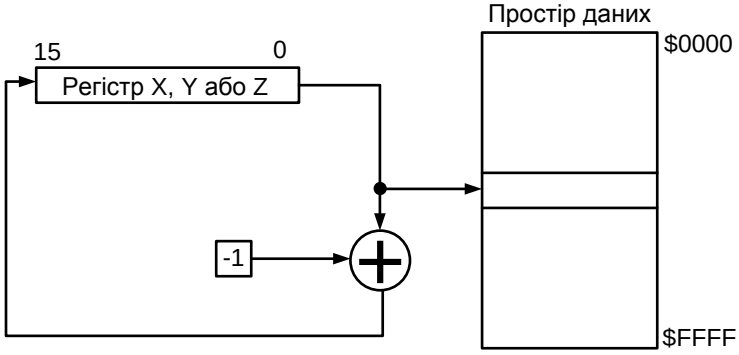


Рисунок 1.15 – Непряма адресація даних з постінкрементом

Після виконання операції регістр X, Y чи Z інкрементуться. Адресою операнда є вміст X, Y чи Z регістра попередньо інкрементовано.

Адресація константи з використанням команд LPM (рис. 1.16).

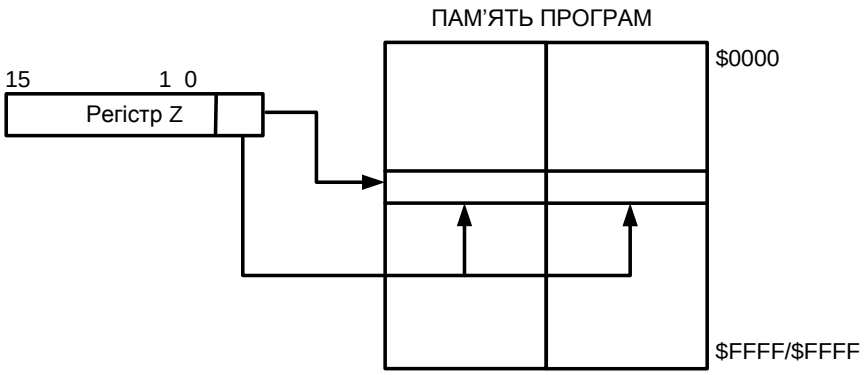


Рисунок 1.16 – Адресація константи коду пам'яті

Адреса байта константи визначається вмістом регістра Z. Старші 15 бітів визначають слово адреси (від 0 до 1К х 16).

Непряма адресація пам'яті програм, команди IJMP і ICALL (рис.1.17).

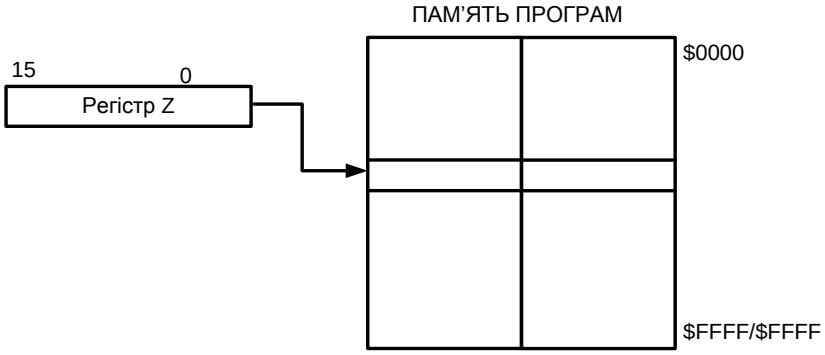


Рисунок 1.17 – Непряма адресація пам'яті програм

Виконання програми продовжується з адреси, що міститься в регістрі Z (тобто лічильник команд завантажується вмістом регістра Z).

Відносна адресація пам'яті програм, команди RJMP і RCALL (рис.1.18).

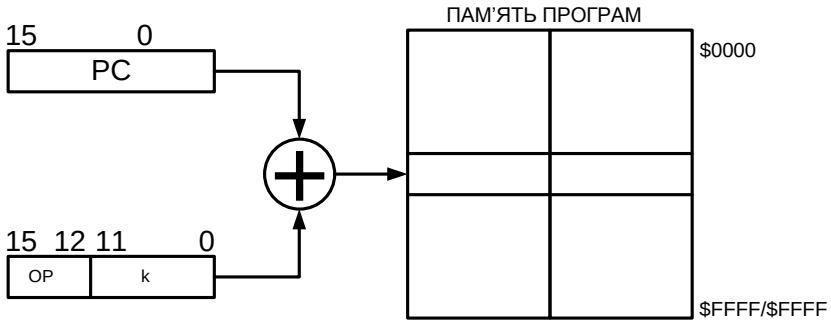


Рисунок 1.18 – Відносна адресація пам'яті програм

3. Команди передачі даних

На рис. 1.19 зображена програмна модель AVR-мікроконтролерів, що являє собою діаграму програмно доступних ресурсів AVR.

Центральним блоком на цій діаграмі є реєстровий файл на 32 оперативних регістра (R0-R31), безпосередньо доступних ALU. Старші регістри об'єднані парами й утворять три 16-розрядних регістри, призначених для непрямої адресації комірок пам'яті .

Всі арифметичні і логічні операції, а також частина операцій роботи з бітами виконуються в ALU тільки над вмістом оперативних регістрів.

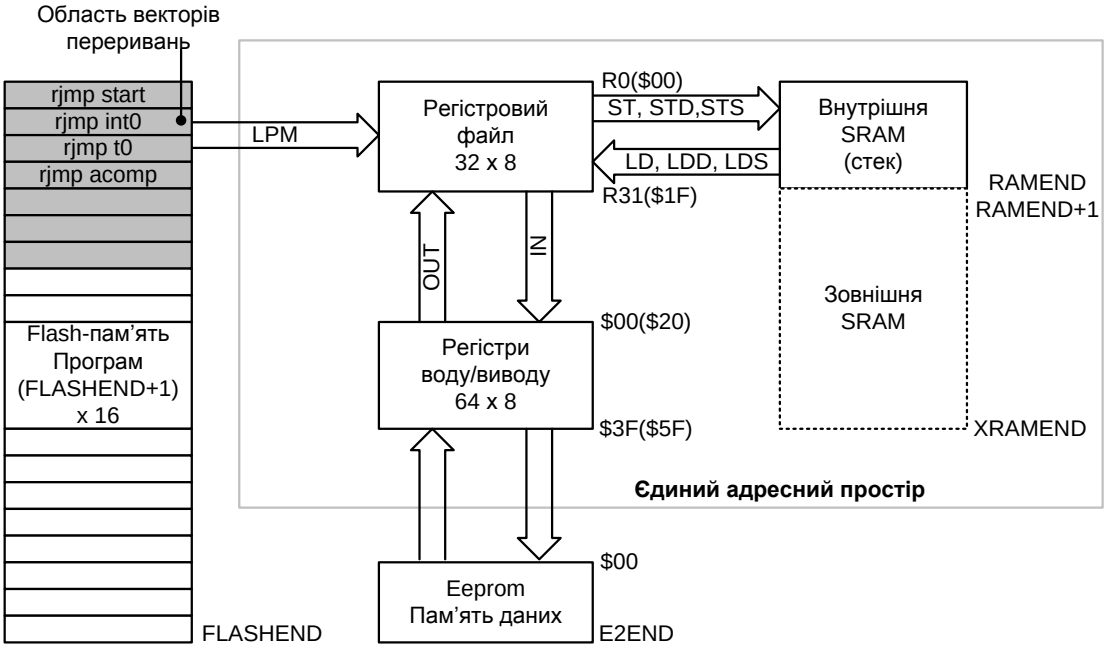


Рисунок 1.19 – Програмна модель AVR-мікроконтролерів

Завдання до лабораторної роботи

Записати в кодах мікроконтролера AT90S2313 програму, яка:

- записує в комірку SRAM1 константу CONST1;
- записує в комірку SRAM2 константу CONST2;
- переписує зміст SRAM1 в регістр з номером X;
- переписує зміст SRAM2 в регістр з номером Y;
- початкова адреса програми ADR1.

Номер варіанту завдання взяти із табл. 1.1 за порядковим номером у журналі викладача.

Зміст звіту

Короткий опис структури МК.
Таблиця команд передачі даних.
Завдання до лабораторної роботи.
Текст програми з поясненнями.

Таблиця 1.1 - Варіанти завдань

Номер	SRAM1	SRAM2	CONST1	CONST2	X	Y	ADR1
01	71	83	FE	CA	0	2	114
02	62	64	FF	AB	1	3	12F
03	83	62	FD	BC	2	4	03E
04	94	71	FC	CD	3	5	04A
05	A5	62	FB	DE	4	6	035
06	A6	73	FA	EF	5	7	026
07	C7	84	F1	FF	6	0	017
08	78	85	F2	1A	7	1	078
09	69	76	F3	2B	3	2	069
10	6A	67	F4	3C	0	3	05A
11	7B	78	F5	4D	1	4	04B
12	8C	C3	F6	5E	2	5	03C
13	9D	A3	F7	6F	3	6	02D
14	AE	B4	F8	7A	4	7	01E
15	7F	C5	F8	8B	5	0	07F
16	6E	6D	F9	9C	6	1	06E
17	8A	7E	F1	2D	7	2	05A
18	8D	6F	F2	5E	0	3	04D
19	9C	E2	F3	6F	1	4	03C
20	D6	D3	F4	7A	2	5	026
21	D5	9A	FF	8B	3	6	04B
22	67	8D	1F	9D	4	7	03C
23	CF	E4	2E	A0	5	7	02D
24	6D	B3	3D	B1	6	6	01E
25	8A	71	4C	C2	7	5	07F
26	9E	69	5B	D3	1	4	06E
27	7F	9E	6A	E4	2	3	05A

Таблиця 1.2 - Інструкції передачі даних

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
MOV	Rd,Rr	Скопіювати регістр	Rd = Rr	None	1
LDI	Rd,K8	Завантажити константу	Rd = K	None	1
LDS	Rd,k	Пряме завантаження	Rd = (k)	None	2*
LD	Rd,X	Непряме завантаження	Rd = (X)	None	2*
LD	Rd,X+	Непряме завантаження з пост-інкрементом	Rd = (X), X=X+1	None	2*
LD	Rd,-X	Непряме завантаження з пре-декрементом	X=X-1, Rd = (X)	None	2*
LD	Rd,Y	Непряме завантаження	Rd = (Y)	None	2*
LD	Rd,Y+	Непряме завантаження з пост-інкрементом	Rd = (Y), Y=Y+1	None	2*
LD	Rd,-Y	Непряме завантаження з пре-декрементом	Y=Y-1, Rd = (Y)	None	2*
LDD	Rd,Y+q	Непряме завантаження з заміщенням	Rd = (Y+q)	None	2*
LD	Rd,Z	Непряме завантаження	Rd = (Z)	None	2*
LD	Rd,Z+	Непряме завантаження з пост-інкрементом	Rd = (Z), Z=Z+1	None	2*

Продовження таблиці 1.2

LD	Rd,-Z	Непряме завантаження з пре-декрементом	Z=Z-1, Rd = (Z)	None	2*
LDD	Rd,Z+q	Непряме завантаження з заміщенням	Rd = (Z+q)	None	2*
STS	k,Rr	Пряме збереження	(k) = Rr	None	2*
ST	X,Rr	Непряме збереження	(X) = Rr	None	2*
ST	X+,Rr	Непряме збереження з пост-інкрементом	(X) = Rr, X=X+1	None	2*
ST	-X,Rr	Непряме збереження з пре-декрементом	X=X-1, (X)=Rr	None	2*
ST	Y,Rr	Непряме збереження	(Y) = Rr	None	2*
ST	Y+,Rr	Непряме збереження з пост-інкрементом	(Y) = Rr, Y=Y+1	None	2
ST	-Y,Rr	Непряме збереження з пре-декрементом	Y=Y-1, (Y) = Rr	None	2
ST	Y+q,Rr	Непряме збереження з заміщенням	(Y+q) = Rr	None	2
ST	Z,Rr	Непряме збереження	(Z) = Rr	None	2
ST	Z+,Rr	Непряме збереження з пост-інкрементом	(Z) = Rr, Z=Z+1	None	2
ST	-Z,Rr	Непряме збереження з пре-декрементом	Z=Z-1, (Z) = Rr	None	2
ST	Z+q,Rr	Непряме збереження з заміщенням	(Z+q) = Rr	None	2
LPM	Нет	Завантаження з програмної пам'яті	R0 = (Z)	None	3
LPM	Rd,Z	Завантаження з програмної пам'яті	Rd = (Z)	None	3
LPM	Rd,Z+	Завантаження з програмної пам'яті з пост-інкрементом	Rd = (Z), Z=Z+1	None	3
SPM	Нет	Збереження в програмній пам'яті	(Z) = R1:R0	None	-
IN	Rd,P	Читання порту	Rd = P	None	1
OUT	P,Rr	Запис у порт	P = Rr	None	1
PUSH	Rr	Занесення регістра в стек	STACK = Rr	None	2
POP	Rd	Витяг регістра зі стека	Rd = STACK	None	2

* Для операцій доступу до даних кількість циклів зазначена за умови доступу до внутрішньої пам'яті даних, і не коректно при роботі з зовнішнім ПЗП. Для інструкцій LD, ST, LDD, STD, LDS, STS, PUSH і POP, необхідно додати один цикл плюс по одному циклі для кожного чекання.

Контрольні запитання

1. Нарисуйте спрощену архітектуру мікроконтролера AT90S2313.
2. Що може бути використане в якості генератора тактового сигналу?
3. Наведіть схему приєднання тактового генератора.
4. Які операції необхідно виконати для запису байта у запамятовуючий пристрій EEPROM?
5. Які операції необхідно виконати для читання байта з запамятовуючий пристрій EEPROM?
6. Що входить до групи переферійних пристроїв?
7. Які Ви знаєте режими адресації пам'яті програм і даних?
8. Поясніть принцип відносної адресація пам'яті програм.
9. Розкажіть про безпосередню регістрову адресацію та наведіть їх структурні реалізації.
10. Поясніть будову непрямої адресації даних та наведіть відомі Вам структурні реалізації.

Лабораторна робота №2

ВИВЧЕННЯ ОСОБЛИВОСТЕЙ STK - 500 ТА КОНФІГУРУВАННЯ ПРОГРАМНО-ВІДЛАГОДЖУВАЛЬНИХ ЗАСОБІВ

Мета роботи: вивчити структуру мікропроцесорного комплекту, навчитися конфігурувати програмні засоби для програмування мікроконтролерів, написання, компіляція та завантаження тестової програми у мікроконтролер.

Теоретичні відомості

Корпорація Atmel повідомила про випуск нової відлагоджувальної системи - STK500, що полегшує роботу з AVR-мікроконтролерами і забезпечує підтримку програмування, як через паралельний, так і через послідовний інтерфейси. Додатково система може бути використана в якості ISP-программатора для задач прикладного програмування. Для спрощення налагодження схеми, усі AVR-порти можуть бути доступні через pin-роз‘єми.

Використовуючи інтегровану систему розробки AVR Studio 3.2 (чи вище), STK500 забезпечує розробку в режимах симуляції і емуляції (використовуючи внутрішньосхемний емулятор) і навіть програмувати усередині AVR-контролера. AVR Studio входить у комплект із STK500 чи може бути завантажена зі сторінки корпорації Atmel.

Система підтримує такі мікроконтролери:

- AT90S1200, AT90S2313, AT90S2323, AT90S2343, AT90S4433, AT90S8515, AT90S8535;
- ATmega163, ATmega161;
- ATtiny11, ATtiny12, ATtiny15, ATtiny28.

STK 500 звичайно поставляється з мікроконтролером AT90S2313 або AT90S8515, який встановлено у слот SCKT3000D3. Необхідні перемикачі (джампери) встановлено таким чином, щоб мікроконтролер працював від внутрішнього тактового генератора та джерела живлення, які встановлено на платі.

Використовуючи додаткові 10-pin кабелі необхідно з'єднати роз'єм введення/виведення "PORTB" із роз'ємом введення/виведення, який позначено "LEDS", та роз'єм "PORTD" із роз'ємом "SWITCHES". PORTB використовується для виведення значень на світлодіодні плати. PORTD використовується для введення значень із встановлених на платі перемикачів (кнопок).

Плата живиться зовнішнім джерелом живлення постійного струму 10 - 15 В. Коли живлення плати включено горить червоний світлодіод живлення, а світлодіод стану змінює свій колір (червоний, жовтий, зелений).

Схема розташування елементів програматора STK 500 представлена на рис. 2.1.

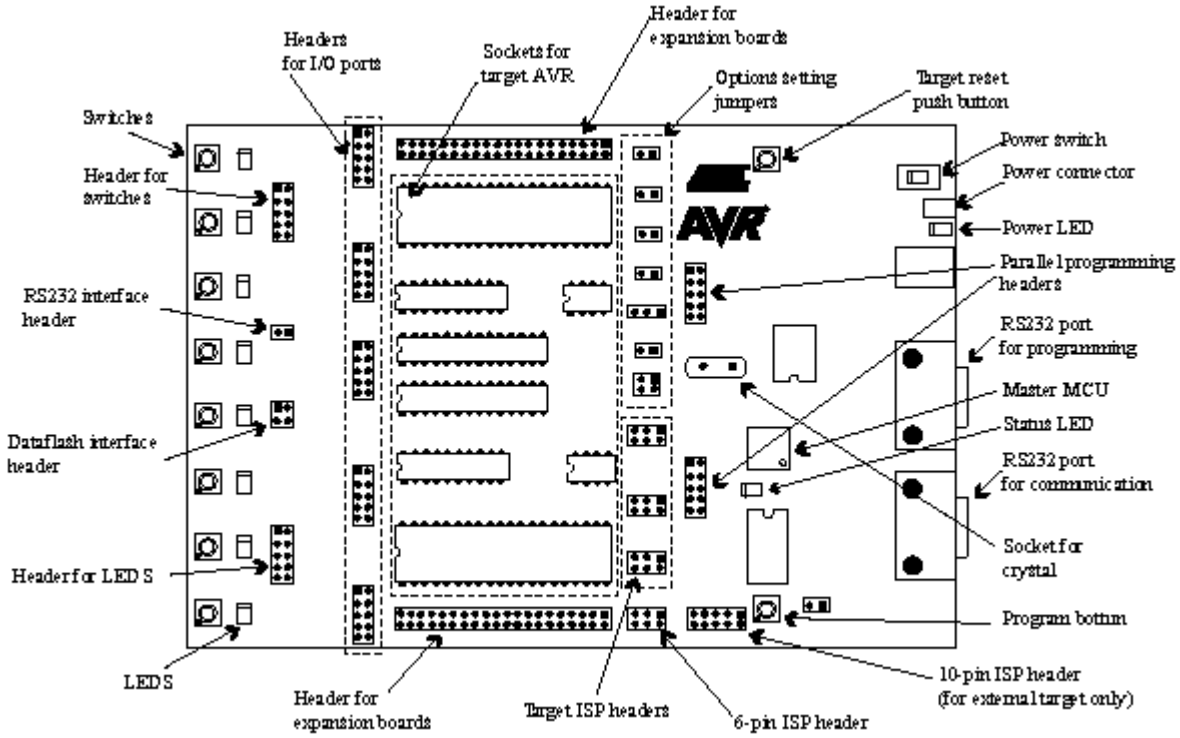


Рисунок 2.1 – Схема розташування елементів програматора STK 500

Для програмування мікронтролера AT902313 або AT908515 необхідно з'єднати 6-pin кабелем роз'єм "ISP6PIN" із роз'ємом "SPROG3" та підключити послідовний кабель від порту "RS232 CTRL" до COM порту комп'ютера.

До складу STK 500 входить програмний пакет AVR Studio, який використовується для написання програм для мікроконтролерів на мові програмування асемблер. Також AVR Studio використовується для управління та налагодження плати і для зручного завантаження програми у Flash-пам'ять.

Для написання програм на мові програмування C та C++ для мікроконтролера використовуються додаткові програмні засоби, які поставляються за ліцензією GNU та доступні користувачам Інтернет. В лабораторних роботах для написання програм на C та C++ буде використовуватись програмний пакет WinAvr, що являє собою набір виконавчих інструментів розробника для програмування Atmel AVR series of RISC мікроконтролерів на базі платформи Windows.

Завдання до лабораторної роботи

1. Programmers Notepad (PN) - це редактор програм із виділенням синтаксу. Він може бути сконфігурован таким чином, щоб виключити необхідність роботи із командною строкою для запуску програм.

Запустіть програму "Programmers Notepad" (c:\WinAVR\pn\pn.exe). При завантажені програми зявляється вікно, що зображене на рис. 2.2.

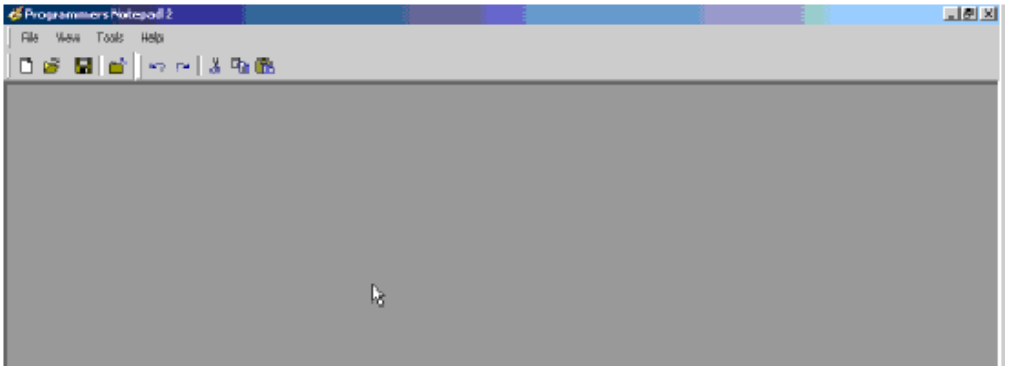


Рисунок 2.2 – Вікно завантаженої програми "Programmers Notepad"

Для конфігурації відкриваємо меню Tools -> Options menu та вибираємо пункт 'Tools' з лівої сторони діалогового вікна (рис. 2.3).

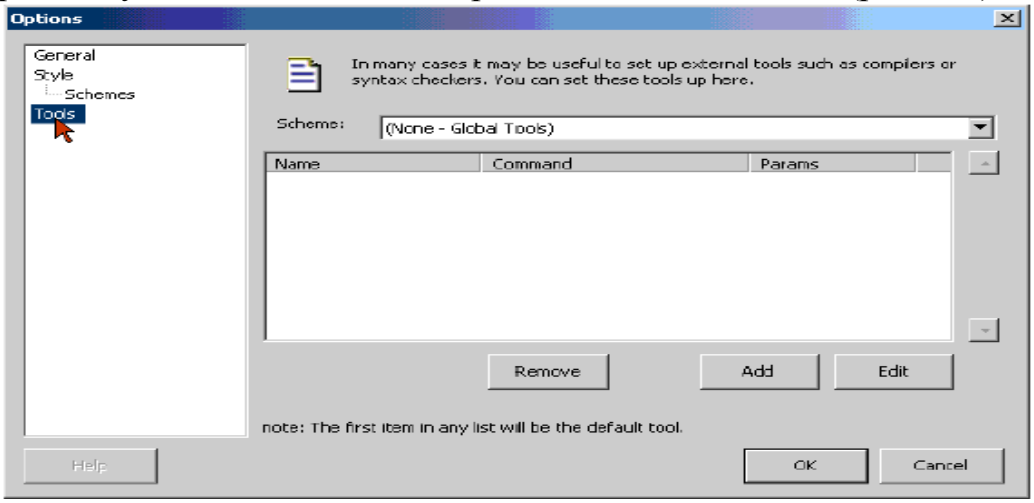


Рисунок 2.3 – Меню "Tools -> Options menu"

Потім необхідно натиснути кнопку 'Add' для створення додаткової команди, використання якої замінює роботу з командної строки. Далі вводимо дані із табл. 2.1 у відповідне поле, для кожної нової команди. Наприклад, створення команди 'Make All' виглядає таким чином (рис. 2.4).

Таблиця 2.1 – Дані для роботи із лістінгами програм

Name	Command	Folder	Parameters	Capture Output?	This tool will modify..	Save:	Clear Output?	Use Built-in Parser.
Make All	make	%d	all	Main	No	Current File	No	Yes
Make Clean	make	%d	clean	Main	No	None	No	Yes
Make Extcoff	make	%d	extcoff	Main	No	None	No	Yes
Make Coff	make	%d	coff	Main	No	None	No	Yes
Program Device	make	%d	program	Main	No	None	No	Yes

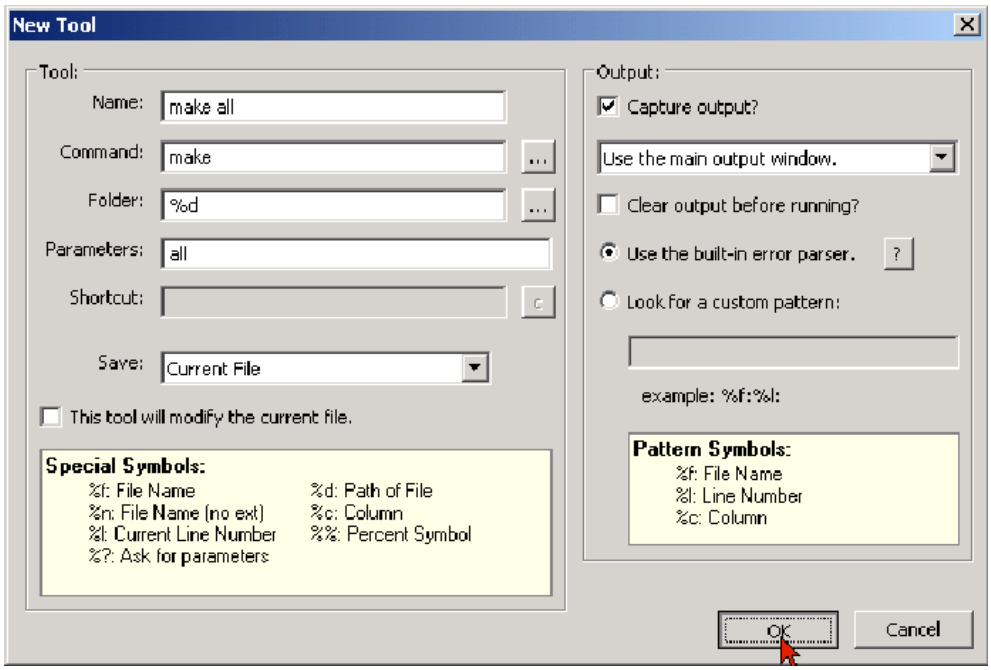


Рисунок 2.4 - Створення команди “Make All“

Задавши всі команди закриваємо вікно ‘Options’ натисканням кнопки ‘Ок’. Після цього додаткові команди роботи із програмами буде сконфігуровано.

На цьому етапі нічого не зміниться в меню 'Tools' до відкриття файлу програми, хоча все сконфігуровано. Тільки після відображення вікна з програмою з’являються додаткові команди роботи з програмою.

Команда “Make All” - використовується для компіляції програми та створення додаткових відлагоджувальних файлів, які необхідні для роботи.

Команда “Make Clean” - використовується кожного разу перед компіляцією для видалення старих файлів компіляції.

Команда “Program Device” - використовується для програмування мікроконтролера.

2. Написання мінімальної програми

Відкриваємо вікно написання нової програми C/C++.

Вводимо таку програму:

```
#include <avr/io.h>          /* підключення бібліотеки вводу/виводу  */
#include <avr/io8515.h>  /*підключення бібліотеки для роботи з
            мікропроцесором  8515  */
int main()
{
  DDRB = 0xFF;                  /*встановлення порту B, як порт виводу*/
  PORTB = 56;                   /*вивід числа 56 в порт B*/
  Return 0;                /*команда повернення „0” функції main –
команда нормального закінчення програми */
}
```

Бажано зберегти програму в окремій папці.

3. Для того, щоб створити виконавчу програму *(prog.hex)* з вихідного файлу *(prog.c)* потрібно виконати деякі інструкції. Вони мають повторюватися після кожної зміни програми. Необхідні інструкції занесені в так названий *makefile*, який компілюється за допомогою вбудованої нами у компілятор команди *MakeAll*. Початковий файл *makefile* знаходиться в папці C:\winavr\Tests\M*akefile*, перед внесенням змін його необхідно скопіювати в каталог з вашою програмою. *Makefile* виконує крок за кроком визначенні інструкції і робить так, щоб отримати відкомпільовану програму, яку можливо завантажувати у мікроконтролер.

Структура makefile, що підлягає змінам:

Запишіть назву (ім'я) контролера, якій Ви плануєте програмувати

MCU = at90s8515

В цьому місті Ви можете вставити назву для нової програми

TRG = prog

Зверніть увагу, що назва програми вказується без розширення (*.c).

Назви додаткових программ, які мають бути переведені

SRC = owninc1.c owninc2.c ownprog.c

Додаткові файли ассемблера

ASRC =

Додаткові бібліотеки

LIB =

Інші параметри бажано не змінювати без необхідності внесення додаткових команд компіляції.

4. Після редагування та збереження M*akefile* запустити команду MakeAll. В результаті у каталозі із програмою з'являються нові файли, які створенні компілятором.

5. Необхідно виконати завантаження *.hex фала у пам'ять мікроконтролера. Для цього необхідно завантажити програму "Пуск->Программи->Atmel AVR Tools->AVR Studio 4".

Для програмування файлу *hex* в AVR мікроконтролер, виберіть "STK500" з меню " tools " в меню "AVR Studio".

Виберіть AVR пристрій „Device" із меню у вікні "STK500->Program" і вкажіть *hex* файл, щоб завантажити його у Flash-пам'ять.

Натисніть кнопку "Program" для завантаження, при цьому LED статусу буде "жовтим", а якщо під час програмування виникла помилка він стане "червоним". Після програмування світлодіод знову стане "зеленим" і мікроконтролер почне виконання завантаженої програми.

Зміст звіту

Короткий опис структури STK500.

Таблиця команд для роботи із текстами програм.

Результати роботи мінімалбної програми з поясненнями.

Контрольні запитання

1. Які ви знаєте особливості AVR мікроконтролерів?
2. Нарисуйте спрощену схему розташування елементів на STK 500?
3. Які засоби використовуються для програмування мікро-контролерів?

Лабораторна робота № 3

ПРОГРАМУВАННЯ ПОРТІВ ВВЕДЕННЯ/ВИВЕДЕННЯ

Мета роботи: вивчити структуру портів введення/виведення, навчитися використовувати світлодіоди та кнопки введення програматора STK 500 для імітації роботи мікроконтролера з портами, а тпкож навчитися писати програми для мікроконтролера, що використовують порти введення/виведення для відображення отриманої інформації на світлодіодах..

Теоретичні відомості

Периферія, яка використовується AVR процесором доступна через регістри введення/виведення. Для роботи з цими регістрами необхідні спеціальні команди доступу, які визначені в підключаємих файлах <io.h>, *<iomacros.h>*, з допомогою інструкцій асемблера.

Ці регістри також називають “special function registers” (SFR) і вони визначені у файлі специфікації процесора (для AT90S8535: *<io8535.h>).*

Всі AVR порти мають дійсну функціональність "зчитування – зміна (модифікація) - запис" коли використовуються як загальні цифрові порти вводу/виводу. Це значить, що напрямок одного виводу порта може бути змінено без зміни напряму дії будь-якого іншого виводу.

Кожен порт складається з трьох регістрів:

- ***DDRX:***

це регістр напрямку зміни даних: **D**ata **D**irection **R**egister of port **X** (замість X використовується назва порта – наприклад *DDRA* це регістр порта A). Якщо ви встановити у цей регістр значення $FF, весь порт X буде визначено як порт виведення. Якщо в регістр встановити $00, весь порт X буде визначено як порт введення.

- ***PORTX:***

це регістр даних порта X (X може бути A, B, C або D). Якщо Ви бажаєте записати або зчитати дані з порта, ви маєте використовувати цей регістр.

PORTA = *0xAA*;

ця інструкція завантажує в регістр даних порта А шістнадцяткове значення *0xAA*. В цьому випадку порт А використовується як вихідний, тому ви маєте ініціалізувати його як вихідний ($FF -> *DDRA*).

int result = PORTA;

ця інструкція завантажує значення регістру даних до змінної result. В цьому випадку порт А використовується як вхідний, тому ви маєте ініціалізувати його як вхідний ($00 -> *DDRA*).

- ***PINX:***

PINX (X може бути A, B, C або D) це адреси вхідних виводів порта X. Коли зчитується PORTX, флаг даних порта X зникає, а коли зчитується PINX, логічні значення присутні на зчитаних виводах. Вхідні виводи порта X мають стан “read only”, а регістр даних та регістр напрямку даних мають стан “read/write”.

Перед використанням портів, на початку програми, необхідно ініціалізувати бажані для використання порти як порти входу чи виходу. Це виконується за допомогою регістра DDRX.

Наприклад:

DDRA= 0xFF - порт А встановлено як порт виведення;

DDRB = 0x00 - порт В встановлено як порт введення.

Для створення програмної затримки роботи мікропроцесора використовується додаткова бібліотека "delay", яка знаходиться в підключаємому файлі <avr/delay.h>. В ній визначені дві функції затримки роботи мікропроцесора :

_delay_loop_1(unsigned char __count); - 8-bit count, 3 cycles/loop;

_delay_loop_2(unsigned int __count); - 16-bit count, 4 cycles/loop.

Ці макроси дозволяють встановити затримку в мілісекундах.

Порт B 8-розрядний – двонаправлений порт для введення/виведення. Для обслуговування порту відведено три регістри: регістр даних PORTB ($18, $38), регістр напрямку даних — DDRB ($17, $37) та виводи порту B ($16, $36). Адреса виводів порту B призначена тільки для читання, у той час як регістр даних і регістр напрямку даних — для читання і запису.

Усі виводи порту мають резистори, що підтягують, що підключаються окремо. Виходи порту B можуть поглинати струм до 20мА і безпосередньо керувати світлодіодними індикаторами. Якщо виводи PB0...PB7 використовуються як входи і замикаються на землю, то при включених внутрішніх резисторах, що підтягують, виводи є джерелами струму. Додаткові функції виводів порту B приведені в таблиці 3.1.

При використанні альтернативних функцій виводів (alternative functions of conclusions) регістри DDRB і PORTB повинні бути установлені відповідно до опису альтернативних функцій (рис. 3.1 та рис. 3.2). При використанні можливості внутрішньосхемного

програмування мікроконтролера варто враховувати, що програматор використовує для своєї роботи лінії MOS1, MISO і SCK. Відповідно, пристрої, підключені до цих ліній, не повинні заважати роботі програматора.

Таблиця 3.1 - Альтернативні функції виводів порту B

Вивід	Альтернативна функція
PB0	AIN0(позитивний вхід аналогового компаратора)
PB1	AIN1(негативний вхід аналогового компаратора)
PB3	OC1(вихід збігу таймера/лічильника 1)
PB5	MOS1(вхід даних для SPI)
PB6	MISO(вихід даних для SPI)
PB7	SCK(вхід тактових імпульсів SPI)

Біт	7	6	5	4	3	2	1	0
$17 ($37)	ACD	-	AC0	AC1	ACIE	ACIC	ACIS1	ACIS0
Чт. / зал.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R /W
Поч. знач.	0	0	0	0	0	0	0	0

Рисунок 3.1 – Регістр напрямку даних порту B – DDRB

Біт	7	6	5	4	3	2	1	0
$16 ($36)	ACD	-	AC0	AC1	ACIE	ACIC	ACIS1	ACIS0
Чт. / зал.	R	R	R	R	R	R	R	R
Поч. знач.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Рисунок 3.2 – Виводи порту B – PINB

PINB не є регістром, по цій адресі здійснюється доступ до фізичних значень кожного з виводів порту B. При читанні PORTB читаються дані з регістра-засувки, при читанні PINB читаються логічні значення, що відповідають фактичному стану виводів порту.

Порт B, як порт вводу/виводу загального призначення

Біт DDBn регістра DDRB вибирає напрямок передачі даних (табл.3.2). Якщо біт установлений (тобто дорівнює одиниці), вивід має конфігурацію як вихід. Якщо біт скинутий (тобто дорівнює нулю) — вивід

має конфігурацію як вхід. Якщо PORTBn встановлений і вивід має конфігурацію як вхід, включається КМОП, що підтягує резистор. Для відключення резистора PORTBn повинний бути скинутий чи вивід повинний мати конфігурацію як вихід.

Таблиця 3.2 – Вплив DDBn на висновки порту B

DDBn	PORT B	Вхід/вихід	Резистор, що підтягує	Коментар
0	0	Вхід	Немає	Третій стан(Hi-Z)
0	1	Вхід	є	PBn джерело струму, якщо з'єднаний із землею
1	0	Вихід	Немає	Вихід встановлений у 0
1	1	Вихід	Немає	Вихід встановлений у 1
Примітка: n = 7,6..0 – номер виводу.				

Альтернативні функції PORTB:

Біт 7 SCK – вхід тактової частоти для SPI.

Біт 6 MISO – вихід даних для SPI.

Біт 5 MOSI – вхід даних для SPI.

Біт 3 OC1 – вихід збігу. Цей вивід може мати конфігурацію для зовнішнього виводу події збігу таймера 1. Для цього біт DDB3 повинний бути встановлений у 1 (вивід має конфігурацію як вихід).

Біт 1 AIN1 – негативний вхід аналогового компаратора. Якщо цей вивід має конфігурацію як вхід (DDB1=0), і відключений внутрішній резистор, що підтягує, цей вивід працює як негативний вхід внутрішнього аналогового компаратора.

Біт 0 AIN0 – позитивний вхід аналогового компаратора. Якщо цей вивід має конфігурацію як вхід (DDB0=0) і відключений внутрішній резистор, що підтягує, це вивід працює як позитивний вхід внутрішнього аналогового компаратора.

Для порту D зарезервовані 3 комірки пам'яті: регістр PORTD $12 ($32), регістр напрямку даних – DDRD $11($31) і виводи порту D — PIND $10 ($30). Регістри даних і напрямку даних можуть читатися/записуватися, вивід PIND – тільки для читання.

Порт D – 7-розрядний двонапрямлений з убудованими регістрами, що підтягують. Вихідні буфери поглинають струм до 20 мА. Якщо виводи використовуються як входи і на них поданий низький рівень, то при підключенні резисторів, що підтягують, вони є джерелами струму. Деякі з виводів порту мають альтернативні функції, показані в таблиці.

Якщо виводи порту використовуються для обслуговування альтернативних функцій, вони повинні мати конфігурацію відповідно до опису функції (табл. 3.3).

PIND не є регістром, по цій адресі здійснюється доступ до фізичних значень кожного з виводів порту D. При читанні PORTD читаються дані з регістра-засувки, при читанні PIND читаються логічні значення, що відповідають фактичному стану виводів порту.

Таблиця 3.3 – Альтернативні функції порту D

Вивід	Альтернативна функція
PD0	RXD(вхід даних UART)
PD1	TXD(вихід даних UART)
PD2	INT0(вхід зовнішнього переривання 0)
PD3	INT1(вхід зовнішнього переривання 1)
PD4	T0(зовнішній вхід таймера/лічильника 0)
PD5	T1(зовнішній вхід таймера/лічильника 1)
D6	ICP(вхід захоплення таймера/лічильника 0)

Біт	7	6	5	4	3	2	1	0
$12 ($32)	-	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
Чт. / зал.	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Поч. знач.	0	0	0	0	0	0	0	

Рисунок 3.3 - Регістр даних порту D – PORTD

Біт	7	6	5	4	3	2	1	0
$11 ($31)	-	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
Чт. / зал.	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Поч. знач.	0	0	0	0	0	0	0	

Рисунок 3.4 - Регістр напрямлення порту D – DDRD

Біт	7	6	5	4	3	2	1	0
$10 ($30)	-	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
Чт. / зал.	R	R	R	R	R	R	R	R
Поч. знач.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Рисунок 3.5 – Висновки порту D — PIND

Порт D як порт введення/виведення загального призначення. Біт DDDn регістра DDRD вибирає напрямок передачі даних. Якщо біт установлений, вивід має конфігурацію як вихід. Якщо біт скинутий — вивід має конфігурацію як вхід. Якщо PORTDn встановлений і вивід має конфігурацію як вхід, включається КМОП резистор, що підтягує. Для відключення резистора PORTDn повинний бути скинутий чи вивід повинний мати конфігурацію як вихід (табл. 3.4).

Таблиця 3.4 – Вплив DDDn на виводи порту D

DDDn	PORTDn	Вхід/вихід	Резистор, що підтягує	Коментар
0	0	Вхід	Немає	Третій стан(Hi-Z)
0	1	Вхід	Є	PBn джерело струму, якщо з'єднаний із землею
1	0	Вихід	Немає	Вихід встановлений у 0
1	1	Вихід	Є	Вихід встановлений у 1
Примітка: n = 6..0 – номер виводу.				

Альтернативні функції порту D

Біт 6 ICP – вхід захоплення (entrance of fascination) таймера/лічильника 1.

Біт 5 T1 – тактовий вхід (time entrance) таймера/лічильника 1.

Біт 4 T0 – тактовий вхід таймера/лічильника 0.

Біт 3 INT1 - вхід зовнішніх переривань (entrance of external interrupts)1.

Біт 2 INT0 – вхід зовнішніх переривань 0.

Біт 1 TXD – вихід передавача UART. Якщо дозволена робота передавача UART, незалежно від стану DDRD1 цей вивід має конфігурацію як вихід.

Біт 0 RXD – вихід приймача UART. Якщо дозволена робіта приймача UART, незалежно від стану DDRD0 цей вивід має конфігурацію як вихід. Коли UART використовує вивід для прийому даних, одиниця в PORTD0 підключає вбудований резистор (a resistor is built-in), що підтягує.

Для використання кнопок "switch's" STK500 необхідно встановити порт до якого вони підключені - як порт введення. Для використання "led's" світлодіодів необхідно встановити порт до якого вони підключені - як порт виведення.

При використанні порта введення значення з порту зчитується цілим байтом. Наприклад при натисканні "switch1" стан порту вводу виглядає наступним чином : 0000 0001 в двійковій системі числення або 0x01 у шістнадцятковій; при натисканні switch5 - 0000 0101 або 0x05.

Таким чином, для визначення того, яка натиснута кнопка, необхідно порівнювати значення порта вводу з відповідними значеннями необхідних для використання кнопок. Так само з світлодіодами (by light-emitting diodes), для використання певної їх послідовності, необхідно записати у порт виводу відповідне значення.

Завдання до лабораторної роботи

1.Запустити <PN>.
2.Створити нову програму на мові С, яка встановлює на світлодіодах значення поточного року. Після деякої затримки стан світлодіодів скидається і встановлюється номер поточного місяця, так само після затримки, встановлюється число.
3.Завантажити програму в мікроконтролер і перевірити виконання.
4.Створити нову програму на мові С, яка відображає на світлодіодах хід зміни регістру, від 0 до максимального значення, на 2, 5, 10 у прямому та зворотньому напрямках.
5.Завантажити програми в МК і перевірити їх виконання.
6.Створити програму на мові С, яка при натисканні на одну з кнопок switch1 – switch7 висвітлює відповідний світлодіод, а switch0 використовується для скиду станів світлодіодів.
7.Завантажити програму в мікроконтролер і перевірити виконання.
8.Створити нову програму на мові С, яка при натисканні switch1, 2, 3 змінює регістр, від 0 до максимального значення, відповідно на 2, 5, 10 та відображає поточний стан регістра на світлодіодах.
9.Завантажити програму в мікроконтролер і перевірити її роботу.

Зміст звіту

Завдання до лабораторної роботи.
Лістінги програм на мові С з поясненнями.
Результати роботи програм на STK 500.

Контрольні запитання

1. Які ви знаєте стандартні порти мікроконтролера?
2. З яких регістрів складається порт мікроконтролера?
3. Яким чином встановлюється напрямок зміни даних в портах?
4. Архітектура побудови портів введення/виведення в AVR?
5. Основні характеристики мікроконтролера AT90S8515?

Лабораторна робота №4

КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ. ОРГАГІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ

Мета роботи: вивчення організації адресного простору пам'яті програм мікроконтролера AT90S2313, програмних засобів керування ходом виконання програми, отримання навичків програмування циклічних алгорітмів в кодах мікроконтролера.

Теоретичні відомості

1. Програмна модель AT90S2313

Так само, як і в інших мікроконтролерів, що вбудовуються, система команд AVR включає команди арифметичних і логічних операцій, команди передачі даних, команди, що керують послідовністю виконання програми і команди операцій з бітами.

Маючи 16-розрядну комірку пам'яті програм, AVR відрізняються багатством своєї системи команд у порівнянні з іншими RiSC-мікроконтролерами.

Для зручності написання й аналізу програм всім операціям із системи команд крім двійкового коду зіставлені мнемокоди Ассемблера (символічні позначення операцій), що використовуються при створенні вихідного тексту програми. Спеціальні програми-транслятори переводять потім символічні позначення в двійкові коди.

Спеціальна директива ассемблера

.device <типAVR>

забезпечує контроль відповідності команд, використовуваних у тексті програми, типу зазначеного процесора.

При переході від молодших до старших моделей AVR існує сумісність у змісті системи команд, однак необхідно пам'ятати, що адреси векторів переривання тих самих периферійних вузлів у різних типів AVR різні, що вимагає внесення відповідних змін у програму при її переносі на інший тип AVR.

На рис. 4.1 зображена програмна модель AVR-мікроконтролерів, що являє собою діаграму програмно доступних ресурсів AVR. Центральним блоком на цій діаграмі є реєстровий файл на 32 оперативних регістра (R0-R31), безпосередньо доступних ALU. Старші регістри об'єднані парами й утворять три 16-розрядних регістри, призначених для непрямої адресації комірок пам'яті .

Всі арифметичні і логічні операції, а також частина операцій роботи з бітами виконуються в ALU тільки над вмістом оперативних регістрів. Варто звернути увагу, що команди (SUBI, SBCI, ANDI, ORI, SBR, CBR) у якості другого операнда мають константу, можуть використовувати в

якості першого операнда тільки регістри з другої половини реєстрового файлу (R16-R31). Команди 16-розрядного додавання з константою ADI і віднімання константи SBI у якості першого операнда використовують тільки регістри R24, R26, R28, R30. Під час виконання арифметичних і логічних операцій чи операцій роботи з бітами ALU формує ті чи інші (див. таблицю 1) ознаки результату операції, тобто чи встановлює скидає біти в регістрі стану SREG (Status Register).

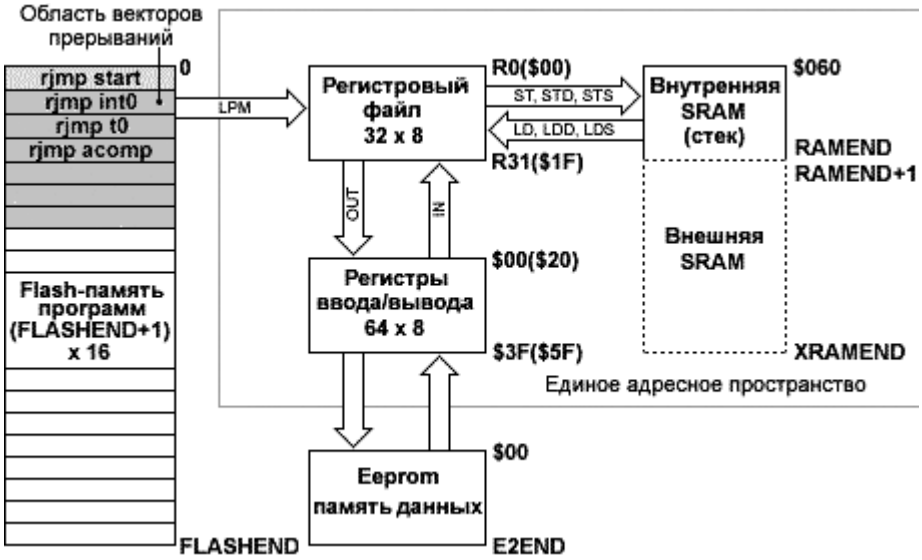


Рисунок 4.1 – Програмна модель AVR-мікроконтролерів

2. Регістр статусу - SREG

Регістр статусу - SREG - розміщений у просторі I/O за адресою $3F ($5F) і його біти визначаються як показано в табл. 4.1.

Bit 7 - I: Global Interrupt Enable - Дозвіл глобального переривання. Біт дозволу глобального переривання для дозволу переривання повинний бути встановлений у стан 1. Керування дозволом конкретного переривання виконується регістрами маски переривання GIMSK і TIMSK. Якщо біт глобального переривання очищений (у стані 0), то жодне з дозволів конкретних переривань, встановлених у регістрах GIMSK і TIMSK, не діє. Біт I апаратно очищається після переривання і встановлюється для наступного дозволу глобального переривання командою RETI.

Bit 6 - T: Bit Copy Storage - Біт збереження копії. Команди копіювання біта BLD (Bit Loa) і BST (Bit STore) використовують біт T як біт джерело і біт призначення при операціях з бітами. Командою BST біт регістра реєстрового файлу копіюється в біт T, командою BLD біт T копіюється в регістр реєстрового файлу.

Bit 5 - H: Half Carry Flag - Прапор напівпереносу. Прапор напівпереносу вказує на напівперенос у ряді арифметичних операцій

Bit 4 - S: Sign Bit, S = N V - Біт знака. Біт S завжди знаходиться в стані, обумовленому логічним що виключає ЧИ (exclusive OR) між

прапором негативного значення N і доповненням до двох прапора переповнення V.

Bit 3 - V: Two's Complement Overflow Flag . Доповнення до двох прапора переповнення. Доповнення до двох прапора V підтримує арифметику доповнення до двох.

Bit 2 - N: Negative Flag - Прапор негативного значення. Прапор негативного значення N вказує на негативний результат ряду арифметичних і логічних операцій.

Bit 1 - Z: Zero Flag -Прапор нульового значення. Прапор нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій.

Bit 0 - C: Carry Flag -Прапор переносу. Ознаки результату операції можуть бути використані в програмі для виконання подальших арифметично-логічних операцій чи команд умовних переходів.

Таблиця 4.1 - Регістр статусу SREG

Біти	7 6 5 4 3 2 1 0	
$3F ($5F)	**I** **T** **H** **S** **V** **N** **Z** **C**	**REG**
Читання/Запис	R/W R/W R/W R/W R/W R/W R/W R/W	

Для збереження оперативних даних програміст, крім реєстрового файлу, може використовувати внутрішню і зовнішню (якщо вони маються) блоки SRAM (рис. 4.1).

Операції обміну з внутрішньою оперативною пам'яттю AVR-мікроконтроллер виконує за два машинних цикли. Доступ до зовнішнього SRAM вимагає одного додаткового циклу на кожен байт у порівнянні з внутрішньою пам'яттю. Крім того, установкою біта SRW у регістрі вводу/виводу MCUSR можна програмно збільшити час обміну з зовнішньої SRAM ще на один додатковий машинний цикл очікування.

Виконувати арифметично-логічні операції й операції зрушення безпосередньо над змістом комірок пам'яті не можна. Не можна також записати чи константу очистити вміст комірки пам'яті. Система команд AVR дозволяє лише виконувати операції обміну даними між осередками SRAM і оперативними регістрами. Достоїнством системи команд можна вважати різноманітні режими адресації комірок пам'яті.

Усі регістри введення/виведення можуть зчитуватися і записуватися через оперативні регістри за допомогою команд IN, OUT (див. групу команд передачі даних). Регістри вводу/виводу, що мають адреси в діапазоні $00 - $1F (знак $ указує на шестнадцатеричную систему числення), мають можливість побітової адресації. Безпосередня установка і скидання окремих розрядів цих регістрів виконується командами SBI і CBI (див. групу команд роботи з бітами). Для ознак результату операції, що є бітами регістра вводу/виводу SREG, мається цілий набір команд установки і скидання. Команди умовних переходів у якості своїх операндов можуть мати як біти-ознаки результату операції, так і окремі розряди побитно адресуемих регістрів введення/висновку.

На рис. 4.1 показаний розподіл адрес у єдиному адресному просторі. Молодші 32 адреси ($0 - $1F) відповідають оперативним регістрам. Наступні 64 адреси ($20 - $5F) зарезервовані для регістрів вводу/виводу. Внутрішня SRAM у всіх AVR починається з адреси $60.

Таким чином, регістри введення/виведення мають подвійну нумерацію. Якщо використовуються команди IN, OUT, SBI, CBI, SBI, SBI, то варто використовувати нумерацію регістрів введення/виведення, що починається з нуля (назвемо її основний). Якщо ж до регістрів введення/виведення доступ здійснюється як до комірок пам'яті, то необхідно використовувати нумерацію єдиного адресного простору оперативної пам'яті даних AVR. Очевидно, що адреса в єдиному адресному просторі пам'яті даних виходить шляхом додатка числа $20 до основної адреси регістра введення/висновку.

Крім оперативної пам'яті програмно доступними ресурсами мікроконтролера є енергонезалежні, электрически програмувальні FLASH і EEPROM блоки пам'яті, що мають окремі адресні простори.

Тому що всі команди AVR являють собою 16-розрядні слова, FLASH-пам'ять організован як послідовність 16-розрядних осередків і має ємність від 512 слів до 64K слів у залежності від типу кристала.

В FLASH-пам'ять, крім програми, можуть бути записані постійні дані, що не змінюються під час функціонування мікропроцесорної

системи. Це різні константи, таблиці знакогенераторов, таблиці лінеаризації датчиків і т.п. Дані з FLASH пам'яті можуть бути програмним образом зчитані в реєстровий файл за допомогою команд LPM, ELPM (див. групу команд передачі даних).

Молодші адреси пам'яті програм мають спеціальне призначення. Адреса $0000 є адресою, з якого починає виконуватися програма після скидання процесора. Починаючи з наступного адреси $0001, комірки пам'яті програм утворять область векторів переривання. У цій області для кожного можливого джерела переривання відведена своя адреса, по якому (у випадку використання даного переривання) розміщають команду відносного переходу RJMP на підпрограму обробки переривання (рис. 1). Варто пам'ятати, що адреси векторів переривання тих самих апаратних вузлів для різних типів AVR можуть мати різне значення. Тому для забезпечення переносимости програмного забезпечення зручно, так само як і у випадку з регістрами вводу/виводу, використовувати символічні імена адрес векторів переривання, що визначені у відповідному inc-файле.

EEPROM блок пам'яті, що электрически стирається, даних AVR призначений для збереження енергонезалежних даних, що можуть змінюватися безпосередньо на об'єкті. Це калібровані коефіцієнти, різні уставки, конфігураційні параметри системи і т.п. EEPROM-пам'ять даних може бути програмним шляхом як лічена, так і записана. Однак спеціальних команд звертання до EEPROM-пам'яті немає. Читання і запис комірок EEPROM виконується через регістри вводу/виводу EEAR (регістр адреси), EEDR (регістр даних) і EECR (регістр керування).

3. Команди передачі керування

Режими адресації пам'яті програм і даних при передачі керування.

При звертанні до Flash пам'яті програм і пам'яті даних (SRAM, реєстровому файлу і пам'яті I/O) AVR Enhanced RISC мікроконтролерами використовуються потужні й ефективні режими адресації.

Непряма адресація даних зі зсувом розражена на рис. 4.2.

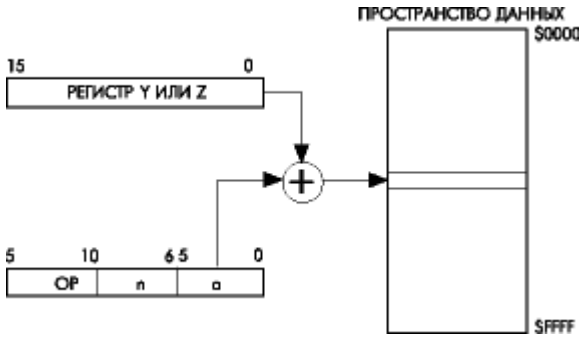


Рисунок 4.2 - Непряма адресація даних зі зсувом

Адреса операнда обчислюється підсумовуванням вмісту регістра Y чи Z з 6 бітами адреси, що містяться в слові команди.

Непряма адресація даних зображена на рис. 4.3.

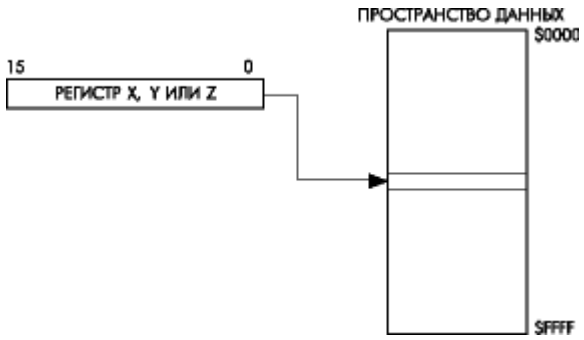


Рисунок 4.3 - Непряма адресація даних

Адреса операнда міститься в регістрі X, Y чи Z.

Непряма адресація даних із преддекрементом (рис. 4.4).

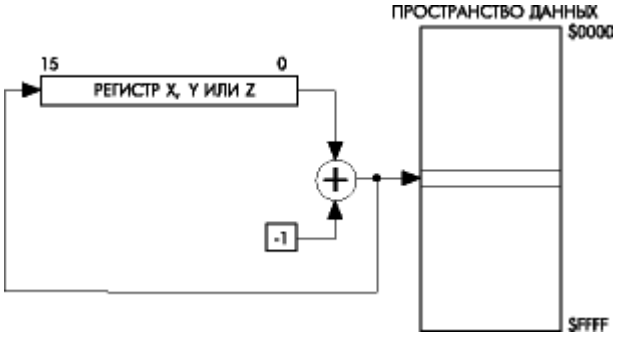


Рисунок 4.4 - Непряма адресація даних із преддекрементом

Перед виконанням операції регістр X, Y чи Z декрементується. Декрементований вміст регістра X, Y чи Z є адресою операнда.

Непряма адресація даних з постінкрементом (рис. 4.5).

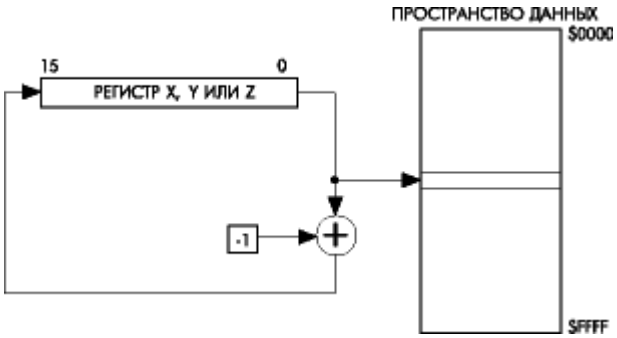


Рисунок 4.5 - Непряма адресація даних з постінкрементом

Після виконання операції регістр X, Y чи Z інкрементуться. Адресою операнда є вміст X, Y чи Z регістра попередньо інкрементовано.

Адресація константи з використанням команд LPM (рис. 4.6).

