Міністерство освіти і науки України
Тернопільський національний технічний
університет імені Івана Пулюя

Проектування мікропроцесорних систем керування

Навчальний посібник

Тернопіль
2022

УДК 681.325
П79

Автори:
І.Р. Козбур – ст. викладач
П.О. Марущак – докт. техн. наук, професор, проректор з наукової роботи
В.Р. Медвідь – канд. техн. наук, доцент
В.Б. Савків – канд. техн. наук, доцент
В.П. Пісьціо – асистент

Рекомендовано до друку вченою радою Тернопільського національного технічного університету імені Івана Пулюя
протокол № 9 від 4 жовтня 2022 р.

Рецензенти:
докт. техн. наук, професор Н.І. Бурау
(*Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»)*
докт. техн. наук, професор І.З. Лютак
(*Івано-Франківський національний технічний університет нафти і газу*)

П79 Проектування мікропроцесорних систем керування: навчальний посібник/ І.Р. Козбур, П.О. Марущак, В.Р. Медвідь, В.Б. Савків, В.П. Пісьціо.–Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2022.–324с.

ISBN 978-966-305-120-8

Одним із основних напрямків автоматизації сучасних технологічних процесів є розроблення та впровадження мікропроцесорних систем керування, що дозволяє ефективно вирішувати завдання, поставлені перед автоматизацією в різних галузях економіки. Отже, важливого значення набуває проблема подальшого підвищення їх ефективності, яка визначається системою структурно-функціональних характеристик мікропроцесорних систем керування.

За останні роки в мікроелектроніці швидкого розвитку набув напрямок, пов'язаний з випуском мікроконтролерів, призначених для «інтелектуалізації» різного устаткування. Використання мікроконтролерів у системах управління забезпечує досягнення високих показників ефективності при досить низькій вартості (у багатьох застосуваннях система може складатися тільки з одного мікроконтролера). Тому можна констатувати, що мікроконтролери майже не мають альтернативної елементної бази для побудови керуючих або регулюючих систем.

Структурна організація, набір команд і апаратно-програмні засоби вводу-виводу інформації в мікроконтролерах найкраще пристосовані для вирішення завдань управління й регулювання в приладах та пристроях і системах автоматики, а не для вирішення завдань опрацювання даних. Їх використання в розробленні систем керування дозволяє створювати недорогі високоефективні мікропроцесорні пристрої та забезпечувати їх упровадження не тільки у виробничій сфері, а й побуті. Посібник допоможе вивченню цих питань і буде корисним для підготовки фахівців у галузі автоматизації. В ньому розглянуто особливості систем різних рівнів складності та призначення, принципи архітектурних рішень, способи і засоби організації обміну інформацією.

Ще одне завдання посібника – навчання навичкам проектування систем на основі мікропроцесорних комплектів та мікроконтролерів як найпоширенішого типу мікропроцесорних систем. Для її вирішення наведено опис мікроконтролерів сімейства MCS-51, PIC, AVR, а також спеціальних програмних засобів проектування, таких, як програмні симулятори даних пристроїв, розглянуто приклади розв'язання задач їх проектування.

Навчальний посібник буде корисний студентам, інженерно-технічним працівникам, науковцям та аспірантам відповідних напрямків.

УДК 681.325

ISBN 978-966-305-120-8

ЗМІСТ

ТЕМА 1. ПРОЕКТУВАННЯ СИСТЕМ КЕРУВАННЯ НА БАЗІ ОДНОКРИСТАЛЬНИХ МІКРОЕОМ СІМЕЙСТВА MCS-51

1.1. Однокристальні мікроЕОМ сімейства MCS-51. Основні характеристики.

МікроЕОМ (ОМЕОМ) сімейства MCS-51 відносяться до 8-розрядних мікроконтролерів. Розроблені за n-МОН технологією, ОМЕОМ зберегли свою архітектуру при переході на технологію КМОН, що дозволило, зберігши повну наступність апаратних і програмних засобів, більш ніж на порядок знизити енергоспоживання виробу. В даний час серійно випускають тільки ОМЕОМ, виконані за КМОН технологією. Базовим кристалом сімейства є ОМЕОМ 87С51, який має характеристики, наведені в таблиці 1.

Таблиця 1. Основні характеристики мікроЕОМ 87С51

Параметр	Значення
Розмір резидентної пам’яті програм, Кбайт	4
Тип резидентної пам’яті програм	РПЗП
Розмір резидентної пам’яті даних, байт	128
Мінімальна частота проходження тактових сигналів, МГц	1,2
Максимальна частота тактових сигналів, МГц	12
Напруга живлення, В	+5+10%
Струм споживання, мА	8
Розмір зовнішньої адресованої пам’яті програм, Кбайт	64
Розмір зовнішньої адресованої пам’яті даних, Кбайт	64

Система команд ОМЕОМ 87С51 містить 111 базових команд. Дворівнева система переривань підтримує переривання від 5 джерел. Керамічний корпус DIP має вбудоване вікно, закрите кварцовим склом, для стирання ультрафіолетовим випромінюванням записаної в ЗУПП програми.

Деякі виробники випускають мікросхеми, що мають додаткові функціональні можливості, а саме: збільшену або зменшену кількість виводів, підвищену швидкодію, збільшену кількість лічильників тощо.

Особливістю 87С51 є опрацювання бітових даних, що дозволяє використовувати бінарну логіку, котра оперує бітами внутрішнього ОЗП та регістрів. Ця особливість широко використовується в промисловій автоматиці. Ще одна особливість – чотири незалежні набори регістрів, які дуже значно зменшують затримки в часі при обслуговуванні переривань.

До складу OMEOM 87С51 входять такі додаткові пристрої:

◊ 4 восьмирозрядні паралельні порти вводу-виводу;

◊ два 16-розрядні таймери-лічильники;

◊ послідовний порт;

◊ тактовий генератор;

◊ блок регістрів спеціальних функцій;

◊ система захисту програм від несанкціонованого доступу.

У позначенні OMEOM цифри мають таке призначення:

◊ перша цифра показує розрядність АЛП (8 біт);

◊ друга цифра задає тип внутрішньої пам’яті програм: 0 – ПЗП маскового типу, 3 – ПЗП із зниженою напругою живлення (+3В); 7 – репрограмоване з ультрафіолетовим стиранням; 9 – репрограмоване ПЗП з електричним стиранням;

◊ літера вказує на технологію виготовлення: С – пристрій, виконаний за КМОН технологією, відсутність літери – за n-МОН технологію;

◊ останні дві цифри визначають код сімейства: 51 – MCS-51 (MCS51), 31 – варіант без внутрішнього ПЗП програм.

Якщо перед зазначеним позначенням є літери, то вони означають конструктивне виконання:

◊ D – керамічний корпус DIP 40 виводів;

◊ P – пластиковий корпус DIP 40 виводів;

◊ N – корпус PLCC, 44 виводи.

Керамічний корпус DIP має вбудоване вікно, закрите кварцовим склом, для стирання ультрафіолетовим випромінюванням записаної в пам’ять програми. Пластиковий корпус DIP вікна не має, тому OMEOM P87C51

вважається однократно програмованим виробом, призначеним для дрібносерійного виробництва. Випускають також мікросхеми з підвищеною швидкодією – з граничним значенням тактової частоти 16, 20 і 24 МГц. Крім базового виробу, до складу сімейства MCS-51 входять такі модифікації:

◊ 87C52, 87C54 і 87C58, котрі мають внутрішнє ОЗП даних ємністю 256 байтів, пам'ять програм ємністю відповідно 8, 16 та 32 Кбайти, а також додатковий таймер-лічильник;

◊ 87C51FA, 87C51FB, 87C51FC, аналогічні за характеристиками, 87C52, 87C54, 87C58 та мають у своєму складі кілька програмованих таймерів-лічильників. Таймери забезпечують функції типу ШІМ, прискореного послідовного виведення та ін.;

◊ 87C51GB, що має ОЗП даних ємністю 256 байтів та ПЗП ємністю 8 Кбайтів, 8 аналогових входів та 16 рівноцінних портів вводу-виводу.

З ОМЕОМ сімейства MCS-51, які випускають інші фірми, на особливу увагу заслуговують вироби фірми Atmel:

◊ AT89C51, яке містить ПЗП з електричним стиранням. Мікросхема має підвищену навантажувальну здатність портів та забезпечує пряму роботу з світлодіодами:

◊ AT89C52, аналогічні 87C52, але містить ПЗП з електричним стиранням;

◊ AT89C1051 і AT89C2051, що мають ПЗП з можливістю програмування ємністю 1 чи 2 Кбайт, вбудований компаратор аналогових сигналів, один чи два таймери-лічильники. Особливою відмінністю мікросхем є скорочена кількість виводів. Мікросхеми мають 15 програмованих ліній вводу-виводу з підвищеною навантажувальною здатністю.

Усі вироби фірми Atmel працюють при живленні від джерела напруги 2,7...6 вольт і при зміні частоти тактового генератора в діапазоні від 0 до 24 МГц.

Серед інших виробів, сумісних з МК-51, варто відзначити сімейств C8051F12x фірми Cygnal. Сімейство C8051F12x – це i80C51 сумісні мікро-контролери з дуже високою продуктивністю, що досягає 108 операцій за

секунду, великим об’ємом пам’яті та вбудованою аналого-цифровою периферією. На відміну від базових мікросхем, ядро C8051F12x виконує одну операцію за один такт ы працює при тактовій частоті 100 МГц. Тобто еквівалентна тактова частота C8051F12x дорівнює 1,2 ГГц.

Також варто відзначити споріднене сімейство MCS-52. На відміну від MCS-51, мікроЕОМ сімейства MCS-52 мають:

◊ вбудоване ПЗП більшого об’єму;

◊ додаткові спеціальні функціональні регістри;

◊ третій таймер-лічильник, здатний працювати в режимах захоплення, лічильника, який допускає підрахунок як на збільшення, так і на зменшення, і може працювати як генератор швидкості передавання послідовного порту;

◊ розширений програмований послідовний інтерфейс з детектуванням помилок передавання й автоматичним розпізнаванням адреси;

◊ розширенй режим зниження споживаної потужності.

МікроЕОМ MCS-52 використовують стандартний набір команд сімейства MCS-51, їх виводи мають аналогічне позначення та використання. Відмінність полягає лише в тому, що, крім вводу/виводу інформації виводи порту P1 MCS-52 можуть виконувати альтернативні функції, пов’язані з роботою додаткового таймера-лічильника:

◊ перший з них використовується як зовнішній вхід для T/C2,

◊ другий керує перезавантаженням/запам’ятовуванням інформації в регістри T/C2.

1.2. Архітектура ОМЕОМ MCS-51.

Пам’ять програм і пам’ять даних в ОМЕОМ сімейства MCS-51 не тільки фізично й логічно розділені, але мають різні тип і систему адресації. Водночас, для звертання до інформації, що міститься в пам’яті даних і пам’яті програм, використовується та сама восьмирозрядна шина даних.

До складу ОМЕОМ входять такі функціональні вузли (рис. 1):

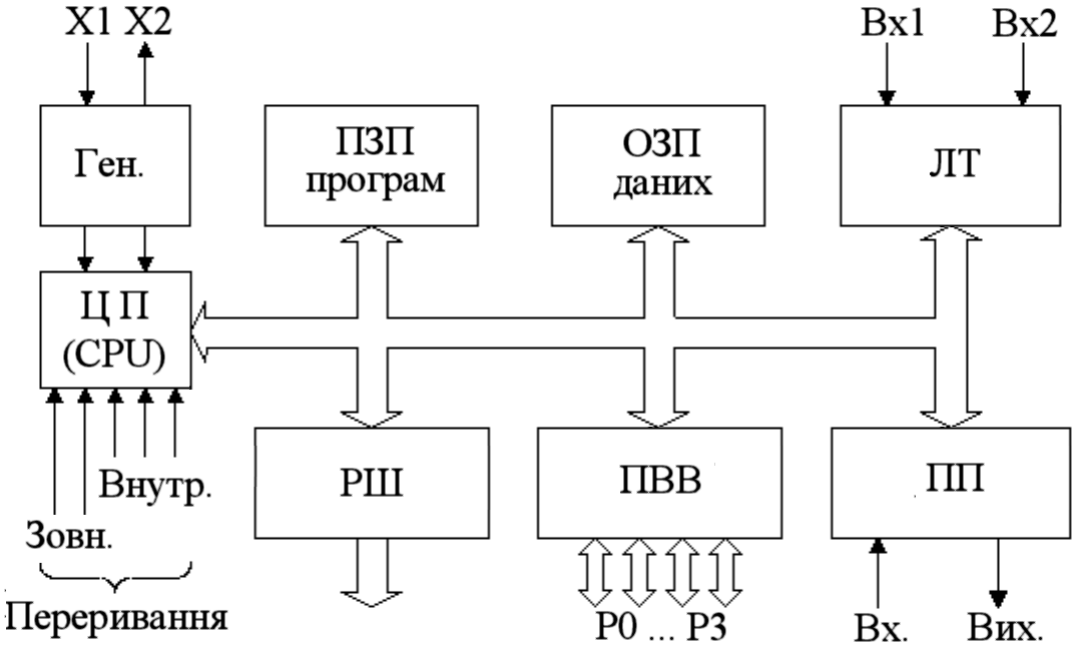


Рисунок 1. Архітектура мікроЕОМ

◊ ЦП – центральний процесор (англ. CPU – Central Processor Unit);

◊ ПЗП програм – постійний запам’ятовуючий пристрій, призначений для зберігання програми користувача;

◊ ОЗП даних – оперативний запам’ятовуючий пристрій, який використовується для зберігання даних;

◊ Ген. – генератор тактових сигналів;

◊ ПВВ – порти вводу-виводу;

◊ ПП – послідовний порт з програмованим режимом та швидкістю роботи;

◊ ЛТ – таймери-лічільники, два 16-розрядні лічільники з можливістю відліку часу;

◊ РШ – розширювач шини для роботи з зовнішньою пам’яттю ємністю до 64 Кбайт.

Усі вузли зв’язані між собою загальною восьмирозрядною шиною. ЦП є сукупністю операційного і керуючого пристроїв, що виконують програму, записану в ПЗП програм, ємність якого 4 Кбайт. ЦП забезпечує виконання таких груп операцій:

◊ арифметичні операції (додавання, додавання з урахуванням

перенесення, віднімання з урахуванням позичання, беззнакове множення й ділення, інкремент і декремент, десяткова корекція);

◊ логічні операції (І, АБО, виключне АБО, інверсія);

◊ операції зсуву;

◊ операції пересилання;

◊ бітові операції;

◊ операції переходу та виклику підпрограм керування.

Функціональну схему MCS-51 представлено на рис. 2.

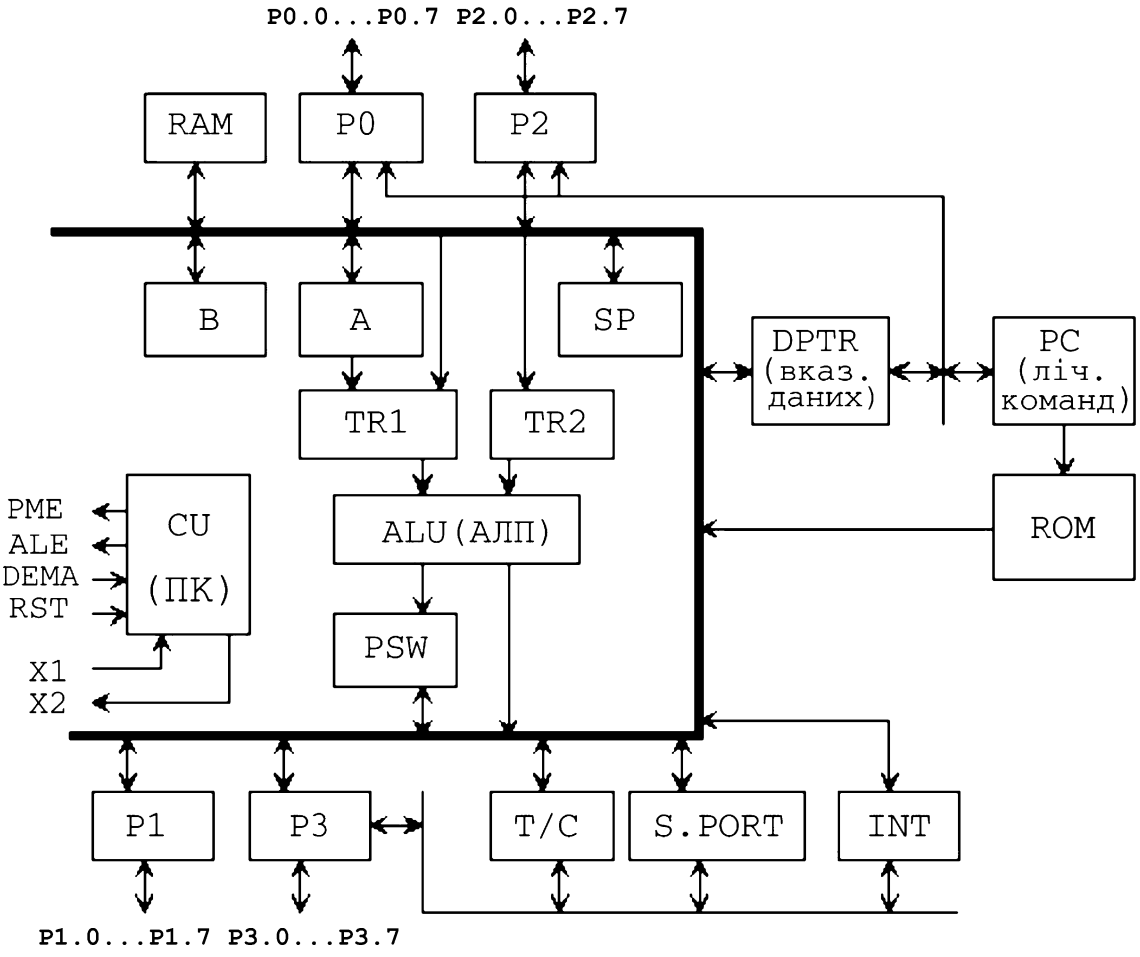


Рисунок 2. Функціональна схема мікроконтролера MCS-51

ЦП розділений на керуючий пристрій ПК (англ. CU - Control Unit) і чотири блоки, що є операційним пристроєм: арифметико-логічний пристрій АЛП (англ. ALU), регістри тимчасового збереження операндів TR1 і TR2, регістр ознак PSW (англ. Program Status Word).

Крім того, з RAM виділені три регістри спеціальних функцій: акумулятор (A), регістр B та регістр покажчика стеку SP, а блок РШ представлений покажчиком даних DPTR (англ. Data PoinTe) з асоційованими шинами обміну й лічильником команд PC (англ. Program Counter), призначення якого полягає у формуванні адреси команди. Наведена на рис. 2 архітектура покликана, у тому числі, підкреслити два важливих аспекти застосування ОМЕОМ 80С51:

- при використанні зовнішнього запам'товуючого пристрою (ЗП) його адресація формується через виводи портів P0 (молодший байт) і P2 (старший байт), а обмін інформацією (ввід кодів команд, вводу\виводу даних) – через виводи порту P0, що, якщо не застосовувати додаткові апаратні засоби, призводить до втрати цих портів для використання;
- таймери/лічильники, послідовний порт і система переривань не мають своїх виводів у корпусі ОМЕОМ, а використовують виводи порту P3 (це називається альтернативними функціями виводів порту P3), і, таким чином, використання таймерів/лічильників у режимі лічильників зовнішніх подій, ліній послідовного порту і зовнішніх переривань зменшує розрядність порту P3.

Проміжні результати обчислень зберігаються в ОЗП (RAM) ємністю 128 байт. Крім того, в адресному просторі ОЗП розміщені всі регістри спеціальних функцій, які визначають стан портів, таймери, регістр акумулятора (A), регістр B, регістр вказівника стеку SP. З блоку РШ в адресний простір ОЗП входить вказівник даних DPTR (англ. Data PoinTeR) з асоційованими шинами обміну й лічильником команд PC (англ. Program Counter), призначення якого полягає у формуванні адреси команди. Крім того, в ОЗП розташований стек, у якому можуть зберігатися адреси повернення з підпрограм та локальні змінні підпрограм.

Швидкість роботи ЦП задається генератором ГЕН, що виробляє необхідні для роботи часові послідовності. Тактова частота ГЕН задається або кварцовим резонатором, що вмикається між виводами X1 і X2, або зовнішнім генератором, що під'єднується до входу X1. З метою забезпечення послідовного доступу до ресурсів процесора при використанні однієї шини,

ГЕН формує машинний цикл процесора з дванадцяти тактів резонатора (задаючого генератора).

Машинний цикл містить 6 станів керуючого автомату S1...S6, кожен стан розділений на дві фази P1, P2, що відповідає різним діям процесора.

Введення у процесор чи виведення інформації, що опрацьовується, може бути здійснено або в паралельній байтовій (ввід восьми розрядів однією командою), або в послідовній (по одному біту) формах. Паралельний обмін інформації можливий через один з чотирьох підтримуваних ОМЕОМ паралельних портів. Послідовний обмін інформацією може бути організований через будь-який з розрядів паралельного порту. Однак для полегшення процесу послідовного обміну й економії обчислювальних ресурсів, необхідних для його реалізації, ОМЕОМ містить вбудований програмований послідовний порт, що дозволяє практично без витрат обчислювальних ресурсів організувати послідовний обмін по кількох видах протоколів.

Крім розглянутих вузлів, до складу ОМЕОМ входять два шістнадцятирозрядні таймери/лічильники, що можуть функціонувати або в режимі таймера, або в режимі лічильника зовнішніх подій.

Режим таймера використовується, головним чином, коли необхідно організувати циклічні процеси з жорстко фіксованим і незалежним від часу виконання програми періодом циклу. Наприклад, при опрацюванні сигналів, коли необхідно забезпечити потрібний інтервал дискретизації.

Режим лічильника зовнішніх подій використовується, наприклад, при підрахунку кількості імпульсів, вимірюванні частоти і т.п.

Розширювач шини РШ використовується для роботи з зовнішнім ЗП – пам’яті програм чи пам’яті даних. Як правило, зовнішнє ЗП використовується, коли для розміщення програми чи даних при вирішенні якогось завдання внутрішніх ресурсів ОМЕОМ виявляється недостатньо. Режим роботи з зовнішньою пам’яттю не є типовим для ОМЕОМ.

Режим звертання до зовнішнього ЗП використовується не тільки за прямим призначенням. Так як зовнішня шина універсальна, то до неї можна

під’єднати будь-який спеціалізований пристрій або навіть спеціалізовану мікроЕОМ і керувати їх роботою безпосередньо з MCS-51.

1.2.1. Умовне позначення та призначення виводів мікроЕОМ.

Слід зауважити, що в мікроЕОМ цього сімейства більшість виводів має подвійне функціональне призначення. Наприклад, при використанні зовнішнього ОЗП його адресація здійснюється через виводи портів P0 (молодший байт) і P2 (старший байт), а обмін інформацією (введення кодів команд, введення/виведення даних) – через виводи порту P0. При цьому керування обміном здійснюється через дві лінії порту P3. Тому, якщо не застосовувати додаткові апаратні засоби, ці порти за наявності зовнішнього ОЗП не можна використати для інших цілей.

Таймери/лічильники, послідовний порт і система переривань не мають своїх виводів з корпусу ОМЕОМ, а використовують виводи порту P3, і, таким чином, використання таймерів/лічильників у режимі лічильників зовнішніх подій, ліній послідовного порту і зовнішніх переривань знижує розрядність порту P3.

Специфічну для певної лінії функцію, яку виконує порт, називають *альтернативною* функцією лінії порту. А функцію простого вводу-виводу називають *основною* функцією порту.

Позначення ОМЕОМ на електричній схемі зображено на рис. 3, а призначення виводів наведено в таблиці 2.

Зауважимо, що лінії портів P1…P3 є *квазідвонаправленими* – фактично, вони працюють як виводи з відкритим колектором, у яких внутрішні підтягуючі до високого рівня напруги резистори знаходяться в корпусі мікроЕОМ. При встановленні відповідного біта порту в стан логічної «1» вивід забезпечує невеликий струм, що обмежується підтягуючим резистором, а при логічному «0» забезпечується невисокий значний струм. Це полегшує узгодження виводу із входами й виходами інших ТТЛ мікросхем.

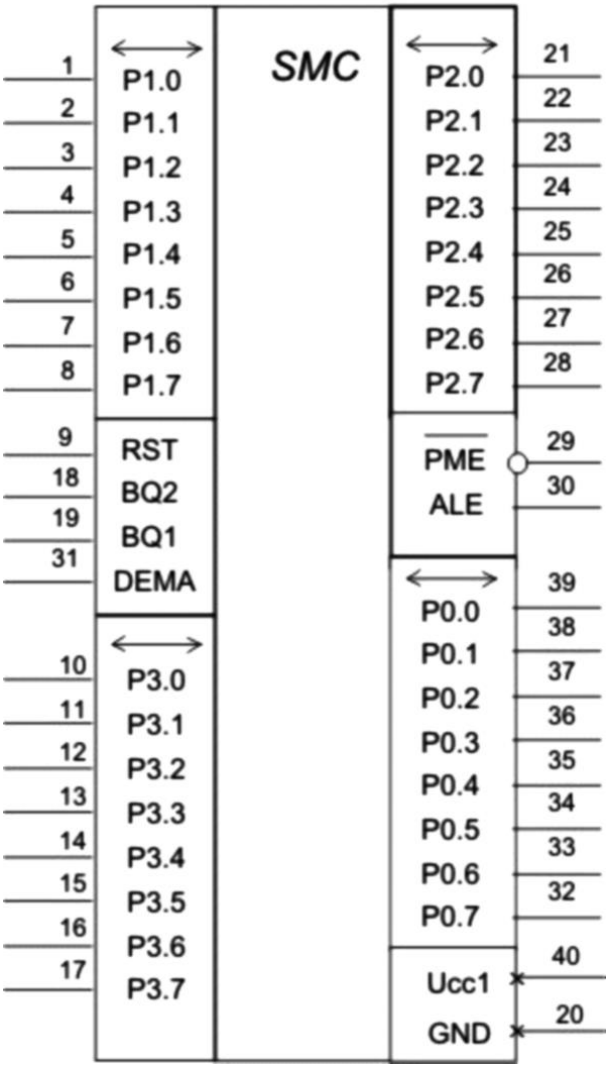


Рисунок 3. Позначення виводів мікроконтролера 87C51

Лінії порту P0 аналогічні лініям портів P1…P3, за винятком того, що внутрішні підтягуючі резистори в них взагалі відсутні – лінії можуть лише приймати струм у стані логічного нуля. Проте при використанні ліній у функції виходів, навантажених на входи мікросхем ТТЛ, внутрішньої підтяжки резисторами ТТЛ-входів достатньо для нормальної роботи виводів.

При зчитуванні стану лінії можливі два типи команд, які зчитують або стан лінії (команди вводу з порту), або стан внутрішнього регістра.

При зчитуванні стану лінії необхідно, щоб на відповідному виході мікроЕОМ було заздалегідь виставлено логічну «1». У протилежному випадку лінія буде переведена в стан логічного «0» і цей нуль і буде прочитаний.

Таблиця 2. Призначення виводів мікросхеми

Позначення	Номер виводу	Призначення
1	2	3
P1.0-P1.7	1–8	Порт вводу-виводу 1. Виводи паралельного порту P1
RST	9	Сигнал скидання мікросхеми або вхід напруги програмування внутрішнього ПЗП. Логічний «0» дозволяє нормальну роботу мікросхеми, логічна «1» протягом двох машинних циклів переводить мікросхему в режим скидання, рівень напруги 12 В на виводі призводить до програмування мікросхеми
P3.0/RxD	10	Лінія 0 порту вводу-виводу P3 може працювати як вхід даних у послідовному коді
P3.1/TxD	11	Лінія 1 порту вводу-виводу P3 може працювати як вихід даних у послідовному коді
P3.2/ $\overline{\text{INT0}}$	12	Лінія 2 порту вводу-виводу P3 може працювати як вхід запиту переривання по переходу сигналу в стан лог. «0»
P3.3/ $\overline{\text{INT1}}$	13	Лінія 3 порту вводу-виводу P3 може працювати як вхід запиту переривання по переходу сигналу в стан лог. «0»
P3.4/T0	14	Лінія 4 порту вводу-виводу P3 може працювати як вхід відліку внутрішнього лічильника 0
P3.5/T1	15	Лінія 5 порту вводу-виводу P3 може працювати як вхід відліку внутрішнього лічильника 1
P3.6/ $\overline{\text{WR}}$	16	Лінія 6 порту вводу-виводу P3 може працювати як сигнал запису у зовнішній ЗП даних або порт вводу-виводу
P3.7/ $\overline{\text{RD}}$	17	Лінія 7 порту вводу-виводу 3 може працювати як сигнал читання зовнішнього ЗП даних або порту вводу-виводу
BQ2 (X2)	18	Вивід для під’єднання зовнішнього резонатора
BQ1(X1)	19	Вивід для під’єднання зовнішнього резонатора
VSS (GND)	20	Напруга 0 В
P2.0/(A8)-P2.7/(A16)	21–28	Порт вводу-виводу 2, лінії порту також використовуються при адресації зовнішньої пам’яті та пристроїв вводу-виводу для виведення старшого байта адреси
PME	29	Сигнал читання зовнішньої пам’яті програм. Логічний «0» на лінії вказує, що відбувається читання зовнішньої пам’яті програм

Закінчення таблиці 2

1	2	3
ALE	30	При нормальній роботі це вихід стробуючого сигналу адреси. Перехід сигналу зі стану лог. «1» у стан лог. «0» вказує на наявність адреси на лініях порту 0. У режимі програмування лінія є входом, на який надходить імпульс програмування
DEMA	31	Вхід заборони роботи внутрішньої пам’яті програм. Логічна «1» на вході дозволяє звертання до внутрішньої пам’яті програм, при логічному «0» на вході при звертанні за адресами, що відповідають внутрішній пам’яті програм, звертання відбувається до зовнішньої пам’яті програм
P0.0-P0.7	39–32	Порт вводу-виводу 0, лінії порту також використовуються при звертанні до зовнішньої пам’яті програм та даних, а також до пристроїв вводу-виводу для виведення молодшого байта адреси та обміну даними
Vcc1	40	Напруга живлення 5В

Мінімальна схема увімкнення OMEOM 87C51 зображена на рис. 4.

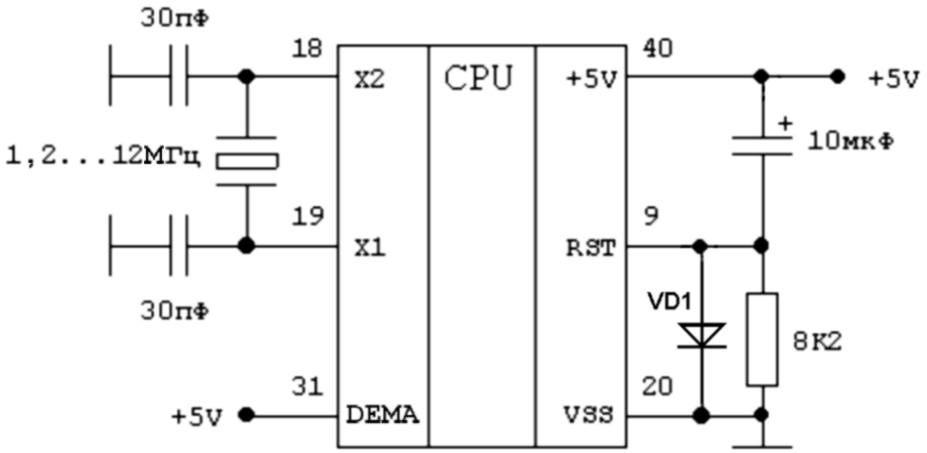


Рисунок 4. Мінімальна схема увімкнення OMEOM 87C51

До виводів X1 і X2 OMEOM під’єднана резонансна схема, що містить кварцовий резонатор. До входу скидання RST під’єднана схема автоматичного рестарту OMEOM при увімкненні живлення.

Коли функціонально-логічних можливостей однокристальної ОМЕОМ недостатньо, можливе розширення мікроконтролерної системи (МК-системи) відносно простими методами з отриманням таких значень параметрів: пам’ять програм – до 64 Кбайт; пам’ять даних – до 64 Кбайт; лінії вводу-виводу – практично необмежено. Крім того, шляхом під’єднання інших спеціалізованих мікросхем, у МК-системі можуть бути реалізовані різні допоміжні функції: зв’язок з дисплеєм і клавіатурою, багаторівнева система переривань, зв’язок з телеграфно-телефонними лініями передавання інформації і т.д.

1.3. МК – системи з зовнішньою пам’яттю програм.

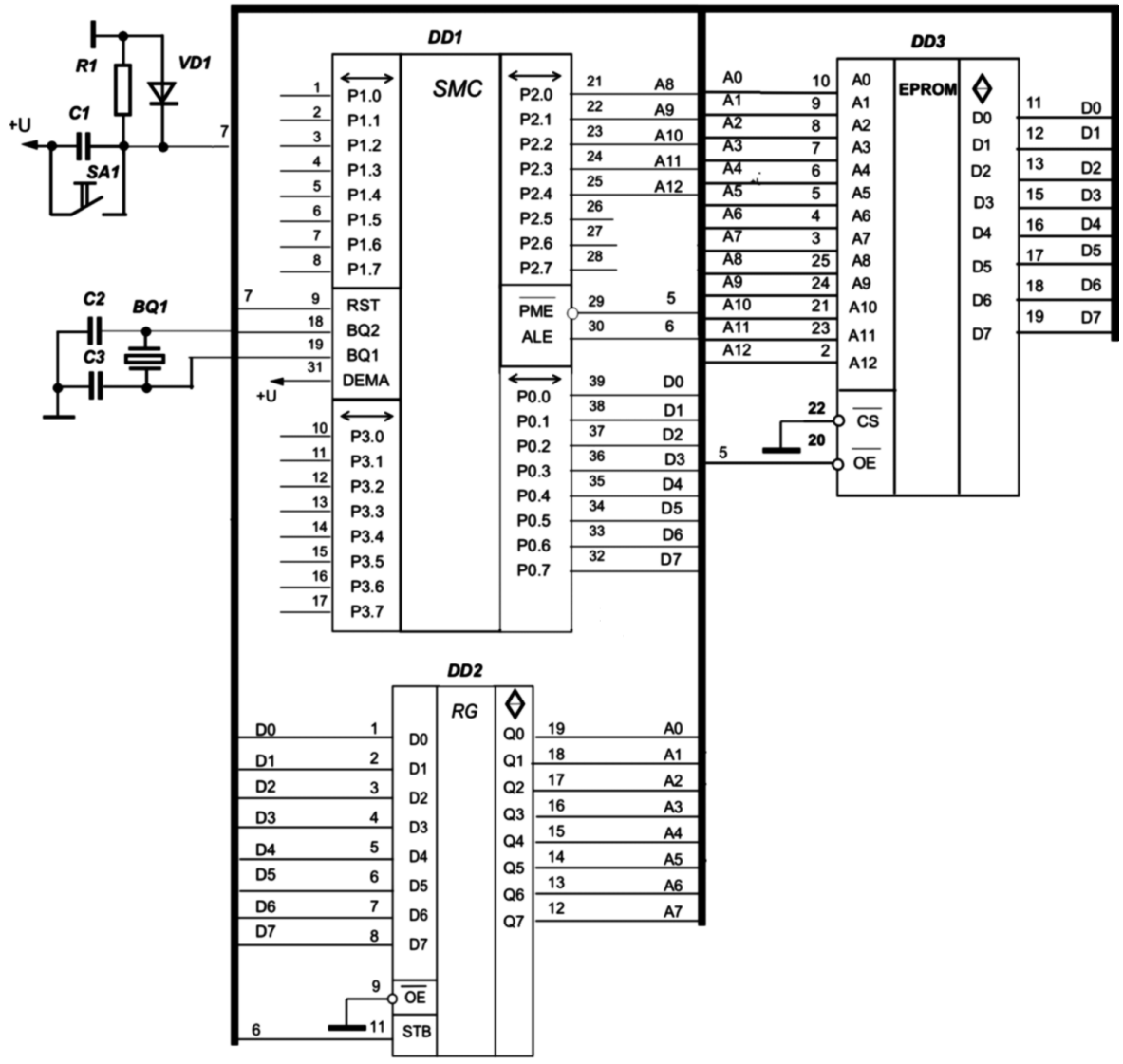


Рисунок 5. Під’єднання нерезидентного ПЗП DD3 (2764)

На рис. 5 зображено структуру МК-системи з зовнішньою (нерезидентною) пам’яттю програм. Шина P0 за своїми властивостями є двонаправленою шиною даних і всі розширення МК виконуються для цієї шини. При звертанні до резидентної (внутрішньої) пам’яті програм МК не генерує зовнішніх керуючих сигналів (за винятком ALE, що ідентифікує кожен машинний цикл, проте в деяких випадках він теж може бути відсутній). Починаючи з адреси 4096, МК автоматично формує керуючі сигнали, що забезпечують вибірку команд із зовнішньої пам’яті об’ємом до 64 Кбайт.

Послідовність процесу вибірки команди з зовнішньої пам’яті така:

◊ вміст лічильника команд виводиться через порт P0 (BUS) і порт P2 (P2.0...P2.7);

◊ за зрізом сигналу ALE на зовнішньому регістрі фіксується адреса;

◊ сигналом PME дозволяється робота зовнішньої пам’яті;

◊ за спадом сигналу PME шина P0 переходить у режим введення.

Додаткова мікросхема пам’яті DD3 (2764) ємністю 8 Кбайт під’єднується до шини P0 своїми інформаційними виходами. Молодший байт адреси за сигналом ALE фіксується на зовнішньому буферному регістрі DD2 (8282). Старша тетрада адреси, виведена через порт P2, немає потреби в буферизації, тому що вона зберігається на лініях порту протягом усього циклу вибірки. У випадку використання великої кількості мікросхем ПЗП, ОЗП та інших пристроїв, під’єднаних до шини даних, необхідно використовувати зовнішній адресний дешифратор та буферизувати шину даних спеціалізованими мікросхемами.

1.4. Розширення ОЗП.

На рисунку 6 зображено схему МК-системи, до складу якої входить додаткова мікросхема статичного ОЗП DD3 (TC5516), на основі якої реалізується пам’ять ємністю 2 Кбайт.

Сигналом ALE непряма адреса, виведена по шині P0, фіксується в буферному регістрі DD2. Сигнали W і R визначають режим роботи ОЗП. Схема,

що на рис. 6, забезпечує адресацію 2 Кбайт комірок ОЗП на додаток до 256 комірок резидентної пам’яті даних (РПД) MCS-51.

Молодші вісім розрядів шини адреси при звертанні до ОЗП видаються на порт P0 і запам’ятовуються в зовнішньому регістрі DD2. Старші розряди шини адреси при використанні команд, що працюють із регістром DPTR, видаються на лінії порту P2 і утримуються там протягом усього циклу. Звертання до зовнішнього ОЗП із використанням 8-бітної адресації не змінює стан ліній порту P2, отже, вимагає попереднього встановлення потрібного значення старших розрядів адреси за допомогою порту P2.

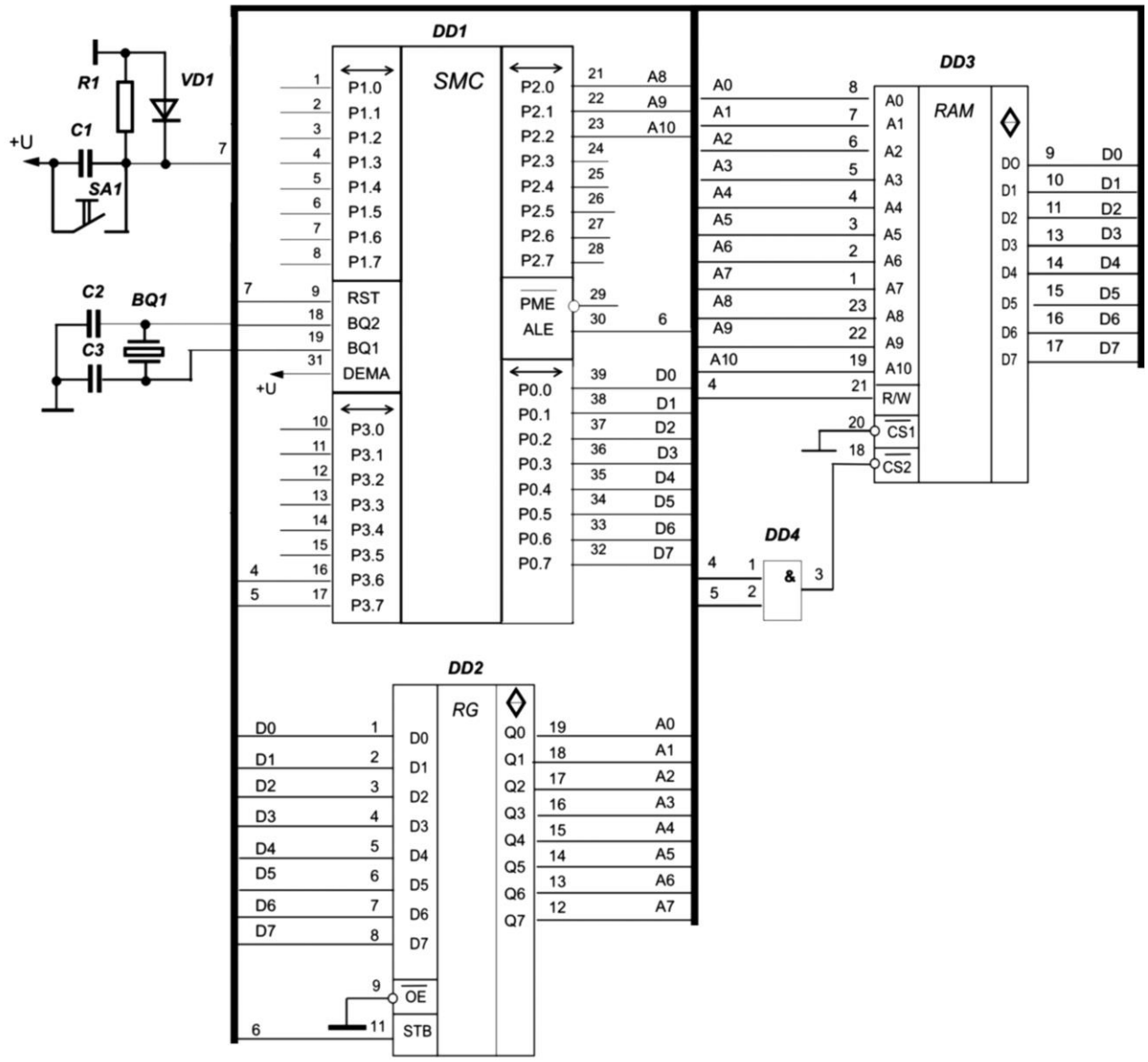


Рисунок 6. Розширення пам’яті даних для мікроЕОМ 87С51

1.5. Ввід-вивід у МК-системах.

Для з'єднання МК з об'єктом, що має велику кількість входів-виходів, можна розширити резидентну систему вводу-виводу за рахунок зовнішніх портів. Таке розширення можна виконати з використанням стандартного розширювача вводу-виводу (РВВ), або інтерфейсних мікросхем.

Розширювач під'єднується до MCS-51 так, як показано на рис. 7. Кожен із чотирьох портів РВВ може використовуватися для введення чи виведення інформації незалежно від інших і забезпечує високу навантажувальну здатність.

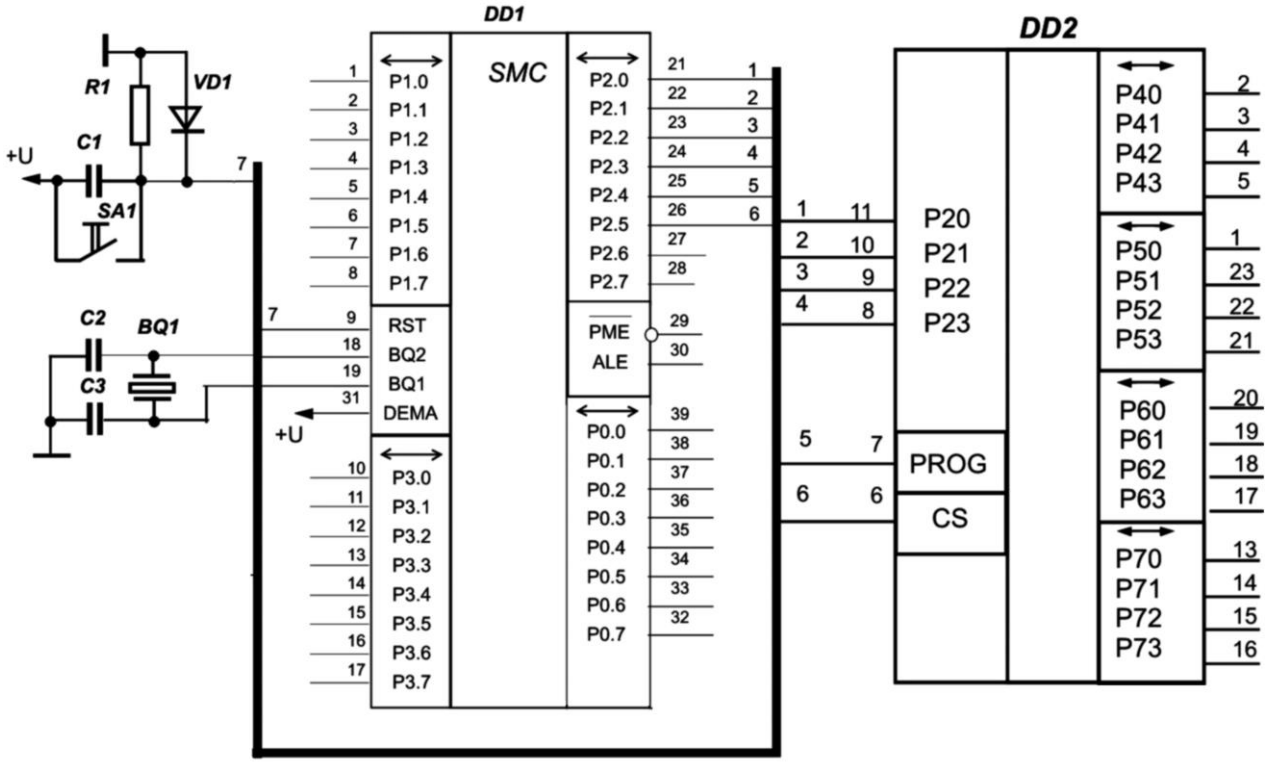


Рисунок 7. Під'єднання розширювача вводу-виводу DD2 (8243) до MCS-51

Розширення ПВВ за допомогою додаткових мікросхем вводу/виводу здійснюється аналогічно прикладу, зображеному на рис. 8.

Мікросхема 8255А для мікроЕОМ відображається на комірки зовнішнього ОЗП даних. Формування адреси здійснюється за допомогою регістра DD2, який фіксує адресу, котра видається мікросхемою в циклі звертання до зовнішньої пам'яті даних. Старший біт адреси надходить на вхід

вибірки CS мікросхеми. Це зроблено для спрощення системи. У випадку необхідності використання кількох мікросхем 8255A, їх вибірку можна здійснювати за допомогою розрядів адреси A2-A7, під'єднаних до сигналів вибірки відповідних мікросхем.

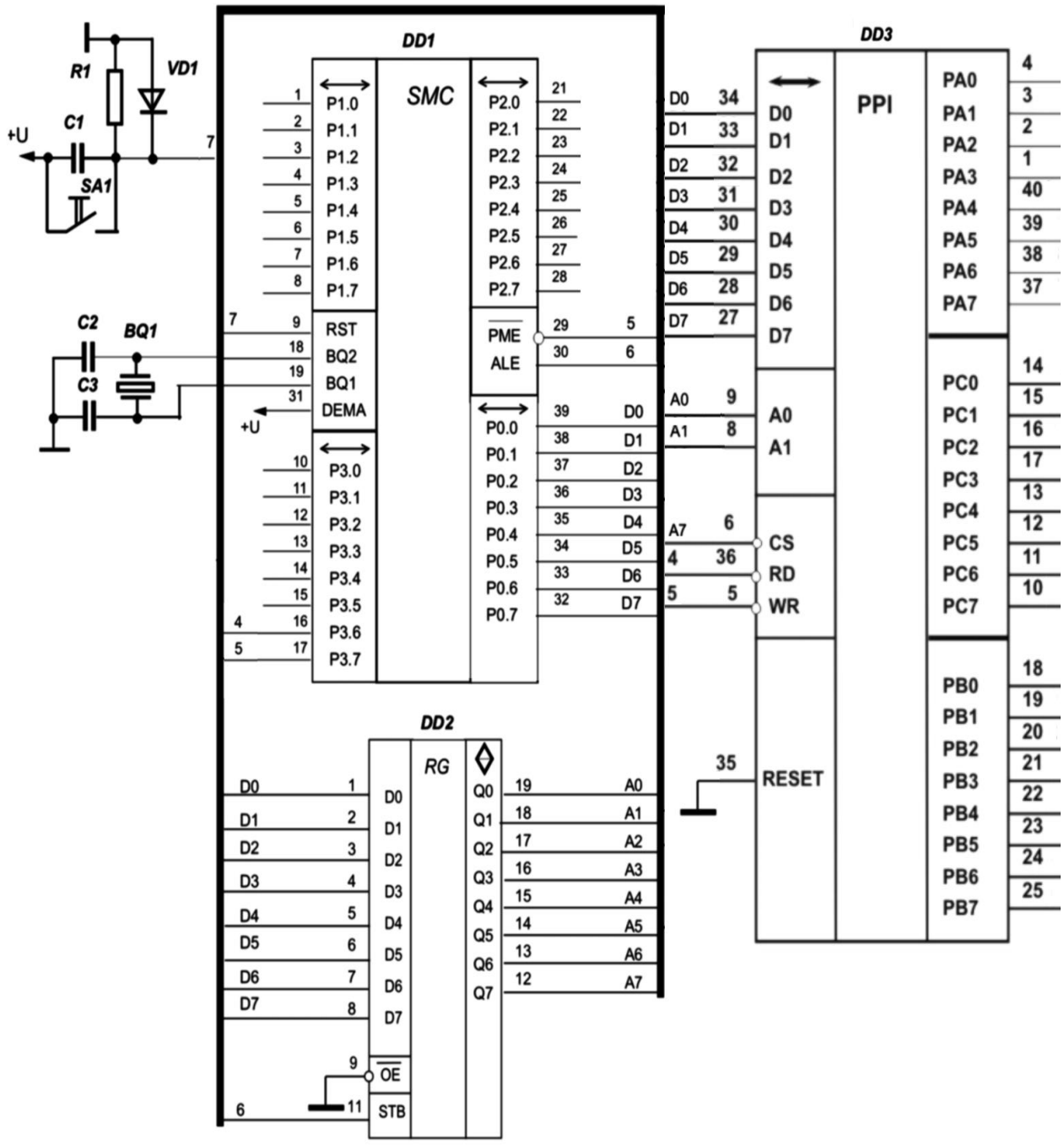


Рисунок 8. Приклад під'єднання ППІ 8255A до мікроЕОМ

У випадку великої кількості необхідних мікросхем формування сигналів вибірки можна здійснювати за допомогою дешифратора, на який подаються

старші біти адреси мікроЕОМ або з використанням для формування сигналів вибірки ліній порту Р1 чи Р2. Організація більшості програмних процедур для MCS-51 повинна враховувати те, що в MCS-51 кожен вивід кожного порту може бути перевірений однією командою без необхідності завантаження стану порту в акумулятор.

1.6. Таймер/лічильник MCS-51.

Два програмовані 16-бітні таймери/лічильники (Т/С0 і Т/С1) можуть бути використані як таймери або лічильники зовнішніх подій. При роботі в режимі *таймера* вміст Т/С інкрементується у кожному машинному циклі, тобто через кожні 12 періодів резонатора. При роботі в режимі *лічильника* вміст Т/С інкрементується під впливом переходу з «1» у «0» зовнішнього вхідного сигналу, що подається на відповідний вивід MCS-51 (Т0 або Т1). Опитування значення зовнішнього вхідного сигналу виконується в момент часу S5P2 кожного машинного циклу. Вміст лічильника буде збільшений на 1 у випадку, якщо в попередньому циклі надійшов сигнал високого рівня «1», а в наступному – сигнал низького рівня «0». Нове значення лічильника буде сформоване в момент S3P1 у циклі, що йде за тим, у якому був виявлений перехід сигналу з рівня «1»у «0». На розпізнавання переходу потрібно два машинні цикли, тому максимальна частота підрахунку вхідних сигналів дорівнює 1/24 частоти резонатора. На тривалість періоду вхідних сигналів обмежень згори немає. Для гарантованого зчитування вхідного сигналу, що підраховується, він повинен утримувати значення «1» як мінімум протягом одного машинного циклу MCS-51.

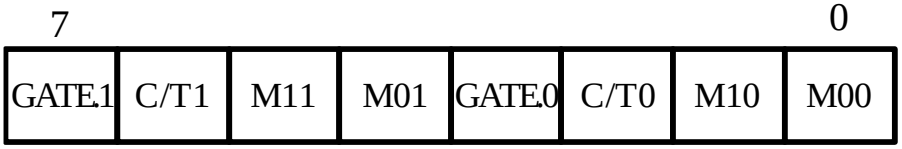


Рисунок 9. Формат регістра режиму TMOD

Таблиця 3. Регістр режиму роботи таймера/лічильника TMOD

Ознака	Ім’я і призначення
GATEx	Керування блокуванням. Якщо біт установлено, то таймер/лічильник «x» дозволений доти, доки на вході «INTx» високий рівень і біт керування «TRx» встановлений. Якщо біт скинутий, то T/Cx дозволяється як тільки біт керування «TRx» встановлюється
C/Tx	Біт вибору режиму таймера або лічильника подій. Якщо біт скинутий, то блок працює в режимі таймера від внутрішнього джерела сигналів синхронізації. Якщо біт встановлено, то блок працює в режимі лічильника від зовнішніх сигналів на вході «Tx»
M1.x, M0.x	Режим роботи, що вибирається згідно з таблицею 4

Старший та молодший байти лічільника носять назву THx та TLx. Керування лічильниками здійснюється загальним *регістром режиму* TMOD та *регістром конфігурації* TCON, стан лічильників відображається у регістрі TCON.

Опис регістрів наведено в таблицях 3, 4 та 5, структура регістрів зображена на рис. 9 та 10.

Таблиця 4. Режим роботи таймера залежно від бітів настроювання

M1	M0	Режим роботи
0	0	Таймер сумісний з MCS-48. TL працює як 5-бітний попередній подільник, TH – у режимі, сумісному з таймером MCS-48
0	1	16-бітний таймер/лічильник. TL і TH вімкнені послідовно
1	0	8-бітний таймер/лічильник, що перезавантажується. TH зберігає значення, що повинно бути перезавантажене в TLx у момент переповнення
1	1	Таймер/лічильник 1 зупиняється. У таймері/лічильнику 0 TL0 працює як 8-бітний таймер/лічильник, і його режим визначається керуючими бітами таймера 0. TH0 працює тільки як 8-бітний таймер, його режим визначається керуючими бітами таймера 1

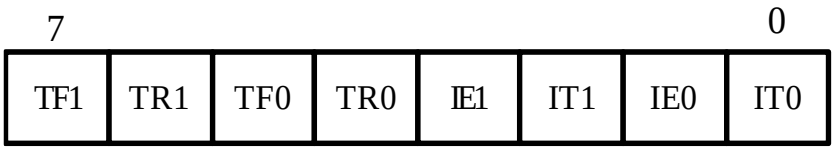


Рисунок 10. Формат регістра керування/статусу таймера

Таблиця 5. Регістр керування/статусу таймера

Ознака	Ім’я і призначення
TFx	Ознака переповнення таймера x. Встановлюється апаратно при переповненні таймера/лічильника. Скидається при обслуговуванні переривання теж апаратно
TRx	Біт керування таймера x. Встановлюється/скидається програмою для його пуску (1) та зупинки (0)
IEx	Ознака запиту переривання x. Встановлюється апаратно, коли детектується активний перехід сигналу INTx. Скидається при обслуговуванні переривання
ITx	Біт керування типом переривання x. При ITx = 0 активним є перехід 1→0, при IT = 0 активним є рівень лог. 0 на вході INTx

Як випливає з опису керуючих бітів TMOD, для обох лічильників режими роботи 0, 1 і 2 однакові.

1.6.1. Режими роботи таймера.

Режим 0. У цьому режимі таймерний регістр має розрядність 13 бітів. При переході зі стану «всі одиниці» у стан «всі нулі» встановлюється ознака переривання від таймера TFx. Вхідний синхросигнал таймера дозволений (надходить на вхід Tx), коли керуючий біт TR1 встановлений в «1» або керуючий біт GATE дорівнює «0», або на зовнішній вивід запиту переривання INT1 надходить рівень «1» (рис. 11 а).

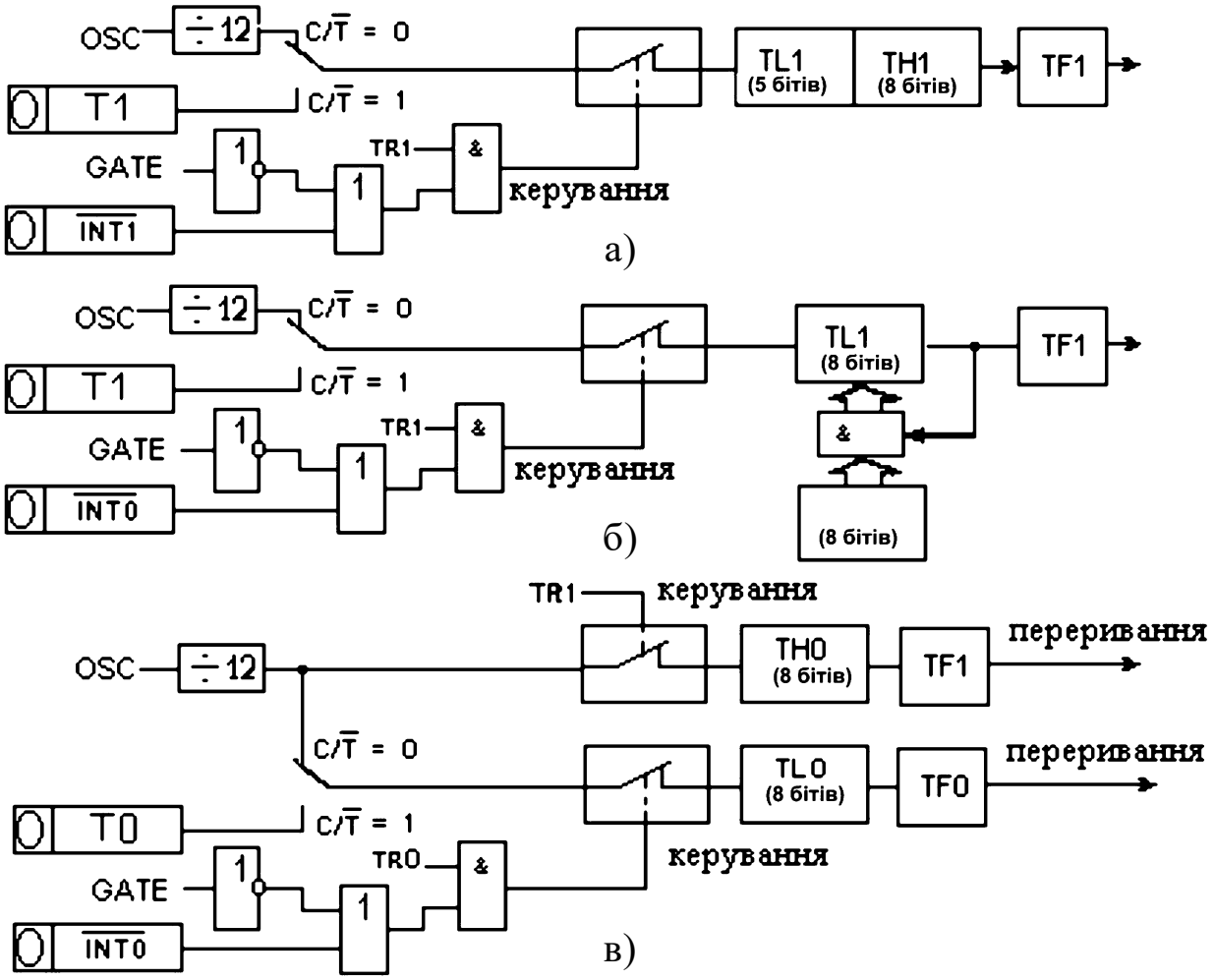


Рисунок 11. Таймер/лічильник подій:
а – у режимі 0: 13-бітний лічильник; б – у режимі 2: 8-бітний лічильник з перезавантаженням; у – T/C0 у режимі 3: два 8-бітні лічильники

Встановлення біта GATE в «1» дозволяє використовувати таймер для вимірювання тривалості імпульсного сигналу, що подається на вхід запиту переривання.

Режим 1. Робота будь-якого T/C у режимі 1 така ж, як і в режимі 0, за винятком того, що таймерний регістр має розрядність 16 бітів.

Режим 2. У режимі 2 (рис. 11 б) робота організована таким чином, що переповнення (перехід зі стану «усі одиниці» у стан «усі нулі») 8-бітного лічильника TLx призводить не тільки до встановлення ознаки TF, але й автоматично перезавантажує у TLx уміст старшого байта (THx) таймерного регістра, що попередньо було задано програмним шляхом. Перезавантаження

залишає вміст THx незмінним. У режимі 2 таймери/лічильники працюють ідентично.

Режим 3. У режимі 3 таймери/лічильники працюють по-різному. Таймер/лічильник 1 зберігає незмінним свій поточний вміст, іншими словами, ефект такий же, як і при скиданні керуючого біта TRI в нуль. Роботу таймера/лічильника 0 у режимі 3 проілюстровано на рис.11 в.

У режимі 3 TL0 і TH0 функціонують як два незалежних 8-бітні лічильники. Роботу TL0 визначають керуючі біти T/C0 (C/T, GATE, TR0), вхідний сигнал INT0 і ознака переповнення TF0. TH0 може виконувати тільки функції таймера, його роботу визначає керуючий біт TR1, TH0 використовує ознаку переповнення TF1.

Режим 3 використовується у випадках, коли потрібна наявність додаткового 8-бітного таймера або лічильника подій. Можна вважати, що в режимі 3 MCS-51 має в своєму складі три таймери/лічильники. У випадку, якщо таймер/лічильник 0 використовується в режимі 3, таймер/лічильник 1 може бути увімкнений, вимкнений, переведений у свій власний режим, використаний послідовним портом у якості генератора частоти, чи, нарешті, використаний для виконання будь-якої функції, що не вимагає переривання.

1.7. Послідовний інтерфейс.

1.7.1. Універсальний асинхронний прийомопередавач.

Суттєвою відмінністю MCS-51 є наявність на кристалі процесора блоків, що дозволяють апаратно реалізовувати канал послідовного зв’язку.

Через послідовний порт, який часто називають універсальним асинхронним прийомопередавачем (УАПП), здійснюється приймання і передавання інформації, поданої послідовним кодом (молодшими бітами вперед), у повному дуплексному режимі обміну. До складу УАПП входять приймаючий та передаючий зсувні регістри, а також спеціальний буферний регістр (SBUF) прийомопередавача. Запис байта в буфер призводить до автоматичного перезаписування байта в зсувний регістр передавача й ініціює початок передавання байта. Наявність буферного регістра приймача дозволяє суміщати операцію читання раніше

прийнятого байта з прийманням чергового байта. Якщо до моменту закінчення приймання байта попередній байт не був зчитаний із SBUF, то він буде втрачений.

Послідовний порт MCS-51 може працювати в *чотирьох* різних режимах.

У *режимі 0* інформація передається і приймається через зовнішній вивід входу приймача (RXD). Приймаються або передаються 8 бітів даних. Через зовнішній вивід виходу передавача (TXD) видаються імпульси зсуву, що супроводжують кожен біт. Частота передавання біта інформації дорівнює 1/12 частоти резонатора.

У *режимі 1* передаються через TXD або приймаються з RXD 10 бітів інформації: старт-біт (0), 8 бітів даних і стоп-біт (1). Швидкість приймання/передавання змінна і задається таймером.

У *режимі 2* через TXD передаються або з RXD приймаються 11 бітів інформації: старт-біт, 8 бітів даних, дев'ятий біт, стан якого програмується, та стоп-біт. При передаванні дев'ятий біт даних може приймати значення «0» або «1», або, наприклад, для підвищення надійності в нього може бути записане значення ознаки паритету зі слова стану програми (PSW.0). Швидкість обміну даними вибирається програмою і може дорівнювати або 1/32, або 1/64 частоти резонатора залежно від керуючого біта SMOD.

Режим 3 аналогічний режиму 2, проте швидкість обміну даними змінна і задається таймером.

1.7.2. Регістр керування/статусу УАПП.

Керування режимом роботи УАПП здійснюється через спеціальний регістр SCON. Він містить не тільки керуючі біти, що визначають режим роботи послідовного порту, але й дев'ятий біт прийнятих або переданих даних (RB8 і TB8) і біти переривання прийомопередавача (RI і TI).

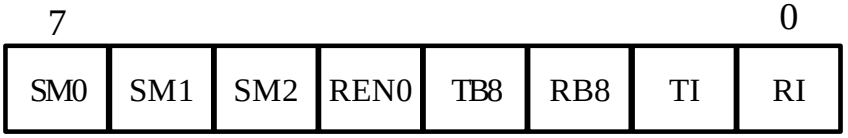


Рисунок 12. Формат регістра керування та статусу УАПП (SCON)

Формат регістра, що керує роботою УАПП, зображено на рис. 12. Призначення бітів регістра керування/статусу УАПП наведено в таблиці 6.

Таблиця 6. Регістр керування/статусу УАПП

Символ	Ім’я й призначення
SM0, SM1	Біти вибору режиму роботи УАПП. Встановлюються та скидаються програмно. Записаний у біти код вказує номер режиму
SM2	Біт керування режимом УАПП. Встановлюється програмно для заборони приймання повідомлення, у якому дев’ятий біт має значення 0
REN	Біт дозволу приймання. Встановлюється та скидається програмно для дозволу або заборони приймання послідовних даних
TB8	Стан восьмого біта передавача. Встановлюється та скидається програмно для задавання дев’ятого переданого біта в 9-бітному режимі УАПП
RB8	Стан восьмого біта приймача. Встановлюється/скидається апаратно для фіксації дев’ятого прийнятого біта в 9-бітному режимі
TI	Ознака переривання передавача. Встановлюється апаратно в момент закінчення передавання байта. Скидається програмно після обслуговування переривання
RI	Ознака переривання приймача. Установлюється апаратно у момент закінчення приймання байта. Скидається програмно після обслуговування переривання

Прикладна програма шляхом завантаження в старші біти SCON 2-бітного коду визначає режим роботи УАПП. В усіх *чотирьох* режимах роботи передавання з УАПП ініціюється будь-якою командою, що записує дані в буферний регістр SBUF. Приймання даних УАПП у режимі 0 здійснюється за умови, що RI = 0 і REN = 1. У режимах 1, 2, 3 приймання даних починається з приходом старт-біта, якщо REN = 1.

У біті TB8 програмно встановлюється значення дев’ятого біта даних, який буде переданий у режимі 2 або 3. У біті RB8 фіксується в режимах 2 і 3 дев’ятий прийнятий біт даних. У режимі 1, якщо SM2 = 0, у біт RB8 заноситься стоп-біт. У режимі 0 біт RB8 не використовується.

Ознака переривання передавача TI встановлюється апаратно в кінці періоду передавання восьмого біта даних у режимі 0 і на початку періоду передавання стоп-біта в режимах 1, 2 і 3. Відповідна підпрограма обслуговування переривання повинна скидати біт TI.

Ознака переривання приймача RI встановлюється апаратно в кінці періоду приймання восьмого біта даних у режимі 0 і в середині періоду приймання стоп-біта в режимах 1, 2 і 3. Підпрограма обслуговування переривання повинна скидати біт RI.

1.7.3. Робота УАПП у мультимікроконтролерних системах.

У системах децентралізованого керування, що використовуються для керування й регулювання в розподілених об’єктах (наприклад, прокатних станах, рухомому складі залізниці й метрополітену, складальних конвеєрах і лініях гнучких автоматизованих виробництв) виникає задача обміну інформацією між множиною мікроконтролерів, об’єднаних у локальну обчислювально-керуючу мережу. Як правило, локальні мережі на основі MCS-51 мають магістральну архітектуру з розподіленим моноканалом (коаксіальний кабель, вита пара, оптичне волокно), по якому здійснюється обмін інформацією між МК.

У регістрі спеціальних функцій SCON мікроконтролера є керуючий біт SM2, що у режимах 2 і 3 УАПП дозволяє відносно простими засобами реалізувати міжконтролерний обмін інформацією в локальних керуючих мережах.

Механізм міжконтролерного обміну інформацією через послідовний порт MCS-51 побудований на тому, що в режимах 2 і 3 програмований дев’ятий біт даних при прийманні фіксується в біті RB8. УАПП може бути запрограмований таким чином, що при отриманні стоп-біта переривання від приймача буде можливим тільки за умови RB8 = 1. Це виконується встановленням керуючого біта SM2 у регістрі SCON.

Процес міжконтролерного обміну інформацією реалізується наступним чином. Нехай головному МК потрібно передати блок даних деякому веденому

МК. З цією метою головний МК у протокольному режимі «широкомовного» передавання (усім веденим МК) видає в моноканал байт-ідентифікатор абонента (код адреси МК – отримувача), який відрізняється від байтів даних тільки тим, що в його дев'ятому біті міститься 1. Програма реалізації протоколу мережевого обміну інформацією повинна бути побудована таким чином, щоб при отриманні байта-ідентифікатора (RB8 = 1) в усіх ведених МК відбулося переривання прикладних програм і виклик підпрограми порівняння байта-ідентифікатора з кодом власної мережевої адреси. Адресований МК скидає свій керуючий біт SM2 і готується до приймання блока даних. Інші ведені МК, адреса яких не збіглася з кодом байта-ідентифікатора, залишають незмінним стан SM2 = 1. При SM2 = 1 інформаційні байти, передані по моноканалу, і ті, що надходять в УАПП ведених МК, не викликають переривання, тобто ігноруються.

У режимі 1 УАПП автономного МК керуючий біт SM2 використовується для контролю істинності стоп-біта (при SM2 = 1 переривання не відбудеться до тих пір, поки не буде отримане істинне (одиничне) значення стоп-біта).

У режимі 0 біт SM2 не використовується і повинен бути скинутий.

1.7.4. Швидкість послідовного обміну.

Швидкість послідовного обміну даними УАПП у різних режимах визначається різними способами.

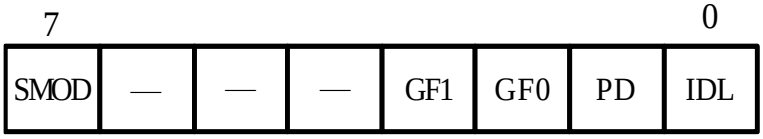


Рисунок 13. Формат регістра спеціальних функцій PCON

У режимі 0 швидкість обміну залежить лише від резонансної частоти кварцового резонатора (fрез) і дорівнює $f_0 = f_{рез}/12$. За один машинний цикл послідовний порт передає один біт інформації. В режимах 1, 2 і 3 швидкість обміну даними залежить від значення керуючого біта SMOD у регістрі спеціальних функцій (табл. 7, рис. 13).

Таблиця 7. Біти регістра спеціальних функцій

Символ	Ім’я і призначення
SMOD	Подвоєна швидкість обміну. Якщо біт встановлений в «1», то швидкість обміну вдвічі більша, ніж при SMOD = «0»
–	Не використовуються
GF1, GF0	Ознаки, що визначаються користувачем
PD	Біт зниженої потужності. При встановленні біта в «1» МК переходить у режим зниженої споживаної потужності
IDL	Біт холостого ходу. Якщо біт встановлений в «1», то МК переходить у режим холостого ходу

У режимах 1 і 3 у формуванні швидкості обміну крім керуючого біта SMOD бере участь таймер 1 (табл. 8). При цьому швидкість обміну залежить від частоти переповнення таймера (OVT1) і визначається в спосіб:

$$f_{1,3} = (2^{SMOD}/32)\cdot f_{OVT1}.$$

Таблиця 8. Налаштування таймера 1 для керування частотою роботи УАПП

Швидкість обміну	Режим роботи	Частота резонатора, МГц	SMOD	Число, що заноситься в регістр TH1
1 МГц	0	12	X	X
375 кГц	2	12	1	X
62.5 кГц	1, 3	12	1	0FFH
19.2 кГц	1, 3	11.059	1	0FDH
9.6 кГц	1, 3	11.059	0	0FDH
4.8 кГц	1, 3	11.059	0	0FAH
2.4 кГц	1, 3	11.059	0	0F4H
1.2 кГц	1, 3	11.059	0	0E8H
137.5 Гц	1, 3	11.059	0	1DH
110 Гц	1, 3	6	0	72H

Переривання від таймера 1 у цьому випадку повинно бути заблоковано. Сам таймер/лічильник може працювати і як таймер, і як лічильник подій у будь-якому із трьох режимів. Однак найзручніше використовувати режим таймера з автоперезавантаженням. При цьому швидкість обміну визначається виразом.

$$f_{1,3} = (2^{SMOD}/32)\cdot(f_{рез}/12)/(256 - (TH1)).$$

У таблиці 8 наведено опис способів налаштування таймера/лічильника для отримання типових швидкостей обміну даними через УАПП.

1.7.5. Особливості роботи УАПП у різних режимах.

1.7.5.1. Режим 0.

На рис. 14 зображено спрощену часову діаграму роботи УАПП у режимі 0. Дані передаються і приймаються через вивід RXD. Через вивід TXD видаються синхросигнали зсуву.

Передавання починається будь-якою командою, за якою у SBUF надходить байт даних. У момент часу S6P2 пристрій керування MCS-51 за сигналом «Запис у буфер» записує байт у зсувний регістр передавача, встановлює тригер дев'ятого біта і запускає блок керування обміном, котрий через один машинний цикл виробляє сигнал дозволу посилання даних. При цьому, в момент S6P2 кожного машинного циклу вміст зсувного регістра зсувається вправо і надходить на вивід RXD. У старші біти зсувного регістра передавача записуються нулі. При отриманні від детектора нуля сигналу «Передавач порожній» блок керування передавачем знімає сигнал «Посилання» і встановлює ознаку TI (момент S1P1 десятого машинного циклу після надходження сигналу «Запис у буфер»).

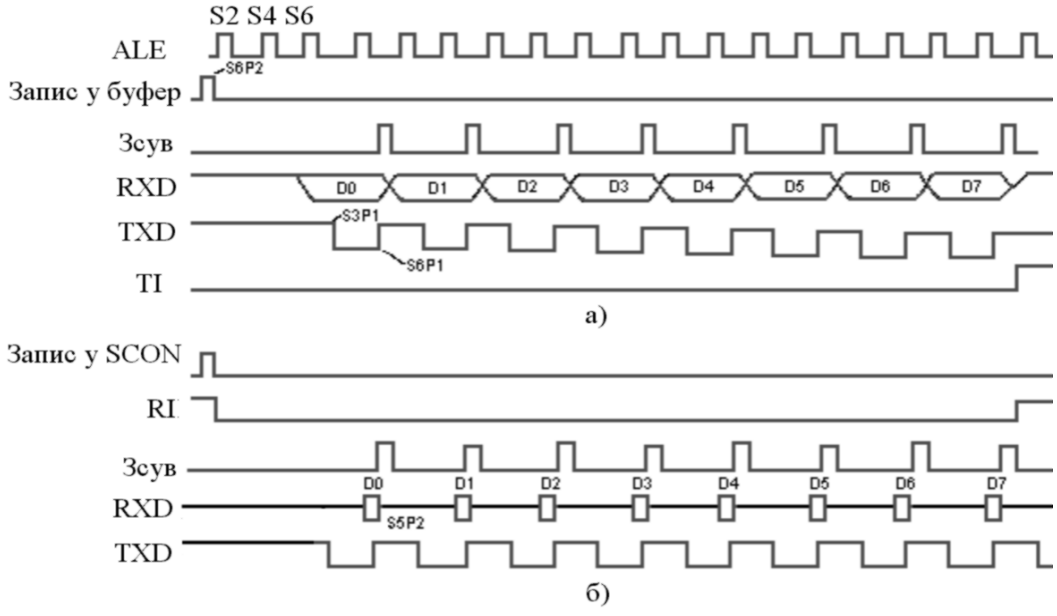


Рисунок 14. Часова діаграма обміну в режимі 0:
а) передавання даних, б) приймання даних

Приймання починається за умови REN = 1 і RI = 0. У момент S6P2 наступного машинного циклу блок керування приймачем формує сигнал дозволу прийому, по якому на вихід TXD передаються синхросигнали зсуву і в зсувному регістрі приймача починають формуватися значення бітів даних, що зчитуються з входу RXD у моменти S5P2 кожного машинного циклу. У момент S1P1 десятого машинного циклу після сигналу «Запис у SCON» блок керування приймачем переписує вміст зсувного регістра в буфер, знімає сигнал, що дозволяє прийом, і встановлює прапорець RI.

1.7.5.2. Режим 1.

Часова діаграма роботи УАПП у режимі 1 зображена на рис. 15. Через вивід TXD УАПП передає, а вивід RXD приймає 10 бітів: старт-біт, 8 бітів даних, можливо, дев'ятий біт даних і стоп-біт. При прийманні стоп-біт надходить у біт RB8 регістра SCON.

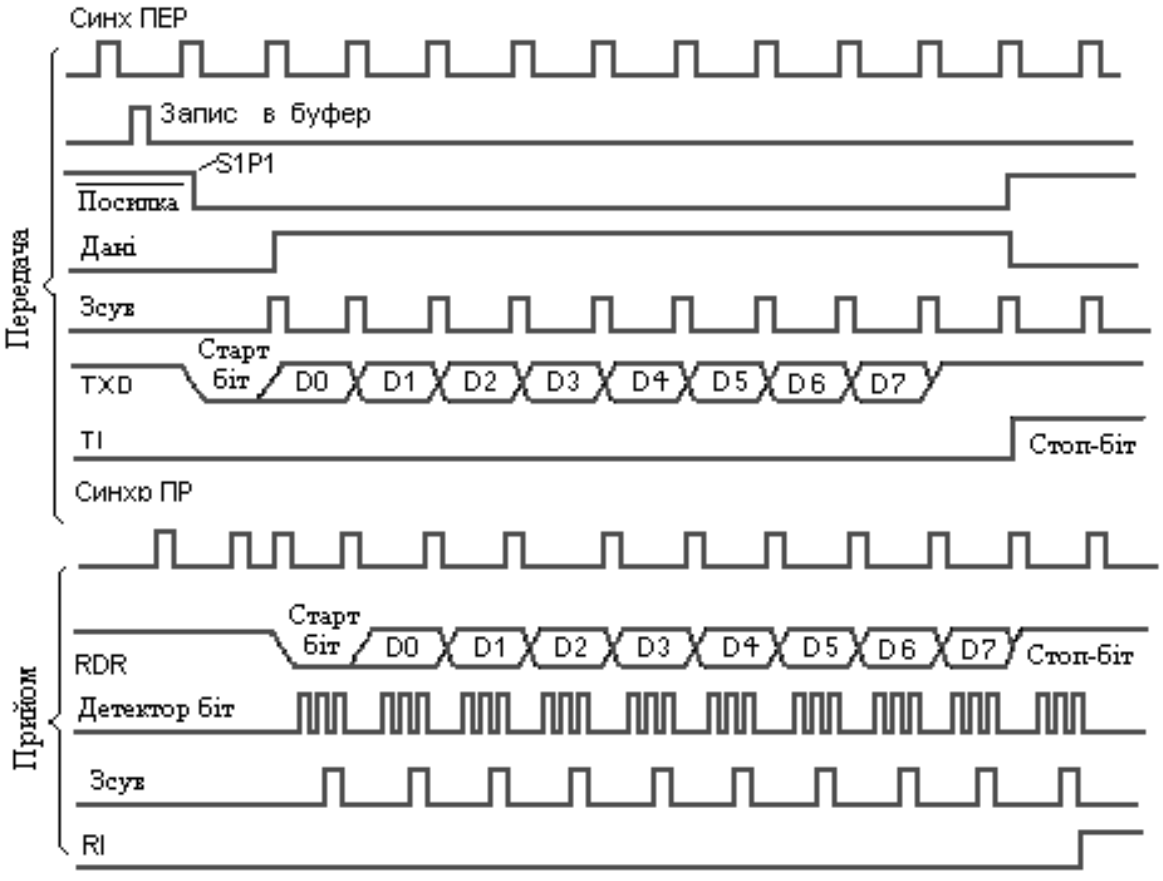


Рисунок 15. Часова діаграма роботи УАПП у режимі 1

Передавання ініціюється будь-якою командою запису в регістр SBUF. При цьому генерується сигнал «Запис у буфер», що завантажує «1» у дев'ятий біт зсувного регістра передавача. За цим сигналом на вивід TXD спочатку надходить старт-біт, а потім за дозволяючим сигналом – біти даних. Кожен період передавання біта дорівнює 16 тактам.

Прийом починається при виявленні переходу сигналу на вході RXD зі стану «1» у стан «0». Для цього під керуванням внутрішнього лічильника вхід RXD опитується 16 разів за період представлення біта. Як тільки перехід із «1» у «0» на вході RXD виявлено, в зсувний регістр приймача завантажується код 1FFH, внутрішній лічильник по модулю 16 негайно скидається і перезапускається для вирівнювання його переходів із границями періодів представлення прийнятих бітів. Таким чином, кожен період представлення біта ділиться на 16 періодів внутрішнього лічильника. У станах 7, 8 і 9 лічильника в кожному періоді представлення біта опитується сигнал на вході RXD. Зчитаним вважається значення, що було отримано двічі з трьох вимірів. Якщо значення, прийняте в першому такті не дорівнює «0», то блок керування прийомом знову повертається до пошуку переходу з «1» у «0». Цей механізм забезпечує видалення помилкових старт-бітів. Блок керування прийомом формує сигнал «Завантаження буфера», встановлює ознаки RB8 та RI лише в тому випадку, якщо в останньому такті зсуву виконуються дві умови:

1) біт RI = «0»;

2) або SM2 = «0», або прийнятий стоп-біт дорівнює «1».

Якщо одна з цих двох умов не виконується, то прийнята послідовність бітів втрачається. В цей час незалежно від того, виконуються вказані умови чи ні, блок керування прийомом знову починає відшукувати перехід із «1» у «0» на вході RXD.

1.7.5.3. Режими 2, 3.