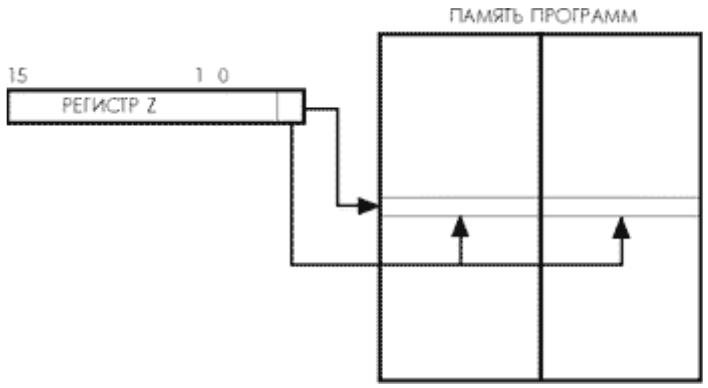


Рисунок 4.6 - Адресація константи коду пам'яті

Адреса байта константи визначається вмістом регістра Z. Старші 15 бітів визначають слово адреси (від 0 до 1К х 16).

Непряма адресація пам'яті програм, команди IJMP і ICALL (рис.4.7).

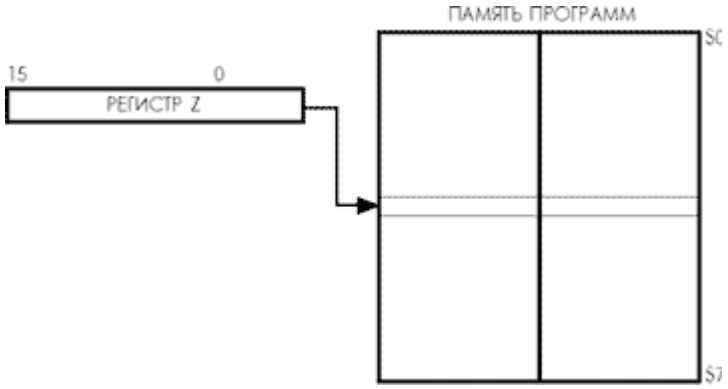


Рисунок 4.7 - Непряма адресація пам'яті програм

Виконання програми продовжується з адреси, що міститься в регістрі Z (тобто лічильник команд завантажується вмістом регістра Z).

Відносна адресація пам'яті програм, команди RJMP і RCALL (рис.4.8).

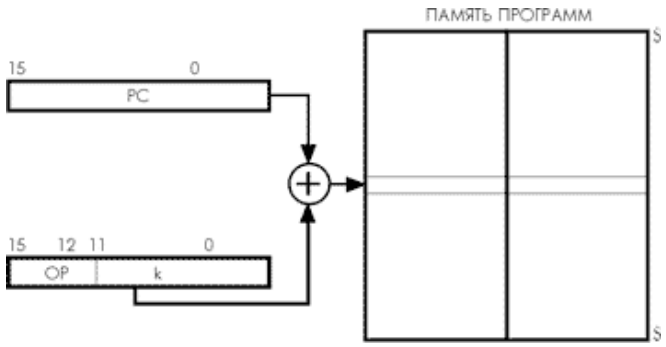


Рисунок 4.8 - Відносна адресація пам'яті програм

Виконання програми продовжується з адреси PC + k + 1. Значення відносної адреси може бути від -2048 до 2047.

Групу команд передачі керування утворюють команди безумновного переходу, умовного переходу, команди виклику підпрограми і команди повернення з підпрограми. Характеристики команд приведені в таблиці 4.2.

Таблиця 4.2 - Інструкції розгалуження

Мнемоніка	Операнди	Опис	Операція		Цикли
RJMP	k	Відносний перехід	PC = PC + k +1	None	2
IJMP	Нет	Непрямий перехід на (Z)	PC = Z	None	2
EIJMP	Нет	Розширений непрямий перехід на (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	None	2
JMP	k	Перехід	PC = k	None	3
RCALL	k	Відносний виклик підпрограми	STACK = PC+1, PC = PC + k + 1	None	3/4*
ICALL	Нет	Непрямий виклик (Z)	STACK = PC+1, PC = Z	None	3/4*
RET	Нет	Повернення з підпрограми	PC = STACK	None	4/5*
RETI	Нет	Повернення з переривання	PC = STACK	I	4/5*
CPSE	Rd,Rr	Порівняти, пропустити якщо рівні	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Порівняти	Rd -Rr	Z,C,N,V, H,S	1
CPC	Rd,Rr	Порівняти з переносом	Rd - Rr - C	Z,C,N,V, H,S	1
CPI	Rd,K8	Порівняти з константою	Rd - K	Z,C,N,V, H,S	1
SBRC	Rr,b	Пропустити якщо біт у регістрі очищений	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3
SBRS	Rr,b	Пропустити якщо біт у регістрі встановлений	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Пропустити якщо біт у порту очищений	if(I/O(P,b)==0) PC = PC + 2 or 3	None	1/2/3
SBIS	P,b	Пропустити якщо біт у порту встановлений	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3
BRBC	s,k	Перейти якщо прапор у SREG очищений	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	Перейти якщо прапор у SREG установлений	if(SREG(s)==1) PC = PC + k + 1	None	1/2
BREQ	k	Перейти якщо дорівнює	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Перейти якщо не дорівнює	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Перейти якщо перенос установлений	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Перейти якщо перенос очищений	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Перейти якщо дорівнює чи більше	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Перейти якщо менше	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Перейти якщо мінус	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Перейти якщо плюс	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Перейти якщо більше чи дорівнює (зі знаком)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Перейти якщо менше (зі знаком)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Перейти якщо прапор внутрішнього переносу встановлений	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Перейти якщо прапор внутрішнього переносу очищений	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Перейти якщо прапор T встановлений	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Перейти якщо прапор T очищений	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Перейти якщо прапор переповнення встановлений	if(V==1) PC = PC + k + 1	None	1/2

Продовження таблиці 4.2

BRVC	k	Перейти якщо прапор переповнення очищений	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Перейти якщо переривання дозволені	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Перейти якщо переривання заборонені	if(I==0) PC = PC + k + 1	None	1/2

*Для операцій доступу до даних кількість циклів зазначена за умови доступу до внутрішньої пам'яті даних, і не коректно при роботі з зовнішнім ОЗУ. Для інструкцій ICALL, EICALL, RCALL, RET і RETI, необхідно додати три цикли плюс по два циклу для кожного чекання в контролерах з PC меншим 16 біт (128KB пам'яті програм). Для пристроїв з пам'яттю програм понад 128KB , додайте п'ять циклів плюс по трьох циклу на кожне чекання.

Rd: Результуючий (і вихідний) регістр у реєстровому файлі;

Rr: Вихідний регістр у реєстровому файлі;

b: Константа (3 біти), може бути константне вираження;

s: Константа (3 біти), може бути константне вираження;

P: Константа (5-6 біт), може бути константне вираження;

K6: Константа (6 біт), може бути константне вираження;

K8: Константа (8 біт), може бути константне вираження;

k: Константа (розмір залежить від інструкції), може бути константне вираження;

q: Константа (6 біт), може бути константне вираження;

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

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

Завдання до лабораторної роботи

В пам’яті програм, починаючи з комірки ADR2, розтошована таблиця кодів довжиною N.

Записати в кодах AT90S2313 програму, яка виконує пересилку даного масиву в SRAM, починаючи з адреси ADR3. Програма повинна починатись з комірки ADR1.

Таблиця 4.3 - Варіанти завдань

Номер	ADR1	ADR2	N	ADR3
01	071	C3	E	6F
02	062	C4	F	63
03	053	C2	D	75
04	044	C1	C	86
05	035	C2	B	68
06	026	C3	A	6D
07	017	A4	6	6F
08	078	A5	7	7A
09	069	A6	F	7C

Продовження таблиці 4.3

10	05A	A7	E	74
11	04B	A8	5	73
12	03C	B3	6	78
13	02D	BA	7	8C
14	01E	BB	8	6C
15	07E	BC	8	6E
16	06E	6D	9	8A
17	05A	7E	F	6A
18	04D	AF	A	6B
19	08C	BE	B	9B
20	026	CD	C	7B
21	04B	88	D	69
22	03C	63	E	87
23	02D	6A	F	85
24	01E	6B	8	83
25	07F	6C	8	AD
26	06E	62	9	A5
27	05A	7E	C	AE

Коди задавати довільно.

Зміст звіту

Короткий зміст статусу SREG;
таблиця команд передачі керування;
дані варіанту завдання до лабораторної роботи;
текст програми з поясненнями.

Приклад виконання завдання

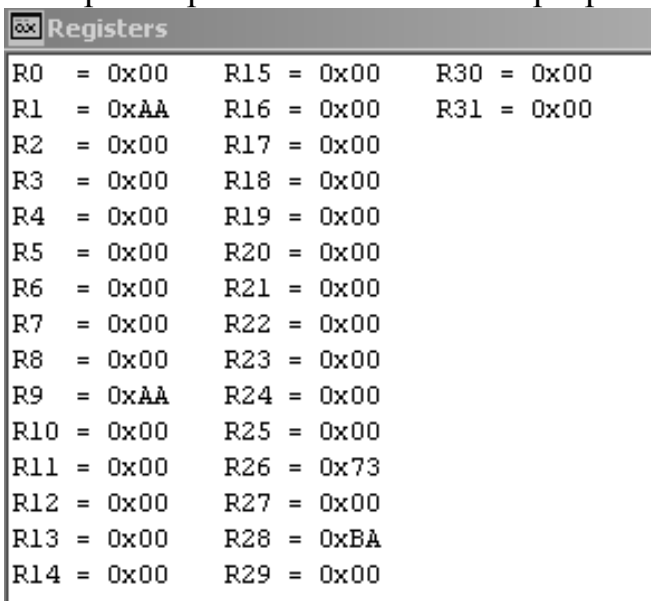
В пам’яті команд з адреси ADR2 = 67h розташовано N = OCh шістнадцятирозрядних кодів. Например: FF, 00, 11, 22, 33, 44, 55, 66, 77, 88, 99, AA. Необхідно переписати їх до пам’яті даних, починаючи з адрес ADR3 = AEh. Програма повинна починатися з адреси ADR1 = 05Ah.

Текст програми:

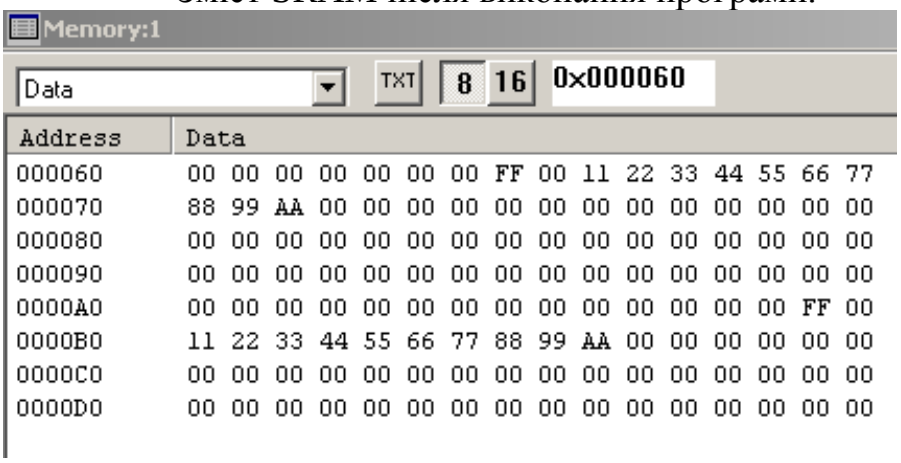
```
LDI R16,$0C;     запис до регистру R16 лічильника чисел N
CLR R27;         обнулення змісту верхньої частки регистрової пари  X
LDI R26,$67;     запис в регістрову пару X адреси  ADR2
CLR R29;         обнулення змісту верхньої частки регистрової пари  Y
LDI R28,$AE ;    запис в регістрову пару Y адреси  ADR3
L1:
LD R1,X;         завантажити в R1 числа, адреса якого знаходиться в X
                 (R27R6)
```

```
MOV R9,R1;      пересилка змісту R1 в R9
ST Y,R9 ;       запис числа з R9 по адресі, яка знаходиться вY (R29R28)
DEC R16;        зменьшити зміст лічильника на 1
ADIW R26,1;     збільшити адресу X на 1
ADIW R28,1;     збільшити адресу Y на 1
CPI R16,0;      порівняти зміст R16 з 0
BRNE L1;        якщо R16 ≠ 0 то перейти по L1
NOP
```

Зміст регистрів після виконання програми



Зміст SRAM після виконання програми.



Контрольні запитання

1. Назвіть призначення бітів регустра статсу SREG.
2. Які Ви знаєте види адресації?
3. Нарисуйте схему програмної моделі AVR-мікроконтролера.

Лабораторна робота №5

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ AVR

Мета роботи: вивчення способів програмування мікроконтролера AT90S2313, програмних засобів (AVR Studio) для відлагодження програмного забезпечення.

Теоретичні відомості

1. Способи програмування енергонезалежної пам'яті AVR

У процесі програмування AVR-микроконтроллеров виконують наступні операції по стиранню, читанню і запису різних елементів енергонезалежної пам'яті кристала:

- операція "Chip erase" (стирання кристала);
- читання/запис FLASH-пам'яті програм;
- читання/запис EEPROM пам'яті даних;
- читання/запис конфігураційних FUSE-битий;
- читання/запис LOCK-біт захисту програмної інформації;
- читання SYGNATURE-біт ідентифікації кристала.

AVR-мікроконтролери поставляються зі стертими убудованими FLASH і EEPROM блоками пам'яті (зміст всіх комірок = $FF), готовими до програмування.

У табл. 5.1 перераховані можливі способи програмування елементів енергонезалежної пам'яті AVR.

Таблиця 5.1 – Способи програмування AVR

	FLASH	LOCK-біти	FUSE-біти	EEPROM
Папалельне програму-вання	+	+	+	+
Послідовне програму-вання	+	+	- (+)	+
Само-програму-вання	Анонсовано	-	-	+

Паралельне програмування вимагає використання додаткового джерела підвищеної напруги (12 В), використовує велике число виводів мікроконтролера і виконується на спеціальних програматорах. Таке

програмування зручне, коли при масовому виробництві необхідно "прошивати" велику кількість кристалів.

Послідовне програмування (Successive programming) не вимагає додаткового джерела живлення і може виконуватися безпосередньо в мікропроцесорній системі (In System Programming) через послідовний SPI-інтерфейс, що використовує всього чотири виводи AVR-мікроконтролера. Можливість внутрісистемного програмування є одним з найважливіших переваг AVR, тому що дозволяє значно спростити та удешевити процес розробки і модернізації програмного забезпечення. Паралельний і послідовний способи програмування припускають використання зовнішнього програмуючого процесора. EEPROM пам'ять може також програмуватися самим AVR під керуванням програми (самопрограмування).

LOCK-біти програмуються як паралельно, так і послідовно. FUSE-біти в молодших моделей AVR можуть програмуватися тільки послідовно, а в старших - і паралельно, і послідовно. SYGNATURE-байти доступні для читання при будь-якому способі програмування, якщо кристал не засекречений LOCK-бітами.

Операція "Chip erase" виконується в обох режимах програмування. Під час її стираються всі комірки FLASH і EEPROM пам'яті, а також LOCK-біти. Причому LOCK-біти стираються тільки після того, як буде очищена вся пам'ять програм. На стан FUSE біт операція "Chip erase" не робить впливу.

FLASH і EEPROM блоки пам'яті програмуються байт за байтом у кожному з режимів програмування.

Для EEPROM пам'яті в режимі послідовного програмування автоматично забезпечується цикл стирання. Таким чином, існує можливість побайтного перезапису окремих комірок EEPROM. Якщо ж потрібно змінити зміст яких-небудь комірок FLASH пам'яті, то необхідно виконувати операцію "Chip erase", що у всіх кристалів стирає не тільки цілком усю FLASH, але і вміст EEPROM.

LOCK-біти (LB1, LB2) призначені для захисту програмної інформації, що міститься в FLASH-пам'яті. Можливі режими захисту перераховані в табл. 5.2. Запрограмувавши біти захисту, стерти їх можна лише під час очищення FLASH-пам'яті (операція "Chip erase"), що знищує і всю програму.

Таблиця 5.2 - Режими захисту програмної інформації AVR

Режим	LB1	LB2	
1	1	1	Захист відсутній
2	0	1	Заборона програмування Flash
3	0	0	Заборона як програмування, так і читання Flash

FUSE-біти дозволяють задавати деякі конфігураційні особливості мікроконтролера. Склад FUSE біт кожного конкретного типу AVR обумовлений особливостями побудови вузлів скидання, тактування і програмування кристала.

Об'єктний файл, створюваний програмою WAVRASM, використовується надалі як вхідний для програми-відладника AVRSTUDI і має спеціальний формат.

2. Створення програм

Популярність мікроконтролерів AVR сприяла тому, що багато фірм-виробників програмних засобів підтримки мікроконтролерів (ассемблерів, компіляторів, відладників) створили програмні пакети підтримки AVR.

AVR Studio - це інтегроване відладне середовище розробки додатків (IDE) для мікроконтролерів сімейства AVR (AT90S, ATmega, ATtiny) фірми Atmel.

IDE AVR Studio містить:

- транслятор мови асемблера (Atmel AVR macroassembler);
- відладник (Debugger);
- програмне забезпечення верхнього рівня для підтримки внутрисхемного програмування (In-System Programming, ISP).

Відладник AVR Studio підтримує всі типи мікроконтролерів AVR і має два режими роботи: режим програмної симуляції і режим керування різними типами внутрисхемних емуляторів (In-Circuit Emulators) виробництва фірми Atmel. Важливо відзначити, що інтерфейс користувача не змінюється в залежності від обраного режиму налагодження.

Відладочне середовище підтримує виконання програм як у виді асемблерного тексту, так і у виді вихідного тексту мови С.

Відладчик AVR Studio по формату об’єктного файлу поєднаний з асемблерами фірм Atmel (AVR Assembler) і IAR Systems (EWA90-Assembler - ftp://www.atmel.com/pub/atmel/ewa90-a.zip).

По формату об’єктного файлу ubrof AVR Studio поєднано з компилятором С фірми IAR Systems (ICCA90 C Compiler - www.iar.com), по формату coff - з компиляторами С фірм Imagecraft (ICCAVR і ICCtiny - http://www.imagecraft.com/software/mdex.htmh) і HP Infotech (Code Vision AVR - http://infotech.ir.ro).

AVR Studio поширюється вільно, його остання версія завжди доступна на сайті фірми Atmel (http://www.atmel.cnm/atmel/products/ prod203.htmy).

3. Створення і трансляція проекту

Після запуску WAVRASM для створення нового проекту необхідно в меню FILE вибрати команду New. У результаті на екрані з'являється

діалогове вікно (рис. 5.1). Новий проект зручніше створювати в окремій папці.

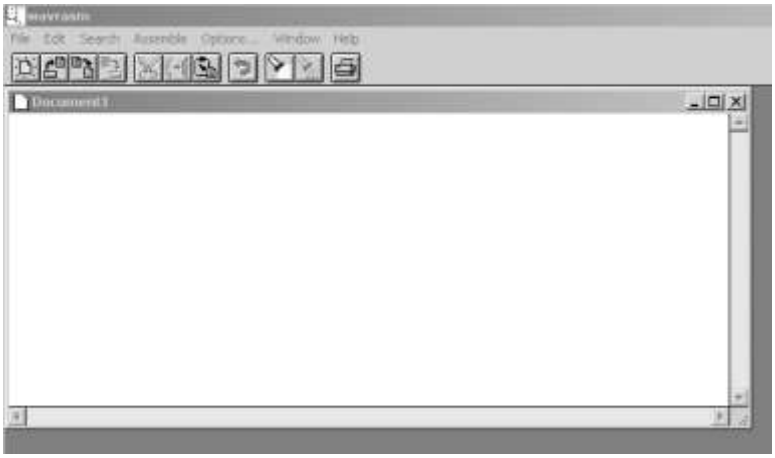


Рисунок 5.1 – Вікно створення нового проекту

Далі до проекту необхідно добавити файл програми мовою асемблера. Це можна зробити різними способами: чи в проект додається вже існуючий файл із розширенням .asm, чи створюється новий. Для цього в вікні проекту необхідно набрати текст програми. Створиться файл з розширенням .asm.

Для редагування вихідного тексту програми необхідно у вікні організації проекту відкрити потрібний файл із розширенням. asm. У вікні, що відкрилося, для редагування файлу можна з клавіатури чи через буфер комп'ютера ввести текст програми мовою асемблера (рис. 5.2).

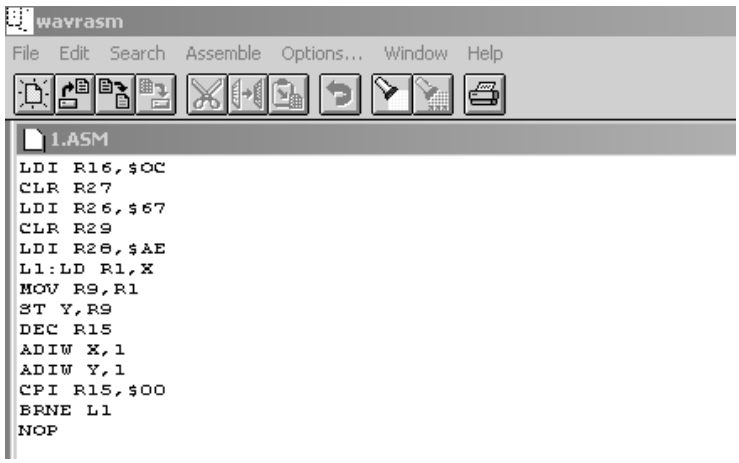


Рисунок 5.2 – Редагування файлу 1.asm

Перед трансляцією потрібно задати установки проекту. У пункті меню **Options**, що відкрилося, вказується необхідний формат вихідного файлу. *AVR Studio* підтримує наступні вихідні формати: Object; Generic; Intel Intellec 8/MDS (Intel Hex); Motorola S-Record.

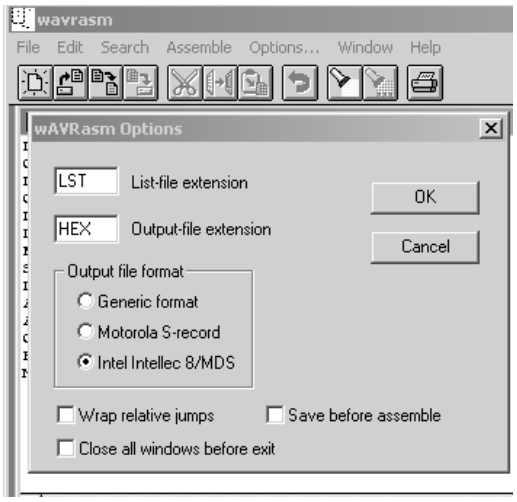


Рисунок 5.3 – Вікно настройки опцій трансляції

Для налагодження в AVR Studio необхідний файл у форматі Object (об'єктний файл). Однак більшість програматорів у якості вхідних використовують файли у форматі Intel Hex.

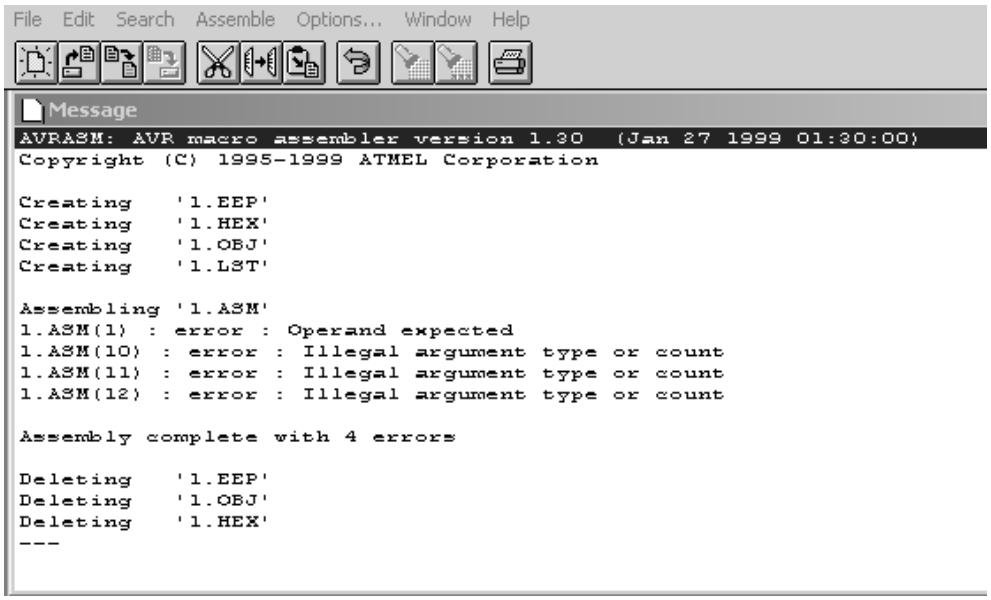
Далі здійснюється трансляція програми і перевірка правильності її написання. Вибирається пункт **Assemble**. Вікно, що відкрилося, містить повідомлення асемблера. У це вікно виводиться інформація про кількість слів коду і даних, про наявність помилок, і інша інформація (рис. 5.4). Для локалізації помилок трансляції у випадку їхньої наявності можна у вікні повідомлень асемблера встановити курсор миші на повідомлення про помилку і два рази клацнути лівою кнопкою миші. При цьому у вікні редагування вихідного тексту програми курсор буде встановлений на рядок, що викликав повідомлення про помилку, і цей рядок буде виділена кольором.

У результаті трансляції створюється вихідний файл у зазначеному форматі. Якщо вихідний асемблерный текст містив сегмент енергонезалежних даних (оголошеною директивою .eseg), то при трансляції буде створений також файл із расширением. *вір*. Цей файл містить дані для внутрішньої EEPROM мікроконтролера і має той же формат, що і вихідний файл.

Якщо в результаті трансляції не видається повідомлень про помилки, можна приступати до налагодження проекту.

4. Режими роботи відладника

AVR Studio дозволяє робити налагодження програм з використанням вбудованого програмного симулятора чи зовнішнього внутрисхемного емулятора. Коли користувач запускає AVR Studio, програма перевіряє наявність емулятора на одному з послідовних портів комп'ютера. Якщо емулятор знайдений, то він вибирається як базова система налагодження. Якщо емулятор не знайдений, то налагодження буде виконуватися на вбудованому програмному симуляторі AVR.



Рисуок 5.4 – Вікно повідомлення асемблера

Налагодження проекту за допомогою програмного симулятора.

Для сумісності з колишніми версіями в AVR Studio передбачений варіант запуску відладника - завантаження отриманого в результаті трансляції об'єктного файлу (File -> **Open**). Але при цьому користувач не має можливості редагувати вихідний текст програми безпосередньо в відладнику. При першому запуску відладника викликається вікно вибору мікроконтролера (рис. 5.5).

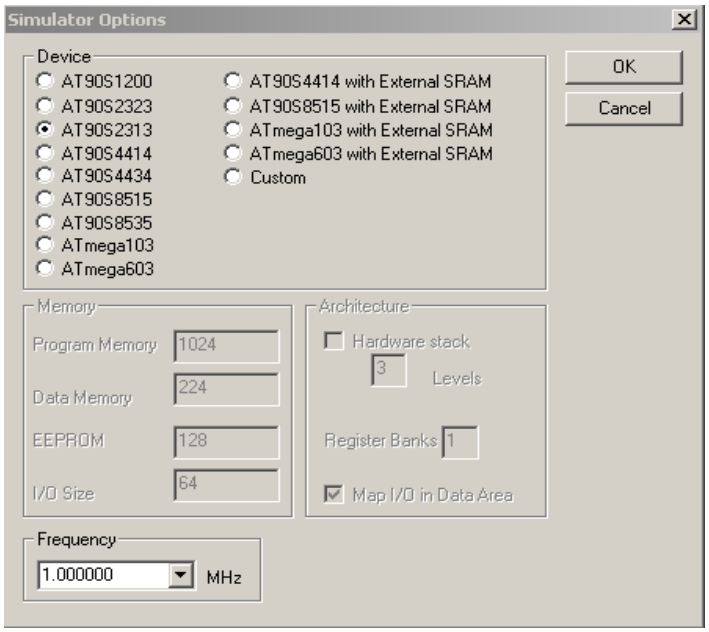


Рисунок 5.5.– Вікно вибору мікроконтролера

Екран AVR Studio у режимі налагодження представлений на рис.5.6.

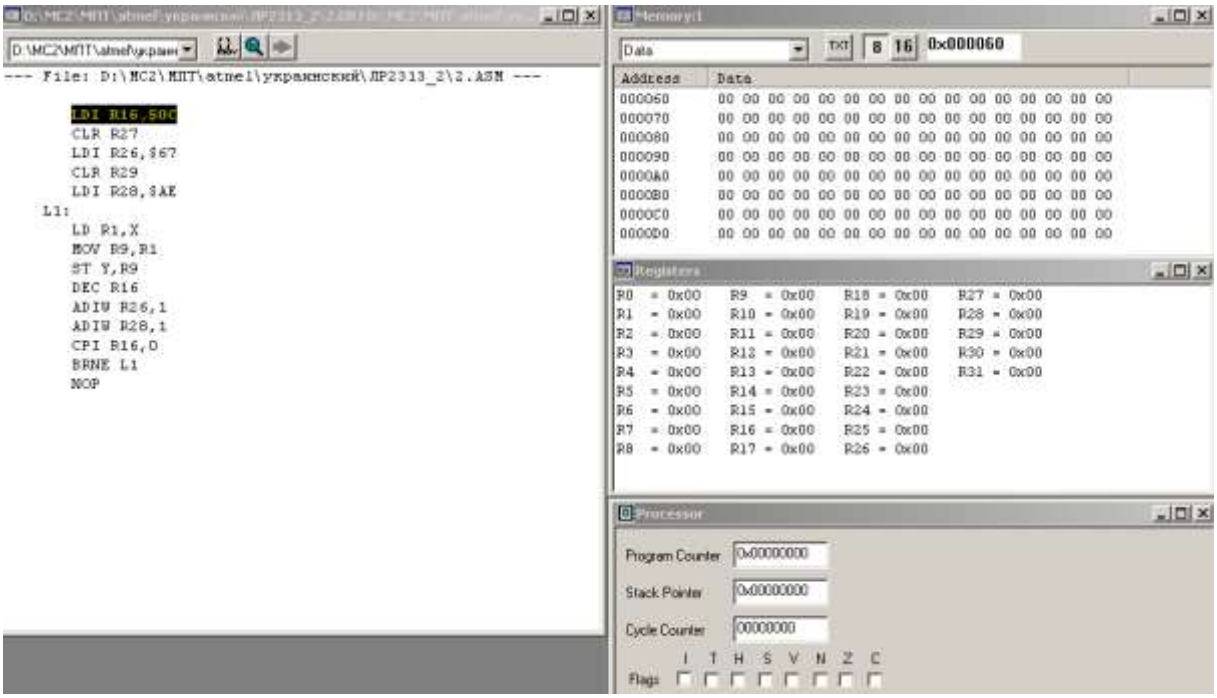


Рисунок 5.6. – Екран AVR Studio у режимі налагодження

При завантаженні об’єктного файлу автоматично відкривається вікно вихідного тексту програми яка виконується мікроконтролером . У вікні відображається код, який виконується в відладниковому оточенні (емуляторі чи програмному симуляторі). Після вибору опцій симулятора в лівому полі вікна асемблерної програми з'являється жовта стрілка, що вказує позицію програмного лічильника мікроконтролера (рис 5.7). Цей показник завжди знаходиться на рядку, що буде виконане в наступному циклі.

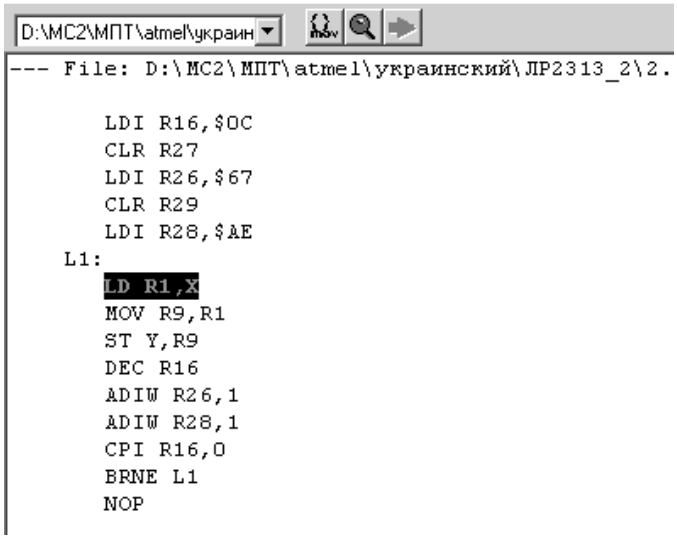


Рисунок 5.7 – Вікно вихідного тексту програми в режимі налагодження

Користувач може виконувати програму цілком у покроковому режимі, трасуючи блоки функцій, виконуючи програму до того місця, де стоїть курсор. У AVR Studio для налагодження програми передбачені дві команди покрокового режиму: **Step Over і Trace into**. Різниця між ними в

тім, що команда **Step Over** не працює в підпрограмах. За допомогою команд покрокового режиму можна простежити зміни значень у регістрах пристроїв вводу/виводу, пам'яті і файлу регистрів. Крім покрокового режиму, можливе налагодження програми з використанням крапок зупинки (**Breakpoints**). Командою Go запускається виконання програми. Програма буде виконуватися до зупинки користувачем чи до виявлення крапки останова.

Для установки крапки останову в AVR Studio є пункт меню **Breakpoints -> Toggle Breakpoint**. Крапка зупинки ставиться в рядку, відзначеної курсором (рис. 5.8).

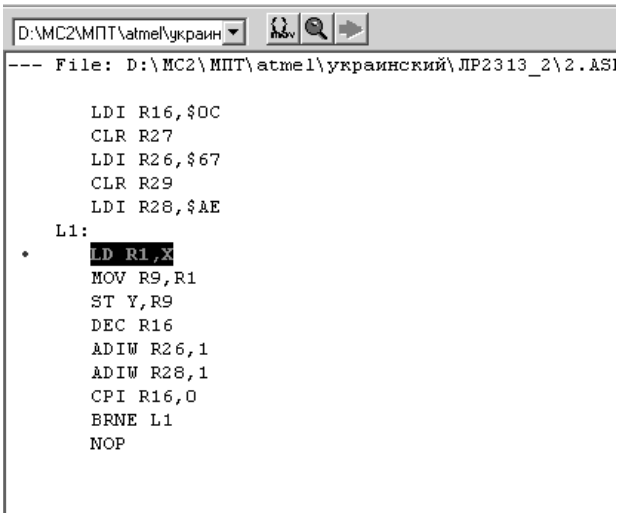


Рисунок 5.8 – Крапка зупинки у вікні вихідного тексту програми в режимі налагодження

Червона помітка в лівому полі вікна вихідного тексту програми показує встановлену крапку зупинки.

У процесі налагодження також можна вибрати пункт меню **Debug -> Run To Cursor**. При виборі цього пункту код, що виконується, буде виконується до досягнення команди, позначеної курсором. При цьому, якщо відладник виявляє крапку останова, установлену раніше положення курсору, то останов буде виконаний тільки у випадку його дозволу у вікні **Debug Option**, в іншому випадку виконання не припиняється. Якщо команда, позначена курсором, не досягається, відладник продовжує виконувати код програми доти , поки виконання не буде перервано користувачем. Оскільки режим **Run To Cursor** залежить від позиції курсору, він доступний тільки при активному вікні вихідного тексту.

Для зупинки виконання програми користувачем служить команда **Break**. У стані зупинка ця команда недоступна. При налагодженні з використанням крапок зупинки, чи якщо адреса зупинки зазначена курсором у вікні вихідного тексту, модифікація інформації у всіх вікнах відбувається тільки при досягненні зупинки (чи при припиненні виконання програми користувачем).

Пункт меню **Debug -> Reset** виконує скидання мікроконтролера Якщо програма при цьому виконується, то її виконання буде зупинено Після скидання інформація у всіх вікнах модифікується. Для спостереження за роботою програми можна відкрити кілька вікон, що відображають стан різних вузлів мікроконтролера Вікна відкриваються натисканням відповідних кнопок на панелі чи інструментів при виборі відповідного пункту меню **View.**

Файл регистрів мікроконтролера AVR відображається у вікні **Registers** (рис. 5.9) Якщо в процесі виконання програми в черговому циклі значення якого-небудь регістра зміниться, то цей регістр буде виділений червоним кольором. При цьому, якщо в наступному циклі значення регістра залишиться незмінним, то кольорове виділення буде зняте. Тако ж кольорове виділення реалізоване у вікнах пристроїв вводу/виводу, пам'яті і змінних.

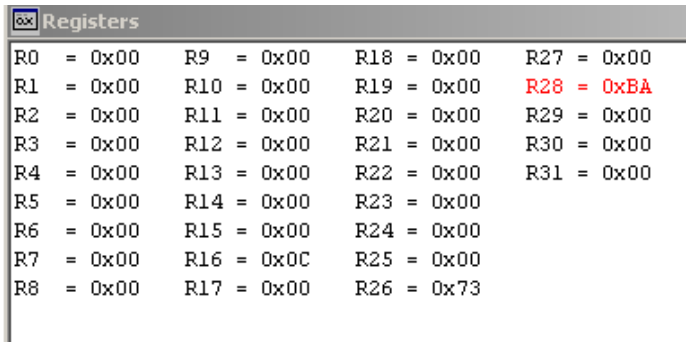


Рисунок 5.9 – Вікно стану регистрів

Будь-який блок може бути розкритий натисканням на його значок. При розкритті блоку у вікні відображаються адреси і стани всіх його регістрів і окремих, доступних для модифікації бітів (рис. 5.10). Кожен доступний для модифікації біт може бути встановлений чи скинутий як програмою по ходу її виконання, так і користувачем вручну (указавши курсором миші потрібний біт і клацнувши лівою кнопкою миші, користувач може змінити значення біта на зворотнє) - у режимі програмної симуляції це є способом імітації вхідного впливу на мікроконтролер.

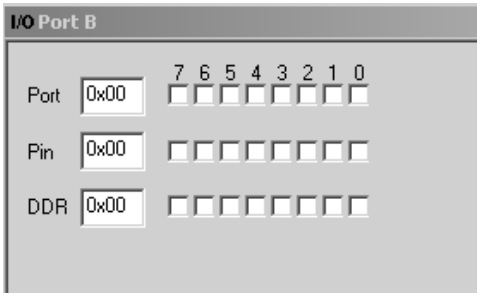


Рисунок 5.10 – Розгорнутий порт PORTB у вікні пристроїв введення/виведення

Іншим способом завдання вхідного впливу на мікроконтролер у режимі симулятора є використання зовнішніх файлів вхідних впливів. Формат файлу вхідного впливу дуже простий

000000000 00
000000039 01
000000040 00 9999999999 FF.

Тут значення, зазначене після роздільника " ", - це шістнадцятирічне представлення сигналів, що впливають на порт мікроконтролера. Значення, зазначене до роздільника, - це десятковий номер циклу (з моменту скидання мікроконтролера), у якому зазначений вплив надходить на виводи порту мікроконтролера Файл вхідного впливу повинен закінчуватися рядком зі свідомо великим номером циклу, в іншому випадку буде видане повідомлення про помилку. Для підключення файлу вхідного впливу служить пункт меню **Options -> Simulator Port Stimuli**. У вікні, що відкрилося, потрібно вказати порт мікроконтролера, на який потрібно подавати вплив, і файл цього впливу. Користувач може створювати файли впливів, записувати зміни значень на виходах портів мікроконтролера у файл (формат цього файлу той же, що й у файлу вхідних впливів). Для запису служить пункт меню **Options -> Simulator Port Logging.** У вікні, що відкрилося, потрібно вказати порт мікроконтролера та ім'я файлу для запису. Записуваний файл буде віддалятися і створюватися знову при кожнім виконанні скидання мікроконтролера (**Debug -> Reset**). Підключати файл вхідного впливу чи задавати ім'я файлу для запису користувач повинен сам при кожному запуску симулятора.

Для індикації стану програмного лічильника, показника стека, вмісту регістра статусу SREG і індексних регістрів X, Y і Z у процесі налагодження програми призначене вікно **Processor** (рис. 5.11).

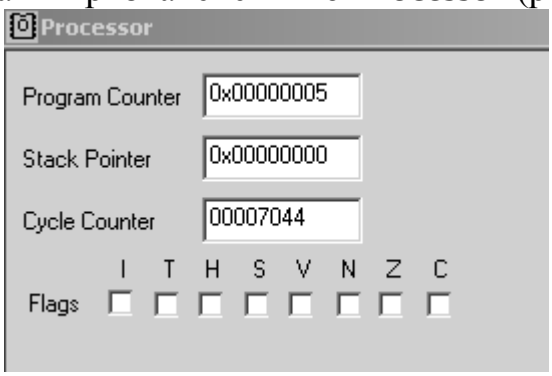


Рисунок 5.11 – Вікно стану процесорного ядра

Перегляд комірок пам'яті програм, пам'яті даних, EEPROM і регістрів портів вводу/виводу в ході виконання програми можливо також за допомогою діалогового вікна **Memory**. Падаюче меню діалогового вікна дозволяє вибрати один з чотирьох масивів комірок пам'яті: *Data, 1/0, EEPROM, Program Memory*. Для одночасного перегляду декількох областей вікно **Memory** може бути відкрито кілька разів. Інформація в діалоговому вікні може бути представлена у виді байтів чи у вигляді слів у шістнадцятирічній системі числення, а також у вигляді ASCI-символів.

У процесі налагодження користувач може ініціалізувати внутрішнє

ПЗП чи EEPROM мікроконтролера (наприклад, даними, що містяться в отриманому при трансляції файлі *.вір*) чи зберегти вміст ПЗП і EEPROM у виді файлів у форматі Intel Hex. Для цього служить пункт меню **File -> Up/Download Memory.**

Для внесення змін у програму в процесі налагодження необхідно редагувати її вихідний текст. При спробі запуску симулятора на виконання програми після редагування на екрані з'являється вікно, що повідомляє про зміну програми і необхідності її компіляції.

При закритті проекту зберігаються всі його настройки. Під час наступного завантаження настройки будуть автоматично відновлені.

Працюючи з програмним симулятором пакета AVR Studio, варто знати, що він поки не підтримує деякі режими роботи мікроконтролерів AVR і їхні периферійні вузли:

- аналого-цифровий перетворювач;
- аналоговий компаратор;
- режим годин реального часу;
- режим зниженого енергоспоживання (інструкція "*sleep*" інтерпретується програмним симулятором як "*nop*");

Завдання до лабораторної роботи

Створити проект з ім’ям група_N.obj, де N – номер варіанту за списком у журналі. В проект занести програму на мові Asembler з лабораторної роботи №2. Виконати симуляцію роботи мікроконтролера AT90S2313 в AVR STUDIO. Перевірити вірність виконання програми. При необхідності внесіть корекцію в програму.

Зміст звіту

Текст програми з поясненнями.
Зміст Program memory в 16 річних кодах.
Зміст даних DATA (напочаток і по закінченні програми).
Зміст REGISTERS (напочаток і по закінченні програми).
Зміст FLAGC (напочаток і по закінченні програми).

Контрольні запитання

1. Які Ви знаєте програмні засоби для відлагодження програмного забезпечення?
2. Перерахуйте операції для стирання, читання і запису різних елементів енергонезалежної пам'яті кристала мікроконтролера.
3. Які Ви знаєте способи програмування мікроконтролерів? Наведіть їхні переваги та недоліки.
4. Як створити проект за допомогою програмного симулятора?
5. Які вихідні формати підтримує AVR Studio?

Лабораторна робота №6

АРИФМЕТИКО-ЛОГІЧНІ ОПЕРАЦІЇ

Мета роботи: придбання навичок програмування арифметико – логічних операцій у кодах мікроконтролера.

Теоретичні відомості

1. Арифметико-логічні операції

Команди даної групи дозволяють виконувати такі операції над 8-бітними цілими двійковими числами: додавання, додавання з урахуванням переносу, десяткову корекцію, інкремент і декремент, віднімання, множення, ділення, диз'юнкцію, кон’юнкцію, виключне або, інверсію, скидання, зсув. Опис команд приведений у табл. 6.1. Ознака паритету P змінюється будь-якими командами, результат яких змінює акумулятор (включаючи команди пересилання).

Таблиця 6.1 - Група команд арифметичних операцій

Мнемо-ніка	Операнди	Опис	Операція	Прапори	Цикли
ADD	Rd,Rr	Підсумовування без переносу	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Підсумовування з переносом	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
SUB	Rd,Rr	Вирахування без переносу	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Вирахування константи	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Вирахування з переносом	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Вирахування константи з переносом	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Логічне И	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Логічне И с константою	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Логічне АБО	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Логічне АБО з константою	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Логічне що виключає АБО	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	Побитна Інверсія	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Зміна знака (Доп. код)	Rd = $00 - Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	Установити біт (біти) у регістрі	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Скинути біт (біти) у регістрі	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Інкрементувати значення регістра	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	Декрементувати значення регістра	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Перевірка на нуль або заперечність	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Очистити регістр	Rd = 0	Z,C,N,V,S	1
SER	Rd	Установити регістр	Rd = $FF	None	1
ADIW	Rdl,K6	Скласти константу і слово	Rdh:Rdl = Rdh:Rdl + K6	Z,C,N,V,S	2
SBIW	Rdl,K6	Вичитати константу зі слова	Rdh:Rdl = Rdh:Rdl - K 6	Z,C,N,V,S	2

2. Виконання арифметичних операцій з цілими числами

Множення беззнакових цілих восьмирозрядних чисел (8x8=16).

Алгоритм множення беззнакових цілих чисел реалізує таку послідовність дій (рис. 6.1):

1. Очистити старший байт результату.
2. Завантажити в лічильник циклу число повторень 8.
3. Множник зрушити на один розряд вправо з використанням біта переносу C.
4. Якщо перенос C установлений, додати множник до результату.
5. Старший байт результату змістити вправо з використанням біта переносу C.
6. Молодший байт результату/множник змістити вправо з використанням біта переносу C.
7. Зменшити на одиницю лічильник циклу.
8. Якщо лічильник циклу ще не дорівнює нулю, то перейти до кроку 4.

Ділення беззнакових цілих восьмирозрядних чисел (8/8=8+8).

Алгоритм підпрограми ділення цілих беззнакових чисел (рис. 6.2) можна представити у виді послідовності таких кроків:

1. Очистити залишок і перенос.
2. Завантажити в лічильник циклу число 9.
3. Ділене змістити вліво з використанням переносу.
4. Зменшити на 1 лічильник циклу.
5. Якщо лічильник циклу дорівнює 0, то вийти з підпрограми.
6. Залишок змістити вліво з використанням переносу.
7. Відняти дільник із залишку.
8. Якщо залишок негативний, додати назад дільник, скинути перенос і перейти до кроку 3.
9. Установити перенос і перейти до кроку 3.

Текст програми mpy8u_c множення 8-и розрядних цілих беззнакових чисел

```
;*****Використання регістрів
.def        mc8u   =r16          ;множене
.def        mp8u   =r17          ;множник
.def        m8u    =r17          ;молодший байт результату
.def        m8u    =r18          ;старший байт результату
.def        mcnt8u=r19           ;лічильник циклу
;*****
mpy8u_c:  clr      m8u                ;очистити старший байт результату
          ldi      mcnt8u,8           ;ініціалізувати лічильник циклу
          lsr      mp8u               ;зрушити вправо множник
m8u_1:    brcc     m8u_2              ;перехід, якщо C=0
          add      m8u,mc8u           ;додати множене до старшого байту
результату
m8u_2:    ror      m8u                ;зрушити вправо старший байт результату
```

```
        ror     m8u                 ;зрушити вправо молодший
                                    ;байт результату і множник
        dec     mcnt8u      ;зменшити на 1 лічильник циклу
        brne    m8u_1       ;перехід, якщо лічильник циклу ще не
дорівнює 0
        ret
```

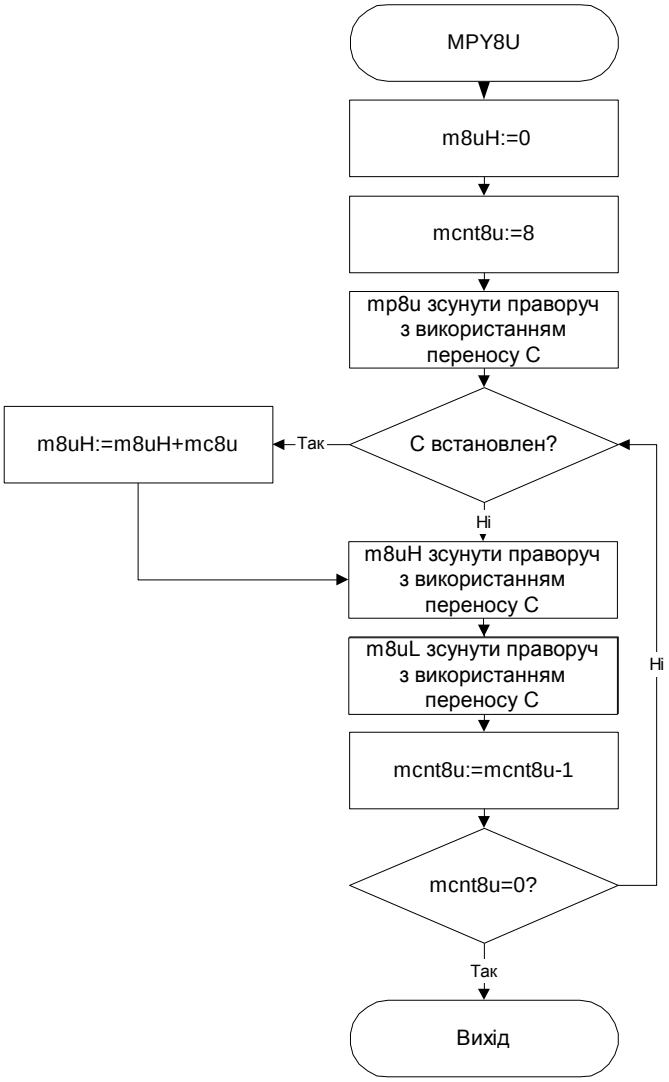


Рисунок 6.1 – Алгоритм програми “mpy8u_c” множення 8-ми розрядних цілих беззнакових чисел

Текст програми div8u_c ділення 8-и розрядних цілих беззнакових чисел

```
;***** Використання регістрів
.def    drem8u=r15      ;залишок
.def    dres8u =r16     ;результат
.def    dd8u   =r16     ;ділене
.def    dv8u   =r17     ;дільник
.def    dcnt8u =r18     ;лічильник циклу
;*****
        sub     drem8u,drem8u       ;очистити залишок і перенос
```

```
            ldi     dcnt8u,9        ;ініціалізувати лічильник циклу
d8u_1:      rol     dd8u            ;ділене/результат зрушити вліво
            dec     dcnt8u          ;зменьшити на одиницю лічильник циклу
            brne    d8u_2           ;перехід, якщо не нуль
            ret                     ;вихід з підпрограми
d8u_2:      rol     drem8u          ;залишок зрушити вліво
            sub     drem8u,dv8u     ;залишок= залишок - дільник
            brcc    d8u_3           ;якщо результат < 0
            add     drem8u,dv8u     ;відновити залишок
            clc                     ;скинути перенос для формування результату
            rjmp    d8u_1           ;інакше
d8u_3:      sec                     ; установити перенос для формування
результату
            rjmp    d8u_1           ;повернутися назад
```

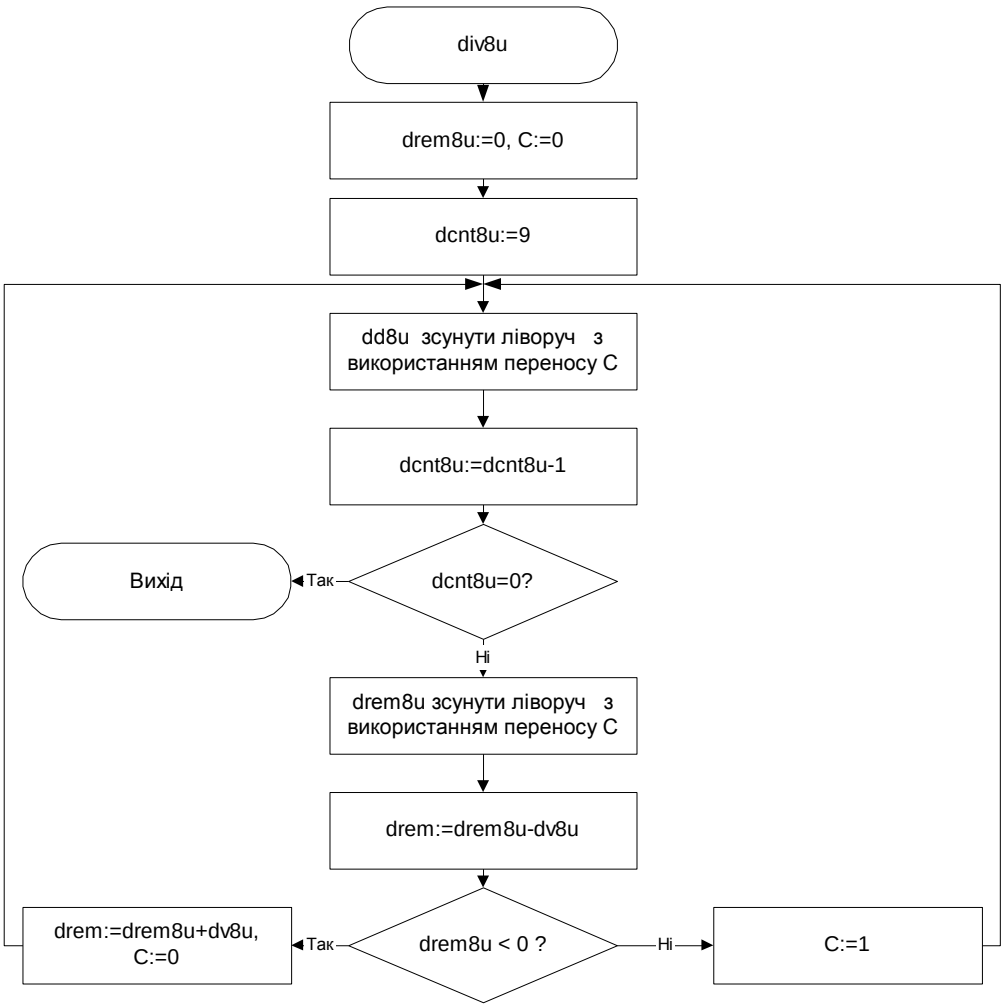


Рисунок 6.2 – Алгоритм програми "div8u_з" розподілу цілих беззнакових восьмирозрядних чисел

Завдання до лабораторної роботи

Нехай у пам'яті програми, починаючи з комірки ADR1 SRAM, розташована таблиця кодів довжиною (X1, i = 1, 2, ..., N, формат – байт).

Записати в кодах AT90S2313 програму, що виконує обчислення заданої функції F над цими кодами. Результат обчислення розмістити в комірку з ADR2. Варіанти завдань приведені в табл. 6.2.

Таблиця 6.2 - Варіанти завдань

Номер	ADR1	ADR2	N	Функция F
01	C3	6F	E	Сумма (Xi)/N
02	C4	63	F	Max(Xi)/N
03	C2	75	D	Min(Xi)/N
04	C1	86	C	Сумма (Xi)\/X1
05	C2	68	B	Max(Xi)\/X1
06	C3	6D	A	Min (Xi)\/X1
07	A4	6F	6	Сумма (Xi)/\X1
08	A5	7A	7	Max(Xi)/\X1
09	A6	7C	F	Min (Xi)/\X1
10	A7	74	E	Сумма (Xi)/\X2
11	A8	73	5	Max(Xi)\/X2
12	B3	78	6	Min (Xi)/\X2
13	BA	8C	7	Сумма (Xi)/\X3
14	BB	6C	8	Max(Xi)/\X3
15	BC	6E	8	Min (Xi)/\X3
16	6D	8A	9	Сумма (Xi)/\X4
17	7E	6A	F	Max(Xi)\/X4
18	AF	6B	A	Min (Xi)\/X4
19	BE	9B	B	Сумма (Xi)⊕X1
20	CD	7B	C	Max(Xi)⊕X1
21	88	69	D	Min (Xi)⊕X1
22	63	87	E	Сумма (Xi)/\Xn
23	6A	85	F	Max(Xi)/\Xn
24	6B	83	8	Min (Xi)/\Xn
25	6C	AD	8	Сумма (Xi)\/X1n
26	62	A5	9	Max(Xi)\/Xn
27	7E	AE	C	Min (Xi)\/Xn

Коди даних задати довільно. Виконати контрольний перерахунок.

Зміст звіту

Таблиця арифметико-логічних команд.
Завдання до лабораторної роботи.
Текст програми з поясненням і контрольним перерахунком.
Виконати роботу в AVR STUDIO.

Контрольні запитання

1. Перерахуйте основні операції над 8-ми бітними цілими двійковими числами.
2. Перерахуйте послідовність операцій, які потрібно виконати для множення беззнакових цілих чисел.
3. Перерахуйте послідовність операцій, які потрібно виконати для ділення беззнакових цілих чисел.

Лабораторна робота №7

БІТОВІ ОПЕРАЦІЇ

Мета роботи: вивчення організації бітового простору пам'яті мікроконтролера AT90S2313, програмних засобів обробки бітів і можливостей програмування бітових операцій у кодах мікроконтролера.

Теоретичні відомості

Всі арифметичні і логічні операції, а також частина операцій роботи з бітами виконуються в ALU тільки над вмістом оперативних регістрів. Варто звернути увагу, що команди (SUBI, SBCI, ANDI, ORI, SBR, CBR) у якості другого операнда мають константу, можуть використовуватись в якості першого операнда тільки регістри з другої половини регістрового файлу (R16-R31). Команди 16-розрядного додавання з константою ADI і віднімання константи SBI у якості першого операнда використовують тільки регістри R24, R26, R28, R30. Під час виконання арифметичних і логічних операцій чи операцій роботи з бітами ALU формує ті чи інші (див. табл. 2.1) ознаки результату операції, тобто встановлює чи скидає біти в регістрі стану SREG (Status Register). Регістр статусу SREG розміщений у просторі I/O за адресою $3F ($5F). Його біти визначаються так:

Bit 7 - I: Global Interrupt Enable - Дозвіл глобального переривання. Біт дозволу глобального переривання для дозволу переривання повинний бути встановлений у стан 1. Керування дозволом конкретного переривання виконується регістрами маски переривання GIMSK і TIMSK. Якщо біт глобального переривання очищений (у стані 0), то жодне з дозволів конкретних переривань, встановлених у регістрах GIMSK і TIMSK, не діє. Біт I апаратно очищається після переривання і встановлюється для наступного дозволу глобального переривання командою RETI.

Bit 6 - T: Bit Copy Storage - Біт збереження копії. Команди копіювання біта BLD (Bit Loa) і BST (Bit STore) використовують біт T як біт джерело і біт призначення при операціях з бітами. Командою BST біт регістра реєстрового файлу копіюється в біт T, командою BLD біт T копіюється в регістр реєстрового файлу.

Bit 5 - H: Half Carry Flag - Прапор напівпереносу. Прапор напівпереносу вказує на напівперенос у ряді арифметичних операцій

Bit 4 - S: Sign Bit, S = N V - Біт знака. Біт S завжди знаходиться в стані, обумовленому логічним що виключає ЧИ (exclusive OR) між прапором негативного значення N і доповненням до двох прапора переповнення V.

Bit 3 - V: Two's Complement Overflow Flag. Доповнення до двох прапора переповнення. Доповнення до двох прапора V підтримує арифметику доповнення до двох.

Bit 2 - N: Negative Flag - Прапор негативного значення. Прапор негативного значення N вказує на негативний результат ряду арифметичних і логічних операцій.

Bit 1 - Z: Zero Flag -Прапор нульового значення. Прапор нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій.

Bit 0 - C: Carry Flag -Прапор переносу. Ознаки результату операції можуть бути використані в програмі для виконання подальших арифметично-логічних операцій чи команд умовних переходів.

Виконувати арифметично-логічні операції (arithmetically boolean operations) і операції зсуву (operations of change) безпосередньо над змістом комірок пам'яті не можна. Не можна також записати константу чи очистити вміст комірки пам'яті. Система команд AVR дозволяє лише виконувати операції обміну даними між осередками SRAM і оперативними регістрами. Перевагами системи команд можна вважати різноманітні режими адресації комірок пам'яті.

Усі регістри введення/виведення можуть зчитуватися і записуватися через оперативні регістри за допомогою команд IN, OUT (див. групу команд передачі даних). Регістри введення/виведення, що мають адреси в діапазоні $00 - $1F, мають можливість побітової адресації. Безпосередня установка і скидання окремих розрядів цих регістрів виконується командами SBI і CBI (див. групу команд роботи з бітами). Для ознак результату операції, що є бітами регістра введення/виведення SREG, мається цілий набір команд установки і скидання. Команди умовних переходів у якості своїх операндів можуть мати як біти-ознаки результату операції, так і окремі розряди побітно адресуємих регістрів введення/виведення (табл. 7.1).

Таблиця 7.1 - Інструкції роботи з бітами

Мнемоніка	Операнди	Опис	Операція	Прапори	Цикли
LSL	Rd	Логічний зсув вліво	Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	Логічне зрушення вправо	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Циклічне зрушення вліво через C	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	Z,C,N,V,H,S	1
ROR	Rd	Циклічне зрушення вправо через C	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Арифметичне зрушення вправо	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Перестановка тетрад	Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0)	None	1
BSET	s	Установка прапора	SREG(s) = 1	SREG(s)	1
BCLR	s	Очищення прапора	SREG(s) = 0	SREG(s)	1
SBI	P,b	Установити біт у порту	I/O(P,b) = 1	None	2
CBI	P,b	Очистити біт у порту	I/O(P,b) = 0	None	2

BST	Rr,b	Зберегти біт з регістра в T	T = Rr(b)	T	1

Продовження таблиці 7.1

BLD	Rd,b	Завантажити біт з T у регістр	Rd(b) = T	None	1
SEC	Hi	Установити прапор переносу	C =1	C	1
CLC	Hi	Очистити прапор переносу	C = 0	C	1
SEN	Hi	Установити прапор негативного числа	N = 1	N	1
CLN	Hi	Очистити прапор негативного числа	N = 0	N	1
SEZ	Hi	Встановити прапор нуля	Z = 1	Z	1
CLZ	Hi	Очистити прапор нуля	Z = 0	Z	1
SEI	Hi	Встановити прапор переривань	I = 1	I	1
CLI	Hi	Очистити прапор переривань	I = 0	I	1
SES	Hi	Установити прапор числа зі знаком	S = 1	S	1
CLN	Hi	Очистити прапор числа зі знаком	S = 0	S	1
SEV	Hi	Установити прапор переповнення	V = 1	V	1
CLV	Hi	Очистити прапор переповнення	V = 0	V	1
SET	Hi	Установити прапор T	T = 1	T	1
CLT	Hi	Очистити прапор T	T = 0	T	1
SEH	Hi	Установити прапор внутрішнього переносу	H = 1	H	1
CLH	Hi	Очистити прапор внутрішнього переносу	H = 0	H	1
NOP	Hi	Немає операції	Hi	None	1
SLEEP	Hi	Спати (зменшити енергоспоживання)	Дивитися опис інструкції	None	1
WDR	Hi	Скидання сторожового таймера	Дивитися опис інструкції	None	1

Завдання до лабораторної роботи

Нехай у SRAM в комірці з ADR1 розташований код CODE. Записати в кодах МК AT90S2313 програму, що виконує обчислення заданої булевої функції F над цими кодами. Результат обчислення повинен бути записаний за адресою ADR2 простору SRAM.

Вигляд функції F для обчислення визначається таким способом: YZ дві останні цифри року народження (залікової книжки).

Код CODE має формат (порозрядно):

X7	X6	X5	X4	X3	X2	X1	X0

F=(X7)OP1(X6)OP2(X5)OP3(X4)OP4(X3)OP5(X2)OP6(X1)OP7(X0),

де OP – булева операція, знаходиться з табл. 7.3, а кожна величина X може входити у вираз прямо або інверсно (визначається за табл. 7.4).

Таблиця 7.2 - Варіанти завдань

НОМЕР	ADR1	ADR2	CODE
01	C3	6F	FE
02	C4	63	3F
03	C2	75	5D
04	C1	86	7C
05	C2	68	8B
06	C3	6D	7A
07	A4	6F	1E
08	A5	7A	F2
09	A6	7C	E3
10	A7	74	FA
11	A8	73	A5
12	B3	78	56
13	BA	8C	F7
14	BB	6C	A8
15	BC	6E	48
16	6D	8A	29
17	7E	6A	F3
18	AF	6B	E2
19	BE	9B	D3
20	CD	7B	A4
21	88	69	B5
22	63	87	3D
23	6A	85	4F
24	6B	83	8F
25	6C	AD	6D
26	62	A5	D9

Таблиця 7.3 – Булеві операції

Z	0	1	2	3	4	5	6	7	9	0
OP1	/\	/\	/\	/\	/\	\/	\/	\/	\/	\/
OP2	⊕	⊕	⊕	/\	/\	/\	\/	\/	\/	\/
OP3	\/	\/	\/	⊕	⊕	⊕	⊕	\/	⊕	/\
OP4	⊕	⊕	\/	/\	\/	/\	/\	/\	\/	\/
OP5	/\	/\	\/	\/	\/	⊕	⊕	⊕	⊕	⊕
OP6	⊕	\/	/\	\/	/\	/\	⊕	⊕	/\	/\
OP7	\/	⊕	⊕	⊕	⊕	⊕	\/	/\	⊕	\/

/\ - операція І

\/ - операція АБО

⊕ - операція виключне АБО (XOR)

Таблиця 7.4 - Значення змінних

Y	0	1	2	3	4	5	6	7	8	9
X0	П	И	П	И	П	И	П	И	П	И
X1	П	П	П	П	И	И	И	И	И	И
X2	П	П	И	И	И	П	П	И	И	П
X3	И	И	И	П	П	П	И	И	И	И
X4	И	И	И	П	И	П	П	И	П	П
X5	П	И	И	П	И	П	П	П	И	И
X6	И	П	П	П	И	И	П	П	П	П

П – змінна в функцію входить прямо (без інверсії), И – змінна в функцію входить інверсно, X7 – входить в вираз для F завжди без інверсії.

Зміст звіту

Таблиця команд роботи з бітами.
Завдання до лабораторної роботи.
Текст програми з поясненнями і контрольним прорахунком.
Результати виконання завдання в AVR STUDIO.

Приклад виконання завдання

Нехай у SRAM в комірці з ADR1=62 розташований код CODE= 2D. Записати в кодах МК AT90S2313 програму, що виконує обчислення заданої булевої функції F=X7∨X6∨X5⊕X4∨X3⊕X2∧X1⊕X0 *над цими кодами. Результат обчислення повинний бути записаний за адресою ADR2=A5 простору SRAM.*

Текст програми

```
LDI R16,$2D        ; записати в R16 число $2D
STS $62,R16        ; зберегти число $2D в SRAM за адресою $62
LDS R18,$62        ; записати в R18 число, яке знаходиться в SRAM за адресою $62
MOV R20,R18        ; переписати зміст R18 в R20
ANDI R20,$0A       ; виконується маскування (виділення) біт X3, X1
COM R20            ; інверсія бітів регистру r20
ANDI R20,$0A       ; виконується маскування (виділення) біт X3, X1
CBR R18,3          ; очищення X3 в регистрі R18
CBR R18,1          ; очищення X1 в регистрі R18
OR R18,R20         ; отримання вхідного числа з проінвертованими бітами X3,X1($2D→$27)
BST R18,7          ; Зберегти біт X7 з регістра R18 в T
BLD R5,0           ; Переписати біт X7 з T в регистр  R5 в біт 0
BST R18,6          ; Зберегти біт X6 з регістра R18 в T
BLD R6,0           ; Переписати біт X6 з T в регистр  R6 в біт 0
OR R5,R6           ; R5 ∨ R6 (X7 ∨ X6)
BST R18,5          ; Зберегти біт X5 з регістра R18 в T
```

```
BLD R6,0        ; Переписати біт X5 з T в регистр  R6 в біт 0
OR R5,R6        ; R5 ∨ R6 (X7 ∨ X6 ∨ X5)
BST R18,4       ; Зберегти біт X4 з регістра R18 в T
BLD R6,0        ; Переписати біт X4 з T в регистр  R6 в біт 0
EOR R5,R6       ; R5 ⊕ R6 (X7 ∨ X6 ∨ X5 ⊕ X4)
BST R18,3       ; Зберегти біт X4 з регістра R18 в T
BLD R6,0        ; Переписати біт X3 з T в регистр  R6 в біт 0
OR R5,R6        ; R5  ∨ R6 (X7 ∨ X6 ∨ X5 ⊕ X4  ∨ X3)
BST R18,2       ; Зберегти біт X3 з регістра R18 в T
BLD R6,0        ; Переписати біт X2 з T в регистр  R6 в біт 0
EOR R5,R6       ; R5  ⊕ R6 (X7 ∨ X6 ∨ X5 ⊕ X4  ∨ X3 ⊕ X2)
BST R18,1       ; Зберегти біт X2 з регістра R18 в T
BLD R6,0        ; Переписати біт X1 з T в регистр  R6 в біт 0
AND R5,R6       ; R5  ∧ R6 (X7 ∨ X6 ∨ X5 ⊕ X4  ∨ X3 ⊕ X2 ∧ X1)
BST R18,0       ;Зберегти біт X1 з регістра R18 в T
BLD R6,0        ; Переписати біт X0 з T в регистр  R6 в біт 0
EOR R5,R6       ; R5  ⊕ R6 (X7 ∨ X6 ∨ X5 ⊕ X4  ∨ X3 ⊕ X2 ∧ X1 ⊕ X0)
STS $A5,R5      ; зберегти результат в комірку пам’яті SRAM  за адресою $A5
```

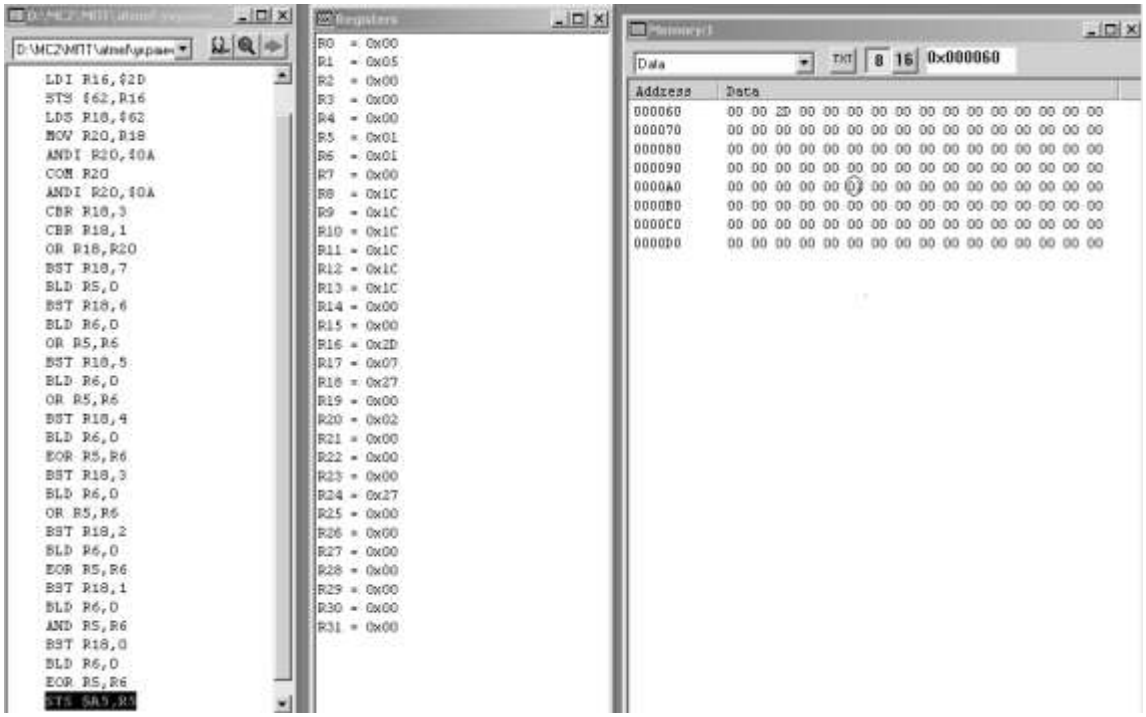


Рисунок 7.1 – Результат виконання програми в AVR STUDIO

Перевірка:
Для числа $2D X7=0, X6=0, X5=1, X4=0, X3=1, X2=1, X1=0, X0=1.

$$F=X7\vee X6\vee X5\oplus X4\vee\overline{X3}\oplus X2/\wedge\overline{X1}\oplus X0 =0 \vee 0 \vee 1 \oplus 0 \vee 0 \oplus 1 \wedge 1 \oplus 1=1$$

Контрольні запитання

1. Які регістри використовуються у якості першого операнда при 16-розрядному додаванні і відніманні з константами?
2. Перерахуйте основні булеві операції та їхні позначення.
3. Розкажіть про організацію бітового простору пам'яті мікроконтролера AT90S2313.

Лабораторна робота №8

ЗВЕРТАННЯ ДО EEPROM ПРИ ЧИТАННІ/ЗАПИСІ

Мета роботи: вивчення організації звертання до EEPROM, отримання навичків програмування EEPROM.

Теоретичні відомості

1. Звертання до EEPROM при читанні/записі

Звертання до EEPROM ведеться за допомогою регістрів розташованих у просторі I/O. Час звертання при записі складає від 2,5 до 4мс, у залежності від напруги VCC.

Опис регістрів адреси EEPROM- EEAR - (EEPROM Address Register) представлено в табл. 8.1.

Таблиця 8.1 – Регістр адреси EEPROM – EEAR

Біти	7 6 5 4 3 2 1 0	
$1E ($3E)	EEAR6 EEAR5 EEAR4 EEAR3 EEAR2 EEAR1 EEAR0	EEAR
Читання/Запис	R R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

Послідовно EEPROM адресний регістр EEAR6...0 адресується до 128 байтів адресного простору (0-127).

Регистр данних EEPROM - EEDR - (EEPROM Data Register) представлено в табл. 8.2.

Таблиця 8.2 - Регистр данних EEPROM - EEDR

Біти	7 6 5 4 3 2 1 0	
$1D ($3D)	MSB LSB	EEDR
Читання/Запис	R/W R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

Bits 7..0 - EEDR7..0: EEPROM Data - Дані EEPROM. У процесі виконання операції запису в регістрі EEDR містяться дані, записувані в EEPROM за адресою, що задається регістром EEAR. При читанні регістр EEDR містить дані, що зчитуються з EEPROM за адресою, обумовленому EEAR.

Таблиця 8.3 - Регістр керування EECR (EEPROM Control Register)

Біти	7 6 5 4 3 2 1 0	
$1C ($3C)	- - - - - EEMWE EEWE EERE	EECR
Читання/Запис	R R R R R R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

Bit 2 - EEMWE: EEPROM Master Write Enable - Керування дозволом запису EEPROM. Біт EEMWE визначає, чи буде встановлений у стан 1 біт EEWE дозволяти запис у EEPROM. При встановленому в стан 1 біті EEMWE установка біта EEWE приведе до запису в EEPROM по заданій адресі. Якщо ж біт EEMWE буде знаходитися в стані 0, то установка біта EEWE ніякого ефекту не зробить. Установлений програмним шляхом біт EEMWE апаратно очищається через чотири тактових цикли. Див. опис процедури запису EEPROM в описі біта EEWE.

Bit 1 - EEWE: EEPROM Write Enable - Дозвіл запису EEPROM.

Bit 0 - EERE: EEPROM Read Enable - Дозвіл читання EEPROM. Сигнал дозволу читання EERE (EEPROM Read Enable Signal) є стробом читання EEPROM. Біт EERE повинний бути встановлений по встановленні в регістрі EEAR необхідної адреси. Після апаратного очищення біта EERE дані, що зчитуються, будуть розташовуватися в регістрі EEDR. Зчитування

байта даних виконується однією командою і не вимагає опитування біта EERE. При встановленому біті EERE CPU зупиняється на чотири тактових цикли, перш ніж почне виконання наступної команди. Користувачу необхідно тестувати стан біта EEWE перед початком операції читання. Якщо нові чи дані адреса будуть записуватися в регістри I/O EEPROM у той час, коли буде виконуватися операція запису, то операція записи буде перерваний і результат запису буде невизначеним.

2. Програмна організація звертання до EEPROM

```
;* This Application note shows how to read data from and write data to the
;* EEPROM. Both random access and sequential access routines are listed.
;*
;***************************************************************************
*
.include "2313def.inc"
        rjmp    RESET           ;Reset Handle
;***************************************************************************
*
;* EEWrite
;* This subroutine waits until the EEPROM is ready to be programmed, then
;* programs the EEPROM with register variable "EEdwr" at address "EEawr"
;* Ця підпрограма очикування, доки EEPROM не буде готова до програмування
;***************************************************************************
*
;***** Subroutine register variables

.def    EEdwr   =r16            ;data byte to write to EEPROM
.def    EEawr   =r17            ;address to write to
;***** Code
EEWrite:
        sbic    EECR,EEWE       ;if EEWE not clear
        rjmp    EEWrite         ;    wait more
        out     EEAR,EEawr      ;output address
        out     EEDR,EEdwr      ;output data
        sbi     EECR,EEWE       ;set EEPROM Write strobe
                                ;This instruction takes 4 clock cycles since
                                ;it halts the CPU for two clock cycles
        ret
;***************************************************************************
*
;* EERead
;*
;* This subroutine waits until the EEPROM is ready to be programmed, then
;* reads the register variable "EEdrd" from address "EEard"
;***************************************************************************
*
;***** Subroutine register variables
.def    EEdrd   =r0             ;result data byte
.def    EEard   =r16            ;address to read from
;***** Code
EERead:
        sbic    EECR,EEWE       ;if EEWE not clear
        rjmp    EERead          ;    wait more
        out     EEDR,EEard      ;output address
        sbi     EECR,EERE       ;set EEPROM Read strobe
                                ;This instruction takes 4 clock cycles since
                                ;it halts the CPU for two clock cycles
        sbi     EECR,EERE       ;set EEPROM Read strobe 2nd time
```

```
                                ;This instruction takes 4 clock cycles since
                                ;it halts the CPU for two clock cycles
        in      EEdrd,EEDR      ;get data
        ret
;*********************************************************************
*
;*
;* EEWrite_seq
;*
;* This subroutine increments the EEPROM address by one and waits until the
;* EEPROM is ready for programming. It then programs the EEPROM with
;* register variable "EEdwr_s".
;*
;*********************************************************************
*
;***** Subroutine register variables
.def    EEwtmp  =r0             ;temporary storage of address
.def    EEdwr_s =r16            ;data to write
;***** Code
EEWrite_seq:
        sbic    EECR,EEWE       ;if EEWE not clear
        rjmp    EEWrite_seq     ;    wait more
        in      EEwtmp,EEAR     ;get address
        inc     EEwtmp          ;increment address
        out     EEAR,EEwtmp     ;output address
        out     EEDR,EEdwr_s    ;output data
        sbi     EECR,EEWE       ;set EEPROM Write strobe
                                ;This instruction takes 4 clock cycles since
                                ;it halts the CPU for two clock cycles
        ret
;*********************************************************************
*
;* EERead_seq
;*
;* This subroutine increments the address stored in EEAR and reads the
;* EEPROM into the register variable "EEdrd_s".
;*
;*********************************************************************
*
;***** Subroutine register variables

.def    EErtmp  =r0             ;temporary storage of address
.def    EEdrd_s =r1             ;result data byte

;***** Code
EERead_seq:
        in      EErtmp,EEAR     ;get address
        inc     EErtmp          ;increment address
        out     EEAR,EErtmp     ;output address
        sbi     EECR,EERE       ;set EEPROM Read strobe
                                ;This instruction takes 4 clock cycles since
                                ;it halts the CPU for two clock cycles
        sbi     EECR,EERE       ;set EEPROM Read strobe 2nd time
                                ;This instruction takes 4 clock cycles since
                                ;it halts the CPU for two clock cycles
        in      EEdrd_s,EEDR    ;get data
        ret
;*********************************************************************
**
;*
;* Test/Example Program
;*
```

```
;***************************************************************************
**
;***** Main Program Register variables
.def    counter =r18
.def    temp    =r19
;***** Code
RESET:
;***** Program a random location
        ldi     EEdwr,$aa
        ldi     EEawr,$10
        rcall   EEWrite         ;store $aa in EEPROM location $10
;***** Read from a random locations
        ldi     EEard,$10
        rcall   EERead          ;read address $10
        out     PORTB,EEdrd     ;output value to Port B
;***** Fill the EEPROM with bit pattern $55,$aa,$55,$aa,...
        ldi     counter,64      ;init loop counter
        ser     temp
        out     EEAR,temp       ;EEAR <- $ff (start address - 1)
loop1:  ldi     EEdwr_s,$55
        rcall   EEWrite_seq     ;program EEPROM with $55
        ldi     EEdwr_s,$aa
        rcall   EEWrite_seq     ;program EEPROM with $aa
        dec     counter         ;decrement counter
        brne    loop1           ;and loop more if not done
;***** Copy 10 first bytes of EEPROM to r2-r11
        ldi     temp,$ff
        out     EEAR,temp       ;EEAR <- $ff (start address - 1)
        ldi     ZL,2            ;Z-pointer points to r2
loop2:  rcall   EERead_seq      ;get EEPROM data
        st      Z,EEdrd_s       ;store to SRAM
        inc     ZL
        cpi     ZL,12           ;reached the end?
        brne    loop2           ;if not, loop more
forever:rjmp    forever         ;eternal loop
```

Завдання до лабораторної роботи

В комірки EEPROM починаючи з ADR1 записати N чисел. Скопіювати N-2 чисел до регистрів, починаючи з R2. Перевірити виконання програми в AVR STUDIO. При наявності лабораторного макету виконати програмування мікроконтролера і перевірити вірність роботи програми (після програмування зчитати дані з EEPROM). Для цього в AVR STUDIO вибрати File – Up/Download Memories).

При написанні програми необхідно замість EECR підставляти $1C, EEAR - $1E, EEDR - $1D, EEWE – 1, EERE – 0. результати виконання програми в AVR STUDIO дивитись Memory –I/O.

Зміст звіту

Завдання до лабораторної роботи.
Алгорітм програми з поясненнями.
Текст программ з поясненнями.
Результати виконання програми в AVR STUDIO.

Таблица 8.4 - Варіанти завдань

Номер	ADR1	N	Коди чисел
01	$10	E	6F
02	$12	C	63
03	$14	D	75
04	$16	B	86
05	$18	A	68
06	$1A	9	6D
07	$1C	8	6F
08	$1E	7	7A
09	$20	6	7C
10	$22	5	74
11	$24	4	73
12	$26	E	78
13	$28	C	8C
14	$2A	D	6C
15	$2C	B	6E
16	$2E	A	8A
17	$30	9	6A
18	$32	8	6B
19	$34	7	9B
20	$36	6	7B
21	$38	5	69
22	$3A	4	87
23	$3C	E	85
24	$3E	C	83
25	$0A	D	AD
26	$0C	B	A5
27	$0E	A	AE

Контрольні запитання

1. Дайте характеристику основних станів регістра адреси EEPROM Address Register (EEAR).
2. Дайте характеристику основних станів регістра даних EEPROM Data Register (EEDR).
3. Дайте характеристику основних станів регістра керування EEPROM Control Register (EECR).

Лабораторна робота №9

СХЕМИ ВІДОБРАЖЕННЯ СТАТИЧНОЇ ТА ДИНАМІЧНОЇ ІНДИКАЦІЇ

Мета роботи: вивчення схем динамічної і статичної індикації. Розробка програм для мікроконтролера AT90S2313 для відображення цифрової інформації на пристроях динамічного і статичного типу, а також на одиничних індикаторах.

Теоретичні відомості

1. Системи відображення інформації

Найпростішими приладами відображення інформації в цифрових пристроях є світлодіоди і цифрові індикатори. Схеми включення найпростіших одиничних індикаторів представлено на рис. 9.1.

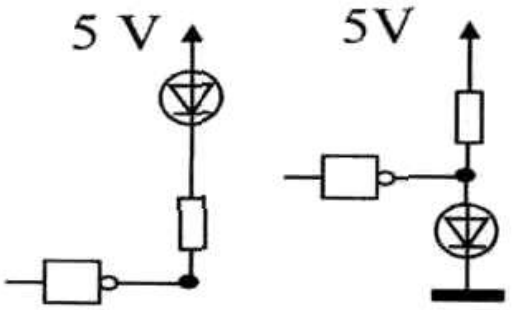


Рисунок 9.1 – Включення одиничних індикаторів

У напівпровідникових світлодіодах використовується властивість p-n переходу випромінювати світло у видимій частині спектра при протіканні через нього прямого струму ($I_{пр}$=5-20мА, $U_{пр}$=2-3В). Варіанти включення індикаторів представлені на рис. 9.1.

Для відображення цифрової інформації найбільше поширення одержали семисегментні індикатори (sevensegment indicators), у яких зображення цифри складають із семи лінійних світлодіодів сегментів розташованих у вигляді цифри 8.

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

Статична індикація (Static indication) полягає в постійному підсвічуванні індикаторів HG2-n від одного джерела інформації рис. 9.2.

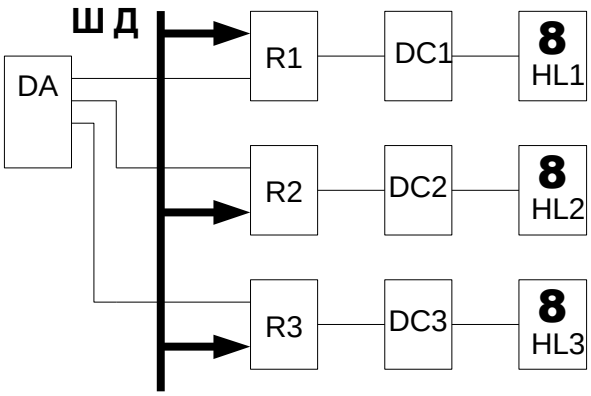


Рисунок 9.2 – Структурна схема статичної індикації

DA – дешифратор (decipherer) адреси, необхідний для вибірки відповідного регістра; R1-R3- регістри, в яких тимчасово зберігається значення коду числа для відображення (відповідний регістр вибирається DA); DC1-DC3 - семисегментні дешифратори, які перетворюють двіковий код у семисегментний код; HG2-HL3 - семисегментні індикатори; ШД - шина даних, по ній здійснюється передача даних на індикацію.

У такій системі кожен індикатор HG2-n підключений через власний дешифратор DCl-n і регістр-засувку RGl-n до шини даних, вибір регістрів RGl-n виробляється за допомогою селектора адреси CA. Апаратні витрати при такій організації складають n пару регістр + дешифратор при n десяткових розрядів індикатора.

Сутність динамічної індикації (dynamic indication) полягає в почерговому циклічному підключенні кожного індикатора HG-n до джерела інформації через загальну шину даних, рис. 9.3.

Вибір індикатора здійснюється дешифратором DA. У регістрі RD зберігається цифровий код призначений для відображення. У регістрі RA зберігається адреса індикатора.

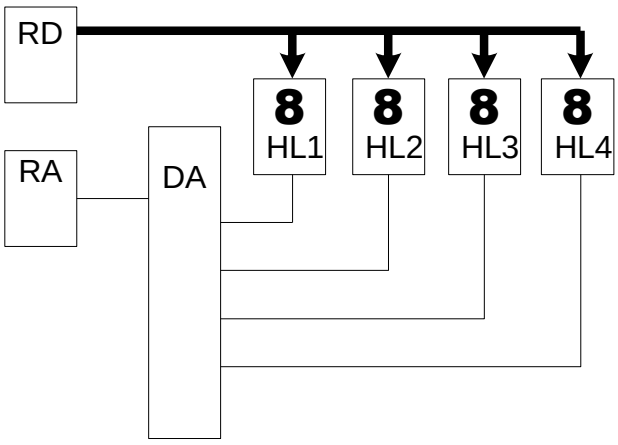


Рисунок 9.3 - Структурна схема динамічної індикації

RD- регістр даних для тимчасового збереження числа або символу; RA - регістр адреси для тимчасового збереження двійкового коду адреси

вибраного індикатора; DA - для перетворення адреси, що задається двійковим кодом в позиційний код; HL1-HL4 - семисегментні індикатори.

При такому вмиканні значно зменшуються апаратні витрати. Але необхідно забезпечити достатній час світіння одного індикатора, для того щоб не зменшувалася яскравість. Також необхідно забезпечити таку частоту перебору індикаторів, щоб не було помітне мерехтіння. Переваги такого способу помітні при кількості десяткових розрядів індикації вже більше 5-ти.

2. Паралельні порти введення-виведення

До складу порту PX (X = A, B, C, D, E) входять три регістри введення-виведення з іменами DDRX, PORTX і PINX. Регістр PINX не має апаратної реалізації. Це ім'я використовується в командах, по яких виконується читання байтів на виводах порту.

Число розрядів у регістрах дорівнює числу виводів порту. На рис.9.4 зображена структурна схема розряду Y (Y=0, I, ..., 7) порту PX.

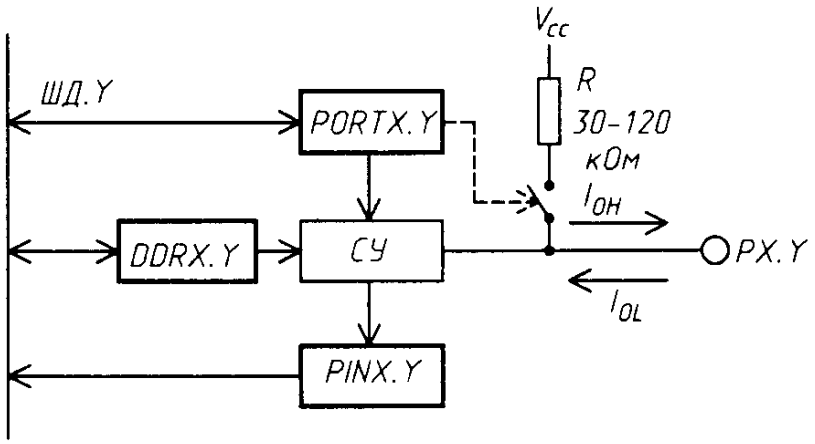


Рисунок 9.4 – Структурна схема розряду Y порту PX

Стан розряду DDRX.Y визначає напрямок передачі біта через вивід порту PX.Y. При DDRX.Y = 0 вивід PX.Y працює в режимі входу, при DDRX.Y = 1 - у режимі виходу.

У режимі входу стан розряду PORTX.Y визначає стан виводу PX.Y. При PORTX.Y = 0 вивід знаходиться у високоімпендансному стані (Z-стан), при PORTX.Y = 1 вивід порту через внутрішній резистор з опором 30-120 кОм підключається до шини VCC. У режимі входу вивід PX.Y з'єднаний із шиною даних ШД.Y.

Значення вхідного сигналу на окремому виводу порту може бути визначене з використанням команд умовного переходу з мнемокодом SBI PINX, Y чи SBI PINX, Y. При одному значенні сигналу вибирається одне продовження програми, при іншім значенні сигналу - інше продовження.

У режимі виходу розряд PORTX.Y визначає значення вихідного сигналу на виході PX.Y. При PORTX.Y = 0 вихідний сигнал має низький рівень напруги , при PORTX.Y = 1 - високий рівень напруги .

У статичному стані при низькому рівні вихідного сигналу струм навантаження Іо, повинний бути не більш, ніж 20 мА, при високому рівні сигналу струм навантаження Іон Повинний бути не більш, ніж 3 мА. При цьому сумарний струм навантаження для усіх виводів мікроконтролера повинний бути не більш, ніж деяке граничне значення (200-400 ма для мікроконтролерів різних типів) і сумарний струм навантаження для виводів мікроконтролера, розташованих на одній стороні корпуса мікросхеми, також повинний бути не більш, ніж деяке граничне значення (100-200 мА для мікроконтролерів різних типів).

Задане значення вихідного сигналу на окремих виводах порту виконується з використанням команд із мнемокодами CBI PORTX, Y (для низького рівня) і SBI PORTX, Y для високого рівня.

При спільному використанні всіх розрядів порту для прийому і видачі байтів використовуються команди з мнемокодами IN Rd, PINX і OUT PORTX, Rr відповідно.

Звертання до паралельного порту для прийому і видачі байтів може бути виконане з використанням команд зі звертанням по адресах в адресному просторі SRAM. При цьому в якості адрес використовуються номери регістрів вводу/виводу, збільшені на $20.

При пуску і перезапуску мікроконтролера всі розряди регістрів DDRX і PORTX усіх портів скидаються в нульовий стан і виводи портів працюють у режимі входів і знаходяться в Z стані.

Порт B.

Порт B є 8-розрядним двунаправленним I/O портом і оснащений вбудованими навантажувальними резисторами. Взаємодія з портом B здійснюється трьома розташованими в просторі I/O пам'яті даних регістрами: регістром даних - PORTB, $18($38), регістром напрямку даних - DDRB, $17($37) і регістром адрес виводів входу - PINB, $16($36). Регістр адрес виводів входу порту B забезпечує можливість тільки читання, регістри даних і напрямку даних порту B забезпечують можливість і читання і записи. Усі виводи порту B оснащені індивідуально будованими навантажувальними резисторами. Вихідні буферим виводів порту B забезпечують струм, який протікає, до 40 мА, що досить для прямого керування LED дисплеями. Якщо виводи з PB0 по PB7 використовуються як входи і зовнішній сигнал утримуються на низькому рівні, то струм, що випливає, забезпечується підключенням внутрішніх навантажувальних резисторів. Виводи порту B можуть виконувати, додатково до основної функції, функції, що представлені в табл. 9.1.

Таблиця 9.1 - Додаткові функції виводів порта B

Вивід порта	Додаткова функція
PB7	Тактовий сигнал послідовної SPI шини - SCK (SPI Bus Ser ial Clock)
PB6	Установка Ведучий вхід/Відомий вихід SPI шини - MISO (SPI Bus Master Input/Slave Output)
PB5	Установка Ведучий вихід/Відомий вхід SPI шини - MOSI (SPI Bus Master Output/Slave Input)
PB3	Порівняння виходу і PWM вихід A таймера/лічильника1 - OC1A/PWM1A (Output Compare and PWM Output A for Timer/Counter1)
PB1	Негативний вхід компоратора (AI1)
PB0	Позитивний вхід компоратора (AI0)

Включення виводів для виконання додаткових функцій виробляється за допомогою регістрів DDRB і PORTB.

Таблиця 9.2 - Регістр даних порта B - PORTB - (PORT B Data Register)

Біти	7 6 5 4 3 2 1 0	
$18 ($38)	PORTB7 PORTB6 PORTB5 PORTB4 PORTB3 PORTB2 PORTB1 PORTB0	PORTB
Читання/Запис	R/W R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

Таблиця 9.3 - Регістр направлення даних порта B - DDRB - (PORT B Data Direction Register)

Біти	7 6 5 4 3 2 1 0	
17(37)	DDB7 DDB6 DDB5 DDB4 DDB3 DDB2 DDB1 DDB0	DDRB
Читання/Запис	R/W R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

PINB - адреса виводів входу порту B не є регістром у повному змісті цього слова, ці адреси забезпечують зчитування фізичного стану кожного виводу порту. При зчитуванні PORTB зчитується стан фіксаторів даних порту B , а при зчитуванні PINB зчитуються безпосередньо логічні стани виводів.

Таблиця 9.4 - Регістр висновків входу порту B - PINB - (PORT B Input Pins Address)

Біти	7 6 5 4 3 2 1 0	
$16($36)	PINB7 PINB6 PINB5 PINB4 PINB3 PINB2 PINB1 PINB0	PINB
Читання/Запис	R R R R R R R R	
Початковий стан	Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z	

Таблиця 9.5 - Вплив бітів DDBn на характер роботи виводів порта B

DDBn	**PORTBn**	**I/O**	**Навантажувальний резистор**	Опис
0	0	Вхід	Не підключений	Третій стан (Hi-Z)
0	1	Вхід	Підключений	При низькому рівні PBn забезпечують струм, що випливає
1	0	Вихід	Не підключений	Низький рівень, двотактний вихід
1	1	Вихід	Не підключений	Високий рівень, двотактний вихід

Біти DDBn регістра DDRB визначають напрямок роботи відповідного виводу. При встановленому в стан 1 біті DDBn вивід PBn конфигурується як вивід виходу. При очищеному біті DDBn (скинутому в 0) вивід PBn конфигурується як вивід входу. якщо біт PORTBn встановлений у стан 1, коли відповідний вивід сконфігурований як вхід, то навантажувальний MOS резистор активується (підключається). Для відключення навантажувального резистора біт PORTBn необхідно очистити чи ж необхідно сконфігурувати вивід як вихід.

Порт D.

Порт D є 7 розрядним двунаправленним I/O портом і оснащений вбудованими навантажувальними резисторами. Взаємодія з портом D здійснюється трьома розташованими в просторі I/O пам'яті даних регістрами: регістром даних - PORTD, $12($32), регістром напрямку даних - DDRD, $11($31) і регістром адрес виводів входу - PIND, $10($30). Регістр адрес виводів входу порту D забезпечує можливість тільки читання, регістри даних і напрямку даних порту D забезпечують можливість і читання і запису. Вихідні буфери виводів порту D забезпечують струм, який протікає, до 40 мА. Якщо виводи з PD0 по PD7 використовуються як входи і зовнішній сигнал утримуються на низькому рівні, то струм, що проходить, забезпечується підключенням навантажувальних резисторів. Виводи порту D можуть виконувати, додаткові до основного, представлені в таблиці. При використанні виводів порту для додаткових функцій, їхнє функціонування визначається установками регістрів DDRD і PORTD.

Таблиця 9.6 - Додаткові функції виводів порту D

Вивід порту	**Додаткова функція**
PD0	Вхід приймача даних –RXD
PD1	Вхід передавача данніх - TXD
PD2	Вхід зовнішнього переривання0 - INT0 - (External Interrupt2 Input)
PD3	Вхід зовнішнього переривання0 - INT1 - (External Interrupt3 Input)
PD4	Вхід тактового сигналу таймера/лічильника0 - T0 - (Timer/Counter1 Clock Input)
PD5	Вхід тактового сигналу таймера/лічильника1 - T1 - (Timer/Counter2 Clock Input)
PD6	Вхід захоплення таймера/лічильника1 - ICP - (Timer/Counter1 Input Capture Trigger)

Таблиця 9.7 -Регістр даних порта D - PORTD - (PORT D Data Register)

Біти	7 6 5 4 3 2 1 0	
$12 ($32)	PORT6 PORT5 PORT4 PORT3 PORT2 PORT1 PORT0	PORTD
Читання/Запис	R R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

PIND - адреса виводів входу порту D не є регістром у повному змісті цього слова, і ці адреси забезпечують зчитування фізичного стану кожного виводу порту. При зчитуванні PORTD зчитується стан фіксаторів даних порту D , а при зчитуванні PIND зчитуються безпосередньо логічні стани виводів. Виводи порту PDn є виводами I/O загального призначення. Стану бітів DDDn регістра DDRD визначають напрямок роботи цих виводів. При встановленому в стан 1 біті DDDn вивід PDn конфигурується як вивід виходу, скидання біта DDDn у стан 0 конфигурує вивід PDn як вивід входу. При установці виводу PDn у стан 1, якщо він сконфігурований

як вхід, активується MOS навантажувальний резистор. Для відключення навантажувального резистора вивід PDn повинний бути очищений (скинутий у стан 0) чи ж повинний бути сконфігурований як вивід виходу.

Таблиця 9.8 - Регістр напрямку даних порту D - DDRD - (PORT D Data Direction Register)

Біти	7 6 5 4 3 2 1 0	
$11($31)	DDD6 DDD5 DDD4 DDD3 DDD2 DDD1 DDD0	DDRD
Читання/Запис	R R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

Таблиця 9.9 - Регістр виводів входу порту D - PIND - (PORT D Input Pins Address)

Біти	7 6 5 4 3 2 1 0	
$10 ($30)	PIND6 PIND5 PIND4 PIND3 PIND2 PIND1 PIND0	PIND
Читання/Запис	R R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z Hi-Z	

Завдання до лабораторної роботи

До порта B підключені 8 світлодіодів. Записати в кодах AT90S2313 програму, яка:

- виконує пересилку до регистру R числа з SRAM і відображає його на світлодіодному індикаторі;
- відображає на індикаторі по черзі молодшу та старшу частину числа з регистру R.

Перевірити виконання програми в AVR STUDIO. При наявності лабораторного макету виконати програмування мікроконтролера і перевірити вірність роботи програми.

Таблица 9.10 - Таблица варіантів завдань

Номер	SRAM	CONST1	R	Номер	SRAM	CONST1	R
1	71	FE	0	15	7F	F8	5
2	62	FF	1	16	6E	F9	6
3	83	FD	2	17	8A	F1	7
4	94	FC	3	18	8D	F2	0
5	A5	FB	4	19	9C	F3	1
6	A6	FA	5	20	D6	F4	2
7	C7	F1	6	21	D5	FF	3
8	78	F2	7	22	67	1F	4
9	69	F3	3	23	CF	2E	5
10	6a	F4	0	24	6D	3D	6
11	7B	F5	1	25	8A	4C	7
12	8C	F6	2	26	9E	5B	1
13	9D	F7	3	27	7F	6A	2
14	AE	F8	4				

Зміст звіту

Схема лабораторного макета (при відсутності макета схему розробити самостійно).
Завдання до лабораторної роботи.
Текст програми з поясненнями.
Результати виконання програми з AVR STUDIO.

Приклад виконання завдання

До порта B підключені 8 світлодіодів. Записати в кодах AT90S2313 програму, яка:

- виконує пересилку до регистру R2 числа 6A з комірки пам’яті SRAM з адресою 7F і відображає його на світлодіодному індикаторі;
- відображає на індикаторі по черзі молодшу та старшу частину числа з регистру 6A.

Виконання першої частини завдання

```
.def temp =R16 ;    присвоїти temp значення R16
clr R27;            очистити старший байт X
LDI R26,$7F;        встановити $7F у молодший байт X
LD R1,X;            завантажити в R1 зміст SRAM за адресою 7F
```

```
        LDI temp,$FF;       ініціалізація порта B
        OUT $17,temp;       порт B настроєний на вивід
    M1: OUT $18,R1;         запис в порт B змісту R1 (загорання світлодіодів)
        RJMP M1
```

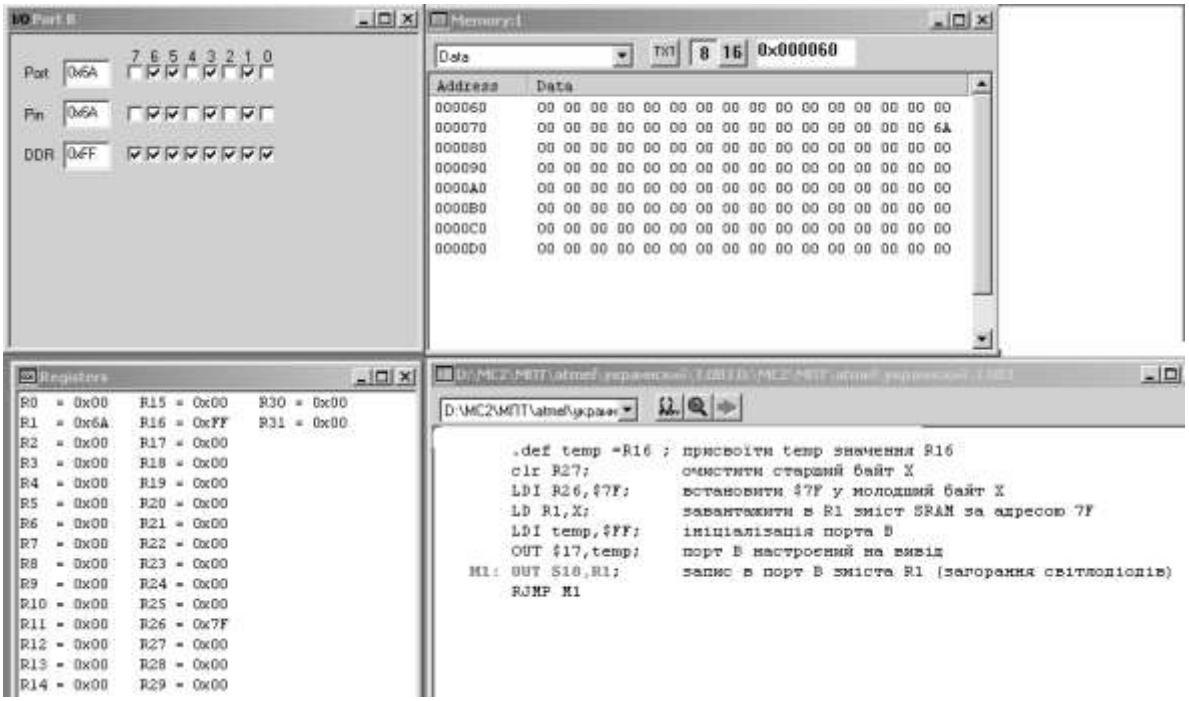


Рисунок 9.5 – Результат виконання першої частини завдання

Програма виконання другої частини завдання

```
    .def temp =R16 ;        Присвоїти temp значення R16
    .def fine =R18;         мітка затримки
    .def medium =R19;       мітка затримки
    .def coarse =R20;       мітка затримки
    clr R27;                очистити старший байт X
    LDI R26,$7F;            встановити $7F у молодший байт X
    LD R1,X;                завантажити в R1 зміст SRAM за адресою 7F
    LDI temp,$FF;           ініціалізація порта B
    OUT $17,temp;           порт B настроєний на вивід
L1: MOV R17,R1;             переписування інформації в R17 з R1
    ANDI R17,$0F;           маскування старшого байту числа
    OUT $18,R17;            запис в порт B змісту молодшої частини R17 (загорання
                            світлодіодів)
    RCALL delay;            виклик підпрограми затримки
    MOV R17,R1;             переписування інформації в R17 з R1
    ANDI R17,$F0;           маскування молодшого байту числа
    OUT $18,R17;            запис в порт B змісту старшої частини R17 (загорання
                            світлодіодів)
    RCALL delay;            виклик підпрограми затримки
    RJMP L1;

delay:  LDI coarse,8;
cagain:     LDI medium,255;     отримання затримки 1/2 секунди
```

```
magain:     LDI fine,255;     при 4МГц тактовій частоті
fagain:     dec fine;
            brne fagain;
          DEC medium;
         BRNE magain;
       DEC coarse;
       BRNE cagain;
       RET
```

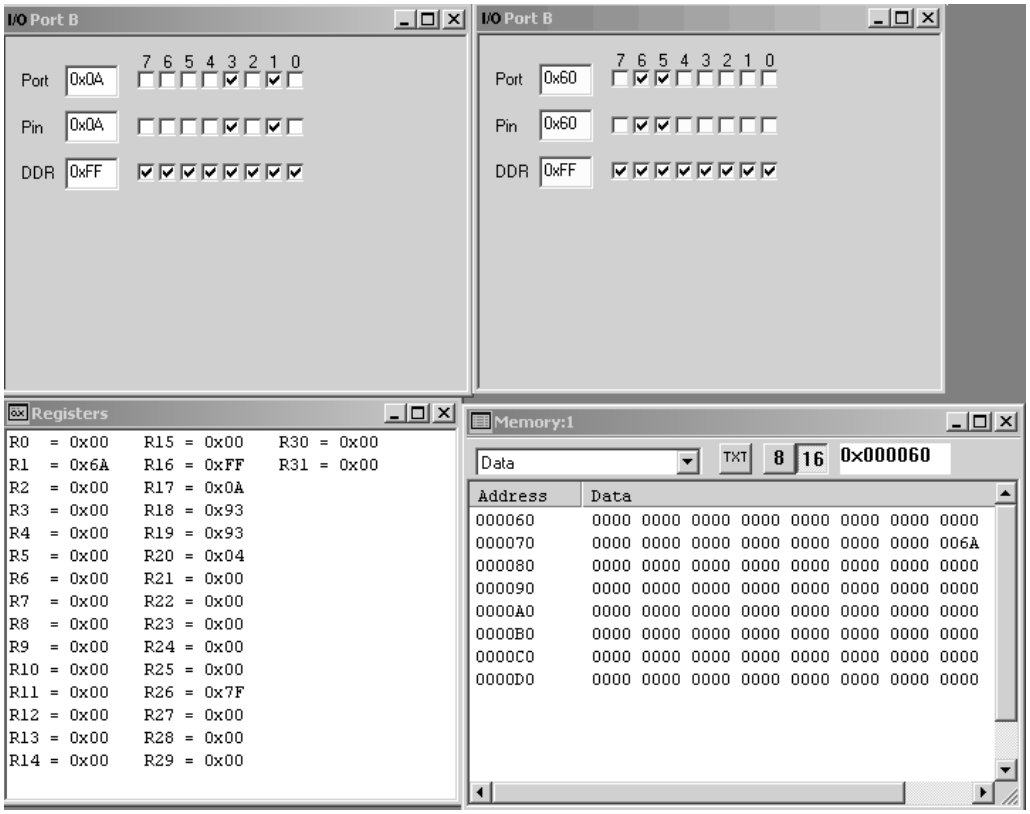


Рисунок 9.6 – Результат виконання другої частини завдання

Контрольні запитання

1. Наведіть найпростіші схеми відображення інформації в цифрових пристроях.
2. Що таке статична індикація? Наведіть схемну реалізацію цієї індикації.
3. Що таке динамічна індикація та як вона реалізовується?
4. Розкажіть про характеристики порта B мікроконтролера.
5. Охарактеризуйте порт D мікроконтролера.

Лабораторна робота №10

СИСТЕМА ПЕРЕРИВАНЬ. ОПИТУВАННЯ ДИСКРЕТНИХ ДАТЧИКІВ

Мета роботи: вивчення режимів роботи системи переривань мікроконтролера AT90S2313, програмна обробка дискретних сигналів.

Теоретичні відомості

1. Опитування дискретних сигналів

Для введення інформації широко застосовуються кнопкові перемикачі і контактні клавіатури. Сигнал таких перемикачів формується шляхом замикання (розмикання) електричного ланцюга. Сигнал, формований контактною парою, супроводжується тремтінням, тривалість якого складає 8 –12 мс (рис. 10.1).

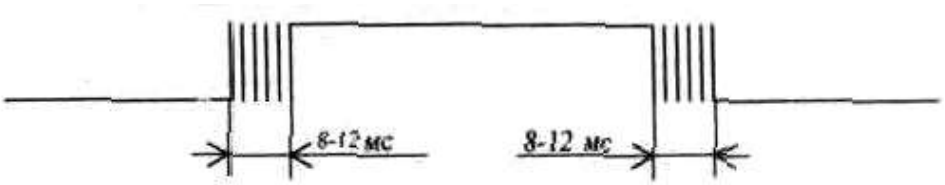


Рисунок 10.1– Сигнал контактної пари

Для усунення тремтіння контактів (shaking of contacts) на виході контактної пари встановлюють спеціальні формувачі. Приклад такого формувача заснованого на принципі безпосередньої установки RS-тригера приведений на рис. 10.2.

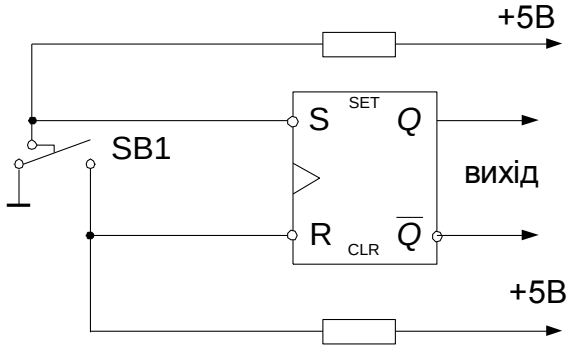


Рисунок 10.2 – Схема усунення тремтіння за допомогою RS-тригера

Для зменшення апаратних витрат застосовують програмне зменьшення тремтіння. Воно полягає в повторному опитуванні контактної пари з затримкою в 12 мс, при збігу результатів опитування кнопка була натиснута, інакше в результаті першого опитування було зафіксовано тремтіння.

2. Обробка переривань і скидання

Мікроконтролер AT90S2313 має 3 джерела сигналу скидання:

- скидання по включенню живлення (Power-On Reset). MCU скидається при підключенні живлення до виводів VCC і GND;
- зовнішнє скидання (External Reset). MCU скидається якщо низький рівень присутній на вході більш 50 нс;
- скидання по сторожовому таймері (Watchdog Reset). MCU - скидається якщо минає період сторожового таймера і сторожовий таймер дозволений.

Протягом скидання всі регістри I/O, за винятком регістра статусу MCU, встановлюються в їхні початкові стани і програма починає роботу з адреси $000. По цій адресі повинна знаходитися команда RJMP - команда абсолютного переходу до підпрограми обробки скидання. Якщо програма ніколи не дозволяє переривання, то вектори переривань не використовуються і по цих адресах можуть розташовуватися коди програми.

Скидання по включенню живлення. Схема скидання по включенню живлення (Power-On Reset - POR) забезпечує запуск мікроконтролера тільки по досягненні напругою Vcc безпечного рівня. Вбудований таймер, трактується вбудованим генератором сторожового таймера, утримує запуск MCU на якийсь час після досягнення граничної напруги вкючения живлення Vpot , що не залежить від швидкості наростання напруги Vcc. Якщо вбудована затримка запуску достатня, то RESET може бути приєднаний до Vcc чи безпосередньо через зовнішній навантажувальний резистор. Утриманням виводу на низькому рівні, під час подачі напруги, період скидання по включенню живлення може бути збільшений.

Зовнішнє керування скиданням. Зовнішнє скидання формується подачею низького рівня на вивід RESET на час не менше двох тактових циклів кварцового генератора. При досягненні напругою на виводі RESET рівня VRST запускається таймер, що затримує запуск MCU на час t TOUT.

Скидання по сторожовому таймері. По закінченні часу, обумовленого сторожовим таймером, таймер формує короткий, тривалістю в один цикл XTAL, імпульс скидання. По падаючому фронті цього імпульсу таймер затримки починає відлік t TOUT.

Блок переривань приймає запити переривання й організує перехід до виконання визначеної програми, що перериває. Запити переривання надходять із зовнішніх джерел і з джерел, розташованих у різних пристроях мікроконтролера.

Для прийому запитів із зовнішніх джерел використовуються виводи паралельних портів введення/виведення, для яких ця функція є альтернативної. При виконанні альтернативної функції вивід порту має альтернативне ім'я INTx (x = 0, 1, ..., 7).

Запит переривання з зовнішнього джерела може бути представлений низьким рівнем сигналу (L), переходом від високого рівня до низького (HL), переходом від низького рівня до високого (LH) або переходом будь-якого напрямку (LH/HL). Вибір способу представлення визначається комбінацією станів розрядів ISCx0 і ISCxl у регістрі MCUCR ($D5).

Кожний запит переривання надходить у блок переривань, якщо переривання в мікроконтролері дозволені (I = SREG.7 = 1) і дозволене переривання по даному запиті. Переривання по окремому запиті дозволено, якщо в одиничному стані знаходиться розряд, що маскує, (MASK) для даного запиту переривання, розташований в одному з регістрів вводу/виводу.

З появою запиту переривання встановлюється в одиничний стан прапор розряд (FLAG) для даного запиту переривання, розташований в одному з регістрів вводу/виводу. Стан прапорця розряду запитується апаратно і, крім того, може бути опитано програмними засобами.

При надходженні запиту блок переривань організує апаратний безумовний перехід до виконання команди, адреса якої (вектор переривання) однозначно зв'язаний з ім'ям запиту переривання По цій адресі в мікроконтролерах серій AT90 повинна бути записана команда безумовного переходу з мнемокодом RJMP k, машинний код який має формат "слово. По цій команді виконується програмний безумовний перехід до першої команди відповідної програми, що перериває, що може бути розташована в будь-якому місці в FlashROM.

При одночасному надходженні в блок переривань декількох запитів у блоці виділяється запит з найбільш високим пріоритетом серед усіх що надійшли і виконується перехід за адресою, який відповідає цьому запиту.

При переході до виконання програми, що перериває, розряд I в регістрі SREG апаратно скидається в нульовий стан і переривання по всіх запитах виявляються забороненими. Розряд I встановлюється в одиничний стан при виконанні команди повернення з програми, що перериває, із мнемокодом RETI. Розряд I може бути встановлений в одиничний стан програмно по команді SEI в програмі, що перериває. Програма, що виконується при пуску мікроконтролера і по запиті RESET, не містить команди RETI і для виконання переривань повинна містити команду SEI.

Переривання по запиті RESET виконується поза залежністю від стану розряду і у регістрі SREG. Опитування стану входів блоку перериванні виконується в кожнім такті. При виявленні запиту код з лічильника команд заноситься в стек, на що затрачається 2 такти, і виконується безумовний перехід по команді з мнемокодом RJMP k (2 такти). Якщо при виявленні запиту переривання процесор не закінчив виконання поточної багататактової команди, до переходу до програми, що перериває, завершується виконання цієї команди.

Мікроконтролери AVR2313 використовують 10 джерел переривання. Ці переривання і вектор скидання розташовують окремими програмними векторами в просторі пам'яті програм. Кожному перериванню привласнений свій біт дозволу який повинний бути встановлений разом з бітом I регістра статусу.

Молодші адреси простору пам'яті програм автоматично визначаються як вектори скидання і переривань.

Повний перелік векторів представлений у табл. 10.1. Перелік представляє також рівень пріоритету для кожного переривання. Переривання з молодшими адресами мають більший рівень пріоритету. RESET має найвищий рівень пріоритету, що випливає є INT0 - запит зовнішнього переривання 0 і т.д.

Перед початком роботи з AVR мікроконтролерами виконується настройка його режимів роботи. Для цього використовуються регистри MCUR, GIMSK, SREG. Даний процес називається ініціалізація.

Регістр статусу – SREG.

Регістр статусу - SREG - розміщений у просторі I/O за адресою $3F ($5F) (рис. 10.2).

Bit 7 - I: Global Interrupt Enable - Дозвіл глобального переривання. Біт дозволу глобального переривання для дозволу переривання повинний бути встановлений у стан 1. Керування дозволом конкретного переривання виконується регістрами маски переривання GIMSK і TIMSK. Якщо біт глобального переривання очищений (у стані 0), то жодне з дозволів конкретних переривань, встановлених у регістрах GIMSK і TIMSK, не діє. Біт I апаратно очищається після переривання і встановлюється для наступного дозволу глобального переривання командою RETI.