Через вивід TXD УАПП передає або з виводу RXD приймає 11 бітів: старт-біт («0»), 8 бітів даних, програмований дев'ятий біт і стоп-біт («1»). На часовій

діаграмі (рис. 16) зображено роботу УАПП при передаванні та прийманні даних у режимах 2 і 3. Режими 2 і 3 відрізняються від режиму 1 тільки наявністю дев'ятого програмованого біта. Внаслідок цього змінюються умови закінчення циклу прийому: блок керування приймачем сформує керуючий сигнал «Завантаження буфера», завантажить RB8 і встановить ознаку RI тільки в тому випадку, якщо в останньому такті зсуву виконуються дві умови:

1) біт RI = «0»;

2) SM2 = «0», або значення прийнятого дев'ятого біта даних дорівнює «1».

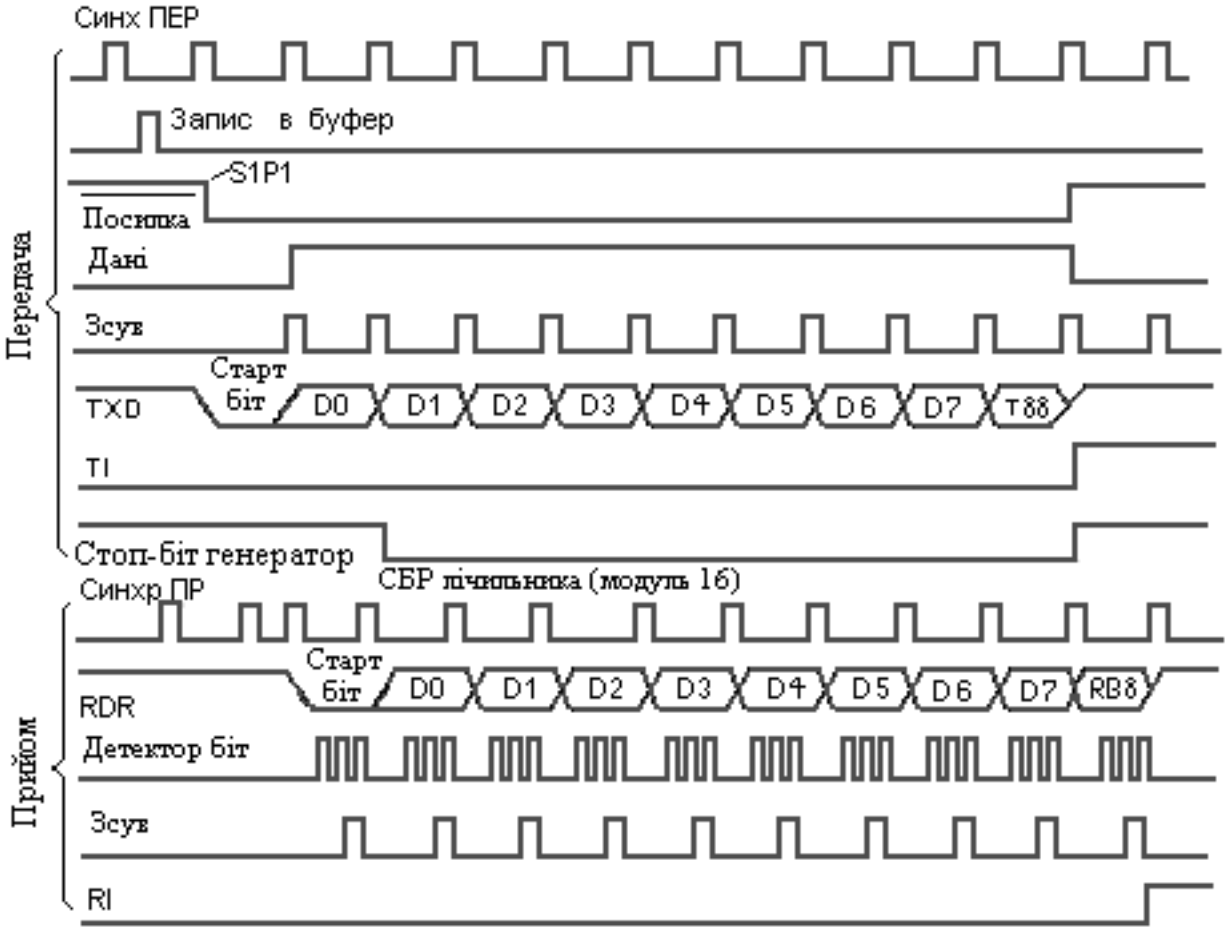


Рисунок 16. Часова діаграма роботи УАПП у режимах 2 і 3

1.8. Система переривань.

Спрощена схема переривань MCS-51 зображена на рис. 17. Зовнішні переривання INT0і INT1 можуть бути викликані або рівнем, або переходом сигналу з «1» у «0» на входах MCS-51 залежно від значень керуючих бітів IT0 і

IT1 у регістрі TCON. Від зовнішніх переривань встановлюються ознаки IE0 і IE1 у регістрі TCON, що ініціюють виклик відповідної підпрограми обслуговування переривання. Скидання цих ознак виконується апаратно тільки в тому випадку, якщо переривання викликане по переходу (зрізу) сигналу. Якщо ж переривання викликане рівнем вхідного сигналу, то скиданням прапорця IE керує відповідна підпрограма обслуговування переривання шляхом впливу на джерело переривання.

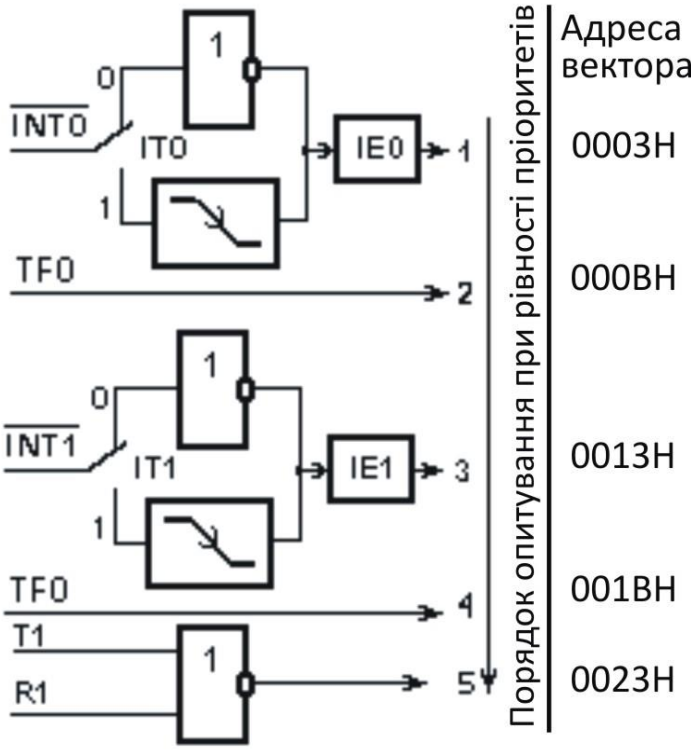


Рисунок 17. Схема переривань MCS-51

Ознаки запитів переривання від таймерів TF0 і TF1 скидаються автоматично при передаванні керування підпрограмі обслуговування. Ознаки запитів переривання RI і TI встановлюються блоком керування УАПП апаратно, але скидатися повинні програмою.

Переривання можуть бути дозволені або заборонені програмою за допомогою скидання розрядів регістра дозволів переривань EI. Формат регістра EI зображено на рис. 18, опис розрядів наведено в таблиці 9.

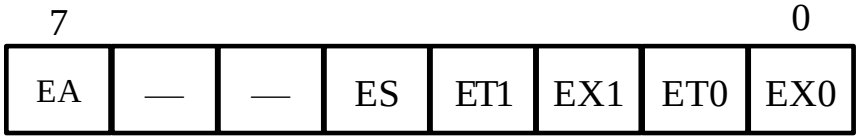


Рисунок 18. Формат регістра дозволу переривань EI

Таблиця 9. Регістр дозволу переривання EI

Символ	Ім’я і призначення
EA	Блокування переривань. Скидається програмно для заборони всіх переривань незалежно від станів IE4 – IE0
ES	Біт дозволу переривання від УАПП. Ознака встановлюється та скидається програмою для дозволу або заборони переривань від ознак TI або RI
ETx	Біт дозволу переривання від таймера x. Ознака встановлюється та скидається програмою для дозволу або заборони переривань від таймера
EXx	Біт дозволу зовнішнього переривання, що надходить на вхід INTx. Ознака встановлюється та скидається програмою для дозволу або заборони переривань від зовнішнього входу INTx

У випадку одночасного надходження переривань першим опрацювали те переривання, пріоритет котрого вищий, а при однаковому пріоритеті – за черговістю опитування.

У блоці регістрів спеціальних функцій є регістр пріоритету переривань IP, котрий вказує рівень пріоритету переривання. Його формат зображено на рис. 19, опис розрядів наведено в таблиці 10. Ознаки переривань опитуються в момент S5P2 кожного машинного циклу. Ранжування переривань за рівнем пріоритету виконується протягом наступного машинного циклу.

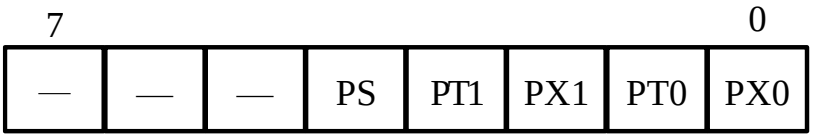


Рисунок 19. Формат регістра пріоритетів переривань IP

Таблиця 10. Регістр пріоритетів переривань IP

Символ	Ім’я і призначення
PS	Біт пріоритету УАПП. Одиниця вказує на високий пріоритет переривання УАПП, нуль – на низький
PTx	Біт пріоритету таймера x. Одиниця вказує на високий пріорітет переривання таймера, нуль – на низький
PXx	Біт пріоритету зовнішнього переривання (INTx). Одиниця вказує на високий пріоритет переривання таймера, нуль – на низький

Система переривань сформує апаратно виклик (LCALL) відповідної підпрограми обслуговування, якщо вона не заблокована однію із таких умов:

1) у даний момент обслуговується запит переривання рівного або вищого рівня пріоритету;

2) поточний машинний цикл – не останній у циклі команди, що виконується;

3) виконується команда RETI або будь-яка команда, пов’язана зі звертанням до регістрів IE або IP.

Якщо ознака переривання встановлена, але по одній з перерахованих вище умов не обслуговувалася і до моменту закінчення блокування вже була скинена, то запит переривання втрачається і ніде не запам’ятовується.

За апаратно-сформованим кодом LCALL система переривання заносить у стек тільки вміст лічильника команд (PC) і завантажує в лічильник команд адресу вектора відповідної підпрограми обслуговування. За адресою вектора повинна бути розташована команда безумовного переходу (JMP) до початкової адреси підпрограми обслуговування переривання. Підпрограма обслуговування у випадку необхідності повинна починатися командами запису в стек (PUSH) стану програми (PSW), акумулятора, розширювача, вказівника даних і т.д. й закінчуватися командами відновлення зі стека (POP) збережених даних. Підпрограми обслуговування переривання обов’язково завершуються командою RETI, за якою в лічильник команд перезавантажується зі стека збережена адреса повернення в основну програму. Команда RET також

повертає керування перерваній основній програмі, але при цьому не знімає блокування переривань, що призводить до необхідності мати програмний механізм аналізу закінчення процедури обслуговування даного переривання.

Переривання можуть бути викликані або заборонені програмою, а всі перераховані ознаки – доступні й можуть бути встановлені або скинені з тим же результатом, якби вони були встановлені або скинені апаратними засобами.

Наведемо приклад організації очікування тривалістю 50 мс за допомогою переривання. Вважаємо, що біт IE.7 встановлено.

```
                        ;організація переходу до мітки
                        ;NEXT при переповненні
                        ;таймера/лічильника 0
    ORG 0BH             ;адреса вектора преривання від
                        ;таймера/лічильника 0
    CLR TCON.4          ;зупинка таймера/лічільника 0
    RETI                ;вихід з підпрограми
                        ;опрацювання переривання
    ORG 100H            ;початкова адреса програми
    MOV TMOD, #01H      ;налаштування
                        ;таймера/лічильника 0
    MOV TLO, #LOW(NOT(5000-1))
                        ;завантаження таймера
    MOV THO, #HIGH(NOT(5000-1))
    SETB TCON.4         ;старт
    SETB IE.1           ;дозвіл переривання
    SETB PCON.0         ;перехід у режим
                        ;холостого ходу
NEXT:
      ...               ;подальший код котрий
      ...               ;буде виконано, коли
      ...               ;закінчиться переривання
```

Для вимірювання тривалості сигналу може використовуватися таймер. Особливо ефективним є його використання в MCS-51 із входом дозволу відліку (альтернативна функція входу INT). Вимірюваний сигнал можна, наприклад, подавати на вхід INT0. Вимірювання тривалості при цьому буде виконуватися в таймері/лічильнику 0. Програма вимірювання тривалості «додатнього» імпульсу буде мати вигляд:

```
  MOV TMOD, #00001001B   ;вибір режиму таймера
  MOV TH0, #0            ;скидання
  MOV TL0, #0
  SETB TCON.4            ;старт
WAITO:
  JNB P3.2, WAITO        ;чекання "1"
WAITC:
  JB P3.2, WAITC         ;чекання "0"
  CLR TCON.4             ;стоп таймера
EXIT:
  RET                    ;вихід з процедури
```

1.9. Проектування МП-систем на основі MSC-51. Приклади практичних схем на основі MSC-51.

1.9.1. Вимірювання температури термопарою.

Наступна схема дозволяє виміряти температуру об’єкта за допомогою термопари K-типу (Хромель-алюмелеві – ТХА), причому, діапазон вимірювання лежить у межах від 0 до 1024 °C.

Вимірювання температур за допомогою термопар (thermocouples) так само як, наприклад, за допомогою резистивних датчиків температури (RDT), найчастіше використовуються в промисловості та наукових дослідженнях.

У даному прикладі роль вимірювального перетворювача сигналу термопари виконує мікросхема MAX6675, цифровий сигнал з якої по інтерфейсу SPI (синхронний послідовний інтерфейс) передається в мікроконтролер, як показано на рис. 20.

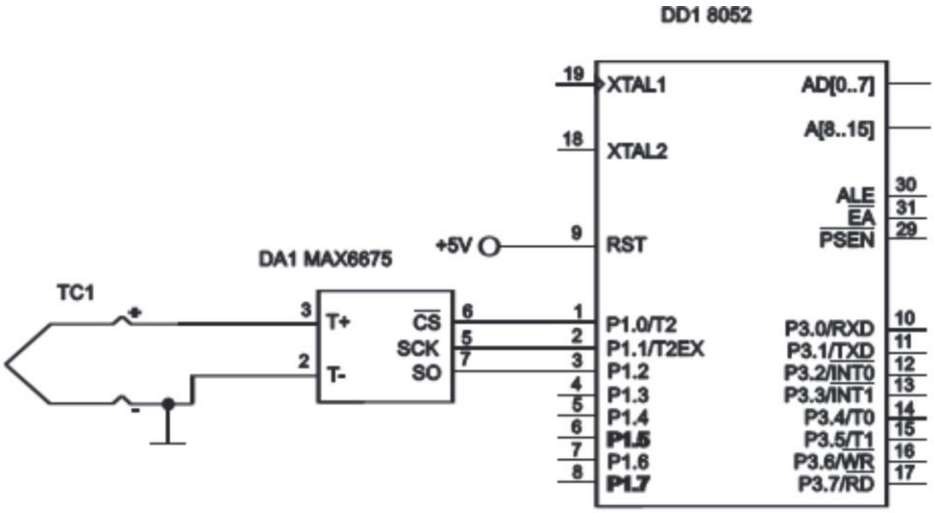


Рисунок 20. Вимірювання температури з використанням інтерфейсу SPI

Мікросхема MAX6675 здійснює повне первинне опрацювання сигналу термопари K-типу, виконуючи розрахунок компенсування ЕРС для холодного спаю і перетворення отриманого сигналу в цифрову форму. Далі 12-розрядний цифровий сигнал передається по SPI-сумісному інтерфейсу в мікроконтролер (8051, 8052 та ін.). Даний перетворювач дозволяє виміряти температуру до 1024 °С з похибкою 0,25 °С.

Мікросхема підключається до термопари наступним чином: вивід «Т +» з'єднується з хромелевим виводом термопари, а вивід «Т-» – з алюмелевими виводами термопари. Вивід SCK мікросхеми використовується для приймання синхронізуючих імпульсів, які ініціюють приймання даних по лінії SO за наростаючим фронтом. Сигнал CS, встановлений у низький рівень, дозволяє приймання даних по інтерфейсу. Сигнали SCK і CS є вхідними для мікросхеми, а SO – вихідним.

1.9.2. Вимірювання компонентів електричних схем через вимірювання частоти.

Вимірювання частоти часто використовується в промисловості і лабораторних дослідженнях при:

- аналізі сигналів віддалених давачів, переданих як послідовність імпульсів;
- для вимірювання величин аналогових сигналів, що надходять від віддалених об'єктів через перетворювачі «напруга – частота» (V to F conversion).

На вимірі частоти базується методика вимірювання значень активних компонентів електричних ланцюгів. В основному, для цих цілей використовуються генераторні схеми, в яких часозадаючі елементи є ємності й опір (RC-генератори для низьких частот) або індуктивності й ємності (LC-генератори для високих частот).

Наприклад, якщо у нас є схема генератора, керованого напругою (VCO, Voltage Controlled Oscillator), то з її допомогою можна виконати як вимірювання величини невідомого опору Rt при відомому значенні ємності Ct, так і навпаки, за відомим значенням Rt можна знайти невідому ємність Ct при фіксованих значеннях вхідної напруги Vin.

У загальному випадку для подібних вимірювань найчастіше використовуються генератори, керовані напругою, подібні тому, схема якого зображена на рис. 21.

Для такої й подібних їй схем, зафіксувавши два з трьох параметрів (Vin, Rt, Ct), по виміряному значенню частоти Fout схеми легко можна визначити або вхідну напругу (для перетворювача V to F), або опір чи ємність.

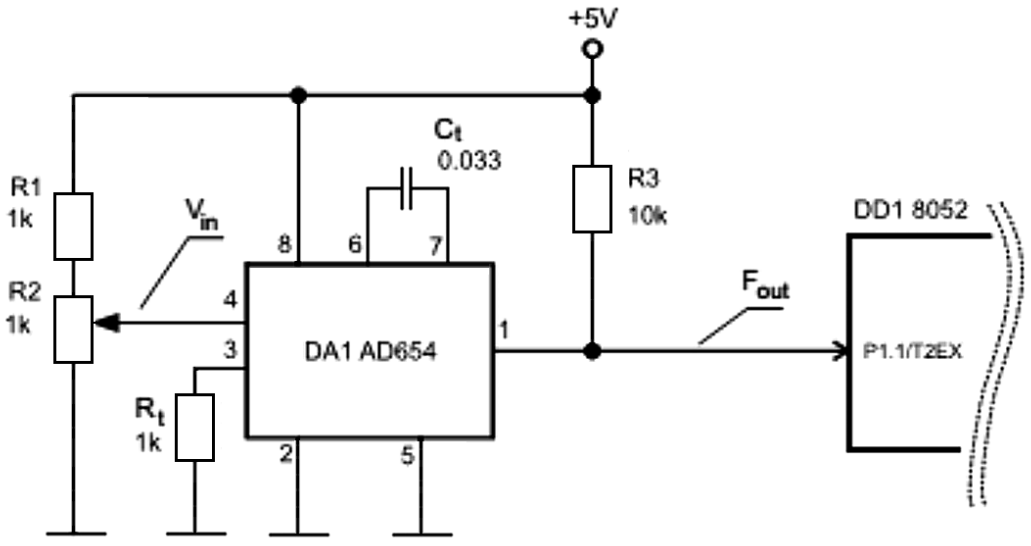


Рисунок 21. Вимірювання значень компонентів електричних схем

Сама електрична схема подібного перетворювача повинна забезпечувати високу точність перетворення за рахунок застосування спеціалізованих чіпів, як, наприклад, AD654 фірми Analog Devices або AD537 тієї ж фірми.

1.9.3. Керування швидкістю обертання двигуна постійного струму.

Мікроконтролери 8051/8052 дозволяють досить легко реалізувати ШІМ, причому це рішення засноване на застосуванні таймера. Багато сучасних 8051- сумісних пристроїв мають вбудовані функціональні вузли для реалізації ШІМ.

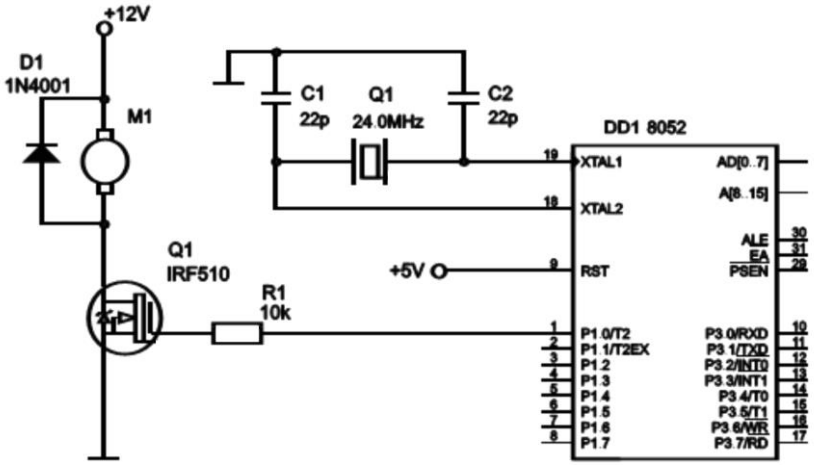


Рисунок 22. Реалізація ШІМ у системі з МК 8051

Схема, що ілюструє застосування ШІМ для керування електродвигуном постійного струму, зображена на рис. 22.

У цій схемі електродвигун M1 керується через потужний польовий транзистор (Power MOSFET) типу IRF510, на затвор якого подаються імпульси управління з виводу P1.0 мікроконтролера. Апаратна частина розроблена для тактової частоти 24,0 МГц, тому для інших значень частоти слід змінити параметри налаштування таймера 2, який використовується для генерації послідовності імпульсів.

Замість транзистора Q1 типу IRF510 можна використовувати будь-який інший N-канальний зі схожимими характеристиками, а в якості електродвигуна M1 може бути будь-який малопотужний двигун постійного струму.

1.9.4. Ввід даних від цифрових датчиків.

Апаратна частина схемотехнічно може бути реалізована так, як показано на рис. 23. Схема дозволяє опрацьовувати 8 сигналів від цифрових джерел по одній лінії, використовуючи мультиплексування.

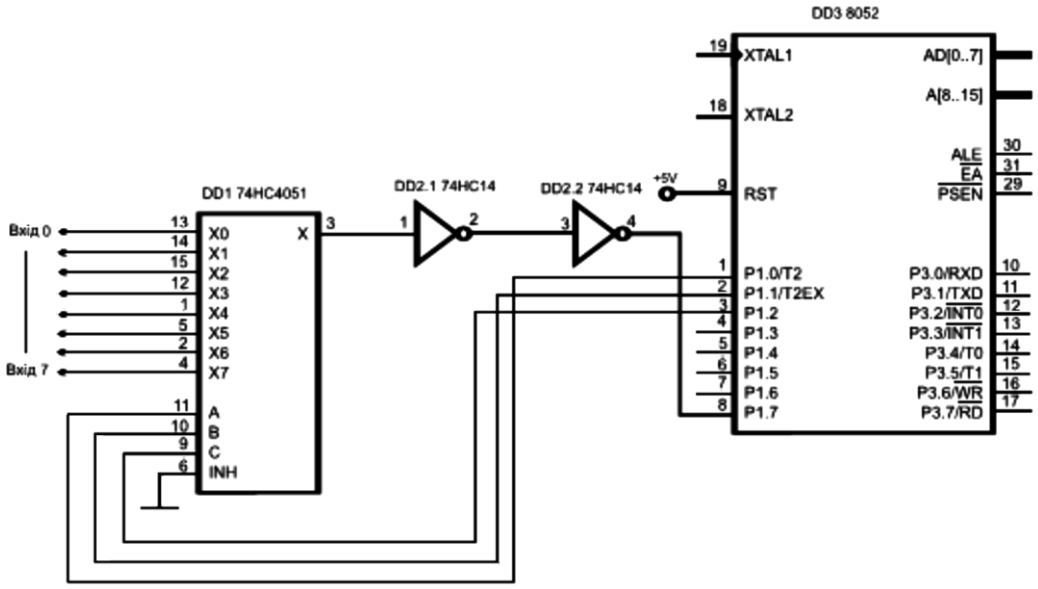


Рисунок 23. Мультиплексування вводу дискретних даних

У схемі для розширення кількості опрацьованих вхідних сигналів застосована мікросхема 74HC4051, що являє собою 8-входовий мультиплексор сигналів. Залежно від двійкового коду на входах A, B та C, на виході X з'являється один із вхідних сигналів. Вхід INH мікросхеми при низькому рівні напруги дозволяє роботу мультиплексора, тому він заземлений. Тригер Шмідта використовується для формування сигналу, що надходить на вхід P1.7 мікроконтролера.

1.9.5. Перетворювання паралельного коду в послідовний.

На рис. 24 представлена апаратна реалізація опрацювання вхідних сигналів через перетворення паралельного коду в послідовний.

Для зчитування вхідних даних у цій схемі використовується регістр зсуву на мікросхемі 74HC166, що дозволяє перетворити паралельний код на входах 0…7 в послідовний на виводі SO. Послідовні дані на цьому виході зчитуються мікроконтролером і перетворюються програмою в паралельний формат.

Використовуваний метод частково уповільнює реакцію на вхідні сигнали в порівнянні з паралельним зчитуванням даних у порт мікроконтролера по 8 лініях.

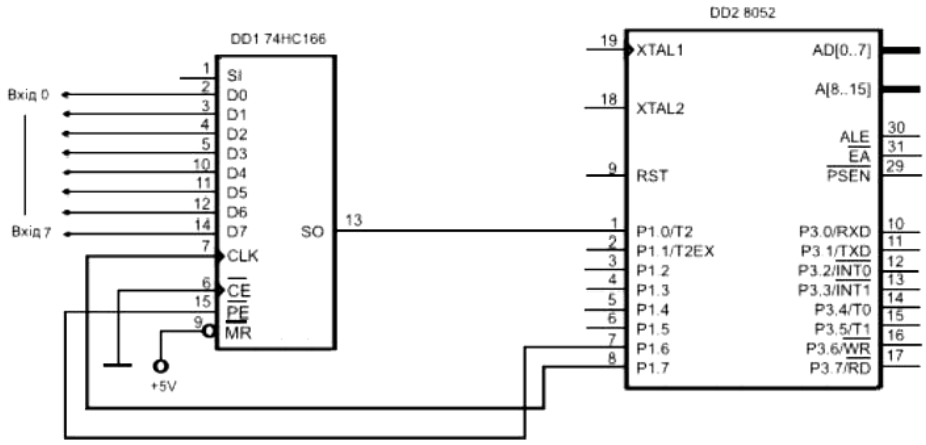


Рисунок 24. Перетворення паралельних вхідних даних у послідовний код

Схема працює наступним чином. Дані зі входів D0-D7 завантажуються в паралельному форматі в регістр DD1 за низьким рівнем сигналу на виводі PE мікросхеми. Потім по фронту синхроімпульсу, що надходить на вхід CLK, дані побітово зсуваються на вихід SO, де зчитуються мікроконтролером.

Замість мікросхеми 74HC166 можна застосувати аналогічні чіпи, при цьому необхідно враховувати швидкодію конкретної мікросхеми й використовувати, де потрібно, програмні затримки.

1.9.6. Вимірювання аналогового сигналу за допомогою МК51.

Розглянемо завдання, в якому необхідно виміряти величину зовнішнього аналогового сигналу, що знаходиться в діапазоні 0…5 В, і вивести його значення через послідовний порт на термінальний пристрій або дисплей персонального комп’ютера.

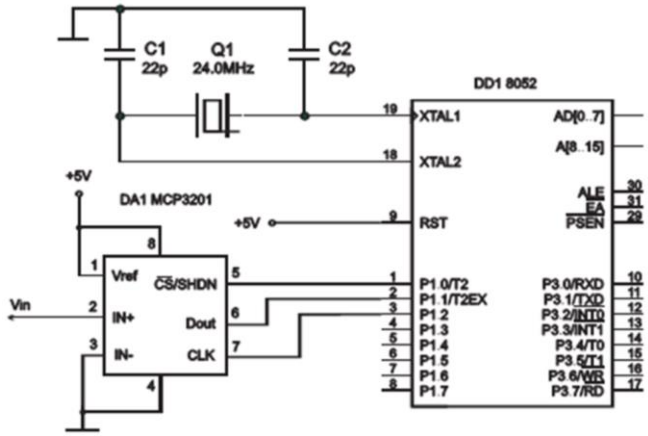


Рисунок 25. Схема опрацювання аналогового сигналу

Для цього слід під'єднати ПК через послідовний порт до мікроконтролера, потім запустити на ПК програму, що працює з послідовним портом, і виконати її.

Принципова схема пристрою зображена на рис. 25.

АЦП (DA1) приєднаний до мікроконтролера таким чином:

- вхід дозволу перетворення CS/SHDN з'єднується з виводом P1.0 МК;
- вхід синхронізації CLK з'єднується з виводом P1.2 МК;
- вихід даних Dout з'єднується з виводом P1.1 мікроконтролера.

Програма повинна зчитувати вхідний сигнал Vin на вході АЦП і відображати його значення на екрані кожні 10 сек.

1.9.7. Вимірювання аналогових сигналів від кількох джерел.

У практичних розробках не часто доводиться вимірювати аналогові сигнали, що надходять від одного джерела. Схема проекту, який дозволяє одночасно вимірювати аналогові сигнали з 8-ми джерел, містить мікросхему LTC1292 і мультиплексор аналогових сигналів 4051, представлена на рис. 26.

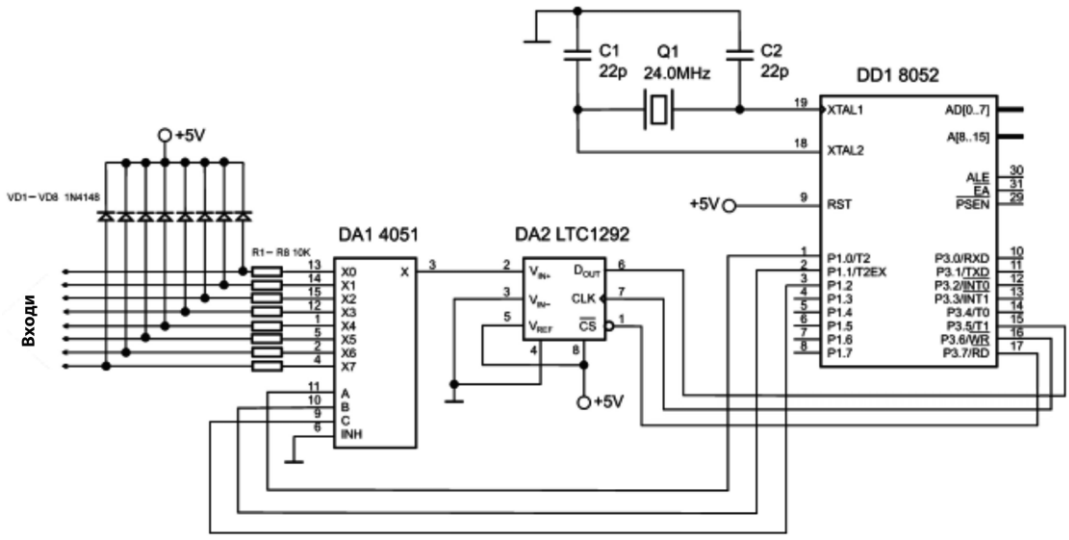


Рисунок 26. Схема 8-канального АЦП на LTC1292

У цій схемі аналогові вхідні сигнали подаються на вісім входів мультиплексора (мікросхема DA1). Мікроконтролер через певні інтервали часу, що задаються таймером 2, сканує входи і після перетворення посилає дані на термінал. Для підвищення точності перетворення можна зменшити номінали резисторів R1R8.

У багатьох випадках можна обійтися без обмежуючих схем на резисторах і діодах, якщо заздалегідь відомо, що вхідні напруги не перевищують 5 В.

1.9.8. Формування аналогового сигналу пристроєм на основі MCS-51

Розглянемо пристрій, в якому вихідний аналоговий сигнал генерується за допомогою мікросхеми LTC1456, що є однокристальним 12-розрядним ЦАП фірми Linear Technology. Цей перетворювач керується 12-розрядним двійковим кодом, що надходить через модифікований варіант SPI інтерфейсу.

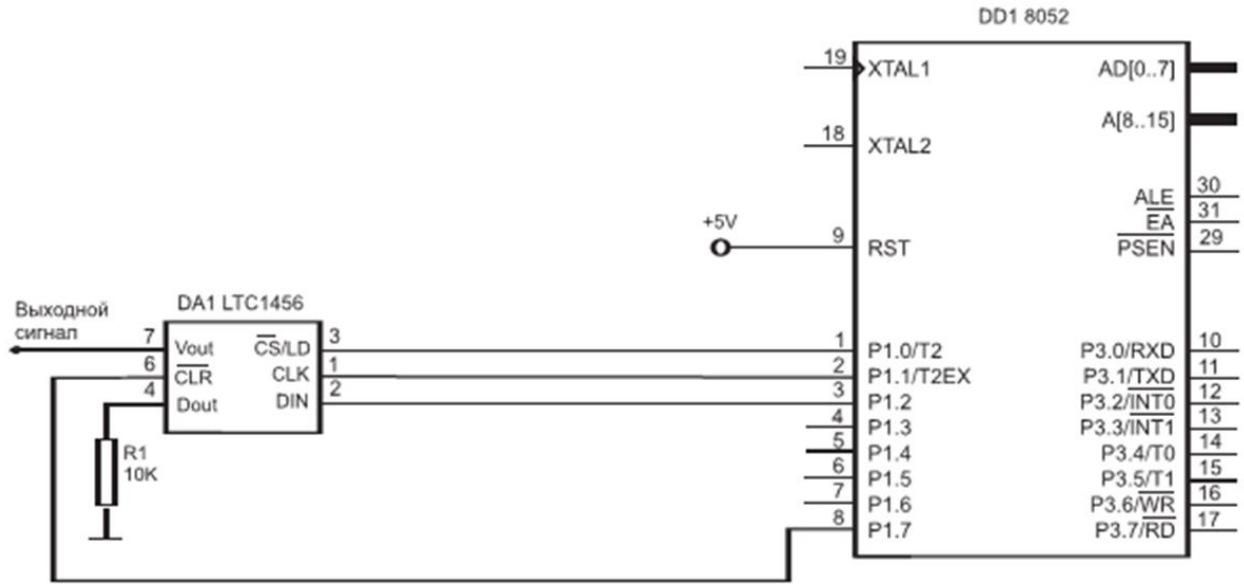


Рисунок 27. Генерація вихідного аналогового сигналу з використанням ЦАП LTC1456

Виводи мікросхеми мають такі призначення:

- CLK – строб синхронізації при прийманні даних;
- Din – вивід, на який надходять дані. Дані на цьому виводі запам’ятовуються в регістрі мікросхеми по наростаючому фронту сигналу CLK;
- CS/LD – сигнал дозволу/встановлення інтерфейсу. При низькому рівні сигналу на вході дозволяється прийом даних по лінії Din, при високому рівні прийом даних по лінії Din забороняється, а прийняті дані завантажуються з регістра зсуву в регістр перетворення, змінюючи значення вихідної напруги перетворювача.

Принципова схема ЦАП, керованого мікроконтролером, показана на рис. 27.

Програмне забезпечення для цього випадку дозволяє ввести з клавіатури терміналу або ПК необхідне значення вихідної напруги. Власне, перетворення цифрового коду в аналогову напругу виконує програма мікроконтролера.

1.10. Приклади програм для роботи з мікроконтролером МК51.

Приклад 1. Записати в резидентну пам'ять даних за адресами 41 та 42 число 1C3Fh:

```
LOAD:
     MOV R0,#41h   ;завантаження в R0 покажчика даних
     MOV @R0,#1Ch  ;завантаження в пам'ять числа 1CH
     INC R0        ;інкремент покажчика
     MOV @R0,#3Fh  ;записати в пам'ять число 3FH
START:
     LJMP LOAD1    ;перехід до програми LOAD
     ORG 2100h     ;директива розташування програми
                   ;за адресою 2100
LOAD1:
     MOV R0,#41h   ;завантаження в R0 покажчика даних
     MOV @R0,#1Ch  ;записати в пам'ять число 1CH
     INC R0        ;інкремент покажчика
     MOV @R0,#3Fh  ;записати в пам'ять число 3FH
     LJMP LOAD     ;зациклювання програми
END                ;директива закінчення трансляції
```

Приклад 2. Додавання. Додати два двійкові багатобайтні числа. Доданки розташовуються в резидентній пам'яті даних, починаючи з молодшого байта. Початкові адреси доданків задані в R0 і R1, формат доданків у байтах – в R2:

```
    CLR C          ;скидання ознаки перенесення
  LOOP:
    MOV A,@R0      ;завантаження в А поточного
                   ;байту першого доданка
    ADDC A,@R1     ;додавання байтів з врахуванням
                   ;перенесення
    MOV @R0,A      ;розміщення байта результату
    INC R0         ;просування покажчиків
    INC R1
    DJNZ R2,LOOP ;цикл, якщо не всі байти
                   ;просумовані
```

Приклад 3. Потрібно помножити ціле двійкове число довільного формату на константу 73. Початкове число розташовується в резидентній пам’яті даних, адреса молодшого байту знаходиться в регістрі R0. Формат числа в байтах зберігається в R1:

```
        MOV A,#0       ;скидання акумулятора
 LOOP:
        ADD A,@R0      ;завантаження множеного
        MOV B,#73      ;завантаження множника
        MUL AB         ;множення
        MOV @R0,A      ;запис молодшого байта
                       ;часткового добутка
        INC R0         ;інкремент адреси
        MOV A,B        ;пересилання старшого байта
                       ;часткового добутка в акумулятор
        XCH A,@R0      ;попереднє формування
                       ;наступного байта добутка
        DJNZ R1,LOOP ;цикл, якщо не всі байти
                       ;початкового числа помножені
                       ;на константу
```

Приклад 4. Операції зі стеком:

```
START:
      MOV R1,#02h  ;завантаження регістрів
      MOV A,#30h
      MOV R2,#00h
      LCALL SUB    ;перехід на підпрограму
      SJMP START
SUB:
      PUSH PSW     ;зберігання в стеку вмісту PSW
      PUSH ACC     ;зберігання акумулятора
      PUSH B       ;зберігання вмісту B в
                   ;акумуляторі
      ADD A,R1     ;опрацювання даних
      MOV R2,A
      POP B        ;відновлення B
      POP ACC      ;відновлення акумулятора
      POP PSW      ;відновлення PSW
      RET          ;повернення
 END
```

Приклад 5. Виконання логічних команд:

```
OUT:
    MOV P1,#10101010b   ;вивід байта в порт P1
    SETB P1.0           ;встановлення в «1»
                        ;нульового розряду
    CLR P1.3            ;скидання в «0» 3-го розряду
    CLR P1.4            ;скидання в «0» 4-го розряду
    CLR P1.5            ;скидання в «0» 5-го розряду
    CLR P1.6            ;скидання в «0» 6-го розряду
    CLR P1.7            ;скидання в «0» 7-го розряду
    XRL P1,#11110000b   ;інверсія розрядів 4–7
    ORL P1,#00001000b   ;встановлення в «1» третього
                        ;розряду
    ANL P1,#11111110B   ;скидання в «0» нульового
                        ;розряду
```

1.11. Програмування MCS-51 із використанням програмної моделі EDSim51. Структура програмної моделі.

1.11.1. Призначення моделі й робота з програмою.

Програма призначена для моделювання роботи мікроконтролера 8051, розроблення та відлагодження програмного забезпечення для мікроконтролерних систем із периферійними пристроями.

Загальний вигляд інтерфейсу програмної моделі EDSim51 мікроконтролера MCS-51 представлена на рис. 28.

Розглянемо використання опцій емулятора.

1.11.2. Синтаксичне підсвічування.

Асемблерний код, написаний в EdSim51, автоматично підсвічується відповідно до синтаксису.

```
ORG 30H
main:
SET IT0         ;set external 0 interrupt as edge
SET EX0         ;enable external 0 interrupt
MOV TMOD,#02    ;set timer 0 as 8 bit auto-reload
MOV TH0,#(-200);put -200 into timer 0
```

Причому команди зафарбовані синім, директиви – фіолетовим, умовні імена – оранжевим кольором, і коментарі – зеленим. Якщо потрібно вимкнути синтаксичне підсвічування, необхідно клацнути правою кнопкою миші де завгодно у вікні коду й змінити опцію.

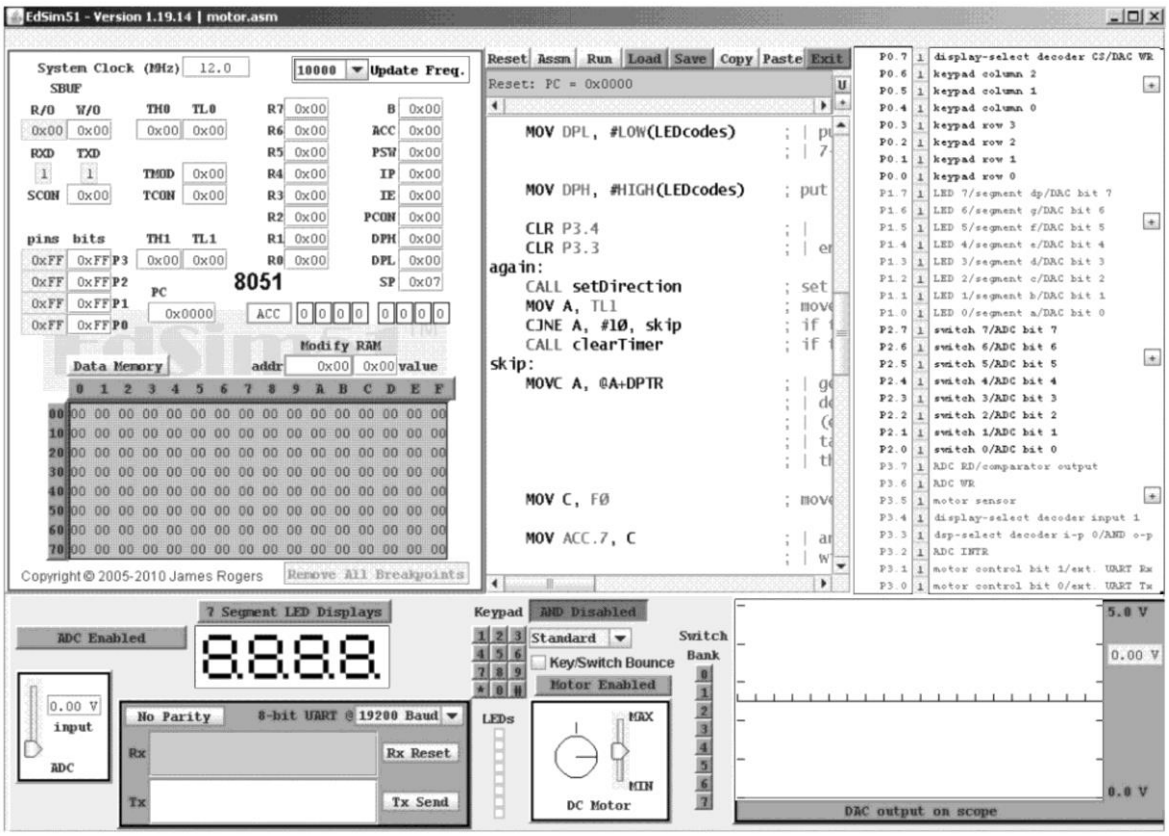


Рисунок 28. Інтерфейс програмного емулятора EdSim51

1.11.3. Зміна тактової частоти системи.

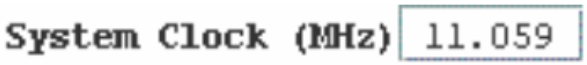


Рисунок 29. Зміна тактової частоти системи

Користувач може задавати значення тактової частоти в МГц. Допустимі значення знаходяться у діапазоні від 0,001 МГц до 999 МГц включно.

1.11.4. РК-модуль.

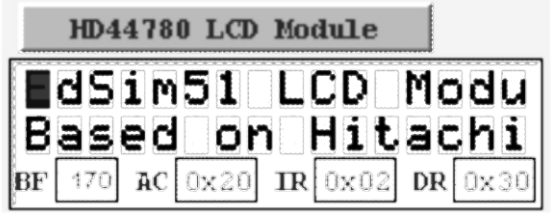


Рисунок 30. РК-модуль

В EdSim51 Simulator реалізовано емулятор модуля Hitachi HD44780 LCD. Користувач може перемикатися між світлодіодними індикаторами та РК дисплеєм, натиснувши на синю кнопку над дисплеєм.

1.11.5. Зміна масштабу.

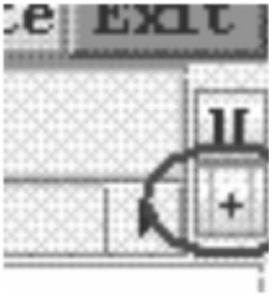


Рисунок 31. Зміна масштабу

Для моніторів високої роздільної здатності необхідно натиснути на кнопку zoom, яка знаходиться трохи нижче кнопки Exit.

1.11.6. Режими клавіатури.

Користувач може вибрати один з трьох режимів роботи:

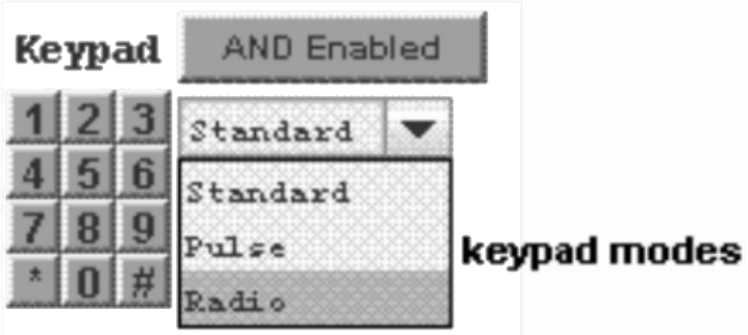


Рисунок 32. Режими клавіатури

Standard – одночасно можна натиснути кілька клавіш.

Pulse – після відпускання кнопки миші клавіша повертається у вихідне положення.

Radio – може бути зафіксована лише одна клавіша.

1.11.7. Оновлення вікна емулятора.

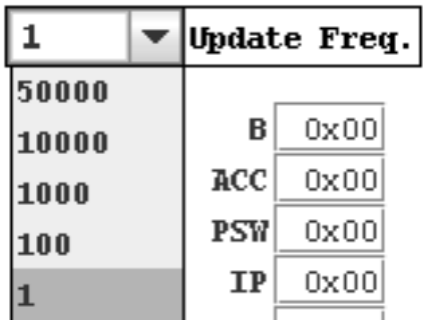


Рисунок 33. Оновлення вікна емулятора

EdSim51 дозволяє користувачеві або виконувати програму покроково (виконання однієї команди за один крок), або запустити програму у безперервне виконання. Паузи у проходженні дозволяють користовачу спостерігати за змінами в апаратній частині й регістрах. Користувач може обирати (показано на рисунку), як часто буде перезавантажуватися екран.

Варіанти:

- оновлення екрану після кожного виконання команди (за замовчуванням),
- після 100, 1000, 10000 або 50000 виконаних команд.

1.11.8. Панель мікроконтролера.

Надає користувачеві доступ до регістрів і пам’яті даних мікроконтролера.

Поля, зафарбовані у білий колір, можуть редагуватися в процесі роботи. В сірий – не можуть редагуватися.

Наприклад, біти портів можуть бути змінені користувачем, але контакти порту керуються зовнішніми пристроями, тому не піддаються редагуванню. Крім того, не можна змінювати значення програмного лічильника PC.

Є можливість переглянути адресу, розташувавши курсор миші над позначенням регістру, як це показано на малюнку для регістра PCON.

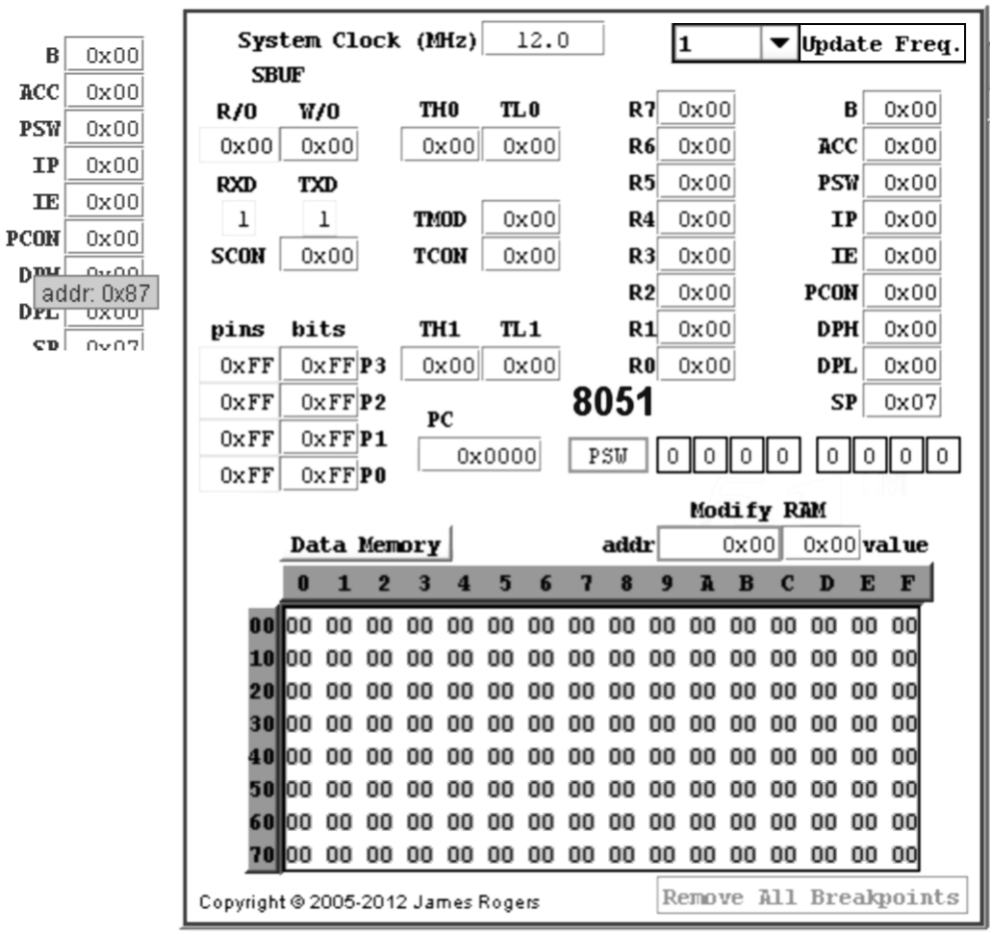
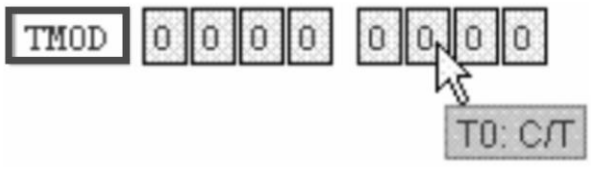


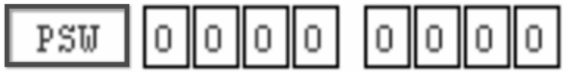
Рисунок 34. Панель мікроконтролера

1.11.9. Бітове поле.

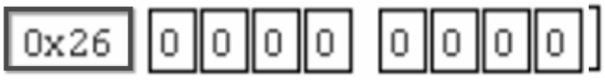
В попередньому пункті відображені окремі біти акумулятора. Користувач може ввести будь-яку адресу або ім’я у полі синього кольору (замінивши ACC) і працювати з бітами за вказаною адресою. Крім того, якщо навести курсор на один з бітів, буде відображено опис біта, як показано на нижче:



це зображення регістра TMOD, другий біт якого – це біт таймера/лічильника. Фон сірий, бо регістр TMOD не адресується за бітами і користувач не може змінити його значення безпосередньо;



регістр PSW дозволяє адресацію за бітами, тому користувач може редагувати значення бітів цього регістра;



бітове поле може бути використане також, щоб побачити біти за будь-якою адресою в оперативній пам’яті (від 0 до 7Fh), ввівши цю адресу в рядку синього кольору. Якщо ця область дозволяє бітову адресацію, поле є білим, і користувач може змінювати його значення.

Таким чином, якщо область не адресується за бітами, поле буде мати сіре забарвлення, як на цьому рисунку.

1.11.10. Пам’ять даних та програмна пам’ять.

За замовчуванням відображається область пам’яті даних. Будь-яка адреса в оперативній пам’яті (від 00H до 7Fh) може бути змінена шляхом введення адреси в рядку синього кольору (addr), а далі – введенням потрібного значення в полі праворуч (value). Пам’ять програми теж може бути відображена та відредагована, як показано на рисунку нижче. Для перемикання між даними та програмою треба натиснути кнопку, позначену як «DataMemory/CodeMemory».

Відображаються перші 127 байт програмної пам’яті. Щоб переглянути вміст інших областей, необхідно ввести початкову адресу в поле синього кольору. Значення змінюється таким самим чином, як і для пам’яті даних.

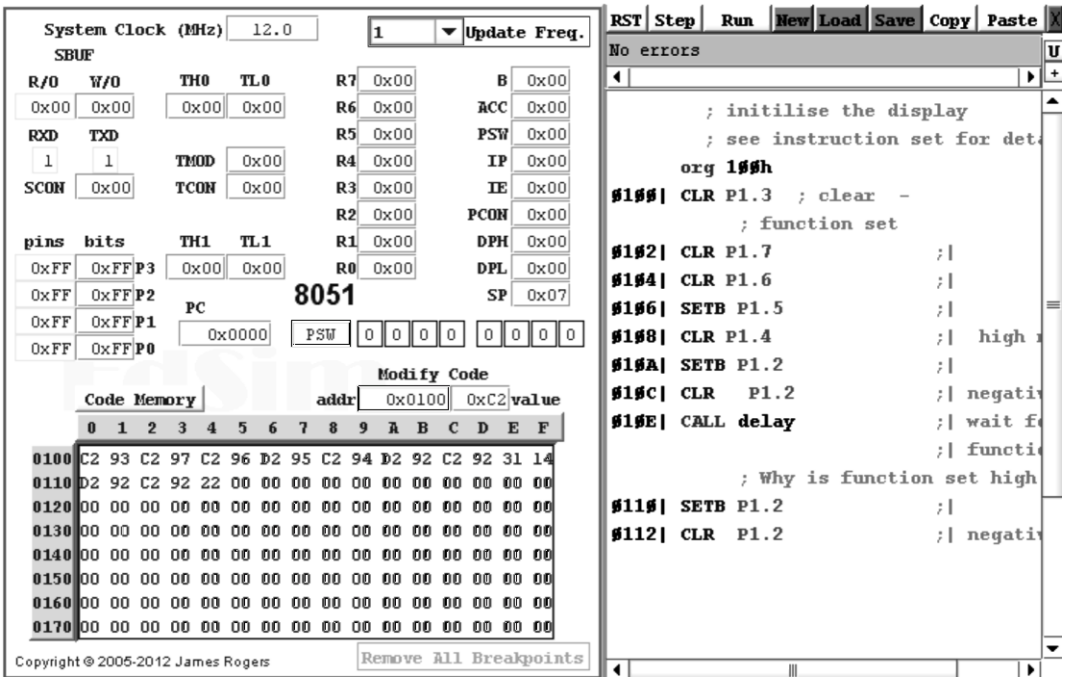


Рисунок 35. Пам’ять даних та програмна пам’ять

Такі зміни відобразяться у машинному коді. На зображенні, наведеному вище, змінений машинний код можна побачити у правій частині.

1.11.11. Панель коду Assembler.

Якщо фон тексту білого кольору, користувач може змінювати програму. Існує можливість вписувати код безпосередньо у це вікно, або ж завантажувати його з файлу після натискання на кнопку Load.

Коли програма готова для тестування, користувач може натиснути на кнопку Step для почергового виконання команд або на кнопку Run для запуску програми у безперервне виконання.

У будь-якому випадку програма спочатку буде скомпільована (assembled). Якщо буде виявлена помилка у коді, з’явиться повідомлення у верхньому полі (з червоним фоном), і рядок, що містить помилку, буде підсвічено червоним.

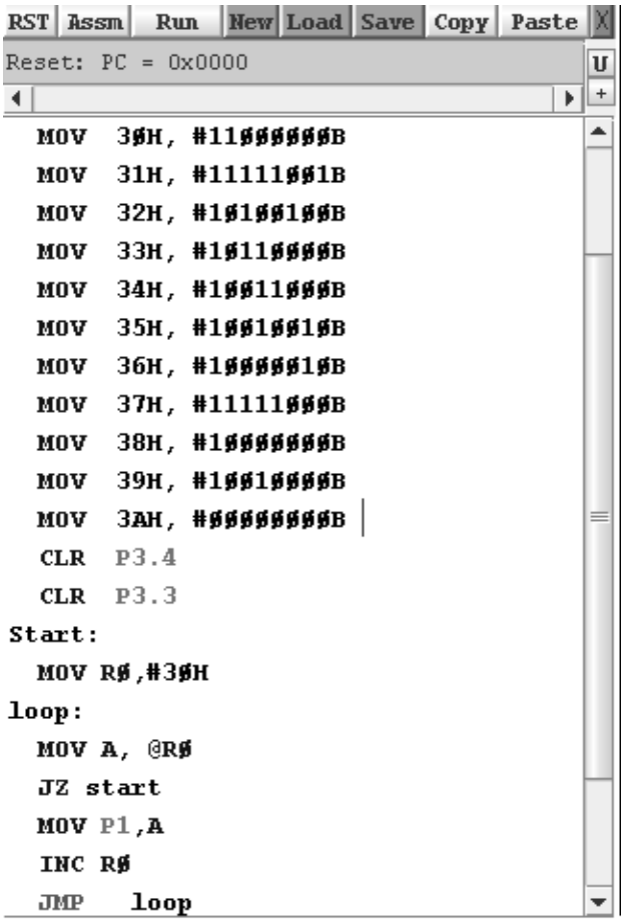


Рисунок 36. Панель коду Assembler

Якщо код асемблюється без помилок, фон текстового поля зміниться на світло-сірий. Тепер програма не може бути відредагована. Якщо ж потрібно повернутися до коду для внесення змін, потрібно натиснути Reset.

1.11.12. Завантаження та збереження.

Програма працює з двома типами файлів.

Перший – текстовий. Програми, написані мовою assembler, зберігаються як звичайні текстові файли, і, зазвичай, мають розширення .asm. За замовчуванням цей формат використовується для вихідного коду.

Інший формат файлів – IntelHEX. Користувач може зберігати дані у цьому форматі, як показано на малюнку нижче.

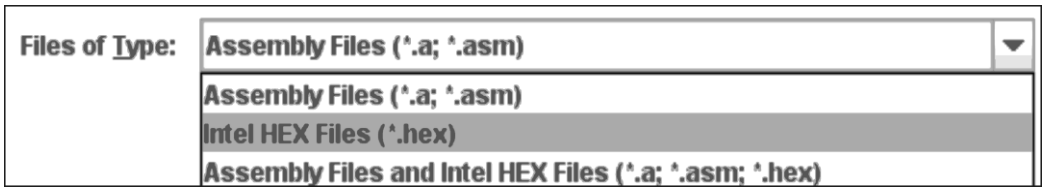


Рисунок 37. Завантаження та збереження

Щоб забезпечити зручніше користування програмою, запам’ятовується останній каталог, до якого звертався користувач (при завантаженні чи збереженні файлу). Таким чином, надалі, коли користувач відкриває діалогове вікно для завантаження або збереження файлу, він відразу переходить до останньої відвіданої директорії.

1.11.13. Копіювання та вставляння.

Можна вибрати фрагмент коду в області тексту та скопіювати його до буферу обміну за допомогою кнопки Copy.

Фрагмент може бути вставлений в інше місце текстової області з використанням кнопки Paste. Також можна зробити це в іншу програму – наприклад, в текстовий редактор, і навпаки – скопіювати текст в іншій програмі та вставити його до текстової області EdSim51.

1.11.14. Налагодження.

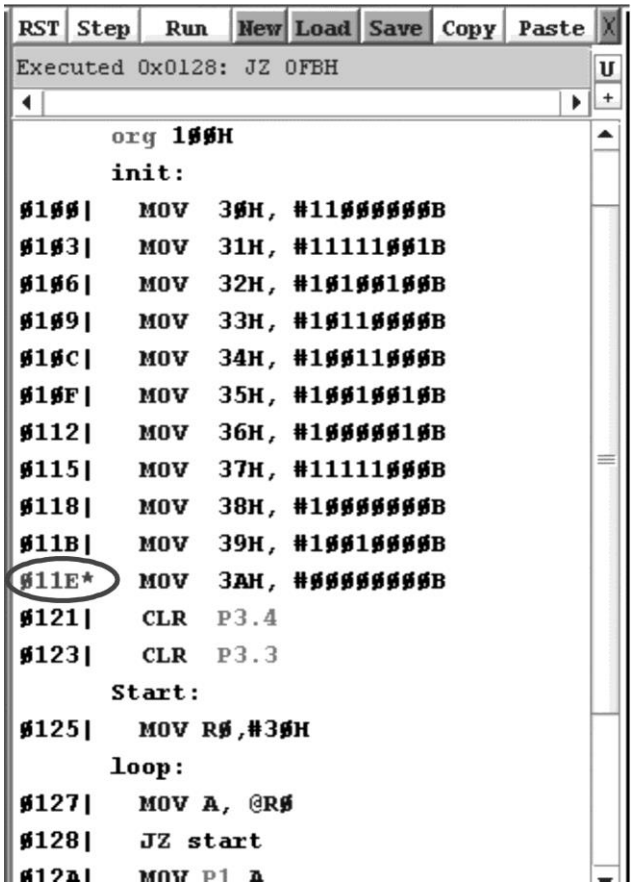


Рисунок 38. Налагодження

Незалежно від того, запущено програму в безперервне виконання чи покрокове, якщо код було скомпільовано без помилок, адреса кожної команди відображатиметься ліворуч.

Під час покрокового виконання коду команда, що виконується, відображається у верхньому сірому полі разом зі своєю адресою (на рисунку це рядок

```
Executed 0x002A: MOV 90H, A).
```

Підсвічується (червоним) адреса наступної для виконання команди (002C).

1.11.15. Точки зупинки.

Точку зупинки можна встановити, двічі клацнувши на адресу команди.

Встановлення точки зупинки: у режимі покрокового виконання програми необхідно розташувати курсор над адресою команди та клацнути двічі. Якщо

точка зупинки встановиться, вертикальна лінія (|) праворуч від адреси буде негайно змінена на зірочку (*), як показано на рисунку праворуч.

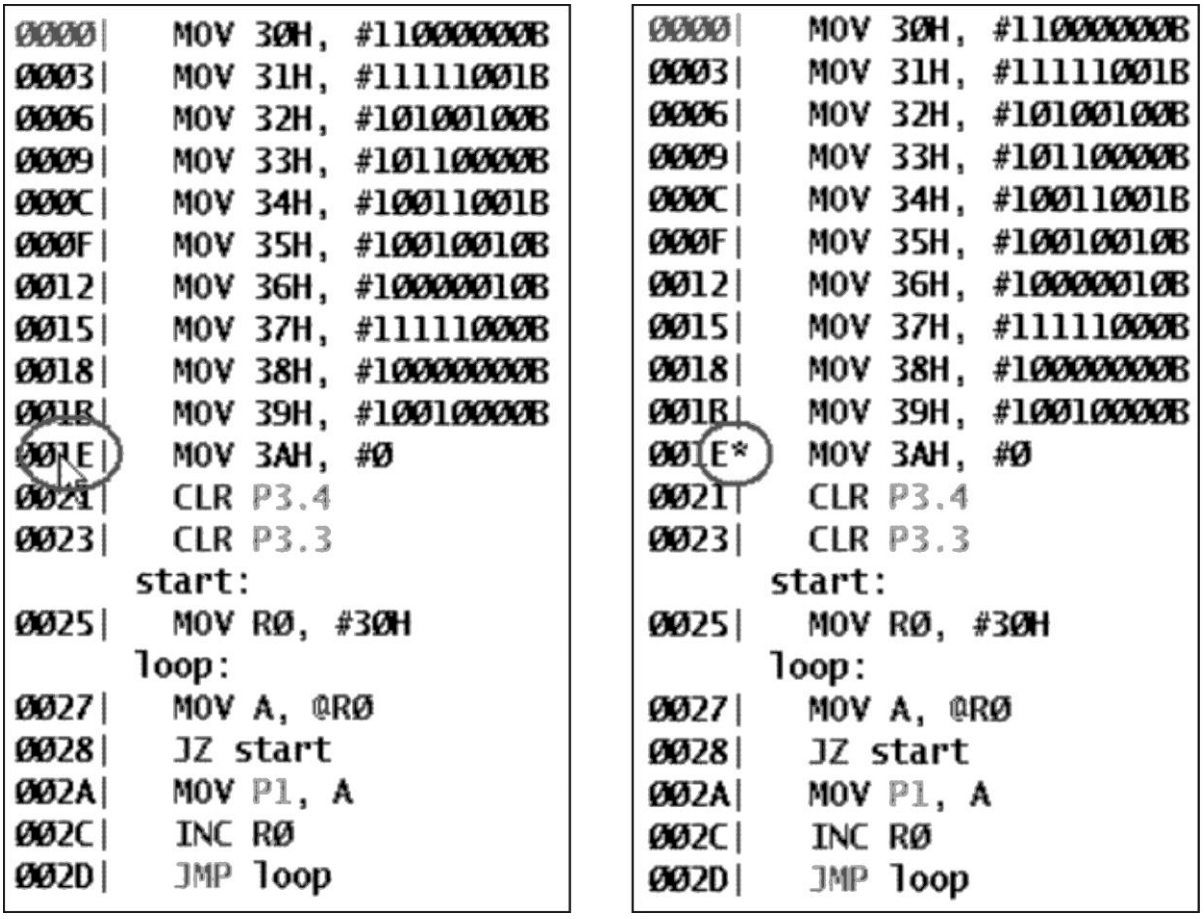


Рисунок 39. Точки зупинки

Видалення точки зупинки: точка зупинки позначається зірочкою (*). Щоб видалити її, потрібно розташувати курсор над адресою команди та двічі клацнути мишею. Зірочка (*) заміниться вертикальною лінією (|). Крім того, можна видалити всі точки зупинки разом, вибравши опцію "Remove All Breakpoints".

Коли програма виконується і зустрічається точка зупинки, виконання зупиняється перед цією інструкцією. Іншими словами, наступною для виконання буде та команда, на якій встановлена точка зупинки.

Користувач може продовжити покрокове проходження чи запустити програму для виконання, починаючи з цієї точки.

1.11.16. Периферійні засоби.

Нижче зображена панель периферії з активним АЦП та 7-сегментними індикаторами (рис. 40 а), а також панель периферії з активним компаратором та РК-модулем (рис. 40 б).

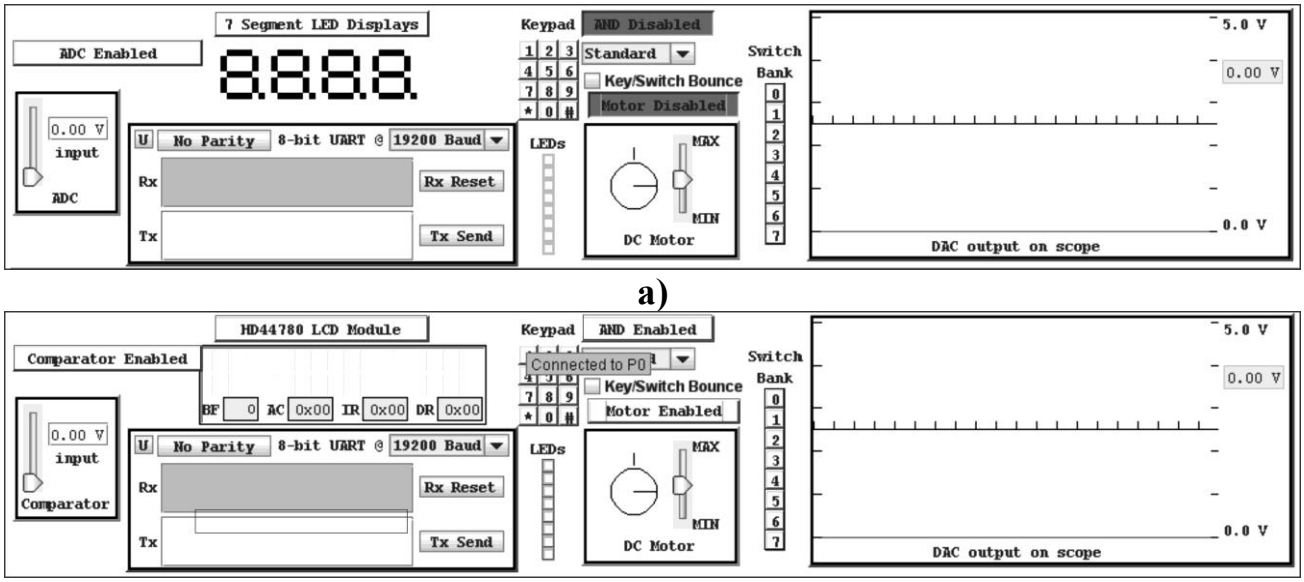


Рисунок 40. Периферійні засоби

Перелік периферійних пристроїв:

- АЦП,
- компаратор,
- чотири 7-сегментні світлодіодні індикатори,
- РК-модуль,
- UART (універсальний асинхронний приймач/передавач),
- клавіатура,
- світлодіодна матриця,
- DC реверсивний двигун,
- набір перемикачів,
- ЦАП (вихідний сигнал відображається на осцилографі).

Нижче наведені зображення показують, як під’єднані кожен з 32 виводів портів. Рисунок ліворуч: з активним 7-сегментним індикатором. Рисунок праворуч – з активним РК-модулем.

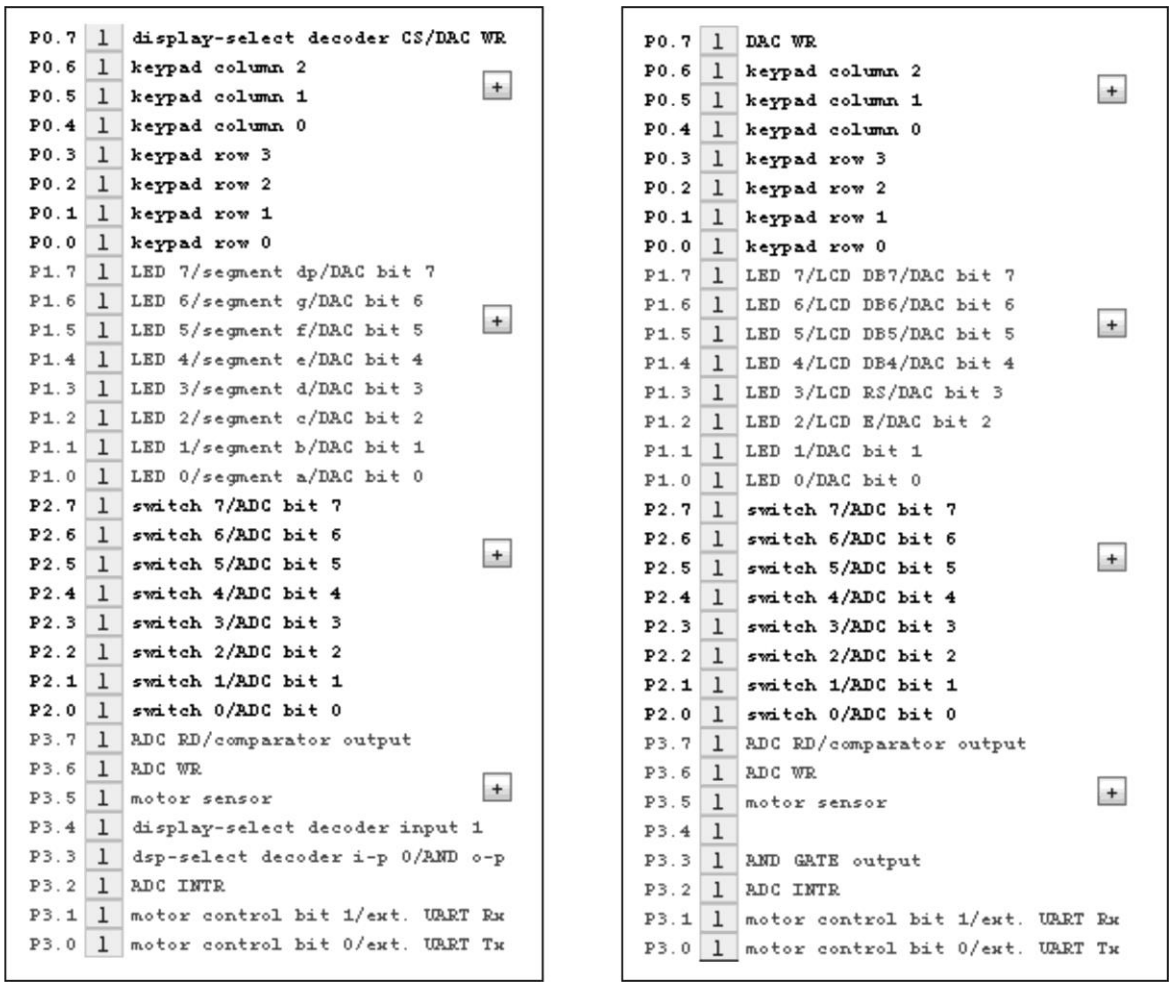


Рисунок 41. Під’єднання виводів портів

Необхідно натиснути на кнопку, позначену як «+», щоб показати відповідні зв’язки портів у окремому вікні. Нижче наведено приклад для порту 2.

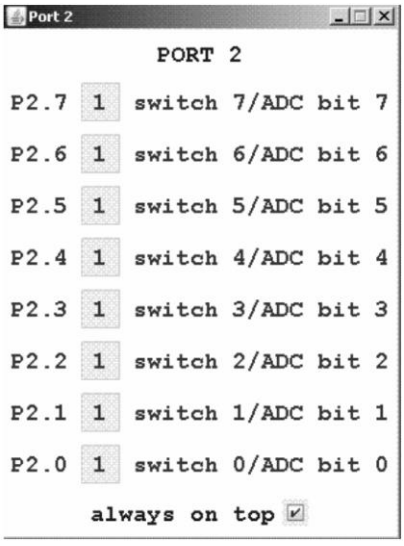


Рисунок 42. Зв’язки портів у окремому вікні.

1.11.17. Порти, світлодіодна матриця, ЦАП та 7-сегментні індикатори.

Нижче наведена схема, роботу якої моделює EDSim51, яка надає уявлення про периферійні зв'язки та взаємодію всіх пристроїв цієї схеми.

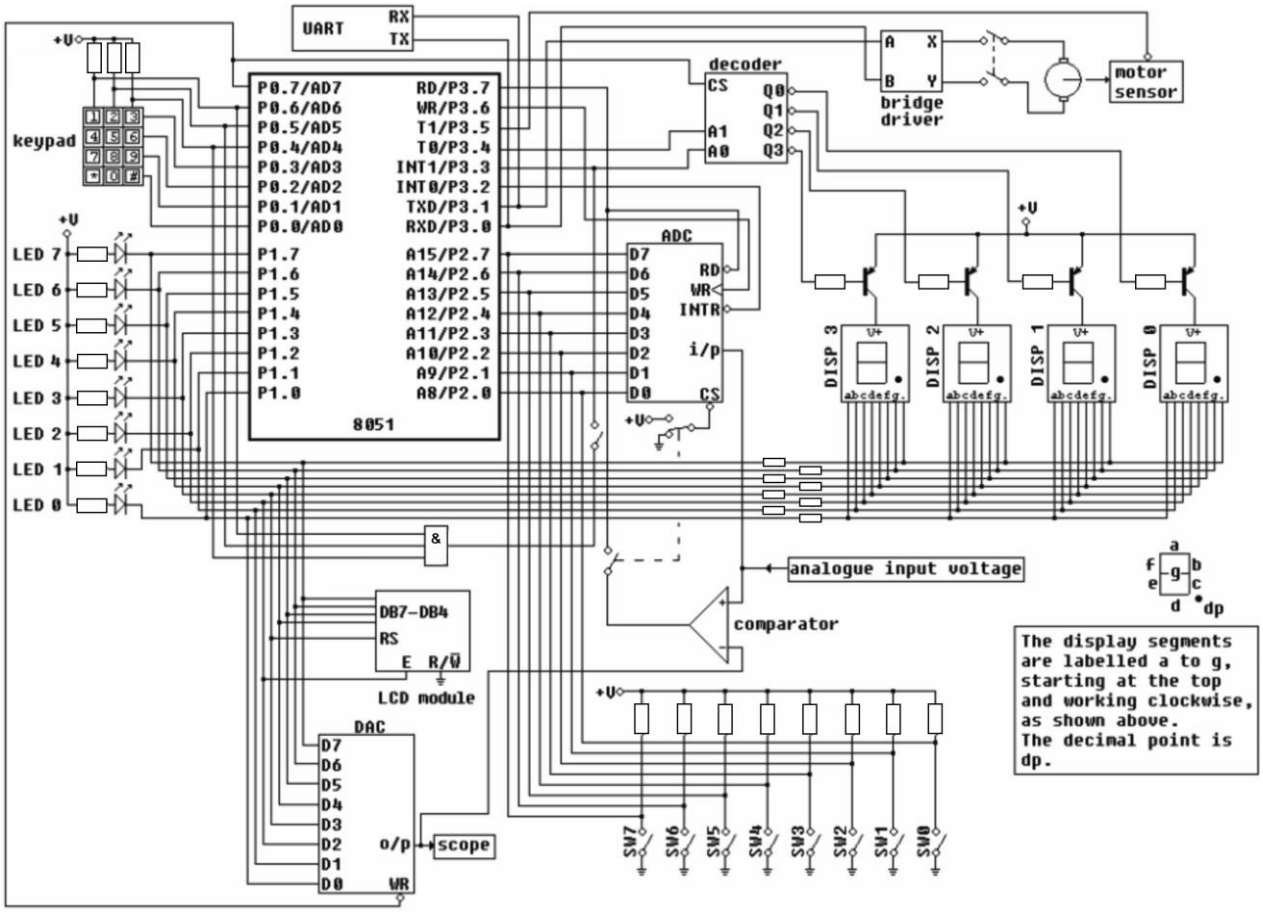


Рисунок 43. Схема, роботу якої моделює EDSim51

1.11.18. UART (універсальний асинхронний приймач/передавач).

Лінії керування двигуном мають одні й ті ж виводи портів, що й послідовний порт RXD та TXD. Зовнішній UART підключений до P3.0 і P3.1.

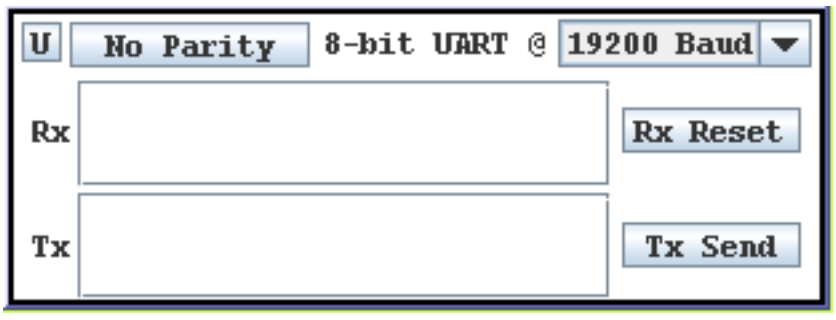


Рисунок 44. UART

Дані, отримані по послідовному порту, відображаються у вікні Rx. Дані в цьому вікні можна очистити в будь-який момент, натиснувши кнопку RxReset.

Дані можна передати по послідовному порту 8051, ввівши текст у вікно Tx і натиснувши кнопку TxSend. Після натискання кнопки фон вікна Tx змінюється на сірий, що означає неможливість редагування інформації. Позначення кнопки TxSend змінюється на TxReset. Повторне її натискання очистить поле Tx. Колір фону зміниться на білий, тобто повернеться можливість редагування та повторного передавання даних.

Дані, що передаються по зовнішньому UART, відокремлюються символом «\r» (ASCIIкод символу – 0Dh). Іншими словами, якщо дані для передавання через порт мають вигляд «abc», то насправді буде надіслано інформацію у вигляді «abc\r» (ASCII – 61H 62H 63H 0DH).

UART також може передавати набір 8-бітних даних замість тексту.

UART може бути встановлений у режим парності, непарності та відсутності парності за допомогою кнопки Parity: NoParity (за замовчуванням)->OddParity (непарність)- >EvenParity (парність).

Крім тексту можуть передаватися 8-бітні числа (записані у форматі HEX). Для цього користувачеві потрібно взяти послідовність у фігурні дужки, й кожне число відокремити комою, як це проілюстровано нижче. Якщо користувач хоче надіслати {56, 3A, 23, E7} у вигляді тексту, а не послідовності 8-розрядних чисел, необхідно почати його символом «\». Таким чином, \ {56, 3A, 23, E7} у полі Tx буде надіслано як {56, 3A, 23, E7} та 0DH.

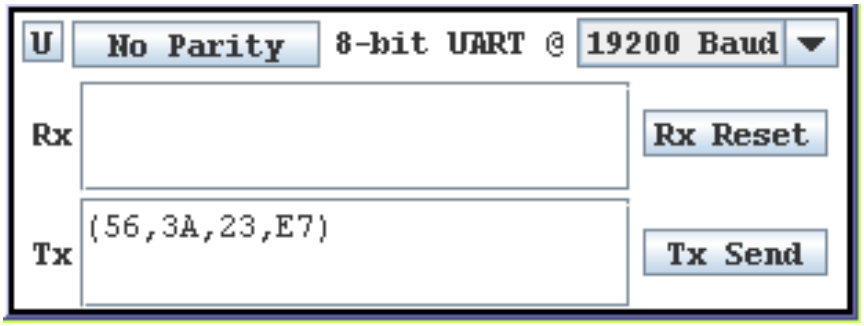


Рисунок 45. Передача даних

1.11.19. Вибір швидкості UART.

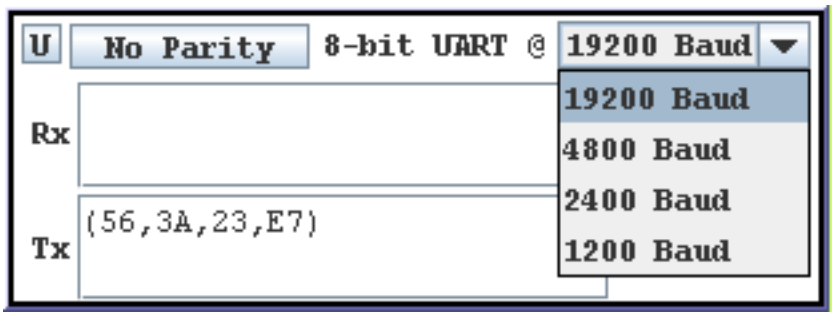


Рисунок 46. Вибір швидкості

Швидкість передавання зовнішнього UART можна вибрати зі списку доступних значень. Це дозволяє експериментувати з різними швидкостями і зрозуміти, як у МК 8051 послідовний порт, таймер 1 та біт SMOD використовуються разом для визначення необхідної швидкості передавання даних.