Таблиця 10.1– Вектори скидання і переривань

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Скидання по виводу і сторожовому таймері (Hardware Pin and Watchdog Reset)
2	$001	INT0	Запит зовнішнього переривання 0 (External Interrupt Request 0)
3	$002	INT1	Запит зовнішнього переривання 1 (External Interrupt Request 1)
4	$003	TIMER1 CAPT1	Захоплення таймера/лічильника 1 (Timer/Counter1 Capture Event)
5	$004	TIMER1 COMP	Збіг A при порівнянні таймера/лічильника 1 (Timer/Counter1 Compare Match A)
6	$005	TIMER1 OVF1	Переповнення таймера/лічильника 1 (Timer/Counter1 Overflow)
7	$006	TIMER0 OVF0	Переповнення таймера/лічильника 0 (Timer/Counter0 Overflow)
8	$007	UART, RX	Завершення прийому UART (UART, Rx Complete)
9	$008	UART, UDRE	Регістр даних UART порожній (UART Data Register Empty)
10	$009	UART, TX	Завершення передачі UART (UART, Tx Complete)
11	$00A	ANALOG COMP	Спрацьовування аналогового компаратора (Analog Comparator)

Таблиця 10.2 - Регістр статусу SREG

Біти	7 6 5 4 3 2 1 0	
$3F ($5F)	I T H S V N Z C	REG
Читання/Запис	R/W R/W R/W R/W R/W R/W R/W R/W	
Початковий стан	0 0 0 0 0 0 0 0	

Bit 6 - T: Bit Copy Storage - Біт збереження копії. Команди копіювання біта BLD (Bit Loa) і BST (Bit STore) використовують біт T, як біт джерело і біт призначення при операціях з бітами. Командою BST біт регістра реєстрового файлу копіюється в біт T, командою BLD біт T копіюється в регістр реєстрового файлу.

Bit 5 - H: Half Carry Flag - Прапор напівпереносу. Прапор напівпереносу вказує на напівперенос у ряді арифметичних операцій

Bit 4 - S: Sign Bit, S = N V - Біт знака. Біт S завжди знаходиться в стані, обумовленому логічним що виключає ЧИ (exclusive OR) між прапором негативного значення N і доповненням до двох прапора переповнення V.

Bit 3 - V: Two's Complement Overflow Flag. Доповнення до двох прапора переповнення. Доповнення до двох прапора V підтримує арифметику доповнення до двох.

Bit 2 - N: Negative Flag - Прапор негативного значення. Прапор негативного значення N вказує на негативний результат ряду арифметичних і логічних операцій.

Bit 1 - Z: Zero Flag -Прапор нульового значення. Прапор нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій.

Bit 0 - C: Carry Flag -Прапор переносу. Ознаки результату операції можуть бути використані в програмі для виконання подальших арифметично-логічних операцій чи команд умовних переходів.

Регістр керування MCU - MCU Control Register – MCUCR.

Біти регістра керування MCU керують виконанням основних функцій MCU (табл. 10.3).

Bit 5 - SE: Sleep Enable - Дозвіл режиму Sleep. Встановлений у 1 біт SE дозволяє переклад MCU у режим sleep по команді SLEEP. Щоб виключити переклад MCU у незапрограмований режим sleep, рекомендується встановлювати біт SE безпосередньо перед виконанням команди SLEEP.

Bits 4 - SM: Sleep Mode. Коли SM=0, то режим - Idle Mode, коли SM=1,то Power Down.

Таблиця 10.3 - Регістр керування – MCUCR

Біти	7 6 5 4 3 2 1 0	
$35B ($55B)	- - SE SM ICS11 ICS10 ICS01 ICS00	MCUCR
Читання/Запис	R R R/W R/W R/W R/W R/W R/W	

Bits 3,2 – ISC11, ICS10 – Керування значенням переривання. Зовнішнє переривання 1 активується на виводу INT1 , якщо встановлений прапор SREG і в регістрі GIMSK відповідна маска переривання:

00 Низький рівень INT1 генерує запит переривання;
01 Зарезервований;
10 Спадаючий рівень INT1 генерує запит переривання;
11 Зростаючий рівень INT1 генерує запит переривання.

Bits 1,0 – ISC01, ICS00 – Керування значенням переривання. Зовнішнє переривання 1 активується на виводу INT0 , якщо встановлений прапор SREG і в регістрі GIMSK відповідна маска переривання:

00 Низький рівень INT0 генерує запит переривання;
01 Зарезервований;
10 Спадаючий рівень INT0 генерує запит переривання;
11 Зростаючий рівень INT0 генерує запит переривання.

РЕЖИМ Idle

Якщо біти SM знаходяться в стані 0 команда SLEEP переводить MCU у режим Idle, зупиняючи CPU , але залишаючи активними таймери/лічильники, сторожовий таймер і систему переривань. Це забезпечує активацію MCU зовнішніми перериваннями і такими внутрішніми перериваннями, як переповнення таймера і завершення прийому UART. Якщо активація по аналоговому компараторі не потрібно, то аналоговий компаратор може бути відключений установкою біта ACD у регістрі керування і статусу аналогового компаратора ACSR. Це дозволить додатково знизити споживання в Idle режимі. При активації MCU з Idle режиму CPU починає виконувати програму негайно.

РЕЖИМ Power Down

При установці битов SM у стан 1 команда SLEEP переводить MCU у режим Power Down. У цьому режимі зупиняється зовнішній генератор. Користувач може дозволити роботу сторожового таймера. Якщо сторожовий таймер дозволений, то активація MCU відбудеться по завершенні встановленого в сторожовому таймері періоду часу. Якщо зовнішнє джерело тактового сигналу підключений до висновку XTAL1, то активація MCU з режиму Power Down може відбуватися без ATMEL затримки, звичайно необхідної для стабілізації XTAL генератора.

Регістр масок (register of masks) зовнішніх переривань – GIMSK.

Регістр статусу (register of status) MCU видає інформацію про джерело, що викликало скидання MCU (табл. 10.4).

Таблиця 10.4 – Регістр масок зовнішніх переривань – GIMSK

Біти	7 6 5 4 3 2 1 0	
$3B ($5B)	INT1 INT0	GIMSK
Читання/Запис	R R R R R R R R	
Початковий стан	0 0 0 0 0 0 0 0	

Bits 7 – INT1: External Interrupt Request 1 Enable - дозвіл запиту зовнішніх переривань 1. При встановленому біті INT1 і встановленому біті I регістра статусу (SREG) дозволяються переривання по відповідним виводам входів сигналів переривань. Біт керування впізнанням переривання регістра керування зовнішніми перериваннями EICR (External Interrupt Control Register) визначає спрацьовування по наростаючому чи спадаючому фронті по логічному рівні. Активація кожного з цих виводів викликає запит переривання навіть якщо вивід буде дозволений як вихід. Це забезпечує можливість організації програмного переривання.

Bits 6 - INT0: External Interrupt Request 0 Enable - дозвіл запиту зовнішніх переривань 0. При встановленому біті INT0 і встановленому біті I регістра статусу (SREG) дозволяються переривання по відповідним входах переривань. Зовнішні переривання завжди викликають переривання низьким рівнем. Активація кожного з цих виводів викликає запит переривання навіть якщо вивід буде дозволений як вихід. Це забезпечує можливість організації програмного переривання. Запит переривання по логічному рівні, якщо він дозволений, буде генерувати запит переривання доти , поки на вході буде знаходитися низький рівень.

3. Організація програмно-апаратної клавіатури

Для керування режимами роботи радіоелектроних приладів використовується клавіатура. Приклад організація клавіатури 4x4 зображен на рис. 10.3. До порта D підключени червоний та зелений світлодіод. Нижче приводиться програма, яка визначає номер натиснутої клавіши. Зелений світлодіод мигає в залежності від номера натиснутої кнопки. Якщо натиснута кнопка з номером 0, то мигає червоний світлодіод 10 раз. Мікроконтролер знаходиться в програмі в Sleep mode та при натисненні кнопки спрацьовує переривання (вхід INT0), мікропроцесор переходить до підпрограми обслуглвування перивання, яка починається з адреси $001.

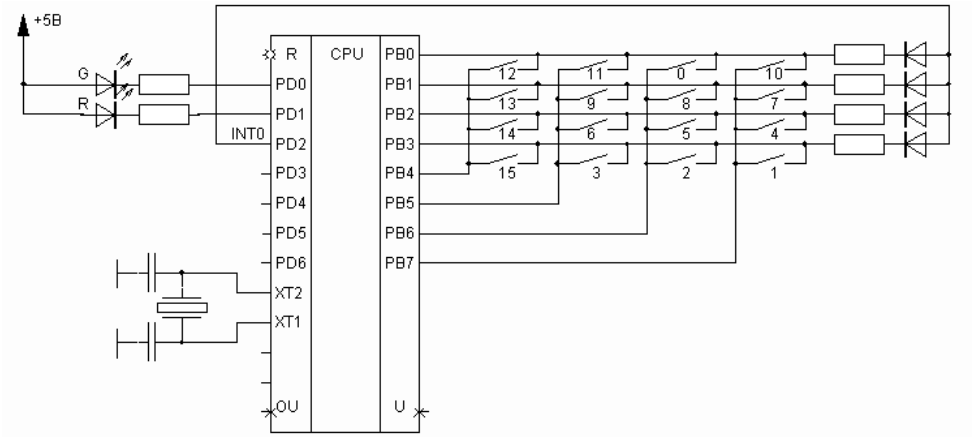


Рисунок 10.3 – Організація клавіатури на мікроконтролері AT90S2313

Лістінг програми

```
;* Title:                4x4 keypad, wake-up on keypress
;* DESCRIPTION
;* This Application note scans a 4 x 4 keypad and uses sleep mode causing the AVR to wake up on keypress.
;*  The design uses a minimum of external components. Included is a test program that wakes up the AVR
;* and performs a scan when a key is pressed and flashes one of two LEDs the number of the key pressed.
;* The external interrupt line is used for wake-up.  The example runs on the AT90S2313 but can be any AVR
;* with suitable  changes in vectors, EEPROM and stack pointer. The timing assumes a 4 MHz clock.
;* A look up table is used in EEPROM to enable the same structure to be used with more advanced programs        ;* e.g ASCII output to displays.
;***********************************************************************
;***** Register used by all programs
;******Global variable used by all routines
.def    temp    =r16    ;general scratch space
;Port B pins
.equ    ROW1    =3      ;keypad input rows
.equ    ROW2    =2
```

```
.equ    ROW3   =1
.equ    ROW4   =0
.equ    COL1   =7       ;keypad output columns
.equ    COL2   =6
.equ    COL3   =5
.equ    COL4   =4
;Port D pins
.equ    GREEN  =0       ;green LED
.equ    RED    =1       ;red LED
.equ    INT    =2       ;interrupt input
.include "2313def.inc"
;***** Registers used by interrupt service routine
.def    key    =r17     ;key pointer for EEPROM
.def    status =r21     ;preserve sreg here
;***** Registers used by delay subroutine
;***** as local variables
.def    fine   =r18     ;loop delay counters
.def    medium =r19
.def    coarse =r20
;*****Look up table for key conversion*******************************
.eseg                                   ;EEPROM segment
.org 0
        .db    1,2,3,15,4,5,6,14,7,8,9,13,10,0,11,12
;****Source code***************************************************
.cseg                                   ;CODE segment
.org 0
                rjmp reset              ;Reset handler
                rjmp scan               ;interrupt service routine
                reti                    ;unused timer interrupt
                reti                    ;unused analogue interrupt
;*** Reset handler ************************************************
reset:
                ldi temp,0xFB           ;initialise port D as O/I
                out DDRD,temp           ;all OUT except PD2 ext.int.
                ldi temp,0x30           ;turn on sleep mode and power
                out MCUCR,temp          ;down plus interrupt on low level.
                ldi temp,0x40           ;enable external interrupts
                out GIMSK,temp
                sbi ACSR,ACD            ;shut down comparator to save power
main:           cli                     ;disable global interrupts
                ldi temp,0xF0           ;initialise port B as I/O
                out DDRB,temp           ; 4 OUT  4 IN
                ldi temp,0x0F           ;key columns all low and
                out PORTB,temp          ;active pull ups on rows enabled
                ldi temp,0x07           ;enable pull up on PD2 and
                out PORTD,temp          ;turn off LEDs
                sei                     ;enable global interrupts ready
                sleep                   ;fall asleep
                rcall flash             ;flash LEDs for example usage
                ldi temp,0x40
                out GIMSK,temp          ;enable external interrupt
```

```
                rjmp main               ;go back to sleep after keyscan
;****Interrupt service routine**************************************
scan:
                in status,SREG          ;preserve status register
                sbis PINB,ROW1          ;find row of keypress
                ldi key,0       ;and set ROW pointer
                sbis PINB,ROW2
                ldi key,4
                sbis PINB,ROW3
                ldi key,8
                sbis PINB,ROW4
                ldi key,12
                ldi temp,0x0F           ;change port B I/O to
                out DDRB,temp           ;find column press
                ldi temp,0xF0           ;enable pull ups and
                out PORTB,temp          ;write 0s to rows
                rcall settle            ;allow time for port to settle
                sbis PINB,COL1          ;find column of keypress
                ldi temp,0              ;and set COL pointer
                sbis PINB,COL2
                ldi temp,1
                sbis PINB,COL3
                ldi temp,2
                sbis PINB,COL4
                ldi temp,3
                add key,temp            ;merge ROW and COL for pointer
                ldi temp,0xF0           ;reinitialise port B as I/O
                out DDRB,temp           ; 4 OUT  4 IN
                ldi temp,0x0F           ;key columns all low and
                out PORTB,temp          ;active pull ups on rows enabled
                out SREG,status         ;restore status register
                ldi temp,0x00
                out GIMSK,temp          ;disable external interrupt
                                        ;have to do this, because we're
                                        ;using a level-triggered interrupt
                reti                    ;go back to main for example program
;***Example test program to flash LEDs using key press data***********
flash:          out EEAR,key            ;address EEPROM
                sbi EECR,EERE           ;strobe EEPROM
                in temp,EEDR            ;set number of flashes
                tst temp        ;is it zero?
                breq zero       ;do RED LED
green_flash:
                        cbi PORTD,GREEN         ;flash green LED 'temp' times
                        rcall delay
                        sbi PORTD,GREEN
                        rcall delay
                        dec temp
                        brne green_flash
exit:           ret
zero:           ldi temp,10
```

```
flash_again:        cbi PORTD,RED  ;flash red LED ten times
                    rcall delay
                    sbi PORTD,RED
                    rcall delay
                    dec temp
                    brne flash_again
                rjmp exit
;****Time Delay Subroutine for LED flash*********************************
delay:
        ldi coarse,8                    ;triple nested FOR loop
cagain:         ldi medium,255          ;giving about 1/2 second
magain:                 ldi fine,255    ;delay on 4 MHz clock
fagain:                 dec fine
                        brne fagain
                dec medium
                brne magain
        dec coarse
        brne cagain
        ret
;***Settling time delay for port to stabilise*****************************
settle:
        ldi temp,255
tagain:         dec temp
                brne tagain
        ret
```

Завдання до лабораторної роботи

До порта B підключені 8 світлодіодів. До порта D підключена клавіатура 3x3. Записати в кодах AT90S2313 програму (режим переривання не використовується), яка виконує завдання згідно табл. 10.5.

Перевірити виконання програми в AVR STUDIO та Proteus VSM.

Таблиця 10.5 - Варіанти індивідуальних завдань

№	Текст індивідуального завдання
1	Натиснути будь-які кнопки на клавіатурі, відображати номер кнопок на індикаторі.
2	Натиснути кнопку S2, здійснити періодичне миготіння числа 43
3	У двійковому виді відображати на одиничних індикаторах номер натиснутої кнопки.
4	Здійснити включення/вимикання одиничних індикаторів HL1-HL8 у кількості відповідному номеру натиснутої клавіші .
5	На індикаторі відобразити число 10. По натисканню кнопки S5, здійснювати зменшення на одиницю відображуваного числа.
6	На індикаторі відобразити число 12. По натисканню кнопки S6, здійснювати збільшування на одиницю відображуваного числа.

Продовження таблиці 10.5

7	На індикаторі відобразити число 14. По натисканню кнопки S7, здійснювати зменьшування на два відображуваного числа.
8	Натиснути кнопку 1 на клавіатурі. Відобразити на індикаторі число 1, здійснити періодичне миготіння числа 4.
9	Натиснути будь-яку кнопку на клавіатурі. На всіх розрядах здійснити відображення номера кнопки.
10	Натиснути кнопку S5 відобразити на індикаторі число 10. Натиснути S6 відобразити на індикаторі число 11.
11	Відобразити на індикаторі число 21. По натисканню кнопки S1 зменшувати на одиницю відображуване число, по натисканню S2 збільшувати на одиницю відображуваного числа.
12	Відобразити на індикаторі число 8. По натисканню кнопки S9 зменшувати на одиницю відображуване число, по натисканню S8 збільшувати на одиницю відображуваного числа.
13	Якщо натиснута кнопка від 0 до 4 відобразити на індикаторі номер цієї кнопки, в усіх інших випадках горять всі індикатори.
14	Сформувати світіння індикаторів HL1-HL8 у залежності від номера натиснутої кнопки на клавіатурі.
15	Якщо натиснута кнопка від 5 до 8 відобразити на індикаторі номер цієї кнопки, в усіх інших випадках горять всі індикатори.

Зміст звіту

Схема лабораторного макета (при відсутності макета схему розробити самостійно).
Завдання по лабораторній роботі.
Текст программ з поясненнями.
Результати виконання програми з AVR STUDIO.

Контрольні запитання

1. Які Ви знаєте способи скидання мікроконтролерів?
2. Охарактеризуйте роботу кожного із відомих способів скидання.
3. Розкажіть про регістр масок зовнішніх переривань GIMSK.
4. Як організувати клавіатуру на мікроконтролері AT90S2313?
5. Наведіть схему усунення тремтіння за допомогою RS-тригера.

Лабораторна робота №11

ВИВЧЕННЯ ОСОБЛИВОСТЕЙ НАВЧАЛЬНОГО СТЕНДА НА ПРОЦЕСОРІ ATmega8535

Мета роботи: вивчити структуру мікропроцесорного комплекту навчального стенду, його технічні характеристики та навчитися конфігурувати програмні засоби для програмування анагогових входів і світлодіодних індикаторів.

Теоретичні відомості

1. Призначення стенду

Високонадійні уніфіковані мікроконтролери AVR розроблені для використання в самих різних бортових обчислювальних системах, розподілених системах комплексної автоматизації, в медичному обладнанні, автоматизації будівель, побутових приладах, пристроях зв'язку, системах охорони та інших областях, де потрібно забезпечити високу надійність роботи в умовах електромагнітних завад і в широкому діапазоні температур.

Навчальний стенд (далі – стенд), оснований на процесорі ATmega8535, завдяки великій кількості цифрових та аналогових входів-виходів і підтримці різних комунікаційних інтерфейсів дозволяє інтегруватися практично в любу систему для виконання широкого спектру задач, пов'язаних із автоматизацією, керуванням, збором та обробкою інформації. Підтримка протоколу програмування в пристрої дозволяє не тільки спростити процес розробки, а й надати пристроям нових властивостей, що дозволяють модифікувати програмне забезпечення безпосередньо у готовому пристрої.

Конфігурація стенду дозволяє вирішувати самі різноманітні задачі:

- створення розподіленої мережі автоматизації;
- моніторинг параметрів технологічного процесу, стану системи і візуальне відображення зібраної інформації;
- обробка зібраної інформації і реалізація нескладних законів керування;
- передача і пройом інформації з комп'ютера оператора по стандартному послідовному інтерфейсу RS232.

2. Основні технічні характеристики

Стенд має такі апаратні ресурси:

- мікроконтролер ATmega8535;
- стабілізатор напруги +5В на основі мікросхеми LM7805;
- вбудований програматор;

- роз’єм типу DRB-9FA для підключення пристроїв по інтерфейсу RS-232 з оптичною розв’язкою (6N137);
- 8 світлодіодів користувача;
- 4 кнопки з фіксацією;
- п’єзодзінок;
- аналоговий мультиплексор на основі мікросхеми К561КП2 (CD4051);
- годинник реального часу PCF8583 із кварцовим резонатором 32,768 кГц;
- індикатор світлодіодний чотирирозрядний на основі статичної індикації;
- генератор частоти.

Для програмування стенду використовується кабель принтера типу LPT. Принципова схема програматора представлена на рис. 11.1.

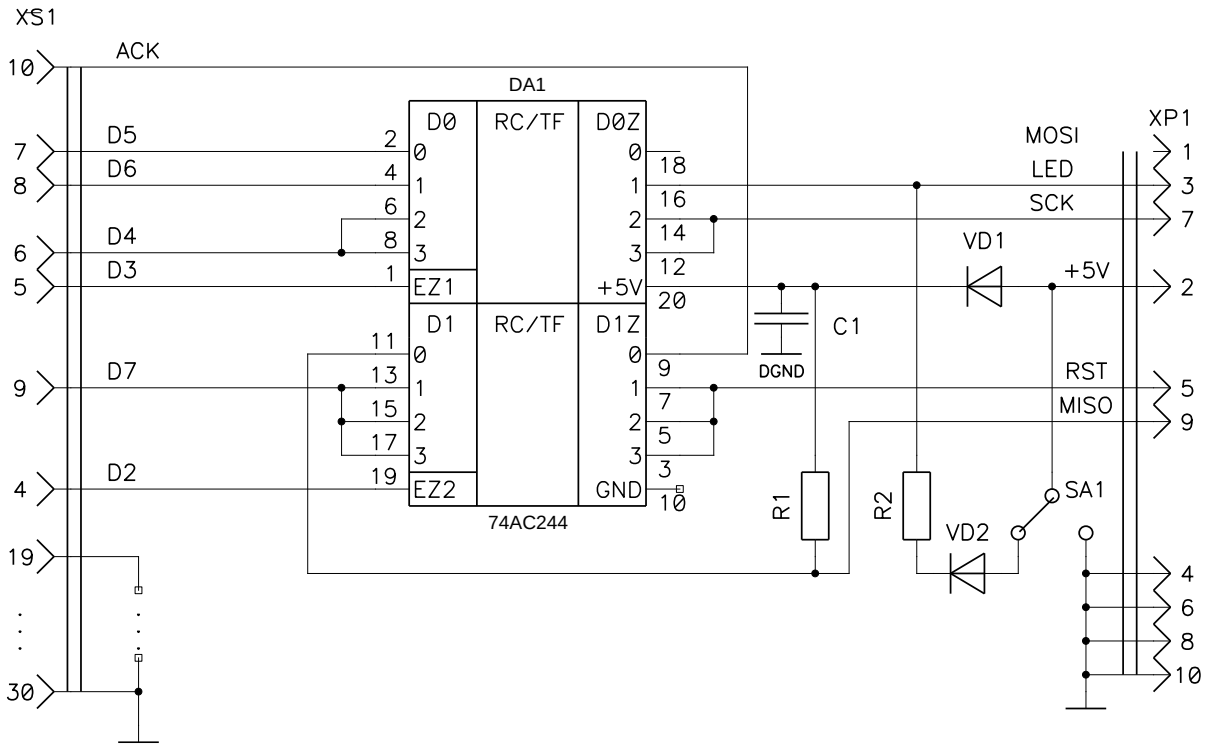


Рисунок 11.1 - Принципова схема програматора

3. Системні вимоги

- 486 процесор і вище;
- оперативна пам’ять не менше 16 МБ;
- вільні 16 МБ на жорсткому диску для установки програмного забезпечення;
- операційна система Windows;
- порт LPT для підключення програматора.

4. Початкові установки і стартова програма

Стенд поставляється із записаною в мікроконтролер програмою. При правильному підключенні стенду до джерела живлення програма починає працювати. Користувач може вносити зміни в програму.

Рекомендується така послідовність дій при першому знайомстві із стендом:

а) ознайомитися із даним керівництвом;

б) встановити на комп'ютер програму AVR Studio;

в) встановити на комп'ютер програму ICC AVR;

г) встановити на комп'ютер програму PonyProg;

д) з'єднати комп'ютер та стенд кабелем для програмування;

е) включити стенд в мережу живлення.

5. Структура стенда

Спрощена структурна схема стенда наведена на рис. 11.2. Із неї видно, що основний елемент – мікроконтролер (МК) – з'єднаний з масою периферійних пристроїв, що підключаються за допомогою додаткових шлейфів. Роботу схеми зручніше аналізувати за принциповою схемою, що наведена в додатку.

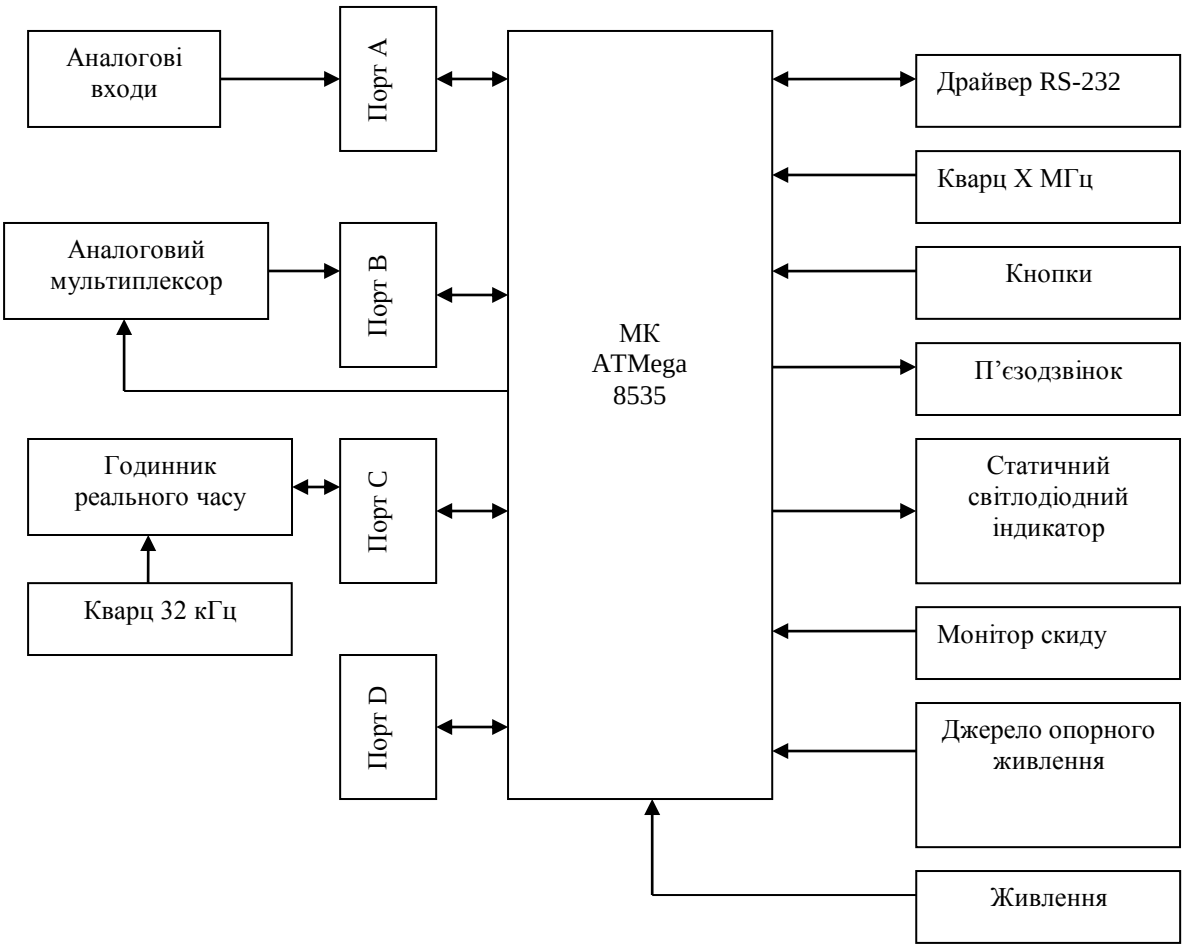


Рисунок 11.2 – Спрощена структкрна схема навчального стенда

Призначення роз’ємів (рис. 11.3):

XP1 – введення сигналів через аналоговий комутатор до аналогового компаратора мікроконтролера (рис. 11.4);

XP2 – послідовний інтерфейс RS232C;

XP3 – програмування мікроконтролера від персонального комп’ютера;

XP4 – дискретний порт введення/виведення інформації Port D;

XP5 – введення аналогових сигналів до аналого-цифрового перетворювача мікроконтролера (Port A) (рис. 11.6);

XP6 – дискретний порт введення/виведення інформації Port C (рис.11.5);

XP7 – стабілізоване живлення мікроконтролера +5В;

XP8 – підключення годинника реального часу PCF8583;

XT1 – нестабілізована напруга +12В.

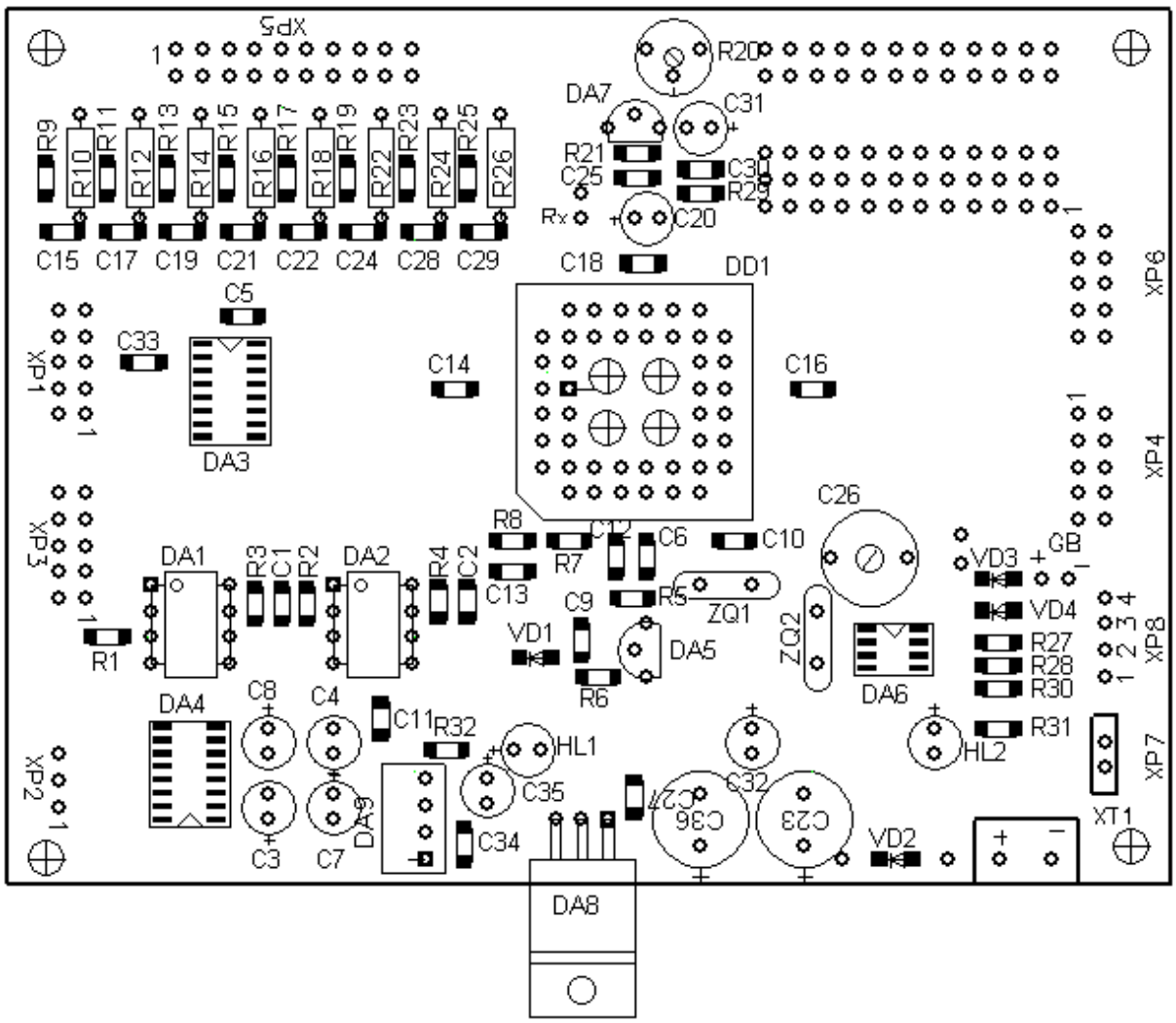


Рисунок 11.3 - Розташування елементів на лабораторному стенді

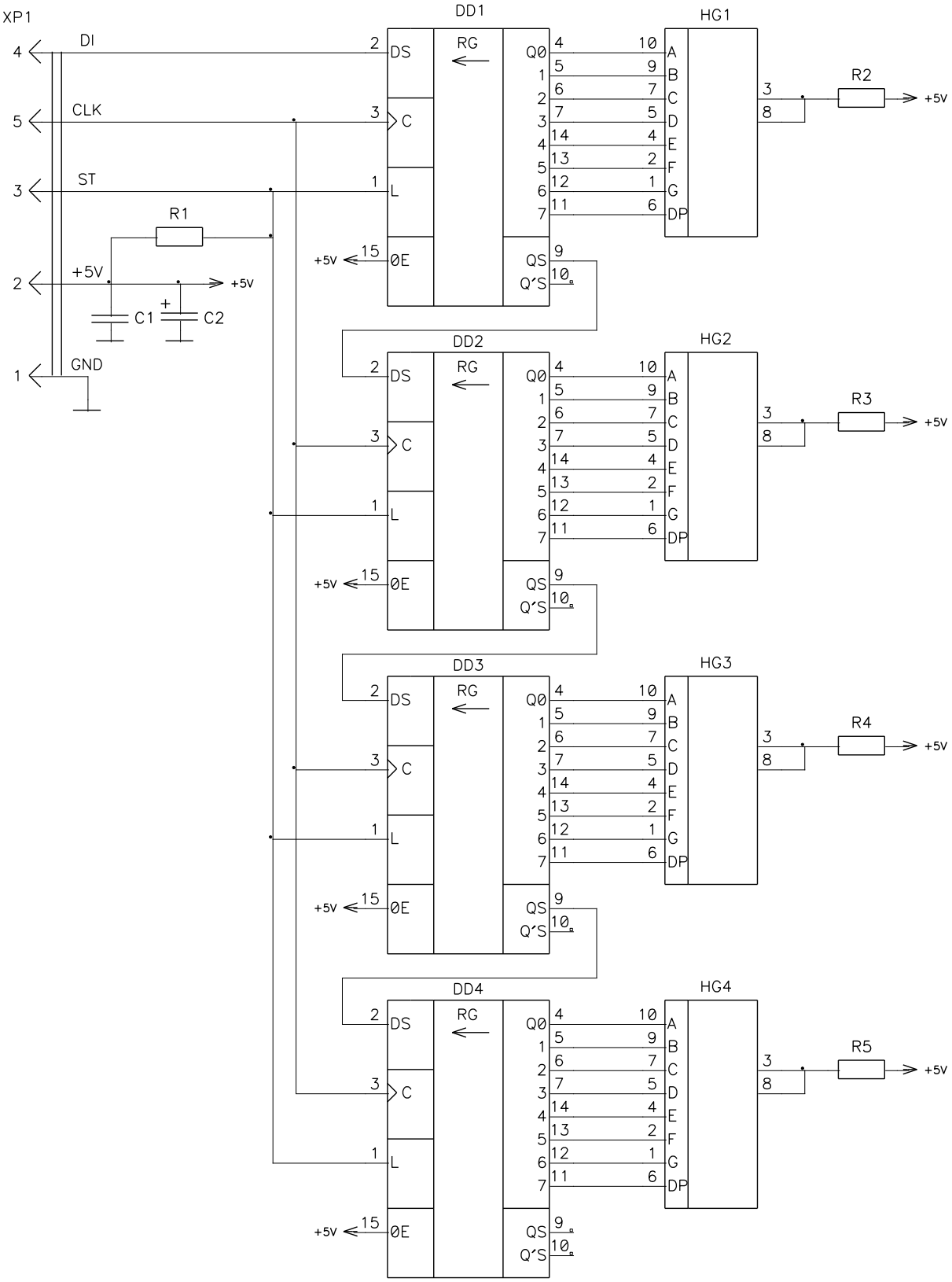


Рисунок 11.4 - Схема підключення роз'єму XP1 (статичний світлодіодний індикатор)

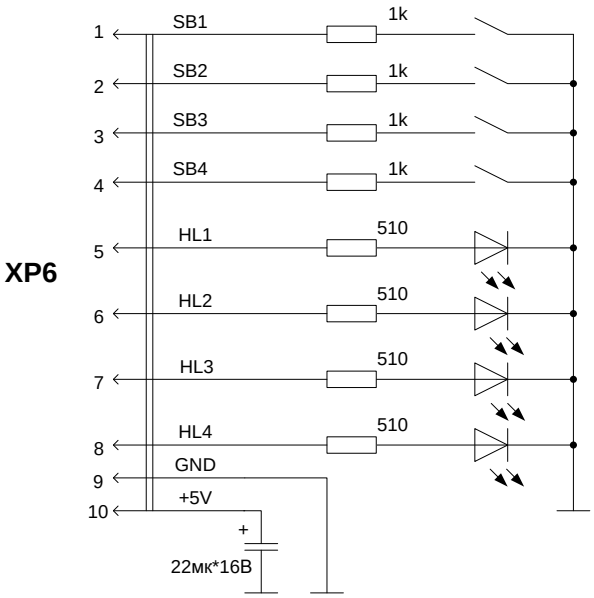


Рисунок 11.5 - Схема підключення роз'єму XP6 (кнопки та світлодіодні індикатори)

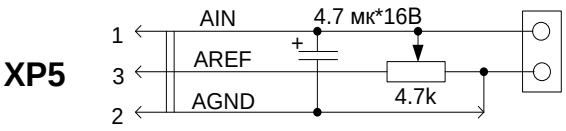


Рисунок 11.6 - Схема підключення роз'єму XP5 (змінний резистор до аналогового входу)

6. Програмування мікроконтролера

Програмування мікроконтролера здійснюється від персонального комп'ютера. Для цього необхідно з'єднати стенд і LPT-порт ком'ютера за допомогою кабеля, що входить в комплект поставки. Схема електрична принципова лабораторного стенда із вказанням всіх входів-виходів представлена на рис. 11.7.

Контрольні запитання

1. Розкажіть про основне призначення навчального стенду.
2. Які технічні характеристики навчального стенду ATmega8535?
3. Як здійснюється програмування навчального стенду?
4. Наведіть структурну схему навчального стенду.

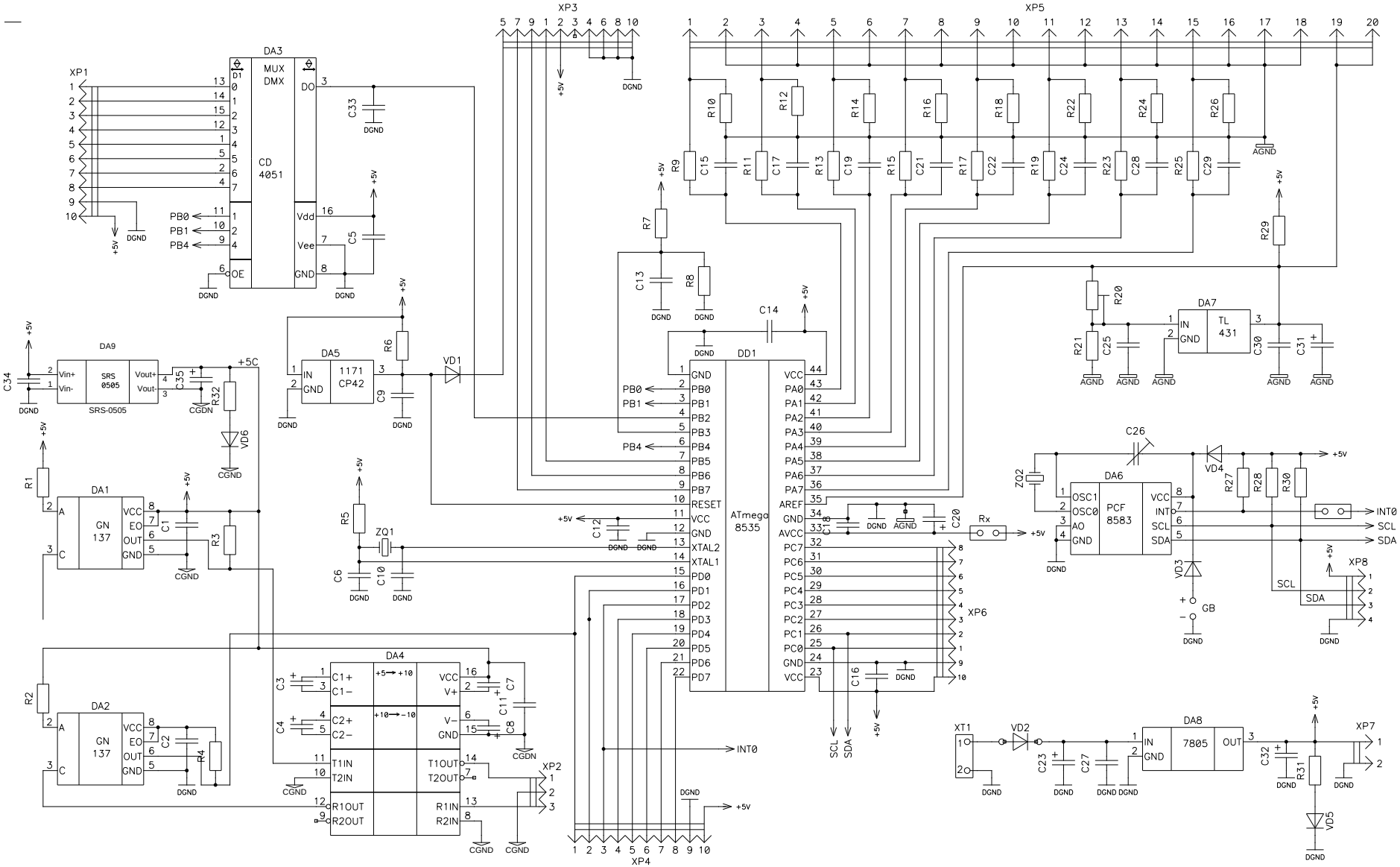


Рисунок 11.7 – Схема електрична принципова стенда

Лабораторна робота №12

ПРОГРАМУВАННЯ АНАЛОГО-ЦИФРОВОГО ПЕРЕТВОРЮВАЧА, ЩО ВБУДОВАНИЙ У МІКРОКОНТРОЛЕРІ ATmega8535

Мета роботи: вивчити принцип дії аналого-цифрового перетворювача, що вбудований у мікропроцесорний комплект навчального стенду, основні команди, навчитися відлагоджувати програму та прошивати її в мікроконтролер.

Теоретичні відомості

1. Принцип дії АЦП

АЦП перетворює вхідну аналогову напругу в 10-ти розрядний код методом послідовного наближеня. Мінімальне значення відповідає рівню землі (GND), а максимальне рівню опорної напруги (AREF) мінус 1 молодший розряд. До виходу AREF опційно може бути підключена напруга живлення мікроконтролера (AVCC) чи внутрішнє джерело опорної напруги (source of supporting tension, ДОН) на 1.22 В шляхом запису відповідних значень у біти REFSn у регістр ADMUX. Незважаючи на те, що ДОН на 2.56 В знаходиться усередині мікроконтролера, до його виходу може бути підключений блокувальний конденсатор для зниження чутливості до шумів, тому що він зв'язаний з виходом AREF.

Канал аналогового введення і каскад диференціального посилення вибираються шляхом запису біту MUX у регістр ADMUX. У якості однополярного аналогового входу АЦП може бути обраний один із входів ADC0...ADC7, а також GND і вихід фіксованого джерела опорної напруги 1,22 В. У режимі диференціального введення передбачена можливість вибору інвертуючих або неінвертуючих входів до диференціального підсилювача.

Якщо обраний диференціальний режим аналогового введення, то диференціальний підсилювач буде підсилювати різницю напруг між обраною парою входів на заданий коефіцієнт підсилення. Посилене в такий спосіб значення надходить на аналоговий вхід АЦП. Якщо вибирається однополярний режим аналогового введення, то каскад посилення пропускається.

Робота АЦП дозволяється шляхом установки біта ADEN у ADCSRA. Вибір опорного джерела і каналу перетворення не можливо виконати до установки ADEN. Якщо ADEN = 0, то АЦП не споживає струм, тому, при переводі в економічні режими сну рекомендується попередньо відключити АЦП.

АЦП генерує 10-розрядний результат, що міститься в парі регістрів даних АЦП ADCH і ADCL. За замовчуванням результат перетворення розміщається в молодших 10-ти розрядах 16-розрядного слова

(вирівнювання праворуч), але може бути опційно розміщений у старших 10-ти розрядах (вирівнювання ліворуч) шляхом установки біта ADLAR у регістрі ADMUX.

Практична користь представлення результату з вирівнюванням ліворуч існує тоді, коли достатньо 8-ми розрядної дозволяючої здатності, тому що в цьому випадку необхідно враховувати тільки регістр ADCH. В іншому ж випадку необхідно першим враховувати вміст регістра ADCL, а потім ADCH, чим гарантується, що обидва байти є результатом того самого перетворення. Як тільки виконане читання ADCL блокується доступ до регістрів даних з боку АЦП. Це означає, що якщо зчитаний ADCL і перетворення завершується перед читанням регістра ADCH, то жоден з регістрів не може модифікуватися і результат перетворення губиться. Після зчитування ADCH доступ до регістрів ADCH і ADCL з боку АЦП знову дозволяється.

АЦП генерує власний запит на переривання по завершенні перетворення. Якщо між читанням регістрів ADCH і ADCL заборонений доступ до даних для АЦП, то переривання виникне, навіть якщо результат перетворення буде загублений.

2. Запуск перетворювача

Одиноке перетворення запускається шляхом запису логічної одиниці у біт запуску перетворення АЦП ADSC. Даний біт залишається у високому стані в процесі перетворення і скидається по завершенні перетворення. Якщо в процесі перетворення переключається канал аналогового введення, то АЦП автоматично завершить поточне перетворення перш, ніж переключить канал.

У режимі автоматичного перезапуску АЦП безперервно оцифровує аналоговий сигнал і обновлює регістр даних АЦП. Даний режим задається шляхом запису логічної одиниці у біт ADFR регістра ADCSRA. Перше перетворення ініціюється шляхом запису логічної одиниці у біт ADSC регістра ADCSRA. У даному режимі АЦП виконує послідовні перетворення, незалежно від того скидається прапор переривання АЦП ADI чи ні.

3. Подільник (divider) АЦП

Якщо необхідна максимальна розрізнювальна здатність (10 розрядів), то частота на вході схеми послідовного наближення повинна бути в діапазоні від 50 до 200 кГц. Якщо достатньою є розрядність, яка менша 10-ти розрядів, але потрібно більш висока частота перетворення, то частота на вході АЦП може бути встановлена понад 200 кГц (рис. 12.1).

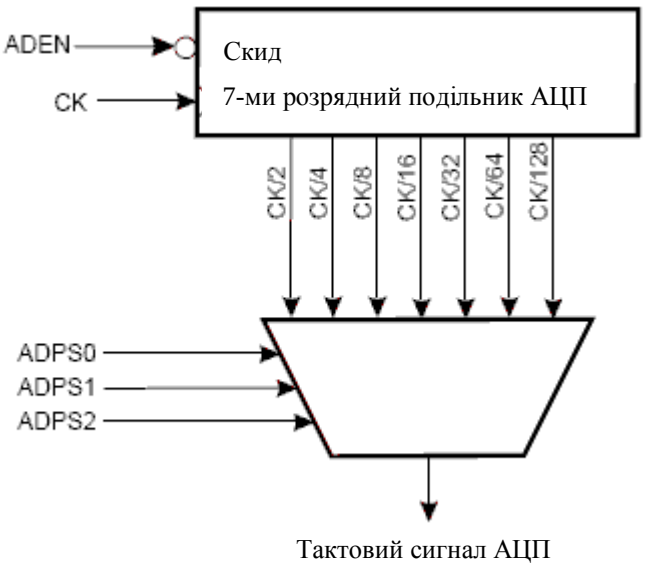


Рисунок 12.1 – Схема подільника АЦП

Модуль АЦП містить подільник, що формує похідні частоти понад 100 кГц стосовно частоти синхронізації ЦПУ. Коефіцієнт поділу встановлюється за допомогою біта ADPS у регістрі ADCSRA. Подільник починає рахунок з моменту включення АЦП установкою біта ADEN у регістрі ADCSRA. Подільник працює до тих пір, поки біт ADEN = 1 і скинутий, коли ADEN=0.

Якщо ініціюється однополярне перетворення установкою біта ADSC у регістрі ADCSRA, то перетворення починається з наступного наростаючого фронту тактового сигналу АЦП.

Нормальне перетворення вимагає 13 тактів синхронізації АЦП. Перше перетворення після включення АЦП вимагає 25 тактів синхронізації за рахунок необхідності ініціалі Тактовий сигнал АЦП еми (рис. 12.2).

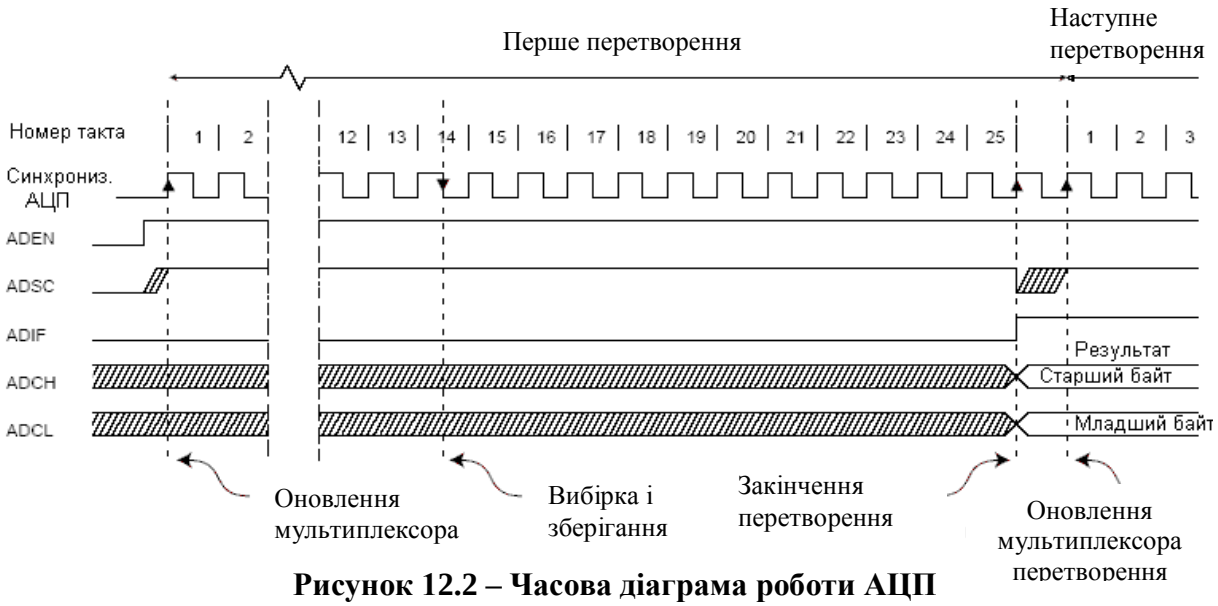


Рисунок 12.2 – Часова діаграма роботи АЦП при першому перетворення в режимі одиничного перетворення

Після початку нормального перетворення на вибірку-зберігання витрачається 1.5 такту синхронізації АЦП, а після початку першого перетворення - 13,5 тактів. По завершенні перетворення результат міститься в регістри даних АЦП і встановлюється прапор ADI. У режимі поодинокого перетворення одночасно скидається біт ADSC (рис. 12.3). Програмно біт ADSC може бути знову встановлений і нове перетворення буде ініційовано першим наростаючим фронтом тактового сигналу АЦП.

У режимі автоматичного перезапуску нове перетворення починається відразу по завершенні попереднього, при цьому ADSC залишається у високому стані (рис. 12.4).

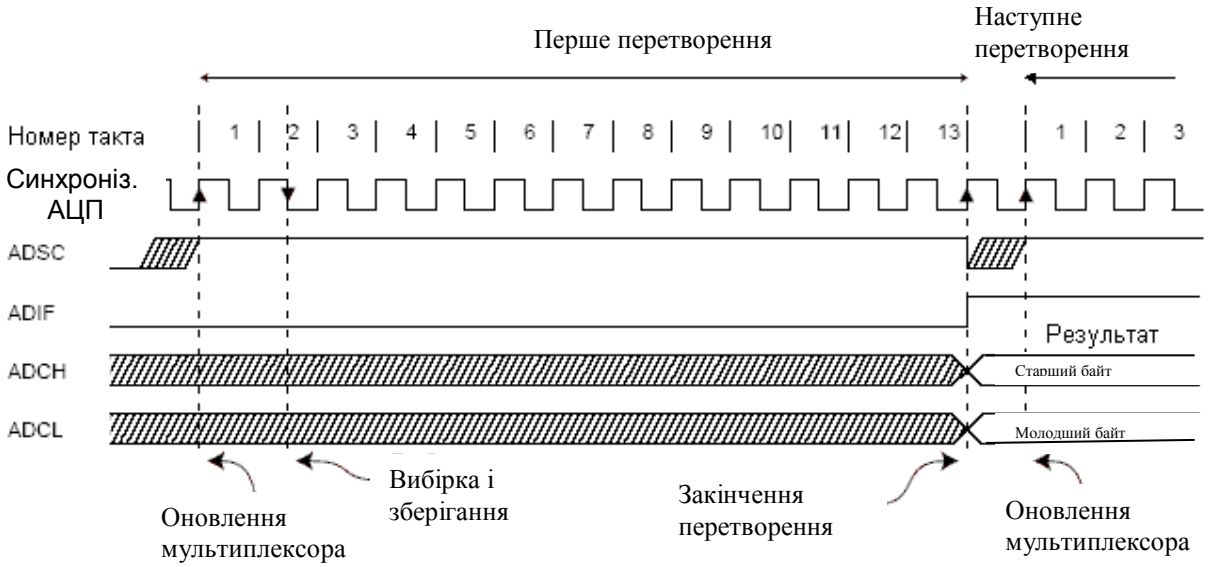


Рисунок 12.3 – Часова діаграма роботи АЦП при одиничному перетворенні

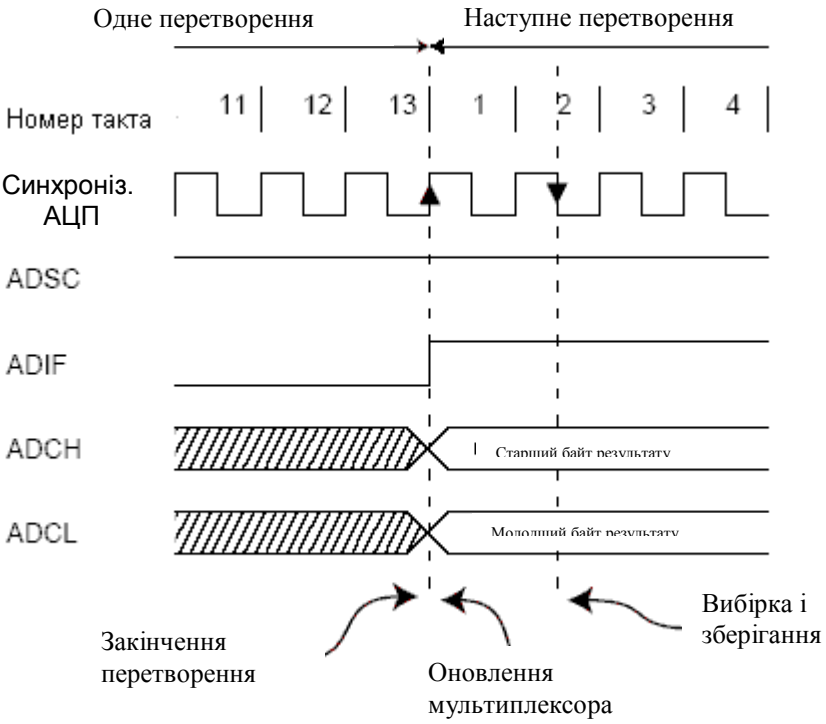


Рисунок 12.4 –Часова діаграма роботи АЦП при автоматичному перезапуску

4. Канали диференціального підсилювача

Якщо використовуються канали диференціального підсилювача, то необхідно прийняти в увагу деякі особливості.

Диференціальні перетворення синхронізовані по відношенню до внутрішньої синхронізації СКАЦП2, частота якого дорівнює половині частоти синхронізації АЦП. Дана синхронізація виконується автоматично інтерфейсом АЦП таким чином, щоб вибірка-зберігання ініціювалася визначеним фронтом СКАЦП2. Якщо перетворення (всі поодинокі перетворення і перше перетворення в режимі автоматичного перезапуску) ініціювалося користувачем, коли СКАЦП2 знаходився в низькому логічному стані, то його тривалість буде еквівалента однополярному перетворенню (13 тактів синхронізації АЦП). Якщо перетворення ініціюється користувачем, коли СКАЦП2 дорівнює логічній одиниці, то воно буде тривати 14 тактів синхронізації АЦП унаслідок роботи механізму синхронізації. У режимі автоматичного перезапуску нове перетворення ініціюється відразу по завершенні попереднього, а так як в цей момент СКАЦП2 дорівнює логічній одиниці, то всі перетворення, що були автоматично перезапущені (тобто всі, крім першого), будуть тривати 14 тактів синхронізації АЦП. Підсилювальний каскад оптимізований під частотний діапазон до 4 кГц для будь-яких коефіцієнтів підсилення. Посилення сигналів більш високих частот буде нелінійним. Тому, якщо вхідний сигнал містить частотні складові вище частотного діапазону підсилювального каскаду, то необхідно установити зовнішній фільтр низьких частот. Зверніть увагу, що частота синхронізації АЦП не зв'язана з обмеженням по частотному діапазоні підсилювального каскаду. Наприклад, період синхронізації АЦП може бути 6 мкс, при якому частота перетворення каналу дорівнює 12 тис. преретворень за секунду, незалежно від частотного діапазону цього каналу.

5. Зміна каналу або вибір опорного джерела

Біти MUXn і REFS1:0 у регістрі ADMUX підтримують одноступінчату буферизацию через тимчасовий регістр. Цим гарантується, що нові настроювання каналу перетворення й опорного джерела наберуть сили в безпечний момент для перетворення. До початку перетворення будь-які зміни каналу й опорного джерела набирають сили відразу після їхньої модифікації. Як тільки починається процес перетворення доступ до зміни каналу й опорного джерела блокується, чим гарантується достатність часу на перетворення для АЦП. Безперервність модифікації повертається на останньому такті АЦП перед завершенням перетворення (перед установкою прапора ADI у регістрі ADCSRA). Зверніть увагу, що перетворення починається наступним наростаючим фронтом тактового сигналу АЦП після запису ADSC. Таким чином, користувачу не рекомендується записувати нове значення каналу або опорного джерела в ADMUX до 1-го такту синхронізації АЦП після запису ADSC.

Особливі міри необхідно почати при зміні диференціального каналу. Як тільки здійснений вибір диференціального каналу підсилювальному каскаду потрібно 125 мкс для стабілізації нового значення. Отже, протягом перших після переключення диференціального каналу 125 мкс не повинне стартувати перетворення. Якщо ж у цей період перетворення усе-таки виконувалися, то їхній результат необхідно ігнорувати.

Таку ж затримку на встановлення необхідно ввести при першому диференціальному перетворенні після зміни опорного джерела АЦП (за рахунок зміни біта REFS1:0 у ADMUX).

Якщо дозволена робота інтерфейсу JTAG, то функції каналів АЦП на висновках порту F 7...4 скасовується.

6. Вхідні канали АЦП

При переключенні вхідного каналу необхідно врахувати деякі рекомендації, що виключать некоректність переключення.

У режимі одиночного перетворення переключення каналу необхідно виконувати перед початком перетворення. Переключення каналу може відбутися тільки протягом одного такту синхронізації АЦП після запису логічної одиниці у ADSC. Однак найпростішим методом є чекання завершення перетворення перед вибором нового каналу.

У режимі автоматичного перезапуску канал необхідно вибирати перед початком першого перетворення. Переключення каналу відбувається аналогічно - протягом одного такту синхронізації АЦП після запису логічної одиниці у ADSC. Але найпростішим методом є чекання завершення перового перетворення, а потім переключення каналу. Оскільки наступне перетворення вже запущене автоматично, то наступний результат буде відповідати попередньому каналу. Наступні перетворення відбивають результат для нового каналу.

При переключенні на диференціальний канал перше перетворення буде характеризуватися поганою точністю через перехідний процес у схемі автоматичного регулювання зсуву. Отже, перший результат такого перетворення рекомендується ігнорувати.

7. Джерело опорної напруги АЦП

Джерело опорної напруги (ДОН) для АЦП ($V_{ДОН}$) визначає діапазон перетворення АЦП. Якщо рівень однополярного сигналу більший за $V_{ДОН}$, то результатом перетворення буде 0x3FF. У якості $V_{ДОН}$ можуть виступати AVCC, внутрішнє ДОН 2,56 В чи зовнішне ДОН, підключене до виводу AREF. AVCC підключається до АЦП через пасивний ключ. Внутрішня опорна напруга 2,56 В генерується внутрішнім еталонним джерелом VBG, буферизованного внутрішнім підсилювачем. У будь-якому випадку зовнішній вивід AREF зв'язаний безпосередньо з АЦП і, тому, можна знизити вплив шумів на опорне джерело за рахунок підключення конденсатора між виводом AREF і загальним. Напруга $V_{ДОН}$ також може

бути виміряна на виводі AREF високоомним вольтметром. Зверніть увагу, що $V_{ДОН}$ є високоомним джерелом і, тому, зовні до нього може бути підключене тільки ємнісне навантаження.

Якщо користувач використовує зовнішнє опорне джерело, підключений до виводу AREF, то не допускається використання іншої опції опорного джерела, тому що це приведе до шунтування зовнішньої опорної напруги. Якщо до виводу AREF не прикладена напруга, то користувач може вибрати AVCC і 2.56 В в якості опорного джерела. Результат першого перетворення після переключення опорного джерела може характеризуватися поганою точністю і користувачу рекомендується його ігнорувати.

Якщо використовуються диференціальні канали, то обране опорне джерело повинне бути менше рівня AVCC.

8. Подавитель шумів АЦП

АЦП характеризується можливістю придушення шумів, що викликані роботою ядра ЦПУ і периферійних пристроїв введення-виведення. Подавитель шумів може бути використаний у режимі зниження шумів АЦП і в режимі холостого ходу. При використанні даної функції необхідно дотримувати наступної процедури:

а) Переконаєтеся, що робота АЦП дозволена і він не виконує перетворення. Виберіть режим одиночного перетворення і дозвольте переривання по завершенні перетворення.

б) Введіть режим зменшення шумів АЦП (чи режим холостого ходу). АЦП запустить перетворення як тільки зупиниться ЦПУ.

в) Якщо до завершення перетворення не виникає інших переривань, то АЦП викликає переривання ЦПУ і програма перейде на вектор обробки переривання по завершенні перетворення АЦП. Якщо до завершення перетворення інше переривання будить мікроконтролер, то це переривання обробляється, а по завершенні перетворення генерується відповідний запит на переривання. АЦП залишається в активному режимі поки не буде виконана чергова команда sleep.

Зверніть увагу, що АЦП не відключається автоматично при перекладі в усі режими сну, крім режиму холостого ходу і зниження шумів АЦП. Тому, користувач повинен передбачити запис логічного нуля у біт ADEN перед переводом у такі режими сну щоб уникнути надмірного енергоспоживання. Якщо робота АЦП була дозволена в таких режимах сну і користувач бажає виконати диференційне перетворення, то після пробудження необхідно включити, а потім виключити АЦП для ініціації розширеного перетворення, чим буде гарантоване одержання дійсного результату.

9. Схема аналогового входу

Схема аналогового входу для однополярних каналів представлена на рис. 12.5. Незалежно від того, який канал підключений до АЦП, аналоговий сигнал, підключений до виводу ADCn, навантажується ємністю виводу і вхідним опором витоку. Після підключення каналу до АЦП аналоговий сигнал буде зв'язаний з конденсатором вибірки-зберігання через послідовний резистор, опір якого еквівалентний усьому вхідному ланцюгу.

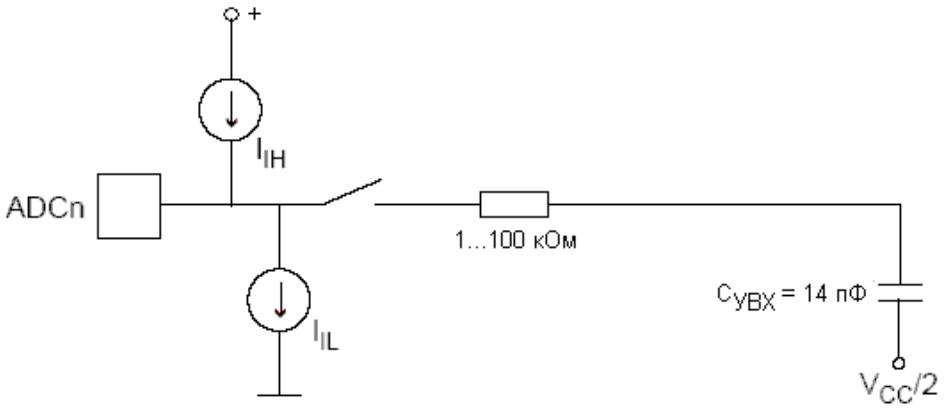


Рисунок 12.5 – Схема аналогового входу

АЦП оптимізований під аналогові сигнали з вихідним опором не більше 10 kОм. Якщо використовується таке джерело сигналу, то час вибірки невеликий. Якщо ж використовується джерело з більш високим вхідним опором, то час вибірки буде визначатися часом, що потрібно для зарядки конденсатора вибірки-зберігання (condenser of selection-storage) джерелом аналогового сигналу. Рекомендується використовувати джерела тільки з малим вихідним опором і сигналами, що повільно змінюються, тому що в цьому випадку буде досить швидким заряд конденсатора вибірки-зберігання.

Стосовно каналів з диференціальним підсиленням рекомендується використовувати сигнали з внутрішнім опором до декількох сотень кОм. Варто передбачити, щоб у попередніх каскадах формування аналогового сигналу до входу АЦП не вносилися частоти вище $f_{ацп}/2$, у противному випадку результат перетворення може бути некоректним. Якщо імовірність проникнення високих частот існує, то рекомендується перед АЦП установити фільтр низьких частот.

10. Рекомендації щодо зниження впливу шумів на результат перетворення

Робота цифрових вузлів усередині і зовні мікроконтролера зв'язана з генерацією електромагнітних випромінювань, що можуть негативно позначитися на точності вимірювання аналогового сигналу. Якщо точність перетворення є критичним параметром, то рівень шумів можна знизити, дотримуючи наступних рекомендацій:

а) Виконуйте шлях аналогових сигналів як можна більш коротким. Стежите, щоб аналогові сигнали проходили над площиною (шаром) з аналоговою землею (екраном) і далеко від провідників, що передають високочастотні цифрові сигнали.

б) Вивід AVCC необхідно зв'язати з цифровим живленням VCC через LC-ланцюг відповідно до рис. 12.6.

в) Використовуйте функцію придушення шумів АЦП, внесених роботою ядра ЦПУ.

г) Якщо який-небудь з виводів АЦП використовується як цифровий вихід, то надзвичайно важливо не допустити переключення стану цього виходу в процесі перетворення.

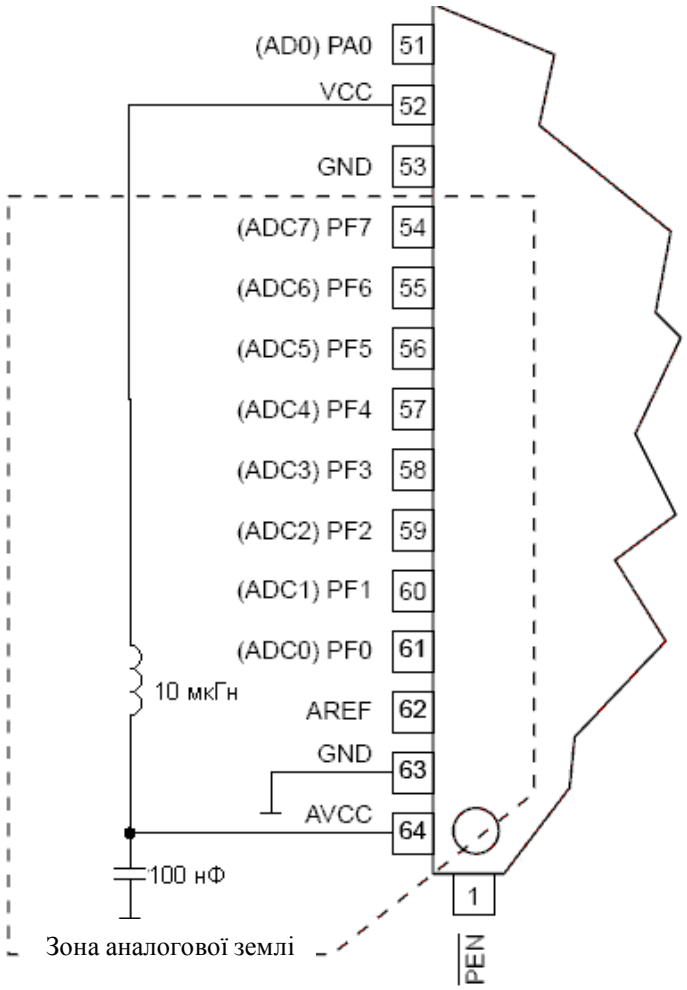


Рисунок 12.6 – Схема підключення живлення АЦП

11. Методи компенсації зсуву

Підсилювальний каскад має вбудовану схему компенсації зсуву, що прагне максимально наблизити до нуля зсув диференційного вимірювання. Зсув, що залишився, можна виміряти, якщо як диференційні входи АЦП вибрати той самий вивід мікроконтролера. Вимірюваний в такий спосіб залишковий зсув можна програмно відняти від результату перетворення. Використання програмного алгоритму корекції зсуву дозволяє зменшити зсув нижче одного молодшого розряду.

12. Визначення похибок аналогово-цифрового перетворення

n-розрядний однополярний АЦП перетворює лінійно напругу між GND і $V_{ДОН}$ з кількістю кроків 2n (молодших розрядів). Мінімальний код = 0, максимальний = 2n-1. Основні похибки перетворення є відхиленням реальної функції перетворення від ідеальної. До них відносяться:

зсув - відхилення першого переходу (з 0x000 на 0x001) у порівнянні з ідеальним переходом (тобто при 0.5 молодшого розряду). Ідеальне значення : 0 молодшого розряду (рис. 12.7).

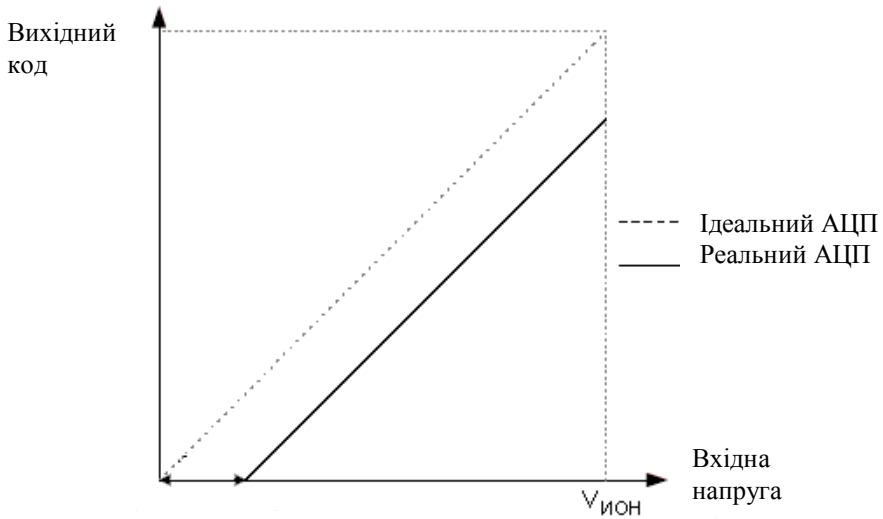


Рисунок 12.7 – Похибка зсуву АЦП

Похибка підсилення. Після коректування зсуву похибка підсилення являє собою відхилення останнього переходу (з 0x3FE на 0x3FF) від ідеального переходу (тобто відхилення при максимальному значенні мінус 1,5 молодшого розряду). Ідеальне значення: 0 молодшого розряду (рис.12.8).

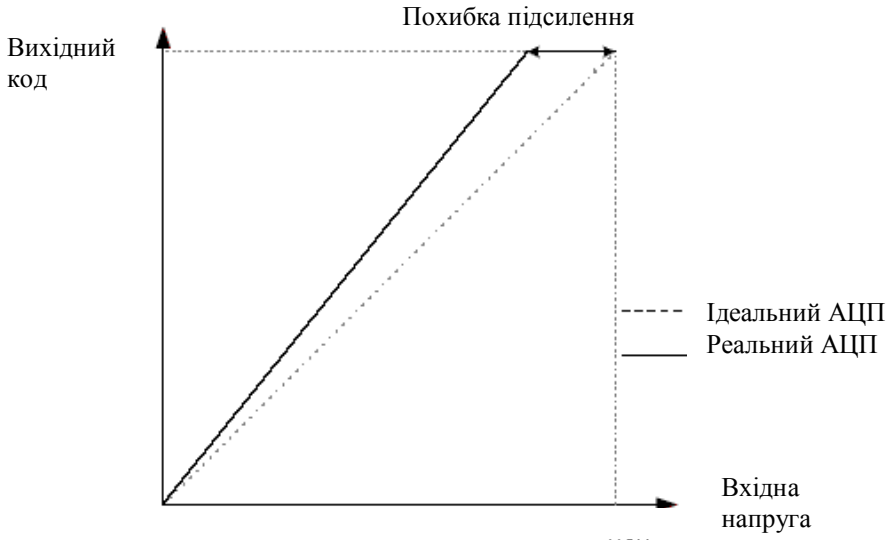


Рисунок 12.8 – Похибка підсилення

Інтегральна нелінійність (ІН). Після коректування зсуву і похибки підсилення ІН являє собою максимальне відхилення реальної функції перетворення від ідеальної для будь-якого коду. Ідеальне значення ІН = 0 молодшого розряду (рис. 12.9).

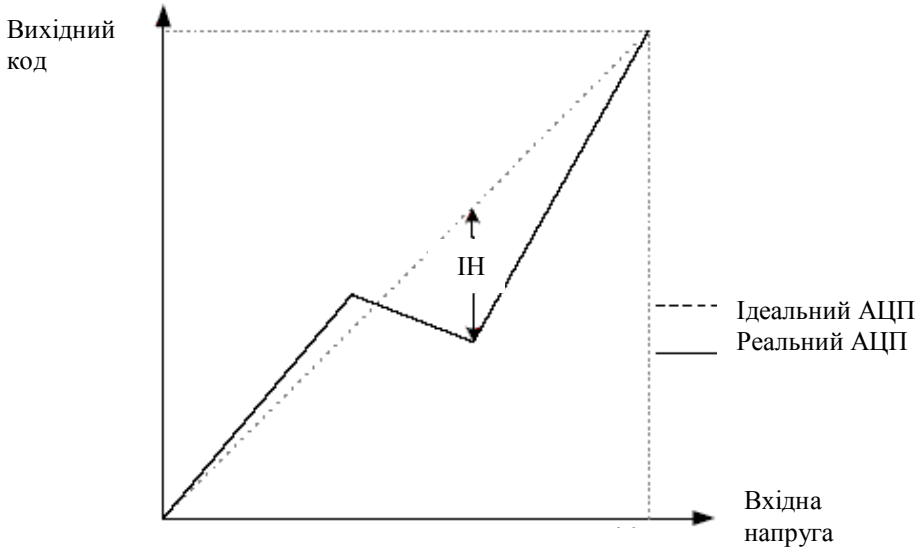


Рисунок 12.9 – Інтергальна нелінійність

Диференційна нелінійність (ДНЛ). Максимальне відхилення між шириною фактичного коду (інтервал між двома суміжними переходами) від ширини ідеального коду (1 молодшого розряду). Ідеальне значення: 0 молодшого розряду (рис. 12.10).

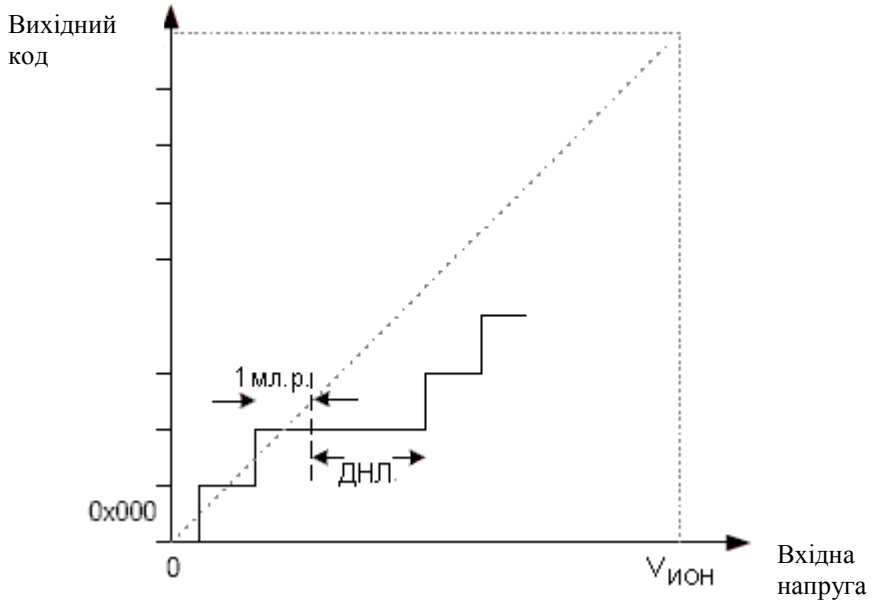


Рисунок 12.10 – Диференційна нелінійність

Похибка квантування. Виникає через перетворення вхідної напруги в кінцеве число кодів. Похибка квантування - інтервал вхідної напруги

довжиною 1 молодший розряд (крок квантування по напрузі), що характеризується тим самим кодом. Завжди дорівнює ±0.5 молодшого розряду.

Абсолютна похибка. Максимальне відхилення реальної (без підстроювання) функції перетворення від реальної при будь-якому коді. Є результатом дії декількох ефектів: зсуву, похибки підсилення, диференціцної похибки, нелінійності і похибки квантування. Ідеальне значення: ±0.5 молодшого розряду.

13. Результат перетворення АЦП

По завершенні перетворення (ADI = 1) результат може бути зчитаний з пари регістрів результату перетворення АЦП (ADCL, ADCH).

Для однополярного перетворення:

$$V_{ADC} = \frac{V_{вх} 1024}{V_{ДОН}}, \qquad (12.1)$$

де $V_{вх}$ – рівень напруги, який прикладений до входу АЦП;

$V_{ДОН}$ – напруга вибраного джерела опорної напруги.

Код 0x000 відповідає рівню аналогової землі, а 0x3FF - рівню напруги ДОН мінус 1 крок квантування по напрузі.

При використанні диференційного каналу:

$$V_{ADC} = \frac{512(V_{POS} - V_{NEG})GAIN}{V_{ДОН}}, \qquad (12.2)$$

де V_{pos} – напруга на неінвертуючому (інвертуючому) вході;

V_{NEG} – коефіцієнт підсилення.

Результат подається в коді двійкового доповнення, починаючи з 0x200 (-512d) до 0x1FF (+511d). Зверніть увагу, що при необхідності швидко визначити полярність результату досить опитати старший біт результату перетворення (ADC9 у ADCH). Якщо даний біт дорівнює логічній одиниці, то результат негативний, якщо ж логічному нулю, то позитивний.

У табл. 12.1 представлені результуючі вихідні коди для диференціальної пари каналів (ADCn - ADCm) з коефіцієнтом підсилення $К_у$ й опорною напругою $V_{ДОН}$.

Таблиця 12.1 – Зв'язок між вхідною напругою и вихідними кодами

$V_{АЦПn}$	Зчитуємий код	Відповідне десяткове значення
$V_{АЦПm} + V_{ДОН}$ /Ку	0x1FF	511
$V_{АЦПm} + 0.999\ V_{ДОН}$ / Ку	0x1FF	511
$V_{АЦПn} + 0.998\ V_{ДОН}$ / Ку	0x1FE	510
…	...	...
$V_{АЦПm} + 0.001\ V_{ДОН}$ / Ку	0x001	1
$V_{АЦПm}$	0x000	0
$V_{АЦПm} - 0.001\ V_{ДОН}$ / Ку	0x3FF	-1
…	...	...
$V_{АЦПm} - 0.999\ V_{ДОН}$ / Ку	0x201	-511
$V_{АЦПm} – V_{ДОН}$ / Ку	0x200	-512

14. Регістр керування мультиплексором АЦП- ADMUX

Таблиця 12.2 – Команди керування мультиплексором АЦП- ADMUX

Розряд	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Чит./зап	Чит./Зап	Чит./Зап.	Чит./Зап.	Чит./Зап.	Чит./Зап.	Чит./Зап.	Чит./Зап.	Чит./Зап.
Поч. Значеня	0	0	0	0	0	0	0	0

Розряд 7: 6 - REFS1:0: Біти вибору джерела опорної напруги

Дані біти визначають яка напруга буде використовуватися в якості опорної для АЦП (табл. 12.3). Якщо змінити значення даних біт у процесі перетворення, то нові установки наберуть сили тільки по закінченні поточного перетворення (тобто коли установиться біт ADI у регістрі ADCSRA). Внутрішнє ДОН можна не використовувати, якщо до виводу AREF підключене зовнішнє опорне джерело.

Таблиця 12.3 - Вибір опорного джерела АЦП

REFS1	REFS0	**Опорне джерело**
0	0	AREF, внутрішнє $V_{ДОН}$ відключене
0	1	AVCC із зовнішнім конденсатором на виводі AREF
1	0	Зарезервовано
1	1	Внутрішнє джерело опорної напруги 2.56 Віз зовнішнім Конденсатором на виводі AREF

Розряд 5 - ADLAR: Біт керування представленням результату перетворення

Біт ADLAR впливає на представлення результату перетворення в парі регістрів результату перетворення АЦП. Якщо ADLAR = 1, то результат перетворення буде мати лівосторонній формат, у противному випадку - правобічний. Дія біта ADLAR набирає сили відразу після зміни, незалежно від перетворення, що паралельно виконується. Повний опис дії даного біта представлено в "Регістри даних АЦП - ADCL і ADCH".

Розряд 4:0 - MUX4:0: Біти вибору аналогового каналу і коефіцієнта підсилення

Дані біти визначають які з наявних аналогових входів підключаються до АЦП. Крім того, з їхньою допомогою можна вибрати коефіцієнт підсилення для диференційних каналів (табл. 12.4). Якщо значення біт змінити в процесі перетворення, то механізм їхньої дії набере сили тільки після завершення поточного перетворення (після установки біта ADI у регістрі ADCSRA).

Таблиця 12.4 - Вибір вхідного каналу і коефіцієнта підсилення

MUX4..0	Однополярний вхід	Неінвертуючий диференційний вхід	Інвертуючий диференційний вхід	Коефіцієнт підсилення, Ку
00000	ADC0	Немає		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	Немає	ADC0	ADC0	10
01001		ADC1	ADC0	10
01010		ADC0	ADC0	200
01011		ADC1	ADC0	200
01100		ADC2	ADC2	10
01101		ADC3	ADC2	10
01110		ADC2	ADC2	200
01111		ADC3	ADC2	200
10000		ADC0	ADC1	1
10001		ADC1	ADC1	1
10010		ADC2	ADC1	1

Продовження таблиці 12.4

10011	Немає	ADC3	ADC1	1
10100		ADC4	ADC1	1
10101		ADC5	ADC1	1
10110		ADC6	ADC1	1
10111		ADC7	ADC1	1
11000		ADC0	ADC2	1
11001		ADC1	ADC2	1
11010		ADC2	ADC2	1
11011		ADC3	ADC2	1
11100		ADC4	ADC2	1
11101		ADC5ADC21, Немає111110B(GND)		
11110	1.23B			

15. Регістр А управління і статуса АЦП – ADCSRA

Таблиця 12.5 - Регістр А управління і статуса АЦП

Разряд	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Читання/запис	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.
Поч. Значення	0	0	0	0	0	0	0	0

Розряд 7 - ADEN: Дозвіл роботи АЦП

Запис у даний біт логічної одиниці дозволяє роботу АЦП. Якщо в даний біт записати логічний ноль, то АЦП відключається, навіть якщо він знаходився в процесі перетворення.

Розряд 6 - ADSC: Запуск перетворення АЦП

У режимі одиночного перетворення установка даного біта ініціює старт кожного перетворення. У режимі автоматичного перезапуску установкою цього біта ініціюється тільки перше перетворення, а всі інші виконуються автоматично. Перше перетворення після дозволу роботи АЦП, ініційоване битому ADSC, виконується по розширеному алгоритмі і триває 25 тактів синхронізації АЦП, замість звичайних 13 тактів. Це зв'язано з необхідністю ініціалізації АЦП.

У процесі перетворення при опитуванні біта ADSC повертається логічна одиниця, а по завершенні перетворення – логічний нуль. Запис логічного нуля у даний біт не передбачене і не робить ніякої дії.

Розряд 5 - ADFR: Вибір режиму автоматичного перезапуску АЦП

Якщо в даний біт записати логічну одиницю, то АЦП перейде в режим автоматичного перезапуску. У цьому режимі АЦП автоматично виконує перетворення і модифікує регістри результату перетворення через фіксовані проміжки часу. Запис логічного нуля у цей біт припиняє роботу в даному режимі.

Розряд 4 - ADI: Прапор переривання АЦП

Даний прапор установлюється після завершення перетворення АЦП і відновлення регістрів даних. Якщо встановлені біти ADI і I (регістр SREG), то відбувається переривання по завершенні перетворення. Прапор ADI скидається апаратно при переході на відповідний вектор переривання. Альтернативно прапор ADI скидається шляхом запису логічної одиниці у нього. Зверніть увагу, що при виконанні команди "читання-модифікація-запис" з регістром ADCSRA очікуване переривання може бути відключено. Дане також поширюється на використання інструкцій SBI і CBI.

Розряд 3 - ADI: Дозвіл переривання АЦП

Після запису логічної одиниці у цей біт, за умови, що встановлено біт I в регістрі SREG, дозволяється переривання по завершенні перетворення АЦП.

Розряди 2:0 - ADPS2:0: Біти керування подільником АЦП

Дані біти визначають на яке значення тактова частота ЦПУ буде відрізнятися від частоти вхідної синхронізації АЦП (табл. 12.5).

Таблиця 12.5 - Керування перериваннями АЦП

ADPS2	ADPS1	ADPS0	Коэффициент ділення
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

16. Регістри даних АЦП - ADCL і ADCH

По закінченні перетворення результат міститься в цих двох регістрах. При використанні диференціального режиму перетворення результат представляється в коді двійкового доповнення.

Якщо виконане читання ADCL, то доступ до цих регістрів для АЦП буде заблокований (тобто АЦП не зможе надалі модифікувати результат перетворення), поки не буде зчитпний регістр ADCH.

Таблиця 12.6 - ADLAR = 0:

Розряд	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	**ADCH**
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	**ADCL**
	7	6	5	4	3	2	1	0	
Читання/запис	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
Поч. Значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Лівосторонній формат представлення результату зручно використовувати, якщо досить 8 розрядів. У цьому випадку 8-розрядний результат зберігається в регістрі ADCH і, отже, читання регістра ADCL можна не виконувати. При правобічному форматі необхідно спочатку зчитувати ADCL, а потім ADCH.

Таблиця 12.7 – ADLAR = 1:

Розряд	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	**ADCH**
	ADC1	ADC0	-	-	-	-	-	-	**ADCL**
	7	6	5	4	3	2	1	0	
Читання/запис	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	Чит.	
Поч. Значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

17. ADC9:0: Результат перетворення АЦП

Дані біти представляють результат перетворення.

18. Приклад лістінгу програми для програмування АЦП мікроконтролера на мові Сі, при подачі на порт PA0 напруги, яку можна змінювати від 0 до 5 В

```
#include <90s8535.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x00
interrupt [ADC_INT] void adc_isr(void) {
PORTB=(unsigned char) ~ADCW;
delay_ms(20);
ADCSR|=0x40;
}
void main(void) {
PORTB=0xFF;
DDRB=0xFF;
ADCSR=0x8E;
#asm("sei")
ADMUX=0;
ADCSR|=0x40;
while (1);
}
```

Завдання до лабораторної роботи

1. Напишіть програму для роботи АЦП мікроконтролкра, яка б при зміні положення (від мінімального до максимального значення) ручки потенціометра, на лабораторному стенді, виводила перетворені значення напруги на семисегментну статичну індикацію.

2. Прошийте написану працюючу програму в мікроконтролер, використовуючи при цьому наявний в комплекті кабель для LPT порту.

3. Продемонструйте роботу навчального стенду.

Зміст звіту

Завдання до лабораторної роботи.

Лістінг програми на мові С з поясненнями.

Результати роботи навчального стенду.

Контрольні запитання

1. Принцип дії та основні характеристики АЦП мікроконтролера Atmegs8535.

2. Наведіль схему та поясніть принцтп дії подільника АЦП.

3. Що Ви знаєте про джерело опорної напруги АЦП?

4. Наведіть схему аналогового входу АЦП, розкажіть про його основні вимоги та наведіть рівняння перетворення АЦП.

5. Перерахуйте основні похибки аналого-цифрового перетворення та розкажіть як вони визначаються.

Лабораторна робота № 13

ОЗНАЙОМЛЕННЯ З ПРОГРАМНИМ ПАКЕТОМ PROTEUS VSM ТА ДОСЛІДЖЕННЯ СИНТЕЗОВАНОГО ПРИСТРОЮ

Мета роботи: Ознайомитись з програмним пакетом PROTEUS VSM. В ході ознайомлення навчилися синтезувати та аналізувати створений пристрій, відкривати проект, запускати і зупиняти симуляцію, видаляти компоненти зі схеми і додавати нові, змінювати властивості компонентів.

Теоретичні відомості

1. Можливості пакету Proteus VSM

Даний програмний продукт дозволяє малювати схеми і виконує наступні функції: автоматичне розведення доріжок та автоматичне розміщення елементів схеми на друкованій платі.

Програма ISIS призначена для виконання принципових схем будь – якої складності та подальшої їх перевірки за допомогою емулятора, який входить до складу програми. Емулятор роботи схеми має власну бібліотеку елементів (які використовуються для малювання) до складу якої входить велика кількість елементів, включаючи мікроконтролери фірми AVR Atmel, та ще багато чого.

Основні властивості програмного продукту Proteus PROFESSIONAL:

- підтримка операційних системах Windows 98/Me/2k/XP та пізніших;
- автоматичне розведення доріжок та виставлення з’єднань;
- потужний інструмент виділення та редагування параметрів елементів;
- повна підтримка шин, включаючи виводи елементів, порти та дроти;
- багато матеріалів та можливостей перевірки;
- генерація списків з'єднань для підтримки усіх улюблених програм розведення плат друкованого монтажу;
- можливість використання багатьох схем з різних документів одночасно;
- генерація псивдо елементів з усіх документів.

Програмний емулятор дозволяє працювати з будь-якими мікропроцесорними системами. До основного складу можливостей емулятора варто віднести такі: праця з мікропроцесорами; наявність індикаторів, віртуальних терміналів, віртуальних приладів для відстеження параметрів схеми; можливість створення/редагування різноманітних елементів написаних на мові С++, вони можуть бути як електричними так і

графічними; праця з діодами (бібліотека DIODE), транзисторами (бібліотека BIPOLAR), тощо.

Що ж до AREAS, то до його переваг можна віднести наявність таких необхідних функцій як: автоматичне розміщення елементів (automatic placing of elements), автоматичне розведення доріжок друкованої плати, зміна масштабу до 1 нм та можливість регулювати кути з точністю до 0,1 градуса, використання списків з'єднань згенерованих за допомогою ISIS, в бібліотеці елементів присутні як звичайні так і СМД компоненти, можливість зберігання у різних форматах в тому числі і BMP.

В цілому, за допомогою цього програмного продукту, можливе не лише складання схем для друку та подальшого розведення доріжок, а й перевірка схеми за допомогою емулятора. Сам емулятор може використовувати не лише елементи з власної бібліотеки (яка на сайті постійно оновлюється), а створювати власні.

2. Основні етапи роботи з програмою ISIS

Після запуску програми ми бачимо головне вікно в якому ми можемо скласти схему (рис. 13.1).

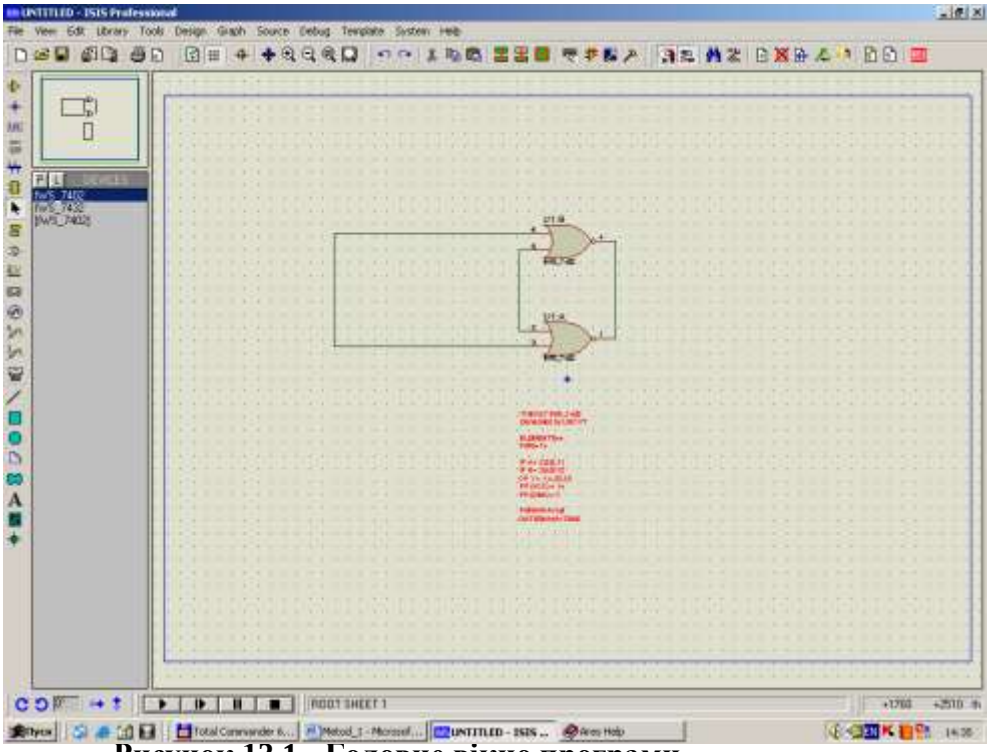
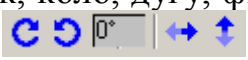


Рисунок 13.1 - Головне вікно програми

Інтерфейс програми здебільшого інтуїтивно зрозумілий для користувача програмними продуктами *Microsoft*. Основне вікно програми, з яким нам і доведеться працювати більшу частину часу, складається з наступних елементів головного меню та робочих панелей, які можна переміщувати по екрану, бібліотеки окремих елементів (які користувач вибирає з загальної бібліотеки), кнопок запуску/зупинки емулятора,

зменшеного варіанта робочого простору програми, кнопки для перевертання будь-якого елемента.

Робочі панелі містять наступні функціональні кнопки:  - створити новий робочий простір,  - відкрити вже існуючий файл,  - зберегти поточний документ,  - імпортувати та експортувати виділені фрагменти як документ,  - друкувати та виділити фрагмент для друку,  - оновити зображення,  - показати сітку,  показує список маркерів в *Object Selector* який ви можете потім вибрати та використовувати для малювання,  - переміщення екрана відносно вибраної точки,  - зменшити/збільшити масштаб зображення,  - показати весь робочий простір та виділити ділянку для збільшення,  - відмінити та повернути,  - вирізати, копіювати, вставити,  - копіювати/пересунути/знищити виділений об'єкт,  - вибрати/додати елемент до окремої бібліотеки, створити з виділеного об'єкту елемент, ввімкнути редагування вже існуючого елементу, розгрупувати виділений об'єкт,  - ввімкнути присипання курсора до виводів, ввімкнути автоматичне проведення з'єднання,  - знайти тег компонента який відповідає заданим умовам,  - створити/видалити/перейти на аркуш,  - перейти на аркуш, який знаходиться на нижчому рівні, перейти на головний аркуш,  - створити специфікацію, перевірити електричні з'єднання,  - створити список з'єднань та перейти в програму AREAS,  - додати прилад,  - точка з'єднання,  - додати назву шині,  - додати скрипт,  - показати з'єднання, чи провести шину,  - використовується для приєднання аркушів нижчого рівня до аркушів які знаходяться на вищому рівні,  - виділяти/редагувати об'єкти,  - додати між аркушеві зв'язки,  - додати вивід елемента,  - додати графічні прилади (аналізатор спектра та ін.),  - диктофон,  - додати генератори,  - вольтметр та амперметр,  - додати віртуальні прилади (на зразок осцилографа),  - намалювати риску, прямокутник, коло, дугу, фігуру,  - додати напис,  - додати символ,  - маркери,  - повернути на відповідний градус, віддзеркалити по вертикалі чи горизонталі, - керування емулятором.

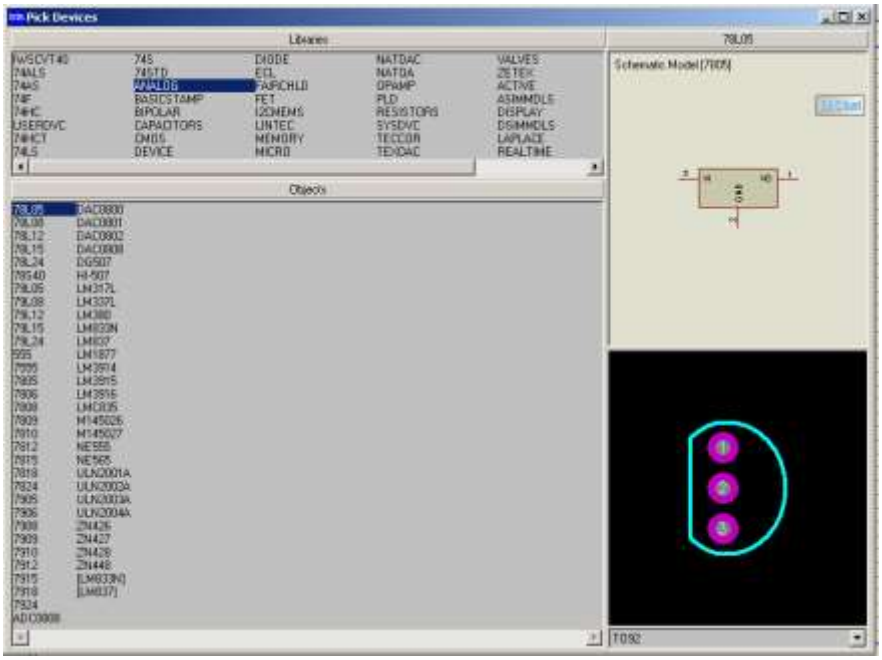


Рисунок 13.2 - Власна бібліотека програми

Перейдемо до розгляду питання контролю електричних параметрів схеми. Для цього в програмі передбачена величезна кількість вимірювальних приладів. Серед них маємо доступ до наступних: осцилограф, логічний аналізатор, лічильник/таймер, віртуальний термінал, генератор зображення, генератор різноманітних сигналів, вольтметри постійної та змінної напругу, амперметри змінного та постійного струму. Роздивимось деякі з приладів детальніше. Так, коли ми хочемо використати віртуальний генератор сигналів виносимо його на робочий простір, під'єднуємо до відповідних точок схеми та запускаємо емулятор. Після чого відкривається діалогове вікно генератора сигналів, в якому власне і вибираємо параметри потрібного сигналу. Серед яких доступно регулювання частоти, амплітуди, форми сигналу та його полярність. Про осцилограф можна сказати наступне: це є звичайний двоканальний осцилограф який може працювати в режимі двох каналів чи в режимі *x-y*, можливе перемикання з двоканального режиму в одно канальний, можна наприклад також закоротити один з каналів на корпус, дивитись змінну складову сигналу, чи постійну разом зі змінною.

Рисунок 13.3 - Діалогове вікно генератора сигналів

Віртуальний термінал (virtual console) уявляє собою звичайний RS232 термінал, коли запускаємо емулятор відкривається діалогове вікно у якому ми вводими данні для відсилання (є можливість використання луни, можливе відображення як в звичайному режимі так і шістнадцятирічному).

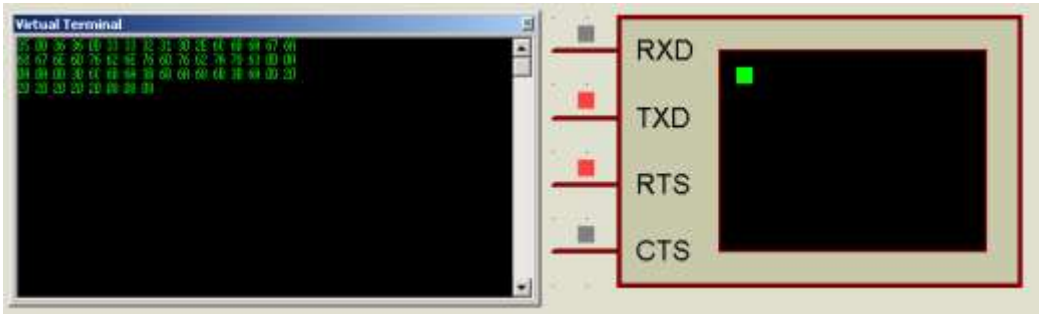


Рисунок 13.4- Віртуальний термінал

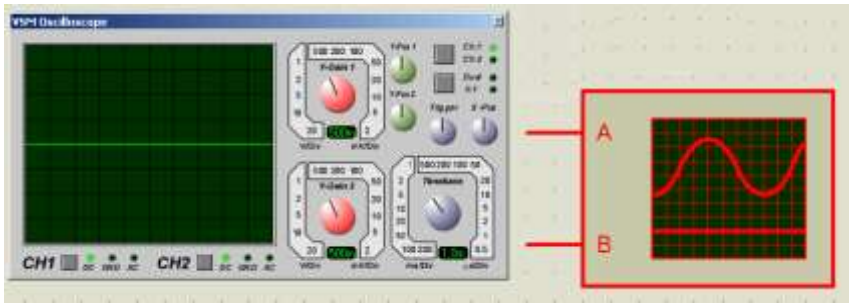


Рисунок 13.5 - Осцилограф

Редагування окремої бібліотеки проводиться наступним чином: подвійний натиск лівою клавішею миші на напису *Devices* і потрапляємо у вікно в якому знаходяться всі доступні елементи бібліотеки програми. Вони розбиті на групи за функціональним призначенням. Кожний елемент має два зовнішніх вигляди, це пов'язано зі специфікою роботи програми яка полягає у тому, що програма має одну єдину бібліотеку елементів які використовуються і для створення структурних схем, і для створення плати друкованого монтажу (один вигляд функціональний, а другий вигляд – зовнішній). Вибираємо необхідні елементи та аналогічним рухом миші додаємо їх до окремої бібліотеки.

Як бачите не всі елементи мають два вигляди. Це обумовлюється тим, що деякі елементи є віртуальними і призначені лише для використання в ISIS, а деякі просто не мають зовнішнього вигляду.

Далі використовуючи інструмент *Bus.* Проводимо з'єднання (які потім будуть використані при розведенні друкованої плати) відповідних контактів (рис. 13.6). Як видно не всі виводи доступні при роботі в ISIS, оскільки такі елементи як живлення та землі з'єднуються автоматично (що значно полегшує роботи і дозволяє сконцентруватись на проектуванні, а не на малюванні). Щоб повісити якісь виводи на землю необхідно зробити наступне: створити позначку в властивостях якої написати GND, аналогічно для живлення. Взагалі можна створювати безліч автоматично з'єднуваних з'єднань, аналогічно до живлення тільки замість напруги живлення чи землі пишемо, що хочемо наприклад назву інформація яка передається через це з'єднання.

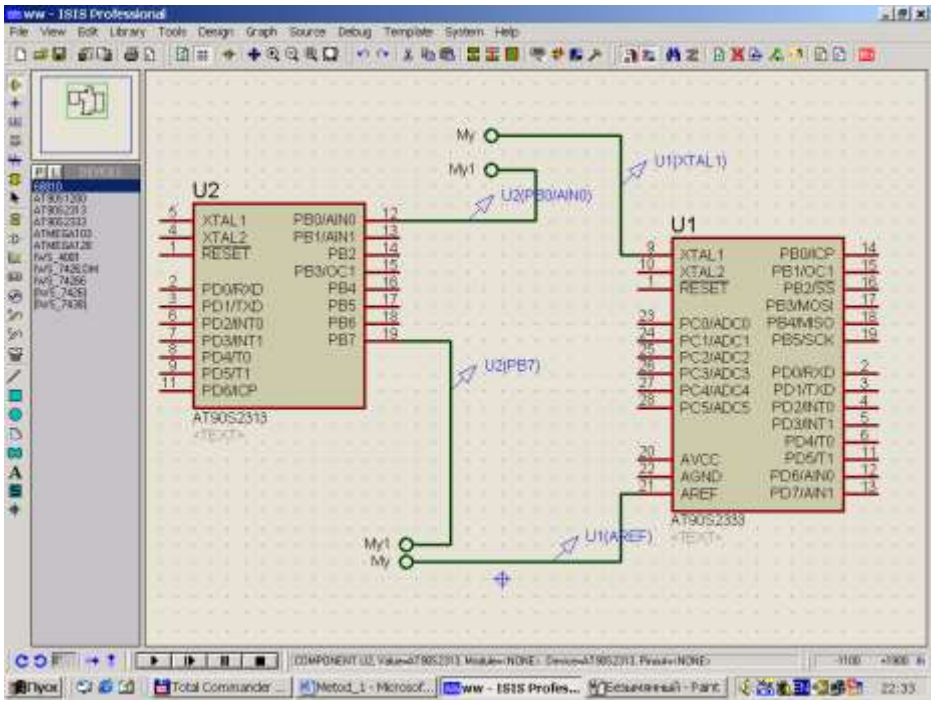


Рисунок 13.6- Робота із з'єднаннями

Також є можливість використання віртуальних пробників, які можуть показувати напругу та струм. Забігаючи наперед скажу, що можливо використання не лише пробників, але й осцилографів та спектрометрів.

За допомогою цього елемента меню, програма генерує перелік елементів з позначенням кількості та параметрів елемента. Також присутня кнопка швидкого переходу до розробки друкованої плати (напис *AREAS* на червоному фоні трохи лівіше).

Рисунок 13.7- Створення специфікації

Емулятор включається натиском на чорний трикутник внизу екрана. Після цього можна лише спостерігати, чи змінювати деякі параметри. Ось вони: змінювати опір змінного резистора (є в бібліотеці), перемикати відповідні перемикачі, натискати клавіші на клавіатурі (також є в бібліотеці). За допомогою  цих двох клавіш можна призупиняти емулятор та робити налагодження по кроках.

Коли схема намальована та перевірена на емуляторі саме час створити список з'єднань та перейти до розведення доріжок друкованої плати. Програма дозволяє зробити це двома шляхами. Перший найзручніший полягає на автоматичному створенні файлу списку та переключенні до програми AREAS. Другий полягає у самостійному створені файлу списку та вже з програми AREAS завантажується цей файл

та проводяться інші операції по розведенню. Другий спосіб можливо використовувати коли немає необхідності одразу розводити доріжки чи коли є необхідність неодноразово повторювати дану операцію.

Отож, для першого способу необхідно лише натиснути на клавішу  , потім вибрати типи корпусів для тих елементів які цього потребують.

3. Методика створення та синтезування схеми в ISIS

З метою навчання – докладно, по крокове створимо схему в ISIS 6 PROFESSIONAL (схему кодового замка), яку в подальшому дослідимо за допомогою приладів.

3.1 Запустіть програму ISIS. За замовчуванням програма встановлюється так, що запустити її можна, вибравши команди **ПУСК - ПРОГРАММЫ - Proteus 6 Professional - ISIS 6 Professional**.

3.2 Після запуску програми, відкривається вікно з новим проектом, в якому можна розпочинати створення необхідної схеми (рис.13.8).

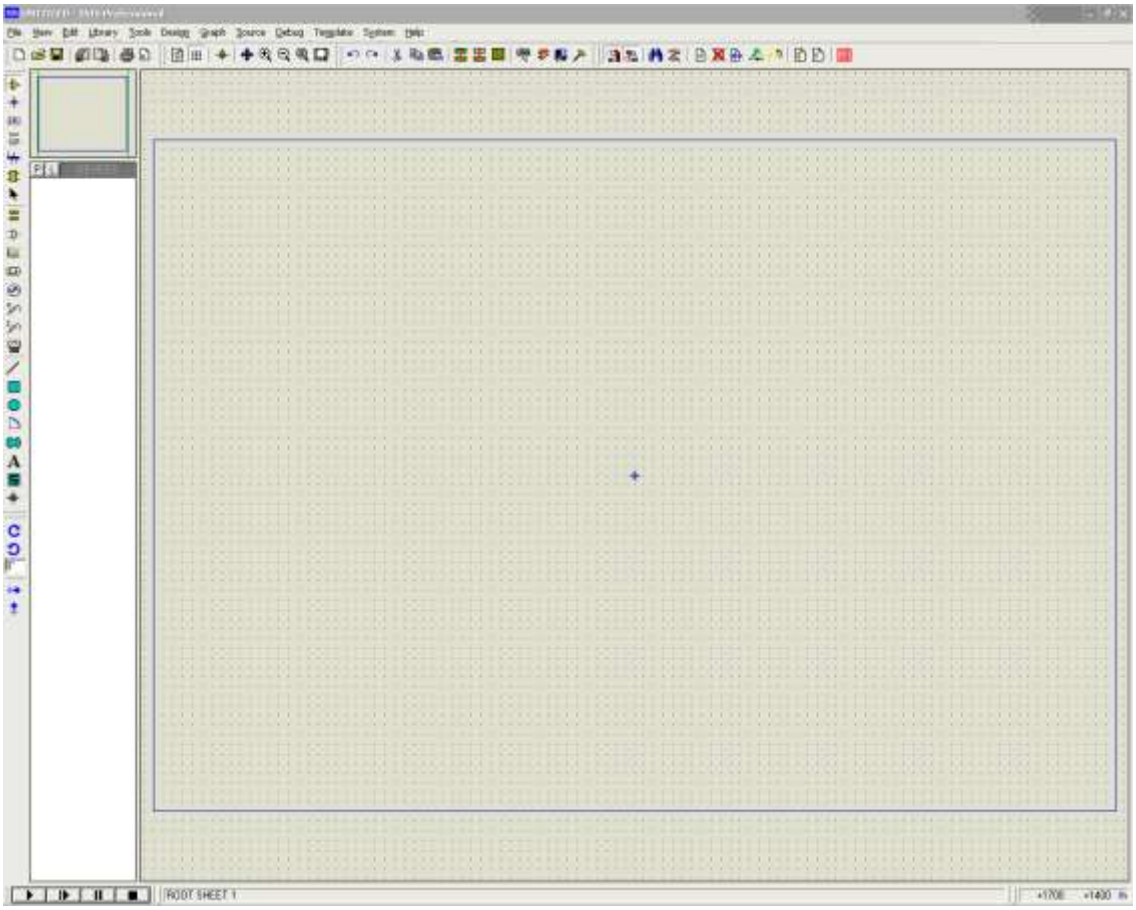


Рисунок 13.8 - Робоче вікно нового проекту

3.3 Для того, щоб перейти безпосередньо до складання нашої схеми, нам необхідно додати до нашого проекту компоненти, які будуть використовуватись у нашій схемі. Для цього ми робимо подвійний клік лівою кнопкою миші на слові DEVICES або натискаємо кнопку „P”(Pick Devices), як показано на рисунку 13.9.

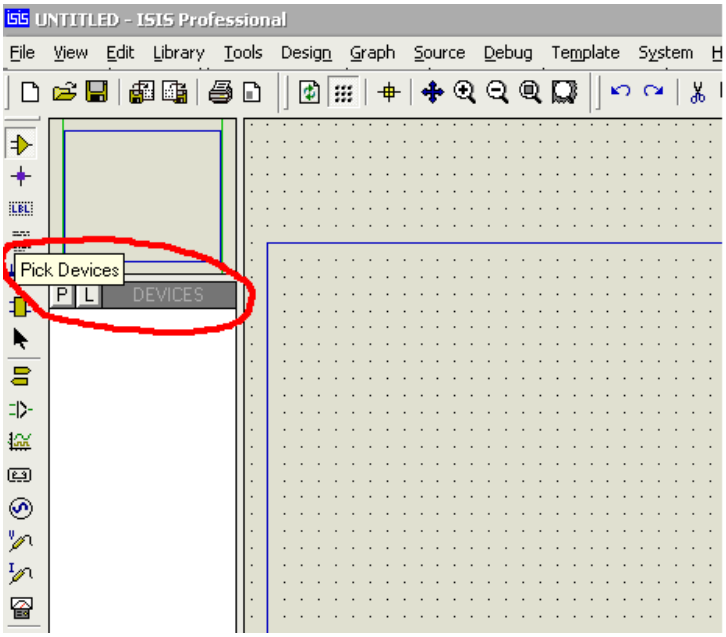


Рисунок 13.9 - Кнопка Pick Devices

3.4 Після цього з’являється вікно з набором різних бібліотек, в яких знаходяться необхідні для нас елементи (рис. 13.10).

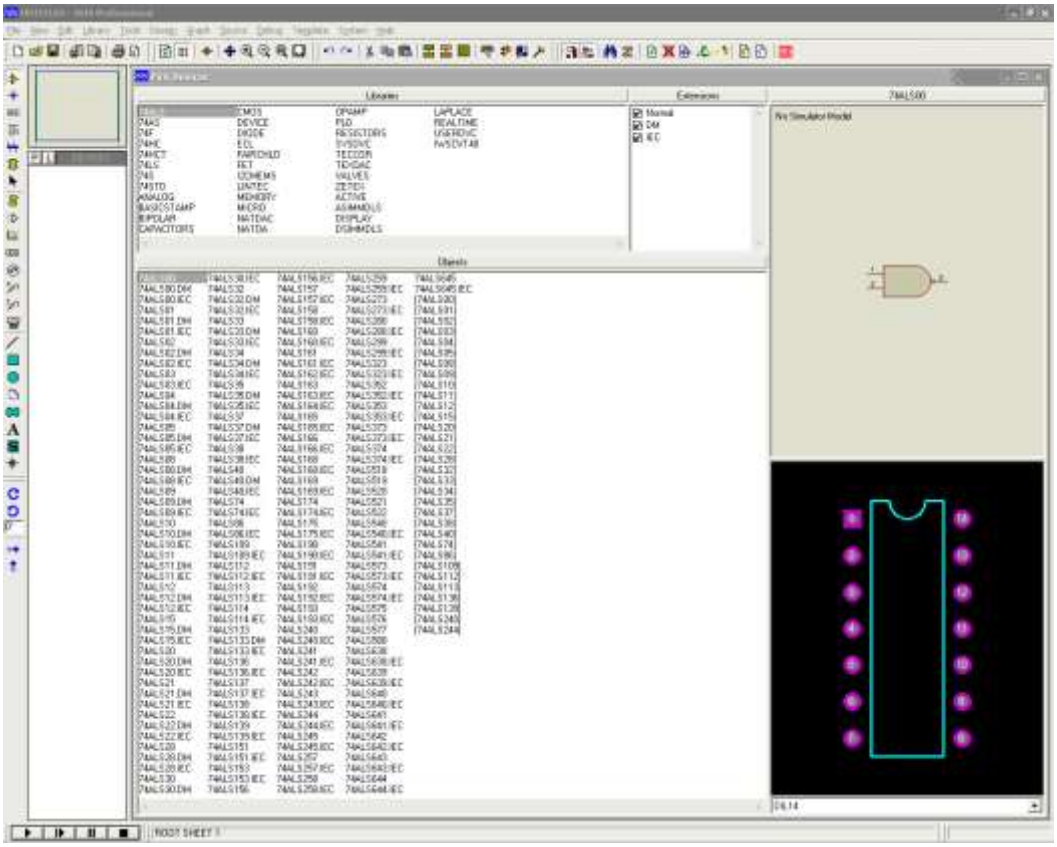


Рисунок 13.10 – Вікно бібліотек та компонент

3.5 Далі нам потрібно додати необхідні для нашої схеми елементи до нашого проекту. Для цього ми повинні натиснути лівою кнопкою миші на

бібліотеці(наприклад MICRO). Знизу під вікном бібліотек, з'явиться перелік доступних для нас компонент даної бібліотеки (рис. 13.11).

3.6 Оберемо необхідні для нас елементи. Для цього потрібно двічі клик нути лівою кнопкою миші по назві компоненту. Після цього наші компоненти з'являться у списку DEVICES (рис. 13.12). Перелік необхідних для нашої схеми компонентів: мікроконтролер *AT90S1200-12PI* у бібліотеці *MICRO*, резистор *MINRES5K1* знаходимо у секції *RESISTORS*, транзистор *BC182* в секції *BIPOLAR*, в секції *ACTIVE* знаходимо вже готову віртуальну телефонну клавіатуру, яка називається *KEYPAD-PHONE*, також звідти беремо кнопку BUTTON, з секції *DEVICE* вибираємо конденсатор *CAP*, з секції *ACTIVE* вибираємо гучномовець *SOUNDER*.