1.11.20. Клавіатура.

На нижньому рисунку показано фрагмент схеми, за якою до МК51 під’єднана клавіатура.

Зовнішнє переривання 1: три стовпчики клавіатури підключені до входів елемента AND (логічне І), вихід якого підключено до P3.3 – контакт зовнішнього переривання 1.

За замовчуванням AND неактивний, тому що цей контакт зазвичай використовується дешифратором вибору індикатора.

Щоб дозволити використання зовнішнього переривання 1 з клавіатурою, необхідно натиснути на кнопку ANDDisabled – переривання активується.

Необхідно пам’ятати, що не можна використовувати одночасно індикатори та переривання з клавіатури. Замість цього можна використати опцію очікування зайнятості на клавіатурі.

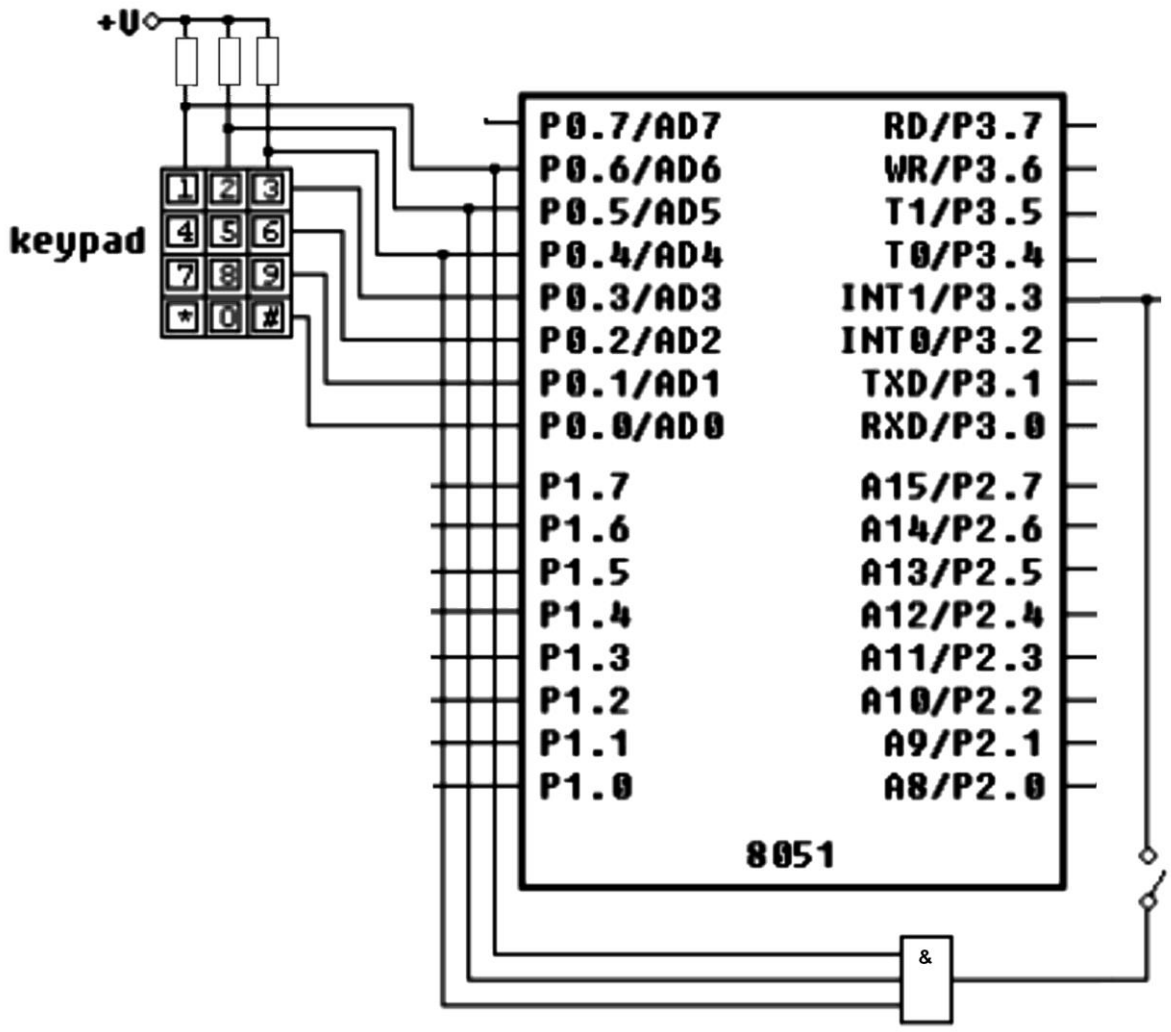


Рисунок 47. Підключення клавіатури

Завантажити програмний емулятор EDSim51 можна за адресою https://www.edsim51.com.

1.11.21. Контрольні запитання.

1. Однокристальні мікроЕОМ сімейства MCS-51. Основні характеристики.
2. Архітектура ОМЕОМ 87С51.
3. Будова арифметико-логічного пристрою.
4. Організація резидентної пам’яті даних та пам’ять програм.
5. Система переривань однокристальних мікроЕОМ сімейства MCS-51.
6. Порти вводу-виводу.

7. Послідовний інтерфейс.
8. Таймери/лічильники. Режими роботи.
9. Під’єднання до MCS-51 зовнішньої пам’яті даних та програм.
10. Проектування систем керування на базі MCS-51. Спряження MCS-51 з розширювачем вводу-виводу.
11. Проектування систем керування на базі MCS-51. Під’єднання до MCS-51 розширеної кількості давачів.
12. Виконання динамічної індикації на основі семисегментних індикаторів.

ТЕМА 2. ПРОЕКТУВАННЯ СИСТЕМИ КЕРУВАННЯ НА БАЗІ PIC16X8X

2.1. Особливості контролерів PIC16X8X.

Мікроконтролери підгрупи PIC 16x8x відносяться до сімейства 8-розрядних КМОН мікроконтролерів групи PIC16CXXX. Низька ціна, економічність, швидкодія, простота використання і гнучкість вводу-виводу робить PIC16X8X привабливим навіть у тих галузях, де раніше не застосовувалися мікроконтролери. Наприклад: таймери, заміна жорсткої логіки у великих системах, співпроцесори. Висока навантажувальна здатність спрощує зовнішні драйвери і тим самим зменшує загальну вартість системи. Розробки на базі контролерів PIC16X8X підтримуються асемблерами, симуляторами, схемними емуляторами й програматорами різних фірм.

Серія PIC16X8X підходить для широкого спектра схем від високошвидкісного керування автомобільними й електричними двигунами до економічних віддалених прийомопередавачів, вимірювальних приладів і процесорів вводу-виводу. Наявність EEPROM даних дозволяє розміщувати параметри настроювання всередині кристалу, чим знижує загальну вартість системи. Малі розміри корпусів як для звичайного, так і для поверхневого монтажу, роблять цю серію мікроконтролерів придатною для портативних цілей.

Вбудований автомат програмування кристалу PIC16X8X дозволяє легко змінювати програму й дані під конкретні вимоги, в тому числі з використанням внутрішньосхемного програмування – програмування кристалу після встановлення його в цільову систему. Ця можливість може бути використана як для тиражування, так і для занесення каліброваних даних уже після остаточного тестування.

Всі регістри контролерів статичні, отже мінімальна тактова частота може дорівнювати 0 Гц, максимальна тактова частота – 10 МГц. Усі команди виконуються за один цикл, що становить 4 машинні такти, крім команд переходів та виклику підпрограм, що виконуються за 2 цикли. Система команд включає 35 простих команд з ортогональною симетрією. Машинні

коди всіх команд 14-бітні й опрацьовують 8-бітні дані, використовуючи пряму, непряму й відносну адресацію даних. При звертанні до підпрограм та виклику переривань використовується вісьмирівневий апаратний стек, що не відображається в область ОЗП або в жодну іншу область пам’яті. Він призначений для зберігання адрес повернення з підпрограм, у тому числі з підпрограм обслуговування переривань. У випадку необхідності глибшого стеку в контролері передбачена можливість організації програмного стеку в ОЗП будь-якої необхідної глибини.

До складу сімейства входять МК PIC16F83, PIC16CR83, PIC16F84 і PIC16CR84. Характеристики МК підгрупи PIC16F8X наведено в таблиці 11.

Таблиця 11. Основні характеристики МК підгрупи PIC16F8X

Параметр	PIC16F83	PIC16CR83	PIC16C84	PIC16F84	PIC16CR84
Максимальна частота, МГц	10	10	10	10	10
Flash-пам’ять програм, слів	512	-	-	1K	-
EEPROM пам’ять програм, слів	-	-	1K	-	-
ПЗП програм, слів	-	512	-	-	1K
Пам’ять даних, байт	36	36	36	68	68
Пам’ять даних у РПЗП (EEPROM), байт	64	64	64	64	64
Таймери	TMR0	TMR0	TMR0	TMR0	TMR0
Кількість джерел переривань	4	4	4	4	4
Кількість ліній вводу-виводу	13	13	13	13	13
Діапазон напруги живлення, В	2.0–6.0	2.0–6.0	2.0–6.0	2.0–6.0	2.0–6.0
Кількість виводів і тип корпуса	18 DIP, SOIC	18 DIP, SOIC	18 DIP, SOIC	18 DIP, SOIC	18 DIP, SOIC

Мікроконтролери підгрупи PIC16F8X різняться між собою об'ємом ОЗП даних, а також об'ємом і типом пам'яті програм. Наявність у складі підгрупи МК із Flash і EEPROM-пам'яттю програм полегшує створення прототипів зразків виробів.

Підсистема пам'яті складається з:

◊ пам'яті програм об'ємом 1024x14, в якості якої використовується PROM (в PIC16C84), FLASH (в PIC16F84) або масковий постійний запам'ятовуючий пристрій (в PIC16CR84);

◊ пам'яті даних об'ємом 36x8 (в PIC16F83, PIC16CR83, PIC16C84), або 68x8 (в PIC16CR84 та в PIC16F84), що може використовуватися в якості регістрів загального призначення;

◊ EEPROM даних об'ємом 64 байти, що можуть бути запрограмовані як у процесі програмування мікросхеми, так і за нормальної роботи контролера при виконанні відповідних команд програми.

Мікросхема має можливість використовувати чотири джерела переривання:

◊ зовнішній вхід INT;

◊ переповнення таймера RTCC;

◊ переривання при зміні сигналів на лініях порту B;

◊ після завершення запису даних у пам'ять EEPROM.

Мікросхема має 13 ліній вводу-виводу з індивідуальним настроюванням на ввід або на вивід. Вхідний струм при виведенні на вихід логічного нуля становить 25 мА. Вихідний струм при виведенні логічної одиниці – 20 мА. Існує можливість генерації переривання по зміні стану відповідного виводу, а також відлік кількості змін сигналів на визначеному виводі.

У мікросхемі присутній 8-бітний таймер/лічильник RTCC з 8-бітним програмованим попереднім дільником, який може працювати в режимі таймера та лічильника імпульсів, що надходять на відповідний вхід.

Для підвищення надійності в PIC16X8X присутні можливості автоматичного скидання при пропаданні та появі живлення, таймери затримки виходу з режиму скидання, а також Watchdog-таймер (WDT).

Програмування здійснюється через вбудований пристрій програмування пам’яті програм і даних, що використовує лише два виводи кристала. При цьому саме програмування може виконуватися як в програматорі, так і в цільовій системі. Для захисту коду від несанкціонованої модифікації та копіювання можливе використання бітів захисту інформації. Також користувач при програмуванні може вибрати тип генератора синхронізації, що використовується з: RC-генератора (RC), звичайного кварцового резонатора (XT), високочастотного кварцового резонатора (HS), економічного низькочастотного кварцового резонатора (LP).

Також мікросхема володіє низьким енергоспоживанням:

◊ 3 мА при напрузі живлення 5В і частоті 4 МГц;

◊ 50 мкА при напрузі живлення 2В і частоті 32 кГц.

2.2. Призначення виводів та позначення мікросхеми

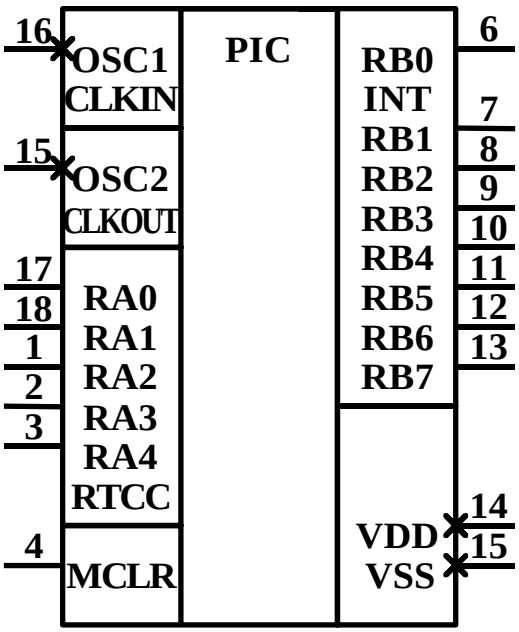


Рисунок 48. Умовне позначення мікросхеми

Умовне позначення мікросхеми наведено на рис. 48, а призначення виводів – у таблиці 12.

Таблиця 12. Призначення виводів мікросхеми

Позначення	Номер виводу	Призначення
RA0-RA3	17, 18, 1, 2	Двонаправлені лінії вводу-виводу. Вхідні та вихідні рівні відповідають ТТЛ логіці
RA4 RTCC	3	Двонаправлена лінія вводу-виводу. Вхід через тригер Шмідта. Вивід працює як вивід з відкритим стоком, також може використовуватися як вхід частоти для таймера/лічильника. В такому режимі роботи вивід має тригер Шмідта на вході
RB0/INT	6	Двонаправлена лінія вводу-виводу порту В. Лінія може використовуватися як вхід переривання, при цьому вхід буферизується тригером Шмідта
RB1 – RB7	7-13	Двонаправлені лінії вводу-виводу порту В. Лінії можуть бути запрограмовані в режимі внутрішніх активних навантажень на лінію живлення по усіх виводах. Виводи RB4...RB7 можуть бути програмно налаштовані як входи переривання по зміні стану на кожному з входів. При програмуванні МК RB6 використовується як тактовий, а RB7 – як вхід/вихід даних. У режимі програмування лінії RB6 та RB7 мають на вході тригер Шмідта. В інших режимах сигнали на лініях відповідають ТТЛ логіці
MCLR	4	Сигнал скидання. Логічний нуль на лінії призводить до скидання мікросхеми. В режимі програмування призначений для подавання напруги програмування. Лінія на вході має тригер Шмідта
OSC1 CLKIN	16	Вхід під’єднання кварцу, зовнішньої RC-ланки або зовнішнього джерела синхронізації. Вхід має тригер Шмітта, коли конфігурується в режимі RC-генератора і КМОН в усіх інших випадках
OSC2 CLKOUT	15	Вхід під’єднання кварцу, вихід сигналу синхронізації в режимі синхронізації за допомогою RC ланки або в режимі ввімкнення зовнішнього тактового сигналу. Частота сигналу в режимі роботи з RC-ланкою дорівнює 1/4 частоти OSC1 і вказує швидкість виконання команд
VDD	14	Напруга живлення
VSS	5	Загальний (земля)

Параметри сигналів не мають виходити за межі, вказані в таблиці 13.

Таблиця 13. Граничні значення сигналів

Параметр	Граничні значення
Напруга на будь-якому виводі відносно землі (за винятком VDD та MCLR)	-0.6 В VDD+0.6 В
Напруга VDD (робоча) для PIC16C84, PIC16CR84, PIC16F84	4.0... 6 В
Напруга VDD (робоча) для PIC16LC84	2.0...6 В
Напруга VDD (у режимі SLEEP)	1.5...6 В
Напруга VDD (максимально допустима)	0...7.5 В
Струм логічного нуля на кожному виході, що може прийняти мікросхема	25 мА
Струм логічного нуля на кожному виході, що може видати мікросхема	20 мА
Струм логічної одиниці на кожному виході (у будь-якому напрямку)	20 мА
Сумарний струм, що приймається через лінії порту А	80 мА
Сумарний струм, що видається через лінії порту А	50 мА
Сумарний струм, що приймається через лінії порту В	150 мА
Сумарний струм, що видається через лінії порту В	100 мА
Максимальний струм через лінію VDD	100 мА
Максимальний струм через лінію VSS	150 мА

2.3. Архітектура PIC16X8X.

Спрощена архітектура PIC16X8X зображена на рис. 49. Архітектура контролера базується на концепції роздільних шин та областей пам’яті для даних і команд. Шина даних і пам’ять даних мають ширину 8 бітів, а шина адреси програм і ПЗП програм мають ширину 13 і 14 бітів відповідно. Така концепція забезпечує просту, але потужну систему команд, розроблену так, що бітові, байтові й реєстрові операції працюють із високою швидкістю та з перекриттям за часом вибірок команд і циклів виконання. 14-бітова ширина програмної пам’яті забезпечує вибірку 14-бітової команди в один цикл. Двоступінчастий конвеєр забезпечує одночасну вибірку й виконання команди. Усі команди виконуються за один цикл, крім команд переходів.

Розглянемо основні функціональні блоки структурної схеми мікроконтролера.

Блок вибірки команд призначений для вибірки команд, що виконуються в певній послідовності. До складу блоку входять 4 підблоки:

1. ПЗП програм, призначений для збереження коду програми, під керуванням якої працює мікроконтролер. Структуру ПЗП програм та спеціалізовані адреси в ПЗП розглянемо пізніше.

2. Лічильник команд (ЛК), що має розрядність 13 бітів і призначений для визначення адреси наступної команди, що буде виконуватися.

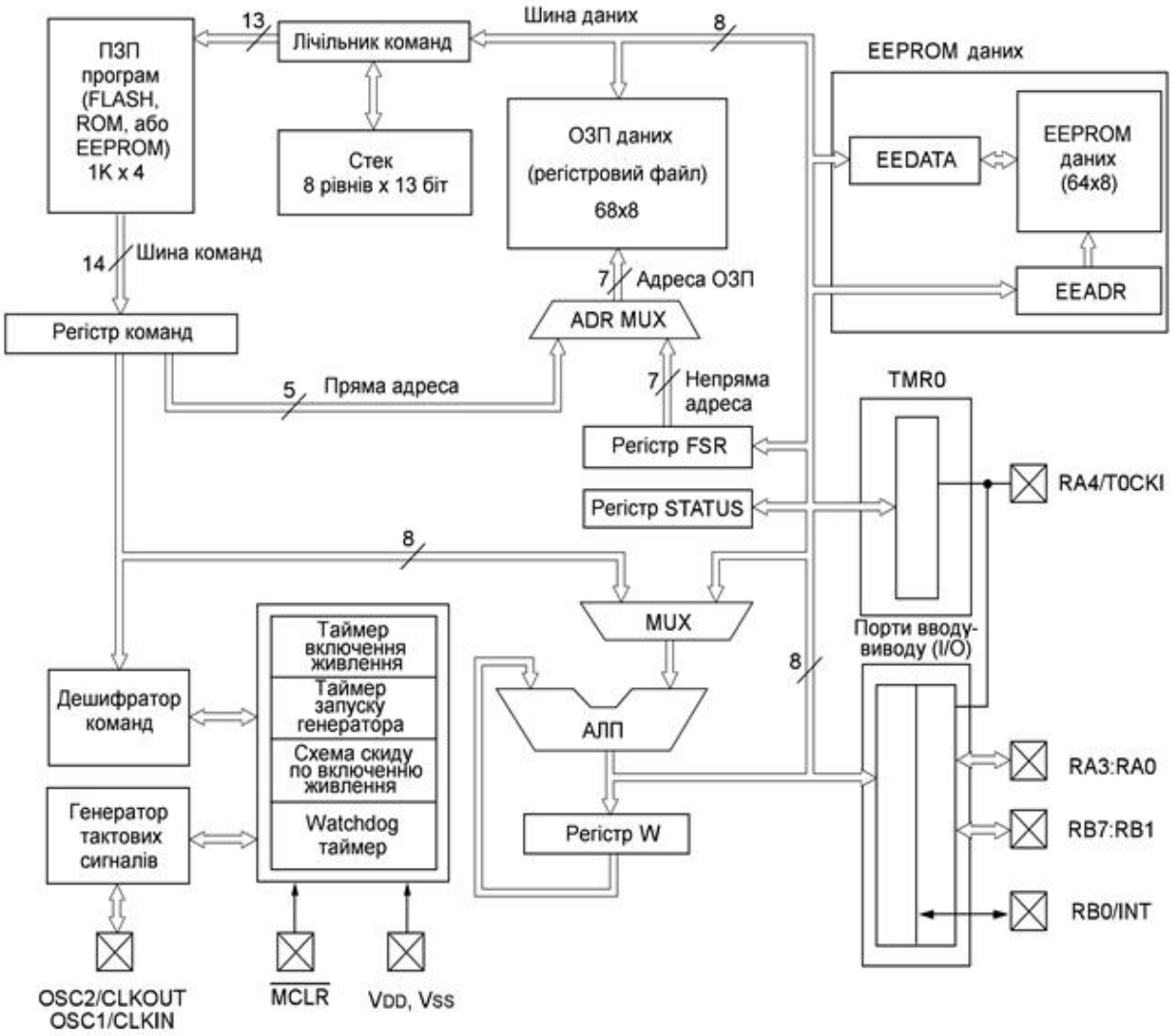


Рисунок 49. Структурна схема мікросхеми

3. Стек, що має 8 рівнів і розрядність 13 бітів, призначений для зберігання тільки адрес повернення з підпрограм.

4. Регістр команд (РК), що використовується для зберігання коду команди, котра виконується. Команда після вибірки з ПЗП програм записується в регістр і зберігається протягом усього часу її виконання.

Блок ОЗП, призначений для зберігання даних і складається із таких трьох підблоків:

1. ОЗП даних (регістровий файл 68x8), призначений для зберігання даних, що використовуються при роботі програми.

2. Адресний мультиплексор (ADR MUX), що визначає джерело адреси, за допомогою якої адресуються комірки ОЗП даних. Такими джерелами можуть бути регістр FSR при непрямій адресації та регістр команд – при прямій.

3. Регістр вибору адреси (регістр FSR), що використовується при непрямій адресації комірок ОЗП даних. Розрядність регістра 8 бітів, отже, за його допомогою можна адресувати будь-яку комірку ОЗП.

Блок синхронізації та керування. Призначений для синхронізації й керування іншими блоками мікросхеми. Він складається із двох таких підблоків:

1. Генератора тактових сигналів (ГТС), який призначений для генерації сигналів синхронізації роботи окремих вузлів мікросхеми.

2. Дешифратора команд (ДК), котрий призначений для перетворення коду команди в сигнали керування іншими блоками мікросхеми. Синхронізація блоку здійснюється генератором тактових сигналів.

Блок формування сигналу скидання призначений для формування сигналу скидання від різних можливих джерел. До блоку входять чотири підблоки:

1. Схема скидання по ввімкненню живлення (ССВЖ). Схема встановлює внутрішній сигнал скидання в активний рівень при ввімкненні живлення (коли на лінії VDD є передній фронт сигналу і напруга перевищує 1.2–1.7 В).

2. Таймер увімкнення живлення (ТВЖ). Призначений для утримання сигналу скидання тривалістю 72 мс після того, як напруга живлення увійде в робочий діапазон. Це знижує ймовірність невірної роботи мікросхеми у випадку нестабільного живлення.

3. Таймер увімкнення генератора (ТВГ). Утримує мікроЕОМ у стані скидання протягом 1024 цикли після вимкнення таймера ТВЖ.

4. Watchdog – таймер (WT), називають також сторожовим таймером – це спеціальний таймер, що призводить мікроконтролер у стан скидання у випадку збою програми.

Розглянуті таймери можуть бути використані вибірково, щоб уникнути небажаних очікувань як при увімкненні, так і при виході з режиму SLEEP.

Операційний блок виконує всі арифметичні й логічні операції та складається із трьох підблоків:

1. Арифметико-логічного пристрою (АЛП), що виконує всі арифметичні та логічні операції.

2. Регістр W – регістр тимчасового зберігання. Розрядність регістра – 8 бітів.

3. Регістр STATUS – регістр, призначений для зберігання ознак виконаних арифметичних і логічних операцій, а також визначення банку пам’яті даних і стану Watchdog – таймера.

Блок EEPROM даних призначено для зберігання даних, які необхідно зберегти після вимкнення живлення. До блоку відносять 3 підблоки:

1. Регістр EEADR, котрий визначає адресу комірки EEPROM, з якою відбувається обмін.

2. Регістр EEDATA, котрий служить для зберігання даних, які записуються в EEPROM, а також для отримання даних з неї.

3. EEPROM даних – матриця пам’яті, в якій зберігається інформація.

Також на структурній схемі позначені:

Таймер/лічильник TMRO – восьмирозрядний лічильник, що може працювати в режимі лічильника подій і таймера. До блоку входить також попередній подільник з коефіцієнтом поділу 1:2, 1:4 .. 1:256.

Порти вводу-виводу призначені для введення/виведення інформації. В мікроконтролері є 13 ліній вводу-виводу, що згруповані в 5-розрядний порт А та 8-розрядний порт В. Кожна лінія будь-якого порту може бути налаштована незалежно від інших на введення і на виведення інформації. При зміні стану ліній порту В можлива генерація переривань.

2.4. Робота мікроконтролера.

Схема тактування й виконання команди зображена на рис. 50. Вхідна тактова частота, що надходить з виводу OSC1/CLKIN, ділиться всередині блоку на чотири і з неї формуються чотири тактові послідовності Ql, Q2, Q3 і Q4, що не перекриваються одна з одною. Лічильник команд збільшується в такті Q1, команда зчитується з пам’яті програми і запам’ятовується в регістрі команд у такті Q4. Команда декодується й виконується протягом наступного циклу в тактах Q1...Q4. Протягом тактів Q2, Q3 і Q4 наступного циклу відбувається декодування й виконання команди. У такті Q2 зчитується пам’ять даних (читання операнда), а запис відбувається в такті Q4.

Цикл виконання команди складається з чотирьох тактів: Q1...Q4. Вибірка команди та її виконання сполучені за часом таким чином, що вибірка команди займає один цикл, а виконання – наступний. Ефективний час виконання команди складає один цикл.

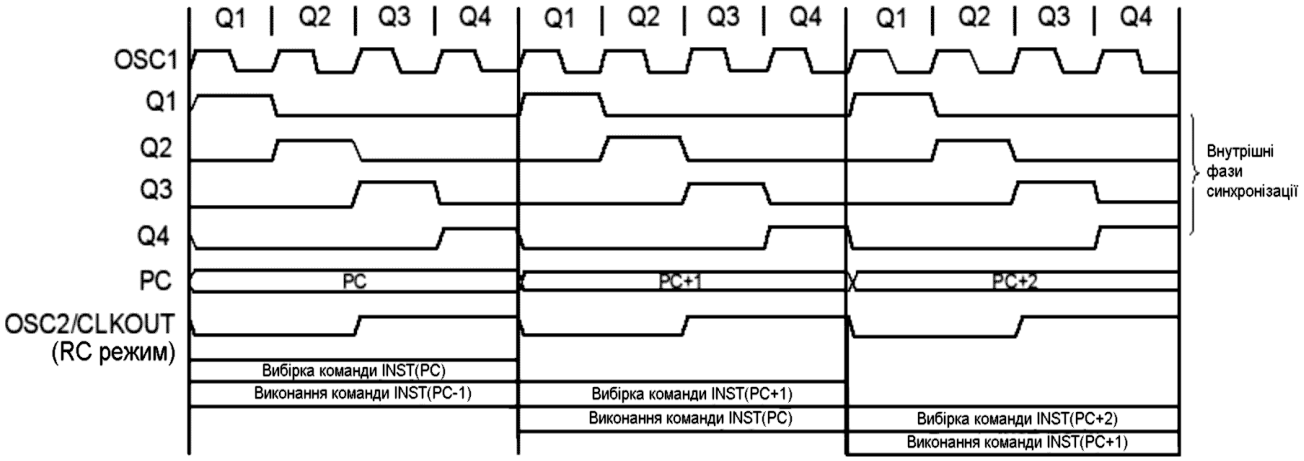


Рисунок 50. Схема тактування й виконання команди

Якщо команда змінює вміст лічильника команд (наприклад, команда GOTO), то для її виконання потрібно два цикли. Після переходу необхідно вибрати код команди за новою адресою, в цей час виконання наступної команди неможливе.

2.5. Структура та робота операційного блоку.

Мікроконтролер містить 8-розрядний арифметико-логічний пристрій (АЛП) і робочий регістр W. АЛП є арифметичним модулем загального призначення й виконує арифметичні та логічні операції над умістом робочого регістра і кожного з регістрів контролера.

АЛП може виконувати операції додавання, віднімання, зсуву й логічні операції. Якщо не зазначено інше, то арифметичні операції виконуються в додатковому двійковому коді. Залежно від результату операції, АЛП може змінювати значення бітів регістра STATUS: C (Carry), DC (Digit carry) і Z (Zero). Регістр статусу (STATUS) містить ознаки операції (арифметичні прапорці) АЛП, стан контролера при скиданні та біти вибору сторінок для пам'яті даних. Формат регістра STATUS зображено на рис. 31, а призначення бітів наведено у таблиці 14.

Таблиця 14. Призначення бітів регістра STATUS

Біт	Значення
1	3
IRP	Біт вибору сторінки банку даних (використовується при непрямій адресації): 0 = банк 0, 1 (00h-FFh), 1 = банк 2, 3 (100h-1FFh). Біт IRP не використовується в МК підгрупи PIC16F8X
RP1-RP0	Біти вибору сторінки банку даних (використовуються при прямій адресації): 00 = банк 0 (00h-7Fh); 01 =банк 1 (80h-FFh); 10 = банк 2(100h-17Fh); 11 = банк 3(180h-IFFh). У МК підгрупи PIC16F8X використовується тільки біт RP0
$\overline{\text{T0}}$	Біт спрацювання сторожового таймера. Контролює спрацювання сторожового таймера, встановлюється в одиницю командами CLRWDT і SLEEP, а також сигналом скидання (крім сигналу від сторожового таймера). Скидання відбувається при спрацюванні сторожового таймера
$\overline{\text{PD}}$	Біт зниження споживаної потужності. Логічна «1» вказує на режим зниження потужності споживання. Встановлюється при увімкненні живлення, а також командою CLRWDT. Скидається командою SLEEP

Закінчення таблиці 14

1	2
Z	Біт нульового результату. Одиниця вказує, що результат останньої арифметичної або логічної операції був нульовим, одиниця – не був
DC	Біт десяткового (додаткового) перенесення/позичання. Одиничне значення вказує, що при виконанні останньої арифметичної операції відбулося перенесення з третього в четвертий розряд. Використовується при роботі з двійково-десятковими числами
C	Біт перенесення/позичання. Одиничне значення вказує, що при виконанні останньої арифметичної операції відбулось переповнення або переспустошення. Біт також встановлюється та скидається командами зсуву. При виконанні команд зсуву цей біт завантажується з молодшого чи старшого розряду джерела залежно від команди

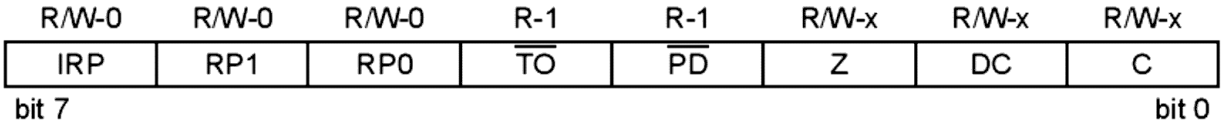


Рисунок 51. Формат регістра STATUS

На рис. 51 прийняті такі позначення: R – біт читання; W – біт запису; S – встановлюваний біт; U – біт, що використовується (читається як «0»), «0» або «1» – значення біта після скидання.

Регістр статусу доступний для будь-якої команди так само, як будь-який інший регістр. Однак, якщо регістр STATUS є регістром призначення для команди, що впливає на біти Z, DC чи C, запис у них буде заборонено. Біти $\overline{T0}$ і $\overline{PD}$ встановлюються апаратно і не можуть бути записані в регістр статусу. Наприклад, команда CLRF STATUS обнулить усі біти, крім бітів $\overline{T0}$ і $\overline{PD}$, і встановить біт Z=l, хоча після виконання цієї команди регістр статусу буде мати ненульове значення.

Рекомендується для зміни регістра статусу використовувати тільки команди встановлення бітів BCF, BSF, MOVWF.

2.6. Структура ПЗП програм.

ПЗП програм у різних типів контролерів сімейства побудовано на основі EEPROM, флеш-пам’яті або масочного ПЗП. Об’єм ПЗП змінюється від 512 до 1024 слів, що мають розрядність 14 бітів. До ПЗП під’єднана 13-розрядна адресна шина, з якої використовуються лише 9 або 10 молодших розрядів. Старші розряди є зарезервованими. Організацію пам’яті програм і стека зображено на рис. 52.

У пам’яті програм є виділені адреси. Вектор скидання знаходиться за адресою 0000 h, вектор переривання – за адресою 0004 h. Зазвичай, за адресою 0004 h розташовується підпрограма ідентифікації й опрацювання переривань, а за адресою 0000 h – команда переходу на програму початкового встановлення, розташовану за підпрограмою опрацювання переривань. Уся пам’ять програм є внутрішньою. Запустити програму з зовнішньої пам’яті неможливо.

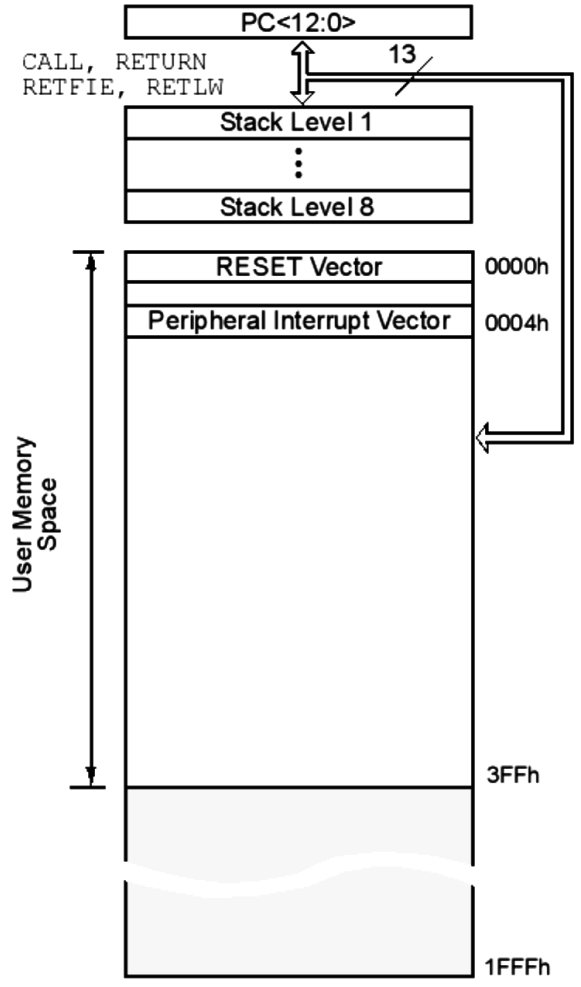


Рисунок 52. Організація пам’яті програм і стека

Для адресації пам’яті програм використовується лічильник команд. Лічильник команд у МК PIC16F8X має розрядність 13 бітів і здатний адресувати 8 К х 14 об’єм програмної пам’яті. Фактично на кристалах PIC16F83 і PIC16CR83 є тільки 512 × 14 пам’яті (адреси 0000h-0lFFh), а в МК PIC16C84, PIC16F84 і PIC16CR84 – 1К×14 пам’яті (адреси 0000h-03FFh). Звертання до адрес вище 1FFh (3FFh) є по суті адресацією перших 512 або 1К комірок пам’яті програм.

Молодший байт лічильника (PCL) доступний для читання і записування й відображається на регістр з адресою 02h. Старший байт лічильника команд не може бути прямо записаний чи зчитаний і береться з регістра PCLATH (PC latch high), адреса якого 0Ah. Вміст PCLATH передається в старший байт лічильника команд, коли він завантажується новим значенням.

Залежно від того, чи завантажується в лічильник команд нове значення під час виконання команд CALL, GOTO, чи в молодший байт лічильника команд (PCL) здійснюється запис, старші біти лічильника команд завантажуються з PCLATH різними способами, як зображено на рис. 53.

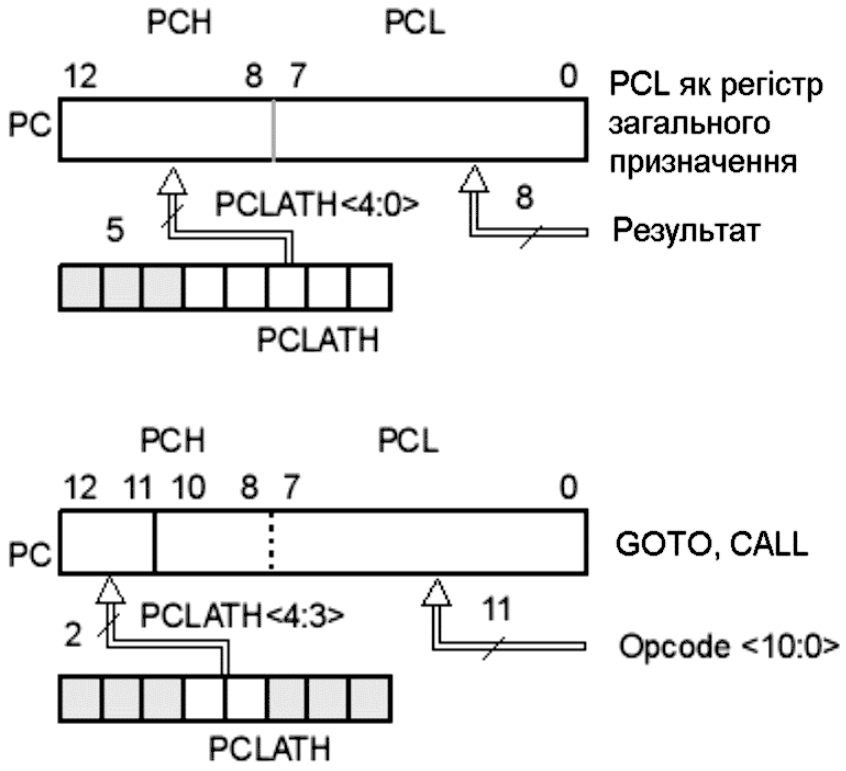


Рисунок 53. Завантаження старших бітів лічильника команд

Можливість виконувати арифметичні операції безпосередньо над лічильником команд дозволяє дуже швидко й ефективно здійснити табличні перетворення в PIC-контролерах.

Команди CALL і GOTO оперують 11-розрядним адресним діапазоном, достатнім для зсуву в межах сторінки програмної пам'яті об'ємом 2К слів. Для МК підгрупи PIC16F8X цього вистачає. З метою забезпечення можливості розширення пам'яті команд для майбутніх моделей МК передбачене завантаження двох старших бітів лічильника команд із регістра PCLATH<4:3>. При використанні команд CALL і GOTO користувач повинен переконатися в тому, що ці сторінкові біти запрограмовані для виходу на потрібну сторінку. При виконанні команди CALL чи виконанні переривання весь 13-бітний лічильник команд завантажується в стек, тому для повернення з підпрограми не потрібні маніпуляції з розрядами PCLATH<4:3>.

Мікроконтролери підгрупи PIC16F8X ігнорують значення бітів PCLATH<4:3>, що використовуються для звертання до сторінок 1, 2 і 3 програмної пам'яті. Однак застосовувати біти PCLATH<4:3> у якості комірок пам'яті загального призначення не рекомендується, тому що це може вплинути на сумісність із майбутніми поколіннями виробів.

Мікроконтролери підгрупи PIC16F8X мають восьмирівневий апаратний стек шириною 13 бітів. На відміну від попередньо розглянутих мікропроцесорів і мікроЕОМ, у даному сімействі стек ніяк не відображається на область пам'яті даних. Стек у мікроЕОМ цього типу призначений лише для збереження адрес повернення з підпрограм.

Область стека не належить ні до програмної області, ні до області даних, а покажчик стека користувачу недоступний. Поточне значення лічильника команд посилається в стек, коли виконується команда CALL чи здійснюється опрацювання переривання. При виконанні процедури повернення з підпрограми (команди RETLW, RETF1E чи RETURN) уміст лічильника команд відновлюється зі стека. Регістр PCLATH при операціях зі стеком не змінюється.

Стек працює як циклічний буфер. Отже, після того, як стек був завантажений 8 разів, дев’яте завантаження перепише значення першого. Якщо стек був вивантажений 9 разів, лічильник команд стає таким же, як після першого вивантаження.

Положення стека в контролері не передбачено, тому програміст повинен самостійно стежити за рівнем вкладання підпрограм.

2.7. Структура ОЗП.

Мікроконтролер може за допомогою прямої або непрямої адресації звертатися до регістрів пам’яті чи даних. Усі регістри спеціальних функцій, включаючи лічильник команд, відображаються на пам’ять даних (рис. 54).

Пам’ять даних МК поділена на дві області.

Перші 12 адрес – це область *регістрів спеціальних функцій* (SFR), друга – область *регістрів загального призначення* (GPR). Область SFR керує роботою мікросхеми.

Обидві області, у свою чергу, поділені на банки 0 і 1.

Банк 0 вибирається обнуленням біта RP0 регістра статусу (STATUS). Встановленням біта RP0 в одиницю вибирається банк 1.

Кожен банк має довжину 128 байт. Однак, для PIC16F83 і PIC16CR83 пам’ять даних існує тільки до адреси 02Fh, а для PIC16F84 і PIC16CR84 – до адреси 04Fh.

Деякі регістри спеціального призначення продубльовані в обох банках, а до деяких звертання можливе лише в одному банку.

Регістри з адресами 0Ch-4Fh (0Ch-2Fh) можуть використовуватися як регістри загального призначення, що за будовою є статичним ОЗП. Адреси регістрів загального призначення банку 1 відображаються на банк 0. Отже, коли встановлено банк 1, то звертання до адрес 8Ch-CFh фактично адресує банк 0.

У регістрі статусу, крім біта RP0, є біт RB1, що дозволяє звертатися до чотирьох банків, які можуть бути присутні в майбутніх модифікаціях цього мікроконтролера.

До комірок ОЗП можна звертатися напряму, використовуючи абсолютну адресу кожного регістра або за допомогою непрямої адресації, через регістр-вказівник FSR.

Непряма адресація використовує поточне значення розрядів RP1:RP0 для доступу до банків.

Рисунок 54. Організація пам’яті даних

2.7.1. Пряма та непряма адресація даних.

Коли здійснюється пряма адресація даних, молодші 7 бітів беруться з коду операції, а два біти покажчика сторінок (RP1, RP0) – з регістра статусу, як зображено на рис. 55.

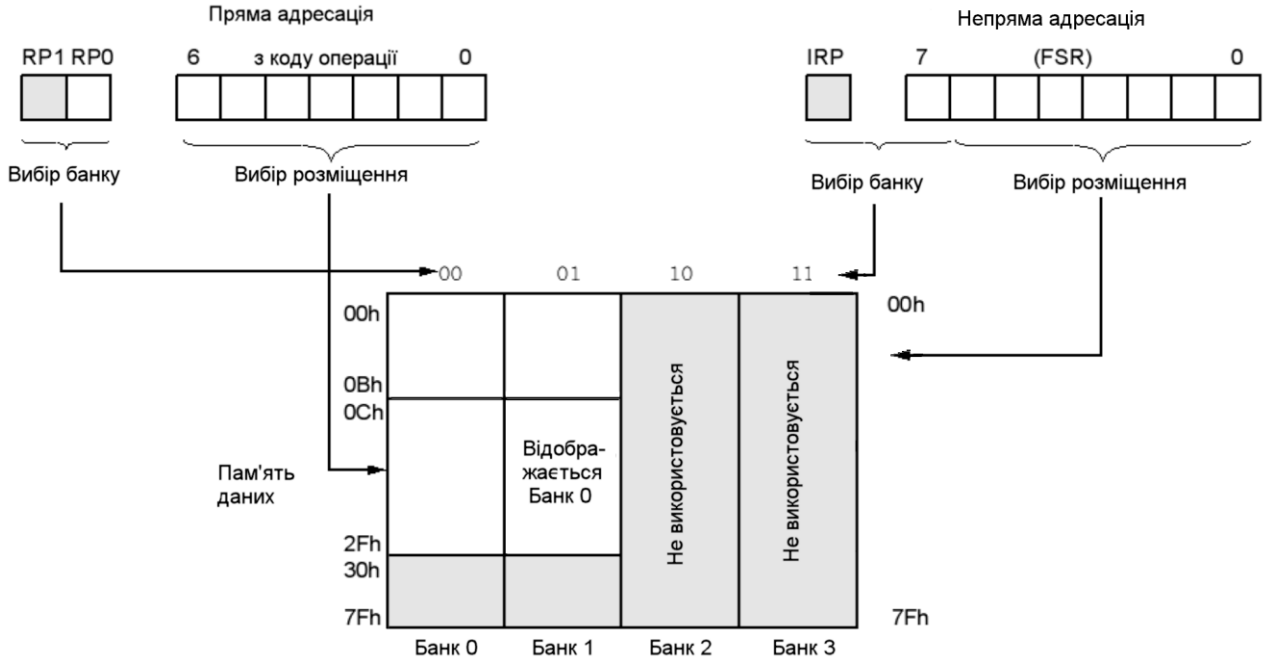


Рисунок 55. Методи адресації даних

Ознакою непрямої адресації є звертання до регістра INDF. Будь-яка команда, що використовує INDF (адресу 00h) у якості операнда, звертається до покажчика, що зберігається в FSR (адреса 04h). Непряме читання самого регістра INDF дасть результат 00h. Непрямий запис у регістр INDF призведе до втрати даних, що записувалися, хоча і може змінити біти стану. Необхідна 9-бітна адреса формується об’єднанням вмісту 8-бітного регістра FSR і біта IRP з регістра статусу.

2.8. EEPROM даних PIC16X8X.

Мікроконтролери підгрупи PIC16X8X мають енергонезалежну EEPROM пам’ять даних об’ємом 64x8 бітів, що допускає запис і читання під час нормальної роботи в усьому діапазоні напруги живлення. Ця пам’ять не належить до області ОЗП даних. Доступ до неї здійснюється за допомогою

непрямої адресації через регістри спеціальних функцій: EEDATA (з адресою 08h), через який здійснюється записування і читання даних, і EEADR (адреса 09h), котрий визначає адресу комірки, до якої здійснюється звертання. Для керування процесом читання та записування використовуються регістри EECON1 (адреса 88h) і EECON2 (адреса 89h).

При записуванні байта в комірку EEPROM попереднє значення стирається і записуються нові дані. Всі операції зі стирання й записування здійснює вбудований автомат записування EEPROM. Уміст комірок цієї пам'яті при вимиканні живлення зберігається.

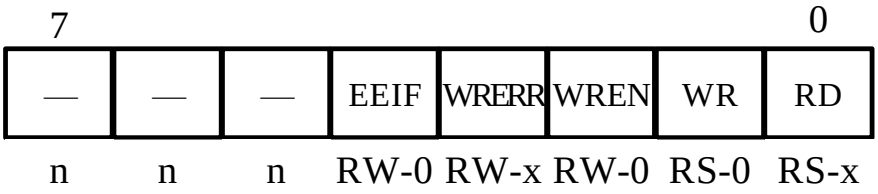


Рисунок 56. Формат регістра EECON1

Регістр EEADR може адресувати до 256 байтів даних EEPROM. У МК підгрупи PIC16X8X використовуються тільки перші 64 байти, адресовані шістьма молодшими бітами EEADR<5:0>. Однак старші два біти також декодуються. Тому ці два біти повинні бути встановлені в «0», щоб адреса потрапила в доступні 64 біти адресного простору.

Формат регістра EECON1 зображено на рис. 56, призначення бітів регістра наведено в таблиці 15.

Таблиця 15. Призначення бітів регістра EECON1

Біт	Значення
1	2
EEIF	Біт запиту переривання після закінчення записування в EEPROM 0 = операція записування незакінчена або не починалася 1 = операція записування закінчена Біт скидається програмно

Закінчення таблиці 15

1	2
WRERR	Біт ознаки помилки записування в EEPROM 0 = операція записування завершена 1 = операція записування передчасно перервана
WREN	Біт дозволу записування в EEPROM 0 = записування у EEPROM заборонений 1 = записування у EEPROM дозволений
WR	Біт керування записом. Встановлення біта в «1» ініціює цикл записування, біт скидається апаратно після завершення записування і не може бути повернений програмно в «0»
RD	Біт керування читанням. Встановлення біта в «1» ініціює цикл читання і займає один цикл. Біт скидається апаратно після завершення читання і не може бути програмно скинутий в «0»

Регістр EECON2 призначений для запобігання записування в EEPROM при збоях програми. Для дозволу записування в EEPROM перед записуванням кожного байта необхідно в регістр подати спеціальну ключову послідовність (спочатку байт 55h, потім AAh).

Тоді в наступному машинному циклі при встановленні біта WR регістра EECON1 почнеться занесення даних у EEPROM. Регістр EECON2 використовується винятково при організації записування даних у EEPROM. Читання регістра EECON2 дає нулі.

При зчитуванні даних з пам’яті EEPROM необхідно записати потрібну адресу в регістр EEADR і потім встановити біт RD EECON1<0> в одиницю. Дані з’являться в наступному командному циклі в регістрі EEDATA і можуть бути прочитані. Зауважимо, що дані в регістрі EEDATA фіксуються. Немає необхідності виконувати ще раз цикл читання у випадку повторного читання тієї ж комірки пам’яті.

При записуванні в пам’ять EEPROM необхідно спочатку записати адресу в регістр EEADR і дані в регістр EEDATA. Потім необхідно виконати спеціальну послідовність команд, що виконує записування даних:

```
movlw 55h    ;
movwf EECON2 ; записати в регістр EECON2 перший
             ; ключовий код
movlw AAh
movwf EECON2 ; записати в регістр EECON2 другий
             ; ключовий код
bsf EECON1.WR; встановити WR біт,
             ; встановлення біта відразу після
             ; подавання ключових кодів
             ; починає процес запису.
```

Під час виконання цієї ділянки програми всі переривання повинні бути заборонені для точного виконання часової діаграми. Час записування – приблизно 10 мс. Фактичний час записування може змінюватися залежно від напруги, температури та індивідуальних властивостей мікросхеми. В кінці записування біт WR автоматично обнулюється, а ознака завершення записування EEIF (вона ж є запитом на переривання) встановлюється.

Для запобігання випадкових записів у пам’ять даних передбачено спеціальний біт WREN у регістрі EECON1. Рекомендується тримати біт WREN вимкненим, кодові сегменти, що встановлюють біт WREN, і ті, котрі записують дані в EEPROM, варто зберігати на різних адресах, щоб уникнути випадкового виконання їх обох при збої програми.

2.9. Регістри спеціальних функцій.

Загальний опис регістрів спеціальних функцій наведено в таблиці 16. Більшість регістрів спеціальних функцій розглядаються в інших розділах, тому опис їх у цьому розділі не наведено. Виняток становить регістр конфігурації.

Таблиця 16. Регістри спеціальних функцій

Адреса	Назва	Призначення
00H	INDF	Фіктивний регістр непрямої адресації. Звертання до регістра відповідає звертанню до комірки пам'яті з адресою, котра задана в регістрі FSR
01H	TMR0	Регістр-лічильник таймера-лічильника. Зберігає поточне значення лічильника
02H	PCL	Молодший байт лічильника команд
03H	STATUS	Регістр стану мікросхеми
04H	FSR	Вказівник при непрямій адресації
05H	PORTA	Регістр, що вказує стан порту А. Значення мають лише молодші 5 бітів
06H	PORTB	Регістр, що вказує стан порту В
07H	-	Не використовується (зарезервований для порту С)
08H	EEDATA	Регістр даних для обміну з EEPROM
09H	EEADR	Регістр адреси для обміну з EEPROM
0AH	PCLATH	Старші 5 розрядів для записування в лічильник команд
0BH	INTCON	Регістр умов переривання
80H	INDF	Відображається регістр з адреси 00h
81H	OPTION	Регістр конфігурації
82H	PCL	Відображається регістр з адреси 02h
83H	STATUS	Відображається регістр з адреси 03h
84H	FSR	Відображається регістр з адреси 04h
85H	TRISA	Регістр вказує напрямок передавання порту А
86H	TRISB	Регістр вказує напрямок передавання порту В
87H	-	Не використовується (зарезервований для порту С)
88H	EECON1	Регістр режиму роботи EEPROM
89H	EECON2	Регістр ключа записування в EEPROM
8AH	PCLATH	Відображається регістр з адреси 0Ah
8BH	INTCON	Відображається регістр з адреси 0Bh

2.9.1. Регістр конфігурації (OPTION).

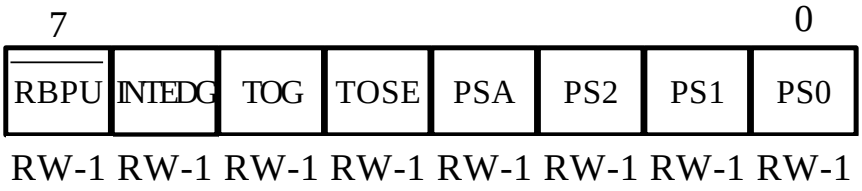


Рисунок 57. Формат регістра конфігурації OPTION

Регістр конфігурації (OPTION) доступний для читання й записування. Призначення бітів регістра наведено в таблиці 17, а формат регістра – на рис. 57.

Таблиця 17. Формат регістра конфігурації (OPTION)

<table>
<tr><td>Назва</td><td colspan="3">Призначення</td></tr>
<tr><td>$\overline{\text{RBPU}}$</td><td colspan="3">Біт встановлення підтягуючих «pull-up» резисторів на виводах PORTB:
0 – резистори під’єднані;
1 – резистори від’єднані</td></tr>
<tr><td>INTEDG</td><td colspan="3">Біт вибору переходу сигналу переривання:
0 – переривання по спаду сигналу на лінії RBO/INT;
1 – переривання по фронту сигналу на лінії RBO/INT</td></tr>
<tr><td>TOCS</td><td colspan="3">Біт вибору джерела сигналу таймера TMRO:
0 – внутрішній тактовий сигнал (CLKOUT);
1 – лінія RA4/TOCKI</td></tr>
<tr><td>TOSE</td><td colspan="3">Біт вибору переходу джерела сигналу для TMR0:
0 – збільшення по фронту сигналу на лінії RA4/TOCKI;
1 – збільшення по спаду сигналу на лінії RA4/TOCKI</td></tr>
<tr><td>PSA</td><td colspan="3">Біт призначення подільника:
0 – попередній подільник під’єднаний до TMR0;
1 – попередній подільник під’єднаний до сторожового таймера WDT</td></tr>
<tr><td rowspan="10">PS2,
PS1,
PS0</td><td colspan="3">Біти вибору коефіцієнта поділу попереднього подільника</td></tr>
<tr><td>PS2 PS1 PS0</td><td>Коефіцієнт поділу для TMR0</td><td>Коефіцієнт поділу для WDT</td></tr>
<tr><td>000</td><td>2</td><td>1</td></tr>
<tr><td>001</td><td>4</td><td>2</td></tr>
<tr><td>010</td><td>8</td><td>4</td></tr>
<tr><td>011</td><td>16</td><td>8</td></tr>
<tr><td>100</td><td>32</td><td>16</td></tr>
<tr><td>101</td><td>64</td><td>32</td></tr>
<tr><td>110</td><td>128</td><td>64</td></tr>
<tr><td>111</td><td>256</td><td>128</td></tr>
</table>

Він містить керуючі біти для конфігурації попереднього подільника, зовнішніх переривань, таймера, а також підтягуючих («pull-up») резисторів на виводах PORTB.

Коли попередній подільник не потрібний, його необхідно налаштувати на роботу з WDT з коефіцієнтом поділу 1 (PSA = 1, PS2 PS1 PS0 = 000).

2.10. Порти вводу-виводу.

Контролери PIC16F8X мають два порти: PORTA (5 бітів) і PORTB (8 бітів) із побітовим індивідуальним настроюванням на введення чи виведення.

Порт А (PORTA) – це 5-бітовий фіксатор, що відповідає виводам контролера RA<4:0>. Лінія RA4 має вхід тригера Шмітта і вихід з відкритим стоком. Усі інші лінії порту мають ТТЛ вхідні рівні й КМОН вихідні буфери. Адреса регістра порту А – 05h.

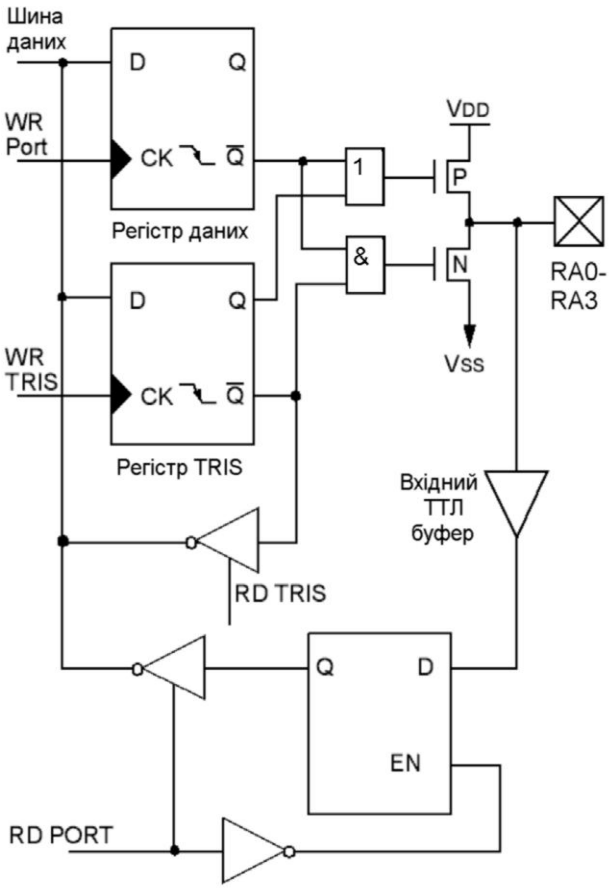


Рисунок 58. Схема ліній RA<3:0> порту А

Кожній лінії порту поставлений у відповідність біт напрямку передавання даних у керуючому регістрі TRISA, розташованому за адресою 85h. Якщо біт регістра TRISA має значення «1», то відповідна лінія буде встановлюватися на

ввід. Нуль перемикає лінію на вивід і одночасно виводить на неї вміст відповідного регістра-фіксатора порту.

При увімкненні живлення всі лінії порту за замовчуванням налаштовані на ввід. На рис. 58 зображена схема ліній RA<3:0> порту A. Зауважимо, що виводи порту мають захисні діоди до VDD і VSS.

Операція читання порту A зчитує стан виводів порту, а запис у порт змінює стан тригерів регістра даних. При скиданні та встановленні ліній порту слід бути уважними. При читанні порту зчитується стан вихідної лінії, а не внутрішньої засувки. Отже, якщо в якийсь момент часу вихід буде тимчасово «посаджений» на нуль, цей нуль і буде зчитаним, навіть коли лінія повинна виводити логічну одиницю.

Вивід RA4 мультиплексований з тактовим входом таймера TMR0. Схема лінії RA4 порту A наведена на рис. 59.

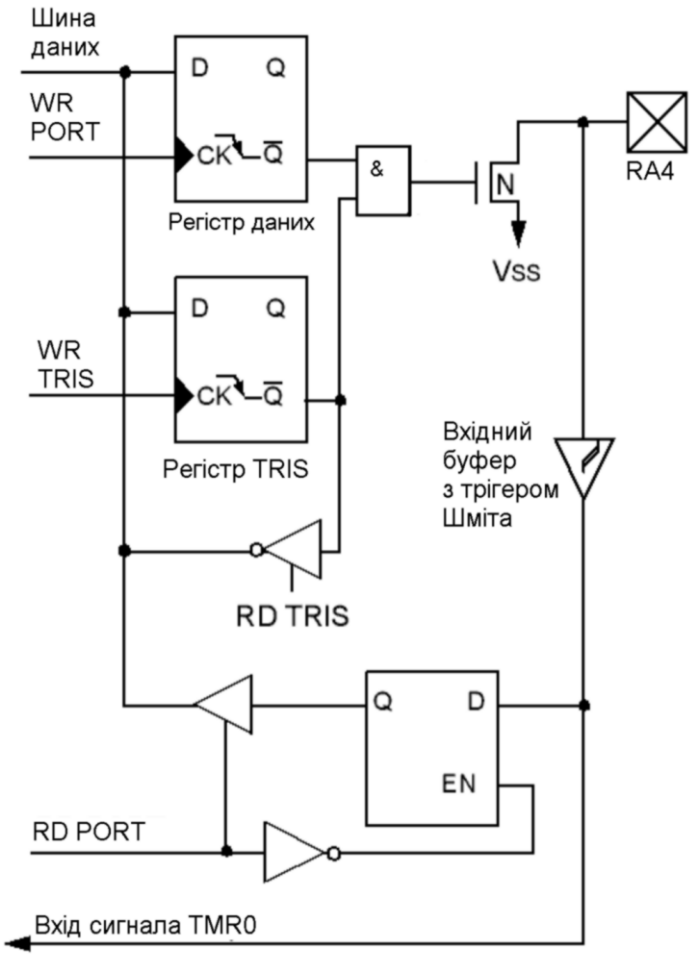


Рисунок 59. Схема лінії RA4 порту A

Порт B (PORTB) – це двонаправлений 8-бітовий порт, що відповідає виводам RB<7:0> контролера і розташований за адресою 06h. Кожній лінії порту поставлений у відповідність біт напрямку передавання даних, що зберігається в керуючому регістрі TRISB, розташованому за адресою 86h. Якщо біт керуючого регістра TRISB має значення «1», то відповідна лінія буде встановлюватися на ввід. Нуль перемикає лінію на вивід і виводить на неї вміст відповідного регістра. При вмиканні живлення всі лінії порту за замовчуванням налаштовані на ввід.

Вивід порту має захисний діод тільки до Vss.

Схеми ліній порту B наведено на рис. 60 і 61.

До кожної ніжки порту B під'єднане невелике активне навантаження (таке, що дає струм близько 10 мкА). Воно автоматично від'єднується, якщо цей вивід запрограмований як вихід. Керуючий біт $\overline{\text{RBPU}}$ регістра OPTION<7> може від'єднати (при $\overline{\text{RBPU}}$ = 1) всі навантаження. Скидання при вмиканні живлення також вимикає всі навантаження.

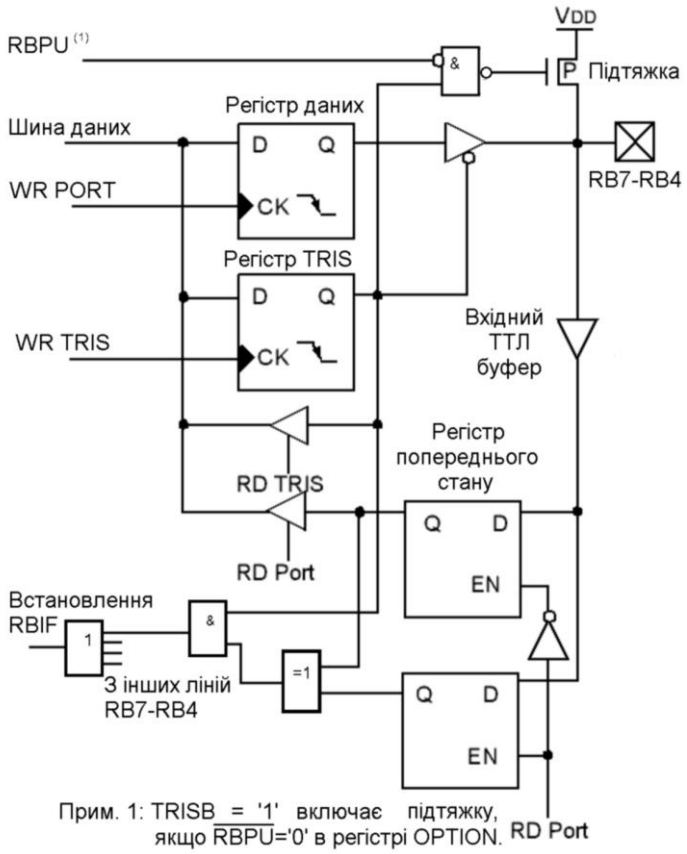


Рисунок 60. Схема ліній RB<7:4> порту B

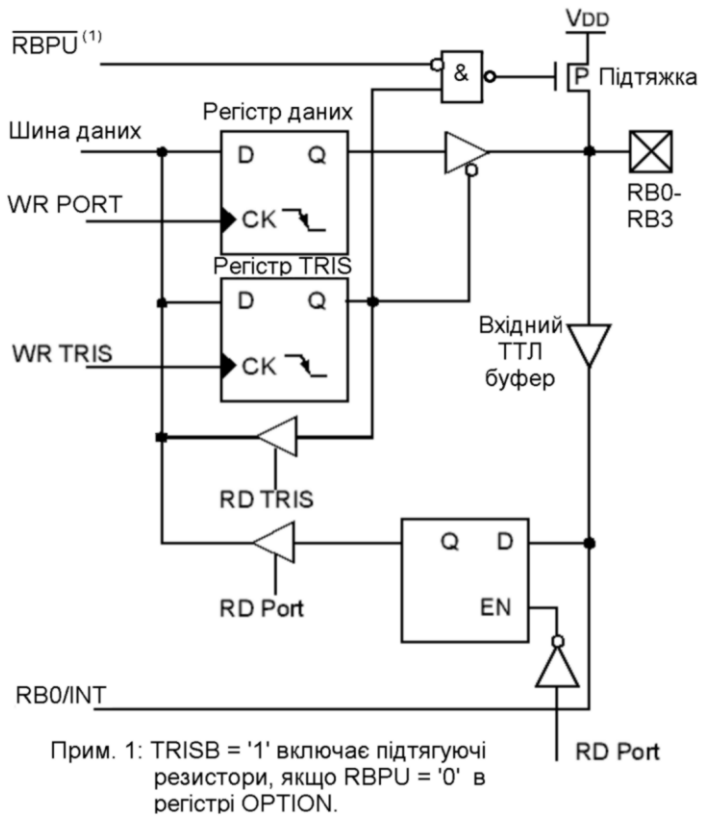


Рисунок 61. Схема ліній RB<3:0> порту B

Чотири лінії порту B (RB<7:4>) можуть викликати переривання при зміні значення сигналу на кожній з них. Якщо ці лінії налаштовані на ввід, то вони опитуються і запам’ятовуються в циклі читання Q1. Нове значення вхідного сигналу порівнюється зі старим у кожному командному циклі. При розбіжності значення сигналу на ніжці й у фіксаторі генерується запит переривання.

Виходи детекторів різниці ліній RB4, RB5, RB6, RB7 поєднуються по АБО і генерують переривання RBIF (запам’ятовується в регістрі INTCON<0>).

Будь-яка лінія, налаштована на вивід, участі в порівнянні не бере.

У підпрограмі опрацювання переривання варто скинути запит переривання одним із таких способів:

◊ прочитати (чи записати) порт B. Це зніме стан порівняння;

◊ обнулити біт RBIF регістра INTCON<0>.

Водночас слід мати на увазі, що умова зміни буде продовжувати встановлювати ознаку RBIF. Тільки читання порту B може усунути розбіжність і дозволить обнулити біт RBIF. Переривання по зміні стану і програмно

встановлювані внутрішні активні навантаження на цих чотирьох лініях можуть забезпечити простий інтерфейс, наприклад, із клавіатурою, з виходом із режиму SLEEP після натискання клавіш.

При організації двонаправлених портів необхідно враховувати особливості організації вводу-виводу даних МК. Будь-яка команда, що здійснює встановлення або скидання ліній порту, виконується як команда «читання – модифікація – записування». Наприклад, команди BCF і BSF зчитують порт повністю, модифікують один біт і виводять результат назад. Команда BSF PORTB, 5 (встановити в одиницю біт 5 порту B) спочатку зчитує значення всіх сигналів, що є в даний момент на виводах порту. Потім виконують дії над бітом 5, і нове значення байта записується у вихідні фіксатори. Якщо інший біт регістра PORTB використовується в якості двонаправленого вводу-виводу (наприклад, біт 0) і в даний момент він визначений як вхідний, то вхідний сигнал на цьому виводі буде зчитаний і записаний назад у вихідний тригер-фіксатор цього ж виводу, стираючи попередній стан. Доти, поки ця лінія залишається в режимі вводу, ніяких проблем не виникає. Якщо ж пізніше лінія 0 перемикнеться в режим виводу, її стан буде змінено порівняно з попереднім.

2.11. Модуль таймера PIC16X8X.

Структура модуля таймера-лічильника TIMER0 і його взаємозв’язок з регістрами TMR0 і OPTION зображені на рис. 62. TIMER0 є програмованим модулем і містить такі компоненти:

◊ 8-розрядний таймер-лічильник TMR0 з можливістю читання і записування;

◊ 8-розрядний програмно керований попередній подільник;

◊ мультиплексор для вибору внутрішнього чи зовнішнього тактового сигналу;

◊ схему вибору фронту зовнішнього тактового сигналу;

◊ формувач запиту переривання після переповнення регістра TMR0 (зі стану FFh у стан 00h).

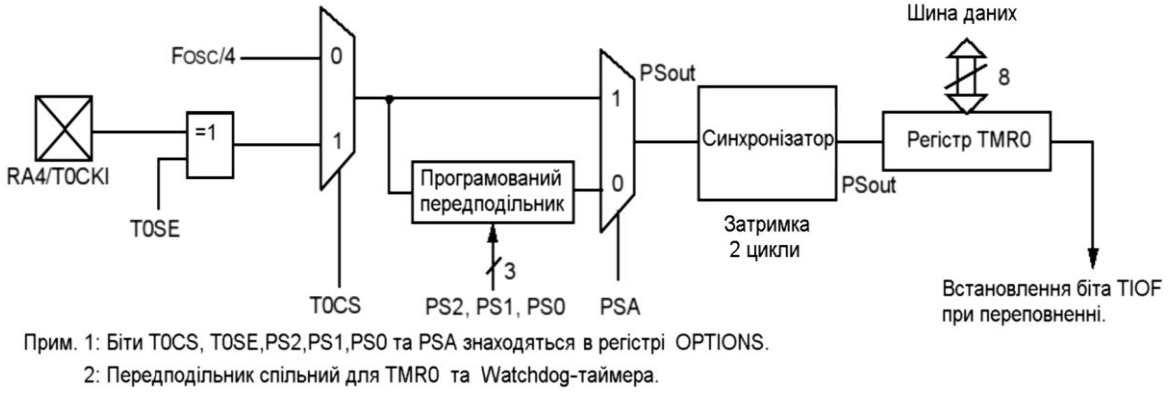


Рисунок 62. Структурна схема таймера-лічильника TMR0

Режим таймера вибирається шляхом скидання в нуль біта TOCS регістра OPTION <5>. У режимі таймера TMR0 інкрементується кожен командний цикл (без подільника). Після записування інформації в TMR0 інкрементування його почнеться після двох командних циклів. Це відбувається з усіма командами, що проводять запис чи модифікацію TMR0 (наприклад, MOVF TMR0, CLRF TMR0). Якщо потрібно перевірити, чи дорівнює TMR0 нулю без зупинки відліку, слід використовувати інструкцію MOVF TMR0,W.

Режим лічильника вибирається шляхом встановлення в одиницю біта TOCS регістра OPTION<5>. У цьому режимі регістр TMR0 буде інкрементуватися зростаючим або спадаючим фронтом на виводі RA4/TOCKI. Напрямок фронту визначається керуючим бітом TOSE у регістрі OPTION<4>. При TOSE = 0 буде обраний зростаючий фронт.

Попередній подільник може використовуватися разом з TMR0 чи з Watchdog-таймером. Під’єднання попереднього подільника контролює біт PSA регістра OPTION<3>. При PSA = 0 попередній подільник приєднається до TMRO, а його вміст програмі буде недоступний. Коефіцієнт поділу попереднього подільника програмується бітами PS2...PS0 регістра OPTIOM<2:0>.

Переривання по TMR0 здійснюється, коли відбувається переповнення регістра таймера/лічильника при переході від FFh до 00h. У момент переповнення встановлюється біт запиту TOIF у регістрі INTCON<2>. Дане переривання можна замаскувати бітом TOIE у регістрі INTCON<5>. Біт запиту

TOIF повинен бути скинутий програмно при обслуговуванні переривання. Переривання по TMR0 не може вивести процесор із режиму SLEEP тому, що таймер у цьому режимі не функціонує.

При PSA=1 попередній подільник буде приєднаний на вихід Watchdog-таймера. Можливі варіанти використання подільника зображені на рис. 63.

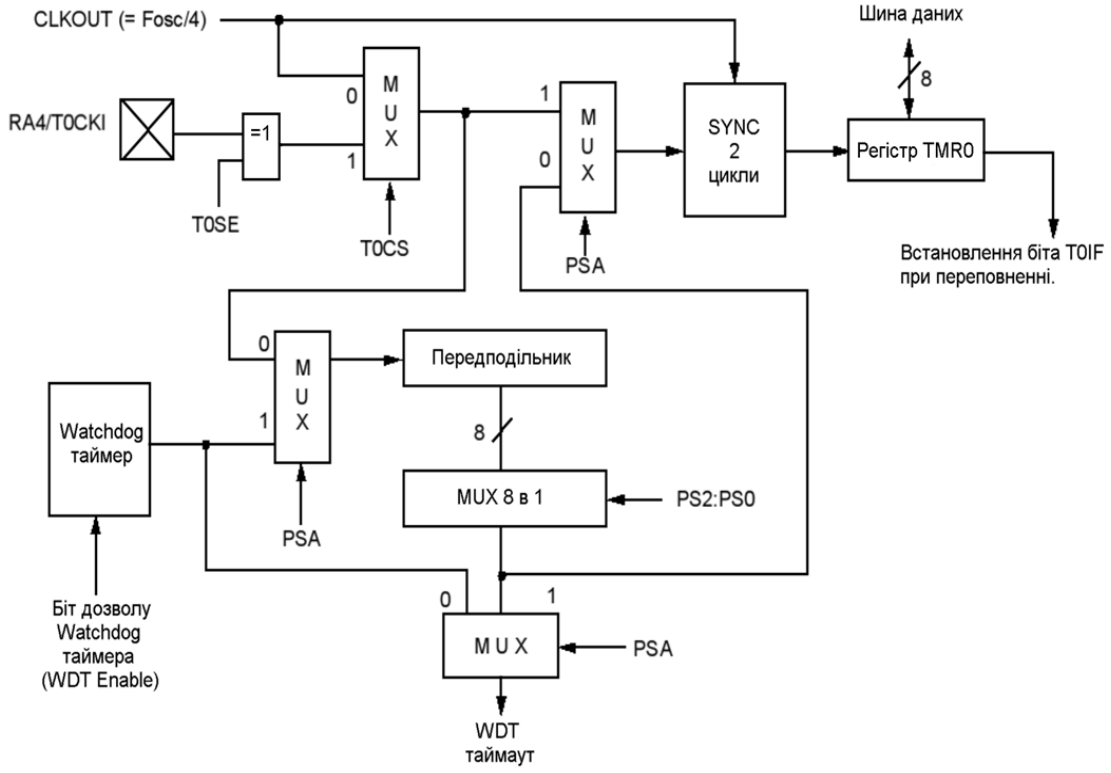


Рисунок 63. Структура й можливі варіанти використання подільника

При використанні попереднього подільника разом із TMR0 усі команди, що змінюють вміст TMR0, обнулюють передподільник. Якщо попередній подільник використовується разом з WDT, команда CLR WDT обнулює вміст попереднього подільника разом з WDT.

При використанні модуля TIMER0 у режимі лічильника зовнішніх подій необхідно враховувати, що зовнішній тактовий сигнал синхронізується внутрішньою частотою Fosc. Це призводить до появи затримання в часі фактичного інкрементування вмісту TMR0.

Синхронізація відбувається у момент закінчення 2-го і 4-го тактів роботи МК. Якщо передподільник не використовується, то для фіксації вхідної події

необхідно, аби тривалості високого й низького станів сигналу на вході RA4/TOCKI були не менше двох періодів тактової частоти Tosc плюс деяка затримка (≈20 нс).

Якщо модуль TIMER0 використовується разом із попереднім подільником, то частота вхідного сигналу ділиться асинхронним лічильником так, що сигнал на виході попереднього подільника стає симетричним. При цьому необхідно, щоб тривалості високого й низького рівнів сигналу на вході RA4/TOCKI були не менше 10 нс. Синхронізація сигналу відбувається на виході подільника, тому існує невелика затримка між фронтом зовнішнього сигналу і часом фактичного інкременту таймера-лічильника. Ця затримка знаходиться в діапазоні від 3 до 7 періодів тактового генератора.

2.12. Організація переривань PIC16F8X.

МК підгрупи PIC16X8X мають чотири джерела переривань:

◊ зовнішнє переривання з виводу RB0/INT;

◊ переривання від переповнення таймера/лічильника TMRO;

◊ переривання від зміни сигналів на лініях порту RB<7:4>;

◊ переривання в момент закінчення записування даних у EEPROM.

Усі переривання мають той самий вектор та адресу початку процедури обслуговування (0004h). Однак у керуючому регістрі переривань INTCON відповідним бітом-ознакою записується, від якого саме джерела надійшов запит переривання. Виняток складає переривання в момент завершення записування в EEPROM, ознака якого знаходиться в регістрі EECON1.

Регістр умов переривання (INTCON) є доступним для читання та записування.

Призначення бітів регістра наведено в таблиці 18, а формат регістра – на рис. 64. Переривання INT може вивести процесор з режиму SLEEP, якщо перед входом у цей режим біт INTE був встановлений в одиницю. Стан біта GIE також визначає, чи буде процесор переходити на підпрограму переривання після виходу з режиму SLEEP.

Таблиця 18. Біти регістра умов переривання

Назва	Призначення
GIE	Біт дозволу всіх переривань. Нуль вказує, що всі переривання заборонені. «1» дозволяє незамасковані переривання
EEIE	Біт дозволу переривання записування в EEPROM. Нуль вказує на заборону переривання в момент закінчення записування в EEPROM, одиниця – на дозвіл
TOIE	Біт дозволу переривання в момент переповнення TMR0. Нуль вказує на заборону переривання від TMR0, «1» – на дозвіл
INTE	Біт дозволу переривань із входу RB0/INT. Нуль вказує на заборону переривання, «1» – на дозвіл
RBIE	Біт дозволу переривань по зміні PORTB. Нуль вказує на заборонену переривань, «1» – на дозвіл
TOIF	Біт переривання за переповнення TMR0. Нуль вказує, що переповнення TMR0 не було, одиниця – що відбулося переповнення TMR0
INTF	Біт запиту переривання по входу RB0/INT. Нуль – переривання по входу RB0/INT відсутнє, «1» – переривання по входу RB0/INT відбулося
RBTF	Біт запиту переривання за зміною стану PORTB: нуль вказує, що на жодному зі входів RB7-RB4 стан не змінився зі звертання, «1» – хоча б на одному з входів RB7-RB4 стан змінився

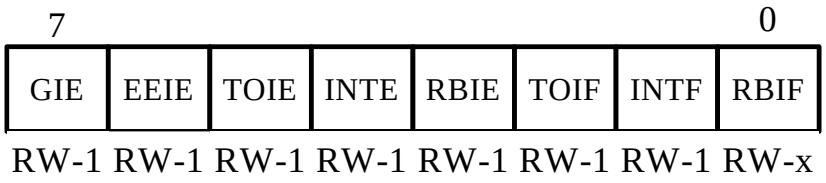


Рисунок 64. Формат регістра умов переривання

Варто звернути увагу, що скидання бітів запитів переривання здійснюється відповідною програмою опрацювання переривання.

Біт загального дозволу/заборони переривання GIE (INTCON <7>) дозволяє (GIE = 1) всі індивідуально незамасковані переривання, або забороняє їх (у випадку GIE = 0). У момент скидання біт GIE обнулюється. Кожне переривання окремо може бути додатково дозволено встановленням відповідного біта в регістрі INTCON. Скидання відповідного біта забороняє кожне переривання окремо.

Біт дозволу всіх переривань GIE скидається автоматично за таких обставин:

◊ за ввімкненням живлення;

◊ за зовнішнім сигналом /MCLR при нормальній роботі;

◊ за зовнішнім сигналом /MCLR у режимі SLEEP;

◊ за закінченням затримки таймера WDT при нормальній роботі;

◊ за закінченням затримки таймера WDT у режимі SLEEP.

Також біт GIE обнулюється на початку процедури обслуговування переривання, щоб заборонити повторне входження у переривання. Адреса повернення посилається в стек, а в програмний лічильник завантажується адреса 0004h. Час реакції на переривання для зовнішніх подій, таких, як переривання від лінії INT чи порту B, складає програмних п’ять циклів. Це на один цикл менше, ніж для внутрішніх подій, таких, як переривання у зв’язку з переповненням таймера TMRO.

У підпрограмі обслуговування переривання джерело переривання може бути визначене за відповідним бітом у регістрі INTCON. Ознака джерела переривання повинна бути програмно скинута у підпрограми обслуговування. Ознаки запитів переривань не залежать від відповідних маскуючих бітів і біта загального маскування GIE.

Команда повернення з переривання RETFIE завершує підпрограму переривання і встановлює біт GIE, щоб знову дозволити переривання. Логіка переривань контролера зображена на рис. 65.