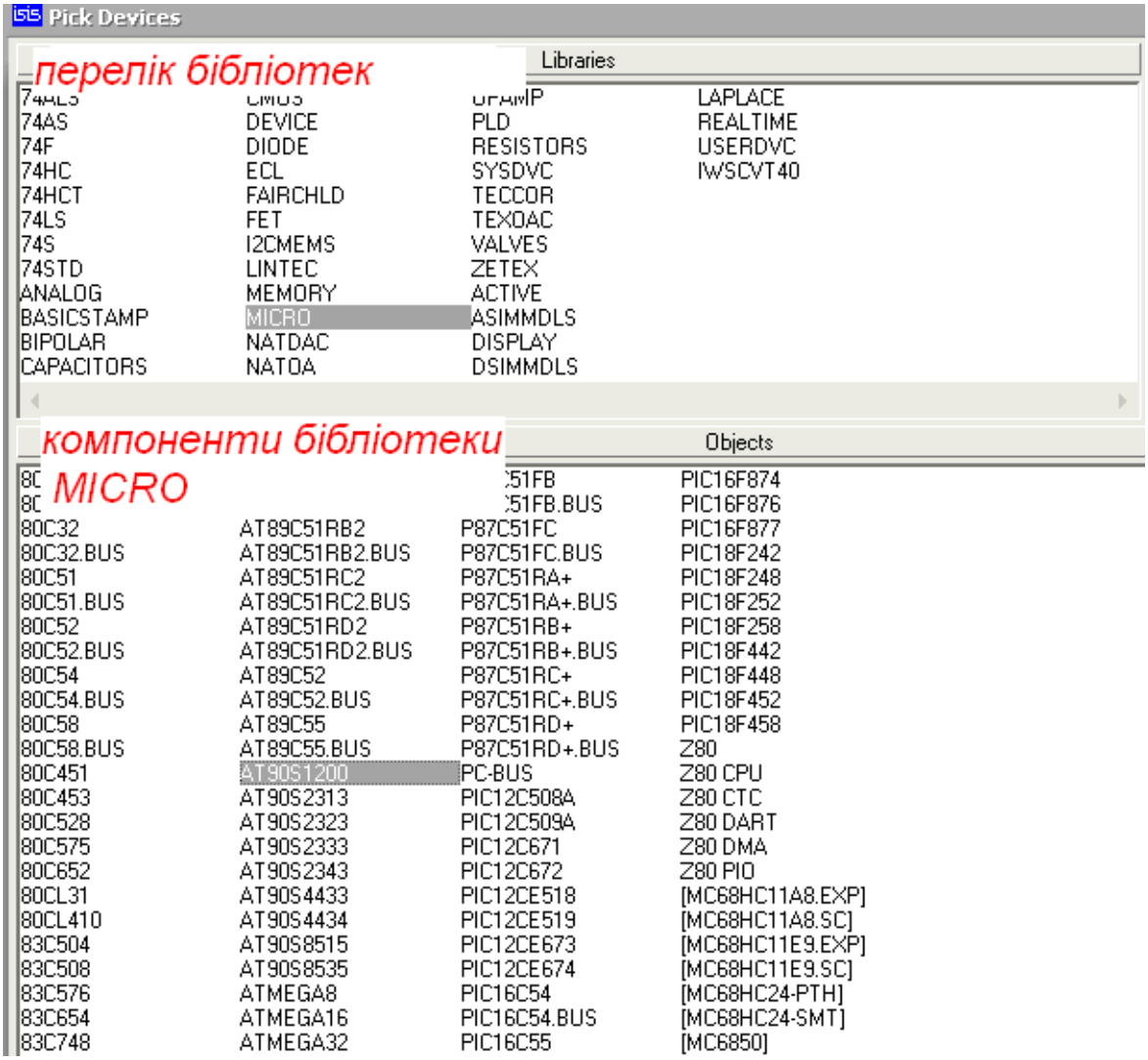


Рисунок 13.11 – Вікно компонент бібліотеки MICRO

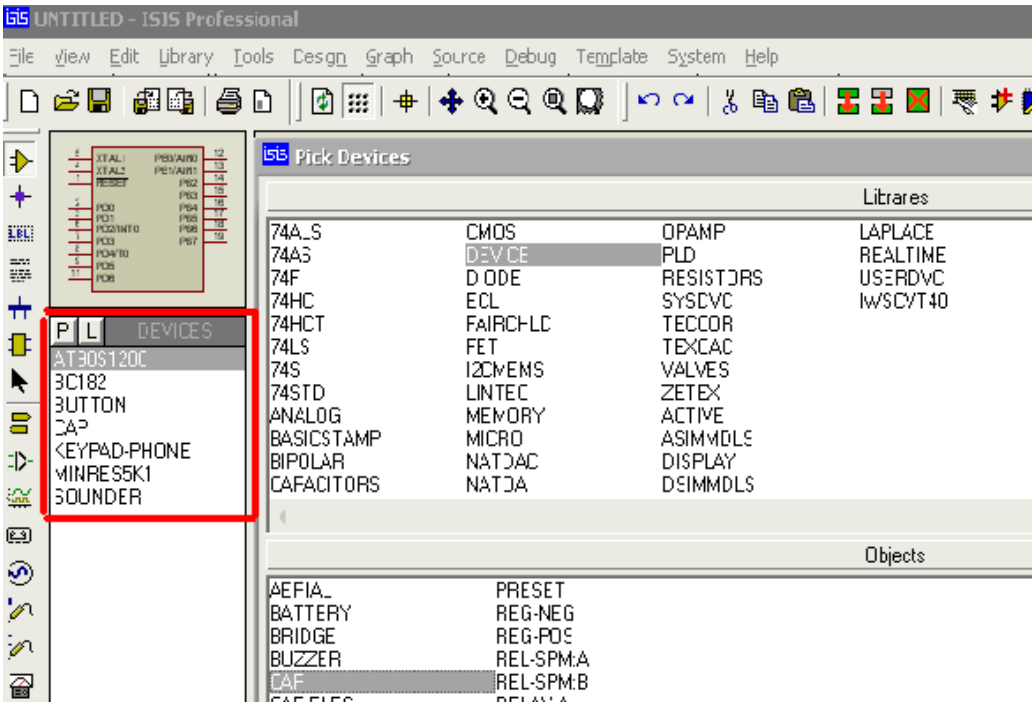


Рисунок 13.12 – Список компонент у вікні DEVICES

3.7 Перейдемо до складання нашої схеми. Для цього необхідно винести наші компоненти на робочу зону. Натискаємо на кнопку Component (рис. 13.13).

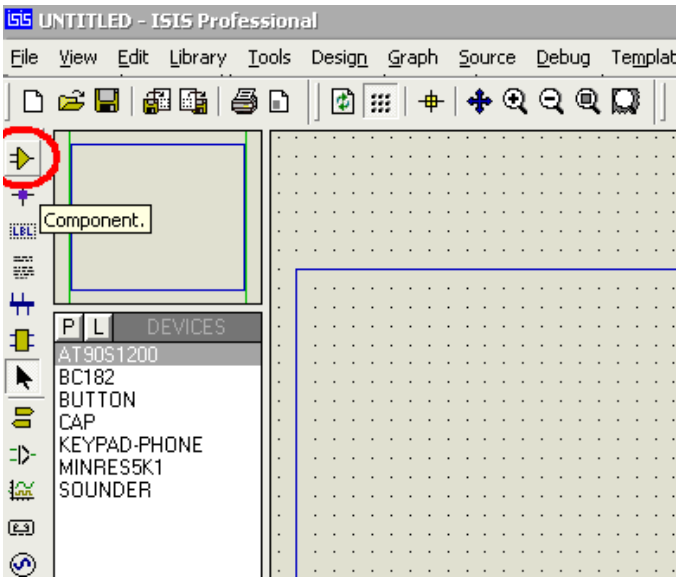


Рисунок 13.13 – Кнопка Component

Лівою кнопкою миші обираємо необхідний компонент, а потім ще раз клацаємо лівою кнопкою миші вже на робочій зоні, після чого з’являється наш компонент (рис. 13.14).

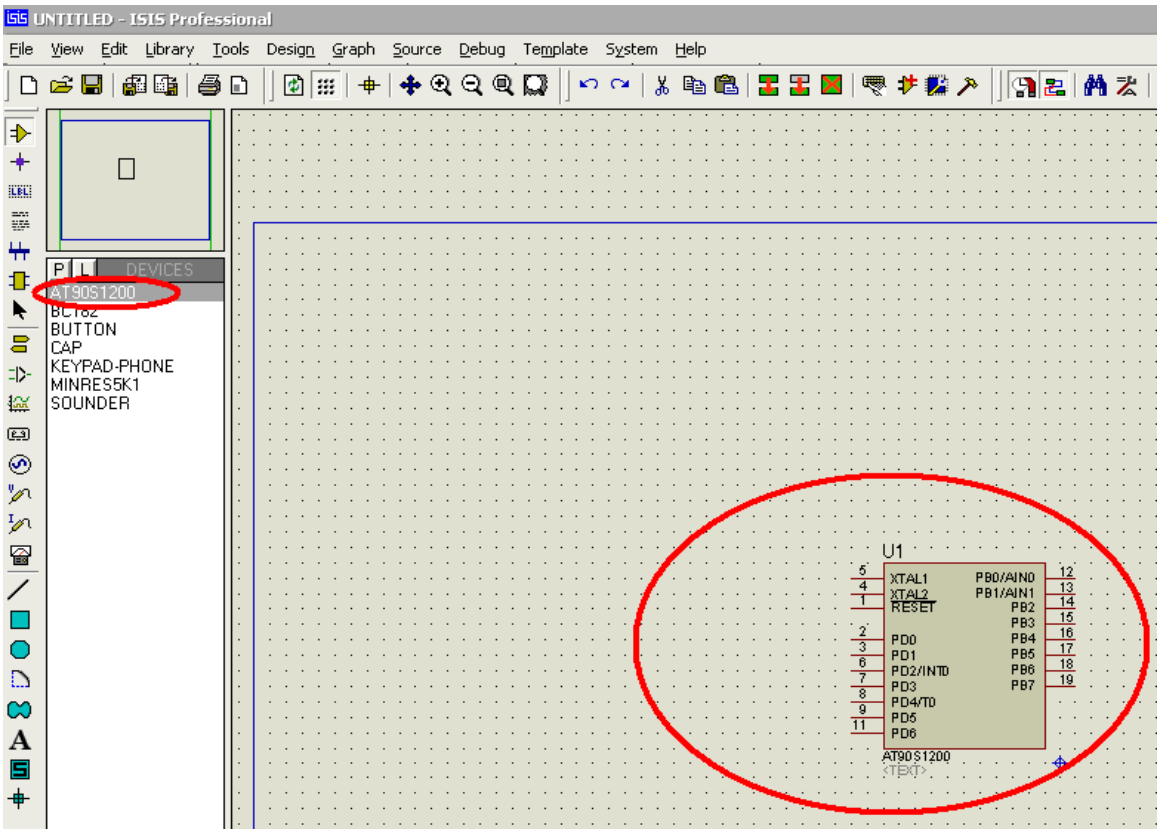
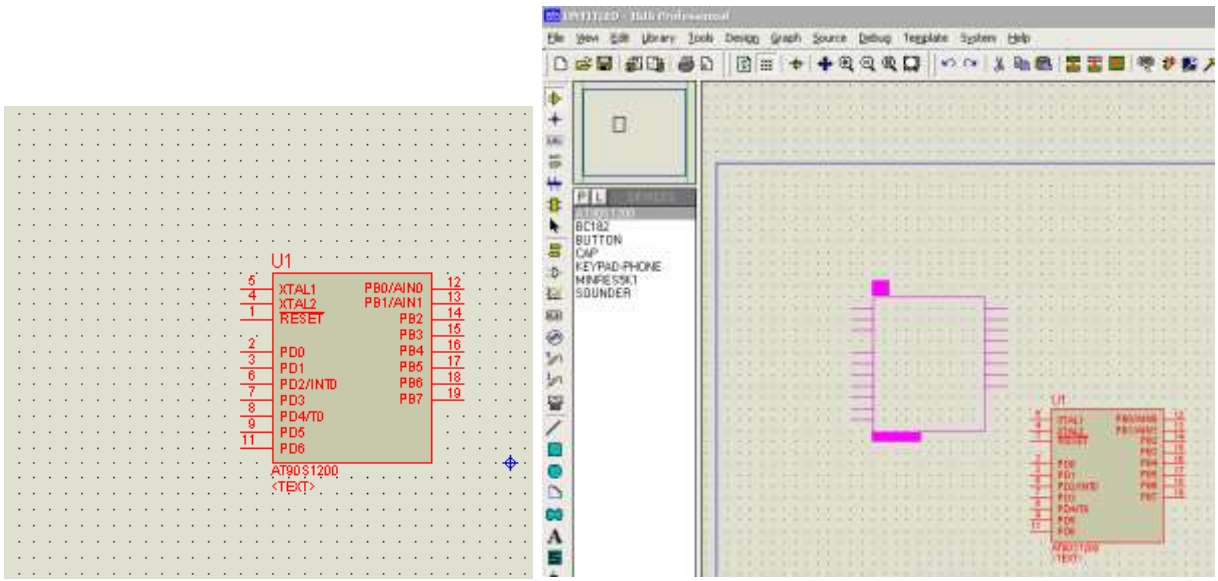


Рисунок 13.14 – Вигляд компонента на робочому столі

3.8 Для того, щоб розташувати або перемістити елемент в необхідну частину робочої зони потрібно: а) один раз клацнути правою кнопкою миші по елементу (після цього він стає червоним (рис. 13.15 а, б) навести курсор на елемент, та зажавши ліву кнопку миші перенести елемент в необхідне місце (рис. 13.15 б). Якщо обраний елемент нам не потрібен, клацаємо правою кнопкою миші на пустій частині робочої зони.



а) б)

Рисунок 13.15 – Вигляд виділеного елемента – а) та перенесення компонента - б)

Подвійний натиск правою кнопкою миші на елементі, призведе до його видалення зі схеми.

3.9 Розташуємо всі необхідні елементи нашої схеми на робочій зоні (рис. 13.16).

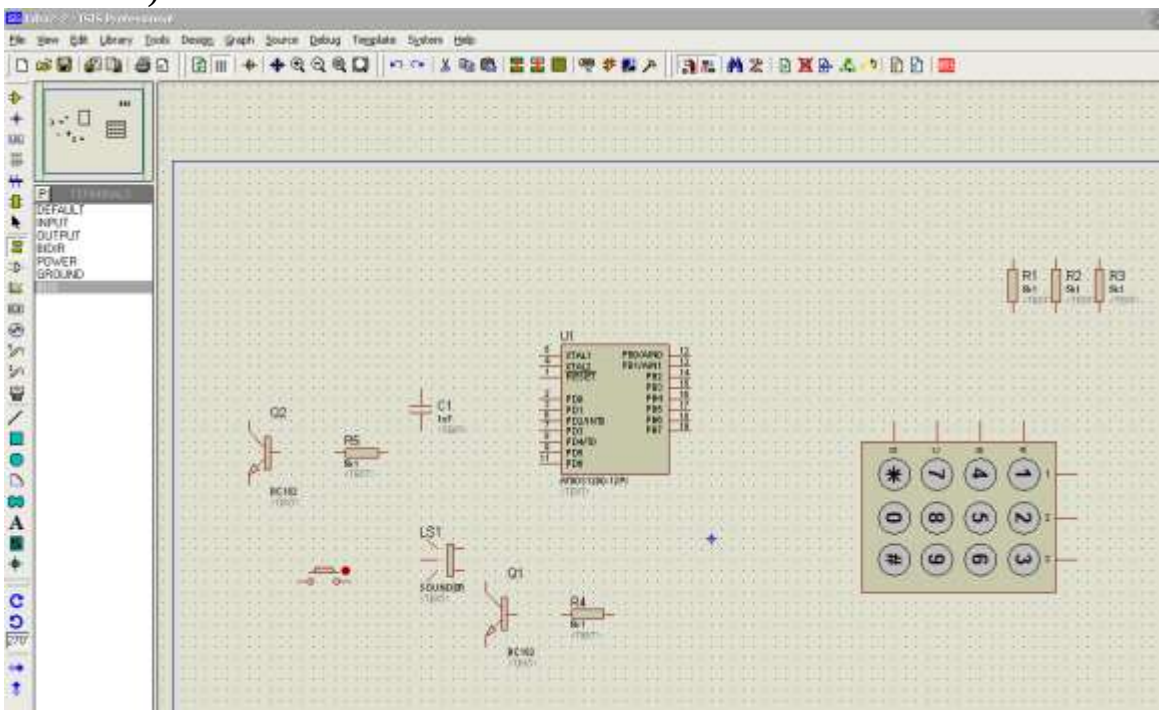


Рисунок 13.16 – Компоненти на робочій зоні

3.10 Тепер нам необхідно з'єднати наші компоненти в схему, для цього потрібно за допомогою шини поєднати всі виводи наших елементів.

Лівою кнопкою миші натискаємо на кнопку BUS (рис. 13.17).

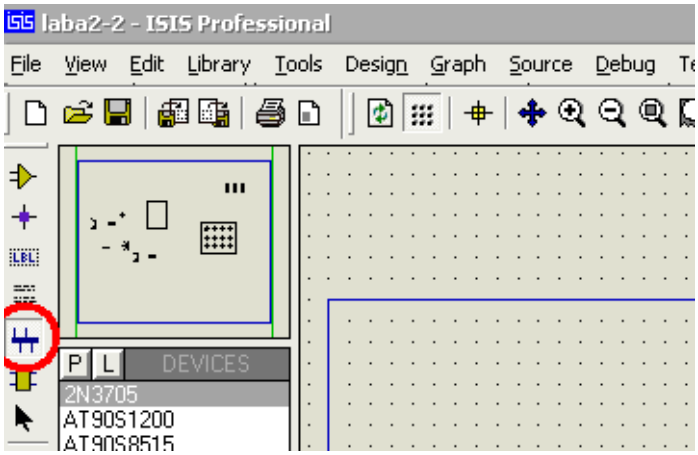


Рисунок 13.17 – Кнопка BUS

Тепер наводимо курсор миші на кінець виводу потрібного нам елементу, клацаємо один раз лівою кнопкою миші, далі підводимо курсор до іншого виводу з яким нам треба з'єднатись та ще раз клацаємо лівою кнопкою миші, після чого з'явиться з'єднання (рис. 13.18).

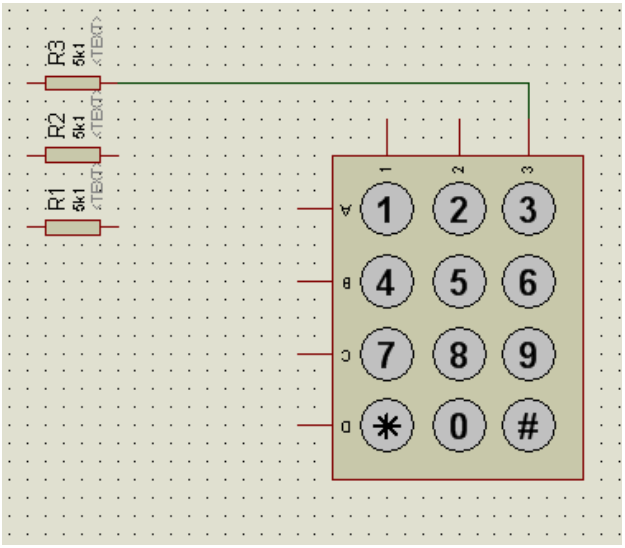


Рисунок 13.18 – Вигляд з’єднання

3.11 З’єднаємо всі елементи нашої схеми так як це показано на рис.13.19.

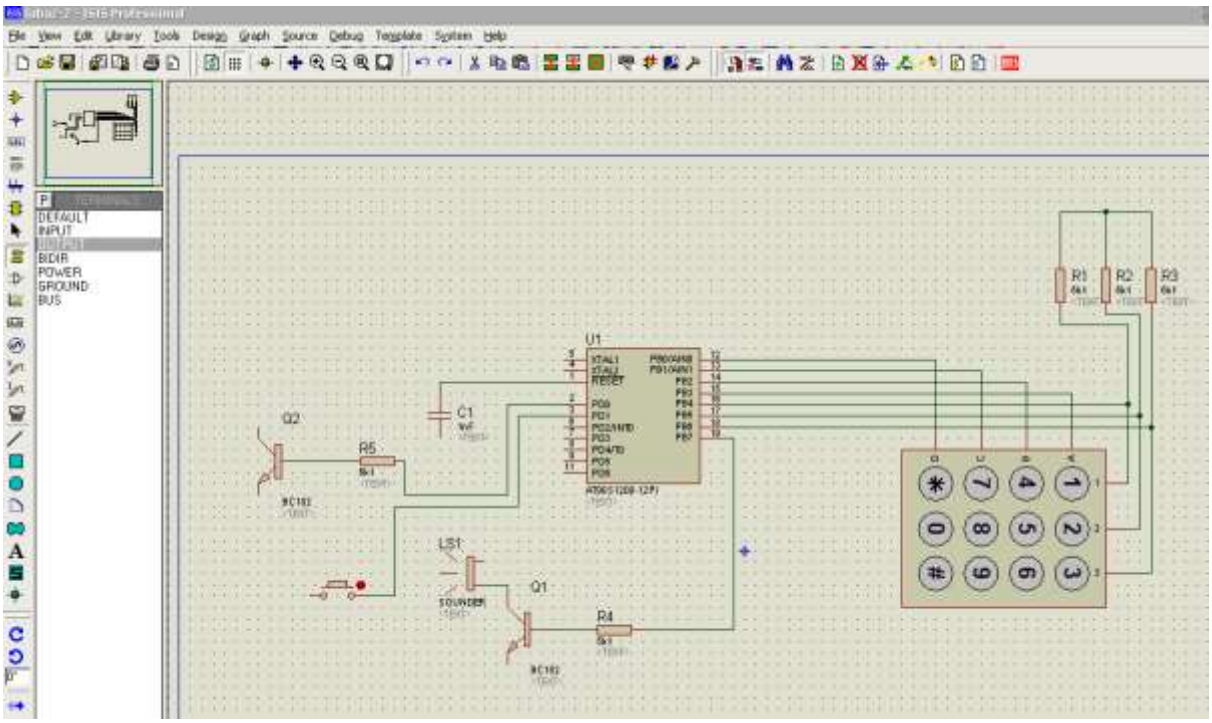


Рисунок 13.19 – Синтез схеми

3.12 Також нам необхідно поставити на деякі елементи заземлення та подати живлення. Для цього натискаємо на кнопку Inter-sheet Terminal (рис. 13.20) та виносимо необхідні нам елементи на схеми таким же чином, як і наші компоненти (пункт 3.7, 3.8).

3.13 Повний робочий варіант нашої схеми зображений на рис.13.21.

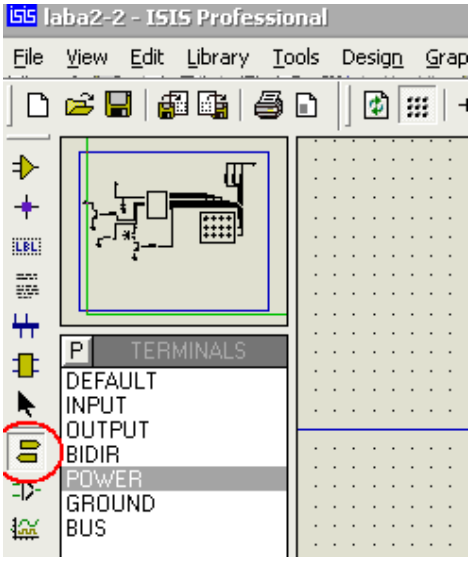


Рисунок 13.20 – Кнопка Inter-sheet Terminal

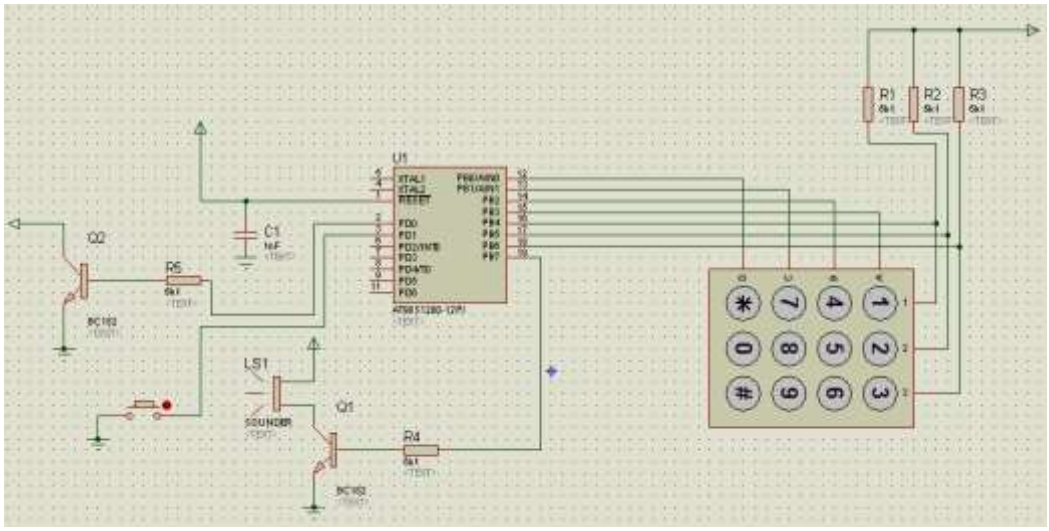


Рисунок 13.21 – Робочий варіант схеми

3.14 Перед тим як перейти до емуляції роботи схеми, необхідно мати файл прошивки(для мікроконтролера у форматі *Hex, UBROF, COFF*) для даної схеми. Для того щоб додати файл прошивки нам необхідно: а) виділити мікроконтролер правою клавішею миші; б) зайти до властивостей елемента за допомогою натискання лівої клавіші миші (рис. 13.22); в) навпроти поля *Program file* натискаємо на кнопку відкрити, відкриваємо потрібний файл прошивки, натискаємо *OK*.

3.15 Останнім кроком є емуляція роботи нашої схеми (рис. 13.23). Для цього ми натискаємо клавішу **F12**. Якщо все було зроблено правильно, то ми побачимо емуляцію роботи нашої схеми(при натисканні клавіш клавіатури буде видаватися звук, а при натисканні клавіші *(зірочка), програватись мелодія).

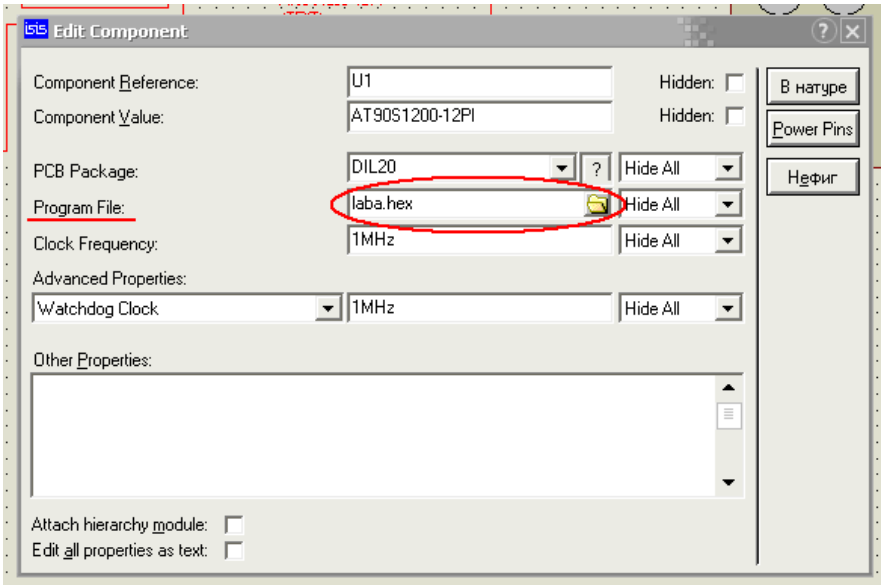


Рисунок 13.22 – Вікно властивостей компонента AT90S1200-12PI

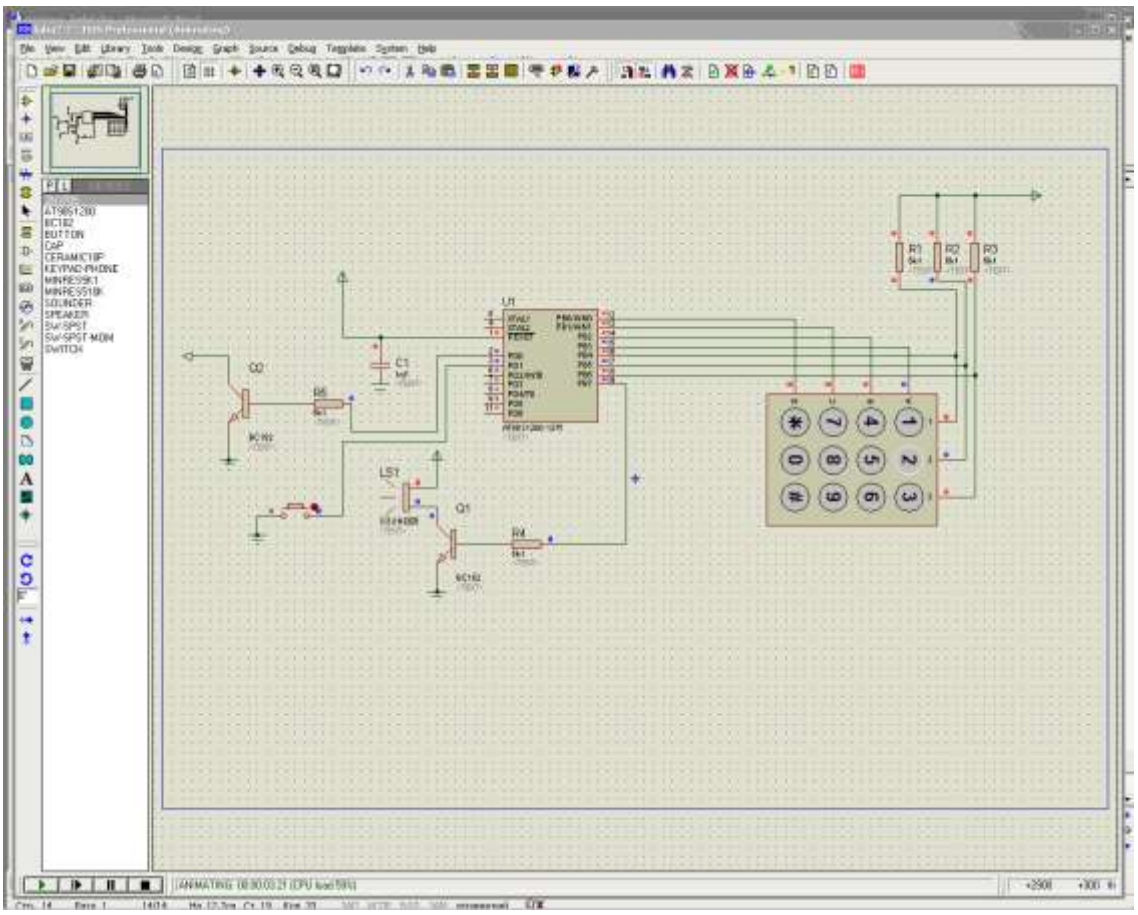


Рисунок 13.23 – Процес емуляції роботи схеми

4 Дослідження схеми за допомогою осцилографа

Розглянемо роботу та підключення віртуального осцилографа у програмі ISIS (рис. 13.24).

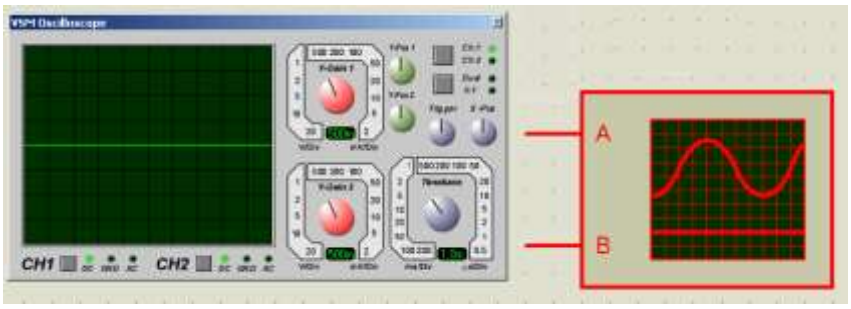


Рисунок 13.24 – Панель керування осцилографа

Для цього нам необхідно:

а) додати віртуальний осцилограф до нашої схеми, тобто винести його на робочу зону нашого проекту. Натискаємо на кнопку Virtual Instruments та обираємо за допомогою кліку лівої кнопки миші осцилограф, після чого робимо ще один клік лівою кнопкою миші по робочій зоні для додавання осцилографу безпосередньо у наш проект (рис.13.25);

б) далі нам необхідно під'єднати осцилограф в будь-якій точці нашої схеми (рис. 13.26);

в) для подальшої роботи нам необхідно запустити емулятор (кнопка F11). Після цього в панелі меню обираємо вкладку Debug – VSM Oscilloscope (рис. 14.27).

Як результат, ми побачимо наш віртуальний осцилограф готовий до роботи (рис. 13.28);

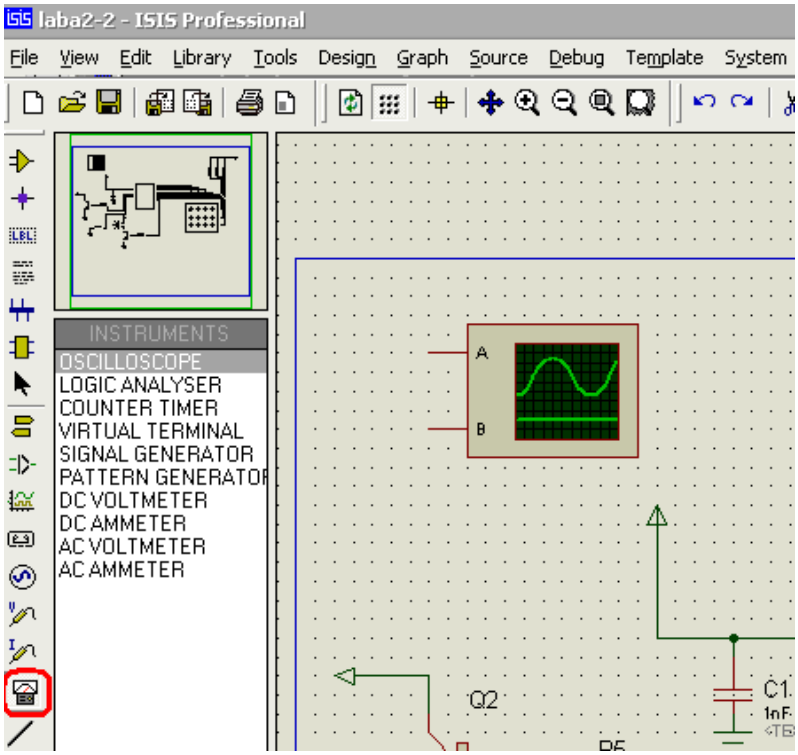


Рисунок 13.25 – Осцилограф на робочій зоні

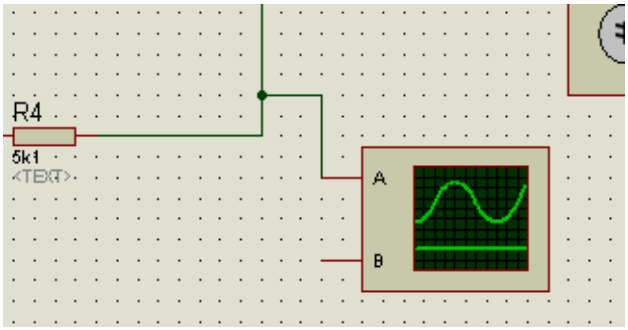


Рисунок 13.26 – Під’єднання осцилографа до схеми

г) розглянемо детальніше осцилограф. Так як осцилограф двоканальний, відповідно є присутньою можливість виводу та настроювання обох каналів.

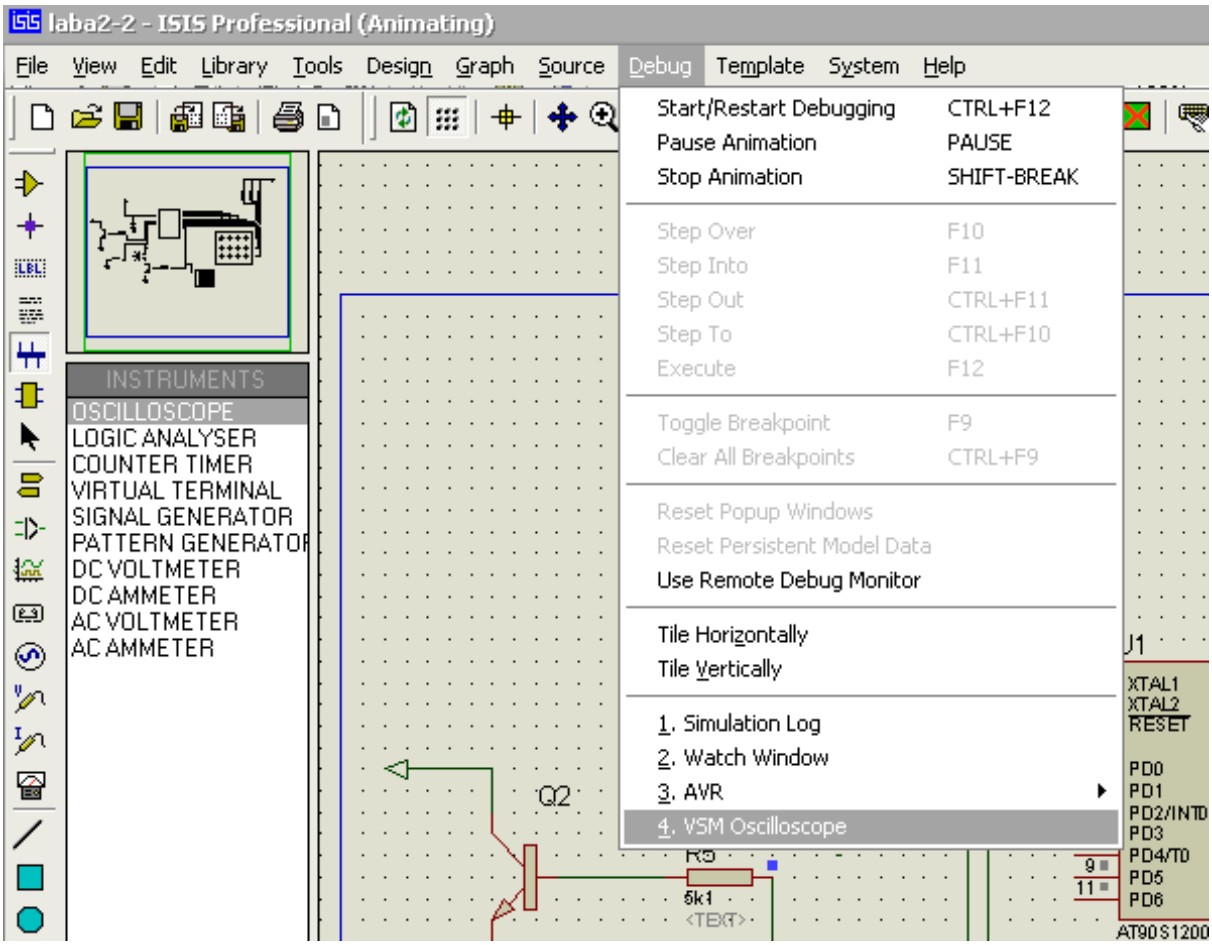


Рисунок 13.27 – Вкладка меню Debug

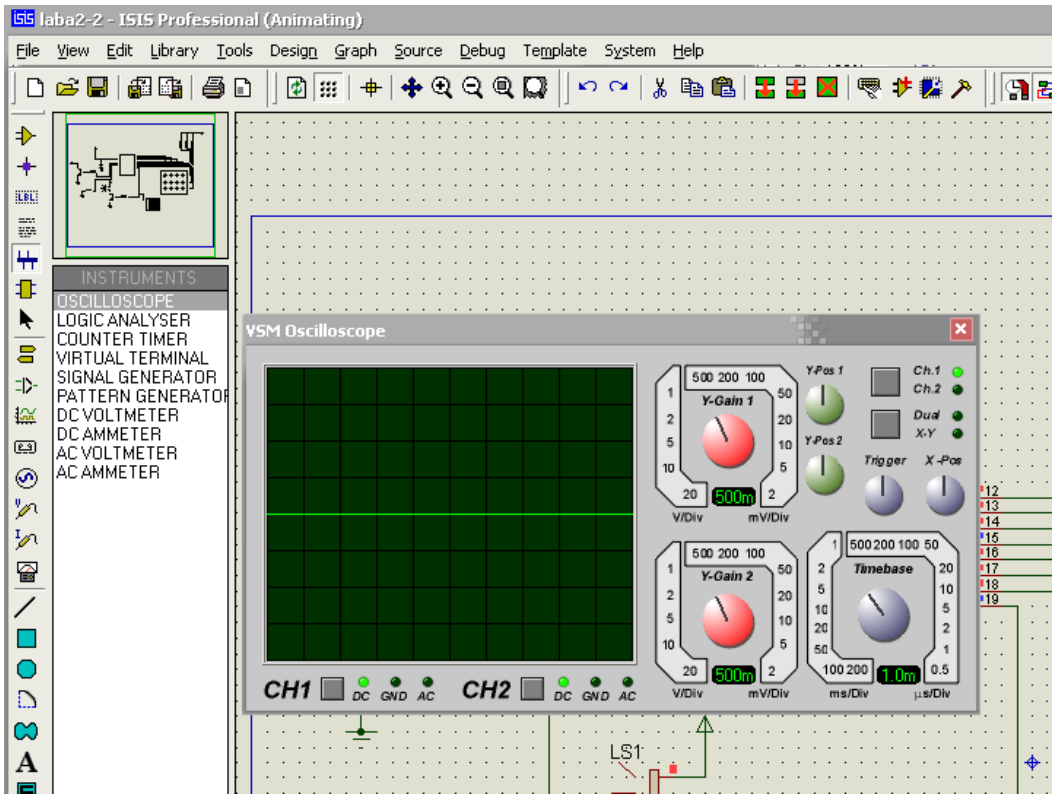
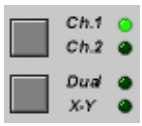
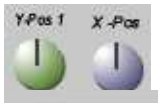


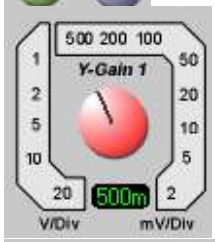
Рисунок 13.28 – Панель керування осцилографа в роботі



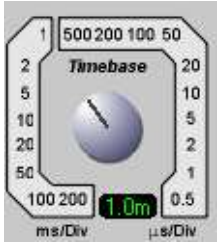
Дані кнопки відповідають за режим виводу каналів на екран осцилографа. Ch.1 та Ch.2 відповідно переключають вивід зображення першого каналу на другий та навпаки. Duad і X-У дають змогу переключати режим роботи осцилографа(двоканальний або X-У).



Дані ручки регулюють положення сигналу на моніторі відповідно по осям X та У.



Дана ручка регулює значення сигналу в діапазоні амплітуд від 500mV до 20V.



Параметр імпульсів від 500µs до 200ms.

д) для того щоб перевірити наш осцилограф в дії, виставимо необхідні параметри осцилографа та натиснемо на кнопці клавіатури, так як осцилограф підключено до динаміку то при кожному натисканні

клавіатури повинен програватись звук через динамік і відповідно ми повинні побачити сигнал на екрані осцилографу (рис. 13.29).

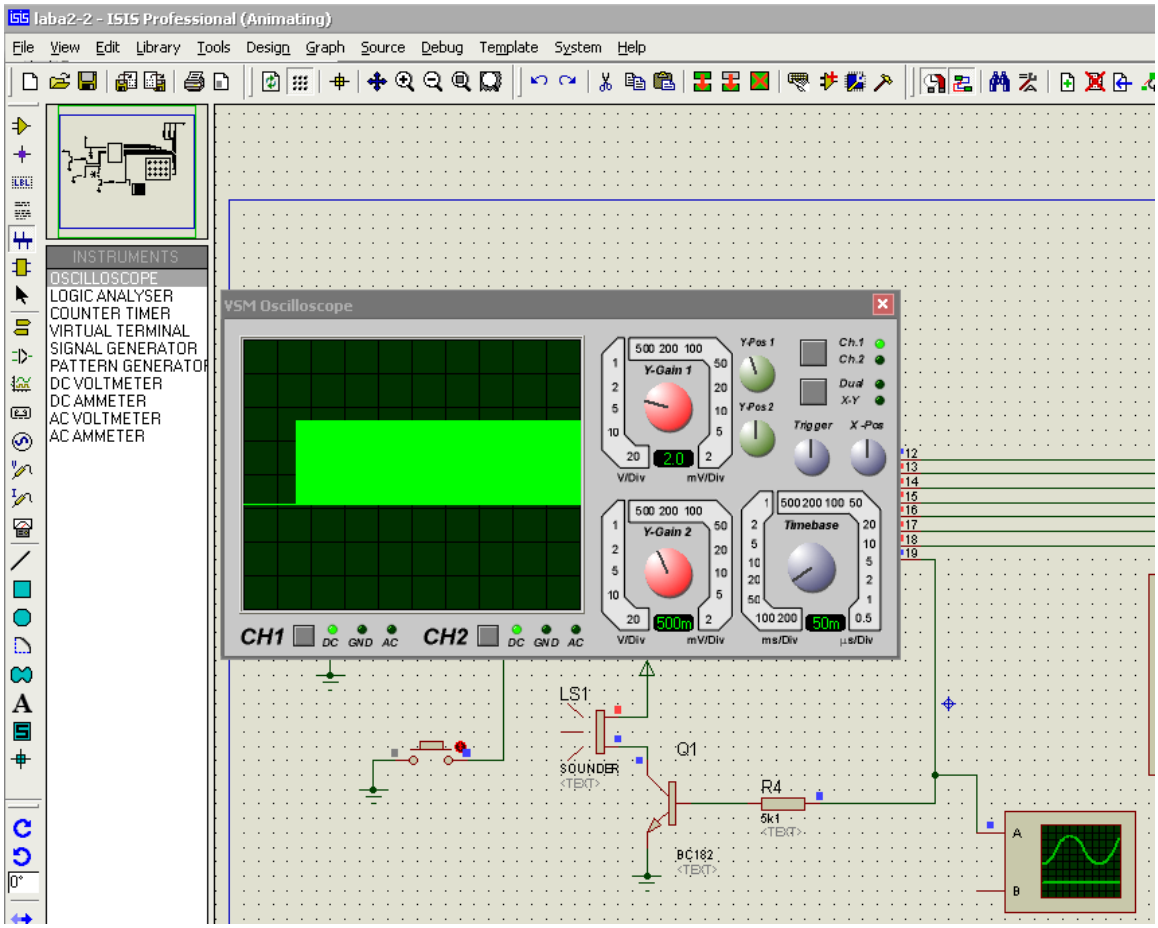


Рисунок 13.29 – Сигнал на екрані осцилографа

5. Зміна номіналів елементів та їх властивостей

Розглянемо основні пункти властивостей компонентів. Для того щоб увійти до властивостей любого компоненту нам необхідно виділити його кліком правою кнопки миші, а потім натиснути по натиснути на ньому лівою кнопкою миші.

Для прикладу візьмемо компонент у вигляді мікроконтролера AT90S1200-12.

Після того, як ми ввійшли до його властивостей ми бачимо діалогове вікно з рядом параметрів даного компоненту (рис. 13.30).

Розглянемо детальніше усі властивості.

Component Reference(U1) – текстова мітка(назва) компоненту, що буде відображатися на нашій схемі у якості підпису під елементом, для легшої орієнтації серед компонентів схеми.

Component Value – цей параметр виставляє значення компоненту, наприклад для резистору ми хочемо задати якесь значення, для цього в даній строчці ми просто прописуємо необхідне для нас значення, наприклад 100R (рис. 13.31) Для інших компонентів(конденсатор, напруга живлення) значення прописується в цій же строчці.

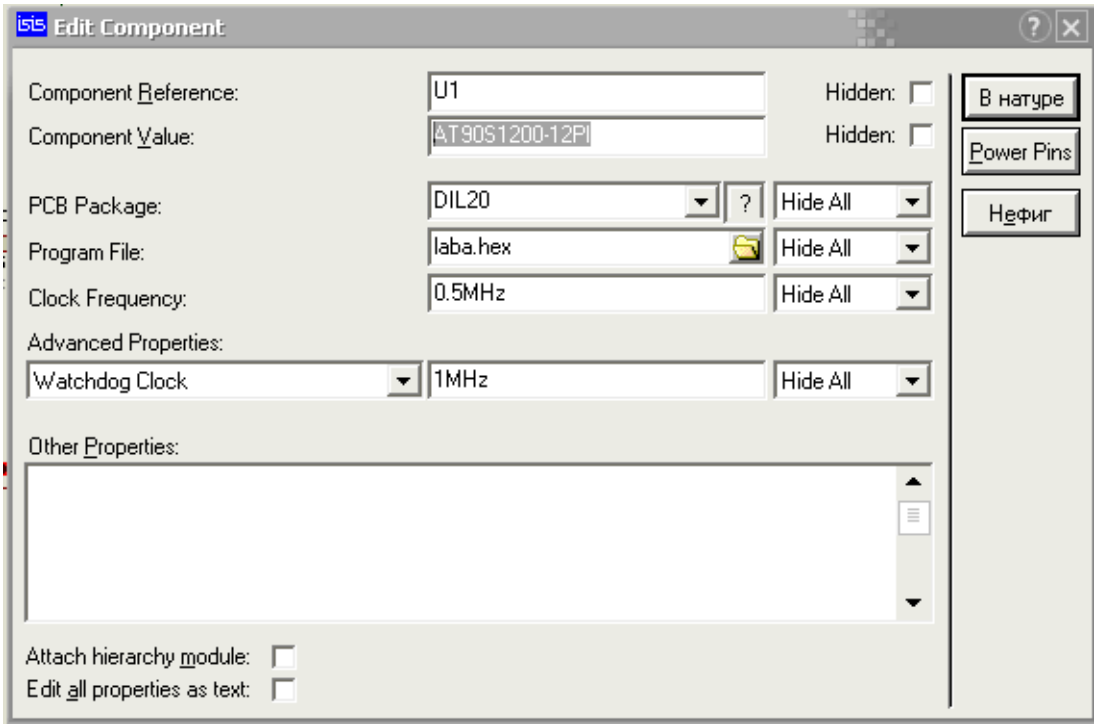


Рисунок 13.30 – Вікно параметрів компонента

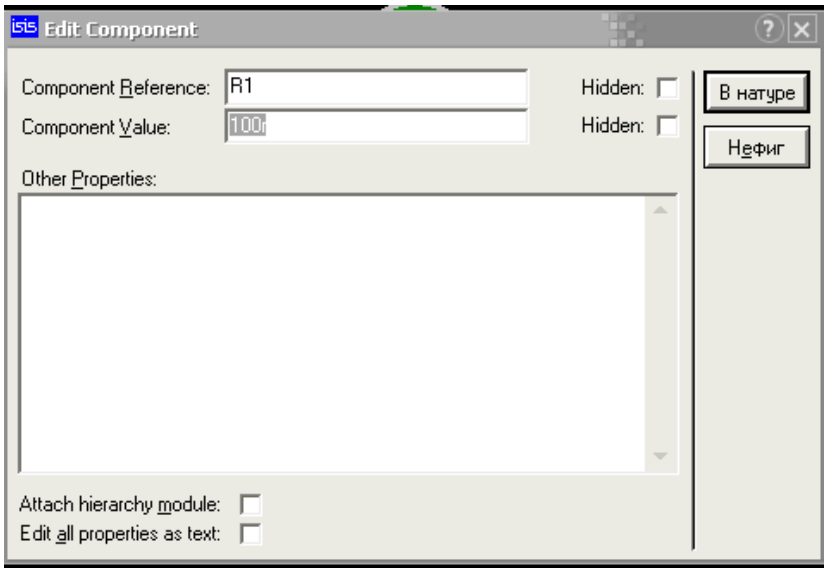


Рисунок 13.31 – Зміна значення резистора

PCP Package – в даному параметрі прописується пакет в якому знаходиться наш компонент.

Program File – тут прописується шлях до файлу прошивки нашого мікро контролера(це можна зробити за допомогою діалогу вибору файлу, натиснувши відповідну кнопку).

Clock Frequency – дана властивість відповідає за тактову частоту на якій буде працювати мікроконтролер.

За допомогою даних параметрів можна скрити/показати властивості компоненту на робочій зоні проекту

– дана кнопка слугує для виводу вікна параметрів в якому можна змінити позначення(назву) виводів живлення(Pin VCC) та заземлення (Pin GND) даного мікроконтролера.

6 Дослідження роботи синтезованої схеми

Розглянемо роботу логічного аналізатора (рис. 13.32).

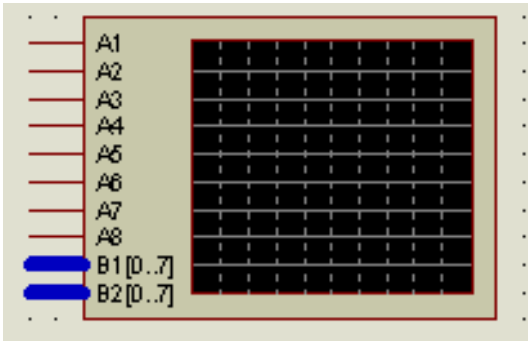


Рисунок 13.32 – Логічний аналізатор

Логічний аналізатор (logic analyzer) працює, безупинно роблячи запис вхідних цифрових даних у великий буфер збору даних. Цей процес здійснення вибірки, має коректуючи здатність, що дозволяє визначати самий короткий імпульс, що може бути зареєстрований. Секція виклику контролює вхідні дані і змушує процес збору даних зупиняти якийсь час після того, як умова(стан) виклику виникла. Результатом є те, що зміст збору даних буферизує, і після спускового механізму, може бути відображено час. Так як буфер збору даних дуже великий (10000 вибірок, у цьому випадку), забезпечуються засоби зміни масштабу зображення і панорамування. Маються також рухливі маркери виміру, що дозволяють задавати точні розміри(виміру) вибору часу розмірів імпульсу тощо.

Логічний аналізатор має такі властивості:

·8 x 1 розрядні виводи і 2 x 8 розрядні виводи шини.

·10000 x 24 біт фіксований буфер.

·Здатність збору даних від 200us у вибірку до 0.5ns у вибірку з відповідним збором даних часу від 2s до 5ms.

·Дисплей змінює масштаб зображення діапазону від 1000 вибірок у розділ(розподіл) до 1 однієї в розділ.

·Спусковий механізм позиціонує процентне значення у 0, 25, 50, 75 і 100 % буфера зборів даних.

·Передбачено два курсори, що задають точні розміри(виміру) вибору часу.

Підключимо логічний аналізатор до схеми:

а) Натискаємо кнопку Virtual Instruments, обираємо логічний аналізатор (рис. 13.33), після чого переносимо його на робочу зону нашого проекту.

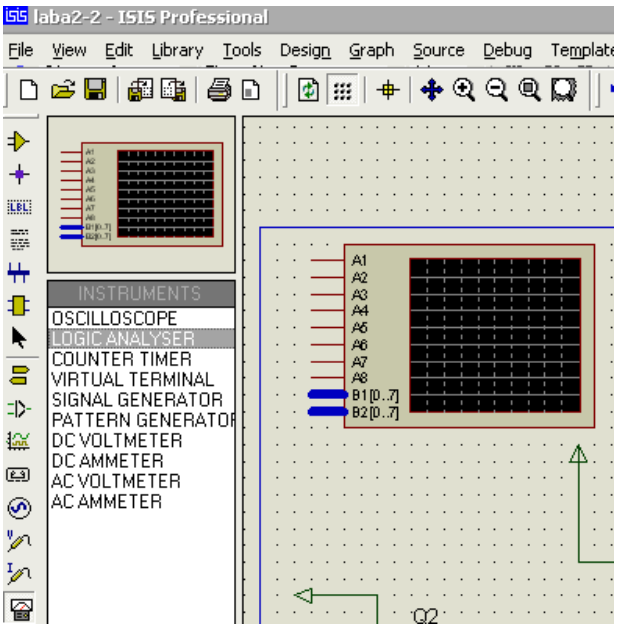


Рисунок 13.33 – Перенесення логічного аналізатора на робочу зону

б) Далі під'єднуємо необхідні нам виводи логічного аналізатора до схеми (рис. 13.34).

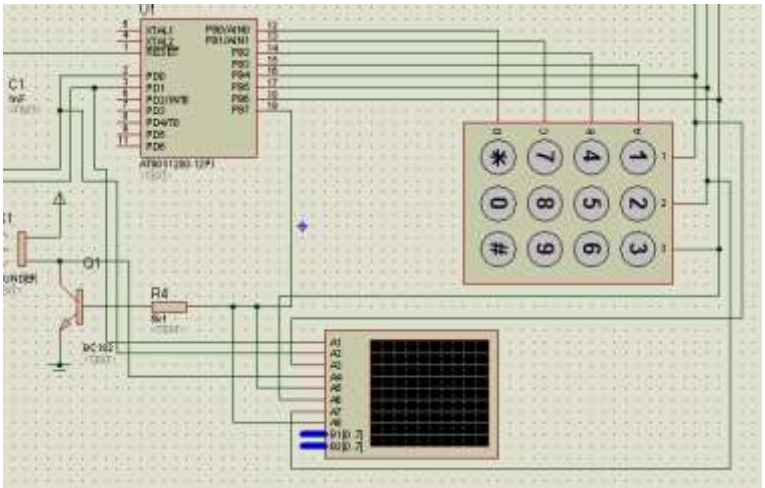


Рисунок 13.34 – Під'єднання логічного аналізатора до схеми

в) Після того, як ми підключили логічний аналізатор до схеми запускаємо проект на виконання (клавіша F11). Далі на панелі меню обираємо вкладку Debug – VSM Logic Analyser (рис. 13.35).

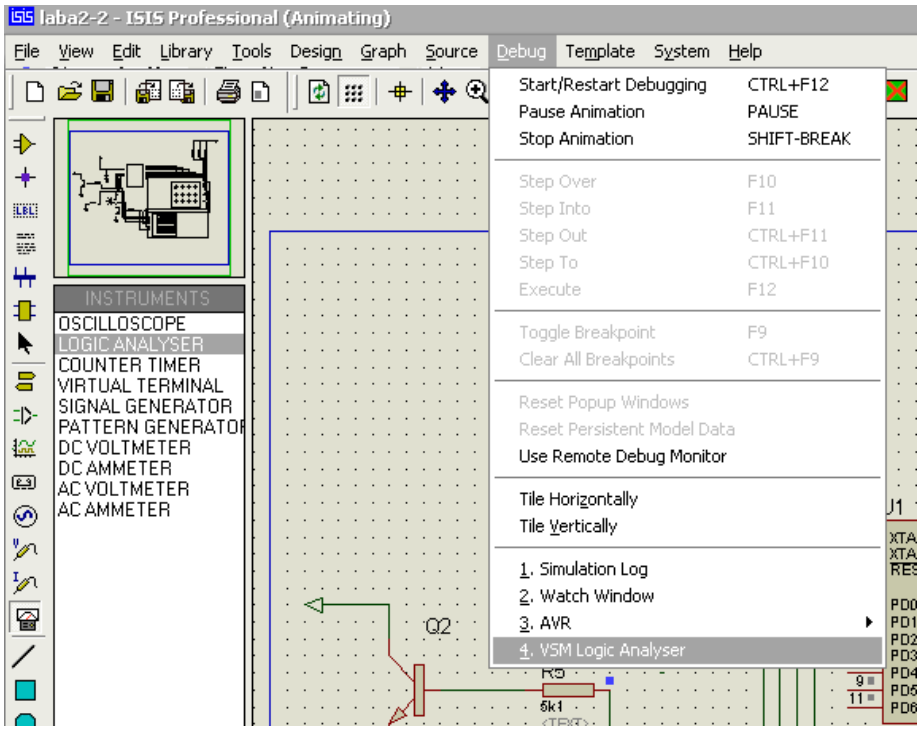


Рисунок 13.35 – Вкладка головного меню Debug – VSM Logic Analyser

г) Як результат, ми побачимо наш віртуальний логічний аналізатор готовий до роботи (рис. 13.36).

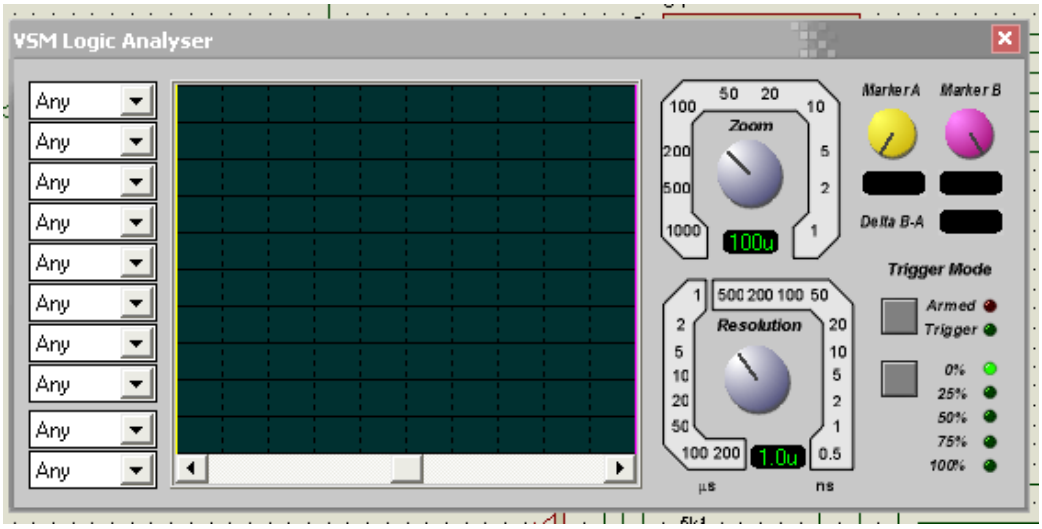


Рисунок 13.36 – Панель керування логічного аналізатора

д) Для початку робот з логічним аналізатором нам необхідно виставити необхідні параметри за допомогою ручок Zoom та Resolution і натиснути кнопку Trigger Mode. Після натискання кнопки повинен загорітися червоний індикатор з написом Armed (рис. 13.37).

Рисунок 13.37 Кнопка Trigger Mode

е) Далі натискаємо кнопки клавіатури на нашій схемі, після цього ми побачимо, що індикатор Trigger Mode перейшов в положення Trigger і загорівся зеленим кольором, а на екрані аналізатора з'явилися відповідно отримані дані, з відповідних виходів аналізатора, підключених до схеми (рис. 13.38).

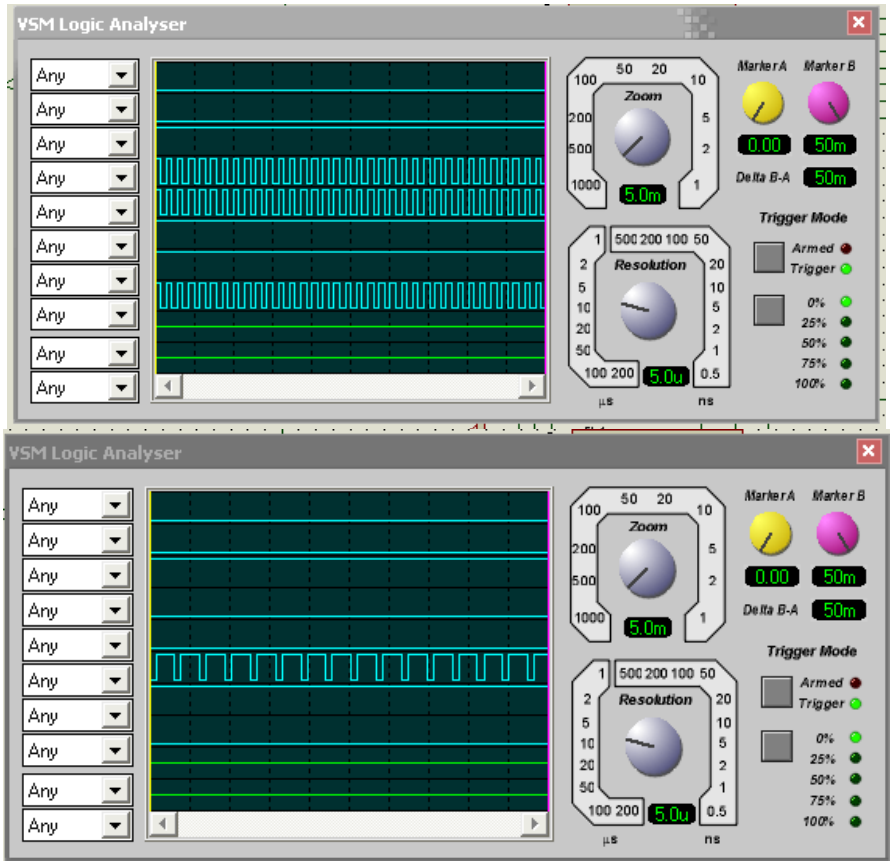


Рисунок 13.38 - Отримання даних на екран логічного аналізатора

Завдання до лабораторної роботи

1. На основі теоретичних відомостей навчитись синтезувати та аналізувати створений пристрій в запропонованому програмному продукті Proteus 6 Professional.

2. Провести огляд віртуальних інструментів програми ISIS, що входить до складу пакету Proteus 6 Professional та відповідно вміти аналізувати схему за допомогою цих інструментів.

3. Спираючись на п. 3 теоретичних відомостей, покроково створити схему в ISIS 6 PROFESSIONAL (схему кодового замка).

4. Спираючись на п. 4 теоретичних відомостей, дослідити створену схему за допомогою осцилографа.

5. Спираючись на п. 5 теоретичних відомостей, навчитись змінювати номінали елементів та їх властивості.

6. Спираючись на п. 6 теоретичних відомостей, дослідити роботу синтезованої схеми за допомогою логічного аналізатора.

Зміст звіту

Звіт з виконаної роботи повинен містити: мету роботи, тексти розроблених в дома програм, результати виконання цих програм, зображення схеми у вікні програми, осцилограми визначені в контрольних точках, а також висновки по роботі.

Контрольні запитання

1. Розкажіть про можливості пакету Proteus.
2. Перерахуйте основні етапи роботи з програмою ISIS.
3. Охарактеризуйте основні пункти властивостей компонентів.

Лабораторна робота №14

СИНТЕЗУВАННЯ ЗАДАНОЇ СХЕМИ У ПАКЕТІ PROTEUS

Мета роботи: навчитись створювати свій власний проект, синтезувати схему по власному варіанту завдання, створювати та реалізовувати підпрограму для мікроконтролера, вміти покроково досліджувати створену схему.

Теоретичні відомості

Опанувавши необхідним мінімумом знань, спробуйте створити власну модель пристрою, для прикладу запропонуємо свій варіант схеми. Своєрідний мікроконтролерний "сигналізатор". Передбачається, що частоту спалахів світлодіоду HL1 можна змінювати програмною прошивкою мікросхеми.

При запуску програма ISI створює новий проект автоматично. Якщо на початку роботи довелося завантажити інший проект, щоб створити новий, досить вибрати в меню File пункт New Design.

Збережете новий проект (поки порожній), вибравши в меню File пункт Save Design. Перед виконанням цієї операції система попросить

змінити ім'я, привласнене за замовчуванням (UNTI-TLED.DSN - безіменний), будь-яким іншим на вибір користувача. Зберегти проект можна й у папці, пропонованою системою, але краще створити для нього спеціальну, назвавши її, наприклад, blum. Надалі саме в ній буде зберігатися всі файли, зв'язані з проектом, у тому числі програма мікроконтролера і допоміжні файли, створювані системою в процесі роботи.

Роботу над моделлю почніть з підбору необхідних компонентів. Знайдіть у бібліотеці світлодіод червоного світіння LED-RED . Простіше всього це робити, набираючи умовні найменування компонентів у вікні Keywords. В міру нагромадження досвіду можна буде шукати компоненти інакше, заходячи за ними в потрібні розділи бібліотеки.

Нагадую, щоб перенести знайдений компонент у вікно селектора об'єктів, потрібно двічі "клацнути" по рядку з його назвою в бібліотеці. У результаті повинний вийти список, показаний на рис. 14.1.

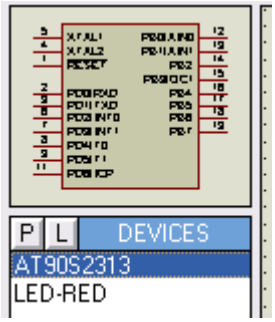


Рисунок 14.1 – Необхідний список елементів

Розміщаючи і з'єднуючи компоненти на екрані комп'ютера в основному вікні програми, відкривайте вікна властивостей кожного з них і задавайте значення Component Reference (позиційне позначення), Resistance (опір) або Capacity (ємність) у залежності від схеми. Властивість Full drive current (номінальний струм) світлодіода краще зменшити до 1...2 мА, щоб у включеному стані він виглядав на екрані яскравіше.

З властивостей мікроконтролера обов'язково вкажіть Processor Clock Frequency (тактову частоту процесора) - 10 МГц. Ім'я Hex-файла програми (Program file) можна задати відразу, якщо він готовий, або пізніше, після того як він буде створений. Про існування в ISI можливості для цього і засобах налагодження програм буде розказано далі.

Щоб "заземлити" вивід світлодіоду, буде потрібно спеціальний компонент GROUND. Візьміть його зі списку, що з'являється у вікні селектора об'єктів при натисканні на кнопку  . Крім того, додайте в зібрану модель осцилограф (див. п. 4 лабораторної роботи № 13).

У результаті повинне вийти зображення, подібне показаному на рис.14.2. "Підводити" до моделі мікроконтролера живлення не потрібно, хоча при необхідності можна зробити і це.

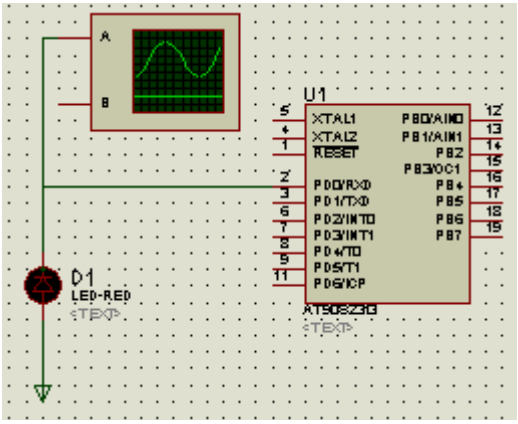


Рисунок 14.2 – Зібрана схема

Збережіть створений проект. Цю операцію рекомендується робити частіше, особливо перед початком налагодження готової моделі або перед внесенням у неї змін. Це звільнить від необхідності повторювати всю пророблену роботу у випадку збою. Так буває, якщо в "зібраній" моделі маються неправильні з'єднання компонентів або несумісні значення їхніх властивостей.

Невеликий відступ. Якщо модель пристрою з мікроконтролером призначена тільки для налагодження його програми, не намагайтеся точно відтворити схему-прототип. Скрізь, де можливо, заміняйте аналогові елементи (наприклад, транзисторні ключі) цифрову, виконуючу аналогічну функцію. Цим комп'ютер буде звільнений від рутинних розрахунків перехідних процесів в аналогових елементах, що займають левину частку процесорного часу. Моделі багатьох компонентів, у тому числі резисторів, мають "цифровий" і "аналоговий" варіанти.

Залишилося підготувати програму для мікроконтролера і завантажити її в модель. Вихідний текст програми мовою ассемблера AVRASM, по якій повинен працювати "сигналізатор", приведений рис.14.3. Наберіть його в будь-якому текстовому редакторі і збережіть в папці проекту (blum) під ім'ям blum.asm.

```
    ldi r21,0xdf
    out 0x3d,r21
    sbi 0x11,0
    cbi 0x12,0
m1: sbi 0x12,0
    call ZATRUMKA
    cbi 0x12,0
    call ZATRUMKA
    rjmp M1
;************************************
;programa zatrumku
;************************************
 ZATRUMKA: ldi r18,0x02
      m4: ldi r17,0xff
      m3: ldi r16,0xff
      m2: dec r16
          brne m2
          dec r17
          brne m3
          dec r18
          brne m4
          ret
```

Рисунок 14.3 – Вихідний текст програми

Одразу для більшої зрозумілості спробуємо описати дану програму, отже для початку наведемо загальну частину програму:

```
        .device at90s2313      ] ініціалізація стеку
        LDI r21,0xdf           ]
        OUT 0x3d,r21           ] настройка порту D на
        SBI 0x11,0             ] вивід
        CBI 0x12,0             ] запалити світлодіод
m1:     SBI 0x12,0             ] (записати D1 у port D
        CALL ZATRUMKA          ] викликати підпрограму затримки
        CBI 0x12,0             ] погасити світлодіод
        CALL ZATRUMKA          ] (записати D0 у port D
        RJMP m1                ] безкінчений цикл
```

Тепер доцільно вказати підпрограму затримки, яка визначає тривалість світіння. Також вкажемо алгоритм затримки (рис.14.4).

```
ZATRUMKA:   LDI r18,0x02
      m4:   LDI r17,0xff
      m3:   LDI r16,0xff
      m2:   DEC r16
            BRNE m2
            DEC r17
            BRNE m3
            DEC r18
            BRNE m4
            RET
```

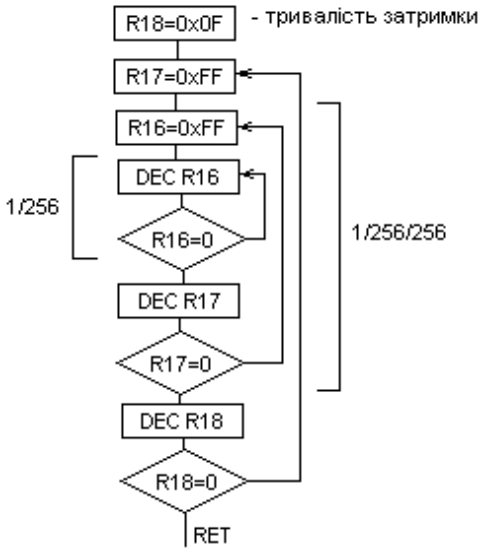


Рисунок 14.4 – Алгоритм затримки

Розібравшись з програмою продовжимо. Відкрийте пункт Add/Remove Source Files... у меню Source. Буде відкрите вікно, показане на рис. 14.5, у якому необхідно натиснути на екранну кнопку New, потім увести Source Code Filename - ім'я файлу вихідного тексту програми. Знайти його допоможе кнопка Change. У відповідь на її натискання на екран буде виведене вікно, що дозволяє вибрати потрібний файл із наявних на дисках комп'ютера.

Далі виберіть Code Generation Tool - засіб трансляції програми. Їхній список "випадає" при натисканні на кнопку зі стрілкою у відповідного вікна. У даному випадку необхідний ассемблер AVRASM.

Ввівши у вікно Add/Remove Source Files... усі необхідні дані, закрийте його, натиснувши на кнопку OK. Скориставшись пунктами Define Code Generation Tool... і Setup External Text Editor... меню Source, можна зробити "тонке настроювання" транслятора і текстового редактора,

використовуваного для перегляду і коректування вихідного тексту. Але в даному випадку цього не потрібно. Параметри, прийняті за замовчуванням, цілком задовільні.

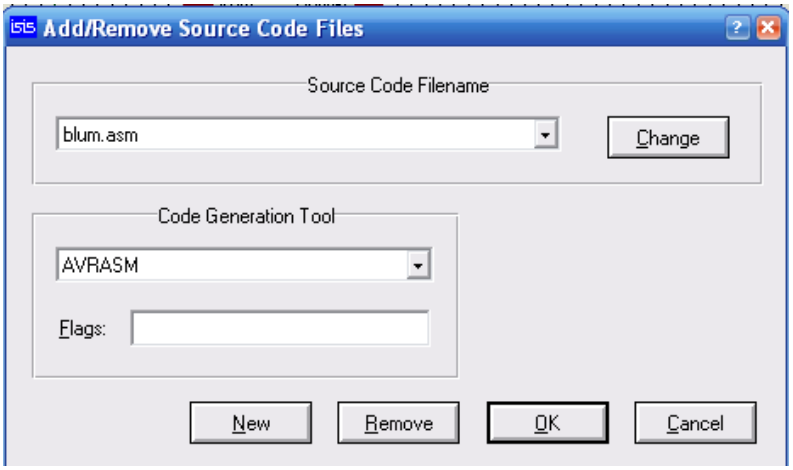


Рисунок 14.5 – Необхідне вікно програми

Запустите трансляцію, вибравши в меню Source пункт Build all. По її закінченні на екрані з'явиться вікно із повідомленням транслятора про виявлені помилки або їхню відсутність. Помилки можна виправити, "клацнувши" по імені вихідного файлу в нижній частині меню Source. Буде відкрите вікно текстового редактора. Внісши необхідні зміни, збережете відкоректований файл і повторите трансляцію (Build all).

Після успішної трансляції в папці blum з'явиться файл blum.HEX, який необхідно "завантажити" у модель мікроконтролера. Для цього, закривши вікно з повідомленнями транслятора, виділіть в схемному вікні компонентів U1 (мікроконтролер). Відкривши вікно його властивостей, вкажіть у якості Program File ім'я створеного Hex-файла. Слово конфігурації (Program configuration word), якщо воно, як у нашому випадку, не було задано у вихідному тексті програми, можна ввести в шістнадцятковому форматі або прийняти пропоноване за замовчуванням.

Отже, усе готово до пуску моделі. У черговий раз збережіть її і натисніть на кнопку  . Модель оживе, програма ISI почне симуляцію роботи спроектованого пристрою. У виводів компонентів з'являться квадрати, кольори яких відповідають діючим логічним рівням: червоний - високий, синій - низький, сірий - невизначений (проміжний). Світлодіод D1 почав мерехтіти.

Зупинимо симуляцію, натиснувши на кнопку  , а потім знову запустимо, цього разу кнопкою  (пауза). У результаті модель буде ініціалізована, але "заторможена" у вихідному стані. У вікні, що з'явилось на екрані відладчика (рис. 14.6) виведений фрагмент вихідного тексту програми. Якщо таке вікно не з'явилося, потрібно поставити "галочку" у пункту CPU Source Code меню Debug.

Рядок програми, що відповідає поточному значенню програмного лічильника (у даному випадку - нульовому), відзначена червоним

трикутником. Вона ж виділена кольором, але виділення можна переміщати по рядках мишею або клавішами керування курсором.

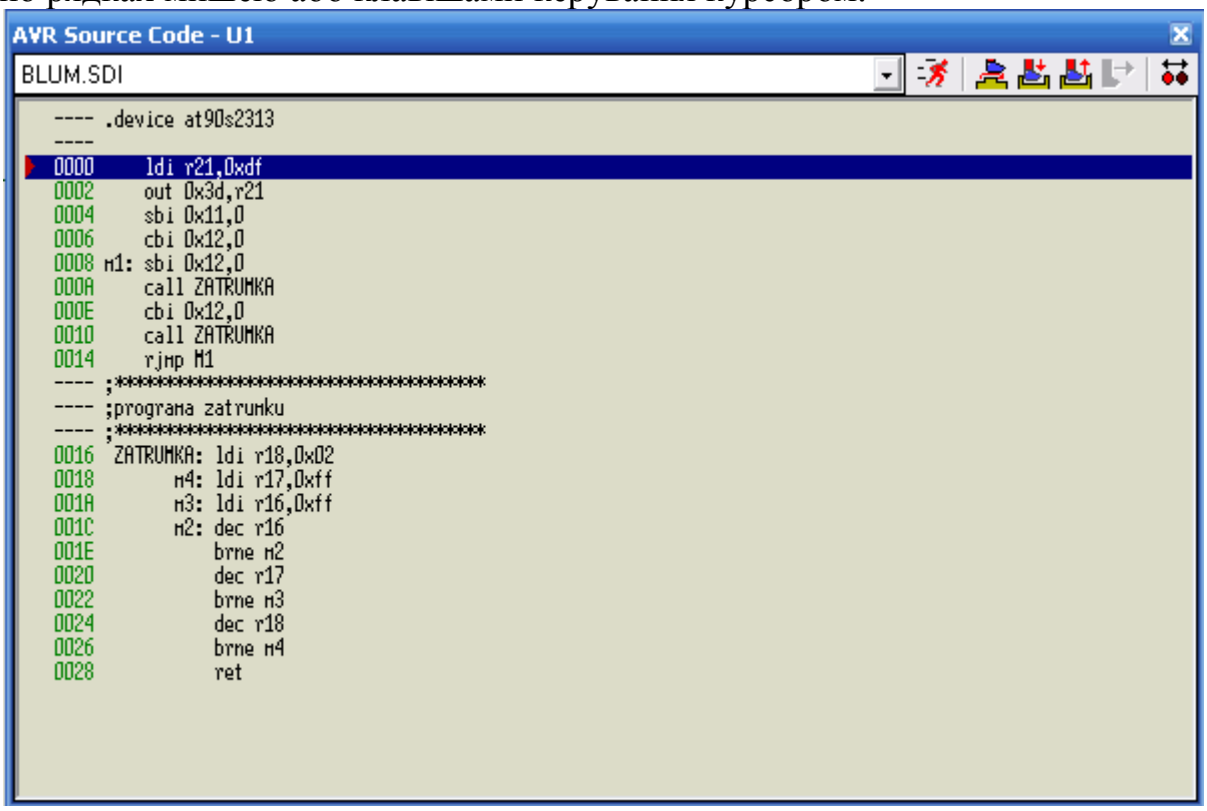


Рисунок 14.6 – Вікно програми

Значки у верхній частині вікна налагодження дають можливість зробити наступне:

- запустити програму в режимі реального часу, починаючи з поточного рядка;
- виконати поточний рядок програми, якщо в ній команда call, виконати всю підпрограму;
- виконати поточний рядок програми;
- якщо поточний рядок знаходиться усередині підпрограми, виконати неї до команди RETURN включно;
- виконати програму, починаючи з поточного рядка, до рядка, виділеної кольором;
- установити контрольну крапку або зняти раніше встановлену.

"Клацнувши" правою кнопкою миші в поле вікна відладчика, можна відкрити меню, що дозволяє знайти рядок програми по заданій адресі або порядковому номері, знайти в програмі заданий текстовий фрагмент, керувати контрольними крапками, включати і виключати відображення деяких параметрів у вікні відладчика (наприклад, номерів рядків). Там же можна вибрати шрифт, яким відображається інформація (для правильного виводу російського тексту - COURIER NEW).

Виділите рядок за адресою 001C. Натиснувши один раз на кнопку , поставте тут контрольну крапку (рис. 14.7). Досягши її в процесі

симуляції, програма буде автоматично зупинена. Рядок з активною контрольною крапкою відзначає червоний овал. Врахуйте, при другому натисканні на кнопку суцільний овал перетвориться в букву О - контрольна крапка стане пасивної, а при третім натисканні зникне - контрольна крапка вилучена.

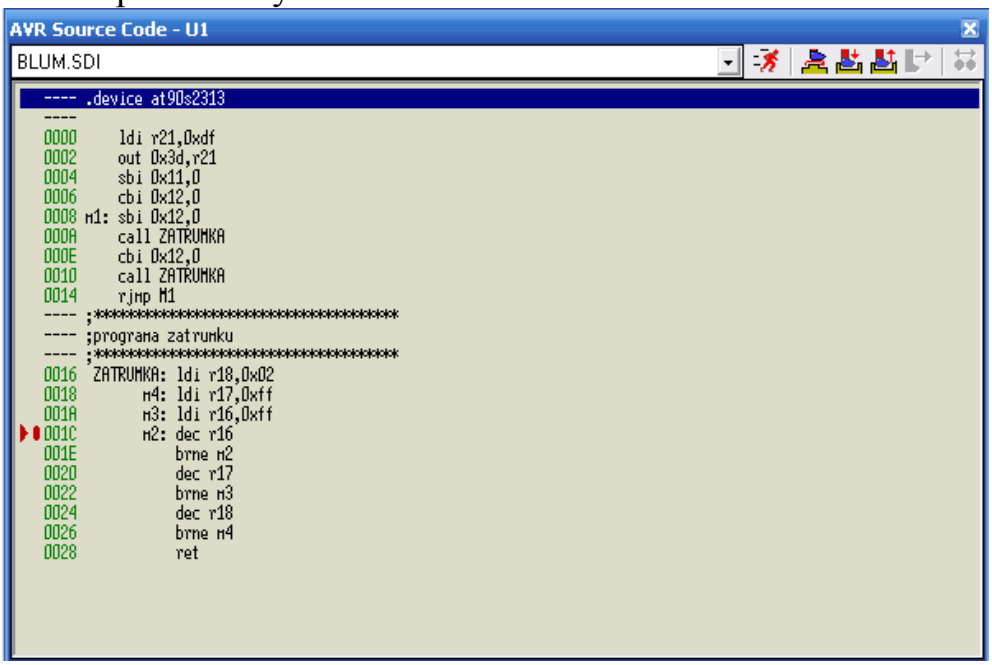


Рисунок 14.7 – Вікно програми

Завдання до лабораторної роботи

1. Розробити програму керування МК на мові «Асемблер» і створити HEX-файл.
2. Спроектувати у програмному пакеті PROTEUS VSM власну схему.
3. Завантажити HEX-файл в папку створеного проекту з огляду на теоретичні вказівки.
4. Перевірити створену схему у проекті на працездатність.
5. Покроково проаналізувати роботу власної схеми.
6. Провести дослідження схеми, за допомогою приладів, шляхом підключення у задані викладачем контрольні точки.

Зміст звіту

Звіт з виконаної роботи повинен містити: мету роботи, завдання, тексти розроблених програм, результати виконання цих програм, зображення схеми у вікні програми, осцилограми визначені в контрольних точках, а також висновки по роботі.

Контрольні запитання

1. Як покроково дослідити створену схему?
2. Які кроки необхідно виконати, щоб створити власний проект?
3. Наведіть методику дослідження сигналів у точках схеми за допомогою осцилографа.

Лабораторна робота № 15

СИНТЕЗ РОБОТИ МІКРОКОНТРОЛЕРА З ВБУДОВАНИМ АНАЛОГО-ЦИФРОВИМ ПЕРЕТВОРЮВАЧЕМ У ПРОГРАМНОМУ ПАКЕТІ PROTEUS VSM

Мета роботи: навчитись створювати свій власний проект, синтезувати схему АЦП по власному варіанту завдання, створювати та реалізовувати підпрограму для мікроконтролера, вміти покроково досліджувати створену схему вносивши в неї зміни.

Теоретичні відомості

Для початку наведемо короткий опис роботи контролера та АЦП.

Отже для реалізації даного пристрою використаємо контролер типу AT90S4433. В таблиці 15.1 наведемо основні команди введення-виведення контролера.

Блок-схема аналого-цифрового перетворювача (Analog to Digital Converter) зображена на рис. 15.1. Для живлення ADC використовуються два окремий виводи: AVCC і AGND. Вивід AGND повинен бути приєднаний до GND і напруга AVCC не повинна відрізнятися від напруги VCC більш ніж на 0,4 В.

Таблиця 15.1 – Команди введення-виведеня контролера AT90S4433

I/O Address (SRAM Address)	Name	Function
$3F ($5F)	SREG	Status REGister
$3D ($5D)	SP	Stack Pointer
$3B ($5B)	GIMSK	General Interrupt MaSK register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag register
$35 ($55)	MCUCR	MCU general Control Register
$34 ($54)	MCUSR	MCU general Status Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$2F ($4F)	TCCR1A	Timer/Counter1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter1 High Byte
$2C ($4C)	TCNT1L	Timer/Counter1 Low Byte
$2B ($4B)	OCR1H	Timer/Counter1 Output Compare Register High Byte
$2A ($4A)	OCR1L	Timer/Counter1 Output Compare Register Low Byte
$27 ($47)	ICR1H	Timer/Counter1 Input Capture Register High Byte
$26 ($46)	ICR1L	Timer/Counter 1 Input Capture Register Low Byte
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1E ($3E)	EEAR	EEPROM Address Register
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$18 ($38)	PORTB	Data Register, Port B

I/O Address (SRAM Address)	Name	Function
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$15 ($35)	PORTC	Data Register, Port C
$14 ($34)	DDRC	Data Direction Register, Port C
$13 ($33)	PINC	Input Pins, Port C
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D
$0F ($2F)	SPDR	SPI I/O Data Register
$0E ($2E)	SPSR	SPI Status Register
$0D ($2D)	SPCR	SPI Control Register
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	UCSRA	UART Control and Status Register A
$0A ($2A)	UCSRB	UART Control and Status Register B
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register
$07 ($27)	ADMUX	ADC Multiplexer Select Register
$06 ($26)	ADCSR	ADC Control and Status Register
$05 ($25)	ADCH	ADC Data Register High
$04 ($24)	ADCL	ADC Data Register Low
$03 ($23)	UBRRHI	UART Baud Rate Register High

Таблиця 15.2 – Вектори скиду й переривань

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMP	Timer/Counter1 Compare Match
6	$005	TIMER1 OVF	Timer/Counter1 Overflow
7	$006	TIMER0 OVF	Timer/Counter0 Overflow
8	$007	SPI, STC	Serial Transfer Complete
9	$008	UART, RX	UART, Rx Complete
10	$009	UART, UDRE	UART Data Register Empty
11	$00A	UART, TX	UART, Tx Complete
12	$00B	ADC	ADC Conversion Complete
13	$00C	EE_RDY	EEPROM Ready
14	$00D	ANA_COMP	Analog Comparator

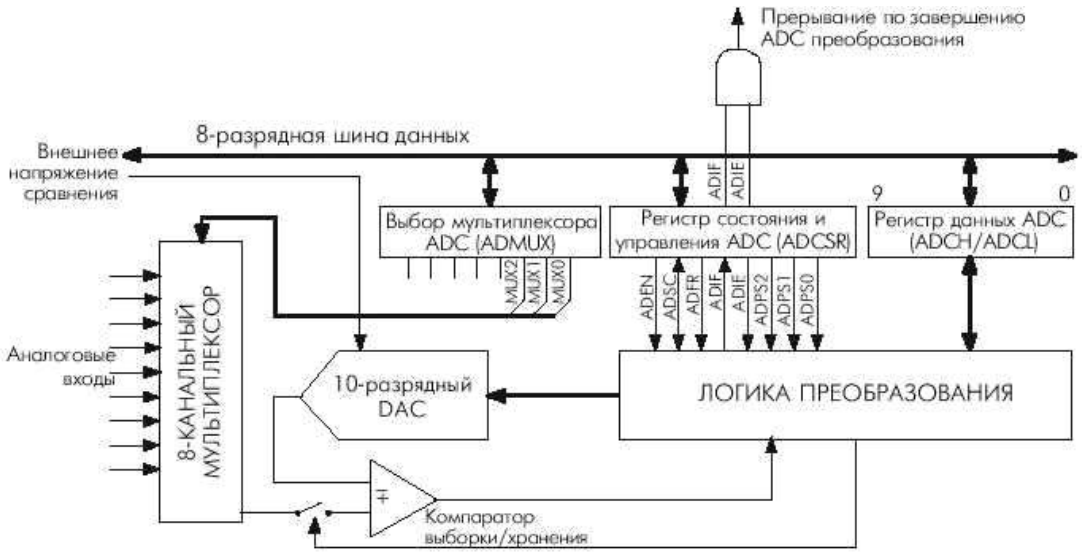


Рисунок 15.1 – Блок-схема аналого-цифрового перетворювача

Зовнішня порівняльна напруга подається на вивід AREF і повинна знаходитись в діапазоні від 2,7 В до AVCC.

Аналого-цифровий перетворювач може працювати в двох режимах режимі однократного перетворення і режимі циклічного перетворення. В режимі однократного перетворення кожне перетворення ініціюється користувачем. У режимі циклічного перетворення ADC здійснює вибірку і відновлення вмісту регістра даних ADC безупинно. Вибір режиму задається бітом ADFR регістра ADCSR.

Робота ADC дозволяється установкою в стан 1 біта ADEN у регістрі ADCSR. Першому перетворенню, що починається після дозволу ADC, передує порожнє ініціалізуюче перетворення. На користувачі це відбивається лише тим, що перше перетворення буде займати 27 тактових циклів, замість звичайних 14.

Перетворення починається з установки в стан 1 біта початку перетворення ADSC. Цей біт знаходиться в стані 1 протягом усього циклу перетворення і скидається, по завершенні перетворення, апаратно. Якщо в процесі виконання перетворення виконується зміна каналу даних, то ADC спочатку закінчить поточне перетворення і лише потім виконає перехід до іншого каналу.

Оскільки ADC формує 10-розрядний результат, то по завершенні перетворення результуючі дані розміщаються в двох регістрах даних ADCH і ADCL. Для забезпечення відповідності результуючих даних рівневі, що зчитується, використовується спеціальна логіка захисту. Цей механізм працює в такий спосіб: при зчитуванні даних першим повинний бути лічений регістр ADCL. Як тільки ADCL зчитаний звертання ADC до регістрів даних блокується. Таким чином, якщо після зчитування стану ADCL, але до зчитування ADCH, буде довершене наступне перетворення, жоден з регістрів не буде обновлений і записаний раніше результат не буде спотворений. Звертання ADC до регістрів ADCH і ADCL дозволяється по завершенні зчитування вмісту регістра ADCH.

ADC має своє власне переривання, що може бути активоване по завершенню перетворення. Коли звертання ADC до регістрів заборонено, у процесі зчитування регістрів ADCL і ADCH, переривання буде активуватися, навіть якщо результат буде загублений.

Скориставшись попередньо отриманою інформацією створимо схему запропонованого пристрою у програмному пакеті PROTEUS.

Запустимо програму ISI створивши новий проект.

Збережем новий проект (поки порожній), вибравши в меню File пункт Save Design. Перед виконанням цієї операції система попросить змінити ім'я, привласнене за замовчуванням (UNTI-TLED.DSN - безіменний), будь-яким іншим на вибір користувача. Зберегти проект можна й у папці, пропонованою системою, але краще створити для нього спеціальну, назвавши її, наприклад, «adcdac». Надалі саме в ній буде зберігатися всі файли, зв'язані з проектом, у тому числі програма мікроконтролера і допоміжні файли, створювані системою в процесі роботи.

Роботу над моделлю почніть з підбору необхідних компонентів. Знайдіть у бібліотеці наступні елементи: в якості живлення застосуємо батарею «BATTERY», також нам знадобиться мікросхема цифрово-аналогового перетворювача «DAC_12», конденсатор «CAPACITOR», та котушка індуктивності «INDUCTOR». Простіше всього це робити, набираючи умовні найменування компонентів у вікні Keywords.

Нагадаємо, щоб перенести знайдений компонент у вікно селектора об'єктів, потрібно двічі "клацнути" по рядку з його назвою в бібліотеці. У результаті повинний вийти список, показаний на рис. 15.2.

Розміщаючи і з'єднуючи компоненти на екрані комп'ютера в основному вікні програми, відкривайте вікна властивостей кожного з них і

задавайте значення. В нашому випадку задамо ємність конденсатора 100 nF, а індуктивність 10 uH, живлення першої та другої батареї задамо 24 та 6 V відповідно.

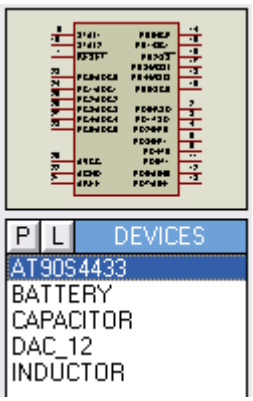


Рисунок 15.2 – Необхідний список елементів

З властивостей мікроконтролера обов'язково вкажіть Processor Clock Frequency (тактову частоту процесора) - 10 Мгц. Ім'я Hex-файла програми (Program file) можна задати відразу, якщо він готовий, або пізніше, після того як він буде створений.

Для "заземленя" буде потрібно спеціальний компонент GROUND. Візьміть його зі списку, що з'являється у вікні селектора об'єктів при натисканні на кнопку . Крім того, додайте в зібрану модель осцилограф.

У результаті повинно вийти подібне зображення, що показано на рис. 15.3. "Підводити" до моделі мікроконтролера живлення не потрібно, хоча при необхідності можна зробити і це.

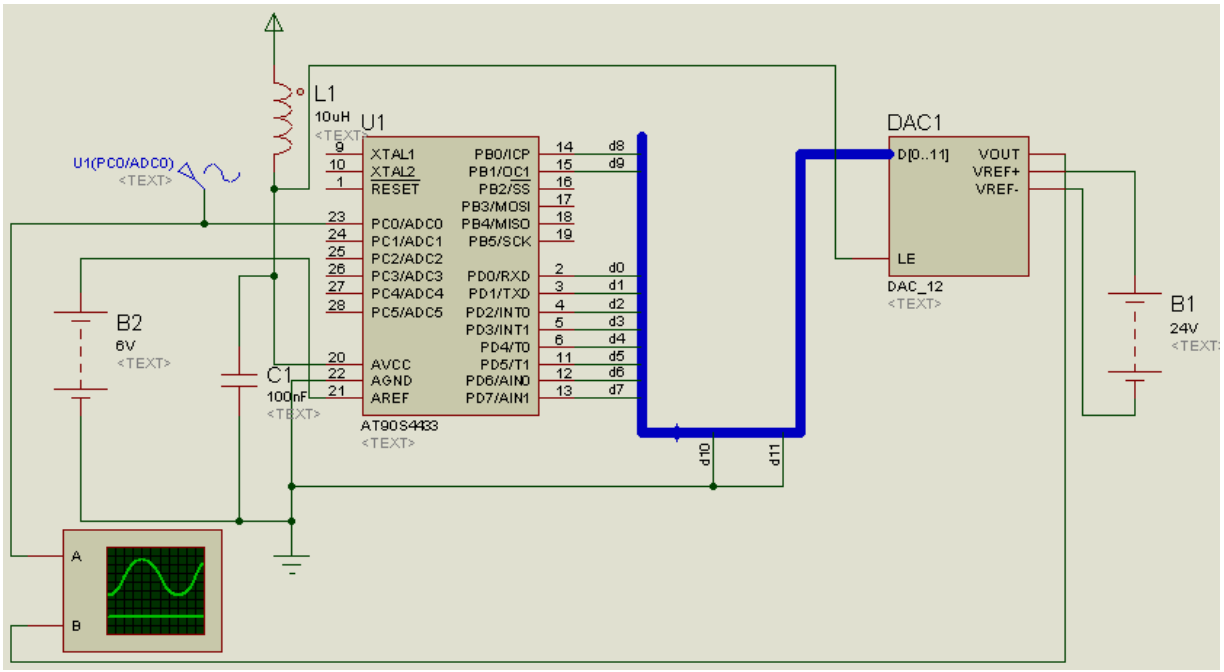


Рисунок 15.3 – Зібрана схема

Збережіть створений проект. Цю операцію рекомендується робити частіше, особливо перед початком налагодження готової моделі або перед внесенням у неї змін. Це звільнить від необхідності повторювати всю пророблену роботу у випадку збою.

Підготуємо програму для мікроконтролера і завантажимо її в модель. Вихідний текст програми мовою ассемблера AVRASM, по якій повинен працювати "сигналізатор", наведений на рис. 15.4. Наберіть його в будь-якому текстовому редакторі і збережіть в папці проекту (adcdac) під ім'ям «adcdac.asm».

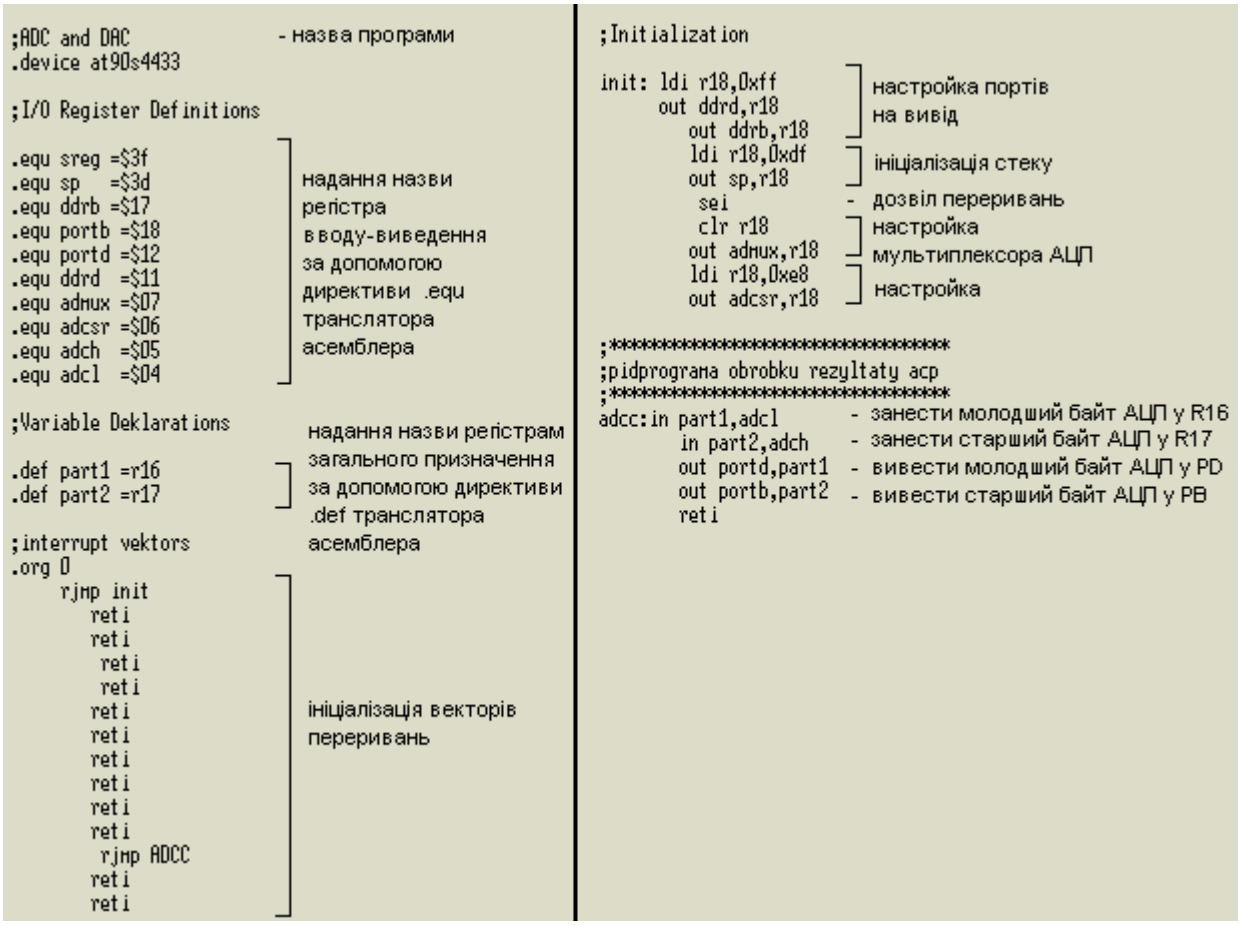


Рисунок 15.4 – Вихідний текст програми

Розібравшись й створивши програму, відкрийте пункт Add/Remove Source Files... у меню Source. Буде відкрите вікно, показане на рис. 15.5, у якому необхідно натиснути на екранну кнопку New, потім увести Source Code Filename - ім'я файлу вихідного тексту програми. Знайти його допоможе кнопка Change. У відповідь на її натискання на екран буде виведене вікно, що дозволяє вибрати потрібний файл із наявних на дисках комп'ютера.

Далі виберіть Code Generation Tool - засіб трансляції програми. Їхній список "випадає" при натисканні на кнопку зі стрілкою у відповідному вікні. У даному випадку необхідний ассемблер AVRASM.

Ввівши у вікно Add/Remove Source Files... усі необхідні дані, закрийте його, натиснувши на кнопку OK. Скориставшись пунктами Define Code Generation Tool... і Setup External Text Editor... меню Source, можна зробити "тонке настроювання" транслятора і текстового редактора, використовуваного для перегляду і коректування вихідного тексту. Але в даному випадку цього не потрібно. Параметри, прийняті за замовчуванням, цілком задовільні.

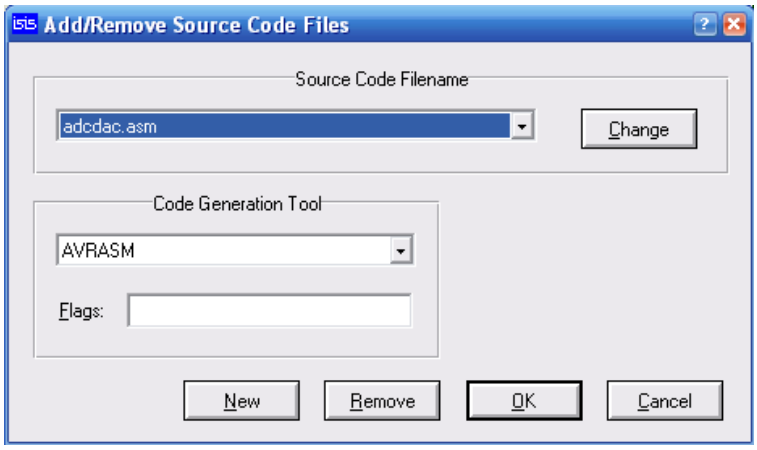


Рисунок 15.5 – Необхідне вікно програми

Запустіть трансляцію, вибравши в меню Source пункт Build all. По її закінченні на екрані з'явиться вікно із повідомленням транслятора про виявлені помилки або їхню відсутність. Помилки можна виправити, "клацнувши" по імені вихідного файлу в нижній частині меню Source. Буде відкрите вікно текстового редактора. Внісши необхідні зміни, збережете відкоректований файл і повторите трансляцію (Build all).

Після успішної трансляції в папці «adcdac» з'явиться файл adcdac.HEX, який необхідно "завантажити" у модель мікроконтролера. Для цього, закривши вікно з повідомленнями транслятора, виділіть в схемному вікні компонентів U1 (мікроконтролер). Відкривши вікно його властивостей, вкажіть у якості Program File ім'я створеного Hex-файла. Слово конфігурації (Program configuration word), якщо воно, як у нашому випадку, не було задано у вихідному тексті програми, можна ввести в шістнадцятковому форматі або прийняти пропоноване за замовчуванням.

Отже, усе готово до пуску моделі. У черговий раз збережіть її і натисніть на кнопку  . Модель оживе, програма ISI почне симуляцію роботи спроектованого пристрою. У виводів компонентів з'являться квадрати, кольори яких відповідають діючим логічним рівням: червоний - високий, синій - низький, сірий - невизначений (проміжний).

Завдання до лабораторної роботи

1 Ознайомитись з даними теоретичними відомостями.

2. Розробити програму керування МК на мові «Асемблер» і створити HEX-файл.

3. Спроектувати у програмному пакеті PROTEUS VSM власну схему.

4. Завантажити HEX-файл в папку створеного проекту з огляду на теоретичні вказівки.

5. Перевірити створену схему у проекті на працездатність.

6. Покроково проаналізувати роботу власної схеми.

7. Провести дослідження схеми, за допомогою приладів, шляхом підключення у задані викладачем контрольні точки.

Зміст звіту

Звіт з виконаної роботи повинен містити: мету роботи, тексти розроблених програм, результати виконання цих програм, зображення схеми у вікні програми, осцилограми визначені в контрольних точках, а також висновки по роботі.

Контрольні запитання

1. Наведіть блок-схему АЦП мікроконтролера.
2. Наведіть алгоритм роботи лістингу програми.
3. Поясніть принцип роботи АЦП, що вбудований в мікроконтролер.

Лабораторна робота №16

ОРГАНІЗАЦІЯ ДИНАМІЧНОЇ ІНДИКАЦІЇ

Мета роботи: вивчення особливостей роботи динамічних цифрових індикаторів. Розробка програм для мікроконтролера AT90S2313 відображення інформації на індикаторах динамічного типу.

Теоретичні відомості

Дуже часто мікроконтролер використовується не тільки для управління роботою конструкції, але і для того, щоб повідомити що-небудь користувачеві і/або отримати від нього які-небудь вказівки про подальшу роботу. Наприклад, електронний годинник, крім власне відліки часу, повинен його ще відображати, а також дозволяти змінювати покази (встановлювати точний час). Якщо вся "інформація" зводиться до мигання парою світлодіодів, яких-небудь спеціальних зусиль по відображенню інформації з боку розробника конструкції не вимагається, але якщо таких світлодіодів виявляється два-три десятки, тут вже потрібне застосування додаткових засобів - як апаратних, так і програмних. Як правило, в цьому

випадку відображення інформації виконують у режимі динамічний індикації - це найбільш економний по числу використаних ліній спосіб.

Загальний принцип динамічний індикації - це матриця, що складається з ліній рядків і ліній стовпчиків (рисунок 16.1). На перетині стовпців і рядків матриці розташований індикаторний елемент - світлодіод. Для того, щоб запалити той або інший елемент, необхідно подати на матрицю не один, як в звичайних індикаторах, а два сигнали: у нашому прикладі це буде логічна 1 на відповідному рядку і логічний 0 на відповідному стовпці матриці. Через односторонню провідність світлодіода кожна комбінація сигналів на входах рядків і стовпців однозначно включає рівно один індикаторний елемент.

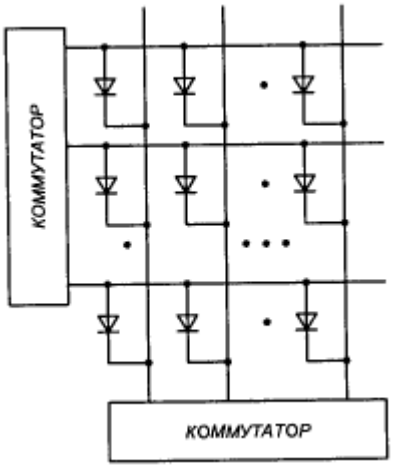


Рисунок 16.1 – Загальний принцип динамічної індикації

Головна перевага динамічної індикації - невелике число ліній, що управляють: для матриці світлодіодів розміром N x N елементів потрібно всього 2N сигналів, що управляють. За таку економію, втім, доводиться платити - справа в тому, що при почерговому виведенні інформації на кожен світлодіод матриці його яскравість світіння, буде в N2 разів нижче, ніж при безпосередньому виведенні інформації на один світлодіод, що "окремо стоїть". Тому в пристроях, що використовують динамічну індикацію, виведення інформації здійснюється не на кожен світлодіод окремо, а на один рядок або на один стовпець цілком - в цьому випадку яскравість світіння світлодіодів падает тільки в N разів.

Розглянемо декілька варіантів реалізації динамічної індикації із застосуванням мікроконтролера. Усюди як індикаторний елемент передбачається застосування семисегментних індикаторів, але кожен такий індикатор з легкістю можна замінити і групою світлодіодів.

Схема реалізації динамічної індикація без додаткових елементів приведена на рисунку 16.2. До порту В мікроконтролера підключені катоди всіх світлодіодів матриці, а до порту А – аноди кожного індикатора, що створюють матрицю. На лініях порту А організовується одиниця, що "біжить". На лінії В порту при кожному положенні одиниці, що біжить, виводиться семисегментний код того символу, який повинен горіти в

даному знакомісці. Для індикаторів із загальним катодом замість одиниці, що біжить, використовується нуль, що біжить. Перевага такого способу індикації - у відсутності яких-небудь додаткових компонент (окрім самих світлодіодних індикаторів), головний недолік – значна перевитрата ліній портів. Таке рішення для мікроконтролера може забезпечити роботу не більше 5 семисегментних індикаторів одночасні, да і то при цьому мікроконтролер стає "глухий і сліпий", оскільки жодної вільної лінії у нього не залишається. При використанні інших мікроконтролерів з великою кількістю ніжок -например ATTiny28, - вказана проблема знімається.

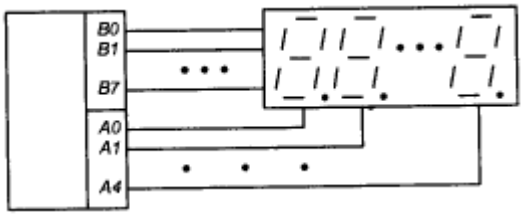


Рисунок 16.2 – Схема реалізації динамічної індикація без додаткових елементів

Схема реалізації динамічної індикації з одним додатковим елементом приведена на рисунку 16.3. До порту B мікроконтролера підключені катоди всіх світлодіодів матриці, а також - входи буферного регістра. До виходів же буферного регістра підключені аноди кожного з семисегментних індикаторів. На порт B спочатку виводиться одиниця, що біжить, яка записується в буферному регістрі сигналом C, що утворюється однією з ліній порту A. Потім на лінії порту B видається семисегментний код символу, який повинен горіти в знакомісці, визначуваному сигналом на виході буферного регістра. В даному випадку лінії порту B використовуються в режимі часового мультиплексування, тобто по ним по черзі передається і код символу, і номер знакомісця. Перевага такого способу індикації - менша витрата ліній портів мікроконтролера (окрім порту B - всього одна додаткова лінія) і можливість роботи до 8 індикаторів одночасно.

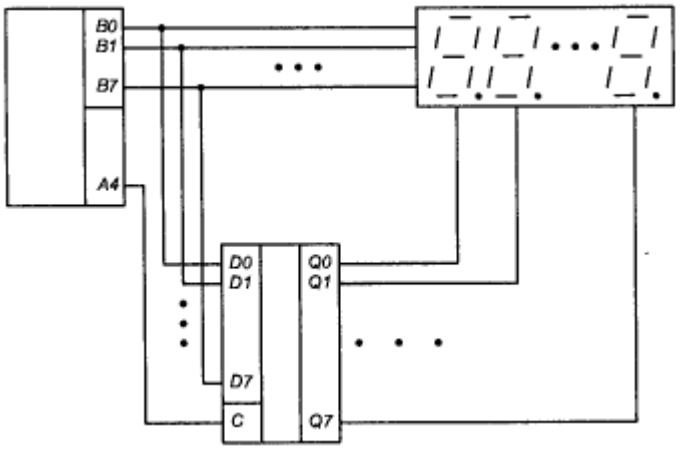


Рисунок 16.3 – Схема реалізації динамічної індикації з одним додатковим елементом

Ще один варіант індикації з одним додатковим елементом приведений на рисунку 16.4. У цьому варіанті одиниця, що біжить, реалізується за допомогою регістра зсуву**.** Порт В у цій схемі також використовується в режимі часового мультиплексування - як для видачі символу, так і для занесення чергового біта в регістр зсуву. Схема також вимагає всього одну додаткову лінію (крім порту В), але за габаритами виходить трохи менше попереднього варіанту.

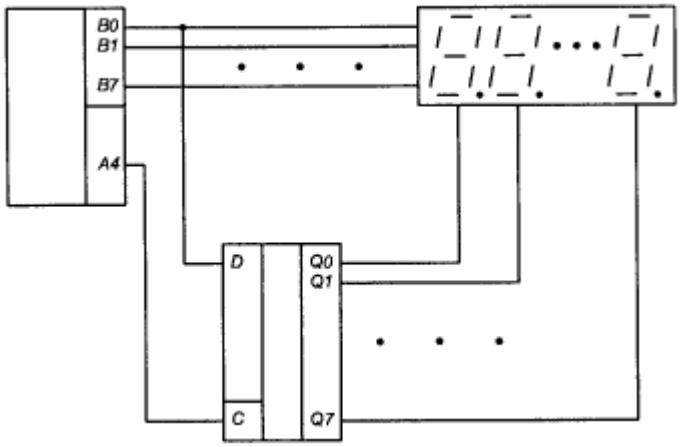


Рисунок 16.4 – Схема реалізації динамічної індикації з регітром зсуву

Ще один варіант реалізації схеми з одним додатковим елементом приведений на рисунку 16.5. Така схема придатна тільки для індикаторів із загальним катодом, оскільки для організації нуля, що біжить, в ній використовується дешифратор. Схема вимагає три додаткові лінії (крім порту В), проте у багатьох випадках вона може виявитися цілком корисною.

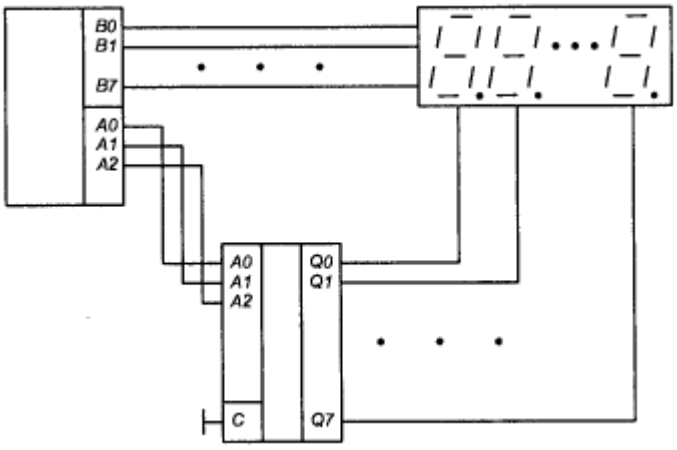


Рисунок 16.5 – Схема реалізації динамічної індикації з дешифратором

Схема реалізації динамічної індикації з двома додатковими елементами приведена на рисунку 16.6. У схемі використовуються два послідовні регістри зсуву: один - для розгортки зображення по стовпцях

(замінює порт А), інший - для розгортки зображення по рядках (замінює порт В). Перевага такого рішення – всього три лінії порту мікроконтролера. Крім того, при такому варіанті реалізації блок динамічної індикації легко оформити і у вигляді окремої плати. Недолік такого рішення - два додаткових компонента.

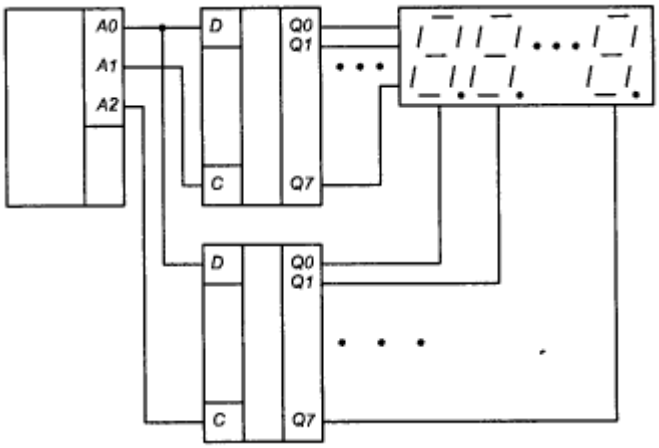


Рисунок 16.6 – Схема реалізації динамічної індикації з двома додатковими елементами

Ще одна схема реалізації приведена на рисунку 16.7. Перевага такої схеми - всього дві лінії порту. Недолік - складніша програма управління і велика тривалість формування вихідних сигналів, що викликає деяке паразитне підсвічування індикаторів (помітно тільки в темноті).

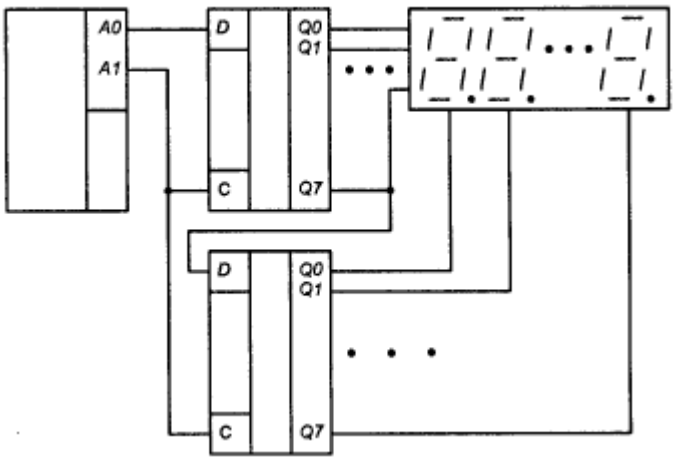


Рисунок 16.7 – Схема динамічної індикації з двома лініями керування

Практична схема підключення динамічного індикатора на мікроконтролері AT90S2313 за схемою, що приведена на рисунку 16.2, зображена на рисунку 16.8. Символ, що буде світитись на іфндикаторі визначається сигналом логічної одиниці на вивід COM1,2,3 або 4 індикатора. На шині даних (A, B, C …) активним рівнем являється рівень логічного 0.