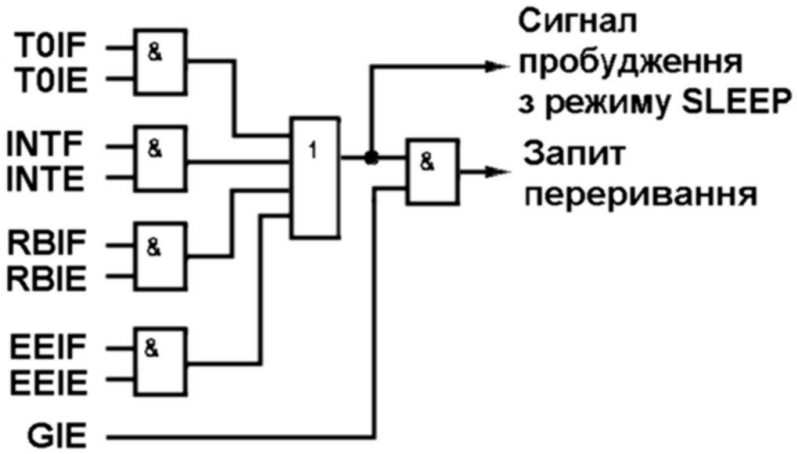


Рисунок 65. Логіка переривань мікроконтролера

Зовнішнє переривання на лінії RB0/INT здійснюється або за наростаючим (якщо в регістрі OPTION біт INTEDG = 1), або за спадаючим (якщо INTEDG = 0) фронтом. Коли фронт імпульсу з'являється на лінії INT, біт запиту INTF встановлюється в одиницю (INTCON<1>). Це переривання може бути замасковане скиданням керуючого біта INTE в нуль (INTCON<4>). Біт запиту INTF необхідно очистити підпрограмою обслуговування переривання перед тим, як знову дозволити його. Переривання INT може вивести процесор з режиму SLEEP, якщо перед входом у цей режим біт INTE був встановлений в одиницю. Стан біта GIE також визначає, чи буде процесор переходити на підпрограму переривання після виходу з режиму SLEEP.

Переповнення лічильника TMR0 (перехід зі стану FFh у стан 0h) встановлює в одиницю біт запиту T0IF (INTCON<2>). Це переривання може бути дозволене або заборонене встановленням біта маски TOIE (INTCON<5>). Скидання запиту T0IF – справа підпрограми обслуговування переривання.

Будь-яка зміна сигналу на одному з чотирьох входів порту RB<7:4> встановлює в одиницю біт RBIF (INTCON<0>). Це переривання може бути дозволено або заборонено встановленням біта маски RBIE (INTCON<3>). Скидання запиту RBIF – справа підпрограми обслуговування переривання.

Ознака запиту переривання в момент завершення записування даних в EEPROM, EEIF (EECON1<4>) встановлюється в одиницю в момент закінчення записування даних. Це переривання може бути замасковане скиданням біта EEIE (INTCON<6>).

2.13. Початкова ініціалізація та скидання у початковий стан.

2.13.1. Джерела скидання.

PIC16X8X має кілька джерел сигналу скидання:

◊ при ввімкненні живлення (Power-on Reset) POR;

◊ за зовнішнім сигналом MCLR;

◊ за сигналом Watchdog таймера (WDT Reset);

◊ за таймером увімкнення живлення;

◊ за таймером запуску генератора.

Сигнал скидання з джерел скидання формується за допомогою схеми, зображеної на рис. 66.

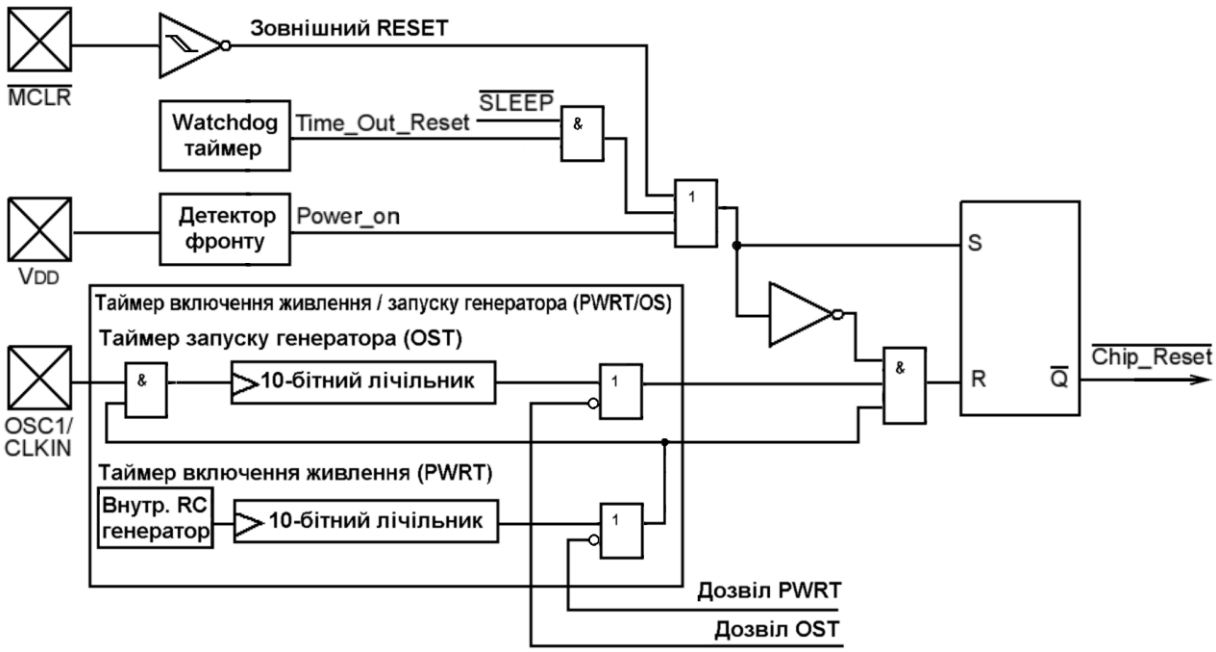


Рисунок 66. Схема формування сигналу скидання

Таймер увімкнення живлення (PWRT) дає фіксовану витримку часу в 72 мс (номінальне значення) при увімкненні живлення. Таймер працює на внутрішньому RC-генераторі. Він дозволяє дочекатися стабілізації напруги живлення в процесі запуску контролера.

Таймер запуску генератора дозволяє після увімкнення живлення або при виході з режиму зниженого енергоспоживання (SLEEP) дочекатися стабілізації частоти та режиму генерації кварцового чи керамічного резонатора. Таймер очікує надходження з генератора 1024 імпульси перед зняттям сигналу скидання.

Таймер увімкнення живлення і таймер запуску генератора можуть бути вимкнені у разі необхідності. Таймер запуску генератора, найчастіше, не використовується при роботі з RC генератором і при роботі з зовнішнім джерелом синхросигналів.

Таймер увімкнення живлення може не використовуватись у системах із зовнішнім формувачем сигналу скидання, наприклад, у системах із супервізором живлення.

2.13.2. Скидання при увімкненні живлення.

Кристал PIC16X8X має вбудований детектор увімкнення живлення. Коли напруга живлення перевищить рівень 1,2...1,7 В, формується сигнал скидання, що вмикає таймер увімкнення живлення запуску. Після закінчення витримки (близько 72 мс) вважається, що напруга досягла номіналу й запускається інший таймер витримки для стабілізації кварцового генератора. Програмований біт конфігурації дозволяє або забороняє затримування від згаданих вбудованих таймерів запуску. Затримка запуску змінюється залежно від кристалу, живлення й температури.

Таймер для стабілізації генератора відраховує 1024 імпульси від генератора, який почав роботу. Вважається, що кварцовий генератор за цей час увійшов у нормальний режим роботи. При використанні RC-генераторів затримка часу для стабілізації частоти не використовується.

Далі вмикається таймер чекання зовнішнього скидання MCLR. Це необхідно для випадків, коли потрібно синхронно запустити в роботу кілька PIC-контролерів через загальний для всіх сигнал MCLR. Якщо такого сигналу немає, то через час Tost формується внутрішній сигнал скидання і контролер починає працювати за програмою. Час Tost програмується бітами конфігурації в EEPROM.

Коли VDD наростає надто повільно і, попри всі затримки на запуск, живлення ще не досягло свого мінімального значення VDD(min) для нормального функціонування, рекомендується використовувати зовнішні RC-ланки для скидання за сигналом MCLR (рис. 67). В інших випадках необхідності в такому колі немає.

Рекомендується вибирати опір резистора R не більший за 40 кОм. Опір резистора R1, що обмежує струм через лінію, вибирають у межах 100 Ом – 1 кОм.

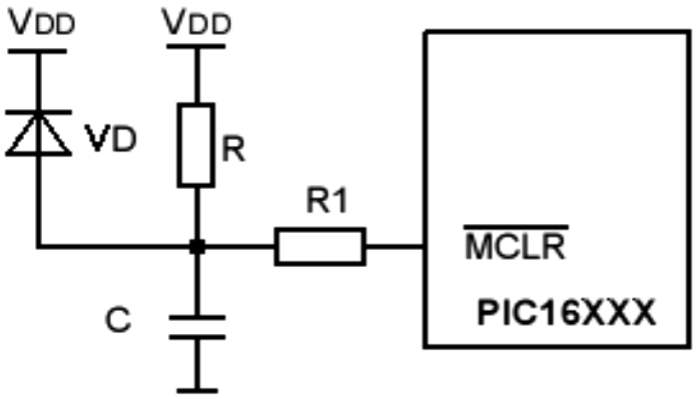


Рисунок 67. Зовнішнє коло скидання

2.13.3. Watcdog-таймер.

Watchdog-таймер (WDT) є повністю незалежним таймером, що працює незалежно від інших блоків мікросхеми. Робота таймера полягає в очікуванні протягом певного періоду звертання від основної програми і формуванні сигналу скидання після закінчення заданого інтервалу часу у випадку відсутності звертань.

Команди «CLRWDT» і «SLEEP» обнулюють WDT і попередній подільник, якщо він під'єднаний до WDT. Якщо сигнал скидання від WDT все ж відбувся, то одночасно обнулюється біт «TO» в регістрі статусу (STATUS). Формування сигналу скидання може бути заборонено записом у біт конфігурації WDTE. Така операція виконується на етапі програмування мікросхеми.

Номінальна тривалість затримування часу WDT складає 18 мс (без використання попереднього подільника). Вона залежить від температури, напруги живлення та особливостей кристала. Якщо потрібен більший інтервал затримування часу, то до WDT може бути під'єднаний внутрішній попередній подільник з коефіцієнтом поділу до 128, що програмується шляхом записування в регістр OPTION. У результаті можуть бути реалізовані затримки часу до 2,5 секунди. Зауважимо, що такий метод формування затримок не є точним – параметри генератора WDT залежать від напруги живлення, температури та інших факторів.

2.14. Режим зниженого енергоспоживання.

Вхід у режим SLEEP здійснюється командою SLEEP, за якою, якщо WDT дозволений, він скидається і починає відлік часу, скидається також біт «PD» у регістрі статусу (STATUS), біт «TO» встановлюється в «1», а генератор тактових сигналів вимикається. Порти вводу-виводу зберігають стан, який вони мали до входу в режим SLEEP.

Вихід з режиму SLEEP здійснюється в результаті:

◊ зовнішнього скидання – імпульсу низького рівня на виводі MCLR;

◊ скидання при спрацюванні WDT (якщо воно дозволене);

◊ переривання з виводу INT або при зміні стану лінії порту B, або при завершенні запису даних в EEPROM.

При першій вказаній події відбувається скидання всього пристрою. Дві інші події допускають продовження виконання програми.

Розглянемо вихід за зовнішнім скиданням. При увімкненні живлення біт «PD» у регістрі статусу (STATUS) встановлюється в «1», а при формуванні скидання іншого виду він залишається в попередньому стані. Обнулення біта може здійснюватися командою «SLEEP». Отже, він може бути використаний для ідентифікації стану процесора до скидання: чи знаходився процесор в режимі «SLEEP» (гарячий старт), чи було вимкнене живлення (холодний старт). Біт «TO» дозволяє визначити, чим був викликаний вихід із режиму SLEEP – зовнішнім сигналом на виводі MCLR чи спрацюванням WDT. Проте такий метод виходу призводить до втрати попереднього (до засинання) стану деяких регістрів, у тому числі лічильника команд. Тому ним не варто користуватися без особливої необхідності.

Для виходу з режиму SLEEP за допомогою переривань, які повинні бути дозволені встановленням відповідної маски в регістрі INTCON, буде виконуватися команда, що йде за командою SLEEP, якщо біт загального дозволу переривань GIE обтулений. В іншому випадку керування буде передано в підпрограму обслуговування переривань зі збереженням у стеку адреси команди, наступної за командою SLEEP.

2.15. Генератор та синхронізація.

2.15.1. Типи генераторів.

PIC16X8X може працювати з чотирма типами генераторів. Програміст може, користуючись двома конфігураційними бітами (FOSC1 і FOSC0), обрати один із режимів:

◊ RC – генератор.

◊ LP – низькочастотний кварцовий (керамічний) резонатор. Частоти резонансу до 100 – 200 кГц. Найчастіше використовують резонатори на 32 кГц та 200 кГц.

◊ XT – середньочастотний резонатор. Частоти резонансу від 100 – 200 кГц до 4 МГц.

◊ HS – високочастотний резонатор. Частоти резонансу від 4 МГц і вище.

2.15.2. Робота з кварцовим резонатором.

Схема увімкнення кварцового резонатора показана на рис. 68.

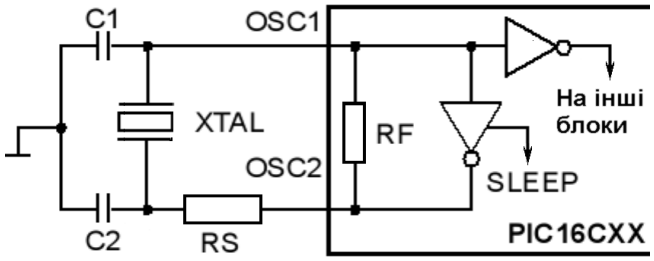


Рисунок 68. Під’єднання кварцового резонатора

При роботі в режимах XT, LP, HS керамічний або кварцовий резонатор під’єднується між виводами OSC1/CLKIN та OSC2/CLKOUT.

Ємності C1 та C2 призначені для підвищення надійності запуску і роботи генератора: збільшення ємності призводить до підвищення стабільності, проте збільшує час запуску, який слід враховувати при виході з режиму зниженого енергоспоживання.

Типове значення ємності для резонаторів на частоту від 100 кГц і вище – близько 33 пФ, для резонаторів на 32 кГц – 68 пФ. Резистор RS може бути

потрібний деяким типам резонаторів для гасіння вищих гармонік коливань.

Залежно від режиму роботи (XT, LP, HS) змінюється значення внутрішнього резистора RF. Тому підбір типу резонатора необхідно здійснювати одночасно з вибором режиму роботи, користуючись при цьому документацією виробника мікросхеми.

2.15.3. Синхронізація від зовнішніх джерел.

PIC16X8X може також синхронізуватися від зовнішніх джерел сигналу. Типова схема увімкнення в такому режимі роботи зображена на рис. 69. При роботі в цьому режимі обов'язково повинен бути обраний режим синхронізації від кварцового резонатора (XT, LP або HS), у випадку вибору RC-резонатора можливе пошкодження кристала.

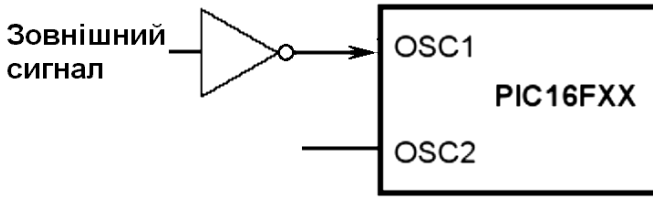


Рисунок 69. Синхронізація від зовнішніх джерел

2.15.4. RC-генератор.

Коли не пред'являються високі вимоги до точності відліку часу, зручно для тактування мікросхеми використовувати RC-генератор. Схема увімкнення RC-генератора зображена на рис. 70.

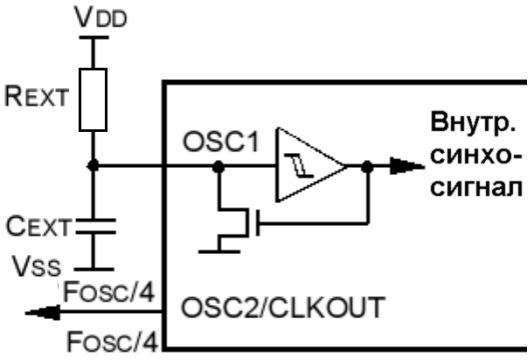


Рисунок 70. RC – генератор

На виводі OSC2/CLKOUT видається сигнал з частотою, що дорівнює 1/4 частоти генератора. Сигнал може бути використаний для синхронізації інших мікросхем.

Типові значення Rext = 5 – 100 кОм, Cext ≥ 20 пФ. Наближені середні значення частоти синхронізації Fosc та похибка встановлення частоти при напрузі живлення 5 В та температурі 25°С наведені в таблиці 19.

Таблиця 19. Середні значення частоти синхронізації

Cext	Rext	Fosc	Похибка	Cext	Rext	Fosc	Похибка
20 пФ	3,3 к	4,68 МГц	± 27%	100 пФ	10 к	620 кГц	± 30%
	5,1 к	3,94 МГц	± 25%		100 к	90,25 кГц	± 26%
	10 к	2,34 МГц	± 29%	300 пФ	3,3 к	524 кГц	± 28%
	100 к	250 кГц	± 33%		5,1 к	415 кГц	± 30%
100 пФ	3,3 к	1,49 МГц	± 25%		10 к	270 кГц	± 26%
	5,1 к	1,12 МГц	± 25%		100 к	25,4 кГц	± 25%

Для значень Rext, менших за 2,2 кОм, генератор може працювати нестабільно або не запускатися взагалі. При дуже великих значеннях Rext (наприклад, 1 МОм) генератор стає чутливим до завад, вологості й струму через ізоляцію корпусу.

2.16. Конфігурація та захист.

2.16.1. Індентифікаційний код.

PIC16X8X має чотири слова, розташовані за адресами 2000h-2003 h. Вони призначені для зберігання ідентифікаційного коду (ID), контрольної суми або іншої інформації. Разом із конфігураційним словом вони можуть бути прочитані та записані лише за допомогою програматора. Доступу за командами програми до них немає.

Якщо кристал захищений, користувачу рекомендується використовувати для ідентифікації тільки молодші сім бітів кожного слова ідентифікаційного коду, а в старші біти записувати «0».

2.16.2. Конфігураційне слово.

Призначення бітів слова конфігурації наведено в таблиці 20, а його формат – на рис. 71.

Таблиця 20. Призначення бітів слова конфігурації

Позначення	Призначення
CP	Біт захисту коду. Одиниця вказує, що захист коду увімкнений, нуль – на відсутність захисту
PWRTE	Біт дозволу витримки часу після увімкнення живлення. Одиниця вказує на наявність витримки в 72 мс при увімкненні, нуль – на відсутність витримки. Біт не впливає на роботу таймера запуску генератора
WDTE	Біт дозволу роботи Watchdog-таймера. Одиниця вказує на дозвіл роботи Watchdog-таймера, нуль – на заборону скидання по Watchdog-таймеру
FOSC1, FOSC0	Біти вибору типу генератора. Усі комбінації крім FOSC1,FOSC0 = 11 вказують на роботу з кварцовим або керамічним резонатором: Fosc1, fosc0 = 00 задає режим з низькочастотним резонатором (lp); FOSC1, FOSC0 = 01 задає режим з середньочастотним резонатором (XT); FOSC1, FOSC0 = 01 задає режим з високочастотним резонатором (HS); FOSC1, FOSC0 = 01 задає роботу RC-генератором

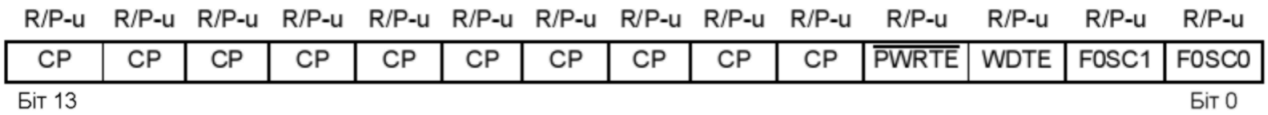


Рисунок 71. Формат слова конфігурації

PIC16F84 має п’ять бітів конфігурації, що зберігаються в пам’яті програм і встановлюються на етапі програмування кристалу. Ці біти можуть бути запрограмовані (читаються як «0») або залишені незапрограмованими

(читаються як «1») для вибору необхідного варіанта конфігурації мікросхеми. Вони розташовані в пам'яті програм за адресою 2007 h. Слід зауважити, що ця адреса знаходиться поза областю кодів і є програмно недоступною.

Програмний код, записаний на кристалі, може бути захищений від зчитування за допомогою встановлення біта захисту (CP) у слові конфігурації в нуль. Якщо встановлено захист, то біт CP можна стерти тільки разом зі стиранням вмісту кристала.

2.17. Система команд PIC-контролерів серії PIC16X8X.

Кожна команда є 14-розрядним словом, що містить поле коду операції OPCODE і поле операндів. Система команд включає команди роботи з байтами і бітами, команди керування та операції з константами.

Усі команди виконуються протягом одного командного циклу, крім таких двох випадків:

◊ виконання команд умовних переходів, якщо результат перевірки умови – істина,

◊ лічильник команд змінив значення у результаті виконання команди.

У цих випадках команда виконується за два цикли з виконанням другого циклу як NOP.

Команди однокристальної мікроЕОМ наведено в таблиці 21.

У таблиці для команд роботи з байтами F позначає регістр, з яким виконується дія, W – операційний регістр. У форматі коду команди використовуються позначення:

- fff ffff – адреса регістра, над вмістом котрого виконується дія;
- d – вид регістра призначення. Якщо d = 0, результат записується в регістр W, якщо d = 1 – у регістр F, заданий у команді;
- bbb – номер біта, з яким відбувається робота;
- kkkk kkkk, kkk kkkk kkkk – 8- або 11-бітову константу чи ідентифікатор.

Таблиця 21. Система команд мікроЕОМ PIC16x8x

Позначення	Функція	Цикли	Код команди	Біти стана	Прим.
1	2	3	4	5	6
ADDLW	Додавання константи і W	1	11 111x kkkk kkkk	C, DC, Z	–
ADDWF	Додавання W с F	1	00 0111 dfff ffff	C, DC, Z	1, 2
ANDLW	Логічне І константи і W	1	11 1001 kkkk kkkk	Z	–
ANDWF	Логічне І W і регістра F	1	00 0101 dfff ffff	Z	1, 2
BCF	Скидання біта в регістрі F	1	01 00bb bfff ffff	–	1, 2
BSF	Встановлення біта в регістрі F	1	01 01bb bfff ffff	–	1, 2
BTFSC	Пропустити команду, якщо біт у F дорівнює нулю	1 (2)	01 10bb bfff ffff	–	3
BTFSS	Пропустити команду, якщо біт у F дорівнює одиниці	1 (2)	01 11bb bfff ffff	–	3
CALL	Виклик підпрограми	2	10 0kkk kkkk kkkk	–	–
CLRF	Скидання регістра F	1	00 0001 1fff ffff	Z	2
CLRW	Скидання регістра W	1	00 0001 0xxx xxxx	Z	–
CLRWDT	Скидання Watchdog-таймера WDT	1	00 0000 0110 0100	–	–
COMF	Інверсія регістра F	1	00 1001 dfff ffff	Z	1, 2
DECF	Декремент регістра F	1	00 0011 dfff ffff	Z	1, 2

Продовження таблиці 21

1	2	3	4	5	6
DECFSZ	Декремент F, пропустити команду, якщо F став дорівнювати «0»	1 (2)	00 1011 dfff ffff	–	1, 2, 3
GOTO	Перехід за адресою	2	10 1kkk kkkk kkkk	–	–
INCF	Інкремент регістра F	1	00 1010 dfff ffff	Z	1, 2
INCFSZ	Інкремент F, пропустити команду, якщо F став дорівнювати «0»	1 (2)	00 1111 dfff ffff	–	1, 2, 3
IORLW	Логічне АБО константи і W	1	11 1000 kkkk kkkk	Z	–
IORWF	Логічне АБО W і F	1	00 0100 dfff ffff	Z	1, 2
MOVF	Пересилання регістра F	1	00 1000 dfff ffff	Z	1, 2
MOVLW	Пересилання константи в W	1	11 00xx kkkk kkkk	–	–
MOVWF	Пересилання W у F	1	00 0000 1fff ffff	–	–
NOP	Команда «немає операції»	1	00 0000 0xx0 0000	–	–
OPTION	Завантаження регістра OPTION	1	00 0000 0110 0010	–	–
RETFIE	Повернення з переривання	2	00 0000 0000 1001	–	–
RETURN	Повернення з підпрограми	2	00 0000 0000 1000	–	–
RETLW	Повернення з підпрограми з завантаженням константи у W	2	11 01xx kkkk kkkk	–	–

Закінчення таблиці 21

1	2	3	4	5	6
RLF	Зсув F вліво через ознаку перенесеня	1	00 1101 dfff ffff	C	1, 2
RRF	Зсув F вправо через ознаку перенесення	1	00 1100 dfff ffff	C	1, 2
SLEEP	Перехід у режим SLEEP	1	00 0000 0110 0011	–	–
SUBLW	Віднімання W з константи	1	11 110x kkkk kkkk	C, DC, Z	–
SUBWF	Віднімання W з F	1	00 0010 dfff ffff	C, DC, Z	1, 2
SWAPF	Обмін місцями тетрад в F	1	00 1110 dfff ffff	–	1, 2
TRIS	Завантаження регістра TRIS	1	00 0000 0110 0fff	–	–
XORLW	Виключаюче АБО константи і W	1	11 1010 kkkk kkkk	Z	–
XORWF	Виключаюче АБО W і F	1	00 0110 dfff ffff	–	1, 2

Якщо модифікується регістр вводу-виводу (наприклад, MOVF PORTB,1), то використовується значення, що зчитується з виходів. Наприклад, якщо у вихідній засувці порту, увімкненого на введення даних, знаходиться «1», а зовнішній пристрій формує на цьому виході «0», то в цьому розряді даних буде записаний «0». Якщо операндом команди є вміст регістра TMR0 (і, якщо припустимо, d = 1), то попередній подільник, якщо він під’єднаний до TMR0, буде скинутий.

Якщо в результаті виконання команди змінюється лічильник команд або виконується перехід у результаті перевірки умови, то команда виконується за два цикли. Другий цикл виконується як NOP.

2.18. Програмування PIC-контролерів із використанням програмного симулятора «PIC Simulator IDE».

2.18.1. Програмний симулятор PIC Simulator IDE.

Програма призначена для відлагодження програм для контролерів microPIC компанії Microchip Technology.

PIC Simulator IDE – це програмне забезпечення, що включає в себе графічне середовище розробки, симулятор, компілятор, асемблер, дизасемблер, відлагоджувач (рис. 72). В даний час симулятор підтримує близько 80 моделей 8-бітних мікроконтролерів від Microchip архітектури Midrange, що належать серіям PIC12F, PIC16F і PIC10F (обмежена підтримка).

Ключовою особливістю програми PIC Simulator IDE є широкі можливості із взаємодії відлагоджуваного коду з віртуальною периферією, що значно зменшує час, який витрачається на розроблення програм, і робить зайвим використання налагоджувальних плат з підключеними зовнішніми пристроями.

Програма містить безліч спеціальних модулів, що імітують роботу різних електронних пристроїв, наприклад, терміналу зв’язку, символьного рідкокристалічного екрана, чотириканального генератора, трифазного крокового двигуна, матриці клавіатури, семисегментних індикаторів, осцилографа, «апаратного» UART, цифрового термометра та іншого обладнання (рис. 73, 74).

Крім того, додаток дає можливість подивитися терморегулятори обраного мікроконтролера для моделювання цифрових і аналогових входів, призначити точки зупину, редагувати FLASH- і EEPROM-пам’ять контролера, змінювати конфігураційні робочі інструменти й т.д. Основні розділи меню містять велику кількість різних режимів роботи, налаштувань та додаткових модулів, дозволяючи користувачам самостійно вибирати варіанти використання даного програмного забезпечення.

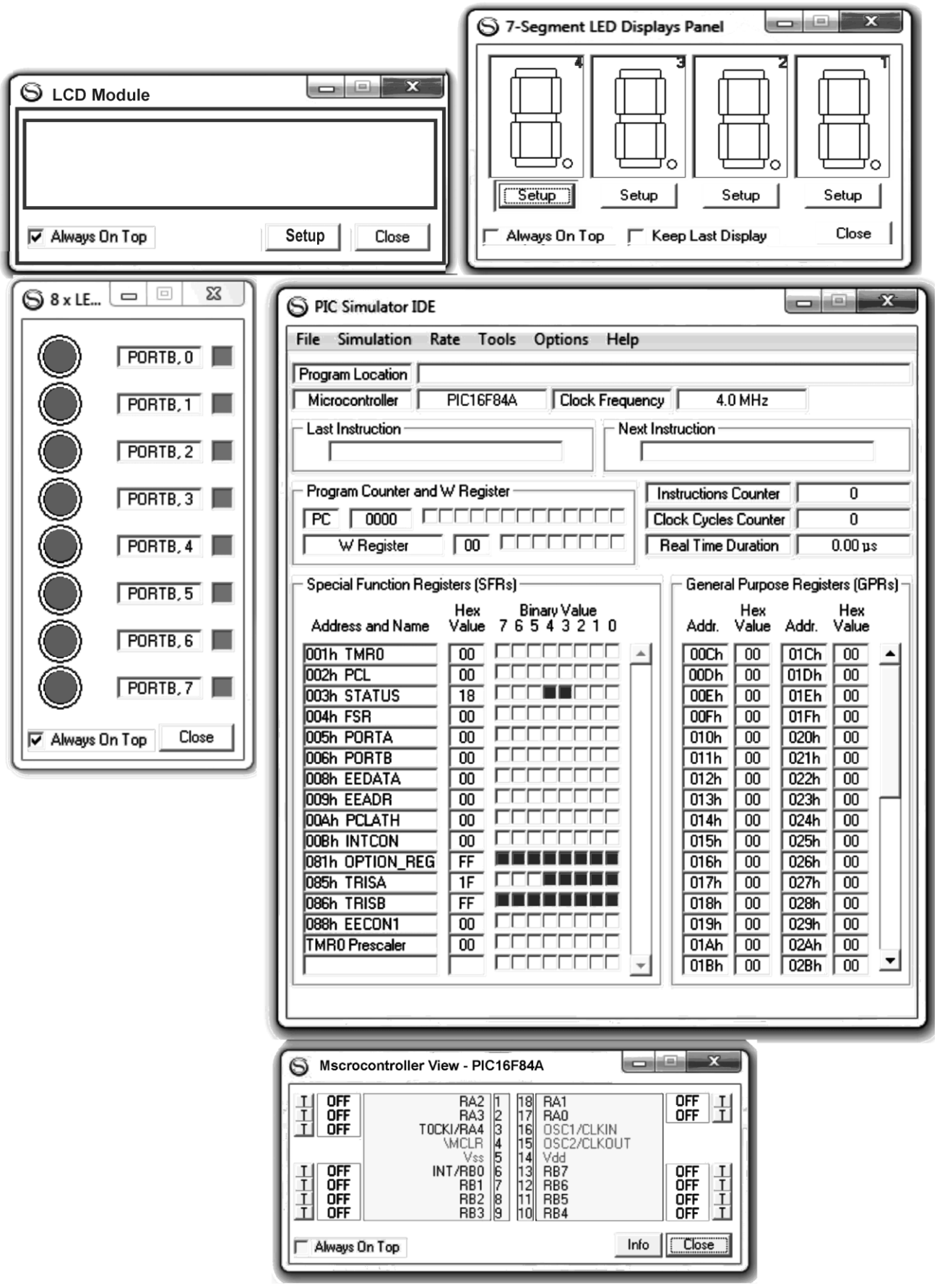


Рисунок 72. Графічне середовище програмного симулятора PIC Simulator IDE

У головному вікні програми зазначено назву робочої програми і шлях до неї, тип обраного мікроконтролера і частота кварцового генератора, яка необхідна для відображення даних про час виконання команди або програми, але не впливає на швидкість налагодження коду в PIC Simulator IDE. Тут же наведено стан керуючих і спеціальних регістрів обраного контролера, які можна змінювати в ході виконання програми.

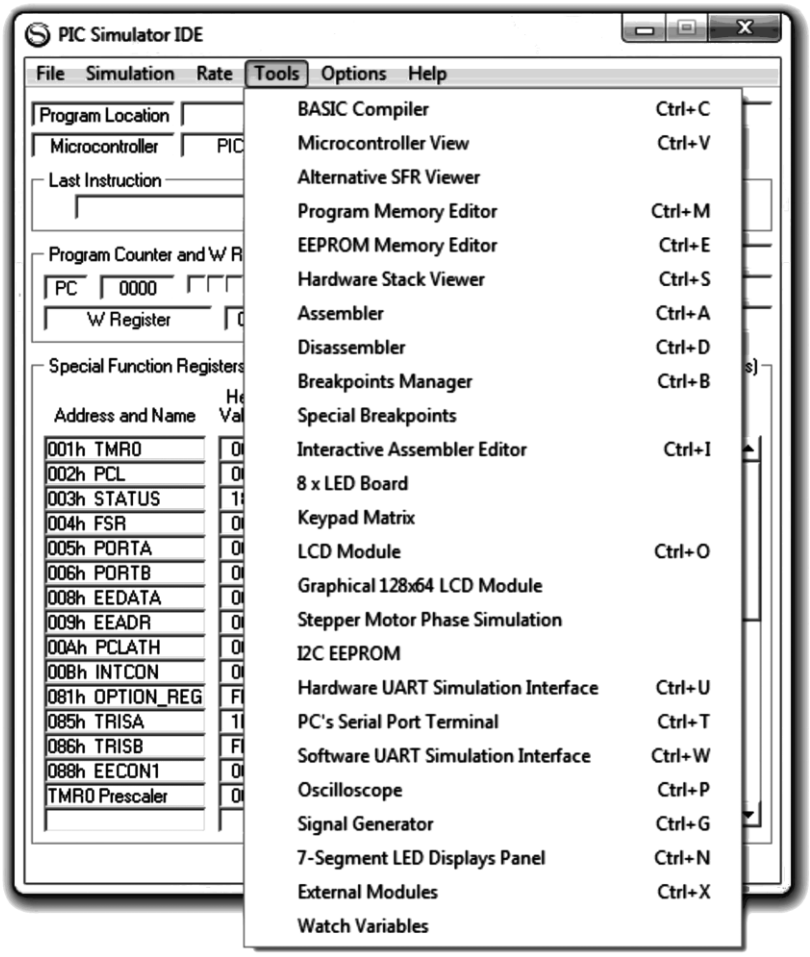


Рисунок 73. Перелік модулів симулятора, що імітують роботу електронних пристроїв

Додаток PIC Simulator IDE працює з файлами програм, написаних мовою асемблер (розширення *.asm), і файлами дампів пам’яті (розширення *.hex), підготовленими для завантаження в контролер.

2.18.2. Встановлення симулятора і робота з програмою.

Необхідно розпакувати архів з програмою, знайти файл setup.exe і

запустити його. На запитання, що з'являються під час встановлення, необхідно тільки натискати кнопки OK, Next або Continue.

Після завершення установки програма «PIC Simulator IDE» з'явиться в меню «Програми».

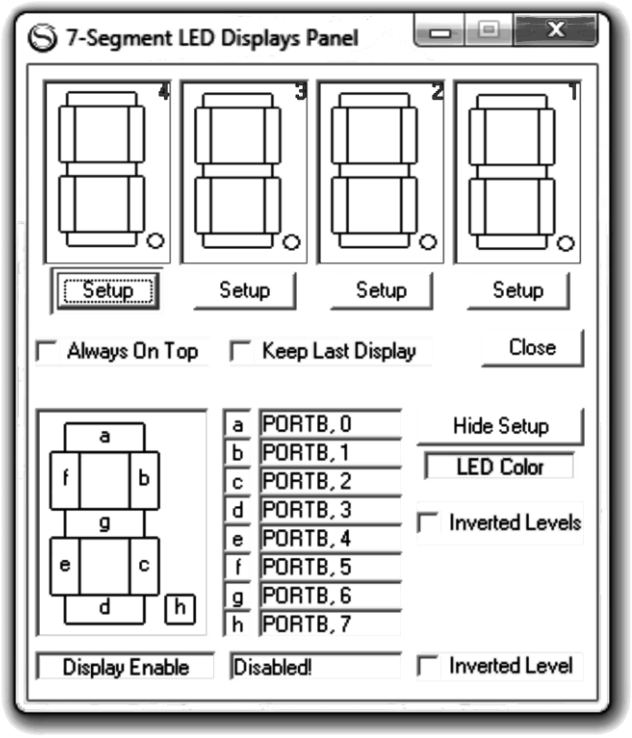


Рисунок 74. Модуль 7-сегментного індикатора стимулятора

Рекомендується вивести ярлик цієї програми на робочий стіл. Запустивши PIC Simulator IDE, побачимо основне вікно цієї програми (рис. 75). У його верхній частині знаходяться різні меню, через які можна отримати доступ до різних основних і додаткових модулів програми (на рис. 1 позначено як «1»).

Далі, в рядку *Program Location*, вказано шлях в обраній програмі, її ім'я (на рис. 1 – «2»). Нижче, в рядку *Microcontrollers*, відображається тип обраного мікроконтролера (на рис. 1 – «3»).

У нижній частині вікна є дві панелі (позначені як «4» і «5»). У них відображається вміст спеціальних і керуючих регістрів МК.

У комплект поставки програми симулятора входить кілька прикладів роботи з програмою. Приклади написані на Basic, компілятор якого вбудований в програму-симулятор.

Усі наведені в описі приклади програм розташовані в папці, в яку встановлено програму PIC Simulator IDE. За замовчуванням, ця папка розташовується за адресою: C: \Program Files\PIC Simulator IDE.

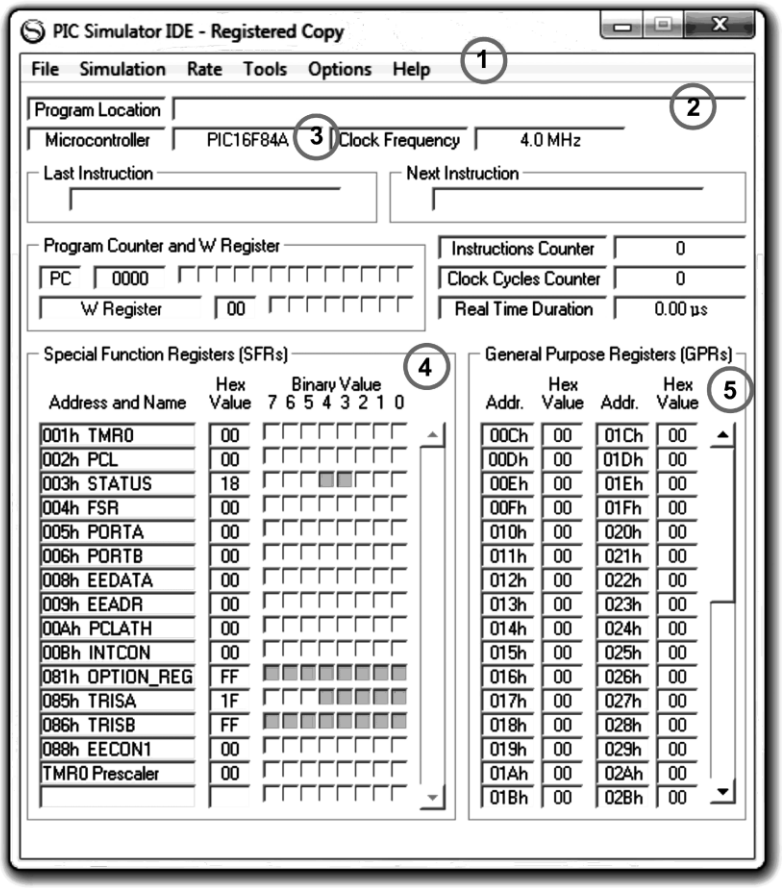


Рисунок 75. Основне вікно програми PIC Simulator IDE

Крім файлів програм на Basic, в цій папці зберігаються вже відкомпільовані файли на Асемблері і файли дампів пам’яті, підготовлені для безпосереднього завантаження в МК.

Файли на Асемблері з розширенням .asm згенеровані вбудованим компілятором Basic. Файли дампів пам’яті з розширенням .hex згенеровані вбудованим Асемблером.

Послідовність роботи з програмним симулятором наступний:

- запуск програми PIC Simulator IDE;
- вибір типу мікроконтролера, для якого написана програма;
- вибір частоти кварцового генератора (впливає тільки на відображувані

програмою дані про час виконання програми або команди, але не на швидкість роботи програми, що налагоджується в PIC Simulator IDE);

● завантаження програми у вигляді hex-файлу або запуск вбудованого компілятора мови асемблера й написання в ньому потрібної програми;

● вибір потрібних модулів віртуальних пристроїв;

● вибір швидкості й режиму роботи програми симулятора;

● запуск процесу симуляції роботи програми на обраному МК.

Якщо потрібно скористатися для роботи з симулятором власною програмою або внести зміни в уже розроблену, необхідно створити або завантажити для цього файл асемблера, з якого після компіляції буде створений необхідний для роботи з симулятором hex-файл.

Для цього:

1. Натиснути Options | Assembler. Відкриється вікно компілятора Assembler – UNTITLED (рис. 76);

2. У вікні Assembler натиснути опцію File. Розкриється закладка (рис. 77), з якої для створення нового файлу потрібно натиснути New, а для завантаження вже створеного – OPEN.

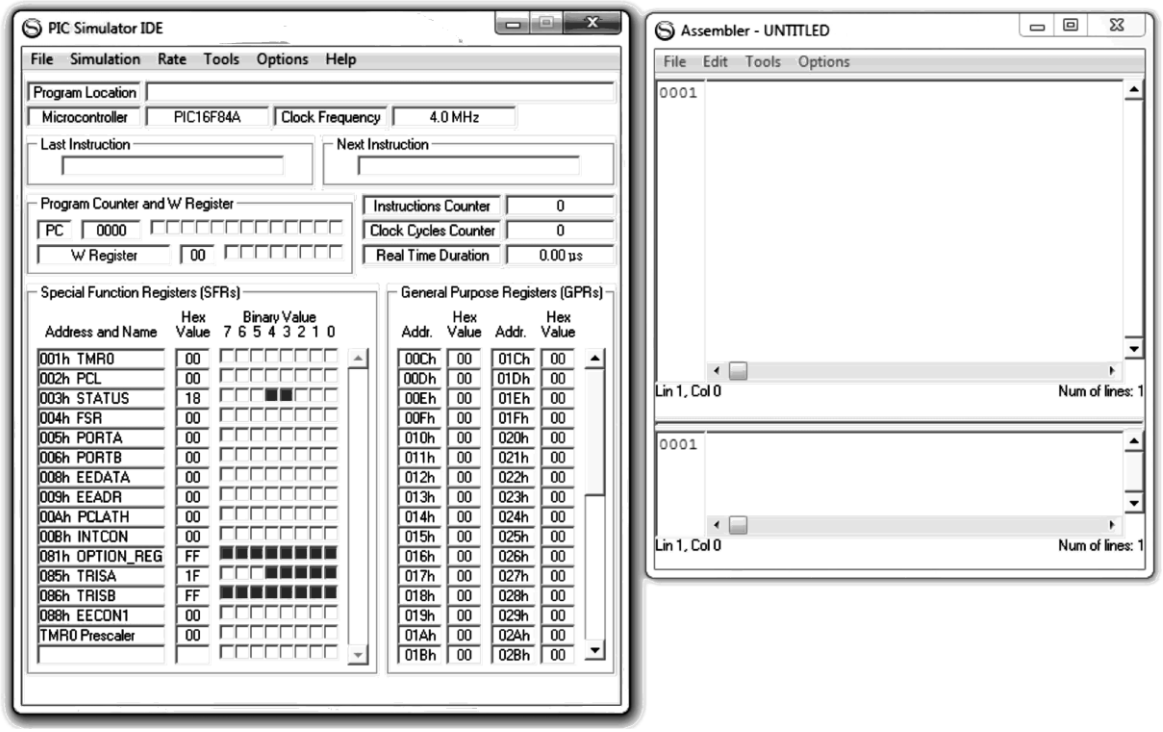


Рисунок 76. Вікно симулятора з відкритим вікном Assembler

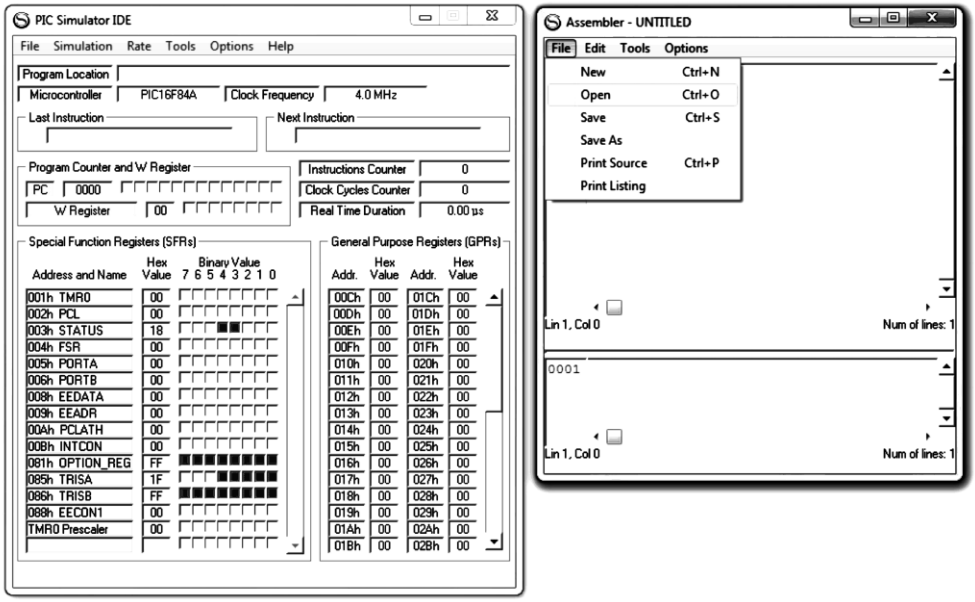


Рисунок 77. Завантаження існуючого або створення нового файлу Асемблера

3. Після вибору і завантаження файлу (наприклад, rb0int.asm), його текст з’явиться у вікні Assembler (рис. 78).

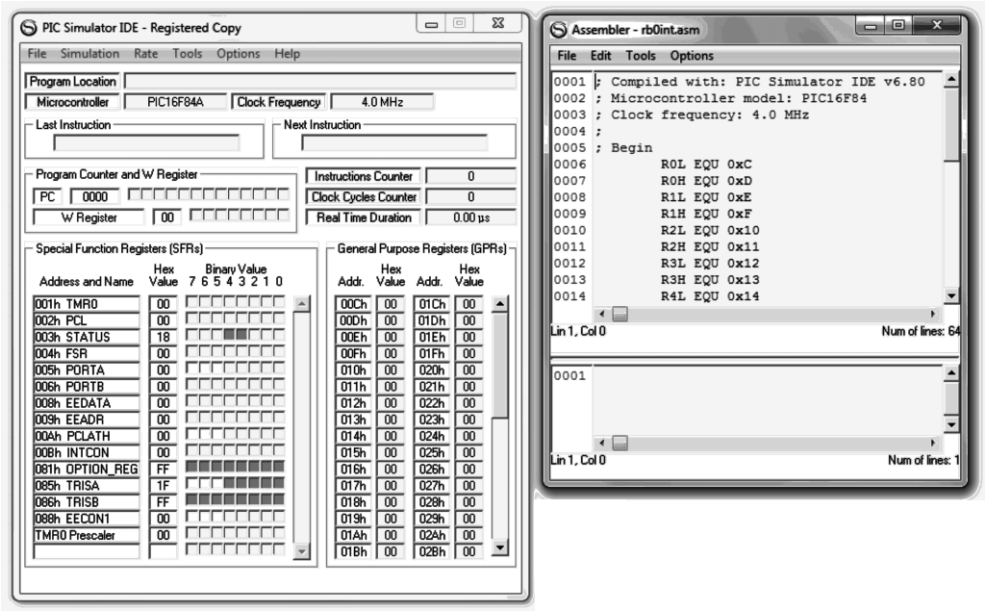


Рисунок 78. Завантаження файлу rb0int.asm

4. Для компіляції створеного або завантаженого і потім зміненого файлу потрібно натиснути Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з’явиться лістинг скомпільованого файлу і, одночасно, за відсутності помилок, буде створений однойменний hex-файл (рис. 79).

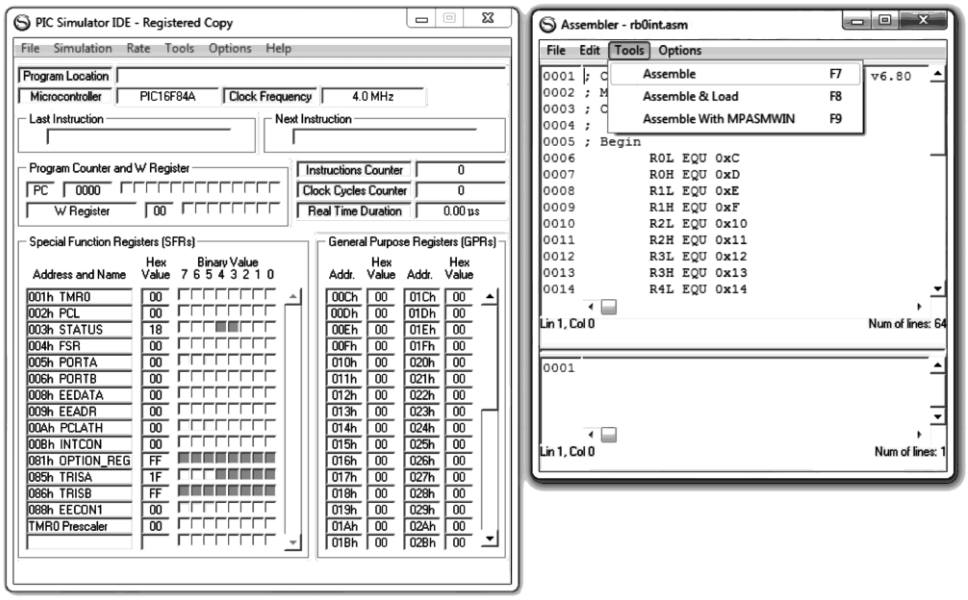


Рисунок 79. Вікно Assembler з лістингом скомпільованого файлу

2.18.3. Приклад 1.

Програма забезпечує введення даних з чотирьох молодших розрядів порту B, зсув їх на чотири розряди вліво і виведення через чотири старші розряди цього ж порту.

Текст програми з файлу «Input\output.asm» має наступний вигляд:

```
; Begin
     ORG 0x0000
     BCF PCLATH,3
     BCF PCLATH,4
BSF STATUS,RP0
     MOVLW 0x80
     MOVWF OPTION_REG
     MOVLW 0x0F
     MOVWF TRISB
     BCF STATUS,RP0
SHIFTING:
     SWAPF PORTB,W
     MOVWF PORTB
     GOTO SHIFTING
END
```

Для виконання цієї програми в PIC Simulator ID необхідно:

1. Запустити PIC Simulator IDE.

2. Натиснути Options | Select Microcontroller.

3. Вибрати PIC16F84 і натиснути кнопку Select.

4. Натиснути Tools і у вікні, що розкриється, вибрати «Assembler».

Відкриється вікно компілятора «Assembler – UNTITLED» (рис. 76).

5. Набрати текст програми Прикладу 1 у вікні «Assembler».

6. Натиснути Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з'явиться лістинг відкомпільованого файлу (рис. 80).

7. Одночасно, за відсутності помилок, буде створений файл «Input\output.hex», для якого можна вибрати ім'я та шлях для записування. Записати його, наприклад, на «Робочий стіл» комп'ютера.

8. Вибрати File|Load Program і завантажити створений файл «Input\output.hex».

9. Натиснути Tools|8 x LED Board. Відкриється вікно з панеллю, що містить вісім світлодіодів (рис. 81).

10. Натиснути Tools|Microcontroller Viev PIC16F84. Відкриється вікно з виводами мікроконтролера (рис. 82).

11. У вікні «Select Pin» (рис. 84) почергово натиснути поле «PORTB» і далі «0», після чого натиснути на поле «Select» внизу вікна. Таким чином, вибрано порт B та його вивід 0. Це повторити для всіх ліній вибраного порту.

12. Вибрати Rate|Normal.

13. Натиснути Simulation | Start (почнеться виконання програми). Якщо при цьому курсором клацнути на одному з виводів мікросхеми (панель «Microcontroller Viev PIC16F84») за номером *n* (це відповідає появі на цьому виводі логічної «1» – світлодіод світиться), то засвітиться світлодіод на виводі мікросхеми з номером *n+4* (рис. 85).

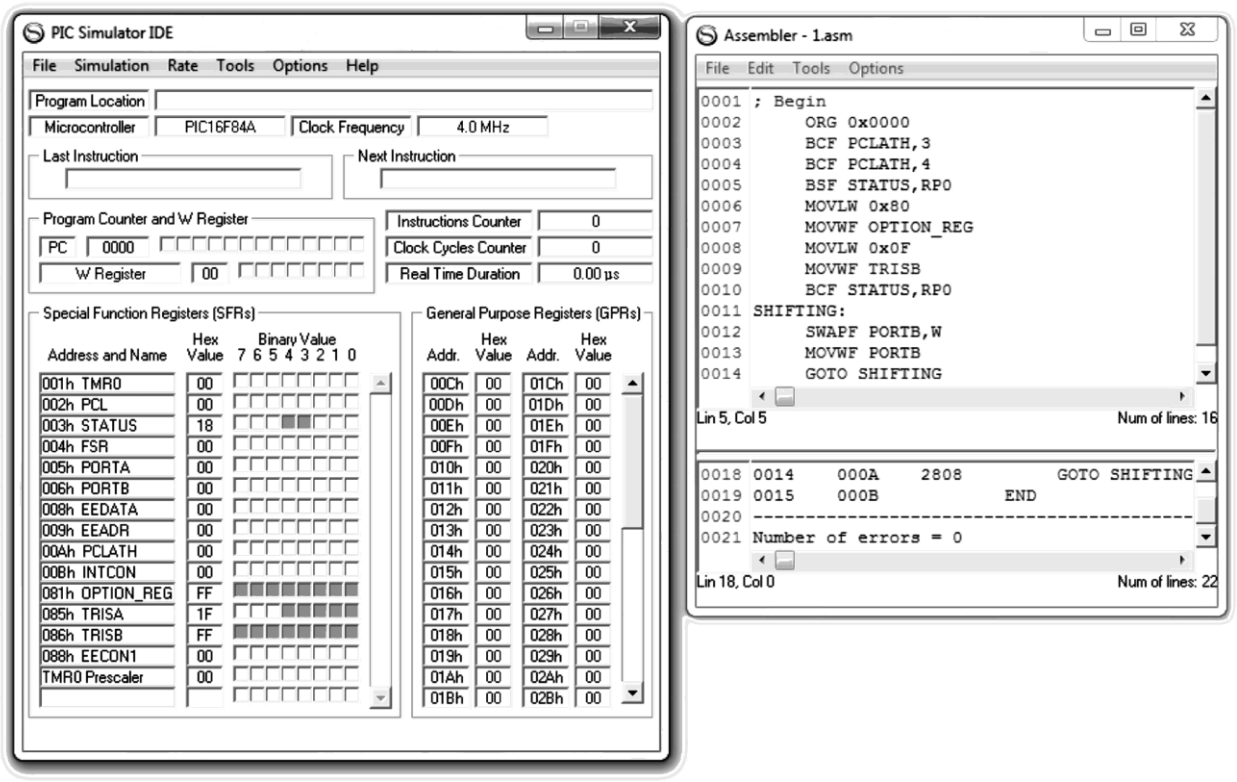


Рисунок 80. Вигляд інтерфейса симулятора з програмою «Input\output» і лістингом скомпільованого файлу

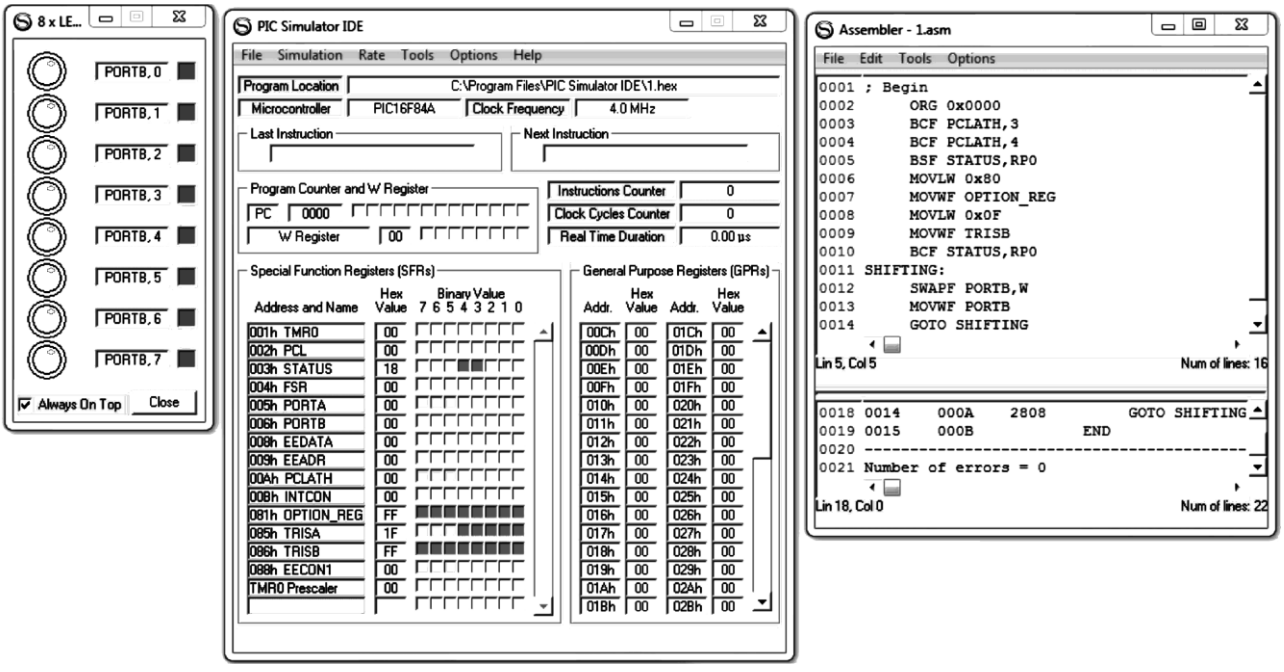


Рисунок 81. Вигляд інтерфейса симулятора з програмою та панеллю «8 x LED Board»

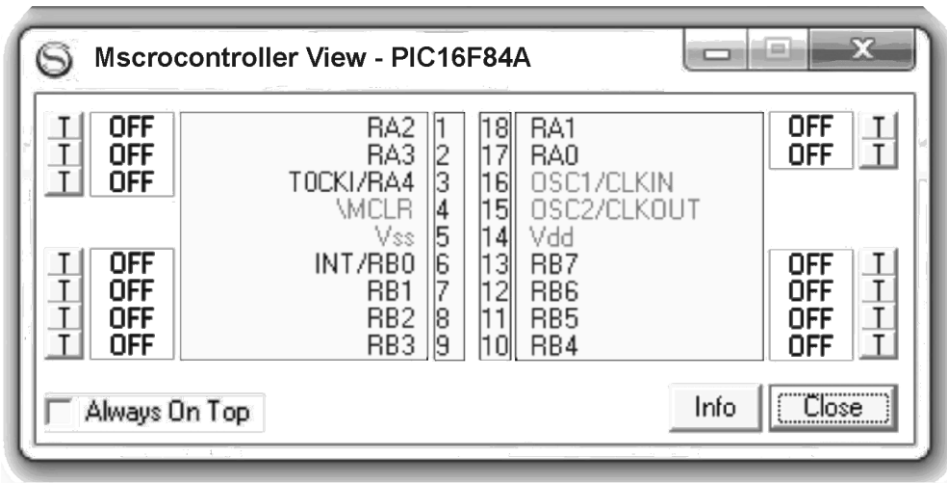


Рисунок 82. Панель виводів мікроконтролера «Microcontroller Viev PIC16F84»

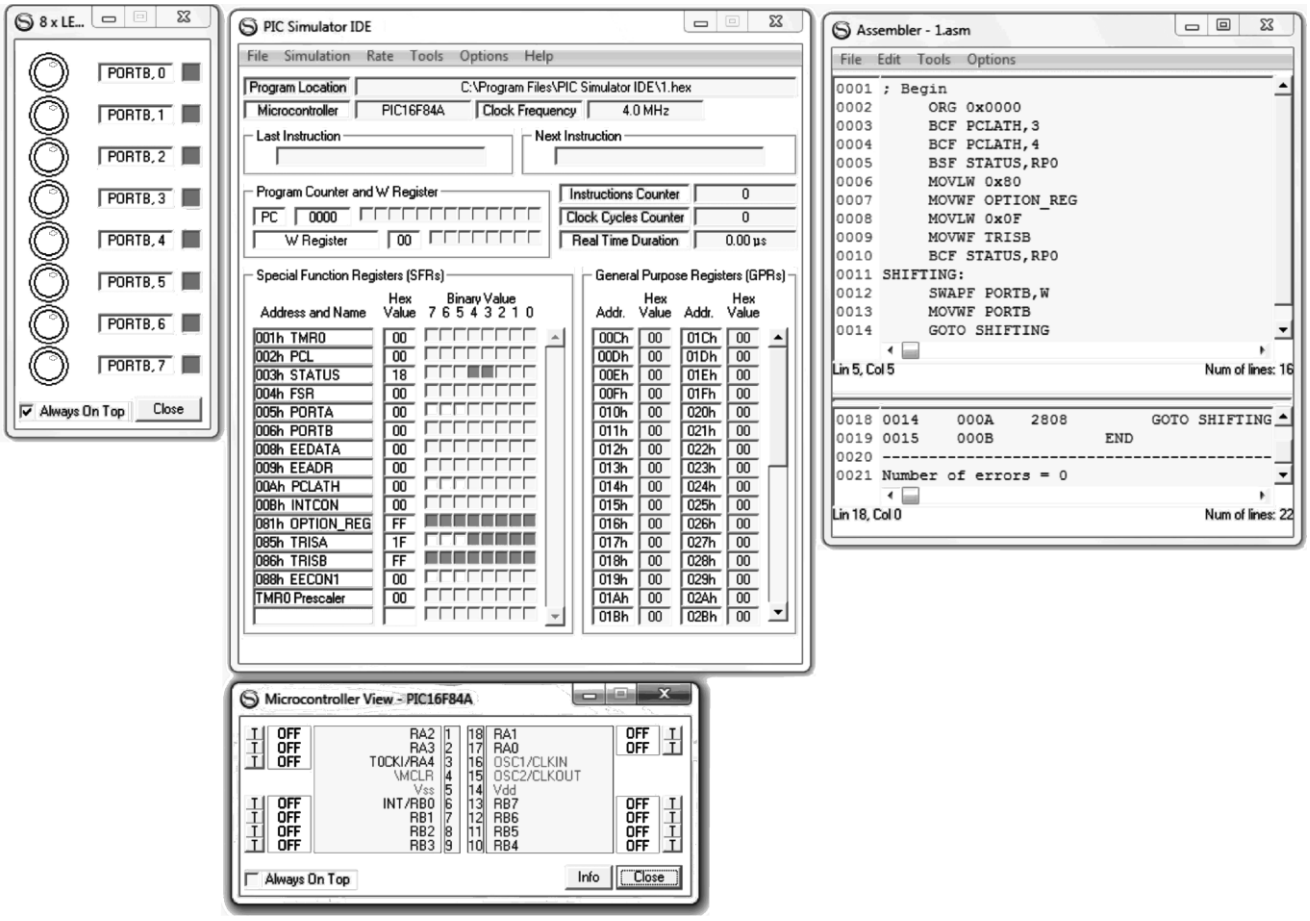


Рисунок 83. Вигляд інтерфейса симулятора з програмою та виводами мікроконтролера – панеллю «Microcontroller Viev PIC16F84»

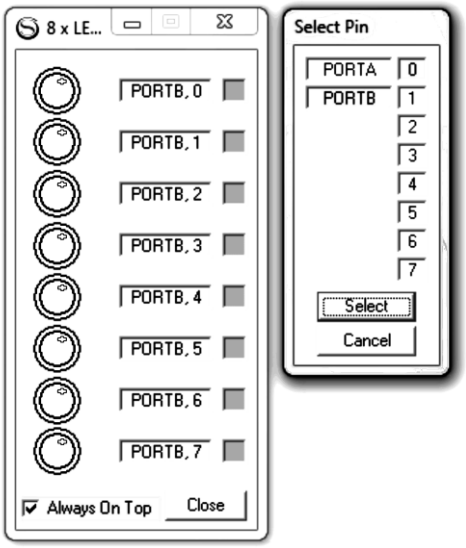


Рисунок 84. Налаштування виводів порту B

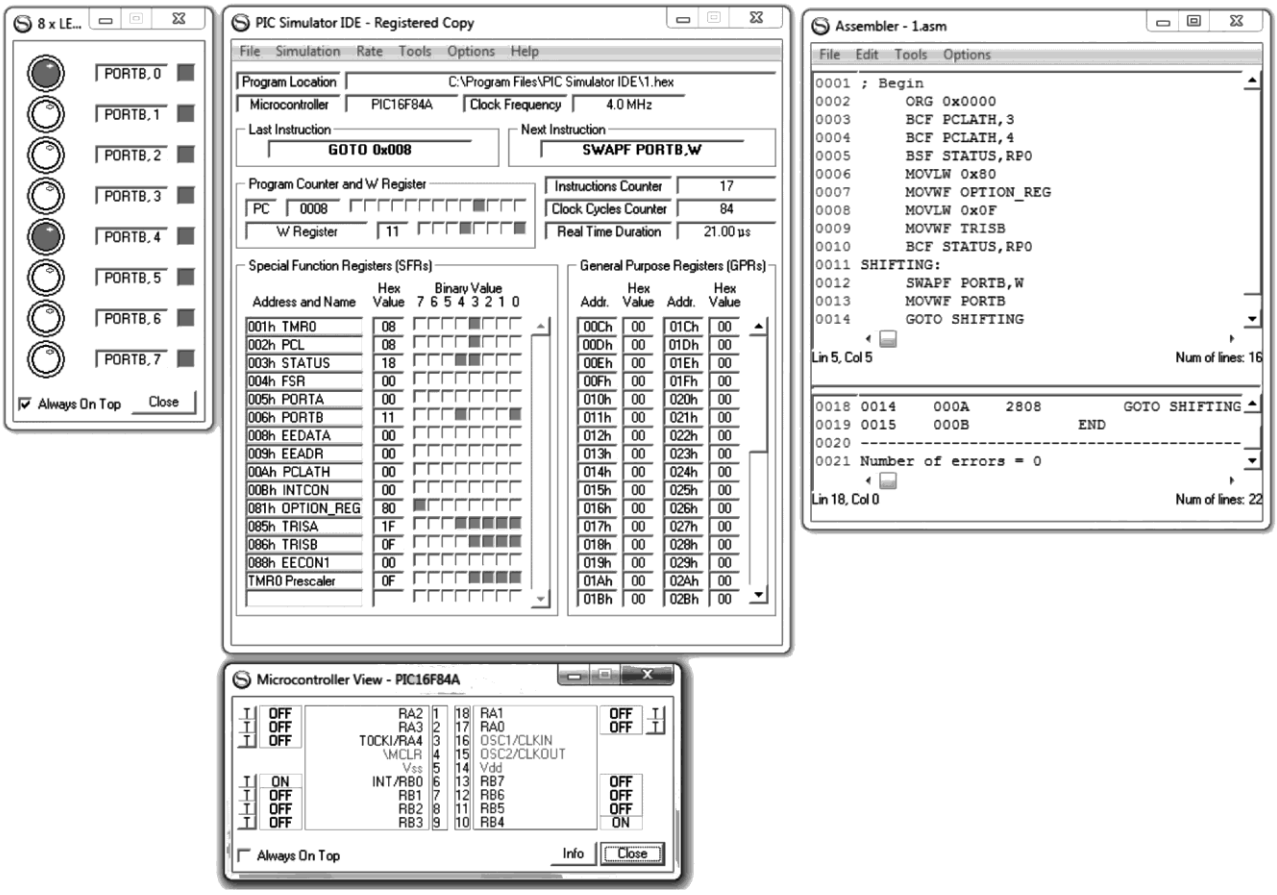


Рисунок 85. Зображення інтерфейса симулятора з виконуваною програмою

14. Щоб зупинити виконання програми, натисніть Simulation | Stop.

Щоб контролювати вміст регістрів після виконання стимулятором кожної команди, потрібно перейти на виконання програми в крокovому режимі роботи:

1. В основному вікні симулятора натиснути Rate | Step By Step, вибрати опцію Simulation і натиснути Start. Симулятор готовий до виконання програми в крокovому режимі.

2. Для виконання наступної команди програми потрібно натиснути на закладку STEP, яка з'явиться справа від закладки HELP вгорі основного вікна симулятора після вибору крокового режиму його роботи.

Вміст регістрів контролера, які використовуються при виконанні команд програми, знайти в області регістрів Adress and Name, яка розташована в лівій нижній частині основного вікна симулятора (виділені рожевим кольором). Усі регістри восьмирозрядні.

В процесі виконання програми по зміні кольору комірок видно, вміст яких регістрів змінюється. Забарвлення комірки відповідного розряду регістру помаранчевим кольором означає наявність «1», білим – «0».

2.19. Проектування пристроїв на основі PIC-контролерів.

2.19.1. Типова схема під'єднання кварцового резонатора та схема початкового скидання PIC-контролера.

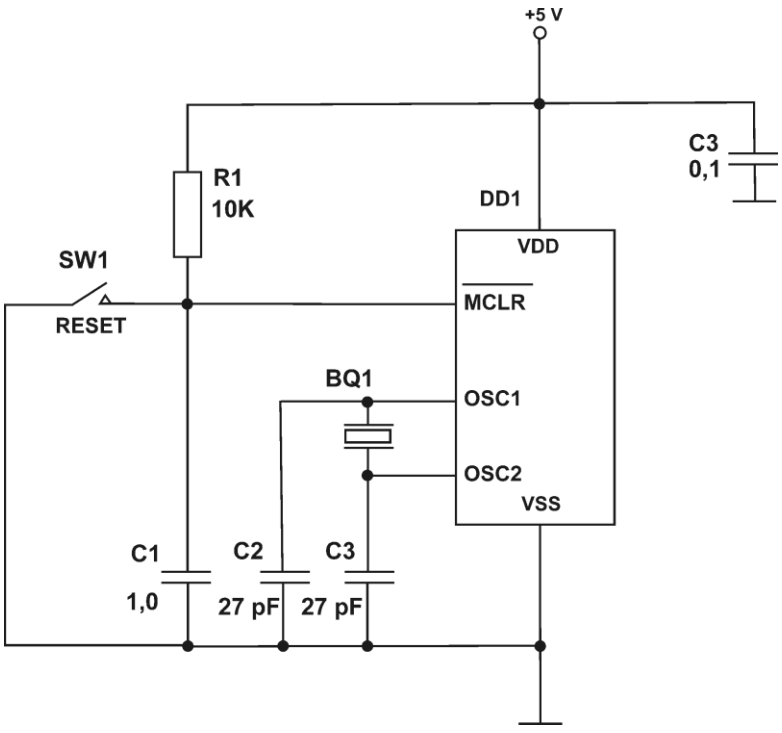


Рисунок 86. Схема скидання контролера

На рис. 86 елементи R1, C3, SW1 реалізують схему скидання контролера. При поданні напруги живлення +5 В напруга на конденсаторі C3 спочатку дорівнює нулю, і при цій напрузі всі регістри контролера обнулюються. З часом напруга на конденсаторі зростає і при досягненні рівня логічної «1» вхід MCLR стає неактивним. Далі контролер переходить у робочий режим.

Перемикач SW1 використовується для примусового скидання контролера уже в процесі роботи (закорочує вхід MCLR на землю).

Типова схема під'єднання кварцового резонатора складається з кварца BQ1 та конденсаторів C1 та C2.

Конденстатор C4 схеми використовується в якості високочастотного фільтра.

2.19.2. Схема підключення навантаження через транзисторний ключ.

У випадку, коли навантаження (в даному випадку світлодіод, див. рисунок 87) потрібно підключити до джерела живлення, напруга якого є більшою за напругу живлення мікроконтролера, використовують транзисторний ключ (на рис. 87 позначений як VT1).

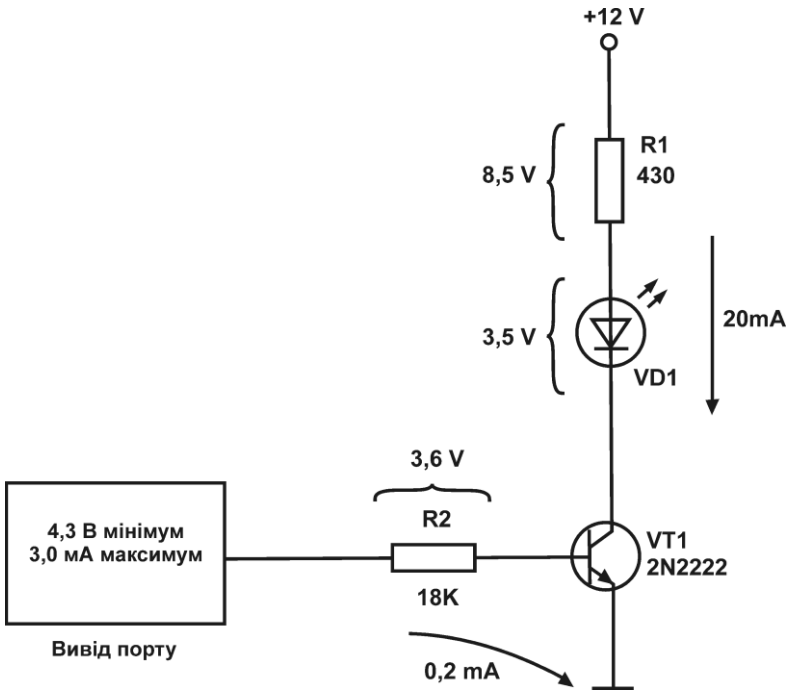


Рисунок 87. Підключення транзисторного ключа до контролера

Навантаженням може бути обмотка реле, соленоїд, лампа розжарення, двигун постійного струму і т.д. Для спрацювання світлодіода (двигуна, реле...) на виході лінії порту потрібно встановити логічну «1».

2.19.3. Схема керування соленоїдом.

На рис. 88 представлена схема керування соленоїдом. Соленоїд може використовуватися для переміщення об’єктів на невеликі відстані.

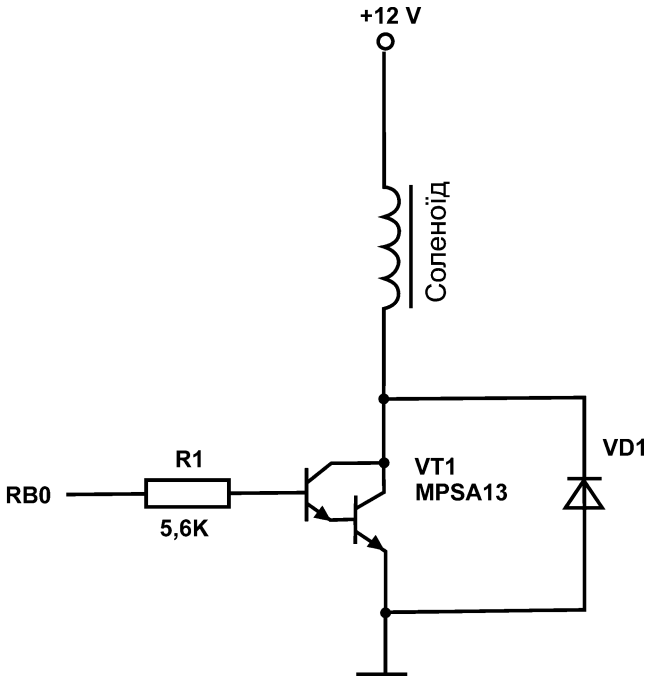


Рисунок 88. Підключення соленоїда до контролера

Обмотка соленоїда підключається до джерела живлення через транзистор VT1, побудований за схемою Дарлінгтона. Для захисту від перенапруги, яка виникає в момент комутації транзистора на обмотці соленоїда, паралельно до транзистора ввімкнено діод VD1. Керування роботою соленоїда здійснюється лінією порту мікроконтролера RB0.

2.19.4. Розширення кількості ліній введення PIC-контролера.

На рис. 89 зображена схема, що реалізує збільшення кількості ліній введення PIC-контролера за рахунок використання послідовного зсувного регістра 74HC7597.

На схемі лінія RB0 використовується як послідовний вхід мікроконтролера, інші дві використовуються для керування роботою регістра: лінія RB1 – вхід тактової частоти послідовного виходу зсувного регістра, RB2 – тактовий вхід з боку завантаження паралельних даних.

За допомогою регістра реалізується 8-розрядний вхідний порт при використанні всього лише трьох ліній порту B мікроконтролера (RB0, RB1, RB2).

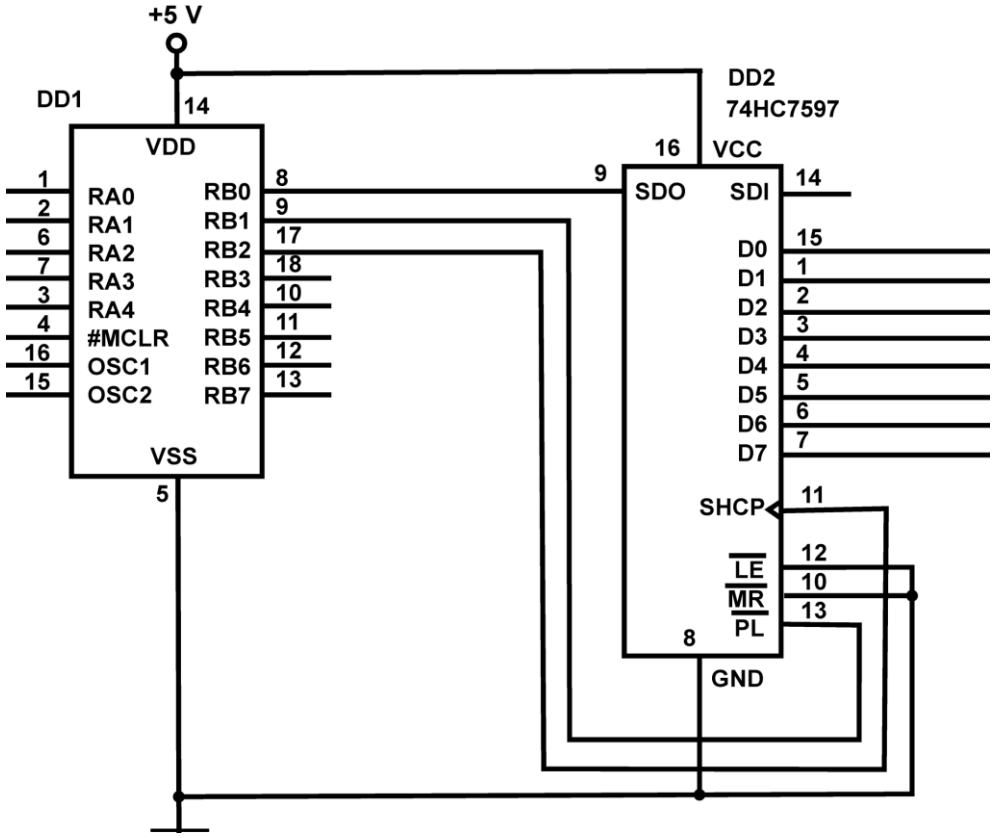


Рисунок 89. Схема розширення ліній вводу/виводу контролера

2.19.5. Підключення датчика температури.

На рис. 90 зображена схема підключення до контролера датчика температури LM70. Датчик містить три лінії керування CS, SI/O та SC, а також дві лінії живлення V+ та GND.

Логічний «0» на лінії CS робить датчик активним.

Вхід SC використовується датчиком як вхід синхронізації частоти.

Вивід SI/O використовується як лінія послідовних даних.

На кожен імпульс на вході SC (за наявності «0» на вході CS) з виводу SI/O датчика мікроконтролером зчитується біт значення температури.

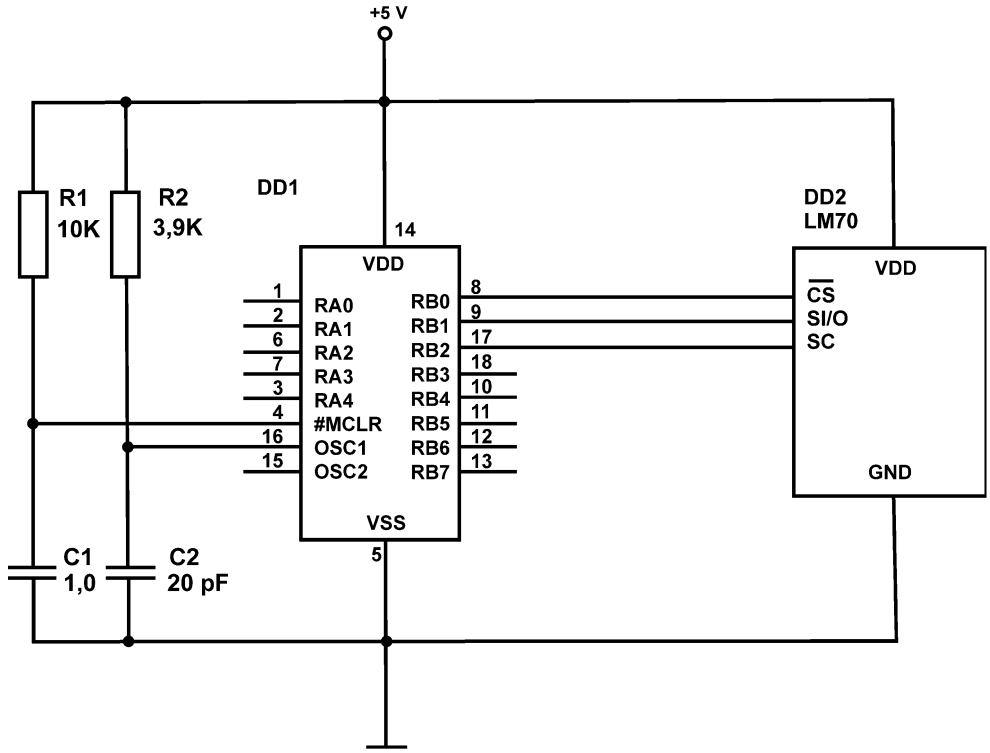


Рисунок 90. Підключення датчика температури до контролера

2.19.6. Схема підключення семисегментного індикатора.

На рис. 91 показано під'єднання світлодіодного семисегментного індикатора зі спільним анодом.