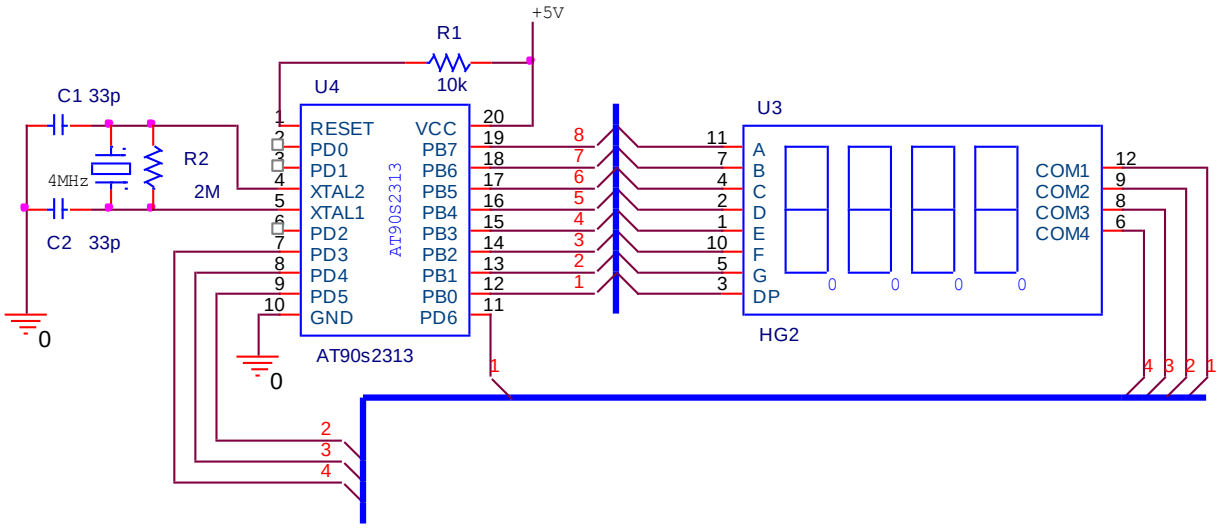


Рисунок 16.8 – Підключення динамічного індикатора до мікроконтролера AT90S2313

Завдання до лабораторної роботи

1. Запустити в Proteus файл Din_IND. Підключити файл Din_IND.HEX. Перевірити як працює схема. Розібрати особливості програми Din_IND.ASM. Скласти алгоритм роботи програми.
2. Запустити в Proteus файл Din_IND_KLAV. Підключити файл INDICA~1.HEX. Перевірити як працює схема. Розібрати особливості програми INDICATE+KEYBOARD.ASM. Скласти алгоритм роботи програми.
3. Запустити в Proteus файл Din_IND. Підключити файл DIN_OGON.HEX. Перевірити як працює схема. Розібрати особливості програми DIN_OGON.ASM. Підключити файл DIN_TEN.HEX. Перевірити як працює схема. Розібрати особливості програми DIN_TEN.ASM. Підключити файл. DIN_TO.HEX. Перевірити як працює схема. Розібрати особливості програми DIN_TO.ASM.
4. Використовуючи програми DIN_OGON.ASM, DIN_TEN.ASM, DIN_TO.ASM написати програму, яка створює ефект «бігучий вогник» по зовнішньому колу (для непарних варіантів) та ефект «бігуча тінь» по зовнішньому колу (для парних варіантів) – оцінка добре. *Номер схеми (рисунок 2– рисунок 7) призначає викладач.*
5. Виконати п.4 для 6 динамічних індикаторів – оцінка відмінно.

Зміст звіту

1. Алгоритм роботи програми та схема до п. 1.
2. Алгоритм роботи програми та схема до п. 2.
3. Схема, алгоритм, програма та результати виконання п. 4 або п. 5.

Контрольні запитання

1. Розкажіть про принцип динамічної індикації та наведіть його схему.
2. Наведіть схемну реалізацію динамічної індикації без додаткових елементів.
3. Наведіть схемну реалізацію динамічної індикації з одним додатковим елементом.
4. Наведіть схемну реалізацію динамічної індикації з регітром зсуву.
5. Наведіть схемну реалізацію динамічної індикації з дешифратором.
6. Наведіть схемну реалізацію динамічної індикації з двома лініями керування.

Лабораторна робота №17

РОБОТА З ТАЙМЕРАМИ-ЛІЧИЛЬНИКАМИ

Мета роботи: Вивчення особливостей роботи таймерів-лічильників TC0/TC1 мікроконтролера AT90S2313. Отримання навиків формування часових інтервалів.

Теоретичні відомості

1. Таймери-лічильники загального призначення

Таймер-лічильник T/CX (X = 0, 1, 2- цифра в імені таймера-лічильника) будь-якого типу містить базовий лічильник TCNTX, що має вісім чи шістнадцять розрядів, і восьмирозрядний регістр керування TCCRX. Регістри TIFR і TIMSK є загальними для всіх таймерів-лічильників мікроконтролера.

Розряд регістра TIFR встановлюється в одиничний стан при формуванні в таймері-лічильнику визначеного запиту переривання. Запит переривання проходить у блок переривань при одиничному стані відповідного розряду регістра TIMSK. Розряд регістра TIFR скидається в нульовий стан апаратно при переході мікроконтролера до виконання відповідної програми переривання.

До складу таймера-лічильника, який виконує функцію порівняння/PWM, входить регістр порівняння OCRX, а до складу таймера-лічильника, що виконує функцію захоплення, - регістр захоплення ICRX. Розрядність регістрів OCRX і ICRX дорівнює розрядності базового лічильника TCNTX.

Для запису коду в шістнадцятирозрядний лічильник чи регістр спочатку виконується команда запису (OUT) байта в старшу половину розрядів (H), при цьому байт, що надходив з регістра загального призначення старший, запам'ятовується в регістрі тимчасового збереження. Потім виконується команда запису (OUT) молодшого байта в молодшу половину розрядів (L), при цьому обидва байти одночасно записуються в лічильник чи регістр.

Для читання коду із шістнадцятирозрядного лічильника чи регістра спочатку виконується команда читання (IN) байта з молодшої половини розрядів (L), при цьому зчитаний молодший байт надходить у регістр загального призначення, а старший байт запам'ятовується в регістрі тимчасового збереження. Потім виконується команда читання байта зі старшої половини розрядів (H), при цьому старший байт із регістра тимчасового збереження надходить у зазначений у команді регістр загального призначення.

Таймер-лічильник типу А

Таймер-лічильник типу А є в мікроконтролерів усіх типів. Він має ім'я T/C0 (X = 0). Таймер-лічильник типу А формує запит переривання T/C0 OVF при переповненні восьмирозрядного базового лічильника TCNTO. Структурна схема таймера-лічильника типу А зображена на рис.17.1.

Тактовий сигнал мікроконтролера СК надходить у перерахункову схему (prescaler) ПС, що представляє собою десятирозрядний лічильник, де виконується розподіл частоти тактового сигналу на 8, 64, 256 і 1024. Сигнали з чотирьох виходів перерахункової схеми надходять у схему керування СК (мультиплексор). При наявності в мікроконтролері таймера-лічильника T/C1 ці ж сигнали надходять у T/C1.

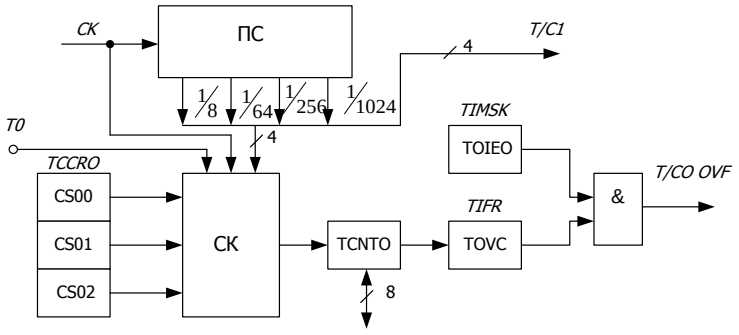


Рисунок 17.1 – Структурна схема таймера-лічильника типу А

У схему керування надходять також тактовий сигнал СК і сигнал із зовнішнього джерела, прийнятий на вхід T0.

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

Попередній подільник таймерів-лічильників 0 і 1 містить чотири ступені розподілу: СК/8, СК/64, СК/256 і СК/1024, де СК - вхідний тактовий сигнал (рис. 17.2). Крім того, як джерела тактових сигналів можуть бути використані сигнали від зовнішніх джерел, тактовий сигнал СК і нульовий тактовий сигнал (STOP).

У регістрі прапорців переривання таймерів-лічильників TIFR зберігаються різні прапори стану регістрів (переповнення, збігу при порівнянні і захоплення події). Установки керуючих сигналів зберігаються в регістрах керування таймерами-лічильниками TCCR0 і TCCR1. Установка дозволу/заборони переривань виробляється в регістрі масок переривань таймерів-лічильників TIMSK.

Для забезпечення правильної синхронізації зовнішнього сигналу необхідно, щоб мінімальний час між двома вхідними тактовими циклами був не менше одного циклу внутрішнього тактового сигналу CPU. Зовнішній тактовий сигнал синхронізується наростаючим фронтом внутрішнього тактового сигналу CPU.

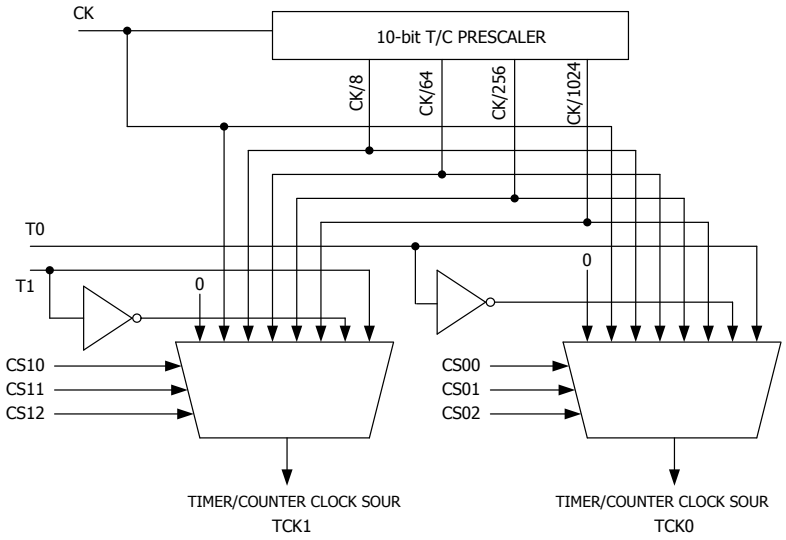


Рисунок 17.2 - Попередній подільник таймера-лічильника

Точність 8-розрядних таймерів-лічильників росте зі зменшенням коефіцієнта попереднього розподілу. Аналогічним образом високий коефіцієнт попереднього розподілу зручно використовувати при реалізації функцій з низькою швидкодією. Обидва таймери-лічильника підтримують дві функції порівняння виходу OCR0 і OCR1 як джерела даних, порівнюваних із умістом таймерів-лічильників (рис. 17.3).

Таймери-лічильники 0 і 1 можна використовувати як 8-розрядні широтно-імпульсні модулятори (PWM). У цьому режимі таймер-лічильник, разом з регістром збігу виходу працюють як автономний ШІМ з центрованими імпульсами і без помилкових викидів (табл. 17.1-17.3).

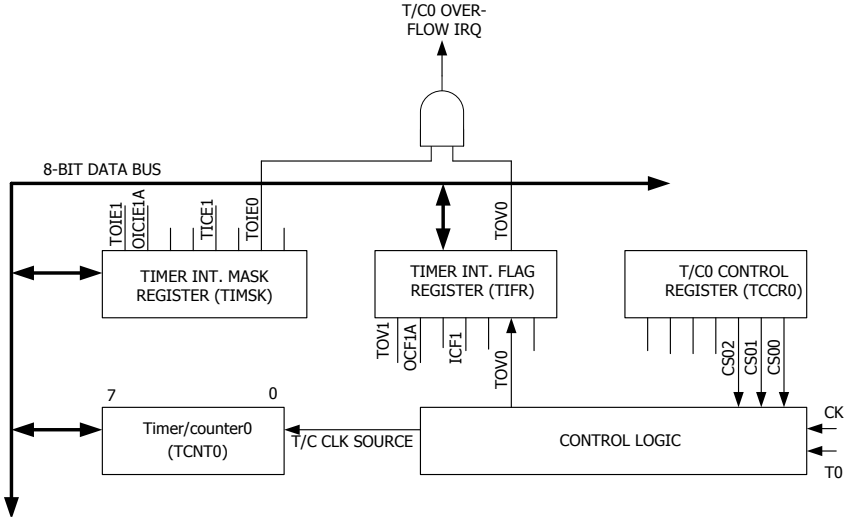


Рисунок 17.3 - Блок-схема таймера-лічильника 0

Таблиця 17.1 – Регістр керування таймером-лічильником 0 - TCCR0 - (The Timer/Counter0 Control Register)

Біти	7	6	5	4	3	2	1	0	
($53)						**CS02**	**CS01**	**CS00**	**TCCR0**
Чит./Зап.	R	R	R	R	R	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Таблиця 17.2 - Керування таймером/лічильником 0 - TCCR0($33)

CS02	CS01	CS00	Опис
0	0	0	Таймер лічильник 0 зупинений
0	0	1	СК
0	1	0	СК/8
0	1	1	СК/64
CS02	CS01	CS00	Опис
1	0	0	СК/256
1	0	1	СК/1024
1	1	0	Зовнішній вхід T0, спадаючий фронт
1	1	1	Зовнішній вхід T0, наростаючий фронт

Таблиця 17.3 – Таймер-лічильник 0 - TCNT0 - (Timer/Counter0)

Біти	7	6	5	4	3	2	1	0	
$32 ($42)	**MSB**							**LSB**	**TCNT0**
Чит/Зап	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Поч. стан	0	0	0	0	0	0	0	0	

Таймер-лічильник типу D

Таймер-лічильник типу D входить до складу периферійних пристроїв мікроконтролерів типу 2313 і 4433, і має ім'я T/C1. Він містить шістнадцатирозрядний базовий лічильник і виконує функції захоплення і порівняння/PWM. Структурна схема таймера-лічильника зображена на рис. 17.4.

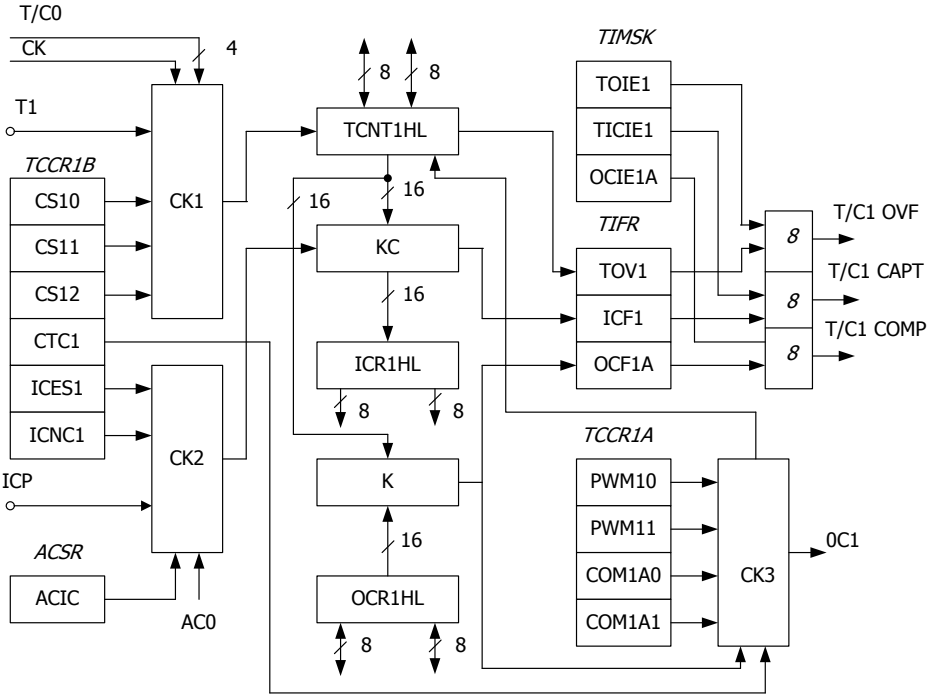


Рисунок 17.4– Структурна схема таймера-лічильника типу D

На рахунковий вхід шістнадцатирозрядного базового лічильника TCNT1H, L з виходу схеми керування СК1 може надходити тактовий сигнал мікроконтролера СК, чи один з чотирьох сигналів з перерахункової схеми, що є загальної для таймерів-лічильників T/C і T/C1, чи сигнал із зовнішнього джерела, прийнятий на вхід T1. Як вхід T1 використовується вивід порту PD5. Вибір сигналу визначається комбінацією станів розрядів CS10, CS11 і CS12 регістра керування TCCR1B відповідно до табл. 17.4 (X = 1, TE=T1). При переповненні базового лічильника встановлюється в одиничний стан розряд TOV1 регістра TIFR і при одиничному стані розряд TOI1 регістра TIMSK у блок перериванні надходить запит переривання T/C1 OVF.

Схема керування СК2 керує виконанням функції захоплення, що полягає в передачі коду, сформованого в базовому лічильнику, через ключову схему КС у шістнадцятирозрядний регістр захоплення ICR1H, L. При цьому встановлюється в одиничний стан розряд ICF1 регістра TIFR і при одиничному стані розряду TICI1 регістра TIMSK у блок переривань надходить запит переривання Т/С1 CAPT.

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

Розряд ICNC1 регістра TCCR1B керує роботою схеми подавлення завад. При ICNC1 = 0 захоплення виконується з кожною появою фронту заданої полярності.

При ICNC1 = 1 захоплення відбувається, якщо перед появою фронту на протязі чотирьох тактів сигнал зберігає постійне значення.

Таблиця 17.4 - Регістр керування А таймера-лічильника 1- TCCR1A (Timer/Counter1 Control Register A)

Біти	7	6	5	4	3	2	1	0	
$2F ($4F)	**COM1A1**	**COM1A0**	-	-	-	-	**PWM11**	**PWM10**	**TCCR1A**
Читання /Запис	R/W	R/W	R	R	R	R	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Bits 7, 6 - COM1A1, COM1A0: Compare Output Mode1A, bits 1 and 0 –Режим 1А порівняння виходу, біти 1 і 0. Керуючі біти COM1A1 і COM1A0 визначають характер сигналу виходу, що випливає за збігом при порівнянні таймера-лічильника 1 (табл. 17.5). Сигнал виходу надходить на вивід OC1A (Output Compare). Оскільки це альтернативна функція порту I/O, то відповідний біт керування напрямком повинний бути встановлений у 1 (вивід працює на вихід).

Таблиця 17.5 - Вибір режиму порівняння 1

COM1X1	COM1X0	Опис
0	0	Таймер/лічильник1 відключений від виходу OC1X
0	1	Переключення вихідної лінії OC1X
1	0	Очищення вихідної лінії OC1X (на лінії низький рівень)
1	1	Установка вихідної лінії OC1X (на лінії високий рівень)

Bit 7 - ICNC1: Input Capture1 Noise Canceller (4 CKs) –Установка режиму подавлення шуму на вході захоплення 1. При скинутому в стан 0 біті ICNC1 функція подавлення шуму вхідного тригера захоплення заборонена (рис. 17.6). Вхід захоплення переключається по першому наростаючому/спадаючому фронту, що надійшов на вивід входу захоплення PD4 (IC1). При встановленому в стан 1 біті ICNC1 виконуються чотири послідовних опитування стану висновку PD4(IC1) і всі чотири вибірки повинні мати однаковий (високий/низький), обумовлений бітом ICES1, рівень. Частота опитування відповідає частоті XTAL.

Таблиця 17.6 - Регістр керування B таймера-лічильника 1 - TCCR1B - (Timer/Counter1 Control Register B)

Біти	7	6	5	4	3	2	1	0	
$2E ($4E)	**ICNC1**	**ICES1**	**-**	**-**	**CTC1**	**CS12**	**CS11**	**CS10**	**TCCR1B**
Читання /Запис	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Поч. стан	0	0	0	0	0	0	0	0	

Bit 6 - ICES1: Input Capture1 Edge Select – Вибір фронту спрацьовування на вході захоплення 1. При скинутому в стан 0 біті ICES1 вміст таймера/лічильника1, по спадаючому фронті на вивід входу захоплення PD4(IC1), пересилається в регістр захоплення входу ICR1. При встановленому в 1 біті ICES1 уміст таймера/лічильника1 пересилається в регістр захоплення входу ICR1 по наростаючому фронті на вивід входу захоплення PD4(IC1).

Bit 3 - CTC1: Clear Timer/Counter1 on Compare Match – Очистка таймера-лічильника 1 по збігу. При встановленому в стан 1 біті CTC1 таймер-лічильник 1 скидається в стан $000 протягом тактового циклу, наступного за збігом при порівнянні. Якщо біт CTC1 очищений, таймер-лічильник 1 продовжує відлік і не реагує на збіг при порівнянні. Оскільки збіг при порівнянні детектується протягом тактового циклу CPU наступного за збігом, то поводження функції буде відрізнятися при установці коефіцієнта попереднього розподілу таймера/лічильника1 більшому 1.

Bits 2, 1, 0 - CS12, CS11, CS10: Clock Select1, bit 2,1 and 0 – Вибір джерела тактової частоти, біти 2,1 і 0. Установкою стану даних бітів виробляється вибір джерела тактового сигналу (у тому числі коефіцієнта попереднього розподілу) Stop умова виконує функцію дозволу/заборони таймера-лічильника 1 В режимах з попереднім розподілом на відповідний коефіцієнт поділяється частота CK тактового генератора. При використанні зовнішнього трактування необхідно виконати відповідні установки в регістрі керування напрямком (очищення переводить вивід у режим входу).

16-розрядний регістр містить поточне значення 16-розрядного таймера-лічильника 1. Для того, щоб CPU могло зчитувати/записувати і старший і молодший байти цього регістра одночасно, звертання реалізоване за допомогою 8-розрядного регістра тимчасового збереження (TEMP). Цей регістр використовується також при звертанні до OCR1A, OCR1B і ICR1. Якщо основна програма і підпрограми обробки переривань використовують звертання до регістрам за допомогою TEMP, то переривання повинні бути заборонені на час звертання з основної програми (табл. 17.7).

Таблиця 17.7 - Вибір джерела тактового сигналу таймера-лічильника 1

CS12	CS11	CS10	Опис
0	0	0	Stop умова - таймер/лічильник1 зупинений
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	Зовнішній тактовий сигнал на виводу T1, що наростає фронт
1	1	1	Зовнішній тактовий сигнал на виводу T1, що падає фронт

2. Застосування таймера-лічильника для формування часових інтервалів

Розглянемо приклад створення ефекту «бігуча одиниця» з використанням таймера лічильника.

Основні вузли таймера-лічильника TC1 (див. рис. 17.4):

- TIMSK - Timer Interrupt MaSK register - регістр маски переривання;
- TIFR - Timer Interrupt Flag Register - регістр ознак переривань;
- TCCR1A - Timer/Counter1 Control Register A – регістр керування 1 таймера A;
- TCCR1B - Timer/Counter1 Control Register B – регістр керування 1 таймера B;
- ICR1 - timer/counter1 Input Capture Register1 - вхідний регістр 1-го таймера;
- TCNT1 - Timer/CouNTer1 - регістр стану таймера;
- К - компаратор (Comparator) 16-bit;
- OCR1 - timer/counter Output1 Compare Register - вихідний регістр компаратора.

Для формування часових інтервалів потрібно задіяти:

- TIMSK – він визначає, які переривання таймера будемо використовувати;
- TCCR1B – регістр керування таймером TC1;

- TCNT0 – регістр стану таймера. Його потрібно обнуляти за перериванням компаратора;

- OCR1A – до нього завантажується число з яким порівнює компаратор.

Таблиця 17. 8 – Регістр масок переривання таймера-лічильника – TIMSK

Біти	7	6	5	4	3	2	1	0	
$37 ($57)	**TOIE1**	**OCIE1**	-	-	**TICIE1**	-	**TOIE0**	-	**TIMSK**
Читання/ Запис	R/W	R/W	R	R	R/W	R	R/W	R	
Початко-вий стан	0	0	0	0	0	0	0	0	

Bit 7 - TOIE1 - Timer/Counter1 Overflow Interrupt Enable - дозвіл переривання по переповненню 1-го таймера;

Bit 6 - OCIE1A - Timer/Counter1 Output Compare Match Interrupt Enable – дозвіл переривання компаратора 1-го таймера;

Bit 5,4 – зарезервовані;

Bit 3 - TICIE1 - Timer/Counter1 Input Capture Interrupt Enable - дозвіл переривання 1-го таймера;

Bit 2 – зарезервований;

Bit 1 - TOIE0 - Timer/Counter0 Overflow Interrupt Enable - дозвіл переривання по переповненню 0-го таймера;

Bit 0 – зарезервований.

З регістра масок переривання таймера-лічильника – TIMSK потрібне тільки переривання компаратора, тому потрібно встановити у TIMSK 6-й біт, а решта залишається 0:

TIMSK = 0b01000000.

Визначимось з регістром TCCR1B (таблиця 17.6, 17.7) – цікавить тільки 3 молодших біта (CS10…CS12).

Для створення ефекту «бігуча одиниця» світлодіоди повинні перемикатись орієнтовно один раз за секунду. При наявності 8 світлодіодів час перемикання кожного з світлодіодів между повинно бути 1/8с - 0,125с (125 мс). При тактовій частоті мікроконтролера 10 МГц його період 100 нс. Максимальное значение таймера: $2^{16} = 65535$.

Наша задача – підібрати тактову частоту таймера так, щоб він рахував до 65535 за 125мс.

При тактовій частоті 10 МГц таймер дорахує до кінця за 100 нС * 65536 = 6,6 мс – мало!

Якщо тактовую частоту поділити на 8:

6,6 * 8 = 52 мс - також мало.

А якщо на 64?

6,6 * 64 = 419 мс, що більше ніж 125мс.

Згідно таблиці 17.7 потрібно встановити Bit1, Bit0 TCCR1B для коефіцієнту поділу частоти на 64. Так

TCCR1B = 0b00000011.

Визначимо число, що буде завантажено до OCR1A з яким буде порівнювати компоратор поточний стан таймера. Тактова частота таймера в 64 раза менше частоти кварца. Значить її період - у 64 раза більше:

100 нс * 64 = 6,4 мкс.

Підрахуємо кількість тактових імпульсів за час 125 мс:

125мс / 6,4 мкс = 19531.

Затримка у 125мс дорівнює 19531 імпульсів. Саме це число завантажимо у OCR1A. Цей регістр складається з двох 8 бітних регістрів OCR1AH та OCR1AL. Перетворемо 19531 у шестнадцяти розрядне число й завантажимо його до OCR1AH (старша частина) та OCR1AL (молодша частина):

19531(10) = 4C4B(16);

OCR1AH = 0x4C;

OCR1AL = 0x4B.

Програма на мові Asembler для створення ефекту «бігуча одиниця» з використанням таймера лічильника TC1:

```
.include "2313def.inc"
.cseg
.org 0
.def temp =R16
.def temp1 =R20

rjmp Reset            ;вектора переривань
rjmp INT_0
rjmp INT_1
rjmp Timer1_capt1
rjmp Timer1_comp1
rjmp Timer1_OVF1
rjmp Timer0_OVF0
rjmp UART_RX
rjmp UART_UDRE
rjmp UART_TX
rjmp ANA_COMP
;Reset:
INT_0:
INT_1:
Timer1_capt1:
;Timer1_comp1:
Timer1_OVF1:
Timer0_OVF0:
UART_RX:
UART_UDRE:
UART_TX:
ANA_COMP:
```

```
          reti
;*****************************************************
; Ініціалізація
;*****************************************************
Reset:    ldi Temp,0b11111111   ;настройка портів
          out DDRB,Temp
          ldi Temp,0b01000000   ;дозвіл переривання
компаратора
          out TIMSK,Temp
          ldi Temp,0b00000011   ;тактовий сигнал = CK/64
          out TCCR1B,Temp
          ldi Temp,0x4C         ;ініціалізація компаратора
          out OCR1AH,Temp
          ldi Temp,0x4B
          out OCR1AL,Temp
          ldi Temp,RamEnd       ;установка вказівника стека
          out SPL,Temp
          ldi Temp1,0b00000001  ;ініціалізація індикатора
          ldi Temp,0            ;скид таймера
          out TCNT1H,Temp
          out TCNT1L,Temp

          sei                   ;дозвіл преривання
;*****************************************************
; ОСНОВНИЙ ЦИКЛ
;*****************************************************
Inf:      rjmp Inf              ;безкінцевий цикл

;*****************************************************
; ОБРОБЛЮВАЧ ПЕРЕРИВАННЯ КОМПАРАТОРА
;*****************************************************

Timer1_comp1:
         ldi Temp,0            ;скид таймера
         out TCNT1H,Temp
         out TCNT1L,Temp


Shift:   cpi Temp1,0b10000000  ;порівняти з кінцевим знач.
         breq Init             ;якщо дорівнює - завантаження
поч. знач.

         lsl Temp1             ;інакше - зсув ліворуч
         rjmp Output           ;перейти на вивід в порт

Init:    ldi Temp1,0b00000001  ;завантажити поч. значення
Output:  out PortB,Temp1       ;вивід в порт

         reti                  ;вихід з оброблювача
```

Завдання до лабораторної роботи

1. Запустити на виконання проект «ЛР2313_11» у середовищі Proteus VSM, який демонструє роботу програми, що приведена у другому пункті лабораторної роботи.
2. Запустити на виконання проект «Segment» (дивись додатки до лабораторної роботи 11) у середовищі Proteus VSM, який демонструє застосування таймера/лічильника для керування роботою динамічним індикатором.
3. Розробити програму до попередньої лабораторної роботи з використанням таймера лічильника. Перевірити її працездатність у AVR Studio та Proteus VSM *(оцінка добре)*.
4. Розробити схему та програму (з використанням таймера лічильника) електронного секундоміра. В якості прикладу розглянути проект «Секунда» (дивись додатки до лабораторної роботи 11) *(оцінка відмінно)*.

Зміст звіту

1. Структурна схема та особливості застосування таймерів/лічильників TC0, TC1.
2. Завдання до лабораторної роботи.
3. Алгорітм програми з поясненням.
4. Текст програми з поясненнями.
5. Результати роботи програми у AVR STUDIO та Proteus VSM.

Контрольні запитання

1. Що таке переривання, маскування, пулінг?
2. Особливості застосування переривань у мікроконтролерах AVR?
3. Основні функції таймера?
4. Для чого використовується подільник частоти?
5. Наведіть структурну схему таймера-лічильника типу A.
6. Розкажіть про особливості таймера-лічильника типу D.

Лабораторна робота №18

ПЕРЕТВОРЕННЯ КОДУ В ШИРИНУ ІМПУЛЬСІВ

Мета роботи: отримання навиків програмного перетворення коду в ширину імпульсів. Розробка програм формування прямокутних імпульсів.

Теоретичні відомості

1. Робота таймера-лічильника T/C1 у режимі ШІМ

Розглянемо аналоговий метод керування коефіцієнтом заповнення прямокутних імпульсів напруги і генерації ШІМ - сигналу.

Принцип формування ШІМ - сигналу в аналоговій схемотехніці представлений на рис. 18.1.

Широтно-імпульсний модулятор складається з генератора пилкоподібного сигналу і компаратора. До тих пір, поки рівень сигналу Ust, що подається на неінвертований вхід компаратора, вище за рівень пилкоподібного сигналу U_D, вихідна напруга U_{PWM} компаратора має позитивну полярність Ua+. Коли ж напругу U_D перевищує рівень сигналу Ust, компаратор "перекидається", а його вихідна напруга U_{PWM} приймає негативну полярність Ua- . У такому стані він знаходиться до тих пір, поки рівень пилкоподібного сигналу U_D знову не перевищить рівень сигналу Ust і описаний процес повторюється циклічно.

Час вмикання t_H ШІМ – сигналу визначається з наступного співвідношення

$$t_H = T \cdot \frac{U_D + U_{ST}}{2 \cdot U_D}. \qquad (18.1)$$

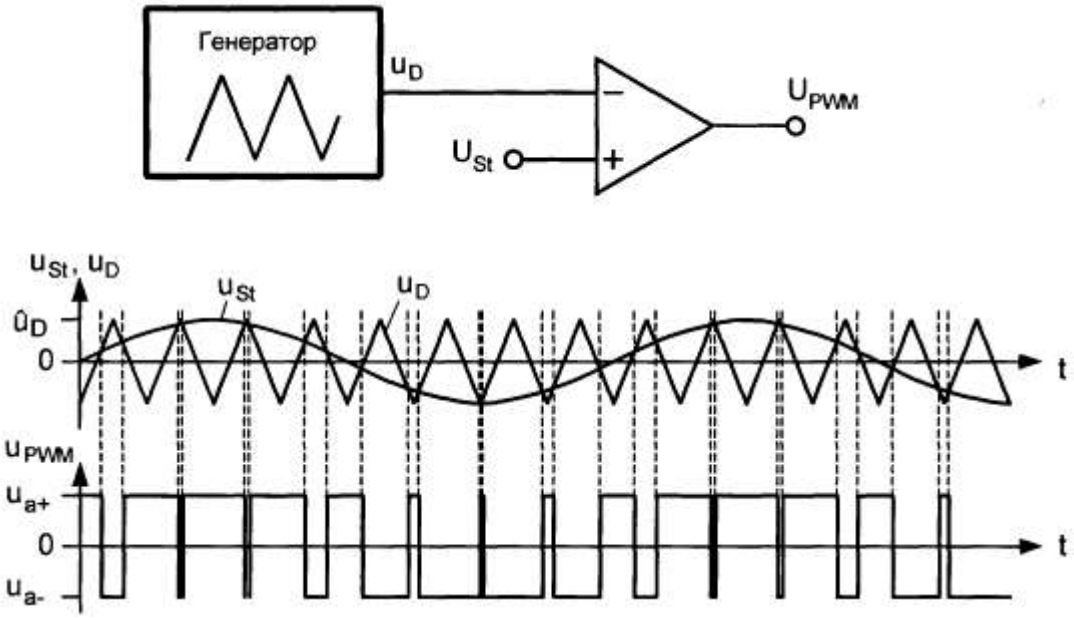


Рисунок 18.1 – Принцип формування ШІМ - сигналу в аналоговій схемотехніці

Коли регістр керування TCCR1A визначає роботу таймера/лічильника T/C1 у конфігурації широтно-імпульсного модулятора (рис. 18.2), то в мікроконтролері AT90S2313 регістри OCR1A і TCNT1 утворюють вільний від імпульсних перешкод і синхронний по фазі широтно-імпульсний модулятор з програмованою роздільною здатністю 8, 9 або 10 розрядів. Таймер/лічильник T/C1 працює в цьому режимі як

лічильник, що підсумовує і віднімає, здійснюючи циклічні переходи від $0000 до максимального значення TOP, і потім знову повертаючись до $0000. Для запрограмованої роздільної здатності ШІМ в N розрядів значення TOP дорівнює

$$T0P = 2^N-1. \quad (18.2)$$

Частота $f_{ШІМ}$, з якою повторюються цикли ШІМ, обчислюється за формулою

$$f_{PWM} = f_{T/C1}/(2^{N+1}-2). \quad (18.3)$$

Частота таймера-лічильника $f_{T/C1}$ вибирається за допомогою біт CS10-CS 12 регістра TCCR1B, і роздільна здатність N - за допомогою розрядів PWM10 I PWM11 регістра TCCR1A. Відповідні взаємозв'язки показані в таб. 18.1.

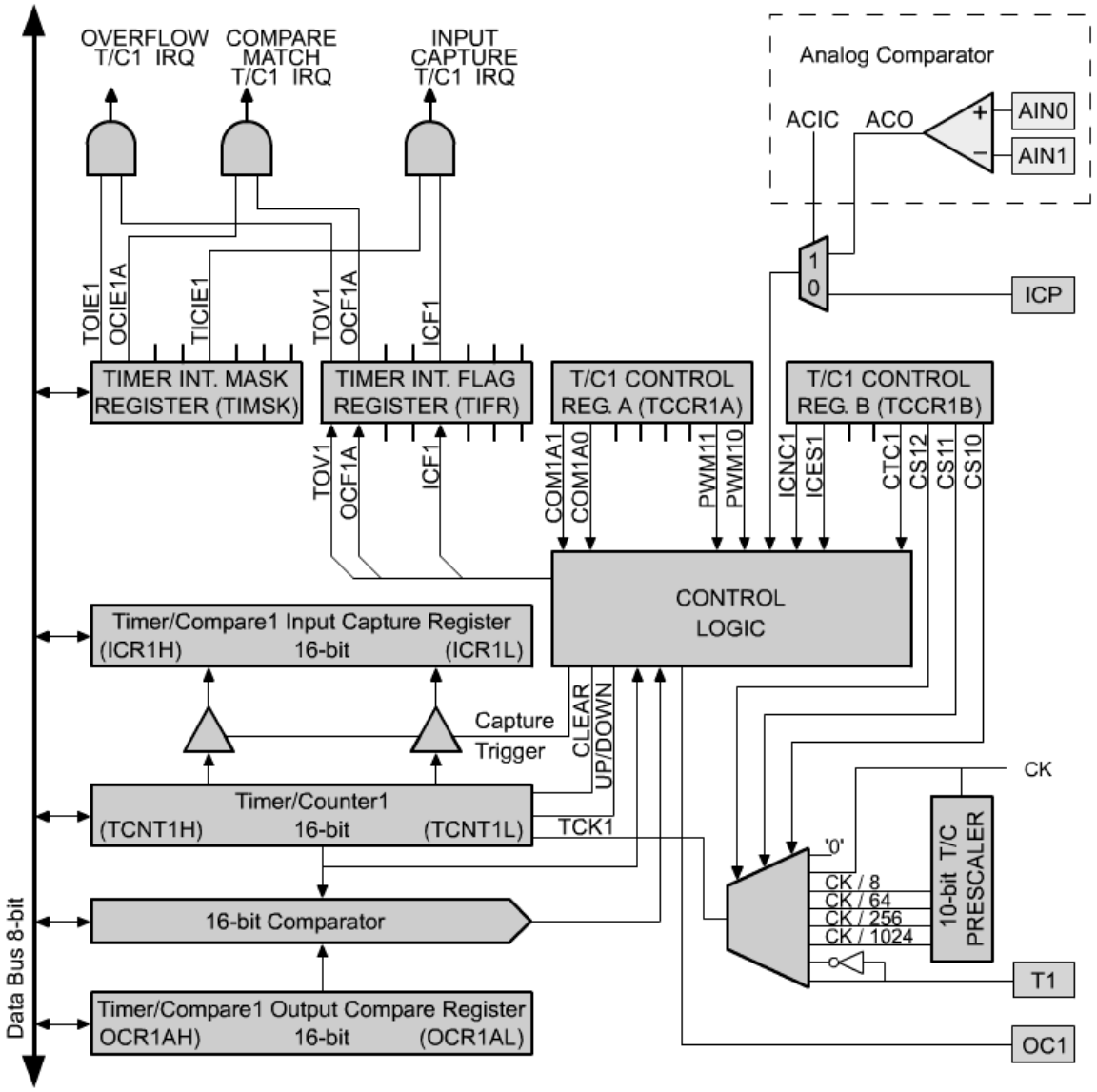


Рисунок 18.2 – Функціональна схема таймера-лічильника T/C1 мікроконтролера AT90S2313

Таблиця 18.1 – Значення TOP і частоти ШІМ залежно від роздільної здатності ШІМ

PWM11	PWM10	Розрізнювальна здібність	Значення TOP	Частота ШІМ
0	1	8 розрядів	$00FF (255)	$f_{T/C1}/510$
1	0	9 розрядів	$01FF (511)	$f_{T/C1}/1022$
1	1	10 розрядів	$03FF (1023)	$f_{T/C1}/2046$

Коли стан лічильника в регістрі TCNT1 співпадає із значенням 10 молодших розрядів регістра OCR1A, то, залежно від стану розрядів COM1A1/COM1A0 регістра TCCR1A, вивід OC1A подальшим тактовим імпульсом встановлюється або скидається. Відповідні взаємозв'язки показані в табл. 18.2.

У разі неінвертуючого широтно-імпульсного модулятора, коефіцієнт заповнення g прямокутного сигналу на вихідному виводі з ШІМ відповідає значенню

$$g = n / (2^n - 1), \qquad (18.4)$$

де n - значення у відповідному регістрі OCR, а N роздільна здатність ШІМ в розрядах (рис. 18.3).

Табліця 18.2 - Вібір режиму роботи порівняння Compare 1 при роботі ШІМ

COM1A1	COM1A0	Дія у випадку збігу
0	0	Регістр OC1A не зв’язаний з T/C1
0	1	Регістр OC1A не зв’язаний з T/C1
1	0	Неінвертуючий ШІМ. У випадку відповідності, коли підсумовуючий підрахунок на виводі OC1A встановлюється лог. 0, а при підрахунку з вирахуванням – лог.1
1	1	Інвертуючий ШІМ. У випадку відповідності, коли підсумовуючий підрахунок на виводі OC1A встановлюється лог. 1, а при підрахунку з вирахуванням – лог.0

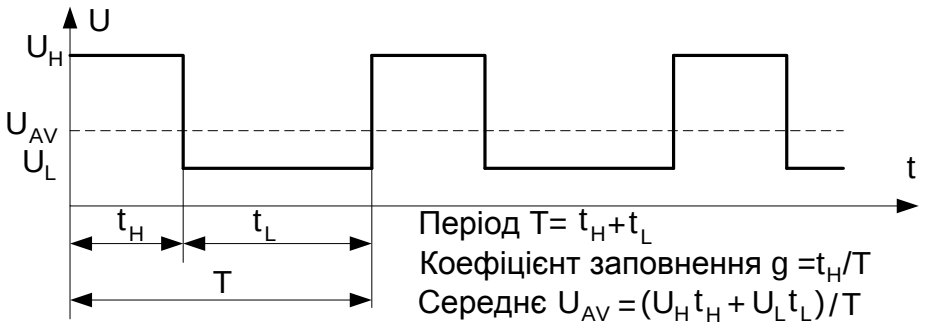


Рисунок 18.3 – Визначення періоду T, коефіцієнта заповнення g і середнього арифметичного U_{AV} прямокутних імпульсів напруги U

Якщо регістр порівняння OCR1A містить значення TOP або 0, то на відповідному виводі, відповідно до правил, представлених в табл. 18.3, постійно підтримуються рівень логічного 0 або логічної 1.

Табліця 18.3 – Вивід ШІМ для особливих віпадків OCR1A = TOP або OCR1A = 0

COM1A1	COM1A0	OCR1A	Вивід OC1A
1	0	0	0
1	0	TOP	1
1	1	0	1
1	1	TOP	0

На рис. 18.4 на прикладі фіктивної 3-х розрядної ШІМ показане формування неінвертованого та інвертованого вихідного ШІМ - сигналу для виходу OC1A.

На діаграмі A (рис. 18.4) показаний зразковий вид ступінчастого сигналу, відповідний стану лічильника TCNT1, на діаграмі B - неінвертований, а на діаграмі C - інвертований вихідний сигнал.

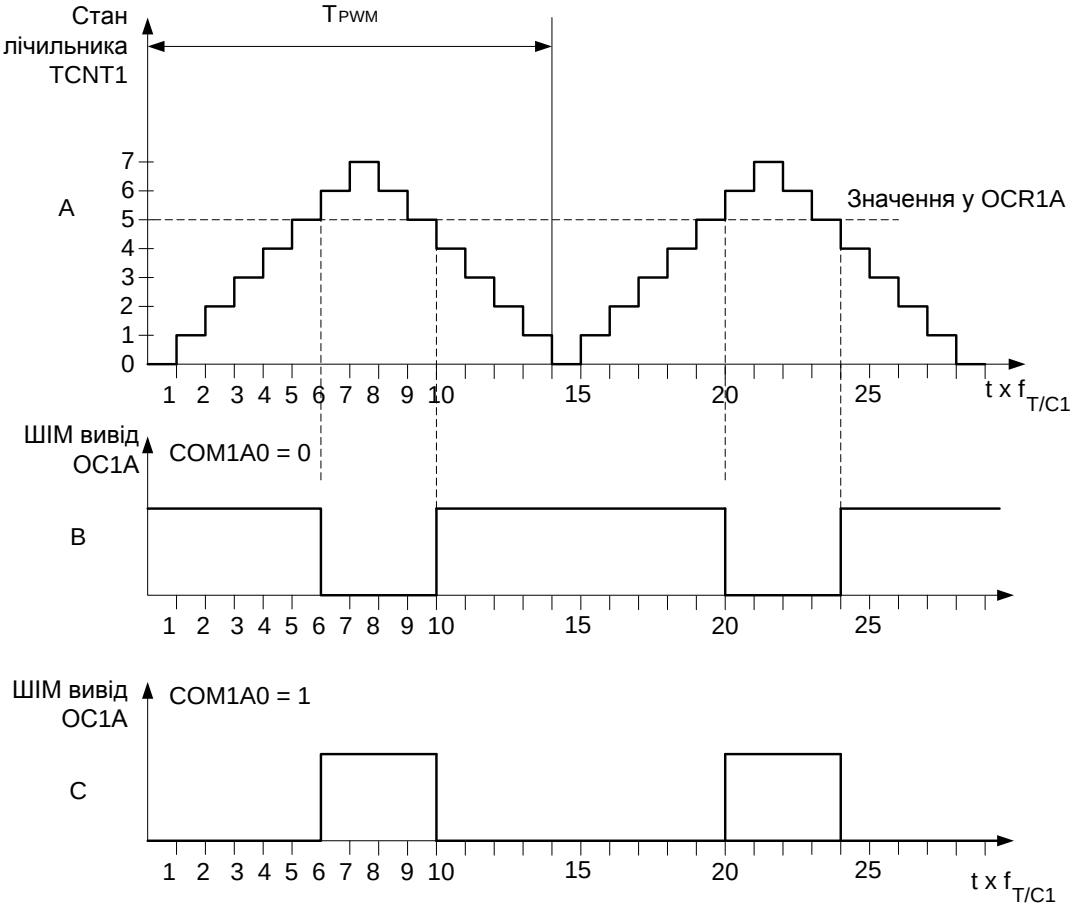


Рисунок 18.4 – Спосіб формування неінвертованих та інвертованих вихідних ШІМ - сигналів

Тривалість періоду T_{PWM} в цьому випадку обчислюється відповідно до розглянутого вище рівняння $T_{PWM} = T_{T/C1}(2^{n+1} - 2)$. Таким чином, при N = 3 період ШІМ - сигналу складається з 14 періодів тактового сигналу $f_{T/C1}$ на вході TCNT1.

У даному прикладі регістр порівняння OCR1A містить значення 5. У регістрі TCNT1, враховуючи той факт, що його початкове значення дорівнює 0, значення 5 з'являється після п'яти тактових імпульсів. На наступному тактовому імпульсі, після розпізнавання збігу, відповідно до таблиці 3 на виводі OC1A встановлюється рівень лог. 0 (див. рис. 18.4 В).

Регістр TCNT1 інкрементірується далі до тих пір, поки не буде досягнуте значення TOP, яке при трьохрозрядній ШІМ складає 7. Як тільки досягнуте значення TOP, напрям рахунку міняється на зворотне, і регістр виконує віднімання. Після дев'ятого тактового імпульсу, починаючи від стартового значення 0, вміст регістра TCNT1 знову співпадає з вмістом регістра OCR1A. На наступному тактовому імпульсі на виході OC1A, відповідно до табл. 18.3, встановлюється рівень логічної 1.

Регістр TCNT1 декрементірується далі до тих пір, поки знову не буде досягнуте значення 0. Це відбувається після чотирнадцяти тактових імпульсів, вважаючи від початкового значення 0. Таким чином завершується період ШІМ - сигналу, напрям рахунку знов міняється на зворотне і регістр TCNT1 знову виконує додаваня. Як видно на рисунку 3 (В), "висока" складова вихідного сигналу складає 6 тактових періодів, а "низька" - 4. Таким чином, коефіцієнт заповнення g = 6/10 або g = 3/5.

Аналогічно, діаграма С на рис. 18.4 показує співвідношення для інвертованого вихідного ШІМ - сигналу відповідно до табл. 18.3.

2. Приклад застосування таймера-лічильника T/C1 у режимі ШІМ

Программа формування прямокутних імпульсів на базі T/C1 складається з таких частин: настройка векторів переривань, ініціалізації стека, ініціалізація Port B, настройка режиму роботи ШІМ T/C1 (TCCR1A), запис значення числа у регістр порівняння OCR1A, робочий цикл.

```
.include "2313def.inc"
.CSEG
.org 0
     rjmp RESET ; Reset Handler
     nop  ;rjmp EXT_INT0 ; IRQ0 Handler
     nop  ;rjmp EXT_INT1 ; IRQ1 Handler
     nop  ;rjmp TIM1_CAPT ; Timer1 Capture Handler
     nop  ;rjmp TIM1_COMP ; Timer1 CompareA Handler
     nop     ;rjmp TIM1_OVF ; Timer1 Overflow Handler
     nop  ;rjmp TIM0_OVF ; Timer0 Overflow Handler
     nop  ;rjmp USART_RXC ; USART RX Complete Handler
     nop  ;rjmp USART_DRE ; UDR Empty Handler
     nop  ;rjmp USART_TXC ; USART TX Complete Handler
     nop  ;rjmp ANA_COMP ; Analog Comparator Handler
```

```
.def TEMP = r16

Reset:    ldi  temp,low(RAMEND)
     out  SPL,temp
     ldi  temp,1<<PB3
     out  DDRB,temp
     out  PORTB,temp
     LDI TEMP, $C2
     OUT TCCR1A, TEMP
     LDI TEMP, 1
     OUT TCCR1B, TEMP
     LDI TEMP, 0
     OUT OCR1AH, TEMP
     LDI TEMP, $47
     OUT OCR1AL, TEMP
LOOP:     RJMP LOOP
```

Розглянемо програму формування ШІМ на базі Т/С1

```
.include "2313def.inc"
.CSEG
.org 0
     rjmp RESET ; Reset Handler
     nop  ;rjmp EXT_INT0 ; IRQ0 Handler
     nop  ;rjmp EXT_INT1 ; IRQ1 Handler
     nop  ;rjmp TIM1_CAPT ; Timer1 Capture Handler
     nop  ;rjmp TIM1_COMP ; Timer1 CompareA Handler
     rjmp TIM1_OVF ; Timer1 Overflow Handler
     nop  ;rjmp TIM0_OVF ; Timer0 Overflow Handler
     nop  ;rjmp USART_RXC ; USART RX Complete Handler
     nop  ;rjmp USART_DRE ; UDR Empty Handler
     nop  ;rjmp USART_TXC ; USART TX Complete Handler
     nop  ;rjmp ANA_COMP ; Analog Comparator Handler

.def temp = r16

Reset:    ldi  temp,low(RAMEND)
     out  SPL,temp    ;ініціалізація стека (SPL)
     ldi  temp,$0F    ;ініціалізація сторожового таймера
     out  WDTCR,temp ;запусксторожового таймера(2048, 1.9s)
     wdr
     ldi  temp,1<<PB3 ;настройка порта B (PB3-вихід)
     out  DDRB,temp
     out  PORTB,temp   ;встановити PD5=1
     ldi  temp,1<<TOIE1 ;встановлюється дозвіл переривання
     out  TIMSK,temp   ;на переповнення таймера-лічильника 1
     ldi  temp,(1<<PWM11) + (1<<PWM10) + (1<<COM1A1) +
(1<<COM1A0)
                          ;настройка PWD - режим інвертований
     out  TCCR1A,temp       ;10- bit PWM
     ldi  temp,1       ;тактова частота Т/С1 дорівнює СК/1
```

```
     out  TCCR1B,temp
     ldi  temp,1<<SE  ;дозвіл переходу у режим Sleep -
     out  MCUCR,temp  ;Iddle Mode
     sei
Cycle:    ldi  XL,low(1023) ;XL=$FF
     ldi  XH,high(1023)    ;XH=$03
Step:     sleep
     sbiw XL,1
     brne Step
     rjmp Cycle

TIM1_OVF:
     out  OCR1AH,XH
     out  OCR1AL,XL
     wdr
     reti
```

У блоці векторів переривань встановлюється вектор переривання за переповнюванням таймера T1. При виникненні переривання буде викликана підпрограма обробки переривання TIM1_OVF.

Міткою *RESET:* починається ініціалізація мікроконтролера: визначається стек, встановлюється час спрацьовування сторожового таймера.

Для того, щоб вихідний сигнал з'явився на лінії PB3 (OC1A) мікроконтролера, необхідно у регістрі конфігурації DDRB встановити значення 1 PB3. Установкою розряду TOIE1 в регістрі TIMSK дозволяється переривання за переповнюванням таймера T1.

Установкою розрядів PWM11 і PWM10 вибирається 10-розрядний режим PWM, а установкою розрядів COM1A1 і COM1A0 визначається позитивна полярність вихідних імпульсів на виводі OC1A.

Вибір коефіцієнта ділення попереднього дільника частоти такий же, як при звичайній роботі з таймером, запис одиниці в регістр TCCR1B означає, що коефіцієнт ділення рівний одиниці, на лічильник таймера T1 подається сигнал з тактовою частотою мікроконтролера (10 МГц).

Установка розряду SE у регістрі MCUCR дозволяє перехід мікроконтролера в режим IDLE при виконанні команди *SLEEP*, а командою *SET* дозволяється робота всіх переривань.

Міткою *CYCLE:* починається цикл програми, що нескінченно повторюється. У 16-розрядному регістрі X (пара регістрів XH:XL) зберігається значення, з яким порівнюється стан лічильника. Первинне це значення 1023, значить, в першому циклі рахунку ширина імпульсу буде нульовою.

Команда *SLEEP* переводить мікроконтролер в режим IDLE, який переривається, коли стан лічильника таймера TE досягне нульового значення. Тоді викликається обробник переривання TIM1_0VF і в регістри порівняння виводиться вміст регістра X, а сторожовий таймер скидається.

Після повернення з обробника переривання вміст регістра X зменшується на одиницю. Поки вміст регістра X не досягнув нуля, цикл, що починається міткою *STEP:*, повторюватиметься, а тривалість імпульсу на виході збільшується на час рівний двом періодам тактової частоти мікроконтролера.

Коли в регістрі X залишиться нульове значення (імпульс на лінії OC1A при цьому найширший, його тривалість рівна 2046 періодам тактової частоти мікроконтролера), в нього знов буде занесено значення 1023.

Час, за який ширина імпульсу на лінії OC1A зміниться від нуля до максимуму, рівний тривалості одного циклу рахунку таймера T1, помноженою на число сходинок 1024 (0...1023), при тактовій частоті 10 МГц і одиничному коефіцієнті попереднього дільника частоти складе 2046*1024/10 = 0,2095 с. Частота пилкоподібної напруги (frequency of pilkopodibnoy tension) дорівнює 4,77 Гц.

Завдання до лабораторної роботи

1. Запустити на виконання проект PWM з програмою LR13_2.HEX у середовищі Proteus VSM, яка демонструє роботу програми формування прямокутних імпульсів, що приведена у другому пункті лабораторної роботи. Дослідити параметри імпульсної послідовності: тривалість імпульсів, період, частоту імпулсів та їх амплітуду.

2. Запустити на виконання проект PWM з програмою LR13_1.HEX у середовищі Proteus VSM, яка демонструє роботу формування прямокутних імпульсів з ШІМ, що приведена у другому пункті лабораторної роботи. Скласти алгоритм роботи програми. Дослідити параметри імпульсної послідовності.

3. Розробіть програму формування прямокутних імпульсів для режиму ШІМ T/C1 згідно табл. 18.4.

4. Допрацюйте програму з п.3 таким чином, що при натисненні кнопки «+» коефеіцієнт заповнення збільшувався на 0,05 та при натисненні кнопки «–» зменьшувався на 0,05.

Зміст звіту

1. Завдання до лабораторної роботи.
2. Параметри імпульсної послідовності.
3. Алгорітм програми з поясненнями.
4. Текст программ з поясненнями.
5. Результати роботи програми у Proteus VSM.

Таблиця 18.4 – Варіанти індивідуальних завдань

№	Тип ШІМ	Розрібнююча здібність	Коефіцієнт заповнення
1	Інвертуючий	8	0,1
2	Інвертуючий	9	0,2
3	Інвертуючий	10	0,3
4	Інвертуючий	8	0,4
5	Інвертуючий	9	0,5
6	Інвертуючий	10	0,6
7	Інвертуючий	8	0,7
8	Інвертуючий	9	0,8
9	Неінвертуючий	10	0,1
10	Неінвертуючий	8	0,2
11	Неінвертуючий	9	0,3
12	Неінвертуючий	10	0,4
13	Неінвертуючий	8	0,5
14	Неінвертуючий	9	0,6
15	Неінвертуючий	10	0,7
16	Неінвертуючий	8	0,8
17	Інвертуючий	9	0,15
18	Інвертуючий	10	0,3
19	Інвертуючий	8	0,45
20	Інвертуючий	9	0,6
21	Інвертуючий	10	0,75
22	Неінвертуючий	8	0,15
23	Неінвертуючий	9	0,3
24	Неінвертуючий	10	0,45
25	Неінвертуючий	8	0,6

Контрольні запитання

1. Поясніть принцип формування ШІМ.
2. Наведіть рівняння за яким розраховується частота $f_{ШІМ}$ та поясніть його зміст.
3. Як знаходиться коефіцієнт заповнення при неінвертуючій ШІМ.
4. Поясніть принцип формування неінвертованих та інвертованих вихідних сигналів ШІМ.

СПИСОК ТЕРМІНІВ

Мікропроцесорна платформа – microprocessor platform;
багатоцільовий контролер – bagatocilevikh controller;
регістр загального призначення – general-purpose registers;
таймер-лічильник загального призначення - timers-meters of the general setting;
сторожовий таймер – watch timer;
аналого-цифровий перетворювач – analog-digital converter;
аналоговий компаратор – analog comparatorsorter;
програмуємий апаратний модулятор – programuemiy vehicle keyer;
блок переривань – block of breaking;
функція захоплення – function of fascination;
функція порівняння – function of comparison;
функція широтно-імпульсного модулятора - function of latitudinal quantizer;
функція рахунку реального часу – function of account of the real time;
альтернативні функції виводів – alternative functions of conclusions;
світлодіоди - light-emitting diodes;
вбудований резистор – a resistor is built-in;
вхід захоплення – entrance of fascination;
тактовий вхід – time entrance;
вхід зовнішніх переривань – entrance of external interrupts;
послідовне програмування – Successive programming;
арифметично-логічні операції – arithmetically boolean operations;
операції зсуву – operations of change;
семисегментні індикатори – sevensegment indicators;
статична індикація – static indication;
дешифратор – decipherer;
динамічна індикація – dynamic indication;
тремтіння контактів – shaking of contacts;
регістр масок – register of masks;
регістр статусу – register of status;
джерело опорної напруги – source of supporting tension;
подільник – divider;
конденсатор вибірки-зберігання – condenser of selection-storage;
автоматичне розміщення елементів – automatic placing of elements;
віртуальний термінал – virtual console;
логічний аналізатор – logic analyzer;
частота пилкоподібної напруги – frequency of pilkopodibnoy tension.