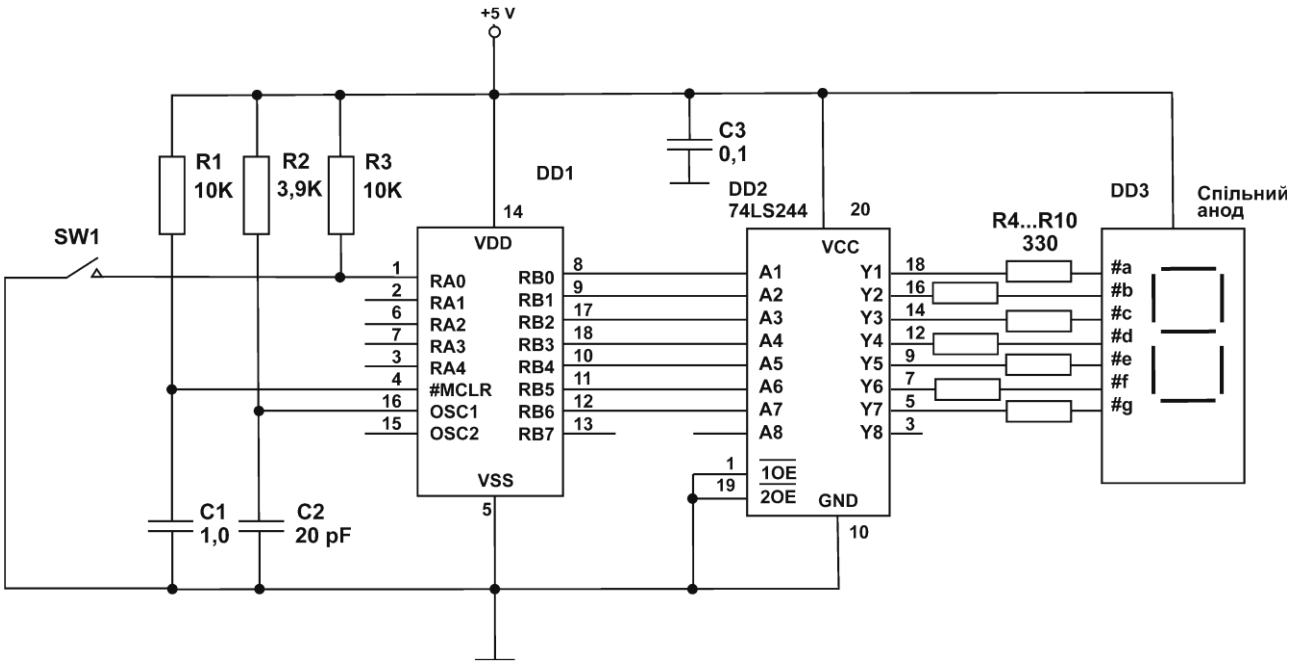


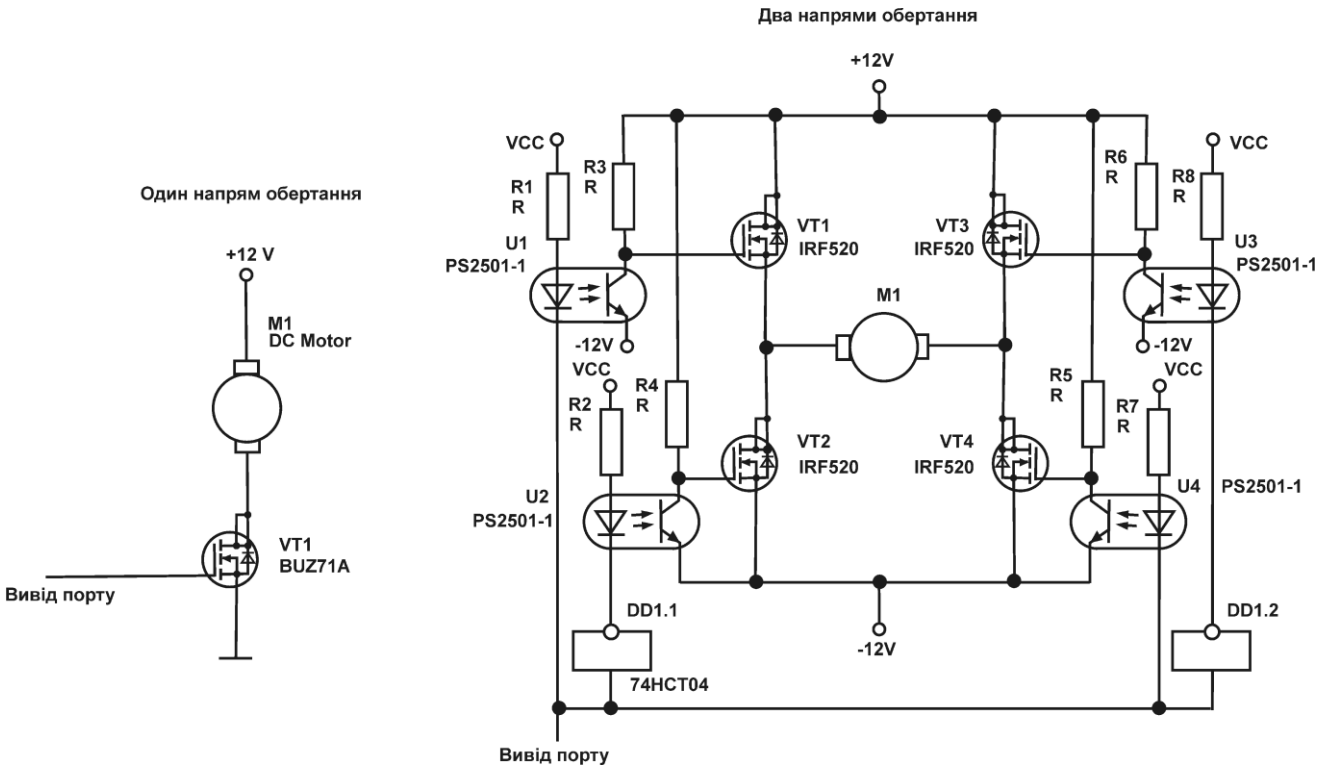
Рисунок 91. Підключення 7-сегментного індикатора до контролера

Індикатор під’єднано через буферний підсилювач струму 74LS244 для забезпечення необхідного струму його сегментів.

Індикація починається після натискання перемикача SW1, з’єднаного з входом RA0 порту A. Сам індикатор під’єднано до порту B.

2.19.7. Керування двигуном постійного струму.

На рис. 92 показані схеми підключення до мікроконтролера двигуна постійного струму.



а) б)

Рисунок 92. Схеми керування двигуном постійного струму

На рис. 92 – а) зображено схему увімкнення/вимкнення двигуна, який під’єднаний до джерела живлення через КМОН транзистор VT1, що працює в режимі ключа. Така схема може забезпечити лише один напрямок обертання двигуна. Затвор транзистора під’єднується до виводу порту мікроконтролера.

На рис. 92 – б) зображено мостову схему увімкнення двигуна. Двигун увімкнено в діагональ мосту, який побудовано на КМОН транзисторах VT1...VT4,

що також працюють у режимі ключа. Транзистори керуються від одного виводу порту мікроконтролера через діодно-транзисторні оптоелектронні пари U1...U4. Для увімкнення плечей мосту в протифазі використовуються інвертори DD1.1, DD1.2. Таке увімкнення двигуна дозволяє змінювати напрям його обертання, почергово переключаючи пари транзисторів VT1,VT4 та VT2,VT3.

2.19.8. Керування кроковим двигуном за допомогою PIC-контролера.

На рис. 93 наведена схема керування кроковим двигуном, чотири обмотки якого через КМОН транзистори VT1...VT4, що функціонують у режимі ключа, під'єднані до ліній порту B мікроконтролера RB0, RB1, RB2 та RB3.

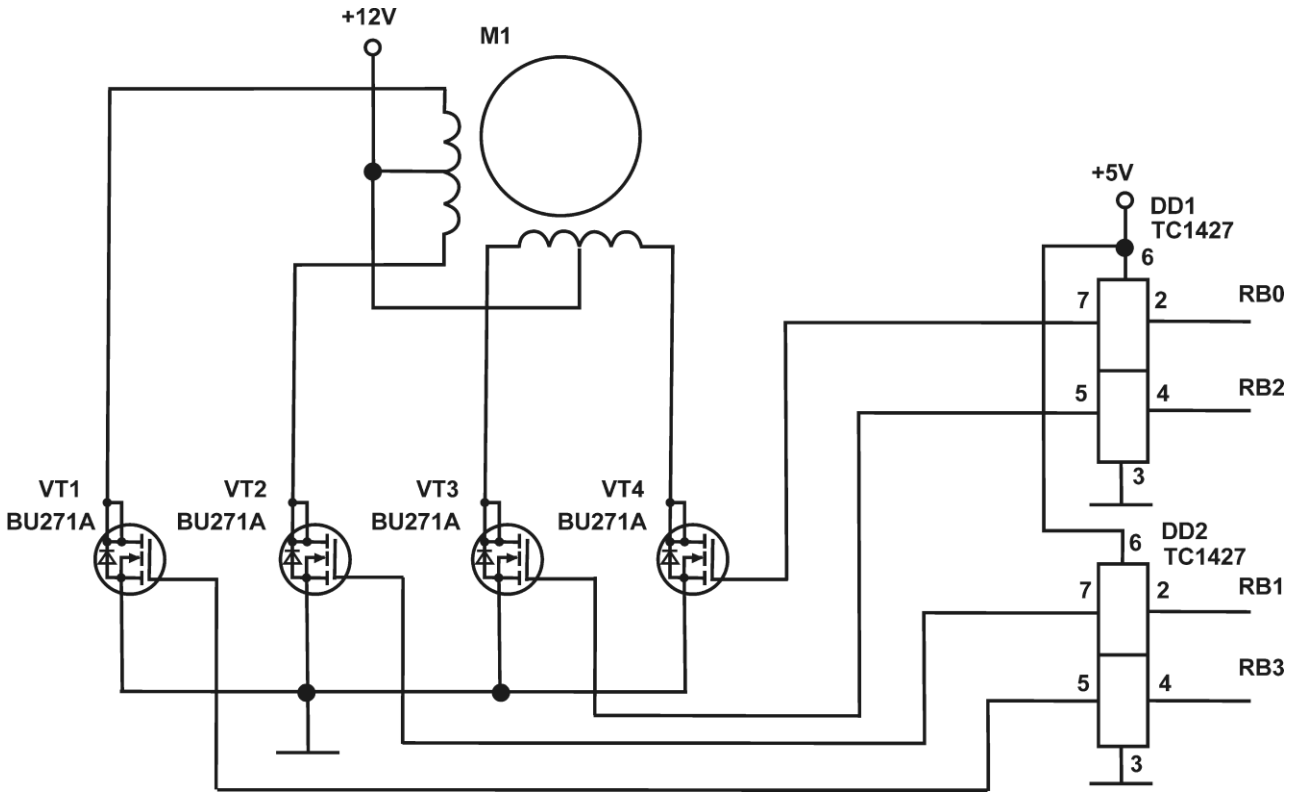


Рисунок 93. Схема керування кроковим двигуном

Для збільшення навантажувальної здатності виводів мікроконтролера в схемі використовують буферні підсилювачі DD1 та DD2.

Для спрощення схеми, щоб уникнути використання буферних підсилювачів, в якості транзисторних ключів можна використати транзистори Дарлінгтона, які мають високий коефіцієнт передавання за струмом.

2.19.9. Використання ЦАП для керування двигуном постійного струму.

На рис. 94 зображена схема, що використовує PIC-контролер DD1 для керування двигуном постійного струму через ЦАП з послідовним входом DD2 (TC1320).

Вхід Vref ЦАП використовується для масштабування вихідної напруги на лінії Vout . На схемі цей вхід з'єднано з напругою +5 В, тому максимальна вихідна напруга ЦАП на лінії Vout також буде +5 В. Лінія SCL – це вхід послідовної синхронізації, SDA – вхід послідовних даних.

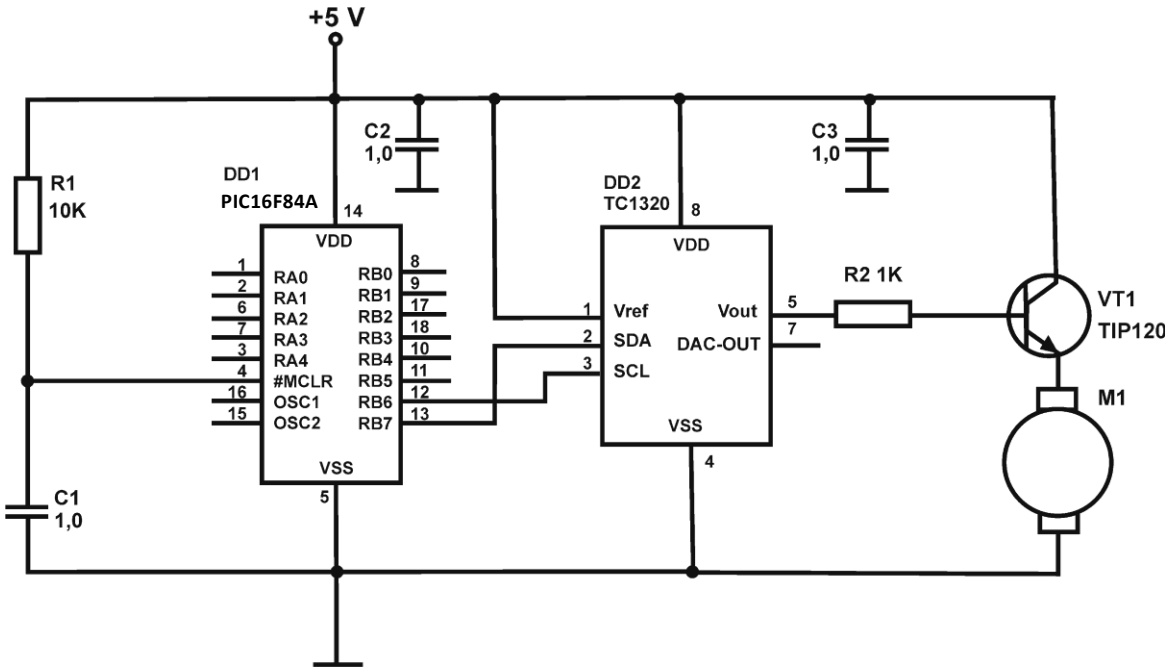


Рисунок 94. Використання ЦАП для керування двигуном постійного струму

Двигун M1 увімкнено в коло емітера транзистора VT1, тобто на двигун подається напруга, величина якої дорівнює напрузі на виході ЦАП Vout.

Вихід DAC-OUT ЦАП використовується в якості виходу за наявності в схемі операційного підсилювача.

2.19.10. Схема простого лічильника на основі PIC-контролера.

На рис. 95 показано схему лічильника з виводом інформації на

чотирирозрядне цифрове табло на основі світлодіодних семисегментних індикаторів зі спільним катодом.

Індикатор під'єднано до виводів порту мікроконтролера через буферний підсилювач струму DS2003 для забезпечення необхідного струму його сегментів.

Імпульсний сигнал подається на вхід RA0 порту А. Сигнальні входи індикаторів з'єднані паралельно й підключені до виводів порту В через підсилювач.

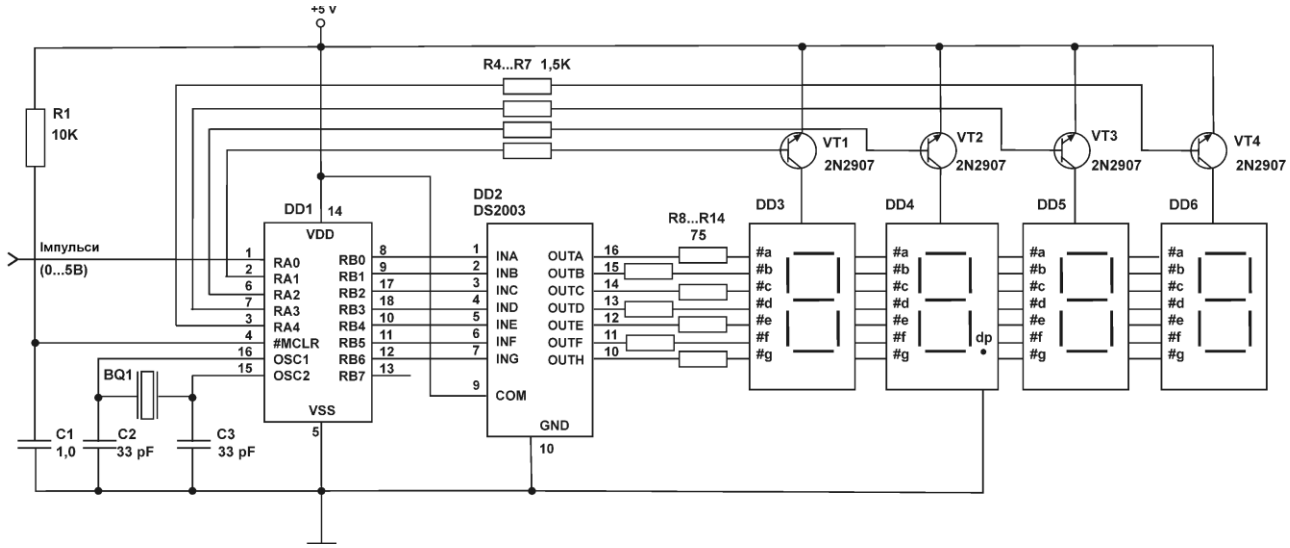


Рисунок 95. Схема лічильника на основі PIC-контролера

Аноди кожного з індикаторів з'єднуються з додатнім полюсом джерела живлення через транзисторні ключі VT1...VT4. У потрібний момент відповідний розряд індикатора під'єднується до джерела живлення через один з транзисторних ключів, які керуються сигналами з ліній порту А.

2.19.11. Частотомір на PIC-контролер.

Призначений для вимірювання частоти логічних сигналів, а також періодичних сигналів непрямокутної форми додатньої полярності. Частотомір (рис. 96) дозволяє вимірювати частоту періодичних сигналів у діапазоні 250 Гц...50 МГц. Похибка вимірювань і відліку для кожного інтервалу частот наведені в таблиці 22. Основний елемент частотоміра – МК PIC16F84. Він

здійснює відлік імпульсів зовнішнього сигналу, що надходить на вхід приладу, опрацювання отриманих значень і виведення результатів вимірювання на індикаційне табло.

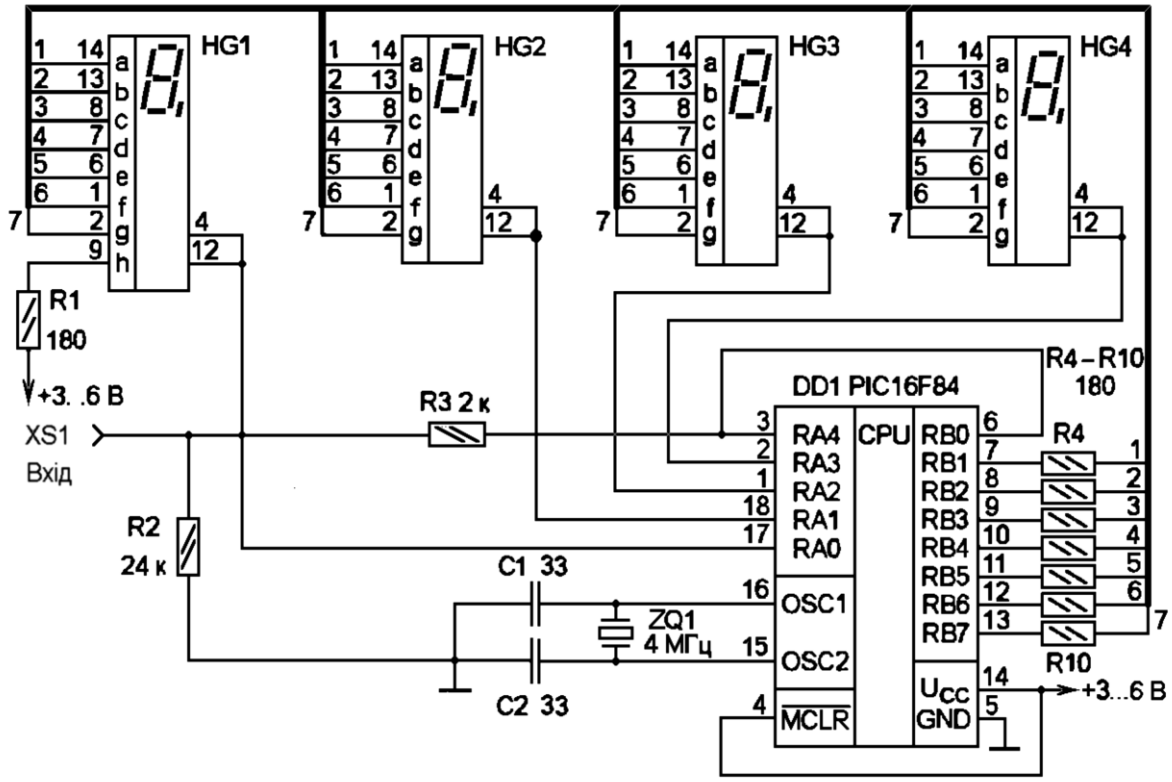


Рисунок 96. Частотомір на базі PIC-контролера

Таблиця 22. Параметри приладу

Інтервал частот, кГц (МГц)	Час вимірювань, мс	Похибка, Гц	
		Вимірювань	Відліку
0,25...0,999 кГц	500	±2	±2
1...9.99 кГц	500	±2	±5
10...69,9 кГц	500	±2	±50
100...127 кГц	500	±2	±500
128...999 кГц	1	±1000	±1000
1...9,99 МГц	1	±1000	±5000
10...50 МГц	1	±1000	±5000

Частота (у герцах) відображається індикаторами HG1-HG4 у форматі X.YZ F Гц, де X.YZ – десяткове значення частоти сигналу, а F – порядок числа

(наприклад, значення «2,25 3» відповідає частоті 2,25·103 = 2250 Гц; «4,32 5» – 4,32·105 = 432 000 Гц = 432 кГц і т.д.).

Мікроконтролер PIC16F84 має у своєму складі восьмирозрядний модуль таймера (TMR0), що використовується з восьмирозрядним попереднім подільником. Останній функціонує асинхронно, тому таймер може рахувати частоту сигналів значно вищу за частоту генератора мікроконтролера, що у даному випадку дорівнює 4 МГц. Мінімальний час високого і низького рівнів вхідного сигналу – 10 нс. Це дозволяє модулеві TMR0 функціонувати від зовнішнього сигналу частотою до 50 МГц.

Вимірюваний сигнал через резистор R2 надходить на вивід RA4 DD1, що є входом зовнішнього сигналу (TOCKI) таймера TMR0. Цей вивід з'єднаний з RB0, перемиканням якого здійснюється керування режимом відліку. Перед вимірюванням відбувається скидання TMR0 (при цьому скидається і попередній подільник). Для вимірювання частоти вивід RB0 конфігурується як вхід на точні інтервали часу, що дозволяє зовнішньому сигналові надходити на вхід таймера. Відлік тривалості інтервалів здійснюється програмою й виконується як точна часова затримка. Після закінчення її вивід RB0 конфігурується як вихід, TMR0 припиняє роботу, оскільки на RA4 встановлюється низький рівень, і зовнішній сигнал перестає надходити на його вхід. Потім зчитується накопичене 16-розрядне значення кількості періодів вхідного сигналу: у старші вісім розрядів записується вміст TMR0, а в молодші – попереднього подільника. Для отримання значення попереднього подільника виконується додаткова підпрограма (з цією метою на виводі RA4 командами BSF і BCF перемикається вихідний рівень, тобто програмно формується послідовність коротких імпульсів). Кожен імпульс інкрементує попередній подільник і лічильник імпульсів N, після чого перевіряється вміст TMR0, щоб визначити, чи збільшився він. Якщо він зріс на 1, восьмирозрядне значення попереднього подільника визначається за вмістом лічильника імпульсів N як 256-N. Далі 16-розрядне двійкове значення частоти перетвориться в шестирозрядне десяткове, котре заокруглюється до тризначного, а потім формується зазначений

вище експоненціальний формат для виведення на табло в динамічному режимі. Сканування індикаторів відбувається з частотою приблизно 80 Гц. Висока навантажувальна здатність мікроконтролера дозволяє під'єднати індикатори безпосередньо до його виводів.

Вимірювання частоти відбувається в два етапи. Спочатку формується інтервал часу (програмна затримка) тривалістю 1 мс, що відповідає області високих частот. Якщо отримане значення частоти більше за 127 (старший байт – значення TMR0, старший розряд молодшого байта – значення попереднього подільника – не дорівнюють 0), воно перетвориться, і результат виведеться на індикатори. Після цього цикл повторюється. Якщо ж значення частоти менше за 127, виконується друге вимірювання (для низьких частот), при якому формується інтервал часу тривалістю 0,5 с. Для оптимізації роботи мікроконтролера він об'єднаний з циклом виведення результату попереднього вимірювання на індикатори. Якщо отримане в результаті вимірювання значення частоти більше за 127, воно перетвориться для індикації, при меншому – проводить третій етап вимірювань. Під його час програмно визначається тривалість періоду імпульсу за допомогою визначення часу, необхідного для надходження 10 імпульсів на вхід частотоміра. Отримане значення перетвориться для індикації за відповідною програмою.

Якщо за 0,2 с 10 імпульсів на вхід не надійшли, вважається, що імпульси взагалі відсутні й покази індикаторів обнулюються. Після цього повний цикл вимірювань повторюється.

2.20. Контрольні запитання.

1. Основні технічні характеристики мікроконтролерів PIC16X8X.
2. Система команд PIC 16x8x.
3. Переваги проектування мікропроцесорних систем на базі PIC 16x8x.
4. PIC-контролер 16C84. Основні характеристики.
5. Архітектура PIC-контролера PIC16F84. Регістри контролера, їх призначення. Регістр статусу. RTCC таймер/лічильник.

6. Організація резидентної пам’яті даних та пам’яті програм. PC і адресація ПЗП.

7. Система переривань. Зовнішнє переривання.

8. Переривання від RTCC. Переривання від порту RB. Переривання від EEPROM.

9. Стек і повернення з підпрограм.

10. Порти вводу-виводу.

11. Регістри портів.

12. Схема ліній порту A.

13. Схема ліній порту B.

14. Байт-орієнтовані команди асемблера PIC16F84.

15. Операції з бітами асемблера PIC16F84.

16. Команди переходів асемблера PIC16F84.

17. Команди викликів та повернення з підпрограм асемблера PIC16F84.

ТЕМА 3. МІКРОКОНТРОЛЕРИ ATMEGA

3.1. Характеристики ядра МК AVR.

З основними характеристиками ядра мікроконтролерів (МК) ATMega32 ознайомимося на прикладі ATMega32. Як і всі контролери цього сімейства воан володіє наступними характеристиками:

◊ статична архітектура з нульовою мінімальною тактовою частотою;

◊ 32 робочі регістри загального призначення;

◊ 131 виконувана команда, більшість команд виконується за 1 такт завдяки гарвардській RISC-архітектурі, при цьому досягається продуктивність 1 MIPS на МГц, а максимальна тактова частота може досягати 16 МГц;

◊ багаторівнева система переривань, підтримка вкладених переривань;

◊ наявність стеку, розміщеного в ОЗП, та можливість операцій з ним;

◊ підтримка інтерфейсу JTAG (стандарт IEEE 1149,1) для програмування та відлагодження.

Пам’ять ATMega32 містить 32 Кб самопрограмованої флеш-пам’яті програм з 10^4 циклами перепрограмування та з можливістю виділити блок самопрограмування. Для довготривалого зберігання даних використовуються 1024 байти EEPROM, що підтримують 10^5 циклів перепрограмування з часом зберігання даних протягом 100 років при 25°C.

Для короткочасного зберігання даних у ATMega32 використовують 2 Кб внутрішньої пам’яті даних.

Мікроконтролер має також кілька таймерів-лічильників, а саме:

◊ два 8-бітні таймери-лічильники з роздільними попередніми подільниками й режимами порівняння, один з лічильників може працювати як лічильник реального часу з окремим генератором;

◊ один 16-бітний таймер-лічильник з окремим попереднім подільником, режимами порівняння та фіксацією даних;

◊ програмований сторожовий таймер з окремим генератором.

Для введення та виведення інформації у паралельному коді використовуються:

◊ 32 лінії вводу-виводу з програмним конфігуруванням та з програмуванням на вхід або на вихід кожного окремо. Усі лінії мають вхідний буфер з тригером Шмідта. Навантажувальна здатність виходів – 20 мА;

◊ чотири канали широтно-імпульної модуляції;

◊ 8-канальний 10-бітний АЦП з можливістю роботи у якості семиканального диференціального АЦП, два диференціальні канали якого мають програмовані коефіцієнти підсилення x1, x10, та x200;

◊ аналоговий компаратор.

Для обміну даних у послідовному коді використовуються:

◊ двопровідний послідовний інтерфейс (аналог I^2C);

◊ стандартний програмований послідовний інтерфейс (USART);

◊ інтерфейс SPI з підримкою веденої та ведучої мікросхем;

◊ інтерфейс відлагодження обладнання JTAG.

Мікроконтролер має також систему зовнішніх та внутрішніх джерел переривання, а також шість режимів зниження енергоспоживання.

3.2. Позначення та призначення виводів МК ATMega32.

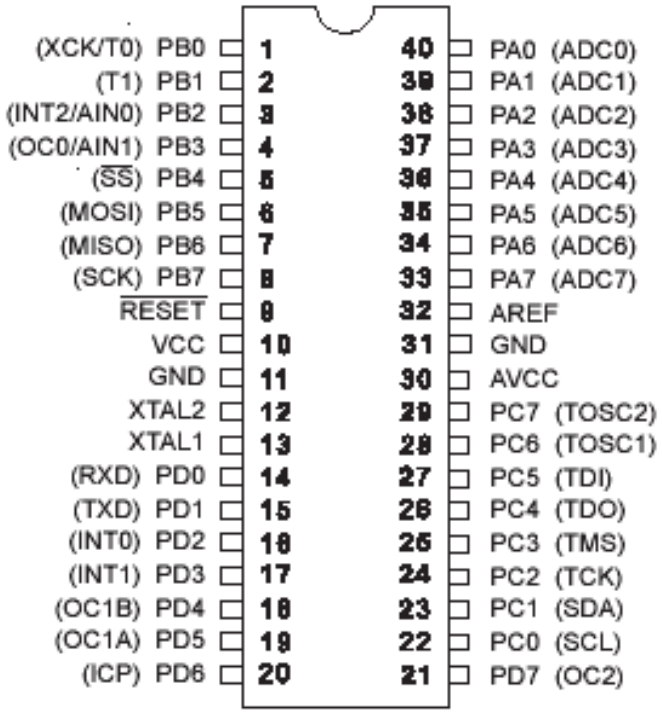


Рисунок 97. Позначення виводів мікросхеми ATMega32

Позначення виводів мікросхеми наведено на рис. 97, а їх призначення – у таблиці 23.

Таблиця 23. Призначення виводів мікросхеми ATMega32

Позначення	Номер виводу	Тип виводу	Призначення
1	2	3	4
XTAL1	13	Вхід	Вхід інвертора генератора і вхід зовнішнього тактового сигналу.
XTAL2	12	Вихід	Вихід інвертора генератора.
RESET	9	Вхід	Вхід скидання. При утриманні низького рівня протягом 50 нс відбувається скидання МК.
PA0-PA7	40–33	Вхід-вихід	Порт А – 8-розрядний двонаправлений порт вводу-виводу з можлівстю аналогового уведення даних. До ліній порту можуть під’єднуватися внутрішні навантажувальні резистори (для кожного біта окремо). Вихідні буфери порту симетричні й можуть як віддавати, так і приймати струм. Після скидання лінії порту переводяться у від’єднаний стан. Порт А має можливість введення аналогових даних.
PB0-PB7	1–8	Вхід-вихід	Порт В – 8-розрядний двонаправлений порт вводу-виводу з внутрішніми підтягуючими резисторами, які вмикаються окремо для кожного біта. Вихідні буфери порту симетричні і можуть як віддавати, так і приймати струм. Після скидання лінії порту переводяться у від’єднаний стан. Лінії порту В мають альтернативне призначення, яке вмикається при активації відповідного блоку мікросхеми.
PB7/SCK	8	Вхід	Сигнал синхронізації шини SPI.
PB6/MISO	7	Вхід-вихід	Сигнал передавання даних для веденої мікросхеми та приймання для ведучої по шині SPI.

Продовження таблиці 23

1	2	3	4
PB5/MOSI	6	Вхід-вихід	Сигнал передавання даних для ведучої мікросхеми та приймання для веденої по шині SPI.
PB4/SS	5	Вхід	Сигнал вибору веденої мікросхеми по шині SPI.
PB3/AIN1/ OC0	4	Вхід/вихід	Негативний вхід аналогового компаратора. Вихід пристрою порівняння таймера-лічильника 0.
PB2/AIN0/ INT2	3	Вхід	Позитивний вхід аналогового компаратора. Вхід другого зовнішнього переривання.
PB1/T1	2	Вхід	Зовнішнй вхід тактового сигналу для таймера-лічильника 1.
PB0/T0/ XCK	1	Вхід	Зовнішній вхід тактового сигналу для таймера-лічильника 0. Сигнал зовнішньої синхронізації послідовного порту (USART)
PC0-PC7	22-29	Вхід-вихід	Порт С – 8-розрядний двонаправлений порт вводу/виводу з внутрішніми навантаженими резисторами, які вмикаються окремо для кожного біта. Вихідні буфери порту симетричні і можуть як віддавати, так і приймати струм. Після скидання лінії порту переводяться у від’єднаний стан. Лінії порту С мають альтернативне призначення, яке вмикається при активації відповідного блоку мікросхеми. Якщо інтерфейс JTAG активований, лінії інтерфейсу активізуються назалежно від сигналу скидання.
PC7 TOSC2	29	Вхід	Вхід пі’єднання резонатора для таймера-лічильника 2.
PC6 TOSC1	28	Вхід	Вхід під’єднання резонатора для таймера-лічильника 2.
PC5 TDI	27	Вхід	Вхід даних для інтерфейсу JTAG.
PC4 TDO	26	Вихід	Вихід даних для інтерфейсу JTAG.
PC3 TMS	25	Вхід	Сигнал вибору тестового режиму інтерфейсу JTAG.
PC2 TCK	24	Вхід	Тактовий сигнал інтерфейсу JTAG.

Закінчення таблиці 23

1	2	3	4
PC1 SDA	23	Вхід-вихід	Лінія даних інтерфейсу I2C.
PC0 SCL	22	Вхід-вихід	Лінія тактового сигналу інтерфейсу I2C.
PC0-PC7	14–21	Вхід-вихід	Порт D – 8-розрядний двонаправлений порт вводу-виводу з внутрішніми навантаженими резисторами, які вмикаються окремо для кожного біта. Вихідні буфери порту симетричні і можуть як віддавати, так і приймати струм. Після скидання лінії порту переводяться у від'єднаний стан. Лінії порту D мають альтернативне призначення, яке вмикається при активації відповідного блока мікросхеми.
PD7 OC2	21	Вихід	Вихід порівняння таймера-лічильника 2.
PD6 ICP1	20	Вхід	Вхід фіксації відліку таймера-лічильника 1.
PD5 OC1A	19	Вихід	Вихід порівняння А таймера-лічильника 1.
PD4 OC1B	18	Вихід	Вихід порівняння В таймера-лічильника 1.
PD3 INT1	17	Вхід	Вхід зовнішнього переривання 1.
PD2 INT0	16	Вхід	Вхід зовнішнього переривання 0.
PD1 TXD	15	Вихід	Вихід послідоного порту (USART).
PD0 RXD	14	Вхід	Вхід послідоного порту (USART).
ARef	32	Вхід-вихід	Вхід зовнішнього та вихід внутрішнього опорного сигналу для вбудованого АЦП.
GND	11, 31	Живлення	Загальний.
Vcc	20	Живлення	Вивід джерела живлення.
AVcc	20	Живлення	Вивід джерела живлення аналогових ланцюгів та порту А.

3.3. Архітектура ATMega32.

Архітектура ATMega32 (рис. 98) є типовою для всього сімейства AVR і виконана за удосконаленою RISC (enhanced RISC) архітектурою. Контролер побудований за Гарвардською архітектурою, що дозволяє здійснювати конвеєризацію: під час виконання поточної команди здійснюється вибірка з пам'яті й дешифрування коду наступної команди.

Центральний процесор мікроконтролера складається з лічильника команд, регістра і дешифратора команд, 32-х регістрів загального призначення (РЗП) та арифметико-логічного пристрою (АЛП).

Арифметично-логічний пристрій (АЛП), який виконує всі обчислення, під'єднаний до 32 робочих регістрів, об'єднаних у регістровий файл. Регістри із регістрового файлу майже рівноправні й можуть виконувати функції акуму-лятора.

Зауважимо, що більшість операцій АЛП виконує за один такт. Регістровий файл та АЛП під'єднані до внутрішньої шини даних.

14-розрядний лічильник команд (PC – Program Counter) використовується для відліку номера комірки пам'яті програм, що містить код команди, яка виконується. Адреса з лічильника через виділену шину надходить на постійну пам'ять ПЗП (Flash). Напрям з програми регістр PC недоступний, при нормальному виконанні програми PC автоматично збільшується на 1 або на 2 залежно від команди, що виконується. Цей порядок порушується при виконанні команд переходу, виклику й повернення з підпрограм, а також при виникненні переривань. Після увімкнення живлення, а також після скидання МК, в PC автоматично завантажується значення 0x000.

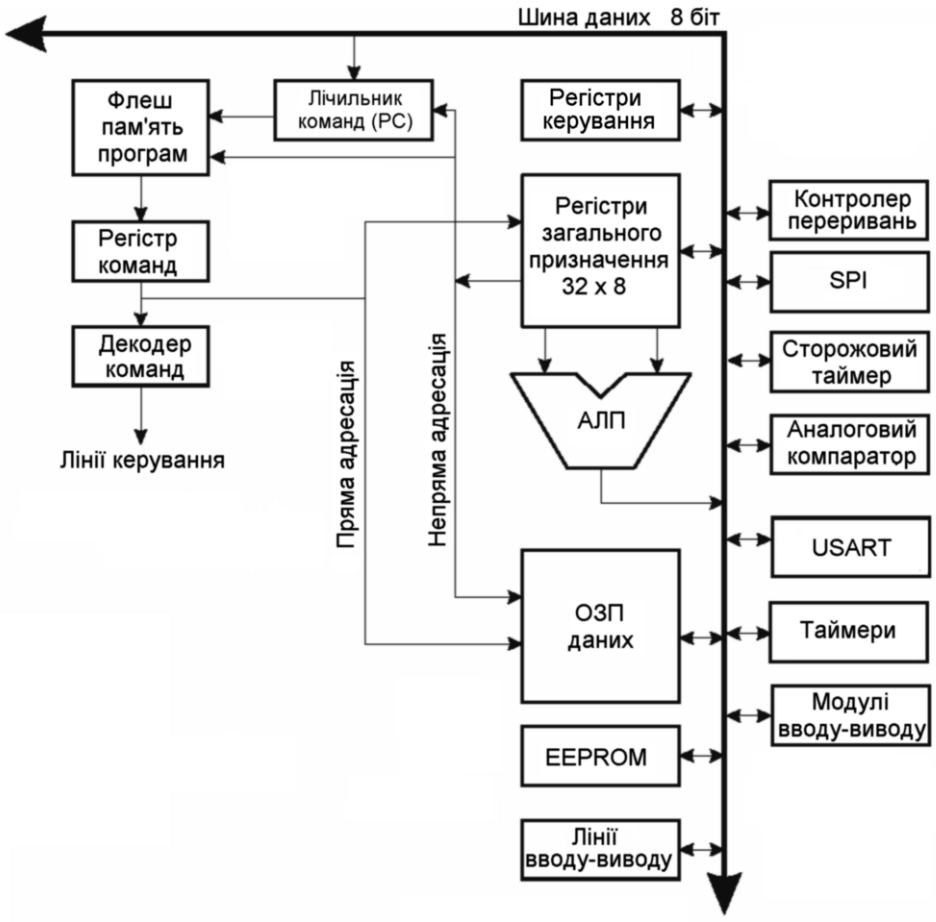


Рисунок 98. Архітектура ATMega32

В *ПЗП* зберігається код команд, які виконуються мікроконтролером. Флеш-пам’ять організована як 16х16 і має об’єм 32 Кб.

В МК AVR майже всі команди (за винятком команд, у яких одним із операндів є 16-розрядна адреса) займають одну комірку пам’яті програм.

Програмування пам’яті здійснюється за допомогою SPI-інтерфейсу безпосередньо на цільовій платі, а також із використанням паралельного програматора. Кількість циклів перезаписування флеш-пам’яті не менша 10^4. Мікросхема має можливість самопрограмування, тобто мікроконтролер здатний самостійно, без зовнішнього програматора, змінювати вміст комірок пам’яті за програмою, записаною в boot-блоку пам’яті. Це дає можливість змінювати робочу програму з використанням будь-якого інтерфейсу.

Регістр команд та дешифратор команд виконують функції зберігання та дешифрування коду команди, що виконується. Код команди перетворюється в сигнали керування іншими блоками мікроконтролера.

Оперативна пам’ять (ОЗП) використовується для зберігання даних, які змінюються в процесі роботи мікроконтролера. ОЗП є у всіх AVR сімейства Tiny, Mega та Classic, крім A90S1200 і її аналогів.

До деяких мікроконтролерів можна під’єднати зовнішню пам’ять даних ємністю до 64 Кб, проте ATMega32 не входить до їх числа.

В ATMega32 об’єм ОЗП становить 2048 байтів.

Усі мікроконтролери AVR мають також *блок енергонезалежної пам’яті даних з електричним стиранням* (EEPROM). Цей тип пам’яті використовують для зберігання даних, різних констант, таблиць перекодувань, каліброваних коефіцієнтів тощо. Дані в EEPROM можна завантажити як через SPI-інтерфейс, так і за допомогою звичайного програматора. В усіх мікроконтролерах EEPROM є доступною для зчитування та побайтової модифікації. Кількість циклів перезапису становить не менше 10^5.

До *периферійних пристроїв* мікроконтролера відносять:

◊ регістри керування,

◊ таймери,

◊ аналогові пристрої,

◊ УАПП (UART),

◊ сторожовий таймер,

◊ порти вводу-виводу,

◊ модуль переривань.

Регістри керування призначені для керування роботою мікроконтролера. До них відносять регістр стану SREG, регістр управління MCUCR, вказівник стеку SP тощо. Залежно від типу мікроконтролера склад і кількість регістрів керування може змінюватися.

Кількість *таймерів* у різних мікросхемах сімейств може бути різною. У ATMega32 до групи таймерів відносять три таймери-лічильники.

Восьмирозрядний таймер T0 використовується для відліку і вимірювання часових інтервалів або як лічильник зовнішніх подій. Регістр відліку таймера може бути записаний і зчитаний. При переповненні таймер генерує запит на переривання.

Таймер T1 (16-розрядний) може генерувати запит на переривання не тільки при переповненні регістра відліку, але й при настанні ряду інших подій.

Усі три таймери можуть працювати в режимі широтно-імпульсного модулятора (ШІМ).

Таймер 1 має можливість фіксації стану таймера за зовнішнім сигналом. Джерелом сигналу для трьох таймерів може бути тактова частота мікроконтролера, поділена на певний коефіцієнт, а також входи T0, T1 та TOSC1 відповідно. Джерела сигналу синхронізації встановлюються незалежно один від одного.

Таймер T2 може працювати як із загальним тактовим сигналом мікросхеми, так і асинхронно відносно інших блоків, використовуючи власний тактовий генератор.

До складу *аналогових пристроїв* в ATMega32 входить:

- аналоговий компаратор;
- 8-канальний 10-розрядний АЦП.

Аналоговий компаратор може порівнювати значення сигналу з позитивного входу AIN0 з сигналом на негативному вході AIN1. Вихід аналогового компаратора може бути налаштований на роботу на вхід захоплення таймера T1, або формувати власне переривання.

ATMega32 у своєму складі має *прийомопередавач (UART)*. Швидкість передавання даних може змінюватися в широких межах. Модуль UART може виявляти й сигналізувати про різні збої при передаванні даних, а саме: переповнення, помилка кадрування, невірний стоп-біт. Для зменшення ймовірності збоїв у модулі реалізована функція фільтрації завад. Для взаємодії з програмою в модулі передбачено три переривання за наступними подіями: «передавання даних завершено», «регістр даних передавача порожній», «прийом завершено».

Виводи МК, які використовуються модулем UART, є лініями порту D. В якості входу приймача (RxD) використовується вивід PD0, а в якості виходу передавача (TxD) – вивід PD1.

Сторожовий таймер WDT (WATCHDOG) призначений для перезапуску програми у випадку появи збою у ході її виконання. Програма, що працює без збоїв, періодично скидає сторожовий таймер, не допускаючи його переповнення. Сторожовий таймер має свій власний RC – генератор, який працює на частоті 1 МГц. На вході WDT увімкнено попередній подільник вхідної частоти з програмованим коефіцієнтом ділення, що дозволяє регулювати часовий інтервал переповнення таймера і скидання мікроконтролера.

Мікросхема має 32-і незалежні *лінії вводу-виводу*, згруповані в 4 порти. Кожна з ліній може запрограмовуватися на ввід або на вивід. Потужні вихідні драйвери забезпечують струмову навантажувальну спроможність 20 мА на лінію, при цьому загальне струмове навантаження на всі лінії одного порту не має перевищувати 80 мА. Зауважимо, що інші мікроконтролери AVR мають кількість ліній вводу-виводу, що коливається від 3 до 53.

Модуль переривань містить схему дозволу/заборони та ранжування запитів за пріоритетом. У ATMega32 переривання поділяються на внутрішні та зовнішні. Джерелами внутрішніх переривань є вбудовані модулі (наприклад,

таймер T0). Зовнішні переривання викликаються скиданням (сигналом на виводі RESET) або сигналами на виводах INT. В МК AVR усім перериванням поставлений у відповідність власний вектор переривання – адреса в початковій області пам’яті програм, за якою розташовується команда переходу до підпрограми опрацювання переривання.

Внутрішній тактовий генератор AVR-МК можна запускати від кількох джерел опорної частоти (зовнішній генератор, зовнішній кварцовий резонатор, внутрішня або зовнішня RC-ланка). Оскільки AVK-МК повністю статичні, мінімальну допустиму частоту нічим не обмежено, тобто можна легко забезпечити навіть покроковий режим виконання програми. Максимальна робоча частота визначається конкретним типом мікроконтролера.

3.4.Функціонування конвеєра, цикл виконання команд мікроконтролера.

У МК процес виконання команд організований так, щоб при виборі команди з пам’яті програм відбувалося виконання попередньої команди, тобто функціонує дворівневий конвеєр. Таким чином, тривалість машинного циклу дорівнює тривалості періоду тактової частоти. Робота цього конвеєра наведена на рис. 99.

Під час першого машинного циклу відбувається вибірка команди з пам’яті програм і її декодування.

Під час другого циклу ця команда виконується, а паралельно відбувається вибірка і декодування другої команди. В результаті фактичний час виконання кожної команди дорівнює одному машинному циклу.

При виконанні певних команд може відбуватися порушення нормальної роботи конвеєра. Типовим прикладом таких команд є команди умовного переходу. Якщо умова, яка перевіряється командою умовного переходу, істинна, то виконання програми буде продовжено з нової адреси. Оскільки в конвеєрі вже відбулася вибірка команди, розташованої після команди переходу, то час виконання команди переходу збільшується на 1 цикл, під час якого відбувається вибірка команди, розташованої за потрібною адресою.

Аналогічно команди безумовного відносного і непрямого переходу, команди виклику підпрограм, команди повернення з підпрограм також змінюють вміст PC. У результаті виконання цих команд відбувається розрив у роботі конвеєра, а внаслідок цього – затримання виконання програми на 2…4 машинні цикли.

За тієї ж причини відбувається порушення роботи конвеєра при виникненні переривання. Мінімальна затримка при цьому становить 4 машинні цикли.

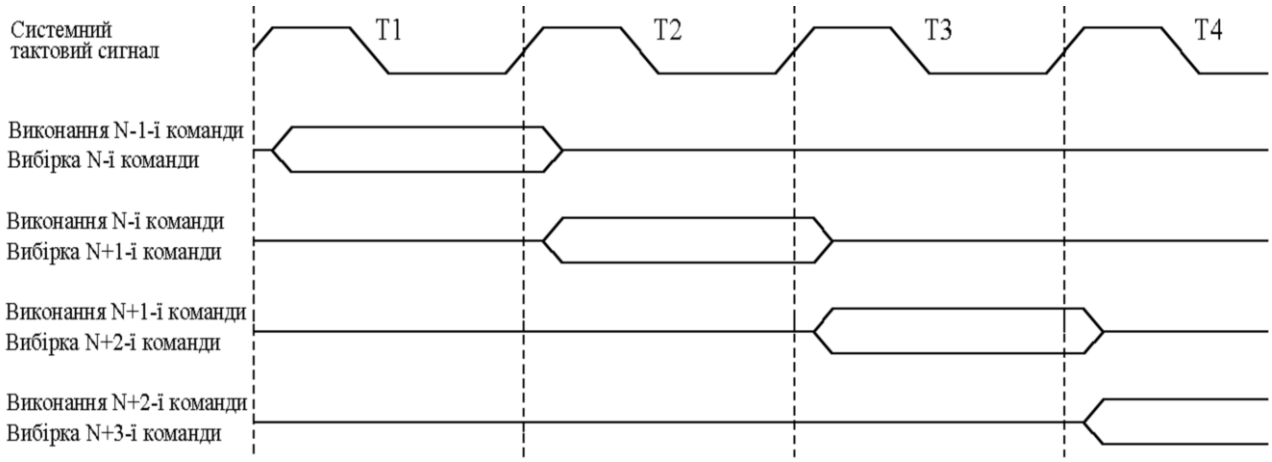


Рисунок 99. Робота конвеєра

3.5. Регістр стану – SREG.

Регістр стану містить інформацію про результати виконання останньої арифметичної або логічної команди. Ця інформація може використовуватися в командах умовного переходу та інших умовних операторах для того, щоб змінити процес виконання програми.

Регістр стану мікроконтролера SREG має такий формат (рис. 100):

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 100. Формат регістра SREG

Біт 7 – I: загальний дозвіл переривань. Внаслідок встановлення цієї ознаки в одиничний стан визначається робота всієї системи переривань. Окремі

види переривань вмикаються і вимикаються за допомогою додаткових регістрів конфігурації.

Якщо ознака «Загальний дозвіл переривань» має нульове значення, всі переривання заблоковані, незалежно від того, увімкнені вони чи ні в додаткових регістрах конфігурації. Ознака I апаратно скидається відразу після виклику відповідної процедури опрацювання переривання і встановлюється при виконанні команди RETI, дозволяючи подальші переривання. Ознака I може бути також встановлена і скинута програмно за допомогою команд SEI і CLI відповідно.

Біт 6 – T: ознака користувача – біт для тимчасового зберігання інформації. Біт T використовується командами BLD (завантаження біту T) і BST (читання біта T) як комірок для тимчасового зберігання інформації. Будь-який біт будь-якого регістра загального призначення може копіюватися в T, а потім вміст T може, в свою чергу, копіюватися в будь-який інший біт того ж або будь-якого іншого регістра.

Біт 5 – H: ознака половинного перенесення. Ця ознака встановлюється в одиницю, якщо відбулося перенесення з молодшої половини байта (з 3-го розряду в 4-й) або зворотне перенесення із старшої половини байта при виконанні деяких арифметичних операцій. Цей біт є аналогічним ознаці AC мікропроцесорів Intel.

Біт 4 – S: ознака знака, $S = N \otimes V$. Ця ознака є результатом операції «Виключне АБО» (XOR) між ознаками N (від'ємний результат) і V (переповнення числа в додатковому коді). Відповідно, ця ознака встановлюється в одиницю, якщо результат виконання арифметичної операції менший нуля.

Біт 3 – V: ознака переповнення додаткового коду. Ця ознака використовується при роботі зі знаковими числами (числами, представленими в додатковому коді). Ознака встановлюється в одиницю, якщо в результаті арифметичної операції відбудеться переповнення числа, представленого в додатковому коді.

Біт 2 – N: ознака від'ємного значення. Ознака встановлюється в одиницю, якщо в результаті арифметичної операції старший розряд результату дорівнює одиниці. Якщо старший розряд результату дорівнює нулю, то ознака N теж дорівнює нулю.

Біт 1 – Z: ознака нуля. Ознака встановлюється в одиницю, якщо результат виконуваної операції дорівнює нулю.

Біт 0 – C: ознака перенесення. Ця ознака відображає переповнення результату (перенесення в старший розряд) при виконанні арифметичної операції. Крім того, ознака перенесення використовується в операціях зсувів.

3.6. Організація пам’яті даних ATMega32.

Пам’ять МК AVR виконана за Гарвардською архітектурою з розділеними адресними просторами пам’яті програм і пам’яті даних.

Пам’ять програм призначена для зберігання команд, що керують роботою МК, а також констант. Пам’ять програм в МК AVR – це Flash-ПЗП з кількістю циклів перезаписування не менше 10^4. Оскільки більшість команд займають у пам’яті 16 бітів, пам’ять програм має 16-розрядну організацію.

Пам’ять даних складається з трьох областей: регістрової пам’яті (регістровий файл), статичного ОЗП і пам’яті EEPROM. Оскільки регістрова пам’ять знаходиться в адресному просторі ОЗП, то ці дві області сприймають як одну. Область EEPROM розташована у своєму власному адресному просторі (рис. 101).

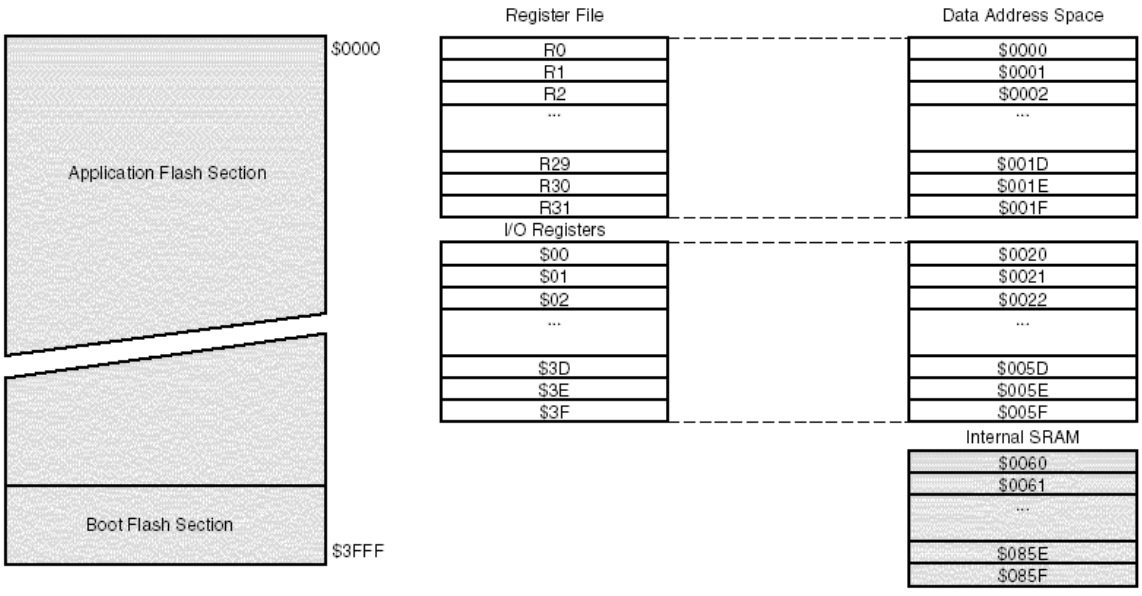


Рисунок 101. Структура пам’яті МК ATMega32

3.6.1. Регістровий файл.

Регістровий файл займає молодші 32 байти у загальному адресному просторі AVR (рис. 102). Шість із 32 регістрів файлу можна використовувати як три 16-розрядні покажчики адреси в процесі непрямого адресування даних. Один із цих покажчиків (Z Pointer) застосовують також для доступу до даних, записаних у пам’яті програм мікроконтролера. Використання трьох 16-розрядних покажчиків (X, Y і Z Pointers) істотно підвищує швидкість пересилання даних під час роботи прикладної програми.

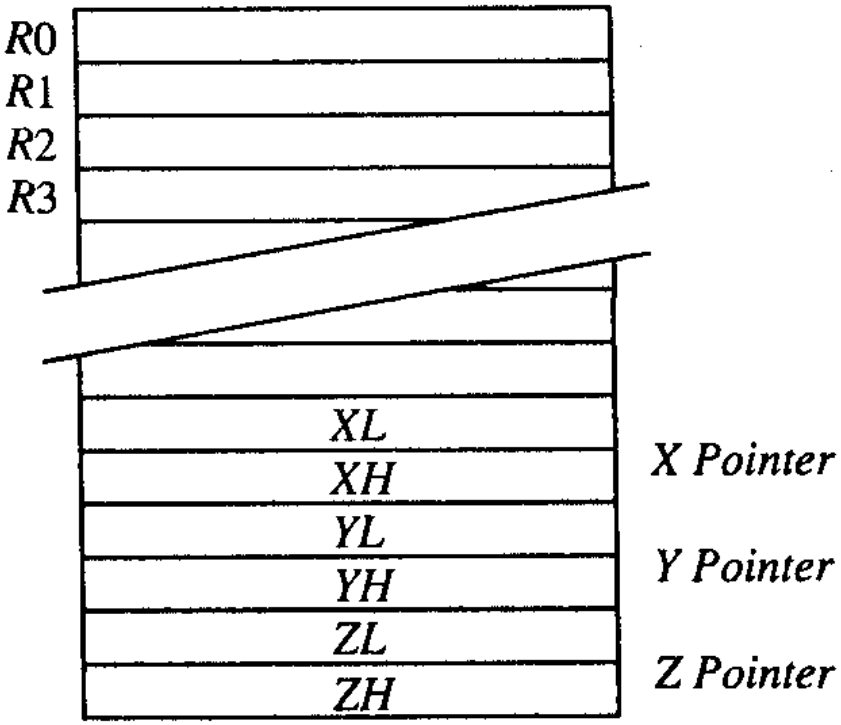


Рисунок 102. Структура регістрового файлу

3.6.2. Регістри вводу-виводу.

В області регістрів вводу-виводу розташовані різні службові регістри (регістр покажчика стеку, регістр стану та ін.), а також регістри керування периферійними пристроями МК. До регістрів вводу-виводу

можна звертатися двома способами: як до регістрів (за допомогою команд IN та OUT) і як до комірок ОЗП. У першому випадку використовуються адреси регістрів вводу-виводу з простору вводу-виводу (0x00…0x3F). У другому випадку адресу регістра вводу-виводу треба збільшити на 0x20.

3.6.3. Стек.

В МК ATMega32 стек реалізований у загальному ОЗП і його глибина визначається лише розміром вільної області пам’яті даних. Оскільки область SRAM займає 2 Кб, то розмір покажчика стеку (Stack Pointer) становить 11 біт. На відміну від деяких інших мікроконтролерів, у ATMega32 *стек росте у бік зменшення адреси*. Як покажчик стеку використовують два регістри вводу-виводу (рис. 103): SPL, розташований за адресою регістра вводу-виводу 0x3D (0x5D – ОЗП), та SPH, розташований за адресою вводу-виводу 0x3E (0x5E – ОЗП).

Вказівник стеку повністю доступний з програми. Крім того, в системі команд МК є команди завантаження в стек (PUSH) і добування зі стеку (POP). Після увімкнення живлення або після скидання покажчик стеку дорівнює нулю, тому на самому початку програми його необхідно проініціалізувати, записавши в нього значення верхньої адреси області пам’яті даних, призначеної для стека.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Рисунок 103. Вказівник стеку

Наприклад:

```
RESET:    rjmp MAIN
           . .  .
MAIN:   ldi R16, 0x00  ; Завантажити регістр r16
        ldi R17, 0x01  ; Завантажити регістр r17
        out SPL, R16   ; Ініціалізація вказівника
        out SPH, R17   ; стека
```

Під час виклику підпрограми адреса наступної команди зберігається в стеку. Значення покажчика стеку зменшується на 2, оскільки для зберігання вмісту лічильника команд потрібно 2 байти. При поверненні з підпрограми ця адреса витягується зі стека і завантажується у лічильник команд (PC), а потім значення покажчика стеку збільшується на 2. Те ж саме відбувається під час переривання.

3.6.4. Пам’ять EEPROM.

ATMega32 містить 1024 байти пам’яті EEPROM даних. Вона організована як окремий простір даних, в якій кожен байт може бути прочитаний і записаний. EEPROM має витривалість щонайменше 10^5 циклів записування та стирання. Доступ до EEPROM здійснюється через регістр адреси EEADR, регістр даних EEData та регістр керування. Ці регістри доступні в просторі вводу/виводу.

Час доступу при записування даних у EEPROM становить приблизно 8,5 мс (8448 циклів каліброваного внутрішнього RC генератора). Сама пам’ять може повідомити програмі про можливість записування наступного байта. З метою запобігання ненавмисних записів у EEPROM необхідно дотримуватися відповідної процедури запису.

Коли EEPROM читається, процесор зупиняється на чотири такти до наступної виконуваної команди. Коли в EEPROM здійснюється записування,

процесор зупиняється на два такти до виконання наступної команди. Формат регістра адреси EEPROM показаний на рис. 104. Старші біти регістра зарезервовані, використовуються лише біти EEAR9..0, що визначають адресу комірки EEPROM, яка читається чи записується.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

Рисунок 104. Формат регістра адреси EEPROM

Формат регістра даних – на рис. 105. Через цей регістр здійснюється читання та записування даних EEPROM. Для керування роботою використовується регістр EECR, зображений на рис. 106.

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 105. Регістр даних EEPROM

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

Рисунок 106. Формат регістра EECR

Біти регістра EECR мають таке призначення:

EERIE – біт дозволу переривань. Дозволяє переривання від EEPROM,

якщо біт встановлений в «1». Переривання генеруються у момент закінчення записування.

EEMWE – біт «ключа» для записування в EEPROM. Коли EEMWE встановлено, встановлення біта EEWE в «1» починає запис у EEPROM за вибраною адресою. Якщо EEMWE дорівнює нулю, встановлення EEWE в «1» не починає запис. Біт EEMWE автоматично скидається у «0» через 4 цикли, тому команда встановлення біта EEWE має йти відразу за командою встановлення біта EEMWE. При записуванні одиничного значення біта EEMWE не слід встановлювати біт EEWE.

EEWE – біт початку записування у EEPROM. Сигнал починає запис у випадку встановлення в «1» біта EEMWE. Біт автоматично скидається в «0» у момент закінчення записування. Програма може опитувати цей біт і чекати появи нуля перед новим зверненням до пам’яті.

EERE – біт дозволу читання. Коли в регістр EEAR записана вірна адреса, біт EERE має бути встановлений в «1» для читання даних. Читання EEPROM займає одну команду і тому дані будуть доступними негайно. Біт автоматично скидається в «0».

Процедура при записування в EEPROM (кроки 3 та 4 не обов’язкові) така:

1. Зачекати, поки EEWE стане дорівнювати нулю.

2. Зачекати, поки біт SPMEN в SPMCR стане дорівнювати нулю.

3. Задати нову адресу EEPROM для EEAR.

4. Задати нові дані EEPROM для EEDR.

5. Встановити біт EEMWE в «1», а біти EEWE та EECR – у нуль.

6. Протягом чотирьох тактів після встановлення EEMWE записати логічну «1» в біт EEWE.

EEPROM не може бути запрограмована у процесі записування флеш-пам’яті програм. Програма повинна перевірити, що програмування флеш-пам’яті програм завершене до початку нового циклу записування в EEPROM. Крок 2 потрібнен лише тоді, коли програмне забезпечення містить Boot Loader. Якщо флеш-пам’ять програм не оновлюється програмою, крок 2

може бути опущений. Переривання між кроками 5 та 6 і зробить цикл запису невдалим, тому що біт EEMWE буде скинутий у нуль. Тому в процесі програмування EEPROM переривання мають бути заборонені.

3.7. Система команд мікроконтролера AVR.

Усі команди мікроконтролерів AVR можна поділити на:

◊ команди пересилання даних;

◊ команди арифметичних операцій та порівняння;

◊ команди операцій із бітами;

◊ команди керування системою;

◊ команди безумовного передавання даних;

◊ команди умовного передавання даних, керування та пропуску команд.

В описі команд використовуються певні позначення, які наведено у наступній таблиці.

Таблиця 24. Позначення, що застосовуються в описуванні команд

Позначення	Регістри й операнди
Rd	Регістр призначення (і джерело) в регістровому файлі
Rr	Регістр джерело в регістровому файлі
R	Результат виконання команди
K	Константа (8 біт)
k	Дані адреси константи для лічильника програм
b	Біт у регістровому файлі або I/O регістр (3 біти)
s	Біт у регістрі статусу (3 біти)
X, Y, Z	Регістр непрямої адресації (X = R27:R26, Y = R29:R28, Z = R31:R30)
P	Адреса порту вводу-виводу
q	Зсув при прямій адресації (6 біт)
STACK	Стек для адреси повернення й завантажених у стек регістрів
SP	Покажчик стека

3.7.1.Група команд пересилання даних.

Команди пересилання даних призначені для пересилання даних між регістрами, між регістрами та комірками пам’яті даних та програм, а також для пересилання даних у порти вводу-виводу. Команди не змінюють регістр ознак.

Усі команди пересилання даних наведено у таблиці 25.

Таблиця 25. Група команд пересилання даних

Мнемоніка	Опис	Операція	Цикли
1	2	3	4
MOV Rd, Rr	Пересилання між регістрами загального призначення	Rd<-Rr	1
MOVW Rd, Rr	Пересилання між парами регістрів	Rd+1:Rd<-Rr+1:Rr	1
LDI Rd, K	Завантаження константи в регістр	Rd<-K	1
LD Rd, X	Непряме читання	Rd<-[X]	2
LD Rd, X+	Непряме читання з постінкрементом	Rd<-[X], X<-X+1	2
LD Rd, -X	Непряме читання з переддекрементом	X<-X-1, Rd<-[X]	2
LD Rd, Y	Непряме читання	Rd<-[Y]	2
LD Rd, Y+	Непряме читання з постінкрементом	Rd<-[Y], Y<-Y+1	2
LD Rd, -Y	Непряме читання з переддекрементом	Y<Y-1, Rd<-[Y]	2
LDD Rd, Y+q	Непряме відносне читання	Rd<-[Y+q]	2
LD Rd, Z	Непряме читання	Rd<-[Z]	2
LD Rd, Z+	Непряме читання з постінкрементом	Rd<-[Z], Z<-Z+1	2
LD Rd, -Z	Непряме читання з переддекрементом	Z<-Z-1, Rd<-[Z]	2
LDD Rd, Z+q	Непряме відносне читання	Rd<-[Z+q]	2
LDS Rd, k	Читання з ОЗП даних	Rd<-[k]	2
ST X, Rr	Непрямий запис	[X]<-Rr	2
ST X+, Rr	Непрямий запис із постінкрементом	[X]<-Rr, X<-X+1	2
ST -X, Rr	Непрямий запис переддекрементом	X<-X-1, [X]<-Rr	2
ST Y, Rr	Непрямий запис	[Y]<-Rr	2
ST Y+, Rr	Непрямий запис з постінкрементом	[Y]<-Rr, Y<-Y+t	2
ST-Y, Rr	Непрямий запис з переддекрементом	Y<-Y-1, [Y]<-Rr	2
STD Y+q, Rr	Непрямий відносний запис	[Y+q]<-Rr	2

Закінчення таблиці 25

1	2	3	4
ST Z, Rr	Непрямий запис	[Z]<-Rr	2
ST Z+, Rr	Непрямий запис з постінкрементом	[Z]<-Rr, Z<-Z+1	2
ST-Z, Rr	Непрямий запис з переддекрементом	Z<-Z-1, [Z]<-Rr	2
STD Z+q, Rr	Непрямий відносний запис	[Z+q]<-Rr	2
STS k, Rr	Запис в ОЗП	[k]<-Rr	2
LPM	Завантаження даних з пам’яті програм у регістр R0	R0<-{Z}	
LPM Rd, Z	Завантаження даних з пам’яті програм	Rd<-{Z}	3
LPM Rd, Z+	Завантаження даних з пам’яті програм і постідекремент Z	Rd<-(Z), Z<-Z+1	3
SPM	Запис у програмну пам’ять	{Z}<-R1:R0	-
IN Rd, P	Пересилання з порта у регістр	Rd<-P	1
OUT P, Rr	Пересилання з регістра у порт	P<-Rr	1
PUSH Rr	Збереження байта в стек	STACK<-Rr	2
POP Rd	Витягування байта зі стека	Rd<-STACK	2

3.7.2. Група команд арифметичних операцій та порівняння.

До групи арифметичних операцій відносять команди, які записані у наступній таблиці. Як бачимо з таблиці, арифметичні команди можна використовувати для додавання і віднімання, збільшення та зменшення, множення як цілих, так і дробових чисел. Команда ділення відсутня. Результат виконання команди може бути записаний у довільний регістр.

Таблиця 26. Група команд арифметичних операцій

Мнемоніка	Опис	Операція	Цикли	Ознаки
1	2	3	4	5
ADD Rd, Rr	Додавання двох регістрів	Rd <- Rd + Rr	1	Z, C, N, V, H
ADC Rd, Rr	Додавання двох регістрів з бітом перенесення	Rd <- Rd + Rr + C	1	Z, C, N, V, H

Закінчення таблиці 26

1	2	3	4	5
ADIW Rd, K	Додавання регістрової пари з константою	Rdh:Rdl <- Rdh:Rdl + K	2	Z, C, N,V, S
SUB Rd, Rr	Віднімання двох регістрів	Rd <- Rd - Rr	1	Z, C, N, V, H
SUBI Rd, K	Віднімання константи із регістра	Rd <- Rd - K	1	Z, C,N, V, H
SBC Rd, Rr	Віднімання двох регістрів із урахуванням біта перенесення	Rd <- Rd – Rr - C	1	Z, C, N, V, H
SBCI Rd, Ko	Віднімання константи із урахуванням біта перенесення	Rd <- Rd - K - C	1	Z, C, N, V, H
SBIW Rd, K	Віднімання константи із регістрової пари	Rdh:Rdl <- Rdh:Rdl - K	2	Z, C, N, V, S
DEC Rd	Декремент регістра	Rd <- Rd - 1	1	Z, N, V
INC Rd	Інкремент регістра	Rd <- Rd + 1	1	Z, N, V
MUL Rd,Rr	Беззнакове множення	R1:R0 <- Rd x Rr	2	Z, C
MULS Rd,Rr	Множення чисел зі знаком	R1:R0 <- Rd x Rr	2	Z, C
MULSU Rd,Rr	Множення числа зі знаком з числом без знака	R1:R0 <- Rd x Rr	2	Z, C
FMUL Rd, Rr	Дробове множення чисел без знака	R1:R0 <- Rd x Rr << 1	2	Z, C
FMULS Rd, Rr	Дробове множення чисел зі знаком	R1:R0 <- Rd x Rr << 1	2	Z, C
FMULSU Rd, Rr	Дробове множення числа зі знаком з числом без знака	R1:R0 <- Rd x Rr <<1	2	Z, C

Команди порівняння призначені для аналізу вмісту регістра. Команди дуже подібні за властивостями до різних модифікацій команди віднімання, але результат операції порівняння нікуди не записується, модифікуються лише біти регістра ознак. Зазвичай, команди порівняння використовуються разом із командами умовного переходу.

Таблиця 27. Група команд порівняння

Мнемоніка	Опис	Операція	Цикли	Ознаки
CP Rd, Rr	Порівняння двох регістрів	Rd - Rr	1	Z, N, V,C, H
CPC Rd, Rr	Порівняння двох регістрів із урахуванням перенесення	Rd - Rr - C	1	Z, N, V, C, H
CPI Rd, K	Порівняння регістра з константою	Rd - K	1	Z, N, V, C, H

3.7.3. Група команд роботи з бітами.

До групи відносять команди логічних операцій. Ці команди подані у наступній таблиці.

Таблиця 28. Група команд роботи з бітами

Мнемоніка	Опис	Операція	Цикли	Ознаки
AND Rd, Rr	Логічне І двох регістрів	Rd <- Rd • Rr	1	Z, N, V
ANDI Rd, Ko	Логічне І регістра та константи	Rd <- Rd • K	1	Z, N, V
EOR Rd, Rr	Виключаюче АБО двох регістрів	Rd <- Rd ⊕ Rr	1	Z, N, V
OR Rd, Rr	Логічне АБО двох регістрів	Rd <- Rd v Rr	1	Z, N, V
ORI Rd, K	Логічне АБО регістра та константи	Rd <- Rd v K	1	Z, N, V
CLR Rd	Скидання всіх розрядів регістра	Rd <- 0	1	Z, N, V
SER Rd	Встановлення всіх розрядів регістра	Rd <- 0FFH	1	-
TST Rd	Перевірка регістра на від’ємне та нульове значення	Rd <- Rd • Rd	1	Z, N, V
NEG Rd	Перетворення у додатковий код	Rd <- 00H - Rd	1	Z, C N, V, H
COM Rd	Перетворення у зворотний код (інверсія регістра)	Rd <- 0FFH - Rd	1	Z, C, N, V

До групи команд операцій з розрядами відносять команди, які записані у наступній таблиці.

Таблиця 29. Група команд операцій з розрядами

Мнемоніка	Опис	Операція	Цикли	Ознаки
CBR Rd, K	Скидання разрядів регістра	Rd<-Rd• $\overline{K}$	1	Z, N, V
SBR Rd, K	Установлення разряду чи розрядів регістра	Rd<-Rd K	1	Z, N, V
CBIA, b	Скидання разряду порта вводу-виводу	A.b<-0	2	-
SBI A, b	Установлення разряду вводу-виводу	A.b<- 1	2	-
BCLRs	Скидання ознаки	SREG.s <- 0	1	SREG.s
BSETs	Установлення ознаки	SREG.s <-1	1	SREG.s
BLD Rd, b	Завантаження разряду регістра з біта T	Rd,b<-T	1	-
BST Rr, b	Запис разряду регістра загального призначення в біт T (SREG)	T<-Rd.b	1	T
CLC	Скидання біта перенесення	C<-0	1	C
SEC	Установлення біта перенесення	C<- 1	1	C
CLN	Скидання біта від'ємного числа	N<-0	1	N
SEN	Установлення біта від'ємного числа	N<- 1	1	N
CLZ	Скидання біта нуля	Z<-0	1	Z
SEZ	Установлення біта нуля	Z<-1	1	Z
CLI	Загальна заборона переривань	I<- 0	1	I
SEI	Загальний дозвіл переривань	I<- 1	1	I
CLS	Скидання біта знака	S<-0	1	S
SES	Установлення біта знака	S<-1	1	S
CLV	Скидання біта переповнення для додаткового коду	V<-0	1	V
SEV	Установлення біта переповнення додаткового коду	V<-1	1	V
CLT	Скидання біта T користувача	T<-0	1	T
SET	Установлення біта T користувача	T<- 1	1	T
CLH	Скидання біта половинного перенесення	H<-0	1	H
SEH	Установлення біта половинного перенесення	H<-1	1	H

До групи команд зсувів відносять команди зсуву вправо чи вліво через біт перенесення та без нього, а також команді обміну місцями тетрад.

Таблиця 30. Група команд зсувів

Мнемоніка	Опис	Операція	Цикли	Ознаки
ASR Rd	Арифметичний зсув вправо	Rd7 -> Rd6 -> Rd5 -> Rd4 -> Rd3 -> Rd2 ->Rd1 -> Rd0	1	Z, C, N, V
LSL Rd	Логічний зсув вліво	C <- Rd7 <- Rd6 <- Rd5 <- Rd4 <- Rd3 <-Rd2 <- Rd1 <- Rd0 <- *0*	1	Z, C, N, V
LSR Rd	Логічний зсув вправо	0 -> Rd7 -> Rd6 -> Rd5 -> Rd4 -> Rd3 -> Rd2 -> Rd1 -> Rd0 -> C	1	Z, C, *N*, V
ROLRd	Зсув вліво через перенесення	C <- Rd7 <- Rd6 <- Rd5 <- Rd4 <- Rd3 <- Rd2<- Rd1 <- Rd0<- C	1	Z,C, N,V
ROR Rd	Зсув вправо через перенесення	C -> Rd7 -> Rd6 -> Rd5 -> Rd4 -> Rd3 -> Rd2-> Rd1 ->Rd0 ->C	1	Z, C, N, V
SWAP Rd	Обмін місцями тетрод	Rd(3 – 0) <-> Hd(7 – 4)	1	-

3.7.4. Група команд керування мікросхемою.

Команди групи (таблиця 31) призначені для задавання режиму роботи мікросхеми та виконують інші допоміжні дії.

Команди групи не змінюють регістра ознак.

Таблиця 31. Група команд керування мікросхемою

Мнемоніка	Опис	Цикли
NOP	Нема операції	1
SLEEP	Перехід у «сплячий» режим	3
WDR	Скидання сторожового таймера	1
BREAK	Зупинка програми. Команда використовується лише при відлагодженні програми	-

3.7.5. Команди передавання керування.

Команди передавання керування призначені для зміни послідовності виконання команд, організації циклів тощо. Команди такого типу поділяються на команди безумовних переходів, що здійснюють перехід незалежно від жодної умови, команди умовних переходів, які перевіряють умови й

здійснюють переходи залежно від умов та команди пропуску за умовою. Команди останнього типу пропускають наступну умову, якщо умова виконалася.

До команд безумовних переходів також віднесені команди виклику та повернення з підпрограм та переривань. Усі команди безумовних переходів наведено в таблиці 32.

Таблиця 32. Команди безумовних переходів

Мнемоніка	Опис	Операція	Цикли
RJMP k	Відносний безумовний перехід	PC <- PC + k + 1	2
IJMP	Непрямий безумовний перехід	PC<-Z	2
JMP k	Прямий безумовний перехід	PC<-k	3
RCALL k	Відносний виклик підпрограми	STACK <- PC+1 PC <- PC + k + 1	3
ICALL	Непрямий виклик підпрограми	STACK <- PC+1 PC<-Z	3
CALL k	Прямий виклик підпрограми	STACK <- PC+1 PC<-k	4
RET	Повернення з підпрограми	PC <- STACK	4
RETI	Повернення з підпрограми опрацювання переривань	PC <- STACK I = 1	4

Таблиця 33. Пропуск команди за умовою

Мнемоніка	Опис	Цикли
CPSE Rd, Rr	Порівняння і пропуск наступної команди при рівності (Rd = Rr)	1/2/3
SBRC Rr, b	Пропуск наступної команди, якщо біт регістра скинутий (Br.b = 0)	1/2/3
SBRS Rr, b	Пропуск наступної команди, якщо біт регістра встановлений (Rr.b = 1)	1/2/3
SBIC A, b	Пропуск наступної команди, якщо біт регістра скинутий (A.b = 1)	1/2/3
SBIS A, b	Пропуск наступної команди, якщо біт регістра встановлений (A.b = 1)	1/2/3

Таблиця 34. Група команд умовних переходів

Мнемоніка	Опис	Цикли
BRBC s, k	Перехід, якщо ознака S регістра SREG скинутий (результат останьої операції додатний)	1/2
BRBS S, k	Перехід, якщо ознака S регістра SREG встановлений (результат останьої операції від'ємний)	1/2
BRCS k	Перехід, є перенесення, тобто біт C = 1	1/2
BRCC k	Перехід, нема перенесення, тобто біт C = 0	1/2
BREQ k	Перехід, якщо результат нульовий, (Z = 1)	1/2
BRNE k	Перехід, якщо результат не нульовий (Z = 0)	1/2
BRSH k	Перехід за умовою «більше або рівно», тобто коли C = 0	1/2
BRLO k	Перехід за умовою «менше», тобто коли C =1	1/2
BRMI k	Перехід за умовою «від'ємне значення» для чисел без знака, якщо N = 1	1/2
BRPL k	Перехід за умовою «додатне значення» для чисел без знака, якщо біт N = 0	1/2
BRGE k	Перехід за умовою «більше або рівно» для чисел зі знаком, тобто за умови (N ⊕ V) = 0	1/2
BRLT k	Перехід за умови «менше» для чисел зі знаком, тобто коли (N ⊕ V) = 1	1/2
BRHS k	Перехід за половинним перенесенням, коли біт H = 1	1/2
BRHC k	Перехід, якщо немає половинного перенесення, коли біт H = 0	1/2
BRTS k	Перехід, якщо біт користувача T встановлений (T = 1)	1/2
BRTC k	Перехід, якщо біт користувача T скинутий (T = 0)	1/2
BRVS k	Перехід за переповненням у додатковому коді V = 1	1/2
BRVC k	Перехід, якщо немає переповнення у додатковому коді V = 0	1/2
BRID k	Перехід, якщо переривання заборонені (I = 0)	1/2
BRIE k	Перехід, якщо переривання дозволені (I = 1)	1/2

Група команд пропуску команди за умовою дозволяє виконати прості дії залежно від умови без створення розгалуження прогами. Всі команди цієї групи можуть дозволити пропускати наступну команду за різних умов. Команди пропуску команди за умовою наведено в таблиці 33.

Команди умовних переходів здійснюють перехід, якщо певний біт чи біти

регістра ознак встановлені у задані значення. Якщо ж біти не встановлені, то здійснюється перехід до наступної команди. Всі команди цієї групи виконують перехід за відносною адресою.

3.8. Порти вводу-виводу.

МК ATMega32 має 4 порти вводу-виводу (ПВВ) A…D, кожен з яких є восьмирозрядним. Отже, загальна кількість ліній вводу/виводу дорівнює 32.

Конфігурування кожної лінії порту (вказання напрямку передавання даних) може проводитися програмно в будь-який момент часу. Вхідні буфери портів побудовані за схемою тригера Шмідта. Для ліній, сконфігурованих як вхідні, є можливість під'єднання внутрішнього підтягуючого резистора опором 30…120 кОм між входом і живленням.

Максимальна навантажувальна здатність вихідних буферів ПВВ при логічному «0» на виході становить 20 мА.

Звертання до портів відбувається через регістри вводу-виводу, причому під кожен порт в адресному просторі вводу/виводу зарезервовано три адреси. Під цими адресами розташовуються регістри:

◊ даних портів PORTx,

◊ напрямку даних DDRx,

◊ виводів порту PINx.

При скиданні МК регістри DDRx і PORTx очищуються, а всі виводи портів після скидання встановлюються в третій (високоімпедансний) стан. Можна задавати конфігурацію кожного виводу незалежно від решти.

PINx насправді не є регістрами, за цими адресами здійснюється доступ до фізичних значень сигналів на виводах порту. Відповідно, вони доступні лише для читання, тоді як PORTx і DDRx доступні й для читання, й для записування.

Запис у порт означає записування потрібного стану для кожного виводу порту у відповідний регістр даних порту PORTx. А читання стану порту виконується або читанням регістра даних порту PORTx, або регістра виводів порту PINx.

На рис. 107 зображена спрощена структурна схема розряду порту.

При читанні регістра виводів порту PINx відбувається зчитування логічних рівнів сигналів, присутніх на виводах порту. А при читанні регістра даних порту PORTx відбувається зчитування даних, які знаходяться в регістрі-засувці порту. Це справедливо як для вхідних, так і для вихідних контактів.

Напрямок передавання даних визначається вмістом регістра передавання даних DDRx. Якщо розряд DDxn цього регістра встановлений в «1», відповідний n-й вивід порту є виходом. Якщо ж розряд DDxn цього регістру скинутий в «0», відповідний вивід порту є входом.

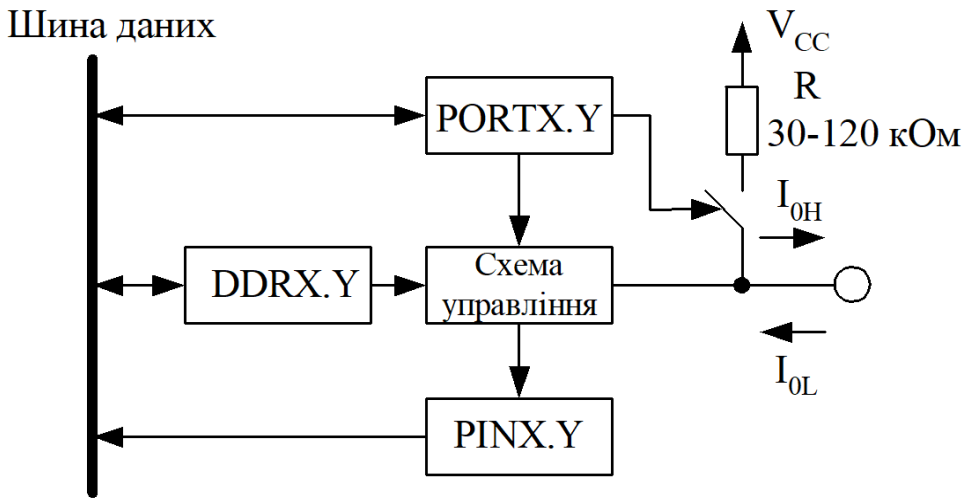


Рисунок 107. Розряд порту вводу-виводу

Управління підтягуючим резистором здійснюється за допомогою регістра даних порту PORTx. Якщо розряд Pxn регістра PORTx встановлений в «1» і відповідний вивід порту є входом, між цим виводом і проводом живлення під’єднується підтягуючий резистор.

Щоб від’єднати підтягуючий резистор, необхідно або скинути відповідний розряд регістра PORTx, або зробити вивід порту виходом (таблиця 35).

Таблиця 35. Задавання режиму роботи порту

DDxn	Pxn	Функція виводу	Резистор	Опис
0	0	Вхід	Від'єднаний	Третій високоімпедансний стан (Hi-Z)
0	1	Вхід	Під'єднаний	При під'єднанні виводу до спільного проводу він є джерелом струму
1	0	Вихід	Від'єднаний	Вихід скинутий в «0»
1	1	Вихід	Від'єднаний	Вихід встановлений в «1»

3.9. Система переривань.

Мікроконтролери ATMega32 мають кілька різних джерел переривань. Усі переривання мають індивідуальні біти дозволу, у які має бути записана «1» для дозволу відповідного переривання. Крім того, є спільний загальний біт дозволу переривань, яким зручно блокувати усі переривання, коли переривання основної програми неможливе.

Коли відбувається переривання, біт глобального дозволу переривань скидається і всі переривання блокуються. У процесі опрацювання переривань можна встановити біт глобального дозволу переривань для виконання вкладених переривань. Також біт глобального дозволу переривань автоматично встановлюється в «1» при виконанні команди повернення з переривання RETI.

Є два основні типи переривань.

Перший тип переривань має ознаку, що встановлюється за деякою подією, а скидання ознаки здійснюється у процесі опрацювання. Якщо переривання не може бути опрацьоване, ознака збереже свій стан і переривання буде викликане при першій можливості.

У другому типі переривання викликається за станом сигналу і, якщо за деякий час переривання не буде опрацьоване, то воно буде втрачене.

При використанні інструкції CLI переривання буде негайно скасоване, навіть якщо воно надійшло у процесі виконання інструкції CLI. Після опрацювання переривання в основній програмі буде виконано хоча б одну команду перед будь-яким новим перериванням.

Таблиця 36. Розміщення векторів скидання та переривань

Номер вектора	Адреса підпрограми	Назва	Визначення
1	$000 + V1	RESET	Скидання від зовнішнього виводу, за увімкненням чи пропаданням живлення, від Watchdog-таймера або сигналу від JTAG
2	$002 + V2	INT0	Зовнішній запит переривань 0
3	$004 + V2	INT1	Зовнішній запит переривань 1
4	$006 + V2	INT2	Зовнішній запит переривань 2
5	$008 + V2	TIMER2 COMP	Сигнал порівняння таймера-лічильника 2
6	$00A + V2	TIMER2 OVF	Сигнал порівняння таймера-лічильника 2
7	$00C + V2	TIMER1 CAPT	Сигнал фіксації таймера-лічильника 1
8	$00E + V2	TIMER1 COMPA	Сигнал порівняння A таймера-лічильника 1
9	$010 + V2	TIMER1 COMPB	Сигнал порівняння B таймера-лічильника 1
10	$012 + V2	TIMER1 OVF	Переповнення таймера-лічильника 1
11	$014 + V2	TIMER0 COMP	Сигнал порівняння таймера-лічильника 0
12	$016 + V2	TIMER0 OVF	Сигнал переповнення таймера-лічильника 0
13	$018 + V2	SPI	Сигнал закінчення передавання даних через послідовний порт SPI
14	$01A + V2	USART	Сигнал закінчення приймання даних через USART
15	$01C + V2	USART	Сигнал вільності буфера передавання USART
16	$01E + V2	USART	Сигнал закінчення передавання даних через USART
17	$020 + V2	ADC	Перетворення ADC завершено
18	$022 + V2	EE_RDY	EEPROM готовий
19	$024 + V2	ANA_COMP	Аналоговий компаратор
20	$026 + V2	TWI	Сигнал переривання від двовивідного послідовного інтерфейсу TWI
21	$028 + V2	SPM_RDY	Пам’ять програм готова до обміну

Також варто відзначити, що регістр статусу не зберігається автоматично при вході в переривання, а його збереження має бути виконане програмою

користувача. Для кожного джерела переривань є окремий вектор, розміщений у спеціальній області пам’яті програм. Найнижчі адреси в пам’яті програм за замовчуванням призначені для вектора скидання та векторів переривань. Повний список векторів показано в таблиці 36.

Параметри V1 та V2 визначаються бітами BOOTRST та IVSEL. Біт BOOTRST задається у слові конфігурування мікросхеми, а біт IVSEL – у головному регістрі керування перериваннями.

Якщо біт BOOTRST запрограмовано, пристрій перейде до адреси скидання в boot-блоці. Коли біт IVSEL в регістрі GICR встановлено, вектори переривань будуть переміщені на початок boot-блоку флеш-пам’яті програм. Усі можливі варіанти значень V1 та V2 наведено в таблиці 37

Таблиця 37. Можливі варіанти значень V1 та V2

BOOTRST	IVSEL	V1	V2
1	0	$0000	$0000
1	1	$0000	Початок boot-блоку
0	0	Початок boot-блоку	$0000
0	1	Початок boot-блоку	Початок boot-блоку

Порядок у списку також визначає пріоритет рівнів переривань. Адреса, що має менший номер, задає вищій рівень пріоритету. Скидання (RESET) має найвищий пріоритет, а наступним іде INT0 – зовнішній запит переривання 0. Деякі із векторів переривань можуть переміщуватися у процесі роботи за допомогою встановлення біта IVSEL в головному регістрі управління перериванням (GICR). Формат головного регістра управління перериванням GICR зображено на рис. 108.

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 108. Головний регістр керування перериваннями GICR

Біти INT2 – INT0 – біти дозволів зовнішніх запитів переривань. Логічна «1» дозволяє відповідний зовнішній запит переривань.

Біт IVCE використовується у якості «ключа» для зміни стану біта IVSEL. Для зміни стану біта IVSEL необхідно, щоб біт IVCE був встановлений в «1». Біт IVCE скидається через 4 цикли після записування даних або після записування даних у біт IVSEL. Для уникнення переходів на хибні вектори переривань встановлення біта IVCE в одиницю забороняє переривання.

3.10. Таймери-лічильники мікросхеми.

Мікросхема ATMega32 у своїй структурі має три таймери-лічильники:

◊ 8-бітний таймер-лічильник 0 (Timer/Counter0);

◊ 16-бітний таймер-лічильник 1 (Timer/Counter1);

◊ 8-бітний таймер-лічильник 2 (Timer/Counter2).

Кожен із таймерів-лічильників орієнтований на виконання свого кола задач і має відповідні особливості, наприклад, таймер-лічильник 2 має незалежне джерело синхронізації.

3.10.1. Попередні подільники таймерів-лічильників.

Попередній подільник призначений для попереднього поділу вхідної частоти на певний коефіцієнт і використовується для збільшення періоду відліку тактового сигналу без збільшення розрядності таймерів-лічильників.

Таймер-лічильник 1 і таймер-лічильник 0 використовують один і той же модуль попереднього подільника, але можуть мати різні налаштування коефіцієнта попереднього поділу. Таймер-лічильник 2 використовує власний попередній подільник.

Будова попереднього подільника таймерів-лічильників 0 та 1 зображена на рис. 109. Таймери-лічильники 0 та 1 можуть бути синхронізовані безпосередньо системним тактовим сигналом, що забезпечує найбільшу швидкість відліку. Крім того, таймери-лічильники 0 та 1 можуть синхронізуватися тактовим сигналом від одного з чотирьох відводів від

попереднього подільника із частотою синхронізації, меншою за частоту тактового сигналу на 8, 64, 256 чи 1024 ($f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, або $f_{CLK\_I/O}/1024$ відповідно).

Попередній подільник таймерів-лічильників 0 та 1 не дозволяє ділити сигнал із зовнішнього входу. Вибір потрібного джерела синхронізації здійснюється за допомогою бітів CS12-CS10 для таймера-лічильника 1 та бітами CS02-CS00 для таймера-лічильника 0.

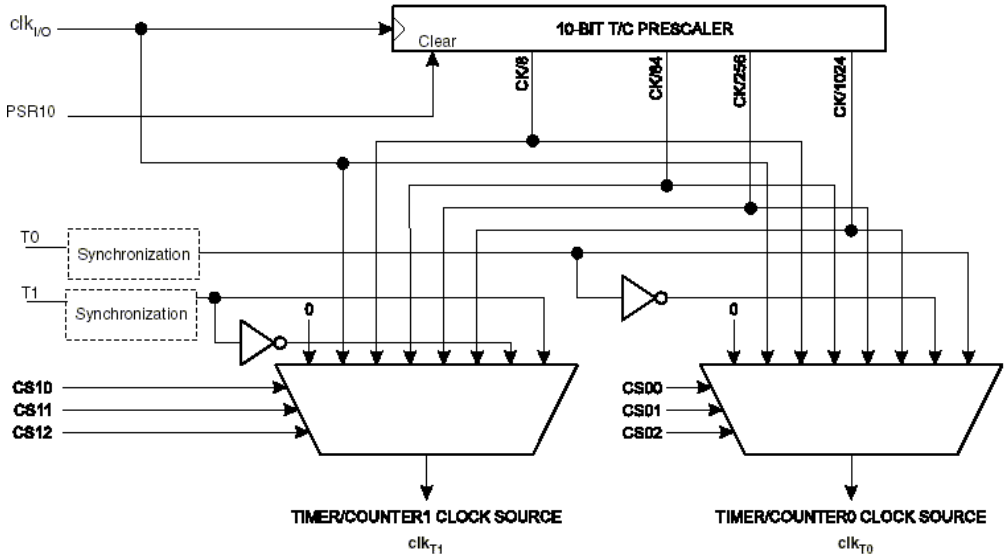


Рисунок 109. Будова попереднього подільника таймерів-лічильників 0 та 1

Попередній подільник таймера-лічильника 2 може ділити частоту вхідного сигналу. Будова попереднього подільника таймера-лічильника 2 зображена на рис. 110. Зі структурної схеми попереднього подільника таймера-лічильника 2 бачимо, що він може синхронізуватися тактовим сигналом від одного з шести відводів від попереднього подільника із частотою синхронізації, меншою за частоту вхідного сигналу у 8, 32, 64, 128, 256 чи 1024 рази. Джерело вхідного сигналу попереднього подільника таймера-лічильника 2 обирається бітом AS2 регістра ASSR і дозволяє під'єднати до входу попереднього подільника як зовнішній вхід, так і системний тактовий сигнал.

Вибір потрібного джерела синхронізації здійснюється за допомогою бітів CS22-CS20.

Попередній подільник не має синхронізації від таймерів-лічильників і працює незалежно від них. Наприклад, якщо попередній подільник налаштовано на коефіцієнт поділу 1024, то перший імпульс після встановлення режиму роботи лічильника може надійти і через 1, і через 1023 тактові сигнали. Якщо необхідно встановити час до надходження першого імпульсу, можна скористатися бітами скидання попередніх подільників.

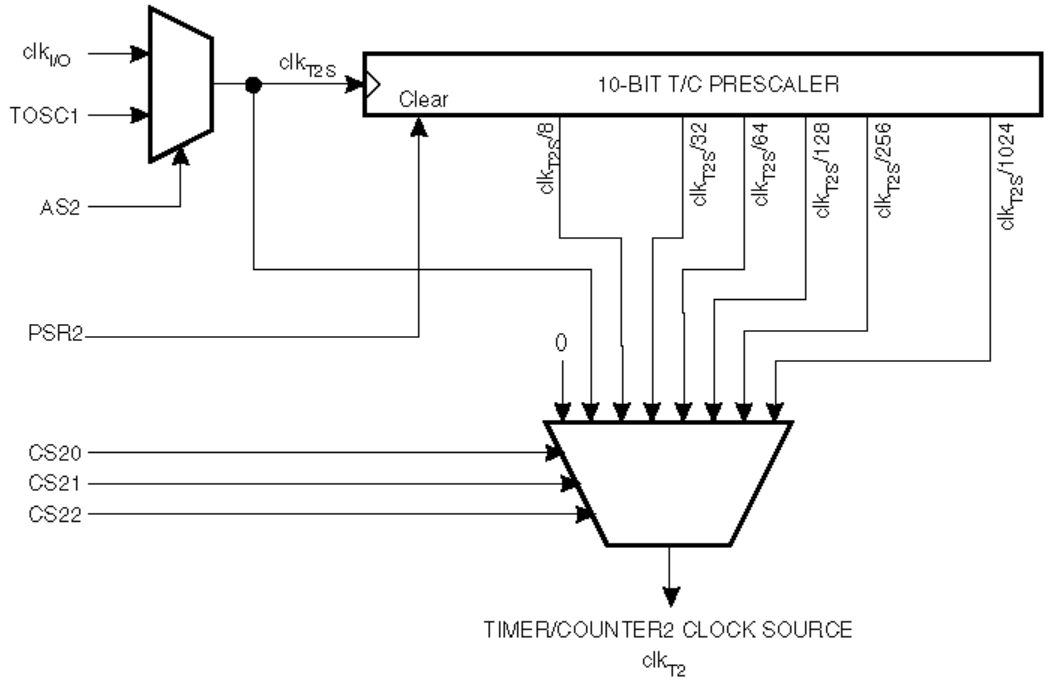


Рисунок 110. Будова попереднього подільника таймера-лічильника 2

Біт скидання попереднього подільника таймерів-лічильників 0 та 1 носить назву PSR10, а біт PSR2 скидає попередній подільник таймера-лічильника 2. Біти PSR10 та PSR2 знаходяться у регістрі SFOIR, формат регістра SFOIR наведено на рис. 111.

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 111. Регістр SFOIR

Запис у біт PSR10 одиниці скидає попередній подільник таймерів-лічильників 0 та 1, аналогічно запис біта PSR2 скидає попередній подільник таймера-лічильника 2. Самі біти PSR10, PSR2 скидаються апаратно після виконання відповідної операції. Запис у біти PSR10, PSR2 нуля не виконує жодної дії. При читанні бітів PSR10 та PSR2 завжди читається нуль, якщо скидання виконане.

3.10.2. Восьмирозрядні таймери-лічильники 0 та 2.

Таймери-лічильники 0 та 2 – це універсальні восьмирозрядні лічильники з модулем порівняння і підтриманням функцій широтно-імпульсної модуляції (ШІМ або PWM). Вони дозволяють формувати задані проміжки часу для роботи в режимі реального часу, а також можуть використовуватися як генератори сигналів.

Восьмирозрядні таймери-лічильники 0 та 2 дуже подібні за своєю будовою, програмуванням та режимами роботи.

Основні особливості таймерів-лічильників 0 та 2:

◊ наявність модуля порівняння;

◊ подвійна буферизація при записуванні в регістри порівняння;

◊ скидання таймера за рівності значення заданому;

◊ наявність симетричного широтно-імпульсного модулятора;

◊ два незалежних джерела переривання TOV0, OCF0 для таймера-лічильника 0, а також TOV2, OCF2 для таймера-лічильника 2.

Таймер-лічильник 2 має іншу будову попереднього подільника та може працювати в асинхронному режимі.

Вхідним сигналом таймера-лічильника 0 може бути як зовнішній сигнал з виводу мікросхеми, так і сигнал із тактового генератора, пропущений через попередній подільник. Спрощена блок-схема восьмирозрядного таймера-лічильника 0 наведена на рис. 112.

Підрахунок кількості імпульсів здійснює лічильник TCNT0 таймера-лічильника. Сигнали про напрям відліку, необхідність скидання та імпульси,

що підраховуються, надходять на нього із блока керування (Control Logic), який керується регістром керування TCCR0. Також блок керування (Control Logic) видає сигнал переповнення таймера TOV0.

Імпульси, що підраховуються, надходять на блок керування від блока вибору сигналу синхронізації (Clock Select), котрий визначає джерело тактового сигналу та режим роботи таймера чи лічильника.

Таймер-лічильник може працювати як від внутрішнього тактового генератора через попередній подільник, так і від зовнішнього тактового сигналу, що надходить на вхід T0.

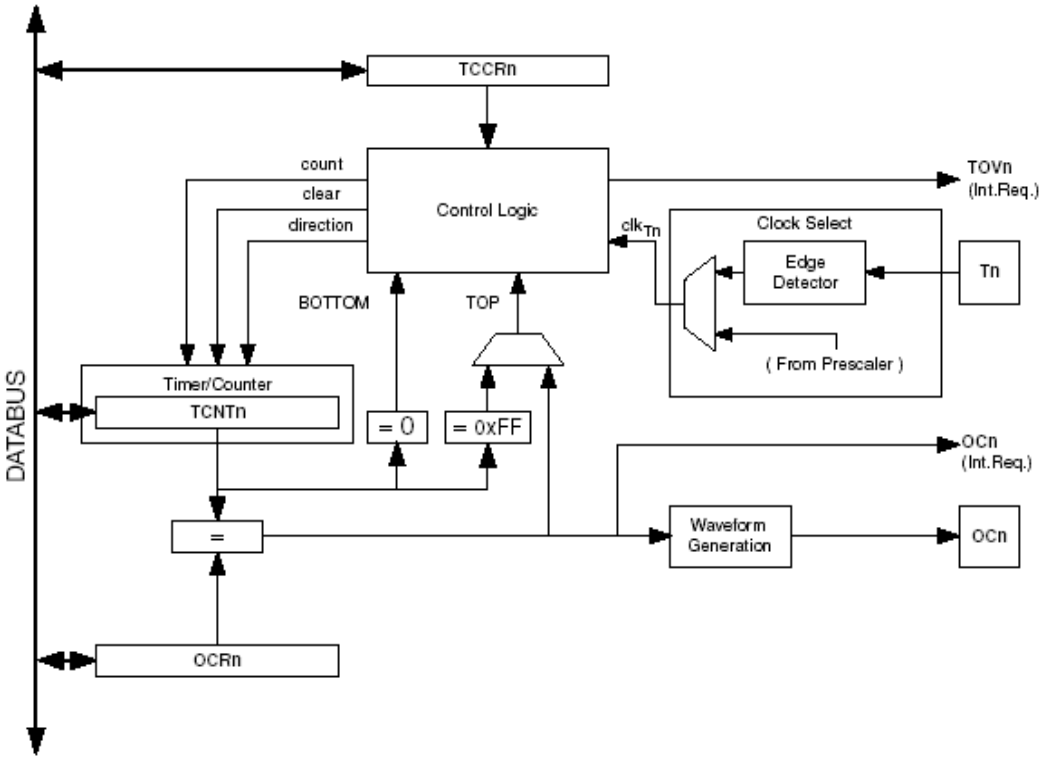


Рисунок 112. Спрощена структурна схема таймера-лічильника 0

Схема вибору джерела тактового сигналу передає тактові імпульси вибраного джерела на вхід таймера-лічильника. Кожен імпульс цього сигналу збільшує (або зменшує) значення регістра TCNT0. Якщо не вибрано жодне з джерел тактового сигналу, таймер-лічильник зупиняється. Джерело синхронізації вибирається бітами CS02-0, розташованими у регістрі керування таймера-лічильника TCCR0.

Код, до котрого дорахував таймер-лічильник 0, надходить на цифровий компаратор, де порівнюється з умістом регістра OCR0. За рівності значень сигнал надходить на запит переривань OC0 та на формувач вихідного сигналу (Waveform Generation). Регістр відліку таймера-лічильника (TCNT0) і регістр порівняння OCR0 є восьмирозрядними.

Кожен із запитів переривань (позначених як Int.Req на рис. 112) індивідуально маскуються в регістрі маски таймерів TIMSK і можуть бути перевірені у регістрі запитів переривань таймерів TIFR незалежно від наявності маскування.

Регістри TIFR і TIMSK не показані на рис. 112, так як вони є спільними для усіх таймерів.

Структурна схема таймера-лічильника 2 відрізняється іншою організацією блока отримання вхідного сигналу і наявністю блоків, що дозволяють асинхронні операції. Будова таймера-лічильника 2 наведена на рис. 113.

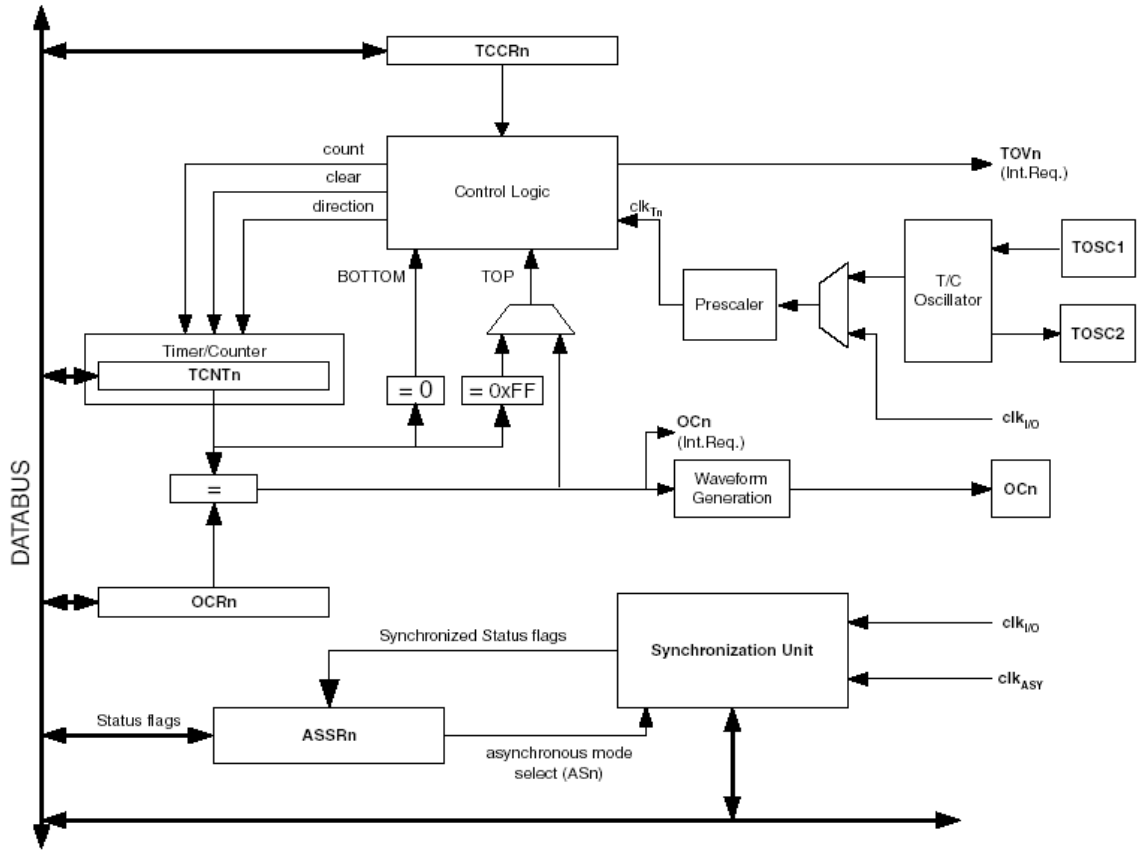


Рисунок 113. Спрощена структурна схема таймера-лічильника 2

У таймер-лічильник 2 введено внутрішній асинхронний генератор (T/C Oscillator), котрий дозволяє під'єднання як зовнішнього тактового сигналу, так і пряме під'єднання кварцового чи керамічного резонатора. Крім того, з'явився пристрій вибору джерела тактового сигналу, внутрішній попередній подільник та блок синхронізації (Synchronization Unit) із регістром ASSR2, що містить біти увімкнення асинхроного режиму та біти синхронізації.

Вхід таймера-лічильника 0.

У якості вхідного сигналу таймера-лічильника 0 використовується або сигнал із входу T0, або сигнал із попереднього подільника, або тактовий сигнал синхронізації пристроїв вводу-виводу.

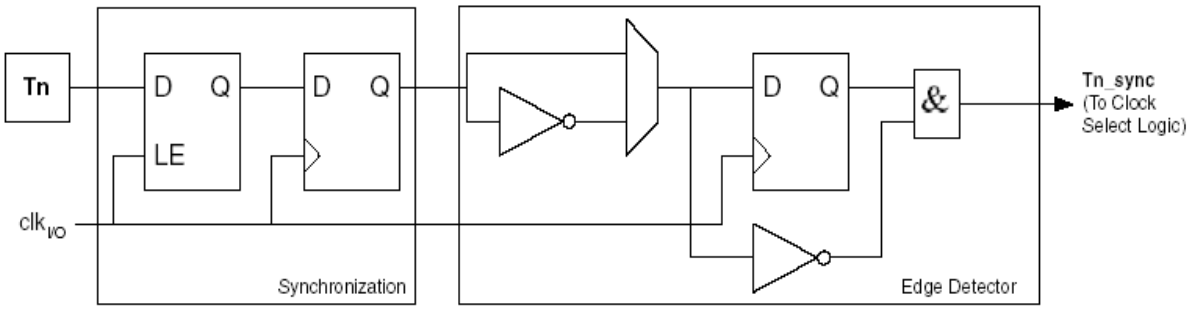


Рисунок 114. Вхідне коло лінії T0

Таймер-лічильник може отримувати імпульси із входу T0 і тоді, коли відповідна лінія вводу-виводу конфігурована як вихід. Фронт сигналу T0, за яким здійснюється перемикання таймера-лічильника, вибирається програмним шляхом. Схема формування сигналу на вході лічильника наведена на рис. 114.

Схема складається із двох основних блоків: синхронізатора Synchronizator та детектора фронту Edge Detector. При високому рівні на лінії $clk_{I/O}$ сигнал з лінії пропускається через вхідний тригер і надходить на другий тригер синхронізатора, де фіксується за переднім фронтом наступного імпульсу сигналу $clk_{I/O}$. Така схемотехніка унеможливлює появу на виході синхронізатора імпульсів малої тривалості. Сигнал із синхронізатора надходить на детектор фронту, який формує імпульси тривалістю один період сигналу $clk_{I/O}$ за фронтом

сигналу з виходу синхронізатора. Вибір фронту, за яким здійснюється формування імпульсу, здійснюється залежно від стану бітів CS2-0.

Синхронізація та логіка детектора фронту створюють затримку від 2,5 до 3,5 тактових циклів від фронту імпульсу до моменту перемикання лічильника. Також зі схеми випливає, що перемикання вхідного тактового сигналу має бути виконане тоді, коли лінія T0 буде стабільною протягом, принаймні, одного періоду сигналу $f_{clkI/O}$. З іншого боку, кожна половина періоду зовнішнього сигналу повинна бути більшою, ніж один такт системного тактового сигналу, для забезпечення правильного відбору фронтів. Тому для надійного введення тактового сигналу рекомендується, щоб максимальна частота зовнішнього джерела синхронізації не перевищувала $f_{clkI/O}/2,5$.

Сигнал із наведеного вище вхідного кола надходить на схему вибору джерела синхронізації, яка наведена на рис. 115.

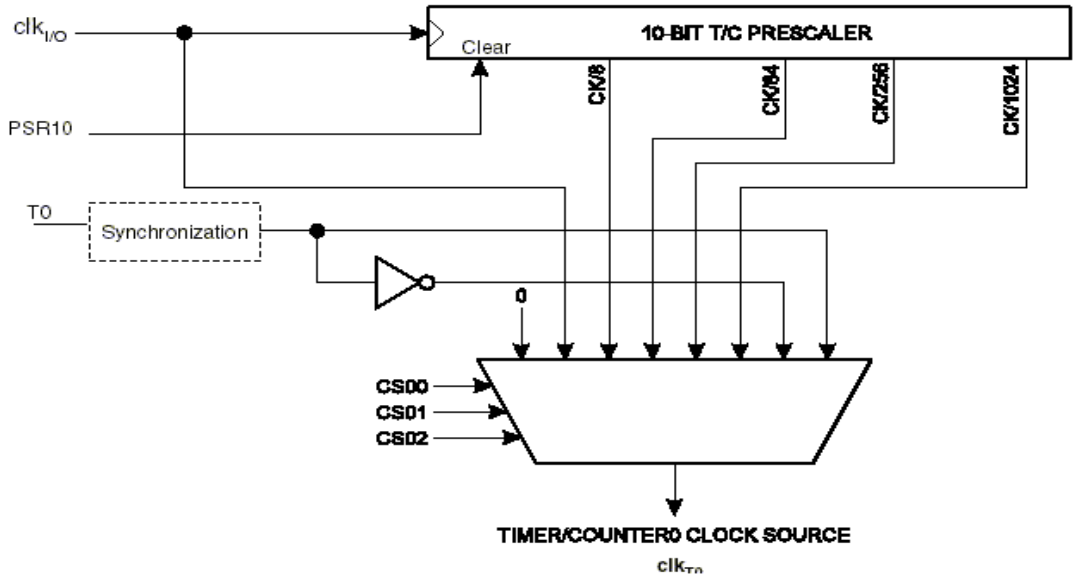


Рисунок 115. Схема вибору джерела синхронізації лічильника 0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 116. Формат регістра TCCR0

Вибір джерела синхронізації здійснюється бітами CS2-0 у регістрі TCCR0 (рис. 116). Біти мають наступне призначення (таблиця 38).

Таблиця 38. Джерела тактового сигналу для таймера-лічильника 0

CS02	CS01	CS00	Джерело тактового сигналу
0	0	0	Немає сигналу. Таймер-лічильник зупинений
0	0	1	$clk_{I/O}$ (Без попереднього подільника)
0	1	0	Частота з попереднього подільника $clk_{I/O}/8$
0	1	1	Частота з попереднього подільника $clk_{I/O}/64$
1	0	0	Частота з попереднього подільника $clk_{I/O}/256$
1	0	1	Частота з попереднього подільника $clk_{I/O}/1024$
1	1	0	Зовнішнє джерело (лінія T0). Перемикання лічильника за спадом імпульсу на лінії
1	1	1	Зовнішнє джерело (лінія T0). Перемикання лічильника за переднім фронтом імпульсу на лінії

Вхід таймера-лічильника 2.

Будова входу таймера-лічильника 2 (рис. 117) відрізняється від входу таймера-лічильника 0. Вхід таймера-лічильника 2 дозволяє пряме під’єднання резонатора між виводами TOSC1 та TOSC2. За необхідності замість резонатора на вхід TOSC1 може бути поданий зовнішній тактовий сигнал.

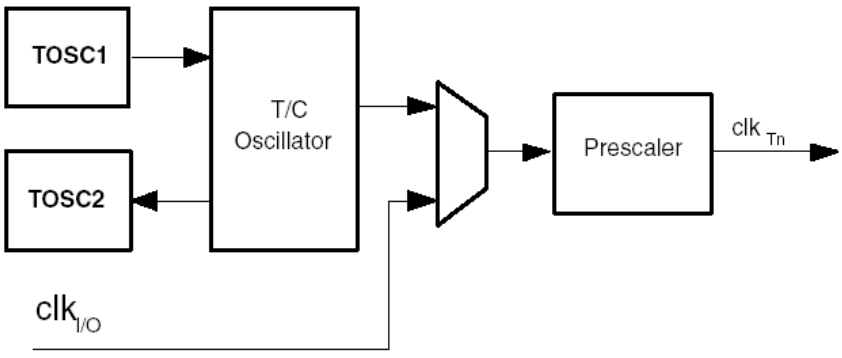


Рисунок 117. Вхідне коло таймера-лічильника 2

У будь-якому випадку частота тактового сигналу на лінії має бути меншою у 4 рази, ніж частота основного тактового сигналу.

Вхідне коло містить блок генератора (T/C Oscilator) та попередній подільник (Prescaler). За необхідності, мультиплексор може перемкнути замість виходу генератора системний тактовий сигнал CLK_{IO}. Перемикання джерела тактового сигналу здійснюється бітом AS2 регістра ASSR (рис. 118). При одиничному значенні біта AS2 джерелом тактового сигналу таймера-лічильника 2 є тактовий генератор, а при нульовому – тактовий сигнал CLKIO. Коли біт AS2 встановлено, виводи PC6 та PC7, що відповідають входам TOSC1 та TOSC2, від'єднуються від порту C та використовуються виключно генератором.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 118. Регістр ASSR

Сигнал із наведеного вище вхідного кола надходить на схему вибору джерела синхронізації, яка наведена на рис. 119.

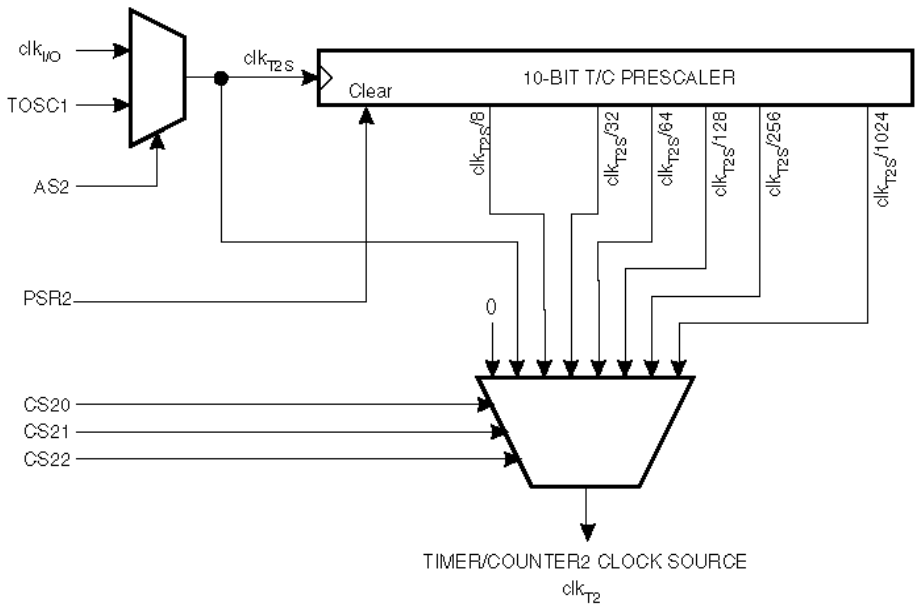


Рисунок 119. Вибір джерела тактової частоти таймера-лічильника 2

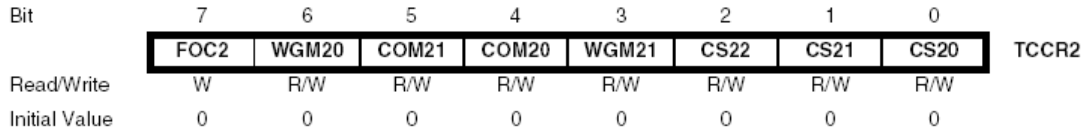


Рисунок 120. Формат регістра TCCR2

Таблиця 39. Джерела тактового сигналу для таймера-лічильника 2

CS02	CS01	CS00	Джерело тактового сигналу
0	0	0	Немає сигналу. Таймер-лічильник зупинений
0	0	1	Тактовий сигнал лічильника Clk без попереднього подільника
0	1	0	Частота з попереднього подільника Clk/8
0	1	1	Частота з попереднього подільника Clk/32
1	0	0	Частота з попереднього подільника Clk/64
1	0	1	Частота з попереднього подільника Clk/128
1	1	0	Частота з попереднього подільника Clk/256
1	1	1	Частота з попереднього подільника Clk/1024

Вибір джерела синхронізації здійснюється бітами CS2-0 у регістрі TCCR2 (рис. 120). Значення бітів регістра відрізняється від бітів регістра TCCR0 і наведені у таблиці 39.

3.10.2.1. Режими роботи таймерів 0 та 2.

Таймери-лічильники 0 та 2 можуть використовуватися у таких режимах роботи:

◊ нормальному (Normal);

◊ зі скиданням за рівністю (CTC);

◊ швидкої широтно-імпульсної модуляції (Fast PWM);

◊ широтно-імпульсної модуляції із коректною фазою Phase Correct PWM.

Встановлення режиму роботи таймера-лічильника 0 здійснюється за допомогою бітів WGM01:0 у регістрі TCCR0 (табл. 40), а таймера-лічильника 2 – за допомогою бітів WGM21:0. Режими роботи ідентичні й наведені у наступній таблиці.

Таблиця 40. Режими роботи таймерів-лічильників 0 та 2

WGMn1	WGMn0	Режим роботи
0	0	Нормальний режим (Normal)
0	1	Режим широтно-імпульсної модуляції із коректною фазою Phase Correct. PWM
1	0	Режим зі скиданням за рівністю (CTC)
1	1	Режим швидкої широтно-імпульсної модуляції (Fast PWM)

У таблиці n дорівнює 0 для таймера-лічильника 0 та дорівнює 2 для таймера-лічильника 2.

Нормальний режим.

Нормальний режим (Normal) – найпростіший режим роботи. У цьому режимі таймер-лічильник n (0 чи 2) рахує від заданого значення до значення 0FF, потім скидається в нуль і продовжує відлік далі. Для таймера-лічильника 0 при переході зі стану 0FF у стан 00 встановлюється в «1» біт TOV0, а для таймера 2 – біт TOV2 (у поєднанні з перериванням за переповненням таймера, виклик якого автоматично очищає біт TOVn). Цей режим може використовуватися для підрахунку часових інтервалів чи кількості подій. У даному режимі роботи блок цифрового компаратора може використовуватися довільним чином.

Режим зі скиданням за рівністю.

У режимі зі скиданням за рівністю (CTC – Clear Timer on Compare Match) регістр компаратора таймера-лічильника n (n дорівнює 0 чи 2) OCRn використовується для збереження максимального значення, до якого може рахувати лічильник. У режимі CTC лічильник скидається в нуль, коли значення лічильника TCNTn стає рівним OCRn, тобто регістр OCRn визначає верхнє значення для таймера-лічильника n. У цьому режимі роботи ознака переповнення таймера-лічильника TOVn не встановлюється до тих пір, поки таймер-лічильник не перейде зі стану 0FF у нульовий стан. Це може відбутися, якщо значення в лічильнику перевищує значення регістра OCRn.

У цьому режимі роботи блок цифрового компаратора може використовуватися для формування сигналу із частотою, що надходить на вихід мікросхеми.

Часова діаграма роботи лічильника в режимі роботи зображена на рис. 121.

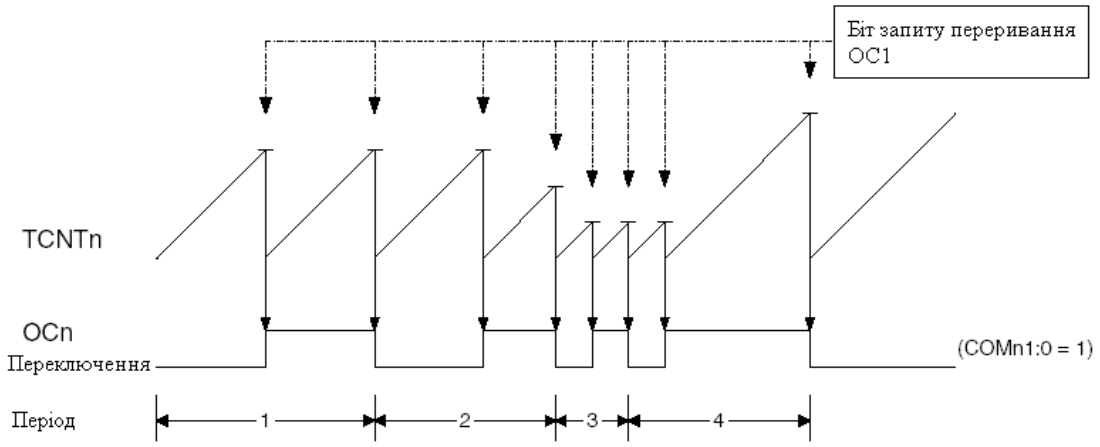


Рисунок 121. Режим зі скиданням за рівністю

Для генерації сигналу на виводі в режимі CTC, вихід OCn може бути встановлений у режим перемикання при кожній рівності (COM n 1 : 0 = 01). Значення на виводі OCn не буде видаватися на вивід порту, поки основна функція порту не налаштована на введення даних. Максимальна частота сигналу на виході OCn дорівнює $f_{OCn} = f_{CLK/2}$ і досягається, якщо регістр OCRn встановлюється в нуль (0x00). В інших випадках частота сигналу на виході визначається за формулою

$$f_{OCn} = \frac{f_{clk}}{2 \cdot N \cdot (1 + OCRn)}$$

де N є значенням коефіцієнта поділу попереднього подільника; f_{CLK} – тактова частота лічильника на вході попереднього подільника.

Режим швидкої широтно-імпульсної модуляції.

У режимі швидкої широтно-імпульсної модуляції (Fast PWM) лічильник n рахує від 0 до максимального значення. Якщо значення у лічильнику перевищує задане у регістрі OCRn, здійснюється перемикання відповідного виходу. Вихід повертається у попередній стан при скиданні лічильника. При кожному скиданні лічильника відбувається встановлення ознаки TOVn, а при досягненні значення OCRn – ознаки OCn. У даному режимі роботи центр імпульсу на виході «плаває» відносно початку відліку.

Режим швидкої широтно-імпульсної модуляції забезпечує високу частоту повторення ШІМ сигналу. Цей режим відрізняється роботою лише за одним схилом. Через роботу за одним схилом робоча частота швидкого режиму ШІМ може бути вдвічі вищою, ніж при роботі в режимі з коректною фазою, що зменшує габарити потрібного фільтра.

Часова діаграма для режиму швидкої ШІМ показана на рис. 122.

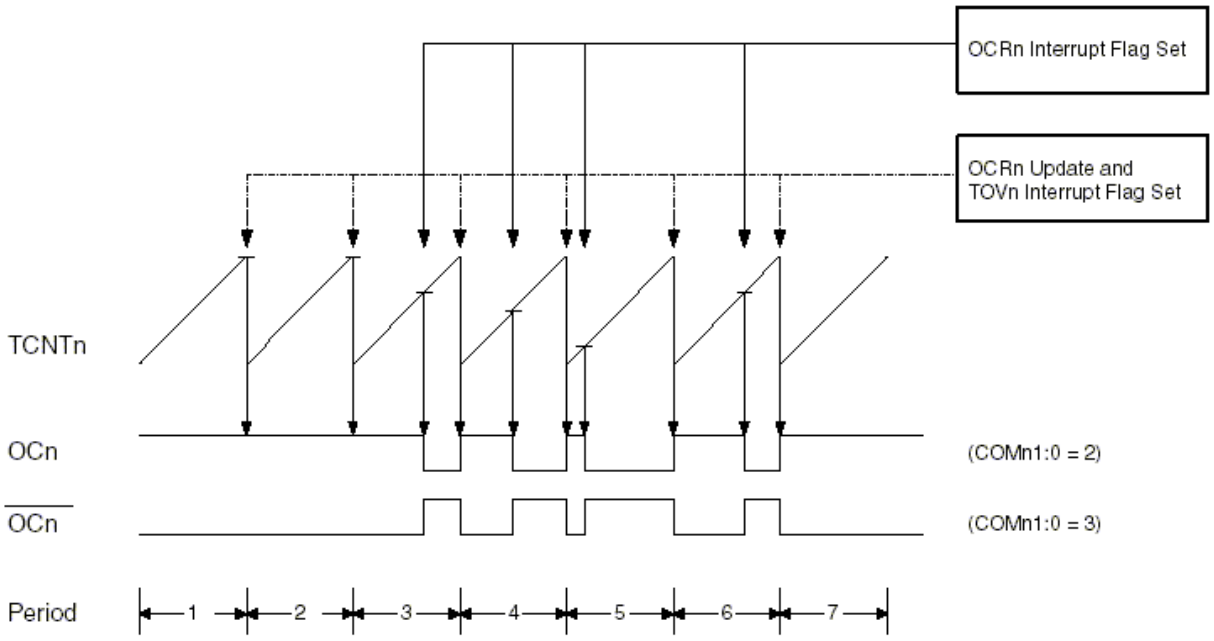


Рисунок 122. Режим швидкої широтно-імпульсної модуляції

Значення регістра TCNT0 для ілюстрації показані як гістограма, яка включає режим неінвертованого й інвертованого ШІМ. Значення частоти вихідного сигналу визначається за допомогою формули

$$f_{OCnPWM} = \frac{f_{clk}}{256\ N}$$

де N є значенням коефіцієнта поділу попереднього подільника, а f_{clk} – тактова частота на вході попереднього подільника.

Режим широтно-імпульсної модуляції з коректною фазою.

У режимі широтно-імпульсної модуляції із коректною фазою (PWM, Phase Correct) лічильник n рахує від 0 до максимального значення, а потім – від максимального значення 0FF до 0. Значення 0FF та 0 зберігаються у регістрі TCNTn протягом одного циклу. Якщо значення в лічильнику перевищує задане у регістрі OCRn, то здійснюється перемикання відповідного виходу. Коли в процесі зворотного відліку значення у лічильнику стане меншим, ніж значення, задане у OCRn, вихід повернеться у початковий стан. У даному режимі роботи положення центру імпульсу на виході відносно початку відліку не залежить від вмісту регістра OCRn. Частота зміни сигналу на виході у цьому режимі роботи менша, ніж у режимі швидкої широтно-імпульсної модуляції. Проте, завдяки симетрії імпульсу та постійній частоті, цей метод має свої переваги.

Часова діаграма для режиму широтно-імпульсної модуляції із коректною фазою зображена на рис. 123.

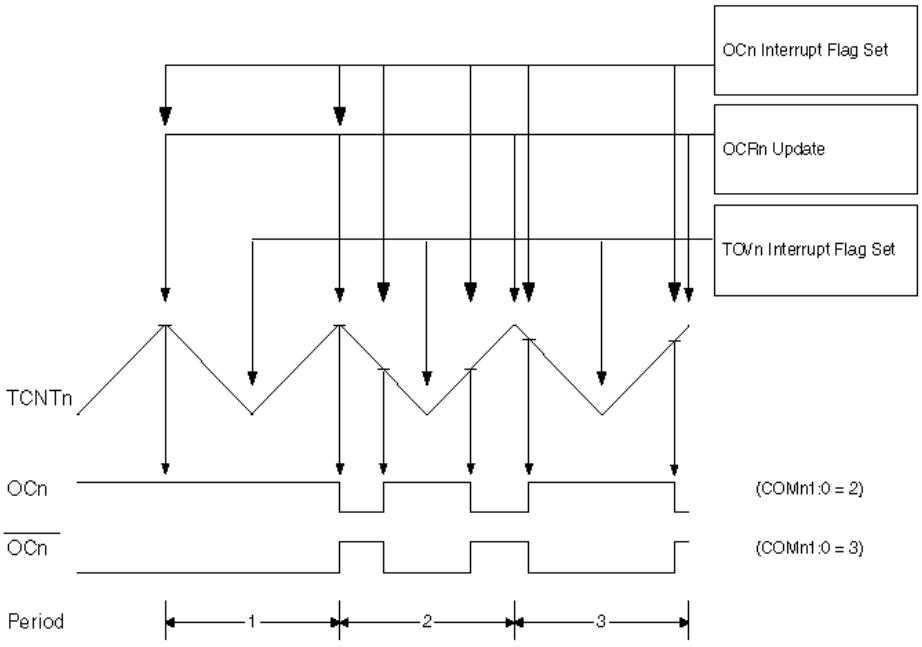


Рисунок 123. Часова діаграма режиму широтно-імпульсної модуляції з коректною фазою

Значення TCNTn зображене у вигляді графіка. На діаграмі також наведені неінвертований та інвертований виходи ШІМ.

З рис. 123 випливає, що TOVn встановлюється у момент встановлення мінімального значення у регістрі TCNTn, а біт OCn – у момент рівності значень TCNTn та OCRn. Значення у регістрі OCR0 змінюється в момент набуття регістром TCNTn максимального значення.

Частота вихідного сигналу визначається за формулою

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{510\ N}$$

де N – значення коефіцієнта поділу попереднього подільника.

Схема порівняння.

Схема порівняння таймерів-лічильників 0 та 2 побудована на основі 8-разрядного компаратора, котрий порівнює вміст лічильника TCNTn (n = 0 або 2) з регістром порівняння (OCRn). Кожен раз, коли вміст регістра TCNTn стає рівним OCRn, компаратор сигналізує сигналом рівності. Сигнал рівності перемикає ознаку OCF0, що запускає запит переривання від таймера. Будова схеми порівняння зображена на рис. 124.

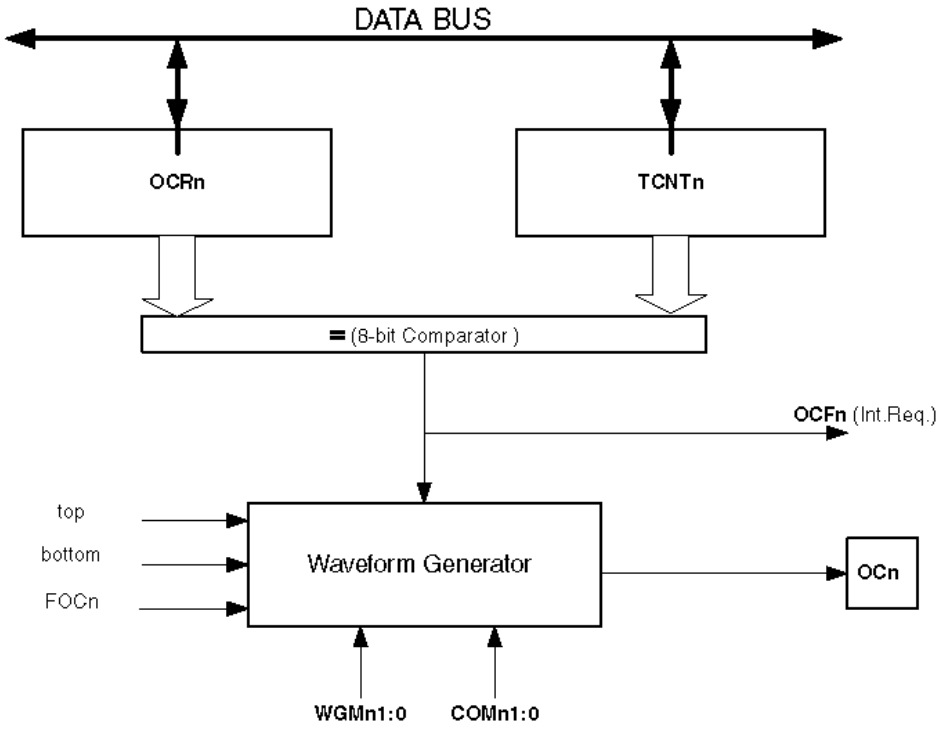


Рисунок 124. Схема порівняння 8-бітного таймера-лічильника n (n = 0 або 2)

При використанні будь-якого режиму ШІМ запис у регістр OCRn буферизований. Для нормального режиму і режиму зі скиданням за рівністю буферизація блокована. Подвійна буферизація синхронізує модифікацію OCRn із досягненням лічильником мінімального чи максимального значення.

Синхронізація запобігає утворенню несиметричних імпульсів на виході ШІМ. Усі операції записування в TCNTn блокують будь-яке порівняння на один цикл, навіть якщо таймер зупинений.

У режимах, не пов’язаних із ШІМ, вихід компаратора може примусово переводитися у стан «1» за допомогою біта FOCn у регістрі TCCRn. Встановлення біта не встановлює ознаку OCFn та не скидає таймер, але стан виведення OC0 буде оновлено як при рівності значень.

Вихід 8-бітного таймера.

Спрощена схемотехніка виходу таймерів-лічильників 0 та 2 однакова і наведена на рис. 125.

Якщо будь-який із бітів COMn1:0 (n = 0 або 2) встановлений, то основну функцію порту вводу-виводу перекриває біт порівняння OCn.

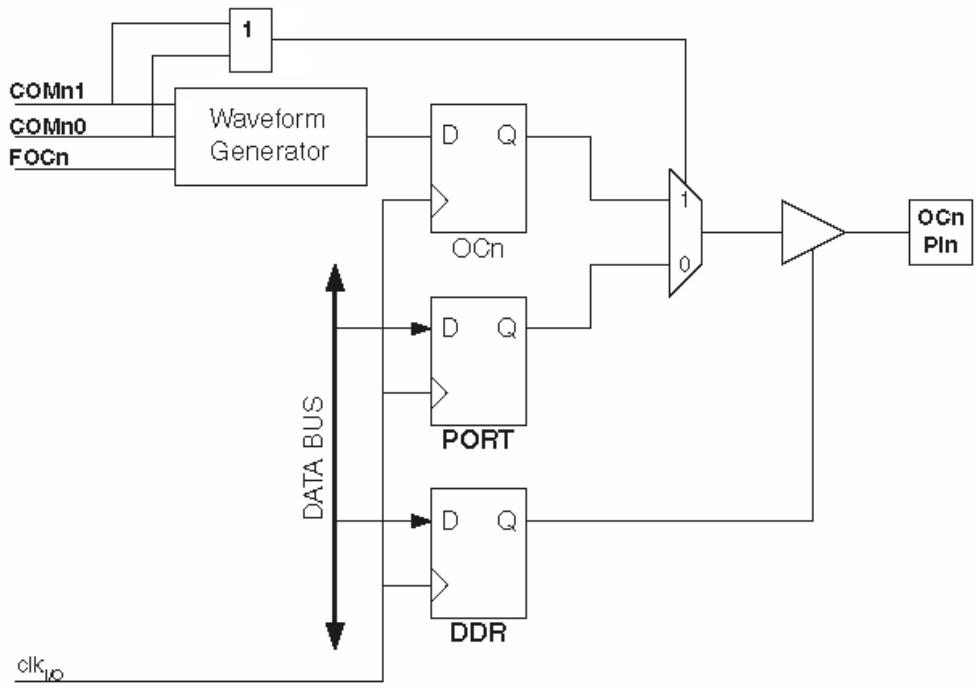


Рисунок 125. Спрощена схема виходу таймера 0

Таблиця 41. Режими роботи лінії OC0 залежно від стану бітів регістра TCCRn у режимах, не пов’язаних із ШІМ

COMn1	COMn0	Опис роботи лінії OCn
0	0	Нормальний режим роботи порту, біт OCn від’єднаний від порту
0	1	Лінія OCn перемикається при кожному співпаданні
1	0	Лінія OCn скидається при кожному співпаданні
1	1	Лінія OCn встановлюється при кожному співпаданні

Таблиця 42. Режими роботи лінії OC0 залежно від стану бітів регістра TCCRn у режимі швидкої ШІМ

COMn1	COMn0	Опис роботи лінії OCn
0	0	Нормальний режим роботи порту, біт OCn від’єднаний від порту
0	1	Зарезервовано
1	0	Біт OCn скидається при рівності вмісту таймера (TCNTn) та регістра OCRn, встановлюється при переході лічильника TCNTn до нульового значення (неінвертуючий режим)
1	1	Біт OCn встановлюється при рівності вмісту таймера (TCNTn) та регістра OCRn, скидається при переході лічильника TCNTn до нульового значення (інвертуючий режим)

Таблиця 43. Режими роботи лінії OC0 залежно від стану бітів регістра TCCRn у режимі з коректною фазою ШІМ

COMn1	COMn0	Опис роботи лінії OCn
0	0	Нормальний режим роботи порту, біт OCn від’єднаний від порту
0	1	Зарезервовано
1	0	Біт OCn скидається при рівності вмісту таймера (TCNTn) та регістра OCRn при зростанні лічильника, встановлюється при рівності вмісту таймера (TCNTn) та регістра OCRn при зростанні лічильника (неінвертуючий режим)
1	1	Біт OCn встановлюється при рівності вмісту таймера (TCNTn) та регістра OCRn при уростанні лічильника, скидається при рівності вміста таймера (TCNTn) та регістра OCRn при зростанні лічильника (неінвертуючий режим)

Проте регістр напрямку DDR все одно керує напрямком передавання даних лінії порту. Тому для виведення стану порівняння біт у регістрі DDR, що відповідає OCn, має бути переведений у режим виведення незалежно від того, котрий режим роботи таймера-лічильника використовується.

Режим роботи виходу залежить від стану бітів COMn1 COMn0 регістра TCCRn (рис. 116) й обраного режиму роботи. Для всіх режимів роботи режими перемикання виводу наведені у наступних трьох таблицях.

Асинхронний режим роботи таймера-лічильника 2.

Таймер-лічильник 2 може працювати асинхронно, тобто із власним сигналом синхронізації, і не залежити від наявності основного тактового сигналу.

Генератор оптимізовано для використання з годинниковим кварцом на 32 768 Гц. Подача зовнішнього тактового сигналу на лінію TOSC1 може призвести до неправильної роботи таймера-лічильника 1. Частота основного тактового сигналу повинна бути більшою, ніж в чотири рази за частоту генератора.

При роботі в асинхронному режимі роботи під час записування в один з регістрів TCNT2, OCR2 чи TCCR2 значення передається у тимчасовий регістр, окремий для кожного регістра, і фіксується через два передніх фронти сигналу з лінії TOSC1. А тому запис TCNT2 не заважає записуванню, наприклад, в OCR2. З іншого боку, запис нового значення не має перекривати процес записування попереднього значення. Готовність регістрів до прийому значень може бути перевірена за допомогою регістра ASSR.

При використанні енергозберігаючих режимів і використанні можливості «пробудження» від переривання від таймера 2, що працює асинхронно, слід віддавати команду переходу в енергозберігаючий режим лише за наявності двох умов:

1. Таймер-лічильник 2 не має встановлених бітів зайнятості у регістрі ASSR.

2. З моменту надходження попереднього запиту переривання від таймера-лічильника 2 надійшов хоча б один тактовий імпульс із лінії TOSC1.

Якщо програма «не знає» необхідного часу до надходження наступного імпульсу на лінії TOSC1, можна скористатися таким алгоритмом:

1. Записати значення у регістр TCCR2, TCNT2 або OCR2.

2. Зачекати, доки скинеться відповідний біт очикування в регістрі ASSR.

3. Перейти у визначений режим зниженого енергоспоживання.

При роботі в асинхронному режимі генератор таймера-лічильника 2 працює завжди, за винятком режимів Power-down та Standby, після ввімкнення живлення або виходу із режимів від'єднання живлення і режиму очікування. Після ввімкнення живлення або виходу із режимів Power-down та Standby необхідно до 1 секунди для стабілізації частоти опорного тактового сигналу. Вміст усіх регістрів таймера-лічильника 2 при цьому можна вважати втраченим.

Читання регістра TCNT2 відразу після пробудження з енергозберігаючих режимів може дати неправильний результат. Щоб уникнути цього, необхідно дочекатися надходження ще одного переднього фронту на лінії TOSC1 або записати у регістр OCR2 чи TCCR2 та дочекатися скидання відповідної ознаки зайнятості.

У процесі перемикання між асинхронним і синхронним режимами роботи вміст регістрів TCNT2, OCR2 і TCCR2 може бути пошкоджений. Безпечна процедура для перемикання джерела тактового сигналу є такою:

1. Вимкнути переривання від таймера, скинувши біти OCIE2 і TOIE2.

2. Обрати джерело синхронізації, задавши значення AS2.

3. Записати нові значення у TCNT2, OCR2 і TCCR2.

4. В асинхронному режимі роботи дочекатися скидання бітів зайнятості в регістрі ASSR.

5. Скинути ознаки переривань від таймера-лічильника 2.

6. Дозволити переривання від таймера, якщо це необхідно.

3.10.2.2. Регістри, пов’язані з таймером-лічильником 0.

Керуючий регістр таймера-лічильника 0 TCCR0 (рис. 126) призначений для задавання режиму роботи мікросхеми. Біти регістра описані вище.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 126. Керуючий регістр таймера-лічильника 0 TCCR0

Регістр відліку TCNT0 (рис. 127) дає прямий доступ як для читання, так і для записування 8-розрядного значення таймера-лічильника 0. Записування в регістр TCNT0 блокує порівняння на наступний період синхронізації.

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 127. Регістр відліку TCNT0

Регістр вихідного компаратора OCR0 (рис. 128) містить 8-разрядне значення, з котрим порівнюється вихідне значення таймера, та дає можливість формувати значення на лінії OC0.

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 128. Регістр вихідного компаратора OCR0

3.10.2.3. Регістри, пов’язані з таймером-лічильником 2

Керуючий регістр таймера-лічильника 2 TCCR2 (рис. 129) призначений для задавання режиму роботи мікросхеми. Біти регістра описані вище. При читанні TCCR2 читається значення в регістрі тимчасового зберігання.

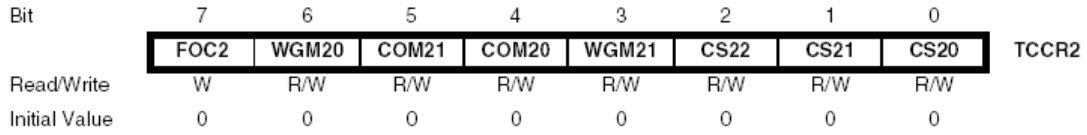


Рисунок 129. Керуючий регістр таймера-лічильника 2 TCCR2

Регістр відліку TCNT2 (рис. 130) дає прямий доступ як для читання, так і для записування 8-розрядного значення таймера-лічильника 2. Записування в регістр TCNT2 блокує порівняння на наступний період синхронізації. При читанні TCNT2 читається фактичне значення таймера незалежно від режиму роботи.

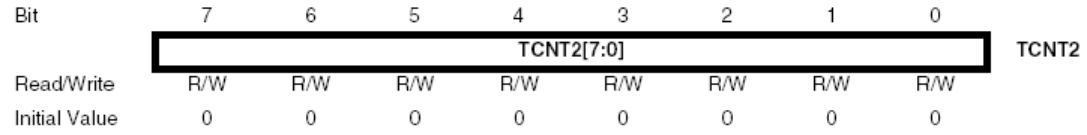


Рисунок 130. Регістр відліку TCNT2

Регістр вихідного компаратора OCR2 (рис. 131) містить 8-разрядне значення, з котрим порівнюється вихідне значення таймера й стає можливим формувати значення на лінії OC2. При читанні OCR2 читається значення в регістрі тимчасового зберігання.

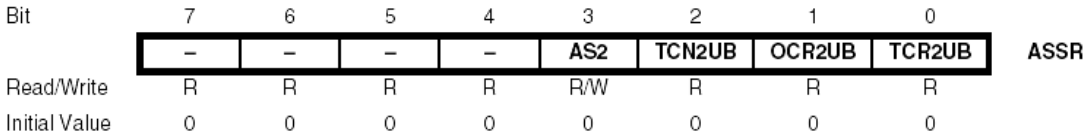


Рисунок 131. Регістр вихідного компаратора OCR2

Регістр ASSR (рис. 132) використовується для задавання та контролю за роботою таймера-лічильника 2 в асинхронному режимі роботи.

Біт AS2 використовується для визначення джерела синхронізації. Нульове значення вказує на синхронізацію від основного тактового сигналу, одиничне –

на синхронізацію від лінії TOSC1 та приєднаного до неї генератора. При зміні значення AS2 вміст регістрів TCNT2, OCR2 і TCCR2 може бути пошкоджений.

Bit	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 132. Регістр асинхронних операцій

Біт TCN2UB вказує на зайнятість таймера оновленням регістра TCNT2. Біт автоматично встановлюється в одиницю при записування в регістр TCNT2 при роботі в асинхронному режимі та автоматично скидається в нуль, коли оновлення регістра TCNT2 завершене.

Біт OCR2UB вказує на зайнятість таймера оновленням регістра OCR2. Біт автоматично встановлюється в одиницю при записування в регістр OCR2 при роботі в асинхронному режимі та автоматично скидається в нуль, коли оновлення регістра OCR2 завершене.

Біт TCR2UB вказує на зайнятість таймера оновленням регістра TCCR2. Біт автоматично встановлюється в одиницю при записування в регістр TCCR2 при роботі в асинхронному режимі та автоматично скидається в нуль, коли оновлення регістра TCCR2 завершене.

3.10.2.4. Біти загальних регістрів, що використовуються таймерами 0 та 2.

Регістр маски переривань TIMSK (рис. 133) є загальним для усіх таймерів. Для таймера 0 використовуються біти OCIE0 та TOIE0, а для таймера 2 – біти OCIE2 та TOIE2.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 133. Регістр маски переривань TIMSK

Для налаштування лічильника, крім регістрів самого лічильника, використовуються регістри, що є загальними для всіх таймерів.

Біт OCIE0, встановлений в «1», дозволяє переривання від блока порівняння таймера-лічильника 0 (біт OCF0).

Біт TOIE0, встановлений в «1», дозволяє переривання при переповненні таймера-лічильника 0 (біт TOV0).

Біт OCIE2, встановлений в «1», дозволяє переривання від блока порівняння таймера-лічильника 2 (біт OCF2).

Біт TOIE2, встановлений в «1», дозволяє переривання при переповненні таймера-лічильника 2 (біт TOV2).

У регістрі переривань таймерів TIFR (рис. 134) відображається стан переривань від усіх таймерів мікросхеми. До таймера-лічильника 0 відносяться біти OCF0 та TOV0, а до таймера-лічильника 2 – біти OCF2 та TOV2.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 134. Регістр переривань TIFR

Біти OCF0 та OCF2 відображають стан запиту переривань від блока порівняння таймера 0 чи 2, а біти TOV0 та TOV2 – переривання при переповненні таймерів 0 чи 2. Біти скидаються автоматично при виклику відповідного переривання, крім того, вони можуть скидатися і програмним шляхом.

3.10.3. 16-розрядний таймер-лічильник 1.

Таймер-лічильник T/C1 – це універсальний шістнадцятибітний лічильник з двома модулями порівняння і одним модулем захоплення, а також підтримкою функції широтно-імпульсної модуляції ШІМ (PWM) зі змінною частотою модуляції (рис. 135).

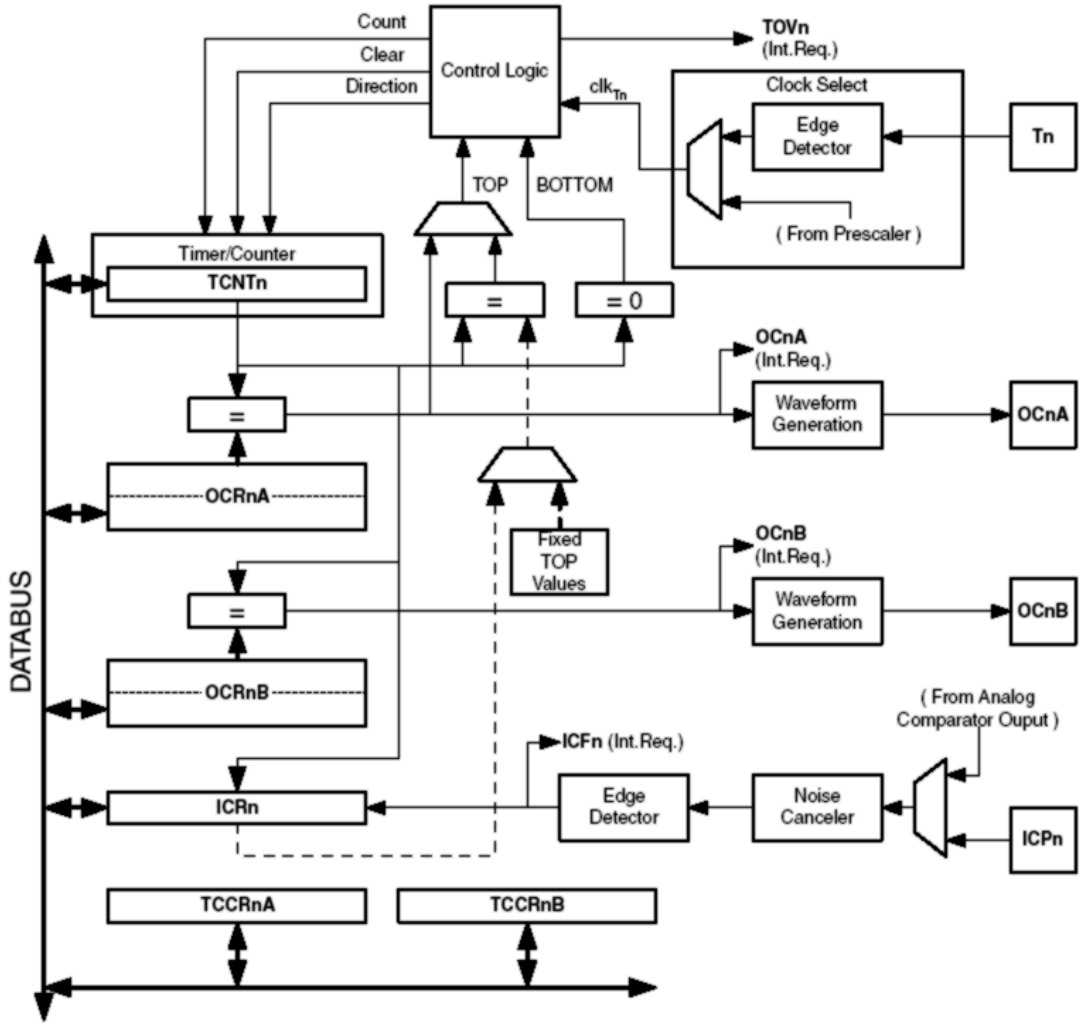


Рисунок 135. Спрощена схема таймера-лічильника 1

Він дозволяє формувати задані проміжки часу для роботи в режимі реального часу, а також може служити генератором сигналів. Вхідним сигналом пристрою може бути як зовнішній сигнал з виводу мікросхеми, так і сигнал із тактового генератора, котрий може бути пропущений через попередній подільник.

Основні особливості таймера:

◊ наявність двох модулів порівняння, що можуть формувати два сигнали на відповідні виходи та два запити переривань від них;

◊ можливість подвійної буферизації при записуванні в регістри порівняння;

◊ скидання таймера при рівності значення заданому;

◊ наявність симетричного широтно-імпульсного модулятора з коректною фазою та частотою;

◊ змінний період у режимі ШІМ;

◊ чотири незалежні джерела переривання (TOV1, OC1A, OC1B, ICF0);

◊ наявність модуля захоплення.

Імпульси, які підраховуються, надходять на блок керування від блока вибору сигналу синхронізації (Clock Select), котрий визначає джерело тактового сигналу та режим роботи таймера чи лічильника. Таймер-лічильник може працювати як від внутрішнього тактового генератора через попередній дільник, так і від зовнішнього тактового сигналу, що надходить на вхід T1. Схема вибору джерела тактового сигналу передає тактові імпульси вибраного джерела на вхід таймера-лічильника і кожен імпульс цього сигналу збільшує (або зменшує) значення регістра TCNT1.

Якщо не вибране жодне з джерел тактового сигналу, таймер-лічильник зупиняється. Джерело синхронізації вибирається бітами CS1 2-0, розташованими у регістрі керування таймера-лічильника TCCR1.

Код, до котрого дорахував таймер-лічильник 1, надходить на два цифрові компаратори, де порівнюються з умістом регістрів OCR1A та OCR1B. При рівності значень сигнал надходить на запит переривань OC1A, OC1B та на відповідні формувачі вихідних сигналів (Waveform Generation). Поточне значення таймера-лічильника може бути зафіксоване за зовнішнім сигналом, що здійснюється модулем захоплення (Input Capture).

Регістр відліку таймера-лічильника (TCNT1), регістри порівняння OCR1A та OCR1A, а також регістр захоплення ICR1 є 16-бітними. Максимальне значення відліку, до якого може рахувати таймер-лічильник 1, може бути фіксованим і рівним 0FF, 1FF, 3FF, 0FFFF, або зберігатись у регістрі OCR1A чи ICR1.

Кожен із запитів переривань (позначених як Int.Req. на рис. 135) індивідуально маскується у регістрі маски таймерів TIMSK і може бути перевірений у регістрі запитів переривань таймерів TIFR незалежно від наявності маскування. Регістри TIFR і TIMSK не показані на рис. 135, тому що вони є спільними для усіх таймерів мікросхеми.

Вхід лічильника-таймера 1.

У якості вхідного сигналу таймера-лічильника 1 використовується або сигнал із входу T1, або сигнал із попереднього подільника, або тактовий сигнал синхронізації пристроїв вводу-виводу. Таймер-лічильник може отримувати імпульси із входу T1 і тоді, коли відповідна лінія вводу-виводу конфігурована як вихід.

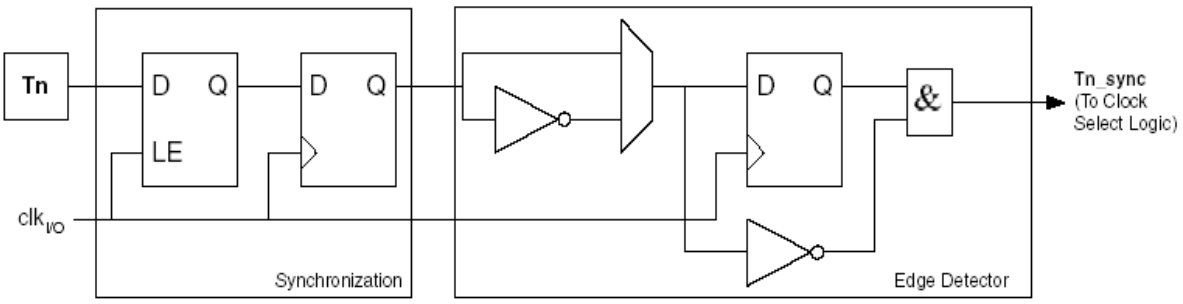


Рисунок 136. Вхідне коло лінії T1

Фронт сигналу T1, за яким здійснюється перемикання таймера-лічильника, вибирається програмним шляхом. Схема формування сигналу на вході лічильника наведена на рис. 136. Схема складається із двох основних блоків: синхронізатора Synchronizator та детектора фронту Edge Detector. При високому рівні на лінії $clk_{I/O}$ сигнал з лінії пропускається через вхідний тригер і надходить на другий тригер синхронізатора, де фіксується за переднім фронтом наступного імпульсу сигналу $clk_{I/O}$. Така схемотехніка унеможливлює появу на виході синхронізатора імпульсів малої тривалості. Сигнал із синхронізатора надходить на детектор фронту, котрий формує імпульси тривалістю один період сигналу $clk_{I/O}$ за фронтом сигналу з виходу синхронізатора. Вибір фронту, за яким здійснюється формування імпульсу, залежить від стану бітів CS12-0.

Синхронізація та логіка детектора фронту створюють затримку від 2,5 до 3,5 тактових циклів від фронту імпульсу до моменту перемикання лічильника. Зі схеми також випливає, що перемикання вхідного тактового сигналу має бути виконане тоді, коли лінія T0 буде стабільною протягом, принаймні, одного періоду сигналу $fclk_{I/O}$. З іншого боку, кожна половина періоду зовнішнього

сигналу повинна бути більшою, ніж один такт системного тактового сигналу, для забезпечення правильного відбору фронтів. Тому для надійного введення тактового сигналу рекомендується, щоб максимальна частота зовнішнього джерела синхронізації не перевищувала $fclk_{I/O}/2{,}5$.

Сигнал із наведеного вище вхідного кола надходить на схему вибору джерела синхронізації, схема котрого наведена на рис. 137.

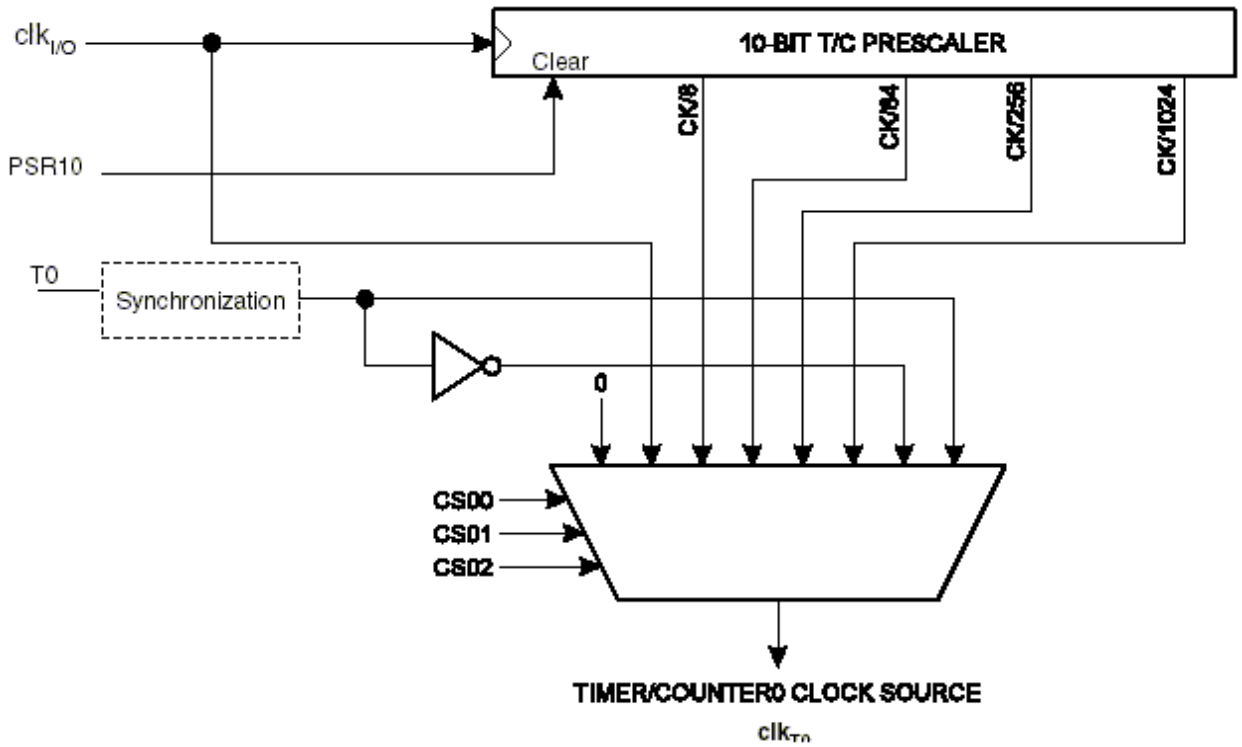


Рисунок 137. Схема вибору джерела синхронізації лічильника 1

Вибір джерела синхронізації здійснюється бітами CS2-0 у регістрі TCCR1B (рис. 138). Біти мають наступне призначення (таблиця 44).

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 138. Формат регістра TCCR1B

Таблиця 44. Джерела тактового сигналу для таймера-лічильника 1

CS12	CS11	CS10	Джерело тактового сигналу
0	0	0	Немає сигналу. Таймер-лічильник зупинений
0	0	1	$clk_{I/O}$ (без попереднього подільника)
0	1	0	Частота з попереднього подільника $clk_{I/O}/8$
0	1	1	Частота з попереднього подільника $clk_{I/O}/64$
1	0	0	Частота з попереднього подільника $clk_{I/O}/256$
1	0	1	Частота з попереднього подільника $clk_{I/O}/1024$
1	1	0	Зовнішнє джерело (лінія T1). Перемикання лічильника за спадом на лінії
1	1	1	Зовнішнє джерело (лінія T1). Перемикання лічильника за переднім фронтом на лінії

Блок захоплення.

Таймер-лічильник 1 включає блок захоплення, що дозволяє легко визначати час появи зовнішньої події. Поточне значення лічильника може бути зафіксоване у регістрі модуля захоплення. Для фіксації значення використовується стробуючий сигнал, що може надходити від аналогового компаратора (Analog Comparator) чи від лінії ICP1.

Мітки часу, що генеруються блоком, можуть використовуватися для обчислення періоду та параметрів сигналу або для створення журналу подій. Будова блока захоплення наведена на рис. 139. Елементи схеми, які не є частиною блока, наведені суцільною лінією.

Вибір джерела стробуючого сигналу для керування блоком визначається бітом ACIC у регістрі ACSR. При записуванні логічної одиниці цей біт вмикає функцію захоплення таймера-лічильника 1 від аналогового компаратора. При записуванні логічного нуля захоплення здійснюється за станом лінії ICP1.

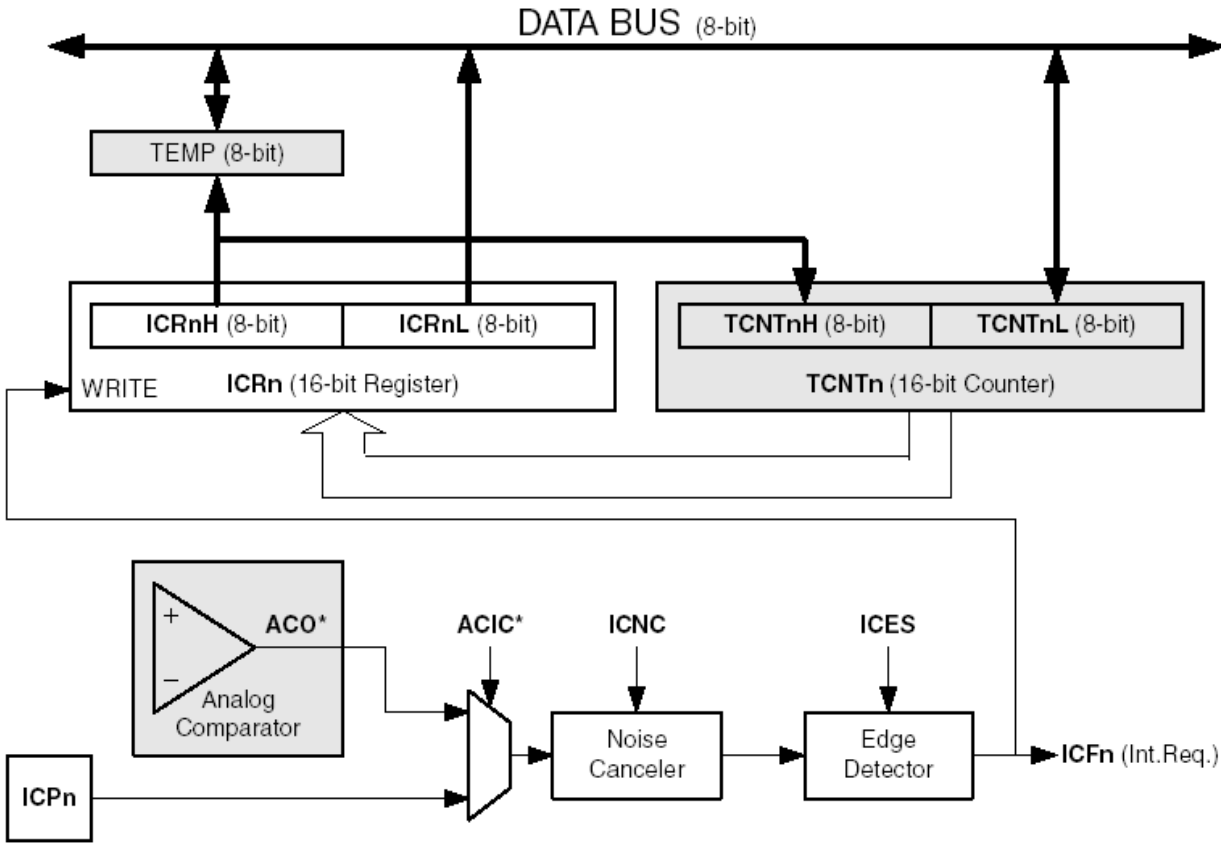


Рисунок 139. Структурна схема блока захоплення

Вихід компаратора чи лінія ICP1 під'єднуються до цифрового фільтра (Noise Canceler) та детектора перепаду (Edge Detector), та керують роботою регістра ICR1, що фіксує поточне значення лічильника TCNT1.

Детектор фронту ідентичний такому ж детектору для входу T1. Цифровий фільтр призначений для усунення багатократних перемикань входу та підвищує перешкодозахищеність за допомогою простої схеми цифрової фільтрації: вхід контролюється протягом чотирьох тактів і якщо у всіх чотирьох тактах сигнал є однаковим, вихід перемикається у новий стан. Цифровий фільтр вмикається установкою біта ICNC1 у регістрі керування TCCR1B. Увімкнений цифровий фільтр вводить затримку в чотири додаткові такти перед оновленням ICR1.

Читання регістра ICR1 програмним шляхом можливе у будь-якому режимі роботи. Записування у регістр ICR1 можливе лише при використанні режиму генерації сигналу, який, у свою чергу, використовує регістр ICR1 для визначення максимального значення лічильника.

У момент фіксації значення в регістрі ICR1 створюється запит переривання ICF1. Якщо ж встановлено біт TICIE1 = 1 у регістрі TIMSK, ознака захоплення входу формує переривання вхідного захоплення, а біт ICF1 при цьому автоматично очищається при переході до обслуговування переривання. Крім того, біт ICF1 може бути скинутий програмно за допомогою команд вводу-виводу.

Блоки порівняння.

Мікросхема має два блоки порівняння, названі Output Compare Unit A та Output Compare Unit B. Блок порівняння A також дозволяє встановити максимальне значення відліку лічильника у деяких режимах роботи. Будова кожного із блоків порівняння наведена на рис. 140, де n позначає номер лічильника, а x – блок A чи B.

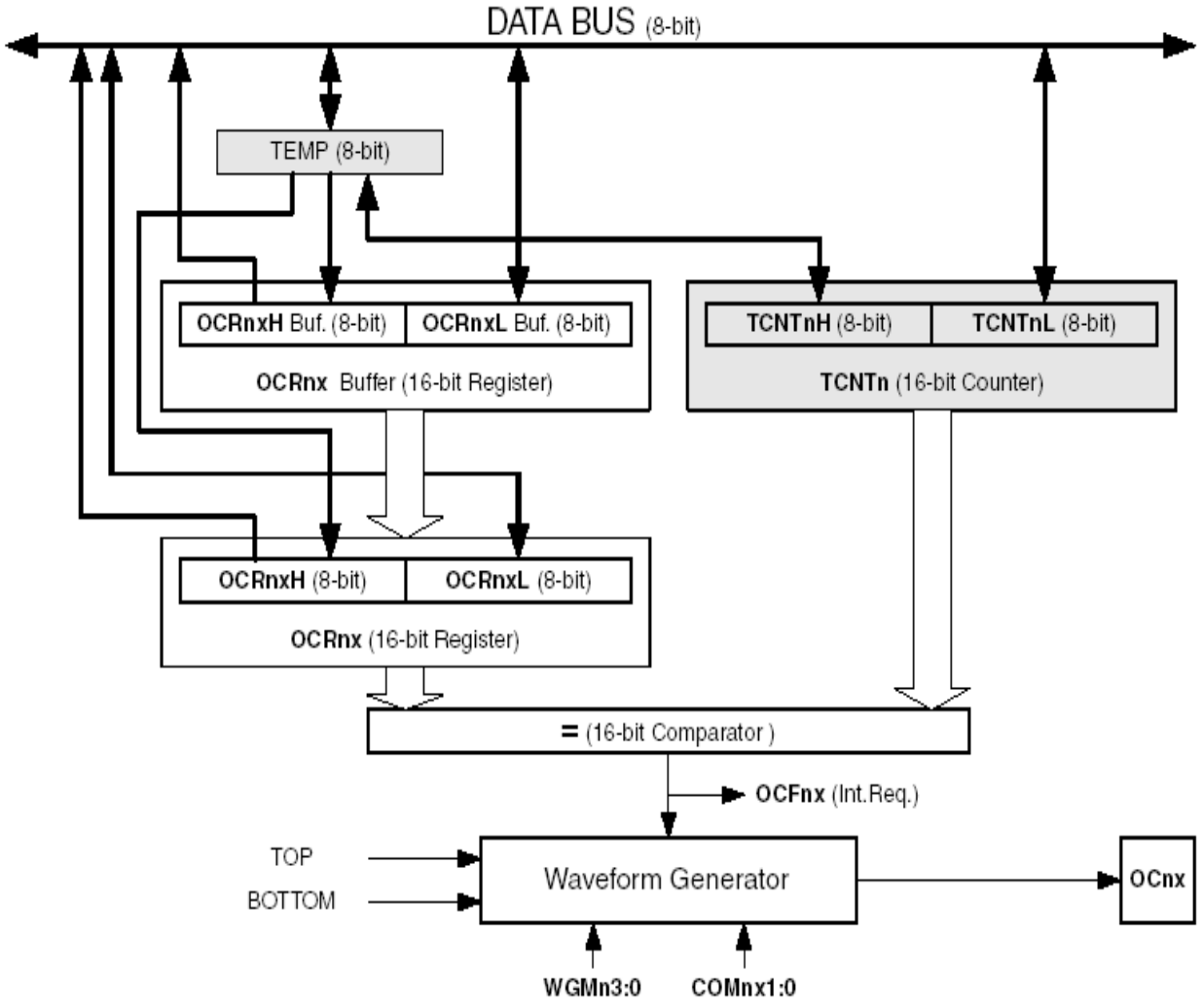


Рисунок 140. Блок порівняння таймера-лічильника 1

Елементи схеми, що не є безпосередньо частиною блока порівняння, виділені суцільною лінією.

Кожен 16-бітний компаратор постійно порівнює вміст лічильника TCNT1 з виходом регістра компаратора OCR1x (OCR1A чи OCR1B). Якщо значення TCNT та OCR1x співпадають, видається сигнал рівності, що встановлює ознаку OCF1A чи OCF1B у наступному такті таймера. У випадку, коли дозволено відповідне переривання від блока порівняння OCIE1x = 1 (OCIE1A = 1 або OCIE1A = 1), вихід компаратора генерує запит переривання. Ознака OCF1x автоматично очищається при старті переривання. Крім того, біти OCF1x можуть бути скинуті програмно.

Сигнал із блока порівняння надходить на генератор вихідного сигналу (Wareform Generator) і використовується залежно від режиму роботи (бітів WGM13:0) і бітів режиму виводу COM1x1:0.

У випадку необхідності, у режимах без ШІМ на виході компаратора може бути примусово отриманий сигнал рівності, але при цьому не буде викликане переривання, не буде встановлена ознака рівності та не буде скинутий лічильник. Примусове встановлення біта компаратора здійснюється записом одиниці у біти FOC1A та FOC1B. Біти примусово скидаються у наступному машинному такті, тобто при наступному читанні біти будуть прочитані як «0».

3.10.3.1. Режими роботи таймера 1.

Таймер-лічильник 1 може використовуватися у таких режимах роботи:

◊ нормальному (Normal);

◊ зі скиданням за рівністю (CTC);

◊ швидкої широтно-імпульсної модуляції (Fast PWM);

◊ широтно-імпульсної модуляції з коректною фазою Phase Correct PWM;

◊ широтно-імпульсної модуляції із коректною фазою та частотою Phase and Frequency Correct PWM.

Встановлення режиму роботи таймера-лічильника 1 здійснюється за допомогою бітів WGM13:0 у регістрах TCCR1A (рис. 141) та TCCR1B (рис. 142), як показано у таблиці 45.

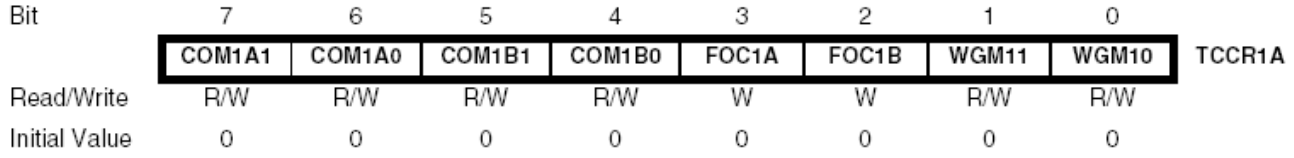


Рисунок 141. Формат регістра TCCR1A

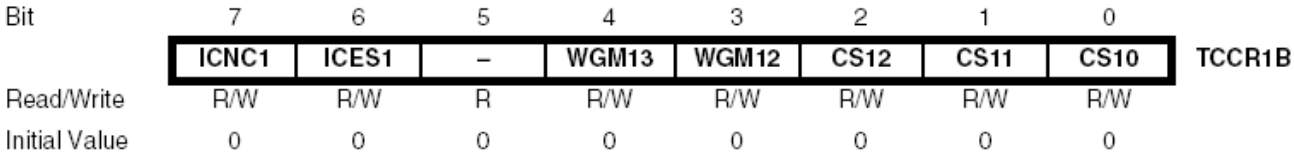


Рисунок 142. Формат регістра TCCR1B

Таблиця 45. Режими таймера-лічильника 1

WGM13-WGM10				Режим роботи таймера-лічильника	Максимальне значення	Завантаження нового значення у регістр OCR1x	Умова встановлення біта TOV1
13	12	11	10				
1	2	3	4	5	6	7	8
0	0	0	0	Нормальний	0xFFFF	Відразу при перезавантаженні регістра OCR1x	При досягненні лічильником значення 0x0FFFF
0	0	0	1	Широтно-імпульсної модуляції з коректною фазою Phase Correct PWM, 8-біт	0x00FF	При досягненні лічильником Значення 0x00FF	При досягненні лічильником значення 0x0000
0	0	1	0	Широтно-імпульсної модуляції з коректною фазою Phase Correct PWM, 9-біт	0x01FF	При досягненні лічильником Значення 0x01FF	При досягненні лічильником значення 0x0000

Продовження таблиці 45

1	2	3	4	5	6	7	8
0	0	1	1	Широтно-імпульсної модуляції з коректною фазою Phase Correct PWM, 10-біт	0x03FF	При досягненні лічильником Значення 0x03FF	При досягненні лічильником значення 0x0000
0	1	0	0	Зі скиданням за рівністю (CTC)	OCR1A	Відразу при перезавантаженні регістра OCR1x	При досягненні лічильником значення 0x0FFF
0	1	0	1	Швидкої широтно-імпульсної модуляції (Fast PWM), 8-біт	0x00FF	При досягненні лічильником Значення 0x0000	При досягненні лічильником значення 0x00FF
0	1	1	0	Швидкої широтно-імпульсної модуляції (Fast PWM), 9-біт	0x01FF	При досягненні лічильником Значення 0x0000	При досягненні лічильником значення 0x01FF
0	1	1	1	Швидкої широтно-імпульсної модуляції (Fast PWM), 10-біт	0x03FF	При досягненні лічильником Значення 0x0000	При досягненні лічильником значення 0x03FF
1	0	0	0	Широтно-імпульсної модуляції з коректною фазою та частотою Phase and Frequency Correct PWM	Значення із ICR1	При досягненні лічильником Значення 0x0000	При досягненні лічильником значення 0x0000

Закінчення таблиці 45

1	2	3	4	5	6	7	8
1	0	0	1	Широтно-імпульсної модуляції з коректною фазою та частотою Phase and Frequency Correct PWM	Значення із OCR1A	При досягненні лічильником значення 0x0000	При досягненні лічильником значення 0x0000
1	0	1	0	Широтно-імпульсної модуляції з коректною фазою Phase Correct PWM	Значення із ICR1	При досягненні лічильником значення ICR1	При досягненні лічильником значення 0x0000
1	0	1	1	Широтно-імпульсної модуляції з коректною фазою Phase Correct PWM	Значення із OCR1A	При досягненні лічильником значення OCR1A	При досягненні лічильником значення 0x0000
1	1	0	0	Зі скиданням за рівністю (CTC)	Значення із ICR1	Відразу при перезавантаженні регістра ICR1x	При досягненні лічильником значення 0x0FFFF
1	1	0	1	Зарезервовано	–	–	
1	1	1	0	Швидкої широтно-імпульсної модуляції (Fast PWM)	Значення із ICR1	При досягненні лічильником Значення 0x0000	При досягненні лічильником значення ICR1
1	1	1	1	Швидкої широтно-імпульсної модуляції (Fast PWM)	Значення із OCR1A	При досягненні лічильником Значення 0x0000	При досягненні лічильником значення OCR1A

Нормальний режим.

Нормальний режим (Normal) – найпростіший. У ньому лічильник рахує від заданого значення до значення 0FFFF, потім скидається в нуль і продовжує відлік далі. При переході зі стану 0FFFF у стан 0000 встановлюється в 1 біт TOV0. У поєднанні з перериванням за переповненням таймера, виклик якого автоматично очищає біт TOV0, цей режим може використовуватися для підрахунку часових інтервалів чи кількості подій тощо. У даному режимі роботи блок цифрового компаратора може використовуватися довільним чином.

Режим зі скиданням за рівністю.

У режимі зі скиданням за рівністю (CTC – Clear Timer on Compare Match) регістр компаратора OCR1A або регістр захоплення ICR1 використовують для збереження максимального значення, до якого може рахувати лічильник. У режимі CTC лічильник скидається в нуль, коли значення лічильника TCNT0 стає рівним OCR1A чи ICR1. У цьому режимі роботи ознака переповнення таймера-лічильника TOV0 не встановлюється до тих пір, поки таймер-лічильник не перейде зі стану 0FFFF у стан 0, що може відбутися. У даному режимі роботи блок цифрового компаратора використовується для формування сигналу із заданою частотою, що надходить на вихід мікросхеми, якщо значення у лічильнику перевищує значення обраного регістра.

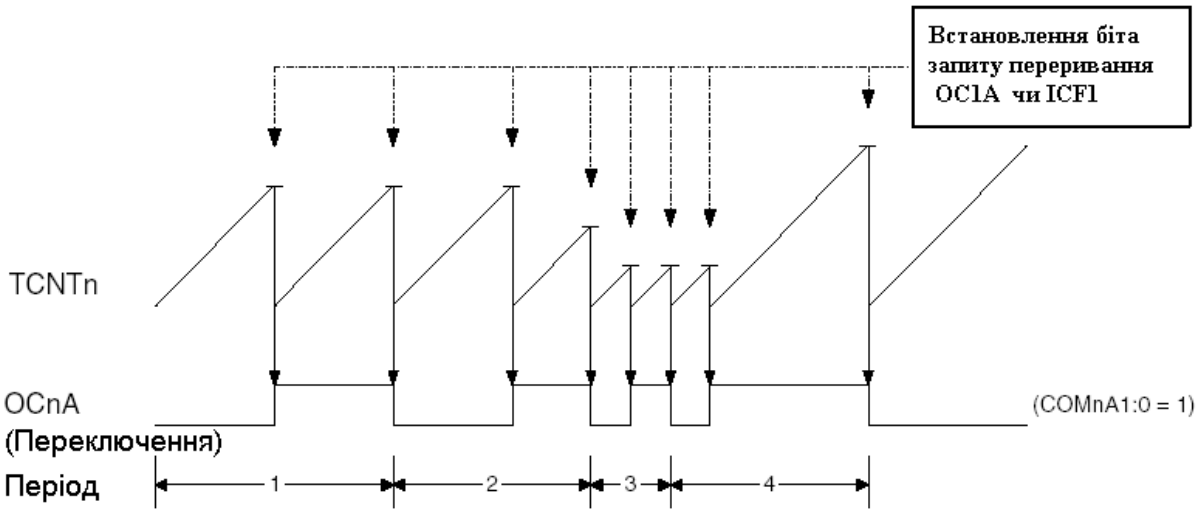


Рисунок 143. Режим зі скиданням за рівністю

Часова діаграма роботи лічильника у цьому режимі роботи зображена на рис. 143. Для генерації сигналу на виході у CTC-режимі вихід OC1A може бути встановлений для перемикання логічного рівня на кожному порівнянні при встановленні бітів режим виводення (COM1A1:0 = 1). Проте значення OC1A не буде видно на виводі мікросхеми, поки вивід не налаштований на вхід (DDR_OC1A = 1). Максимальна частота сигналу на виході OC1 дорівнює $f_{OC1} = f_{clk\_I/O}/2$ і досягається, якщо регістр OCR1A чи регістр ICR1 встановлюється в нуль (0x0000). В інших випадках частота сигналу на виході визначається за формулою

$$f_{OCn} = \frac{f_{clk\_I/O}}{2\,N\,(1 + OCRn)}$$

де N є значенням коефіцієнта поділу попереднього подільника.

Режим *швидкої широтно-імпульсної модуляції.*

У режимі швидкої широтно-імпульсної модуляції (Fast PWM) лічильник рахує від 0 до максимального значення, яке залежить від режиму роботи. Якщо значення у лічильнику перевищує задане в регістрі OCR1, здійснюється перекмикання відповідного виходу. Вихід повертається у попередній стан при скиданні лічильника. При кожному скиданні лічильника відбувається встановлення ознаки TOV0, а при досягненні значення OCR1A – ознаки OC1. Центр імпульсу на виході при цьому «плаває» відносно початку відліку.

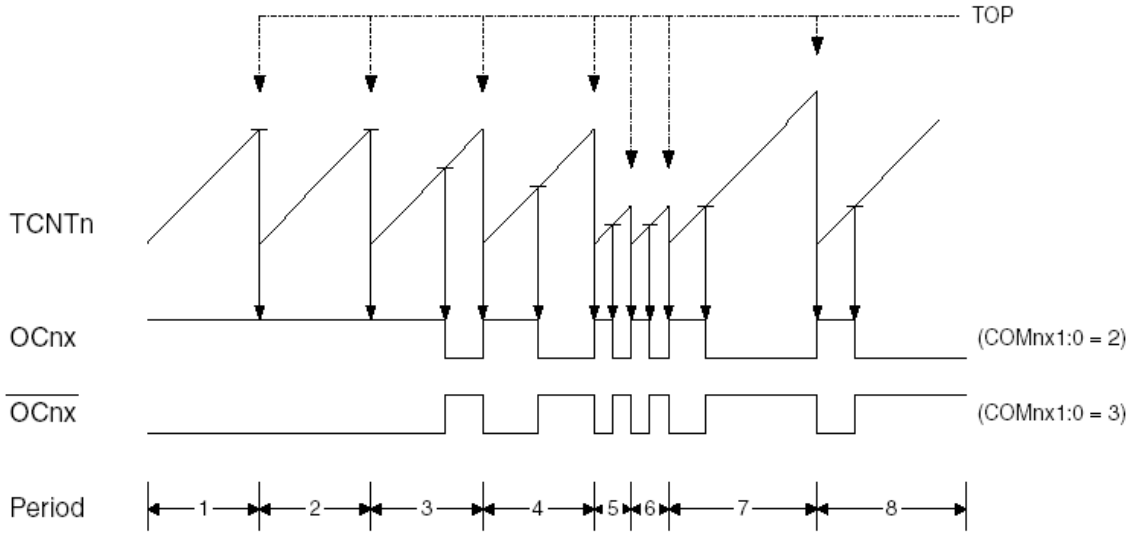


Рисунок 144. Режим швидкої широтно-імпульсної модуляції

У режимі швидкої широтно-імпульсної модуляції лічильник збільшується або до одного із фіксованих значень (0x00FF, 0x01FF, 0x03FF, 0xFFFF), або до значення, записаного у регістрі ICR1 чи регістрі OCR1A. Часова діаграма для режиму швидкої ШІМ показана на рис. 144, де зображено режим із використанням максимального значення з регістра OCR1A чи ICR1 (позначеного TOP). Значення TCNT1 на діаграмі показане у вигляді графіка. На часовій діаграм зображено неінвертований та інвертований режими ШІМ.

Значення частоти вихідного сигналу визначається за допомогою формули

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{2\,N\,(1+TOP)}$$

де N – значенням коефіцієнта поділу попереднього подільника, а TOP – значення, до якого рахує лічильник, що може бути як фіксованим, так і залежати від умісту регістра OCR1A чи регістра ICR1.

Режим широтно-імпульсної модуляції з коректною фазою.

У режимі широтно-імпульсної модуляції з коректною фазою (PWM, Phase Correct) лічильник рахує від 0 до максимального значення, а потім у звортний бік – від максимального значення до 0. Якщо значення в лічильнику перевищує задане в регістрі OCR1A, то здійснюється перемикання відповідного виходу. Коли в процесі зворотного відліку значення в лічильнику стане меншим, ніж значення, задане у OCR1A, то вихід повернеться у початковий стан. У даному режимі роботи положення центру імпульса на виході відносно початку відліку не залежить від вмісту регістра OCR1A. Частота зміни сигналу на виході у цьому режимі роботи є меншою, ніж у режимі швидкої широтно-імпульсної модуляції, проте, завдяки симетрії, цей метод має свої переваги. Часова діаграма для режиму широтно-імпульсної модуляції з коректною фазою зображена на рис. 145. Значення TCNT1 показане у вигляді графіка. На діаграмі також наведені неінвертований та інвертований виходи ШІМ.

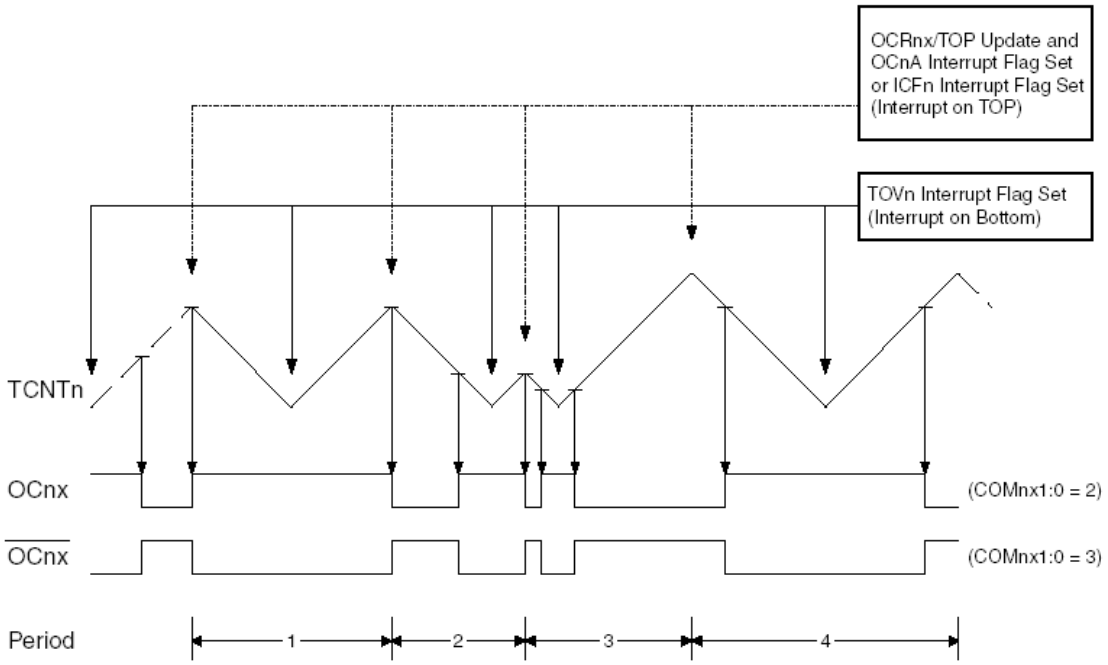


Рисунок 145. Режим широтно-імпульсної модуляції з коректною фазою

Коли біти OCR1A чи ICR1 використовуються для визначення максимального значення, біти OC1A чи, відповідно, ICF1 встановлюється в «1» при досягненні лічильником максимального значення. Одночасно, якщо це потрібно, у регістрі, що використовується для задавання максимального значення, змінюється значення на нове.

Частота вихідного сигналу визначається за формулою

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{2\,N\,TOP}$$

де N – значення коефіцієнта поділу попереднього подільника, а TOP – максимальне значення, до якого може дорахувати лічильник.

Режим широтно-імпульсної модуляції коректний по фазі та частоті.

Режим широтно-імпульсної модуляції, коректний по фазі та частоті, який подібний на режим із коректною фазою, базується на відліку лічильника у двох напрямках. Основна відмінність між режимами полягає в часі оновлення максимального значення, до якого може рахувати лічильник. У режимі з коректною фазою регістр, що задає максимальне значення відліку, змінюється в

будь-який момент часу. У режимі з коректною фазою та частотою значення у регістрі максимального значення відліку змінюється лише у момент, коли лічильник досягнув мінімального значення і почав рахувати у бік збільшення. Часова діаграма режиму наведена на рис. 146.

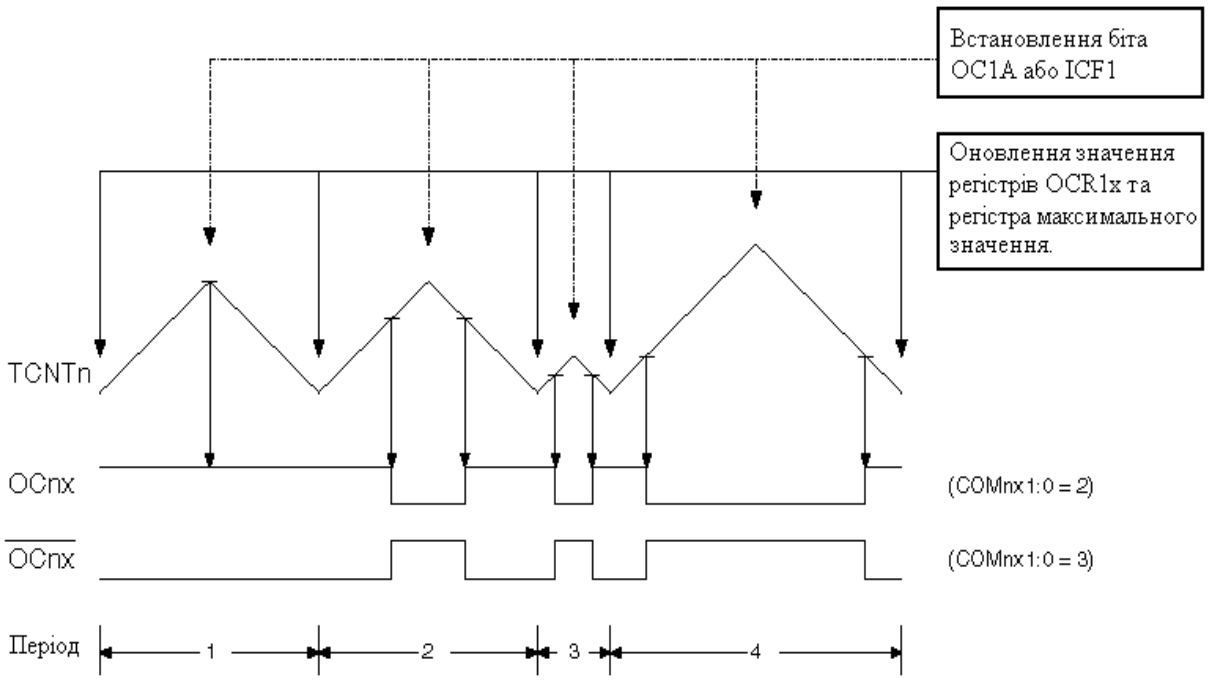


Рисунок 146. Режим широтно-імпульсної модуляції коректний по фазі та частоті

Вихід лічильника-таймера 1.

Таймер-лічильник 1 має два незалежні виходи, виведені на лінії OC1A та OC1. Стан виходу залежить від обраного режиму роботи таймера-лічильника та конфігурації виходів. Конфігурація виходів задається для кожної лінії окремо бітами COM1A1:0 та COM1B1:0.

Спрощена схемотехніка виходу таймера наведена на рисунку 147. З неї бачимо, що видачею даних керує регістр напрямку DDR і для передавання даних необхідно, щоб лінія працювала на вивід. Якщо ж лінія буде працювати на ввід даних, стан виходу не буде перемикатися. При під’єднанні до лінії порту виходу лічильника стан лінії може контролюватися за допомогою відповідного регістра PINx.

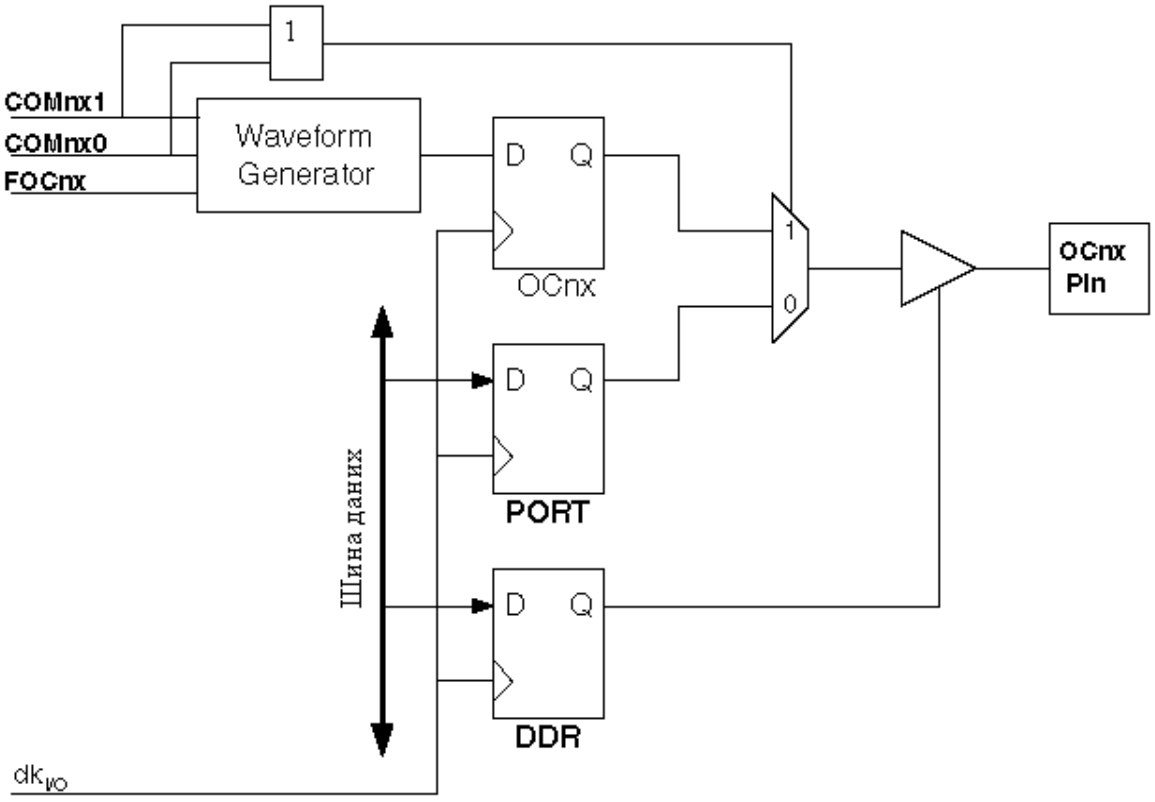


Рисунок 147. Спрощена схема виходів таймера-лічильника 1

Режим роботи виходів OC1A та OC1B (таблиці 46, 47, 48) залежить від ста-ну бітів COM1A1-0 та COM1B1-0 регістра TCCR1A, а також обраного режиму роботи.

Таблиця 46. Режими роботи ліній OC1A та OC1B залежно від стану бітів регістра TCCR1A у режимах, не пов’язаних із ШІМ

COM1x1	COM1x0	Опис роботи лінії OC1x
0	0	Нормальний режим роботи порту, біт OC1x від’єднаний від порту
0	1	Лінія OC1x перемикається при кожному співпаданні
1	0	Лінія OC1x скидається при кожному співпаданні
1	1	Лінія OC1x встановлюється при кожному співпаданні

Таблиця 47. Режими роботи ліній OC1x залежно від стану бітів регістра TCCR1A у режимі швидкого ШІМ

COM1x1	COM1x0	Опис роботи лінії OC1x
0	0	Нормальний режим роботи порту, біти OC1x від'єднані від порту
0	1	Якщо WGM13:0 = 1111, вихід OC1A перемикається при кожному співпаданні, а OC1B вимикається. В усіх решта режимах режим вважається зарезервованим, а біти OC1x від'єднаі від порту
1	0	Біт OC1x скидається при рівності вмісту таймера (TCNT1) та регістра OCR1x, встановлюється при переході лічильника TCNT1 до нульового значення (неінвертуючий режим)
1	1	Біт OC1x встановлюється при рівності вмісту таймера (TCNT1) та регістра OCR1x, скидається при переході лічильника TCNT1 до нульового значення (інвертуючий режим)

Таблиця 48. Режими роботи ліній OC1x залежно від стану бітів регістра TCCR1A у режимах, коректних за фазою та режимі, коретному за фазою та частотою ШІМ

COM1x1	COM1x0	Опис роботи лінії OC1x
0	0	Нормальний режим роботи порту, біти OC1x від'єднані від порту
0	1	Якщо WGM13:0 = 1111 вихід OC1A перемикається при кожному співпаданні, а OC1B вимикається. В усіх решта режимах режим вважається зарезервованим, а біти OC1x від'єднані від порту
1	0	Біт OC1x скидається при рівності вмісту таймера (TCNT1) та регістра OCR1x, встановлюється при переході лічильника TCNT1 до нульового значення (неінвертуючий режим)
1	1	Біт OC1x встановлюється при рівності вмісту таймера (TCNT1) та регістра OCR1x, скидається при переході лічильника TCNT1 до нульового значення (інвертуючий режим)

Доступ до 16-бітних регістрів таймера-лічильника 1.

Регістри TCNT1, OCR1A/B та ICR1, пов'язані з таймером-лічильником 1, є 16-бітними. Для доступу до цих регістрів, зазвичай, використовують дві наступні одна за одною операції читання чи записування. При звертанні до регістрів 16-розрядний таймер має один 8-розрядний регістр для тимчасового зберігання старшого байта, що є загальним для всіх 16-розрядних регістрів. При читанні молодшого байта із 16-бітних регістрів TCNT1, ICR1 старший байт автоматично розміщується у цей тимчасовий регістр і зчитується з нього, коли відбувається читання із регістра старшого байта. Так як буферний регістр є єдиним для всіх регістрів, немає можливості зчитати молодші байти із обох 16-бітних регістрів (TCNT1L, ICR1L), а потім звернутися до регістрів, що зберігають старші байти (TCNT1H, ICR1H, відповідно).

Запис даних у 8-бітні регістри TCNT1H, ICR1H, OCR1A, OCR1B не призводить до моментального оновлення даних. Дані лише розміщуються в тимчасовий регістр, перенесення даних у регістр старшого байта відбувається у момент запису даних у регістр молодшого байта. Тобто, оновлюються усі 16 бітів і молодший байт задається командою, а старший – тимчасовим регістром.

Отже, для коректного звертання до регістрів TCNT1, ICR1 їх читання слід починати із молодшого байта, а запис – із старшого. Для регістрів OCR1A та OCR1B порядок читання байтів не важливий, а запис має починатися з старшого байта. При цьому, не можна чергувати звертання до регістрів із різними назвами, а запис із читанням (крім читання регістрів OCR1A, OCR1B). Якщо у підпрограмі опрацювання переривань звертаються до 16-бітних регістрів таймера, слід унеможливити доступ до них із підпрограми опрацювання, якщо основна програма звернулася до одного з регістрів. У найпростішому випадку, при звертанні до 16-бітних регістрів, які користуються тимчасовим регістром, слід забороняти переривання.

Наступний приклад показує типовий варіант звертання до цих 16-бітних регістрів.

```
    TIM16_WriteTCNT1:
    in r18,SREG ; Зберігаємо загальну ознаку дозволу
переривань
    cli ; забороняємо переривання
    out TCNT1H,r17 ; Записуємо у регістр TCNT1 дані із
регістрів R17:R16
    out TCNT1L,r16
    out SREG,r18 Відновлюємо загальну ознаку дозволу
    ret.
```

3.10.3.2. Регістри таймера-лічильника 1.

Для визначення режиму роботи таймера-лічильника використовуються регістри TCCR1A та TCCR1B. Формат регістра TCCR1A наведений на рис. 148, а регістра TCCR1B – на рис. 149.

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 148. Формат регістра TCCR1A

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 149. Формат регістра TCCR1B

Біти WGM13-10 задають режим роботи таймера-лічильника (табл. 45).

Біти COM1A1-0, COM1B1-0 задають режим роботи виходів OC1A та OC1B.

Біти FOC1A та FOC1B дозволяють примусово встановити вихід у режимах, не пов’язаних із ШІМ. Запис у біт FOC1A чи FOC1B одиниці примусово перемикає вихід OC1A чи OC1B відповідно. Біт не зберігає свого значення й читається завжди як «0».

Біт ICNC1 вмикає цифровий фільтр на вході пристрою захоплення (рис. 129). Біт ICES1 визначає фронт, за котрим спрацьовує пристрій захоплення. Якщо біт має нульове значення, то використовується спадаючий (задній) фронт, одиничне значення вказує на зростаючий фронт.

Біти CS12-10 задають коефіцієнт поділу попереднього подільника.

Регістр відліку TCNT1.

Основним регістром таймера-лічильника 1 є 16-бітний регістр відліку TCNT1, що поділений на 2 регістри TCNT1H та TCNT1L (рис. 150).

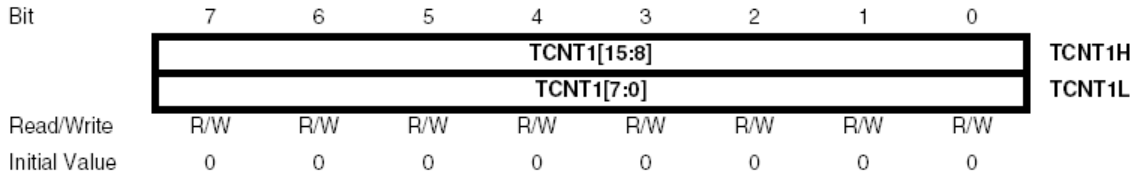


Рисунок 150. Регістр відліку TCNT1

Регістр TCNT1L доступний як для прямого читання, так і для записування, а регістр TCNT1H читається та записується через тимчасовий програмно прозорий регістр TEMP. При читанні регістра TCNT1L старший байт таймера-лічильника розташовується у тимчасовому регістрі, а потім може бути зчитаний при звертанні до регістра TCNT1H. При записуванні нові дані, що пишуться у TCNT1H, розміщуються у регістр TEMP, а потім перезаписуються в регістр лічильника у момент записування в регістр TCNT1L.

Отже, читання регістра TCNT1 має починатися з молодшого байта, а запис – зі старшого. При цьому між доступом до молодшого й старшого байтів TCNT1 не має вклинюватися доступ до інших 16-бітних регістрів таймера-лічильника.

Регістр захоплення.

Регістр захоплення ICR1 призначений для фіксації поточного стану таймера-лічильника за зовнішньою подією. Доступ до регістра здійснюється через два 8-бітні регістри ICR1H та ICR1L. Формат регістрів зображено на рис. 151.

Регістр ICR1L доступний як для прямого читання, так і для запису, а регістр ICR1H читається та записується через тимчасовий програмно прозорий регістр TEMP.

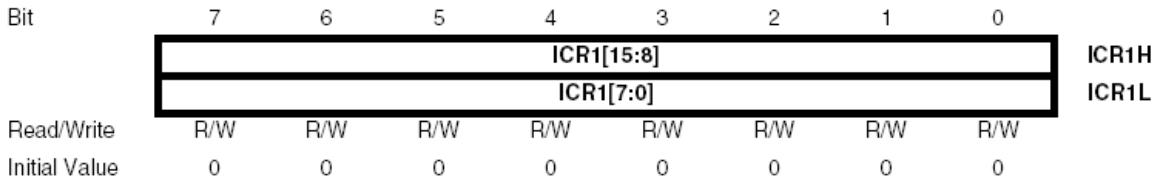


Рисунок 151. Регістри захоплення ICR1H та ICR1L

При читанні регістра ICR1L старший байт таймера-лічильника розміщується у тимчасовий регістр, а потім може бути зчитаний при звертанні до регістра ICR1H.

При записуванні нові дані, що пишуться у ICR1H, розміщуються в регістр TEMP, а потім перезаписуються у регістр лічильника в момент записування в регістр ICR1L. Отже, читання регістра ICR1 має починатись із молодшого байта, а запис – зі старшого, при цьому між доступом до молодшого і старшого байтів ICR1 не має вклинюватися доступ до інших 16-бітних регістрів таймера-лічильника.

Регістри порівняння.

Регістри порівняння призначені для збереження значення, з яким порівнюється значення лічильника. Результат порівняння може бути використаний для генерації переривань, широтно-імпульсної модуляції тощо. Таймер 1 має 2 регістри порівняння OCR1A (рис. 152) та OCR1B (рис. 153). Довжина кожного регістра – 16 бітів. Доступ до кожного з них здійснюється через два 8-бітні регістри. Для доступу до регістра OCR1A використовують OCR1AH та OCR1AL, а для доступу до OCR1B – регістри OCR1BH та OCR1BL.

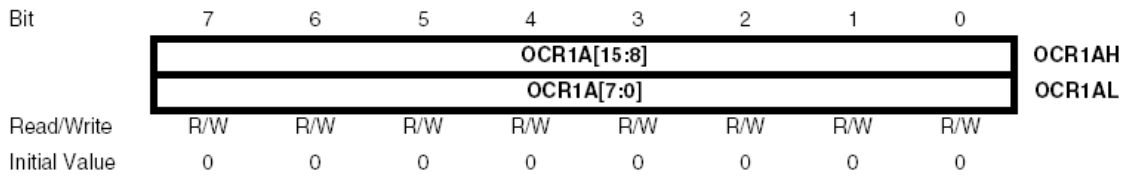


Рисунок 152. Формат регістра OCR1A

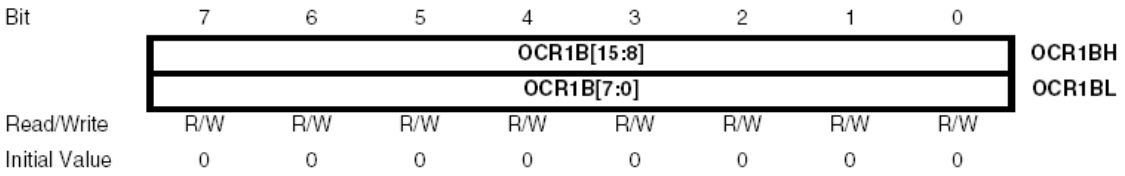


Рисунок 153. Формат регістра OCR1B

Регістри OCR1AL та OCR1BL доступні для прямого читання та записування. У свою чергу, регістри OCR1AH та OCR1BH доступні для прямого читання, а записуються через тимчасовий програмно прозорий регістр TEMP: при записуванні нові дані, що пишуться у OCR1AH та OCR1BH, розміщуються у регістр TEMP, а потім перезаписуються у відповідний регістр у момент записування в регістр OCR1AL чи OCR1BL відповідно. Отже, для коректного записування даних запис в регістри OCR1A та OCR1B має починатися зі старшого байта, при цьому між доступом до молодшого і старшого байтів регістра не повинен вклинюватися доступ до інших 16-бітних регістрів таймера-лічильника.

3.10.3.3. Біти загальних регістрів, що використовує таймер 1.

Регістр маски переривань TIMSK (рис. 154) є загальним для усіх таймерів. Для таймера 0 використовуються біти OCIE0 та TOIE.

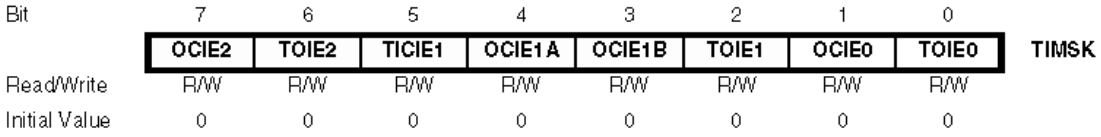


Рисунок 154. Регістр маски переривань TIMSK

Біт TICIE1, встановлений в «1», дозволяє переривання при захопленні для таймера-лічильника 1 (біт ICF1).

Біт OCIE1A, встановлений в «1», дозволяє переривання від блока порівняння A таймера-лічильника 1 (біт OCF1A), а біт OCIE1B дозволяє переривання від блока порівняння B (біт OCF1B).

Біт TOIE1, встановлений в «1», дозволяє переривання при переповненні таймера-лічильника 1 (біт TOV1).

У регістрі переривань таймерів TIFR (рис. 155) відображається стан переривань від усіх таймерів мікросхеми. До таймера-лічильника 1 відносяться біти ICF1 OCF1A OCF1B TOV1.

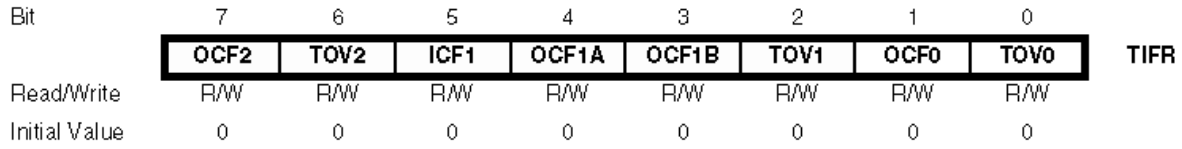


Рисунок 155. Регістр переривань TIFR

Біт ICF1 вказує на захоплення значення лічильника у відповідному регістрі.

Біти OCF1A та OCF1B відображають стани запитів переривань від блоків порівняння A та B, а біт TOV1 – переривання при переповненні таймера. Біти скидаються автоматично при виклику відповідного переривання, крім того, вони можуть бути скинуті програмним шляхом.

3.10.4. Переривання від таймерів.

Переривання, котрі можуть бути викликані таймерами-лічильниками, наведено в таблиці 49.

Таблиця 49. Переривання від таймерів-лічильників

Адреса переходу	Назва біту	Джерело виклику
$008	OCF2	Співпадання в блоку порівняння таймера-лічильника 2
$00A	TOV2	Переповнення таймера-лічильника 2
$00C	ICF1	Відбулося захоплення стану в блоку захоплення таймера-лічильника 1
$00E	OCF1A	Співпадання в блоку порівняння A таймера-лічильника 1
$010	OCF1B	Співпадання в блоку порівняння B таймера-лічильника 1
$012	TOV1	Переповнення таймера-лічильника 1
$014	OCF0	Співпадання в блоку порівняння таймера-лічильника 0
$016	TOVF0	Переповнення таймера-лічильника 0

Типова структура початкової області пам’яті програм наведена нижче.

```
.org 0
$000 jmp RESET        ;Вектор скидання
$002 jmp EXT_INT0     ;Вектор обробки IRQ0
$004 jmp EXT_INT1     ;Вектор обробки IRQ1
$006 jmp EXT_INT2     ;Вектор обробки IRQ2
$008 jmp TIM2_COMP    ;Вектор обробки переривання
                      ;співпадання в блоку
                      ;порівняння лічильника-
                      ;таймера 2
$00A jmp TIM2_OVF     ;Вектор обробки
                      ;переповнення
                      ;лічильника-таймера 2
$00C jmp TIM1_CAPT    ;Вектор обробки захоплення
                      ;стану у блок
                      ;захоплення лічильника-
                      ;таймера 1
$00E jmp TIM1_COMPA   ;Вектор обробки співпадання
                      ;в блоку порівняння A
                      ;лічильника-таймера 1
$010 jmp TIM1_COMPB   ;Вектор обробки співпадання
                      ;в блоку порівняння B
                      ;лічильника-таймера 1
$012 jmp TIM1_OVF     ;Вектор обробки
                      ;переповнення
                      ;лічильника-таймера 1
$014 jmp TIM0_COMP    ;Вектор обробки співпадання
                      ;в блоку порівняння
                      ;лічильника-таймера 0
$016 jmp TIM0_OVF     ;Вектор обробки переповнення
                       ;лічильника-таймера 0
$018 jmp SPI_STC      ;Вектор обробки закінчення
                      ;передавання даних по шині SPI
$01A jmp USART_RXC    ;Вектор обробки закінчення
                      ;прийому USART
$01C jmp USART_UDRE   ;Вектор обробки очищення
                      ;регістра UDR
$01E jmp USART_TXC    ;Вектор обробки закінчення
                      ;передавання USART
$020 jmp ADC          ;Вектор обробки закінчення
                      ;аналого-цифрового перетворення
$022 jmp EE_RDY       ;Вектор обробки готовності
                      ;EEPROM
$024 jmp ANA_COMP     ;Вектор обробки переривання від
                      ;аналогового компаратора
$026 jmp TWI          ;Вектор обробки переривання від
                      ;шини TWI
$028 jmp SPM_RDY      ;Вектор готовності пам’яті
                      ;програм
```

3.11. Загальна характеристика послідовних інтерфейсів мікросхеми.

Мікросхема ATMega32 у своїй структурі має три послідовних інтерфейси:

◊ синхронний послідовний інтерфейс SPI;

◊ асинхронний інтерфейс TWI (аналог інтерфейсу I2C);

◊ послідовний синхронно-асинхронний універсальний інтерфейс USART.

Кожен із інтерфейсів таймерів-лічильників орієнтований на виконання свого кола задач і має відповідні особливості.

Послідовний периферійний інтерфейс SPI (Serial Peripheral Interface) призначений для організації обміну даними між двома пристроями. З його допомогою може здійснюватися обмін даними між мікроконтролером і різними пристроями, такими, як цифрові потенціометри, ЦАП та АЦП, FLASH-ПЗП тощо. За допомогою цього інтерфейсу зручно проводити обмін даними між кількома мікроконтролерами.

Двопровідний послідовний інтерфейс TWI (Two-wire Serial Interface) є повним аналогом базової версії інтерфейсу I2C фірми Philips. Цей інтерфейс дозволяє об’єднати разом до 128 різних пристроїв за допомогою двобічної шини, що складається з лінії тактового сигналу (SCL) і лінії даних (SDA). Проте швидкість обміну за цим інтерфейсом є меншою, ніж за інтрефейсом SPI.

Універсальний асинхронний або універсальний синхронно/асинхронний приймач (Universal Synchronous / Asynchronous Receiver and Transmitter – UART або USART) – зручний і простий послідовний інтерфейс для організації інформаційного каналу обміну мікроконтролера із «зовнішнім світом». Він здатний працювати в дуплексному режимі (одночасне передавання й приймання даних) і підтримує протокол стандарту RS-232, що забезпечує можливість організації зв’язку з персональним комп’ютером.

3.11.1. Послідовний порт SPI.

3.11.1.1. Загальний опис логіки роботи.

Інтерфейс SPI (Serial Peripheral Interface, SPI bus – послідовний периферійний інтерфейс,) – послідовний синхронний стандарт передавання

даних у режимі повного дуплексу, розроблений компанією Motorola для забезпечення простого та дешевого спряження мікроконтролерів та периферії. В інтерфейсі SPI є два типи пристроїв: ведучий, котрий керує передаванням, та ведений, котрий приймає та передає дані за командами ведучого. Зазвичай, на шині SPI є один ведучий пристрій та один чи кілька ведених.

На відміну від більшості інших, інтерфейс SPI є синхронним, у котрому будь-яке передавання даних синхронізоване зі спільним тактовим сигналом, що генерується або ведучим пристроєм, або незалежним тактовим генератором. Ведені мікросхеми синхронізують отримання бітової послідовності з тактовим сигналом. Ведучий пристрій (чи один із ведучих) обирає ведений пристрій за допомогою сигналу SS (slave select). Мікросхеми, не обрані сигналом SS, не мають права видавати дані на шину (хоча й можуть приймати дані). У шині SPI використовують 4 цифрові сигнали (рис. 156):

◊ MOSI (або SI) – вихід ведучого, вхід веденого (Master Out Slave In), використовується для передавання даних від ведучого пристрою до веденого.

◊ MISO (або SO) – вхід ведучого, вихід веденого (Master In Slave Out), використовується для передавання даних від веденого пристрою до ведучого.

◊ SCK (або CLK) – послідовний тактовий сигнал (Serial Clock), використовується для передавання тактового сигналу для ведених пристроїв та, можливо, для ведучих.

◊ SS (або CS) – вибір мікросхеми (Chip Select, Slave Select) для ведених пристроїв – вибір активного веденого пристрою, у деяких ведучих пристроях може використовуватися для арбітражу.

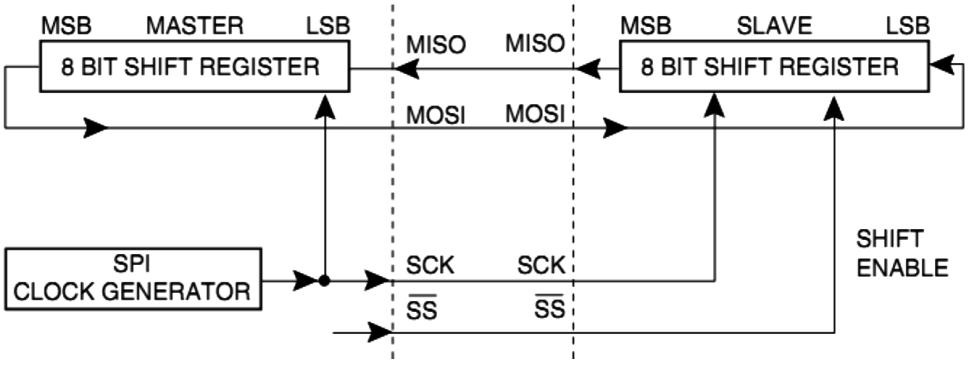


Рисунок 156. Зв’язок за інтерфейсом SPI

Типова структура апаратних засобів може бути представлена за допомогою двох регістрів зсуву, що утворюють 16-бітний кільцевий регістр зсуву. Передавання даних синхронізується тактовим сигналом, котрий може бути як зовнішнім, так і внутрішнім для ведучого пристрою. Перед початком передавання ведучий пристрій передає на лінію SS веденої мікросхеми логічний «0» (рис. 157). Зауважимо, що деякі ведені пристрої потребують певної затримки між подаванням сигналу вибірки та подаванням на лінію SCK тактових сигналів.

Далі, на кожному із тактів сигналу SCK, ведучий пристрій посилає на лінію MOSI новий біт із зсувного регістра, а з лінії MISO читає біт даних, надісланий веденим пристроєм, і розміщує їх у зсувний регістр. Ведений пристрій, у свою чергу, в той самий час посилає на лінію MISO нові біти даних зі свого зсувного регістра, а з лінії MOSI читає біти від ведучого пристрою. Отже, через 8 тактів ведений та ведучий пристрої обміняються байтами даних. Не завжди потрібно проводити передавання даних у двох напрямках – у такому випадку зайві дані можна просто ігнорувати, а лінію, за котрою буде відбуватися зайве передавання даних, можна взагалі не під’єднувати до веденої мікросхеми. У багатьох випадках передавання можуть містити будь-яку кількість бітів але найчастіше, кількість бітів, що передається, є кратним 8.

На додаток до тактової частоти у процесі програмування слід також обрати полярність тактового сигналу та фазу даних відносно тактового сигналу. Зазвичай, вибір полярності тактового сигналу та фази даних виконується бітами CPOL і CPHA у регістрі керування (табл. 50). Біт CPOL задає базове значення тактового сигналу.

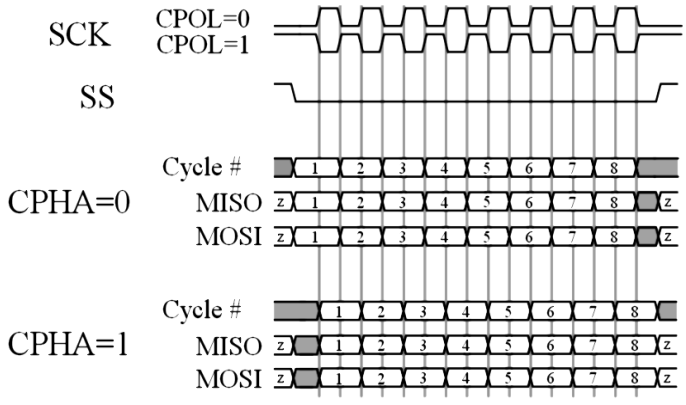


Рисунок 157. Полярність синхронізації та фаза даних, що передаються

Таблиця 50. Значення рівнів сигналів SCK та CPHA

CPOL	Активний рівень сигналу SCK	CPHA	Фаза даних
0	Логічний 1	0	На наявність даних вказує перехід сигналу SCK у активний рівень
1	Логічний 0	1	На наявність даних вказує перехід сигналу SCK у пасивний рівень

3.11.1.2. Конфігурації з багатьма веденими.

При роботі порту у веденому режимі (Slave) інтерфейс SPI залишається вимкненим, а його передавач – у z-стані, поки на лінії SS високий рівень. У цьому стані програма може оновлювати вміст регістра даних SPI, але дані не будуть зсунуті до встановлення на лінії SS низького рівня навіть за наявності тактового сигналу. Така організація дозволяє легко під'єднувати до шини SPI кілька ведених мікросхем. Найширше використовують увімкнення з незалежним використанням мікросхем. У такій схемі кожна ведена мікросхема вибирається власним сигналом SS (рис. 158).

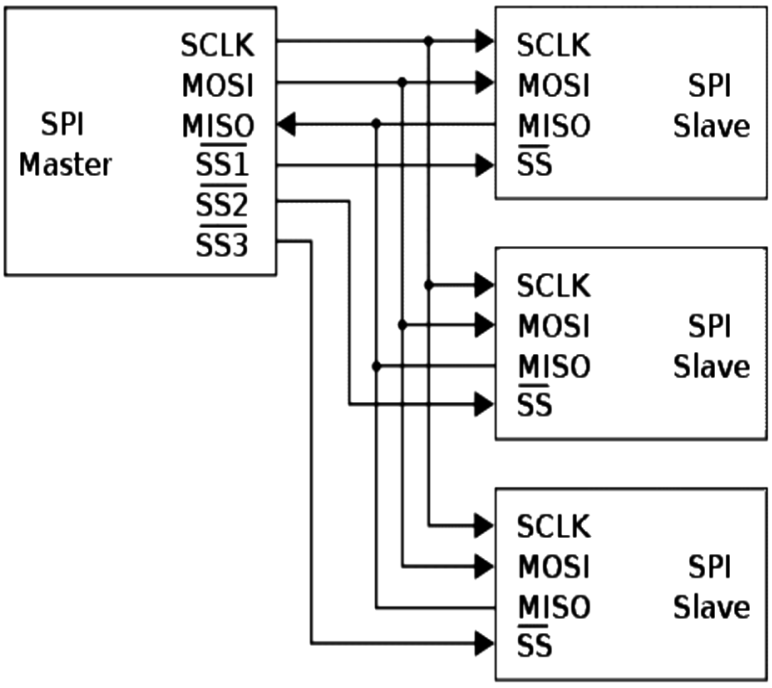


Рисунок 158. Незалежне увімкнення ведених мікросхем

3.11.1.3. Інтерфейс SPI у мікросхемі ATMega32.

У мікросхемах сімейства ATMega інтерфейс SPI має такі можливості:

◊ повний дуплекс при трипровідному синхронному передаванні даних;

◊ використання мікросхеми як у режимі ведучого, так і у режимі веденого пристрою;

◊ можливість передавання даних, починаючи як зі старшого, так і з молодшого бітА;

◊ можливість програмування швидкості обміном.

У сімействі ATMega інтерфейс SPI має однократну буферизацію при передаванні даних і подвійну буферизацію при прийомі: байти, які повинні передаватися, не можуть бути записані в регістр даних SPI до закінчення циклу попереднього передавання даних. Однак при прийманні дані можуть бути прочитані з регістру даних SPI до кінця наступного приймання даних.

При дозволеному SPI напрямі передавання даних лінії MOSI, MISO, SCK і SS налаштовуються у відповідності з табл. 51.

Таблиця 51. Налаштування виводів SPI

Вивід	Ведучий SPI	Ведений
MOSI	Визначається користувачем	Вхід
MISO	Вхід	Визначається користувачем
SCK	Визначається користувачем	Вхід
SS	Визначається користувачем	Вхід

Якщо SPI працює в режимі веденого (Slave), сигнал SS працює як вхід. Коли на лінії SS низький рівень, активізується порт SPI і лінії інтерфейсу працюють як показано у табл. 51. За наявності на лінії SS високого стану, усі виводи є входами, крім, можливо, лінії MISO, а SPI переводиться у пасивний стан, навіть якщо почалося нове передавання чи приймання даних. При переведенні лінії SS у високий рівень ведена мікросхема зупиняє передавання й приймання даних, та видаляє з регістра зсуву будь-які частково прийняті дані.

У режимі ведучого SPI напрям роботи лінії SS задається. Якщо лінія SS налаштована як вихід, стан лінії SS не впливає на роботу SPI. Якщо ж лінія SS

налаштована як вхід, її стан дозволяє провести визначення активної ведучої мікросхеми у системах із кількома ведучими контролерами: високий рівень на лінії SS дозволяє роботу мікросхеми у режимі ведучого, а низький рівень автоматично переводить мікросхему у режим веденої.

Будова послідовного порту SPI зображена на рис. 159. Для отримання тактового сигналу синхронізації використовується тактовий сигнал XTAL, який надходить на подільник DIVIDER, коефіцієнт поділу котрого визначається мультиплексором SELECT. Отриманий тактовий сигнал надходить на логічний елемент (CLOCK LOGIK) та через нього – на зсувний регістр (SHIFT REGISTER). Вихідна логіка узгоджує сигнали із зсувного регістра із заданим режимом роботи ліній передавання даних. Визначення режиму роботи та керування блоками послідовного порту здійснює блок керування (SPI CONTROL), а встановления режиму роботи – регістр режиму (SPI CONTROL REGISTER). Стан послідовного порту контролюється через регістр стану (SPI STATUS REGISTER).

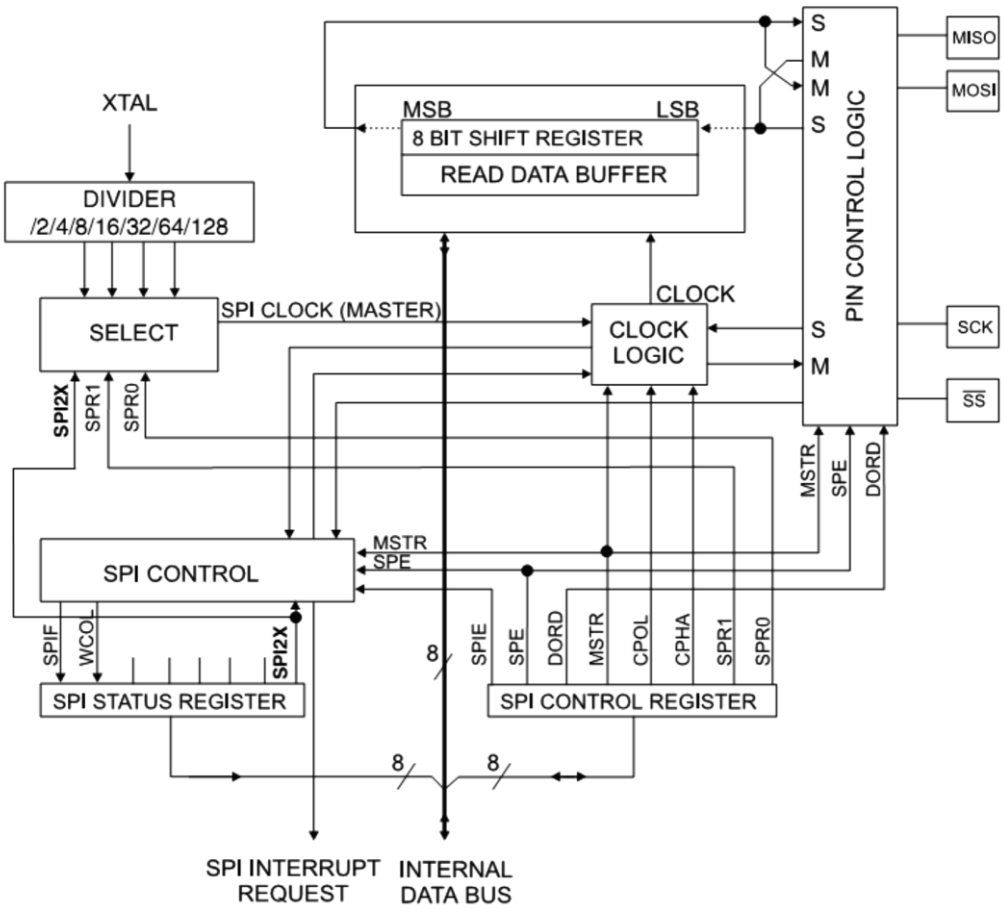


Рисунок 159. Будова послідовного порту

Для контролю стану, програмування, передавання та приймання даних використовують такі регістри:

◊ SPDR – регістр даних SPI (SPI Data Register);

◊ SPCR – регістр керування SPI (SPI Control Register);

◊ SPSR – регістр стану SPI (SPI Status Register).

3.11.1.4. Регістри SPI.

Регістр SPDR призначений для приймання та передавання даних через SPI. Дані, що передаються, не буферизуються, а для прийнятих даних є один однобайтовий буфер. Так як дані, що передаються, не буферизуються, перед записуванням нових даних у регістр SPDR слід дочекатися закінчення попереднього передавання.

Формат регістра SPCR зображений на рис. 160.

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 160. Регістр SPCR

Біти цього регістра мають таке призначення:

SPIE – (SPI Interrupt Enable) дозвіл переривання від SPI. Біт дозволяє генерацію переривання від інтерфейсу SPI при встановленні біта SPIF у регістрі SPSR.

SPE – (SPI Enable) дозвіл роботи інтерфейсу SPI. Одиничне значення біта дозволяє роботу інтерфейсу SPE, нульове – забороняє її.

DORD – (Data Order) порядок передавання даних. Одиниця вказує, що дані передаються від молодших бітів до старших (від розряду D0 до D7), нуль вказує, що дані передаються від старших бітів до молодших (від D7 до D0).

MSTR – (Master/Slave Select) біт вибору режиму роботи «ведучий» чи «ведений». Якщо біт дорівнює «1», то порт працює у ведучому режимі, якщо нуль – у веденому.

CPOL – (Clock Polarity) біт вибору полярності сигналу SCK. Якщо біт дорівнює «1», то активний рівень тактового сигналу – нульовий, а пасивний – одиничний. Якщо нуль – активний рівень сигналу одиничний, а пасивний – нульовий.

Таблиця 52. Частота передавання даних

SPI2X	SPR1	SPR0	Частота сигналу SCK та передавання даних
0	0	0	fosc/4
0	0	1	fosc/16
0	1	0	fosc/64
0	1	1	fosc/128
1	0	0	fosc/2
1	0	1	fosc/8
1	1	0	fosc/32
1	1	1	fosc/64
fosc – частота сигналу генератора			

CPHA – (Clock Phase) біт вибору фази передавання даних. Нульове значення вказує на наявність даних при переході сигналу SCK у активний рівень, одиничне – на наявність даних при переході сигналу SCK у пасивний рівень.

SPR1, SPR0 – (SPI Clock Rate Select) – біти вибору частоти передавання даних. Біти разом із бітом SPI2X визначають швидкість передавання даних по SPI у режимі ведучої мікросхеми із внутрішнім тактовим сигналом. У режимі веденої мікросхеми чи у режимі ведучої із зовнішнім тактовим сигналом біти не використовуються. Частота тактового сигналу визначається згідно з таблицею 52.

Біт SPI2X знаходиться в регістрі SPSR.

Формат регістра SPSR зображений на рис. 161.

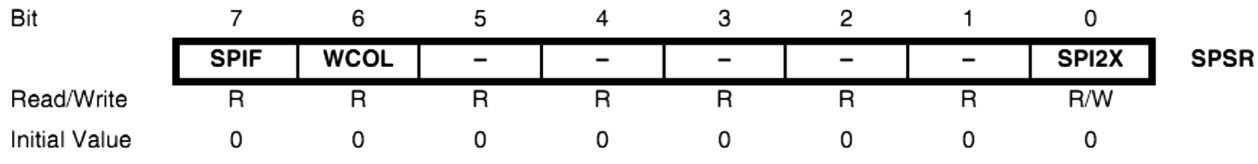


Рисунок 161. Формат регістра SPSR

Біт SPIF (SPI Interrupt Flag) вказує на переривання від порту SPI. Біт встановлюється в «1» при закінченні обміну даними, а у режимі ведучого порту також при передаванні даних керування шиною іншому контролеру (тобто тоді, коли на вхід SS надійшов нуль). При встановленні біта в «1» генерується переривання, якщо біт SPIE у регістрі SPCR встановлений в «1» та встановлений біт загального дозволу переривань. Біт автоматично скидається при виклику відповідного переривання та при виконанні операції читання регістра SPSR.

Біт WCOL (Write COLlision Flag) – біт колізії при записуванні. Одиничне значення вказує на записування у регістр даних SPDR під час передавання даних. Біт автоматично скидається при виконанні операції читання регістра SPSR.

Біт SPI2X (Double SPI Speed Bit) – біт подвоєння швидкості передавання даних. Біт обирає швидкість передавання даних по SPI разом із бітами SPR1, SPR0 (таблиця 52).

3.11.2. Розширення портів вводу-виводу.

Іноді, при проектуванні пристроїв, виникає ситуація, коли виводів мікроконтролера не вистачає, а використати інший мікроконтролер немає можливості. У таких випадках використовують розширення за допомогою зовнішніх розширювачів вводу-виводу, що під'єднуються до мікроконтролера за стандартним інтерфейсом, наприклад, інтерфейсом SPI чи I^2C(TWI). Найпростіший варіант такого розширення – використання зсувних регістрів чи мультиплексорів. Однак у такому випадку лінії портів, що розширюються, будуть працювати виключно на вхід чи на вихід без можливості програмного налаштування напрямку передавання даних. Але таке налаштування у більшості випадків і зайве.

Для введення даних у мікроконтролер може бути використаний взагалі будь-який зсувний регістр, що має паралельні входи завантаження послідовний вихід даних. На схемі, зображеній на рис. 162, розширення вводу здійснюється

за допомогою зсувного регістра типу 74HTC165 (К555ИР9). Перед зчитуванням даних на програмно керовану лінію CAP встановлюється логічний «0», що призводить до записування даних із паралельних входів D15-D0 у зсувний регістр. Потім лінія CAP встановлюється у стан логічної «1» і регістр перемикається в режим зсуву.

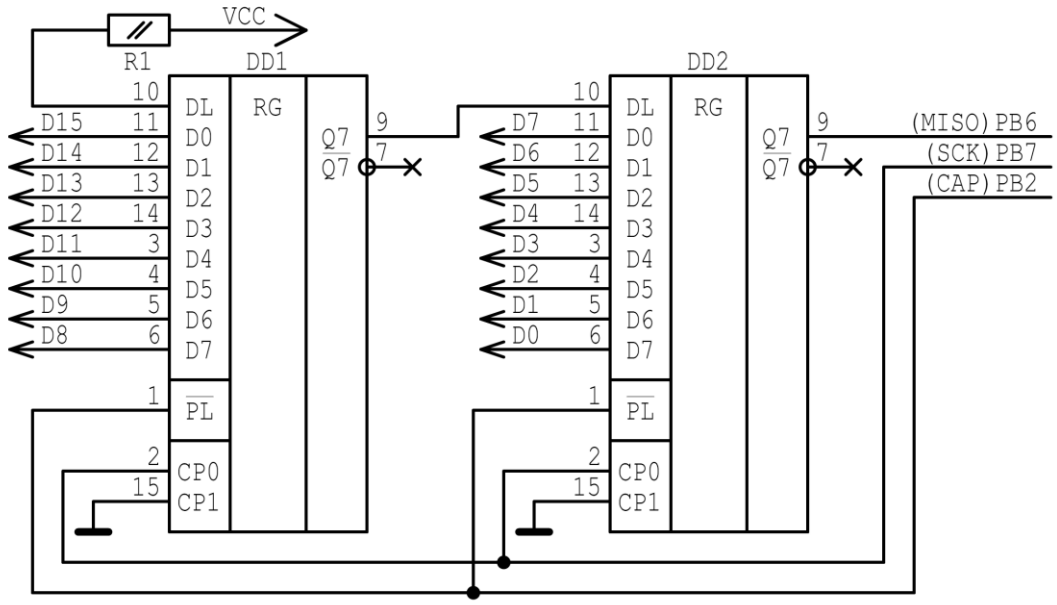


Рисунок 162. Розширення портів вводу за допомогою інтерфейсу SPI. Варіант послідовного читання байтів

Коли послідовним інтерфейсом SPI будуть прийняті 2 байти, зчитування даних буде завершено і регістри можна знову перевести в режим паралельного завантаження. В принципі, у якості сигналу CAP можна використовувати проінвертований сигнал SS (PB4). Позначення D15-D0 відповідають позиціям бітів у двобайтовому слові, що буде прийнято інтерфейсом SPI.

Інший варіант схеми (рис. 163) може мати сигнали вибору, за котрими вихід послідовного коду потрібного зсувного регістра під’єднується до лінії MISO, а не вибрані регістри – вимикаються від неї. Така схема не буде потребувати зчитування усіх байтів для визначення стану одного конкретного порту, але є складнішою, містить більшу кількість мікросхем і потребує додаткового сигналу CSi на кожен додатковий порт вводу. У якості буфера із

Z станом (DD3) можна використати 74LS125 (К555АП8). Вибір джерела сигналу для SPI також можна здійснювати мультиплексором або за допомогою іншої мікросхеми.

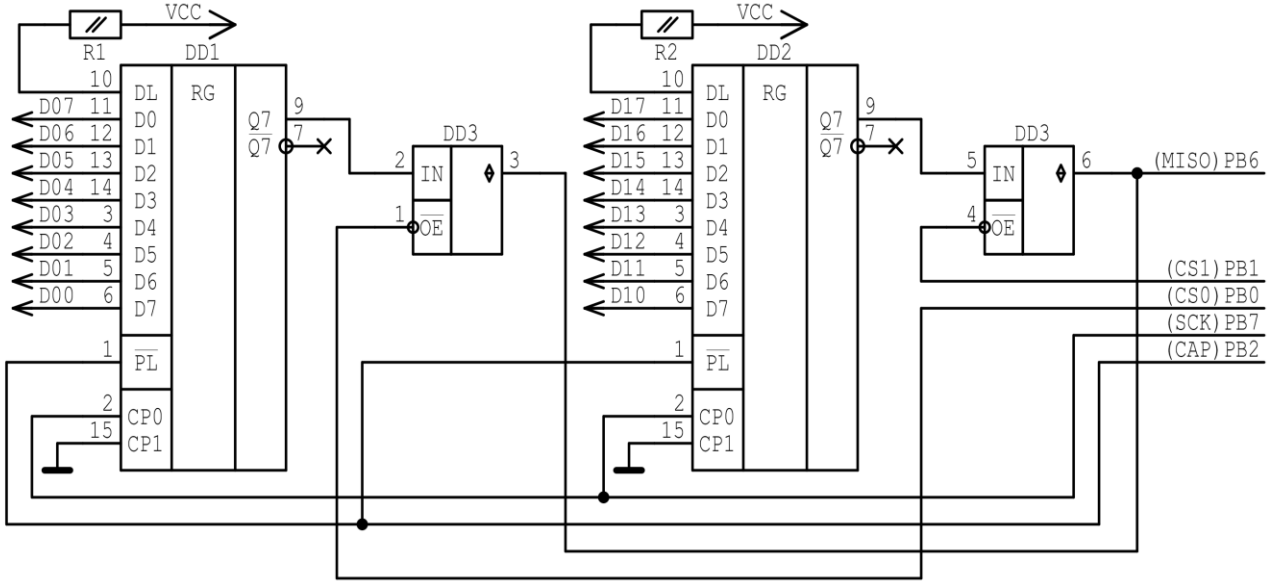


Рисунок 163. Розширення портів вводу за допомогою інтерфейсу SPI. Варіант із вибором регістра окремим сигналом

Натомість, при розширенні порту виводу, використання звичайного зсувного регістра без буферизації може призвести до появи хибних імпульсів на створеному порті виводу. У деяких випадках поява короткочасних імпульсів на виходах, наприклад, коли до виходів під'єднані індикатори, не є критичною і не призводить до хибного спрацювання механізмів чи обладнання. Якщо ж поява таких імпульсів не має бути дозволена, слід застосовувати зсувні регістри з буферизацією даних на паралельних виходах, наприклад, зсувний регістр 74HC595. Цим регістром можна керувати як за допомогою звичайних виводів мікроконтролера, так і за допомогою інтерфейсу SPI. Також його можна каскадувати, з'єднуючи мікросхеми в єдиний великий зсувний регістр.

Мікросхема 74HC595 є 8-розрядним зсувним регістром з регістром зберігання й вихідними буферами з трьома станами (рис. 164).

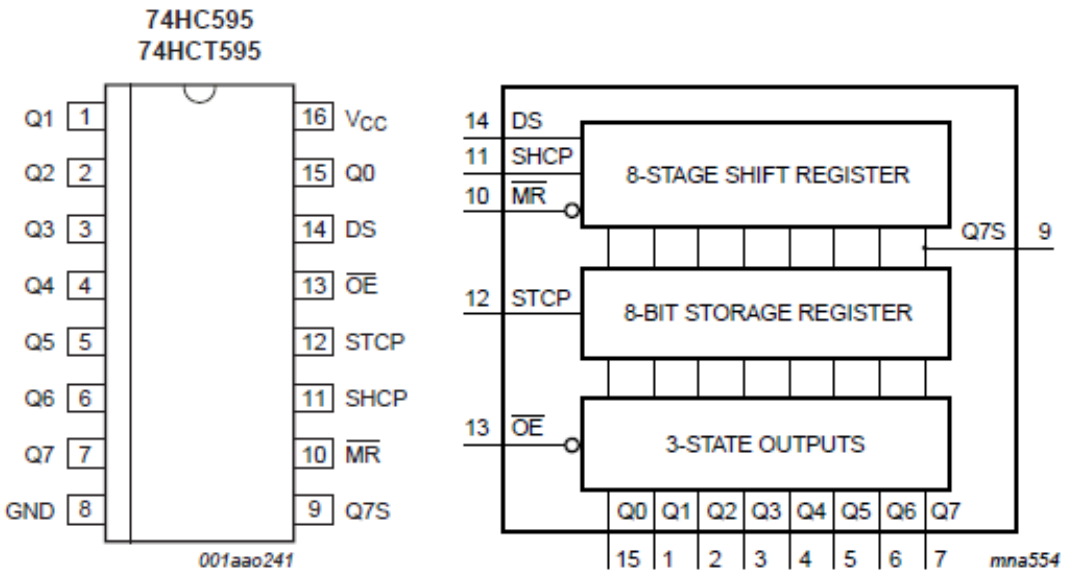


Рисунок 164. Зсувний регістр 74HC595

Призначення її виводів є типовим для таких регістрів:

- DS – вхід даних зсувного регістра у послідовному коді;
- SHCP – тактовий вхід зсувного регістра;
- MR – вхід скиду зсувного регістра;
- STCP – тактовий вхід регістра зберігання даних;
- OE – вхід дозволу вихідних буферів;
- Q0…Q7 – виходи даних у паралельному коді;
- Q7S – вихід каскадування;
- GND, VCC – виводи живлення.

Вхід скидання MR у робочому стані підтягнутий до плюса живлення. Низький логічний рівень на цьому виводі стирає вміст зсувного регістра. Вміст регістра зберігання при цьому ніяк не змінюється.

На вході DS встановлюється необхідний логічний рівень. За позитивним перепадом тактового сигналу на вході SHCP вміст зсувного регістра (розряди з 0 по 7) зміщується на один розряд, а нульовий розряд регістра зберігає логічний рівень, встановлений на вході DS. При зміщенні сьомий розряд зсувного регістра не затирається, а зберігається у внутрішньому тригері й транслюється на вивід Q7S, який призначений для каскадування зсувного регістра.

Для записування одного байта даних описану послідовність потрібно повторити вісім разів.

За позитивним перепадом тактового сигналу на вході STCP дані з виходу зсувного регістра записуються в регістр зберігання. Якщо на виводі OE буде низький логічний рівень, то дані регістра зберігання встановляться на виходах Q0...Q7, в іншому випадку ці виходи будуть знаходитися в третьому стані.

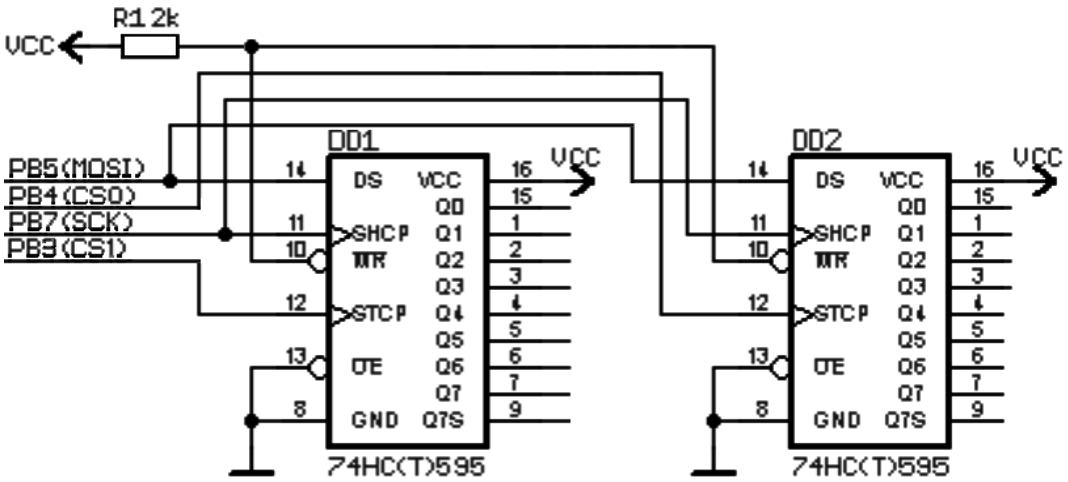
Для каскадування зсувного регістра вихід Q7S під’єднують до входу даних DS наступного регістра, а виводи SHCP, MR, STCP, OE одного регістра з’єднують з аналогічними виводами іншого. Приклад каскадного ввімкнення двох зсувних регістрів можна бачити на схемі нижче.

Управління зсувними регістром 74HC595 можна реалізувати як програмно, так і апаратно, використовуючи SPI модуль мікроконтролера AVR. Останнє можливе завдяки тому, що часова діаграма сигналів керування 74HC595 збігається з діаграмою формованої модулем SPI.

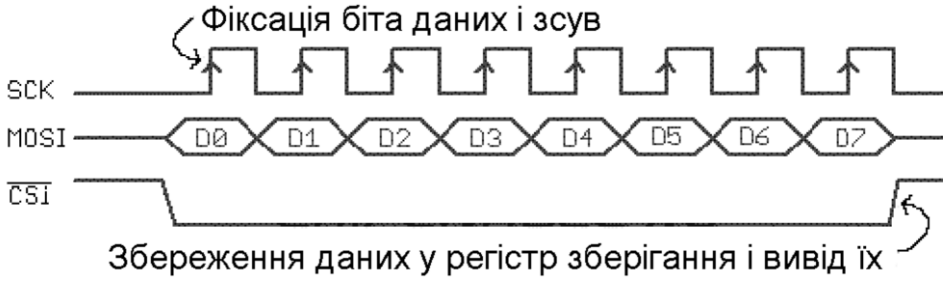
У разі програмного керування виводи зсувного регістра під’єднуються до будь-яких виводів загального призначення. Для апаратного керування зсувним регістром його потрібно під’єднати таким чином (рис. 165):

- DS - > MOSI;
- SHCP - > SCK;
- MR - > через резистор до VCC;
- OE - > GND.

Лінія STCP використовується для вибору потрібного регістра, що буде завантажуватися. Є два варіанти під’єднання: лінію можна використовувати для вибору активного регістра і здійснювати завантаження лише у потрібний регістр, а можна дані пересилати в усі регістри, з’єднавши у ланцюжок усі регістри каскадно й використовувати сигнал для вказування, що всі дані знаходяться на своїх місцях. У першому випадку кожен регістр може бути завантажений незалежно від інших, проте схема потребує сигналу CS на кожен регістр і не може змінювати стан усіх портів одночасно.



а)



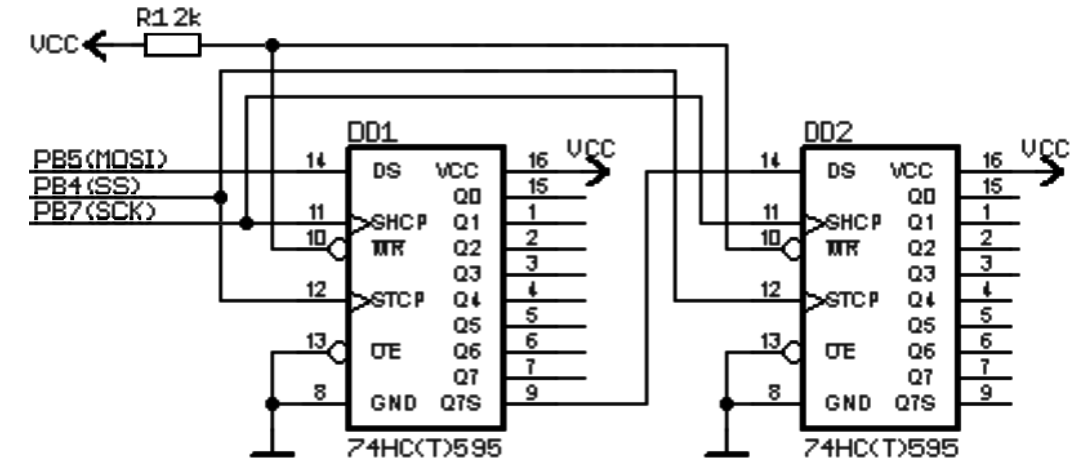
б)

Рисунок 165. Розширення портів виводу за допомогою інтерфейсу SPI. Схема з незалежним вибором регістрів: а) – схема, б) – часова діаграма роботи

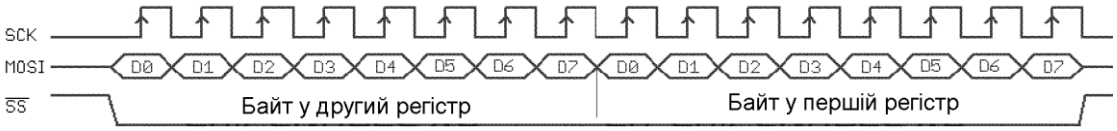
Для записування даних у порт необхідно скинути сигнал CSi відповідного порту, переслати 8 біт даних і встановити сигнал CSi у високий рівень.

У другому випадку усі регістри треба перезавантажувати при зміні стану хоча б одного біта, але схема потребує меншу кількість ліній і змінює стан усіх вихідних портів одночасно. Для записування даних у порт необхідно скинути сигнал SS, переслати всі дані для всіх портів, починаючи із останнього у ланцюжку, і встановити сигнал SS. В обох схемах виходи OE усіх регістрів під’єднані до нуля, що «назавжди» дозволяє виходи регістрів, виводи MR підтягнуті через резистор до плюса живлення (рис. 166).

Усі наведені схеми можуть бути з легкістю розширені до необхідної кількості виводів.



а)



б)

Рисунок 166. Розширення портів виводу за допомогою інтерфейсу SPI. Схема з одночасним записом у регістри: а) – схема, б) – часова діаграма роботи

3.11.3. Універсальний синхронно-асинхронний послідовний прийомо-передавач USART.

3.11.3.1. Загальний опис.

Універсальний синхронно-асинхронний послідовний приймач (Universal Synchronous and Asynchronous serial Receiver and Transmitter – USART) є дуже гнучким пристроєм послідовного передавання інформації. Спрощена блок-схема передавача USART наведена на рис. 167.

Він має такі основні особливості:

◊ незалежні регістри послідовного приймання й передавання даних та можливість одночасного приймання та передавання даних;

◊ синхронний і асинхронний режими роботи;

◊ синхронізацію як від ведучого, так і від веденого пристрою;

◊ вибір швидкості передавання інформації в широких межах;

◊ підтримку кадрів довжиною 5–9 бітів і 1 або 2 стоп-біти;

◊ апаратну підтримку генерації й перевірки сигналу парності;

◊ виявлення переповнювання даних;

◊ виявлення помилок кодування;

◊ низькорівневу цифрову фільтрацію й виявлення помилкового стопового біта;

◊ три джерела переривання: «передавання завершено», «регістр даних передавача порожній», «прийом завершено»;

◊ режим міжпроцесорного зв'язку;

◊ два швидкісні режими асинхронного передавання.

На схемі (рис. 167) штрих-пунктирною лінією обведені три основні частини USART: синхрогенератор (Clock generator), передавач (Transmitter), приймач (Receiver).

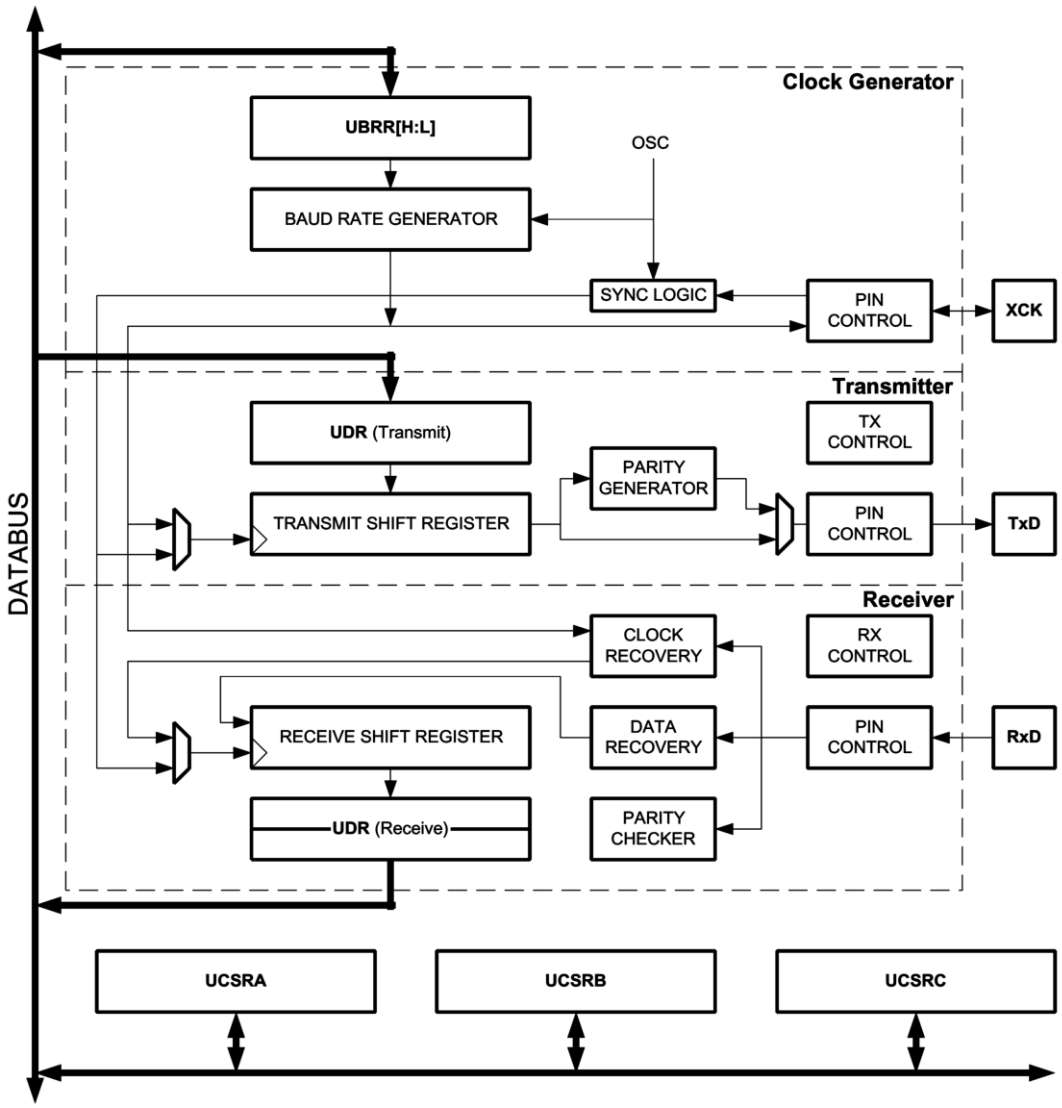


Рисунок 167. Блок-схема USART

Регістри керування загальні для всіх трьох модулів. Логіка генерації тактового сигналу синхронізації складається з:

◊ зовнішнього входу тактового сигналу, що використовується у веденому режимі;

◊ генератора швидкості передавання.

Вхід XCK (тактовий сигнал передавання) використовується тільки в режимі синхронного передавання.

Передавач складається з окремого буфера записування, послідовного зсувного регістра, генератора сигналу парності й логіки контролю для роботи з різними послідовними форматами кадру. Буфер записування забезпечує безперервне передавання даних без затримання між кадрами.

Приймач – найскладніша частина модуля USART через наявність модуля відновлення даних та модуля відновлення тактової частоти. Модулі відновлення використовуються в режимі асинхронного приймання даних. На додаток до модулів відновлення приймач має пристрій перевірки парності, логіку контролю, зсувний регістр і дворівневий буфер прийому (UDR). Приймач підтримує ті ж самі формати кадру, що й передавач, і може виявити помилку кадру, переповнювання даних і помилки парності.

Тактовий генератор виробляє всі основні тактові сигнали – як для передавача, так і для приймача. Модуль USART підтримує чотири режими синхронізації:

◊ звичайний асинхронний;

◊ асинхронний з подвійною швидкістю;

◊ синхронізацію від ведучого (Master) пристрою;

◊ синхронізацію від веденого (Slave) пристрою.

3.11.3.2. Формати кадру.

Кадр – це одне слово даних плюс супутні йому біти синхронізації (стартовий біт, стопові біти). Сюди ж може бути доданий біт парності, який застосовується для перевірки правильності передавання інформації.

Канал USART підтримує 30 різних варіантів формату кадру.

Будь-який допустимий формат має такі елементи:

- один стартовий біт;
- 5, 6, 7, 8, або 9 бітів даних;
- біт парності (якщо ввімкнений контроль парності);
- один або два стопові біти.

Кадр починається зі стартового біта, за яким йде молодший розряд слова даних. Потім йдуть решту інформаційних розрядів, їх може бути до дев’яти. Розряди надсилаються у стандартному форматі – від молодшого до старшого.

Якщо перевірка за паритетом увімкнена, то біт парності вставляється між старшим розрядом слова даних і стоповими бітами. Після передавання одного повного кадру канал може відразу починати передавання нового кадру. Якщо новий кадр даних не готовий, канал переходить у режим очікування. Рис. 168 ілюструє всі можливі комбінації формату кадру. Біти, номери яких розташовані в квадратних дужках, не є обов’язковими.

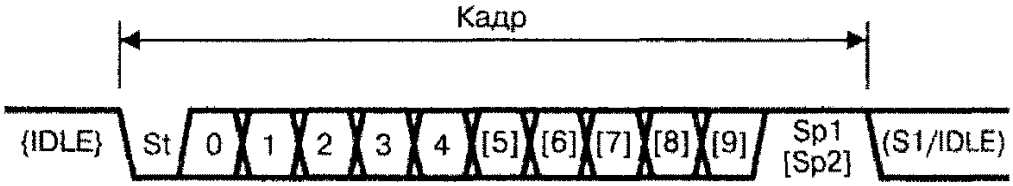


Рисунок 168. Формати кадру

Умовні позначення:

St – стартовий біт, завжди дорівнює нулю;

(n) – біт даних (n = 0 – 8);

P – біт парності (табл. 53);

Sp – стоповий біт, завжди дорівнює одиниці;

IDLE – інформація по лінії (RXD або TXD) не передається. В стані IDLE на лінії повинен бути високий логічний рівень.

Таблиця 53. Правила формування біта паритету

Паритет	Кількість одиничних бітів серед даних	Значення біта P
За непарністю	Парне	0
За непарністю	Непарне	1
За парністю	Парне	1
За парністю	Непарне	0

Формат кадру для каналу USART вибирається за допомогою розрядів UCSZ2:0, UPM1:0 і USBS регістрів UCSRB і UCSRC. Для приймача й передавача повинні бути вибрані одні й ті ж установки.

3.11.3.3. Регістри USART.

Регістр UDR.

Буфер прийому та буфер передавання даних використовують одну й ту ж адресу вводу-виводу й утворюють регістр даних USART, що носить назву UDR (USART Data Register). При записуванні в регістр UDR (рис. 169) дані записуються у відповідний регістр передавача, а при зчитуванні – читаються з регістра приймача. У випадку приймання 5-, 6- чи 7-бітних символів старші біти регістра заповнюються нулями, а при передаванні відповідні біти ігноруються.

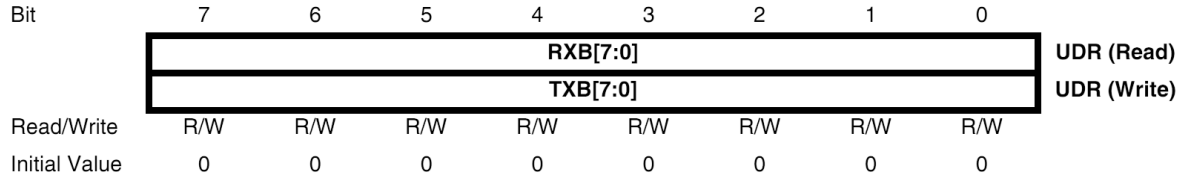


Рисунок 169. Регістр UDR

Записування в регістр UDR можливе лише при встановленні біта UDRE в регістрі UCSRA. В іншому випадку дані будуть ігноруватися передавачем UCSRA. Коли дані записуються в буфер передачі і передавач увімкнений, передавач буде завантажувати дані в регістр зсуву, коли регістр зсуву порожній, і біти у відповідному порядку передаються на лінію TxD.

Регістр приймання даних організований складніше й містить дворівневий буфер FIFO, що дозволяє зберігати до двох прийнятих байтів із послідовного порту. Регістр FIFO може змінювати свій стан у процесі виконання програми, тому не варто використовувати при роботі з регістром стану FIFO команди читання-модифікації-записування.

Регістри задавання режиму роботи й стану USART.

Для задавання режиму роботи, а також для визначення стану USART використовують три регістри: UCSRA, UCSRB, UCSRC.

Формат регістра UCSRA зображений на рис. 170.

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Рисунок 170. Формат регістра UCSRA

У регістрі UCSRA біти мають таке призначення:

Біт 7 – RXC (Receive Complete). Прийом закінчено. Біт встановлюється в «1», якщо є дані, які були прийняті послідовним портом і ще не зчитувалися. Біт скидається в нуль, якщо буфер приймача порожній і всі дані, що були прийняті, вже зчитані. Біт встановлюється і скидається апаратно й може генерувати переривання «Receive Complete interrupt».

Біт 6 – TXC: (Transmit Complete). Передавання даних завершено. Біт встановлюється в «1», якщо дані передані й немає нових даних для передавання даних у регістрі UDR. Біт TXC автоматично скидається, коли викликається відповідне переривання, або вручну за допомогою запису «0» у відповідну позицію.

Біт 5 – UDRE (USART Data Register Empty). Регістр даних USART порожній. Біт UDRE встановлюється в «1», якщо буфер передавача (UDR) готовий прийняти нові дані. При встановленні біта в одиничний стан може генеруватися відповідне переривання. Після скидання мікросхеми біт встановлюється в одиничне значення, щоб показати, що передавання даних дозволено.

Біт 4 – FE: (Frame Error). Помилка кадру. Біт встановлюється в «1», якщо при прийманні даних виникла помилка формату даних.

Біт 3 – DOR: (Data OverRun). Переповнення регістра даних. Біт встановлюється в «1», якщо виявлено переповнення регістрів даних при прийманні – тобто було прийнято більш ніж два символи й виявлено новий стартовий біт передавання. При записуванні у регістр UCSRA біт завжди потрібно скидати в нуль.

Біт 2 – PE: (Parity Error). Помилка паритету. Встановлюється в «1», якщо у режимі з контролем за парністю/непарністю виявлено помилку паритету. При записуванні в регістр UCSRA біт завжди потрібно скидати в нуль.

Біт 1 – U2X: (Double the USART Transmission Speed). Встановлення біта в «1» задає подвоєння швидкості передавання даних. Біт використовується лише при асинхронній роботі USART. При синхронному режимі роботи біт має бути в нульовому стані.

Біт 0 – MPCM: (Multi-processor Communication Mode). Режим багатопроцесорних комунікаційних систем. Якщо біт дорівнює «1», вмикається режим багатопроцесорних систем, у котрому всі байти, які не містять відповідними чином встановленого адресного біта, ігноруються.

Формат регістра UCSRB показано на рис. 171.

Bit	7	6	5	4	3	2	1	0	
	RXCIE	**TXCIE**	**UDRIE**	**RXEN**	**TXEN**	**UCSZ2**	**RXB8**	**TXB8**	**UCSRB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 171. Формат регістра UCSRB

У регістрі UCSRB біти мають таке призначення:

Біт 7 – RXCIE: (RXC Interrupt Enable). Дозвіл переривання за закінченням приймання даних. Запис «1» у біт дозволяє переривання від біта RXC регістра UCSRA.

Біт 6 – TXCIE: (TXC Interrupt Enable). Дозвіл переривання за завершенням передавання даних. Встановлення біта в «1» дозволяє переривання від біта TXC з регістра UCSRA.

Біт 5 – UDRIE: (USART Data Register Empty Interrupt Enable). Дозвіл переривання при порожньому регістрі даних USART. Встановлення біта в одиничний стан дозволяє переривання від біта UDRE.

Біт 4 – RXEN: (Receiver Enable). Дозвіл роботи приймача. Встановлення біту в «1» дозволяє роботу приймача даних USART та перемикає роботу відповідної лінії мікросхеми на режим роботи з USART. Скидання біта в «0» очищає буфер приймача, переводить у недійсний стан біти FE, DOR та PE, а також перемикає лінію RxD мікросхеми на нормальний режим роботи.

Біт 3 – TXEN: (Transmitter Enable). Дозвіл роботи передавача. Встановлення біта в «1» дозволяє роботу передавача USART та переводить лінію TxD у режим роботи з USART. Скидання біта TXEN вимикає передавач лише після передавання байтів, що мають бути передані.

Біт 2 – UCSZ2: (Character Size). Біт довжини слова, що передається. Біт використовується разом із бітами UCSZ1:0 з регістра UCSRC для задавання довжини слова, що передається.

Біт 1 – RXB8: (Receive Data Bit 8). Задає дев’ятий біт даних, що приймаються. Для вірного приймання даних біт RXB8 має бути прочитаний до читання даних з регістра UDR.

Біт 0 – TXB8: (Transmit Data Bit 8). Задає дев’ятий біт даних, що передається при роботі із 9-бітовими кодами. Для вірної роботи біт має бути записаний у регістр UDR до записування даних.

Формат регістра UCSRC показано на рис. 172.

Bit	7	6	5	4	3	2	1	0	
	URSEL	**UMSEL**	**UPM1**	**UPM0**	**USBS**	**UCSZ1**	**UCSZ0**	**UCPOL**	**UCSRC**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Рисунок 172. Формат регістра UCSRC

Регістр UCSRC має спільну адресу з регістром UBRRH. Тип регістра при записуванні визначається бітом URSEL, а при читанні визначається послідовністю читань, яка буде розглянута нижче.

У регістрі UCSRC біти мають таке призначення:

Біт 7 – URSEL: (Register Select). Біт вибору регістра. Регістри UCSRC та UBRRH суміщають єдину адресу. При записуванні даних у регістр UCSRC біт має дорівнювати «1». При читанні цей біт дорівнює «1», а решта бітів відповідають регістру UCSRC.

Біт 6 – UMSEL: (USART Mode Select). Вибір режиму роботи послідовного порту. Одиничне значення біта переводить послідовний порт у синхронний режим роботи, а нульове значення – в асинхронний режим.

Біти 5 та 4 – UPM1:0: (USART Parity Mode). Біти вибору режиму контрою за парністю чи непарністю. Режим контролю задається згідно з таблицею 54.

Таблиця 54. Режим контролю за паритетом

UPM1	UPM0	Тип контролю
0	0	Від’єднаний. Біта контролю за паритетом немає
0	1	Зарезервовано для подальших модифікацій
1	0	Увімкнений контроль за парністю. Біт паритету дорівнює «1», якщо серед інформаційних бітів парна кількість одиниць
1	1	Увімкнений контроль за непарністю. Біт паритету дорівнює «1», якщо серед інформаційних бітів непарна кількість одиниць

При ввімкненому контролі за паритетом при передаванні даних автоматично формується відповідний біт паритету, а при прийманні даних контролюється стан відповідного біта. При невідповідності біта паритету у прийнятих даних із розрахунковим для даного режиму встановлюється біт PE у регістрі UCSRA.

Біт 3 – USBS: (USART Stop Bit Select). Вибір кількості стоп-бітів. Кількість стоп-бітів визначається за таблицею 55.

Таблиця 55. Таблиця 54. Кількість стоп-бітів залежно від стану USBS

USBS	Кількість стоп-бітів
0	1 стоп-біт
1	2 стоп-біти

Біти 2 та 1 – UCSZ1:0 (USART Character Size). Біти розміру символу.

Разом із бітом UCSZ2 із регістра UCSRB визначають кількість інформаційних бітів, що передаються та приймаються через USART (табл. 56).

Біт 0 – UCPOL: (USART Clock Polarity). Полярність тактового сигналу. Біт має значення лише у синхроному режимі роботи. В асинхронному режимі роботи біт не має значення і, зазвичай, встановлюється в нульовий стан.

Полярність тактового сигналу визначається згідно з рис. 173.

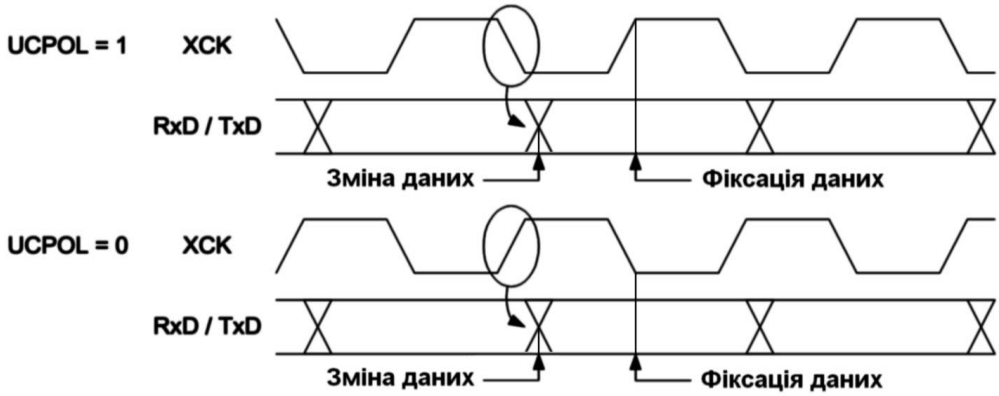


Рисунок 173. Відповідність тактового сигналу і даних залежно від стану біту UCPOL

Таблиця 56. Кількість інформаційних бітів залежно від стану бітів UCSZ2-0

UCSZ2	UCSZ1	UCSZ0	Кількість інформаційних бітів, що передаються чи приймаються
0	0	0	5 бітів
0	0	1	6 бітів
0	1	0	7 бітів
0	1	1	8 бітів
1	0	0	Зарезервовано
1	0	1	Зарезервовано
1	1	0	Зарезервовано
1	1	1	9 бітів

Регістри задавання швидкості приймання й передавання UBRRL та UBRRH.

Регістри UBRRL та UBRRH призначені для задавання швидкості передавання та приймання даних.

Формат регістрів зображений на рис. 174.

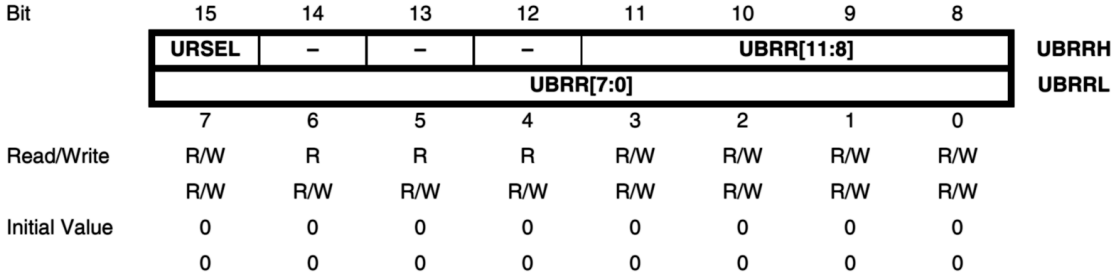


Рисунок 174. Формат регістрів UBRRL та UBRRH

Регістр UBRRH має однакову адресу із регістром UCSRC, а регістр, до якого відбувається звертання, визначається бітом URSEL.

Біт 15 – URSEL (USART Register Select) – біт визначення регістра. Одиничне значення біта вказує, що вибрано регістр UCSRC, нульове – на вибір регістра UBRRH.

Біти 14:12 – зарезервовані для подальшого використання. Для сумістності у ці біти мають бути записані нулі.

Біти 11:0 – UBRR11:0: (USART Baud Rate Register). Регістр вибору швидкості передавання та приймання даних через USART.

Залежність швидкості приймання та передавання наведена у таблиці 57.

Таблиця 57. Швидкість приймання та передавання

Режим роботи	Значення регістра залежно від швидкості	Швидкість
Асинхронний з нормальною швидкістю	$UBRR = \frac{f_{OSC}}{16\ BAUD} - 1$	$BAUD = \frac{f_{OSC}}{16\ (UBRR + 1)}$
Асинхронний з подвоєною швидкістю	$UBRR = \frac{f_{OSC}}{8\ BAUD} - 1$	$BAUD = \frac{f_{OSC}}{8\ (UBRR + 1)}$
Синхроний режим	$UBRR = \frac{f_{OSC}}{2\ BAUD} - 1$	$BAUD = \frac{f_{OSC}}{2\ (UBRR + 1)}$

Записування в регістр проводять у такій послідовності: спочатку записують старші біти у регістр UBRRH, а потім – дані у регістр UBRRL. Зміна коефіцієнта UBRR11:0 здійснюється у момент записування даних у регістр UBRRL.

Доступ до регістрів UBRRH UCSRC.

Регістри UBRRH та UCSRC мають однакову адресу. Для доступу при записуванні даних використовують старший біт регістрів UBRRH та UCSRC URSEL. Якщо біт дорівнює нулю, решта бітів регістра записується у регістр UBRRH, якщо ж біт URSEL дорівнює одиниці, змінюється значення регістра UCSRC. Наступна підпрограма ілюструє записування у два регістри:

```
; встановити у UBRRH код 2, а у регістр UCSRC код 0ff
ldi R16,0x02;
ldi R17,0x0ff
out UBRRH, R16; завантажуємо значення у регістр UBRRH
out UBRRH, R17; завантажуємо значення у регістр UCSRC
```

Читання регістрів є складнішою операцією, хоча більшості програм читання регістрів UBRRH та UCSRC не потрібне. Доступ на читання контролюється послідовністю читань.

Якщо йдуть підряд дві команди читання регістра з адресою UCSRC/ UBRRH, то перший раз буде прочитане значення регістра UBRRH, а другий раз – регістра UCSRC. У випадку, коли між командами читання буде хоча б одна команда, відбудеться читання два рази підряд лише регістра UBRRH. Переривання у процесі читання регістрів UCSRC/UBRRH мають бути заблоковані.

Приклад читання регістрів UCSRC/UBRRH (переривання заборонені):

```
USART_ReadUCSRC:
IN R16, UBRRH  ; читаємо регістр UBRRH
IN R17, UCSRC  ; і вдіразу читаємо регістр UCSRC
ret
```

3.11.3.4. Ініціалізація послідовного порту.

USART має бути ініціалізована до будь-якого передавання чи приймання даних. Процес ініціалізації складається зі встановлення швидкості передавання, встановлення формату кадру й дозволу передавача чи приймача (або обох одразу). Для дозволу переривань глобальний дозвіл переривань на час ініціалізації повинен бути скинутий.

Перед повторною ініціалізацією чи зміною режиму порту слід переконатися, що передавання даних завершена, а у буфері приймання не залишилося потрібних даних. Простий код ініціалізації наведено нижче:

```
    USART_Init:
    ; Регістри R17, R16 задають швидкість передавання
даних
    ; Регістр R18 задає стан UCSRB, а
    ; Регістр R19  - стан UCSRC
    ; задаємо швидкість передавання даних
    OUT UBRRH, R17
    OUT UBRRL, R16
    ; Задаємо формат кадра
    OUT UCSRC, R19
    ; Дозволяємо приймач і передавач
    ORI R18, (1<<RXEN)|(1<<TXEN)
    OUT UCSRB, R18
    RET
```

3.11.3.5. Процедури передавання та приймання даних.

Передавання даних через послідовний порт.

Передавання даних ініціюється завантаженням у буфер передавання UDR даних. Однак для роботи виводу TXD у режимі передавача необхідно дозволити передавач встановленням в «1» біта TXEN (Transmit Enable). Якщо попереднє передавання даних завершене, то дані відразу надійдуть у зсувний регістр, якщо попереднє передавання не завершилося – дані із регістра UDR будуть переміщені в зсувний регістр, коли регістр зсуву буде порожнім.

Наступний приклад показує просте виведення даних у послідовнй порт USART із опитуванням біта UDRE. При використанні форматів даних менше, ніж з 8 бітами даних, старші біти, записані в UDR, ігноруються. Послідовнй порт має бути проініціалізований до передавання даних. Наведена функція очікує, коли буфер передавання стане порожнім за допомогою перевірки біта UDRE перед завантаженням нових даних. Дані для передавання записані у регістрі R16.

```
USART_Transmit:
; Очікування звільнення буфера передавача
USART_Transmit_Ready:
sbis UCSRA,UDRE     ; пропусти наступну команду,
                    ;якщо буфер порожній
rjmp USART_Transmit_Ready
                    ; перейти на очікування
                    ; звільнення передавача
out UDR,r16         ; помістити дані у буфер
                    ; передавача
                    ; та почати пересилання даних
ret
```

Зрозуміло, що підпрограма передавання даних може бути переписана так, щоб, за відсутності можливості передавати дані, вихід з підпрограми відбувався миттєво зі встановленням біта перенесення.

Прийом даних через USART.

Приймач USART вмикається записом у біт (RXEN) (Receive Enable) регістра UCSRB одиниці. Коли приймач увімкнений, нормальна робота послідовного інтерфейсу припиняється і лінія переходить у режим роботи з послідовним портом. Швидкість передавання даних, режим роботи і формат кадру встановлюється однаковими для передавача і приймача. У синхронному режимі роботи при використанні синхронного керування вхід XCK використовується для передавання чи приймання тактового сигналу.

Наступний приклад показує найпростіший варіант приймання даних, що не потребує переривання, а здійснює опитування біта RXC (Receive Complete). На

відміну від підпрограми передавання, у підпрограмі приймання даних не здійснюється очікування наявності даних. Якщо даних немає, то встановлюється одиничне значення біта перенесення. У випадку наявності даних дані приймаються у регістр R16, а біт перенесення скидається. Перед використанням функції послідовний порт має бути дозволений і проініціалізований.

```
USART_Receive:
SBIS UCSRA, RXC    ; Аналіз біта RXC,
                   ; Якщо дані є, пропустити наступну
                   ; команду
RJMP USART_Not_Receive
IN R16, UDR        ; Отримати дані
CLC                ; Скинути ознаку перенесення
RET                ; Повернутися в основну програму
USART_Not_Receive:; Даних немає
SEC                ; Встановити біт перенесення
RET                ; Повернутися в основну програму
```

Зрозуміло, що підпрограма приймання даних може бути переписана так, щоб за відсутності даних відбувалося очікування надходження даних.

3.11.3.6. Багатопроцесорні системи і передавання 9-ти бітів.

У випадку взаємодії багатопроцесорних систем можливе використання режиму 9-бітового передавання з апаратною фільтрацією кадрів (рис. 175).

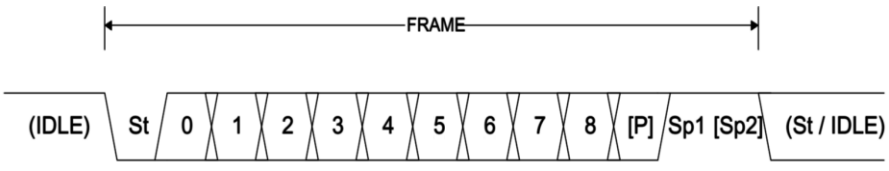


Рисунок 175. Формат кадру 9-бітового передавання

Для такого використання останній біт даних (восьмий) встановлюється в «1» при передаванні адреси та дорівнює нулю при передаванні даних. Ведені контролери, що приймають кадри у такому режимі, ігнорують усі кадри зі

скинутим останнім бітом даних. При отриманні кадру з адресою, що співпадає із заданою веденому контролеру, контролер переходить у режим приймання всіх байтів і приймає чи передає задані дані, а потім повертається у режим з апаратною фільтрацією кадрів.

3.12. АЦП мікроконтролера.

3.12.1. Загальна характеристика АЦП мікросхеми.

Мікросхема ATMega32 у своїй структурі має АЦП із такими властивостями:

◊ роздільна здатність 10 біт;

◊ інтегральна нелінійність дорівнює 0,5 ваги молодшого розряду;

◊ абсолютна точність ±2;

◊ час перетворення 13 – 260 мкс;

◊ частота перетворення до 15 кГц при максимальній роздільній здатності;

◊ кількість несиметричних входів 8;

◊ кількість диференціальних входів 7;

◊ кількість диференціальних каналів з вибором коефіцієнта підсилення 2;

◊ діапазон вхідних напруг для перетворення 0-VCC;

◊ опорна напруга зовнішня і внутрішня;

◊ режими перетворення: циклічне перетворення, із запуском за сигналом чи подією, програмний запуск.

Блок-схема АЦП ATMega32 наведена на рис. 176. Основою блока АЦП є одноканальний 10-розрядний АЦП послідовного наближення, що побудований за класичною схемою на основі 10-бітного ЦАП (10-bit DAC), компаратора із запам’ятовуванням стану (Sample & hold comparator) та відповідного регістра послідовного наближення з логікою (Conversion logic). Синхронізація перетворення здійснюється за допомогою тактової частоти процесора, що може ділитися попереднім подільником (Prescaler). Запуск та зупинка перетворення здійснюються за допомогою тригерної схеми (Trigger select).

Необхідна для перетворювача опорна напруга може задаватися або внутрішнім джерелом опорної напруги Vref напругою 2,56 В (Internal 2,56 V

reference), або входом AREF чи напругою живлення аналогової частини схеми AVCC. Вибір потрібного джерела здійснюється сигналами REFS1 та REFS0. Вхід AREF може використовуватися як вихід для контролю рівня опорної напруги або для подальшої її фільтрації.

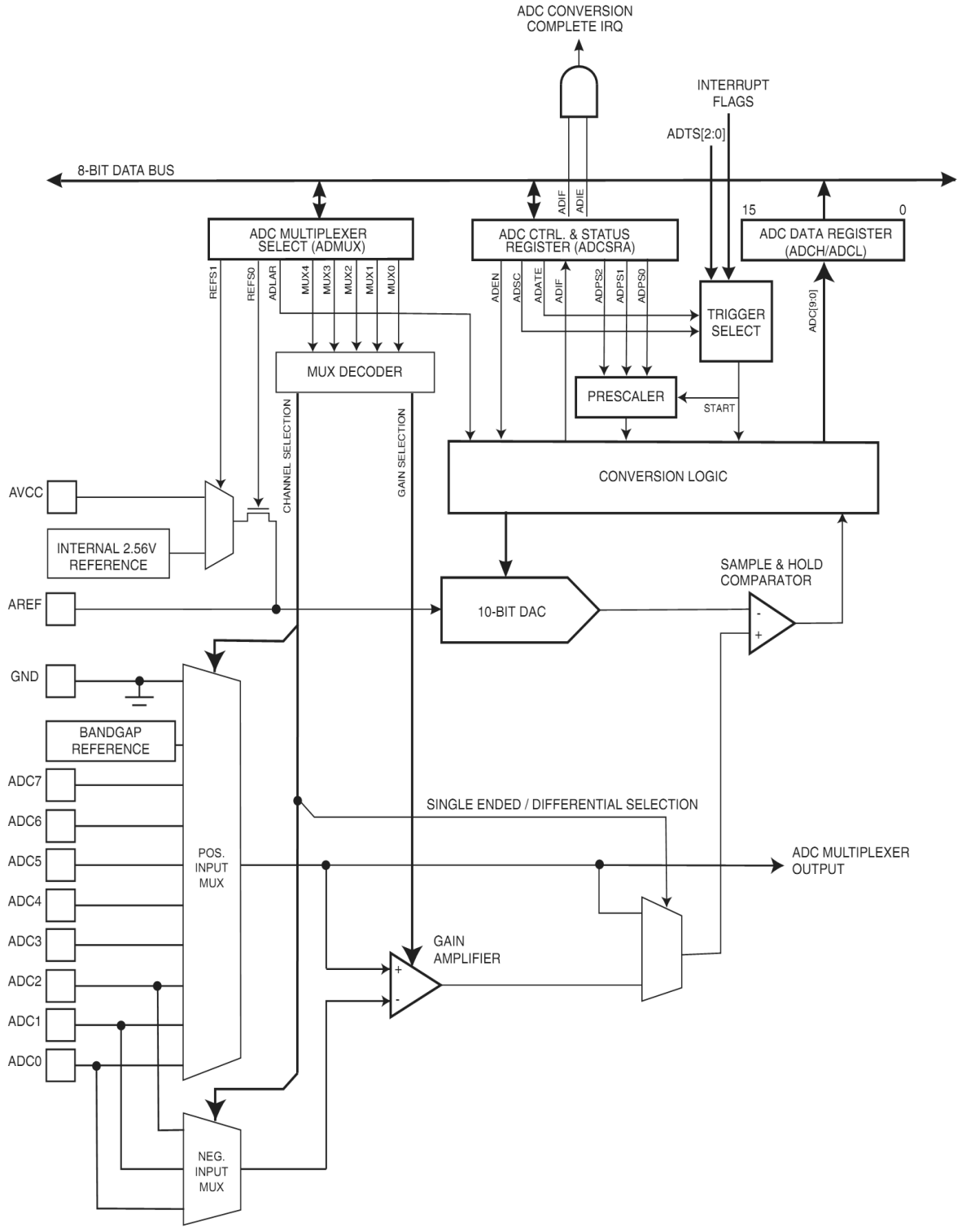


Рисунок 176. Структурна схема АЦП

Вхідний сигнал для АЦП може бути як однополярним, так і диференціальним. Джерела вхідного сигналу, що подається на контакти порту А, обираються мультиплексорами для позитивного (Pos. input mux) і негативного сигналів (Neg. input mux). АЦП підтримує режим 10 однополярних та 16 диференціальних вхідних комбінацій напруги. Серед джерел вхідного сигналу для позитивного входу можуть бути також опорна напруга забороненої зони напівпровідника (Bandgap reference) та напруга на земляному вході, що можуть бути використані для калібрування АЦП. Вибір джерела й типу сигналу здійснюється дешифратором мультиплексорів (Mux decoder). АЦП володіє можливістю внутрішнього підсилення диференціальних сигналів за допомогою підсилювача (Gain amplifier) із фіксованим коефіцієнтом підсилення, що забезпечує підсилення на 20 дБ, чи 46 дБ (тобто x10 або x200). Для контролю роботи, вибору режиму та читання результатів перетворення використовуються регістри:

◊ ADC multiplexer select (ADMUX) – регістр вибору вхідного мультиплексора;

◊ ADC ctrl. & status register (ADCSRA) – регістр контролю й статусу АЦП;

◊ ADC data register (ADCH/ADCL) – регістр даних АЦП;

◊ SFIOR Special FunctionIO Register – регістр спеціальних функцій вводу-виводу, у котрому використовуються біти ADTS 2:0.

Мінімальне значення коду на виході АЦП відповідає напрузі, яка дорівнює 0 В, а значення на 1 більше, ніж максимально можливе, представляє напругу на вході, що використовується як опорна.

3.12.2. Робота АЦП.

АЦП вмикається встановленням біта дозволу АЦП ADEN у регістрі ADCSRA. Без встановлення сигналу ADEN опорна напруга та вхідний сигнал АЦП не будуть набувати вірних значень. Таку особливість варто мати на увазі при роботі із зовнішнім фільтруючим конденсатором на вході AREF. Коли біт ADEN скинутий, АЦП повністю вимкнений і не споживає енергії. При переході у режим енергозбереження рекомендується вимкнути АЦП.

Одиничне перетворення запускається шляхом записування логічної

одиниці в біт запуску перетворення ADSC (ADC Start Conversion) регістра ADCSRA. Цей біт залишається високим, поки перетворення триває, і апаратно очищається при завершенні перетворення. Якщо вибрано інший канал передавання даних у той час, коли триває перетворення, АЦП закінчить поточне перетворення перед виконанням зміни каналу.

Проте перетворення може запускатися автоматично при різних подіях. Автоматичне ввімкнення вмикається установкою біта ADATE (ADC Auto Trigger Enable) у регістрі ADCSRA. Джерело запуску обирається шляхом встановлення бітів ADTS у регістрі SFIOR (рис. 157).

Рисунок 177. Формат регістра SFIOR

Запуск здійснюється позитивним фронтом відповідного сигналу. Джерела запуску АЦП визначаються за таблицею 58.

Таблиця 58. Джерела сигналу запуску перетворення АЦП

ADTS2:0	Джерела запуску АЦП
0 0 0	Вільний запуск. Наступне перетворення запускається за закінченням поточного (Free Running mode)
0 0 1	Аналоговий компаратор (Analog Comparator)
0 1 0	Зовнішній вхід переривань 0 (External Interrupt Request 0)
0 1 1	Таймер-лічильник 0 переривання за рівністю (Timer/Counter0 Compare Match)
1 0 0	Таймер-лічильник 0 переривання за переповненням (Timer/Counter0 Overflow)
1 0 1	Таймер-лічильник 1 переривання за рівністю B (Timer/Counter1 Compare Match B)
1 1 0	Таймер-лічильник 1 переривання за переповненням (Timer/Counter1 Overflow)
1 1 1	Таймер-лічильник 1 переривання за захопленням (Timer/Counter1 Capture Event)

Схема сигналів запуску АЦП наведена на рис. 158.

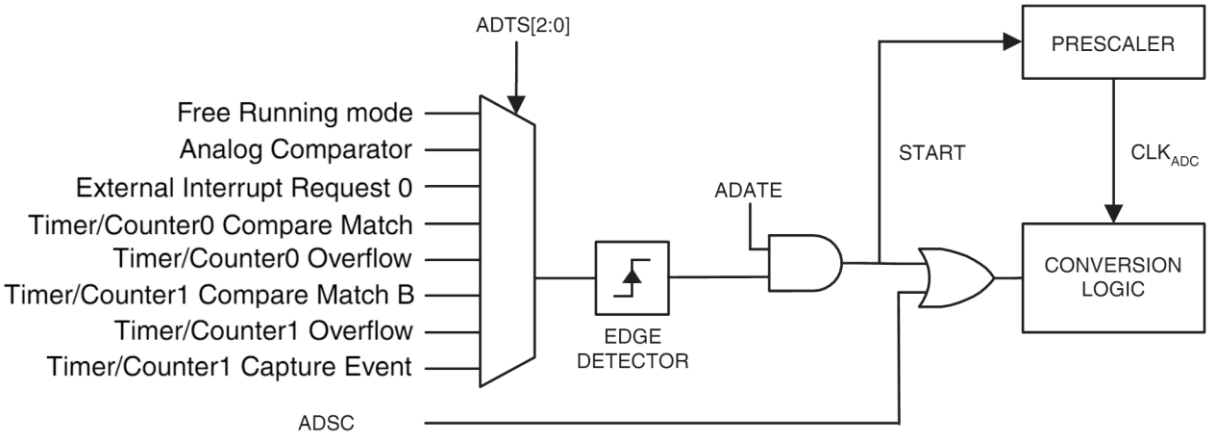


Рисунок 178. Логіка формування сигналу запуску АЦП

Зауважимо, що при запуску АЦП скидається попередній подільник АЦП, який забезпечує фіксований час перетворення. Якщо сигнал запуску залишається встановленим після завершення перетворення, нове перетворення не буде запущене. Також ігнорується повторний позитивний фронт, якщо перетворення не завершилося.

За закінченням перетворення АЦП, незалежно від того, яким методом було воно запущене, встановлюється ознака переривання за закінченням перетворення АЦП. Ознака запиту переривання від АЦП буде встановлена, навіть якщо певне переривання заборонене або скинутий біт глобального дозволу переривання у регістрі SREG. При забороні переривання перетворення може бути викликане таким чином, не викликаючи переривання. Проте, якщо ознака переривання буде використовуватися у подальшому для контролю за станом АЦП, вона має бути скинута в нульовий стан програмним шляхом.

3.12.3. Попередній подільник і частота перетворення АЦП.

АЦП у даній мікросхемі є досить повільним вузлом. За замовчуванням схема послідовного наближення вимагає тактову частоту між 50 кГц і 200 кГц для отримання максимальної роздільної здатності. Таке обмеження пов’язане з низькою швидкодією аналогового компаратора та блока ЦАП.

Якщо немає потреби у 10-бітній роздільній здатності, вхідна тактова частота АЦП може бути вищою за 200 кГц для отримання вищої частоти дискретизації. Проте у багатьох випадках при роботі на максимальній частоті потрібна висока точність перетворення. Для узгодження частоти роботи ядра мікроконтролера і модуля АЦП у модуль АЦП введено попередній подільник частоти, який знижує частоту роботи модуля і вводить її у потрібний діапазон (рис. 159). Коефіцієнт поділу попереднього подільника програмується і визначається бітами ADPS у регістрі ADCSRA.

Попередній подільник починає відлік з моменту вмикання АЦП за допомогою встановлення в «1» біту ADEN у регістрі ADCSRA. Попередній подільник продовжує працювати до тих пір, поки біт ADEN встановлений, і скидається при скиданні біта ADEN, а також при кожному старті перетворення.

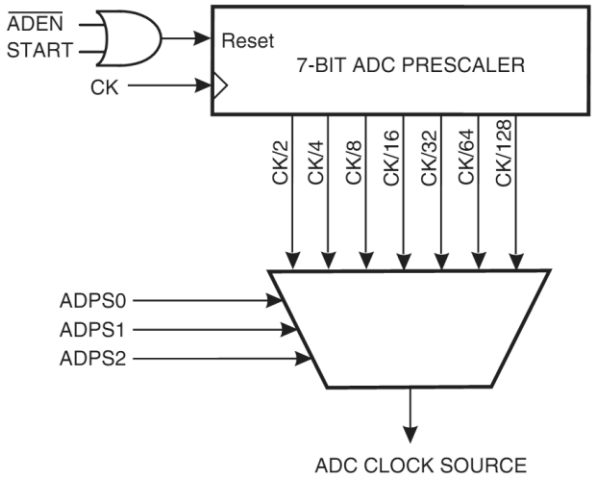


Рисунок 179. Попередній подільник АЦП

Така особливість дозволяє витримати фіксовану тривалість перетворення. Формат регістра ADCSRA зображено на рис. 180.

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 180. Регістр контролю й статусу АЦП ADCSRA

Коефіцієнт поділу попереднього подільника залежно від бітів ADPS визначається за таблицею 59. При ініціалізації перетворення встановленням біта ADSC у регістрі ADCSRA, перетворення починається за наступним зростаючим фронтом тактового сигналу АЦП. На початку перетворення АЦП виділяє певний час для захоплення рівня аналогового сигналу. Тривалість перетворення залежить від режиму роботи АЦП. Сумарний час перетворення для різних режимів роботи у періодах тактового сигналу наведено у таблиці 60.

Таблиця 59. Коефіцієнт поділу попереднього подільника АЦП

ADPS 2:0	Коефіцієнт поділу попереднього подільника АЦП
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

Таблиця 60. Час перетворення для різних режимів роботи

Режим роботи	Час захоплення сигналу	Сумарний час перетворення
Перше однократне перетворення в новому каналі, однополярний сигнал	13,5	25
Однократне перетворення, однополярний сигнал	1,5	13
Циклічне перетворення, однополярний сигнал	2	13,5
Однократне перетворення, диференціальний сигнал	1,5/2,5	13/14
Циклічне перетворення, диференціальний сигнал	2,5	14
Перше однократне перетворення в новому каналі, диференціальний сигнал	13,5	25

Нормальне перетворення займає 13 тактів. Перше перетворення після ввімкнення АЦП (ADEN в ADCSRA встановлений) займає 25 ADC тактів для ініціалізації аналогових схем. Мінімальна тривалість вибірки сигналу триває 1,5 такта АЦП. У режимі однократного перетворення диференціального сигналу тривалість вибірки АЦП може становити або 1,5, або 2,5 такта тактового сигналу АЦП, і залежить від моменту надходження сигналу запуску АЦП. Тому тривалість однократного перетворення диференціального сигналу становить 13 або 14 тактів. При переході на інший канал перша вибірка АЦП триває 13,5 такта, тому повна тривалість перетворення при переході з каналу в канал триває 25 тактів роботи АЦП.

При виборі тактової частоти АЦП слід пам’ятати, що схема внутрішнього підсилювача оптимізована для ширини смуги в 4 кГц на всіх рівнях підсилення. При передаванні вищих частот можуть виникнути спотворення. Для їх уникнення необхідно використовувати зовнішній фільтр нижніх частот. Слід зазначити, що можлива тактова частота АЦП не залежить від коефіцієнта підсилення внутрішнього підсилювача.

3.12.4. Джерела сигналів АЦП.

Зміна активного каналу (чи каналів) АЦП здійснюється бітами MUXn та REFS1:0 у регістрі ADMUX. Формат регістра ADMUX, що задає джерела опорної напруги та вхідного сигналу або сигналів, показано на рис. 181. Призначення бітів, що відповідають вибору джерел сигналу, наведено у наступній таблиці.

7	6	5	4	3	2	1	0	
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Рисунок 181. Формат регістра ADMUX

Як бачимо з таблиці 61, будь-які входи АЦП, а також сигнал опорної напруги, можуть бути обрані в якості несиметричних входів АЦП.

Таблиця 61. Джерела напруги, яка перетворюється

MUX4.0	Джерело напруги, яка перетворюється	Коефіцієнт підсилення
0 0000	Сигнал ADC0, недиференціальний вхід	1
0 0001	Сигнал ADC1, недиференціальний вхід	1
0 0010	Сигнал ADC2, недиференціальний вхід	1
0 0011	Сигнал ADC3, недиференціальний вхід	1
0 0100	Сигнал ADC4, недиференціальний вхід	1
0 0100	Сигнал ADC5, недиференціальний вхід	1
0 0110	Сигнал ADC6, недиференціальний вхід	1
0 0111	Сигнал ADC7, недиференціальний вхід	1
0 1000	Різниця сигналів ADC0 – ADC0, диференціальний вхід 1)	10
0 1001	Різниця сигналів ADC1 – ADC0, диференціальний вхід	10
0 1010	Різниця сигналів ADC0 – ADC0, диференціальний вхід 1)	200
0 1011	Різниця сигналів ADC1 – ADC0, диференціальний вхід	200
0 1100	Різниця сигналів ADC2 – ADC2, диференціальний вхід 1)	10
0 1101	Різниця сигналів ADC3 – ADC2, диференціальний вхід	10
0 1110	Різниця сигналів ADC2 – ADC2, диференціальний вхід 1)	200
0 1111	Різниця сигналів ADC3 – ADC2, диференціальний вхід	200
1 0000	Різниця сигналів ADC0 – ADC0, диференціальний вхід 1)	1
1 0001	Різниця сигналів ADC1 – ADC0, диференціальний вхід	1
1 0010	Різниця сигналів ADC2 – ADC0, диференціальний вхід	1
1 0011	Різниця сигналів ADC3 – ADC0, диференціальний вхід	1
1 0100	Різниця сигналів ADC4 – ADC2, диференціальний вхід	1
1 0101	Різниця сигналів ADC5 – ADC2, диференціальний вхід	1
1 0110	Різниця сигналів ADC6 – ADC2, диференціальний вхід	1
1 0111	Різниця сигналів ADC7 – ADC2, диференціальний вхід	1
1 1000	Різниця сигналів ADC0 – ADC2, диференціальний вхід	1
1 1001	Різниця сигналів ADC1 – ADC2, диференціальний вхід	1
1 1010	Різниця сигналів ADC2 – ADC2, диференціальний вхід 1)	1
1 1011	Різниця сигналів ADC3 – ADC2, диференціальний вхід	1
1 1100	Різниця сигналів ADC4 – ADC2, диференціальний вхід	1
1 1101	Різниця сигналів ADC5 – ADC2, диференціальний вхід	1
1 1110	1,22 В (Ширина забороненої зони напівпровідника)	1
1 1111	0 В	1
	1) Використовується для калібрування встановлення 0 2) Використовується для визначення параметрів температури	

Регістр ADMUX має тимчасовий буферний регістр, що запам'ятовує дані на час перетворення. Це гарантує, що зміна каналу й коефіцієнта підсилення, який використовується, відбувається лише в безпечний момент. Після початку перетворення зміна каналу й коефіцієнта підсилення блокується для забезпечення достатнього часу вибірки АЦП. Отже, зміна стану регістра ADMUX у процесі перетворення не впливає на поточний цикл перетворення і вступає в силу лише після його завершення. Проте не варто відразу змінювати стан регістра ADMUX після встановлення біта запуску АЦП (біта ADSC), так як перетворення починається лише наступного тактового імпульсу АЦП після встановлення біта ADSC, і тактовий імпульс може надійти пізніше, ніж буде записане нове значення в регістрі ADMUX.

У випадку використання запуску АЦП за сигналом, точний час ініціюючої події може бути неоднозначним. Особливу увагу слід звернути при оновленні регістра ADMUX, щоб визначити, яке саме перетворення буде викликане новими налаштуваннями. Також варто звернути увагу на можливість переривання у процесі записування в регістр ADMUX. Якщо обидва біти ADATE і ADEN дорівнюють «1», перетворення може початися в будь-який час. Якщо регістр ADMUX змінюється в цей період, користувач не може сказати, з якого каналу відбувається перетворення сигналу.

Тому часто при роботі в режимі запуску АЦП за сигналом варто зупинити АЦП, змінити налаштування і потім запустити АЦП знову. Якщо це неможливо, ADMUX можна оновлювати, якщо:

1. ADATE або ADEN скинуті.

2. Під час перетворення мінімум через один такт АЦП після події запуску.

3. Після перетворення, перш ніж ознака переривання буде використана як джерело запуску.

При оновленні ADMUX пза однієї із цих умов, нові налаштування будуть впливати на наступний цикл перетворення АЦП.

Після того, як диференціальний канал обраний, перехідний процес

може тривати цілих 125 мкс до стабілізації на новому значенні. Таким чином, перетворення не повинно бути розпочате протягом перших 125 мкс після вибору нового диференціального каналу. Крім того, результати перетворення, отримані протягом цього періоду, повинні бути відкинуті.

При зміні вибору каналу користувач повинен дотримуватися таких правил, аби гарантувати, що обрано правильний канал: у режимі одиничного перетворення завжди спочатку варто обирати канал, перш ніж почати перетворення. Вибір каналу може бути здійснено через один тактовий цикл АЦП після встановлення біта ADSC. Проте найпростіший спосіб – це чекати закінчення перетворення, перш ніж змінювати вибір каналів.

В автономному режимі завжди потрібно вибрати канал, перш ніж почати перше перетворення. Вибір каналу може бути здійснено через один тактовий цикл АЦП після записування одиниці в ADSC. Але найкраще дочекатися закінчення першого перетворення, а потім змінити вибір каналів. Оскільки наступне перетворення вже почалося автоматично, наступний результат буде відображати попередній вибір каналу. Наступне перетворення буде відображати новий вибір каналу.

При перемиканні диференціального каналу з підсиленням, перший результат перетворення може мати низьку точність. Користувач повинен ігнорувати перший результат перетворення.

3.12.5. Вибір опорної напруги.

Для роботи АЦП необхідна опорна напруга, з котрою здійснюється порівняння напруги, що перетворюється. АЦП ATMega32 може працювати як із кількома внутрішніми джерелами опорної напруги, так і з зовнішнім сигналом опорної напруги. Вибір джерела опорної напруги здійснюється бітами REFS 1:0 у регістрі ADMUX. Формат регістра ADMUX, що задає джерела опорної напруги, наведено на рис. 181, а призначення бітів, що відповідають вибору джерела опорної напруги, – у таблиці 62.

Таблиця 62. Джерела опорної напруги

REFS1	REFS0	Джерело опорної напруги
0	0	Зовнішній сигнал із лінії AREF у діапазоні 2,0 В – AVCC, внутрішнє джерело напруги Vref вимкнене
0	1	AVCC з зовнішнім конденсатором на вході AREF у якості фільтра
1	0	Зарезервовано. Фактично внутрішнє джерело напруги Vref ввімкнене, але його напруга вимкнена від виводу AREF, на котрий можна подавати зовнішній сигнал опорної напруги. З точки зору прикладного використання, цей режим нічим не відрізняється від режиму REFS1:0 = 00
1	1	Внутрішнє джерело напруги Vref з напругою 2,56 В із можливістю використання зовнішнього конденсатора на вході AREF у якості фільтра

Сигнал AVCC та внутрішня опорна напруга Vref під’єднуються до АЦП та виводу AREF через пасивний комутатор без будь-якого підсилення за потужністю. Максимальний струм лінії AREF становить 100 мкА при напрузі живлення 2,7 В і 175 мкА – при напрузі живлення 5 В.

Так як джерело внутрішньої опорної напруги Vref має високий імпеданс, то до лінії AREF у цьому режимі роботи можна під’єднувати виключно високоомні входи або фільтруючі конденсатори. І хоча сигнал AVCC має велику потужність, все одно не слід використовувати вивід AREF для використання із низькими приймачами сигналів.

Також слід бути уважним при під’єднанні зовнішнього джерела опорної напруги до виводу AREF. При використанні зовнішнього джерела опорної напруги не слід використовувати інші варіанти ввімкнення опорної напруги, так як вони будуть замкнуті на зовнішню напругу.

3.12.6. Зниження шумів перетворення.

У мікросхемі ATMega32 АЦП має можливість перетворення у сплячому режимі для зменшення шумів, індукованих ядром процесора та іншими периферійними пристроями вводу/виводу. Для використання перетворення у сплячому режимі потрібно скористатися такою процедурою:

1. Перевести АЦП у режим однократного перетворення та глобально

дозволити переривання й переривання від АЦП за закінченням перетворення. Для уникнення проблем із шумом бажано встановити виключно глобальний дозвіл переривання і дозвіл переривання від АЦП.

2. Перевести контролер у режим ADC Noise Reduction (або в режим очікування). АЦП почне перетворення, як тільки процесор буде зупинений.

3. Якщо інші переривання не надійдуть до завершення перетворення АЦП, переривання від АЦП виведе процесор із режиму очікування. Якщо ж інші переривання не заблокувати, то процесор прокинеться до закінчення перетворення в АЦП і на результат перетворення накладеться шум від роботи блоків мікросхеми.

Також варто звернути увагу на те, що мікросхема і зовнішні схеми за межами корпусу мікросхеми генерує завади, які можуть вплинути на точність аналогових вимірювань. Якщо точність перетворення має вирішальне значення, рівень шуму може бути зменшений шляхом застосування таких методів:

1. Лінії аналогових сигналів необхідно робити як можна коротшими. При цьому слід переконатися, що аналогові доріжки знаходяться над аналоговою заземленою поверхнею, і знаходяться як можна далі від високошвидкісних цифрових ліній.

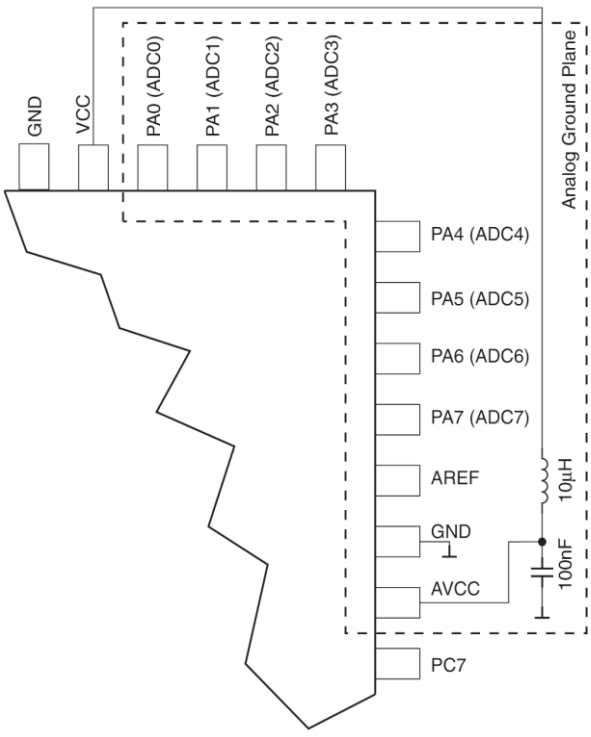


Рисунок 182. Під’єднання живлення аналогової частини мікросхеми

2. Напруга живлення аналогової частини AVCC повинна бути під'єднана до цифрової напруги живлення VCC через LC коло, як показано на рис. 182.

3. Для зменшення шуму від центрального процесора слід використовувати роботу АЦП у режимі сну.

4. Якщо будь-які лінії порту А використовуються як цифрові виходи, важливо, щоб вони не перемикалися під час перетворення АЦП.

3.12.7. Результати перетворення.

Опорна напруга для AREF визначає діапазон перетворення для АЦП. Напруга, що перетворюється, не має виходити за межі діапазону, вказаного AREF. При роботі з недиференціальним каналом, якщо напруга на вході АЦП перевищує AREF, результат перетворення дорівнюватиме 0x3FF.

Після завершення перетворення (коли біт ADIF в одиничному стані, а біт ADSC переходить у низький), результат перетворення можна знайти в регістрах результату перетворення АЦП ADCH та ADCL.

Для однополярного перетворення результат перетворення визначається за формулою:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}},$$

де V_{IN} – вхідна напруга; V_{REF} – опорна напруга; ADC – результат перетворення.

Значення 000 представляє напругу, дорівнює і менша нуля, а значення 3FF представляє напругу, що дорівнює 1023 V_{REF}/1024 й вищу.

Для диференціального перетворення результат перетворення залежить від включеного коефіцієнта підсилення і визначається за формулою:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

де V_{POS} – позитивна вхідна напруга; V_{NEG} – негативна вхідна напруга; V_{REF} – опорна напруга; GAIN – коефіцієнт підсилення; ADC – результат перетворення. Результат представляється числом зі знаком, значення 200 h (-512) відповідає мінімально можливому від'ємному значенню, а значення 1FFh (+511) – максимально можливому позитивному. Тобто старший біт результату в такому випадку є знаковим.

У процесі перетворення АЦП генерує 10-бітовий результат ADC9:0, що представляється кодом у регістрах ADCH і ADCL. За замовчуванням, результат вирівнюється вправо, проте представлення можна змінити за допомогою зміни біта ADLAR у регістрі ADMUX.

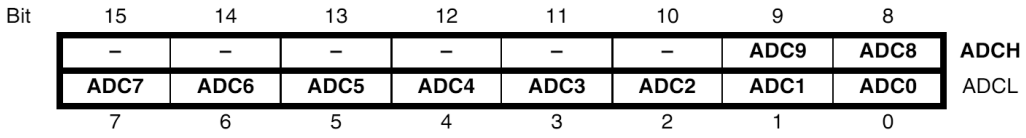
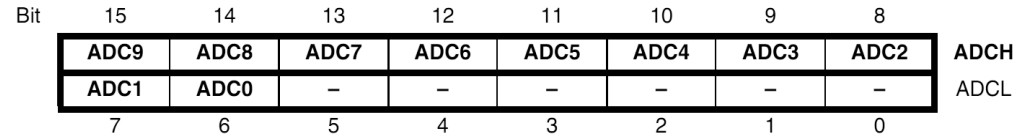
Вибір типу вирівнювання залежить від завдань, котрі вирішуються. Формат результату перетворення в регістрах ADCH ADCL зображено в таблиці 63.

У випадку, коли результат вирівняний вліво і є достатньою 8-бітова точність, досить читати регістр ADCH. Якщо ж потрібна більша точність чи результат, вирівняний вправо, необхідно читати регістри у такому порядку: спочатку прочитати регістр ADCL, а потім – регістр ADCH, щоб гарантувати, що вміст регістрів даних належить до одного й того ж перетворення.

Після читання регістра ADCL доступ АЦП до регістрів даних блокується. Це означає, що коли ADCL було прочитано, і перетворення завершується до читання ADCH, жоден регістр не буде оновлено й результат перетворення буде втрачено.

Доступ регістрів для АЦП знову вмикається після читання регістра ADCH.

Таблиця 63. Вирівнювання результату перетворення

ADLAR	Формат результату перетворення у регістрах ADCH ADCL
0	 Вирівнювання вправо
1	 Вирівнювання вліво

АЦП має своє власне переривання, яке може бути викликане при завершенні перетворення. Коли доступ до регістрів даних для АЦП заборонений між читаннями ADCH і ADCL, переривання буде викликане, навіть якщо результат перетворення був втрачений.

3.12.8. Переривання, пов’язані з АЦП.

Із АЦП пов’язане переривання «ADC Conversion Complete», котре викликається при закінченні перетворення АЦП та готовності даних. Адреса підпрограми опрацювання переривання 020 h. Для керування дозволом переривання від АЦП та визначення стану переривання використовуються біти в регістрі ADCSRA (рис. 180).

Біт ADIF (ADC Interrupt Flag) вказує на наявність запиту переривання від АЦП. Біт встановлюється в одиницю, якщо перетворення даних завершене і значення даних у регістрах даних оновлене. Біт скидається апаратно при виклику відповідного переривання, а також скидається записом одиниці у цей біт.

Біт ADIE (ADC Interrupt Enable) вказує на дозвіл переривання від АЦП. Коли цей біт встановлений в «1» і біт глобального дозволу переривань у регістрі SREG теж встановлений в «1», то при закінченні перетворення буде викликане переривання від АЦП.

3.12.9. Приклади роботи із АЦП.

Приклад 1.

Використовуючи режим безперервного перетворення, оцифрувати всі 8 вхідних каналів, результати вимірювання записати в пам’ять, починаючи з адреси $60. Тактова частота процесора 4 MHz. Коефіцієнт поділу частоти має бути 32, оскільки 4000 кГц/32 = 125 кГц і лежить у межах 50 – 200 кГц.

```
Input1:
 ldi R30, $60        ; настроюємо вказівник Z
 ldi R31, 0
 ldi R16, 0b10111101; АЦП увімкнути, Free Run – режим
 out ADCSR,  R16     ; коефіцієнт ділення 32
 clr R17             ; встановити нульовий канал
 out ADMUX, R17
 sbi ADCSR, ADSC     ; запустити АЦП
loop: inc r17        ; встановити наступний канал
 out ADMUX, R17
 sbis ADCSR, ADIF    ; готово?
 rjmp pc-1           ; перехід на попередню команду
 sbi ADCSR ADIF      ; очищаємо ADIF
 in R16, ADCL        ; читаємо і зберігаємо
 st Z+, R16          ; молодший байт
 in R16, ADCH        ; читаємо і зберігаємо
 st Z+, R16          ; старший байт
 cpi R30, $60+7      ; сьоме вимірювання
 brne free
 cbi ADCSR,    ADFR ; вимкнути режим Free Run
free: cpi R31, $60+8; останнє вимірювання
 brne loop           ; якщо ні – цикл.
 ret
```

Приклад 2.

Використовуючи Idle-режим, виміряти напругу поточного каналу. Дані зберегти в регістрах R1:R0. Тактова частота мікроконтролера 4 MHz.

```
Input2:
ldi R16, 0b10011101 ; режим однократного вимірювання
out ADCSR, R16      ; переривання дозволені
sei                 ; глобальний дозвіл переривання
ldi R16, 0b01000000 ; дозвіл Idle-режиму
out MCUCR, R16
sleep               ; увійти до Idle-режим
in R0, ADCL         ; віднімати молодший байт
in R1,ADCH          ; віднімати старший байт
ret
```

3.13. Компаратор мікроконтролера.

Для порівняння двох аналогових сигнали за рівнем у мікросхемі ATMega32 є блок аналогового компаратора. Аналоговий компаратор порівнює значення вхідної напруги на контакті AIN0 з напругою на контакті AIN1. Структура компаратора наведена на рис. 183.

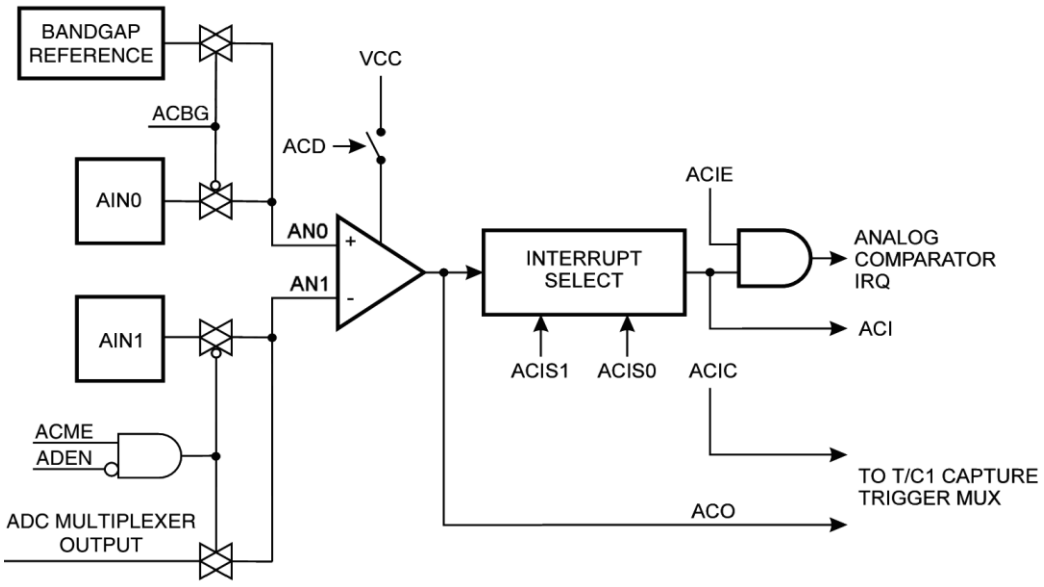


Рисунок 183. Будова аналогового компаратора

До структури компаратора входить джерело опорної напруги на основі напруги забороненої зони (Bandgap reference) зі значенням напруги 1,23 В (1,15 В – 1,35 В), два входи AIN0 та AIN1, що можуть бути заблоковані відповідними ключами, власне сам компаратор та блок вибору сигналу переривання (Interrupt select). Позитивний сигнал, що порівнюється, може надходити від входу AIN0 чи від джерела опорної напруги (Bandgap reference) залежно від стану біта ACBG. Негативний сигнал, що порівнюється, може надходити від виводу AIN1 чи від аналогового мультиплексора АЦП (ADC multiplexor output) залежно від стану ліній ACME та ADEN.

Коли напруга на позитивному вході компаратора AN0 є вищою, ніж напруга на негативному вході AN1, вихід компаратора ACO встановлюється в одиничний стан. Вихід компаратора може бути встановлений, щоб викликати функцію захоплення таймера-лічильника 1 (T/C1 capture trigger). Крім того, компаратор може викликати окреме переривання виключно для аналогового компаратора (Analog comparator IRQ). За допомогою блока вибору сигналу переривання (Interrupt select) програміст може обрати формування переривання при перевищенні вхідного сигналу на лінії AN0 над AN1, при зворотному перевищенні рівня AN1 відносно AN0 та при будь-якому перемиканні стану виходу компаратора.

3.13.1. Вхідні сигнали компаратора.

Як вже відзначалося, у якості позитивного вхідного сигналу компаратора AN0 може бути обрано або аналоговий сигнал з лінії AIN0, або опорний сигнал напруги забороненої зони (Bandgap reference 1,23 В). Вибір джерела позитивного аналогового сигналу здійснюється бітом ACBG у регістрі ACSR (Analog Comparator Control and Status Register). Одиничне значення біта вказує на ввімкнення на вхід напруги забороненої зони, нульове – на під’єднання на вхід AN0 лінії AIN0.

В якості негативного вхідного сигналу компаратора AN1 може бути обрано або аналоговий сигнал з лінії AIN1, або сигнал ADC Multiplexer output з

виходу мультиплексора позитивного аналогового сигналу АЦП (Pos. input mux) (див. рис. 156). Сигнал із позитивного входу аналогового мультиплексора буде передаватися на вхід AN1 лише при комбінації бітів: ACME = 1 та ADEN = 0. Біт ACME знаходиться в регістрі SFIOR, а біт ADEN є бітом ввімкнення/вимкнення АЦП, який знаходиться у регістрі ADCSRA (ADC Control and Status Register A). Нульове значення біта ADEN вказує на вимкнення АЦП. Отже, для використання аналогового мультиплексора разом із компаратором необхідно, щоб АЦП мікросхеми був вимкнений.

Таблиця 64. Вибір від'ємного входу аналогового компаратора

ACME	ADEN	MUX2..0	Негативний вхід AN1 аналогово компаратора
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Вибір потрібного від'ємного входу аналогового компаратора здійснюється згідно з таблицею64. Біт ACME знаходиться в регістрі SFIOR, біт ADEN – в регістрі ADCSRA, біти MUX 2..0 знаходяться в регістрі ADMUX.

3.13.2. Переривання від аналогового компаратора.

Із аналоговим компаратором пов'язане переривання «Analog Comparator Interrupt», котре викликається або при перевищенні рівня сигналу AN0 над AN1, або при зворотному перевищенні, або при будь-якій зміні співвідношення між сигналами AN0 над AN1. Адреса підпрограми опрацювання переривання 024 h.

Для керування дозволом переривання від компаратора та визначення стану переривання використовуються біти у регістрі ACSR (Analog Comparator Control and Status Register).

Біт ACI (Analog Comparator Interrupt Flag) вказує на наявність запиту переривання від аналогового компаратора. Біт встановлюється в одиницю, якщо запрограмована подія, що задється бітами ACIS1 та ACIS0, відбулася. Біт скидається апаратно при виклику відповідного переривання, а також може бути скинутий записом одиниці у цей біт.

Біт ACIE (Analog Comparator Interrupt Enable) вказує на дозвіл переривання від компаратора. Коли цей біт встановлений в «1» і біт глобального дозволу переривань у регістрі SREG теж встановлений в «1». При запрограмованій події буде відбуватися переривання від компаратора.

Біти ACIS1 та ACIS0 (Analog Comparator Interrupt Mode Select) задають тип події, при котрій відбувається запит переривання. При зміні стану бітів переривання від компаратора мають бути заблоковані, інакше може виникнути запит переривання у момент зміни стану бітів ACIS1 та ACIS0. Події, при котрих викликається переривання від аналогового компаратора, занесені у таблицю 65.

Таблиця 65. Події, при котрих викликається переривання від компаратора

ACIS1:0	Подія, при котрій викликається переривання
0 0	Будь-яка зміна співвідношення між входами AN0 та AN1
0 1	Зарезервовано
1 0	Сигнал на лінії AN1 перевищив стан лінії AN0
1 1	Сигнал на лінії AN0 перевищив стан лінії AN1

3.13.3. Регістри контролю аналогового компаратора.

Для контролю стану компаратора та задавання його режиму роботи використовують такі регістри:

◊ ACSR (Analog Comparator Control and Status Register) – регістр контролю та статусу аналогового компаратора;

◊ ADC multiplexer select (ADMUX) – регістр вибору вхідного мультиплексора, що може використовуватися для вибору негативного входу аналогового компаратора;

◊ ADC ctrl. & status register (ADCSRA) – регістр контролю й статусу АЦП, із котрого може використовуватися біт ADEN для вибору негативного входу аналогового компаратора;

◊ SFIOR Special FunctionIO Register – регістр спеціальних функцій вводу-виводу, у котрому використовується біт ACME.

Формат регістра SFIOR наведений на рисунку 184.

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 184. Формат регістра SFIOR

Із компаратором пов’язаний біт ACME: (Analog Comparator Multiplexer Enable) – дозвіл роботи аналогового мультиплексора з аналоговим компаратором. Коли біт встановлений в «1» та АЦП вимкнений (біт ADEN у регістрі ADCSRA дорівнює «0»), мультиплексор вибирає сигнал на потрібний негативний вхід AN1 для аналогового компаратора з ліній AD0:7. Коли біт встановлений у нуль, на негативний вхід аналогового компаратора надходить сигнал з лінії AIN1.

Формат регістра ACSR (Analog Comparator Control and Status Register) – регістр контролю й статусу аналогового компаратора наведений на рис. 185.

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

Рисунок 185. Формат регістра ACSR

У регістрі біти мають таке призначення:

Біт 7 – ACD (Analog Comparator Disable) – увімкнення та вимкнення аналогового компаратора. Встановлення біта в одиничний стан вимикає аналоговий компаратор, встановлення біта в нульовий стан вмикає його. Зміна біта ACD потребує також заборону переривання від аналогового компаратора, інакше можлива генерація переривань при зміні стану біта ACD.

Біт 6 – ACBG (Analog Comparator Bandgap Select) – вибору джерела позитивного сигналу для аналогового компаратора. Одиничне значення біта вказує на ввімкнення на вхід напруги забороненої зони (напруга 1,23 В), нульове – на під'єднання на вхід AN0 лінії AIN0.

Біт 5 – ACO (Analog Comparator Output) – виходу аналогового компаратора. Одиничне значення біта вказує, що сигнал на вході AN0 перевищує значення AN1, а нульове значення вказує, що сигнал на вході AN1 перевищує значення AN0. Стан біта регістра синхронізується з сигналом на виході компаратора із затримкою в 1–2 такти тактової частоти процесора.

Біт 4 – ACI (Analog Comparator Interrupt Flag) – переривання від аналогового компаратора. Біт вказує на наявнсть запиту переривання від аналогового компаратора. Біт встановлюється в одиницю, якщо відбулася подія, задана бітами ACIS1 та ACIS0. Біт скидається апаратно при виклику відповідного переривання, а також може бути скинутий записом одиниці у цей біт.

Біт 3 – ACIE (Analog Comparator Interrupt Enable) – вказує на дозвіл переривання від компаратора. Коли цей біт встановлений в «1» і біт глобального дозволу переривань у регістрі SREG теж встановлений в «1», при запрограмованій події буде відбуватися переривання від компаратора.

Біт 2 – ACIC (Analog Comparator Input Capture Enable) – біт дозволу захоплення від аналогового компаратора. Коли біт знаходиться в одиничному стані, вмикається функція захоплення у таймері-лічильнику 1, а вихід компаратора під'єднується до відповідного входу таймера-лічильника 1. Скидання біта в нульовий стан розриває зв'язок між таймером-лічильником 1 та аналоговим компаратором.

Біти 1 та 0 – ACIS1 та ACIS0 (Analog Comparator Interrupt Mode Select) задають тип події, при котрій відбувається запит переривання. При зміні стану бітів

переривання від компаратора мають бути заблоковані, інакше може виникнути запит переривання у момент зміни стану бітів ACIS1 та ACIS0. Події, при котрих викликається переривання від аналогового компаратора, занесені у таблицю 65.

Формат регістрів ADCSRA та ADMUX наведений на рис. 180 та 181 відповідно.

3.14. Проектування пристроїв на основі мікроконтролерів AVR.

3.14.1. Живлення мікроконтролерів (МК).

Напруга живлення у МК Atmel AVR різниться від 1,8 до 5 В, залежно від серії і моделі. Всі AVR можуть працювати від 5 В.

Плюс напруги живлення зазвичай позначається як Vcc. Нульовий вивід (земля, корпус) позначають GND.

Напруга живлення складає

2.7 – 5.5V в ATmega32L

4.5 – 5.5V ATmega32

Типова швидкодія

ATmega32L: 0 – 8 MHz

ATmega32: 0 – 16 MHz

Серед інших контролерів є особливі низьковольтні серії (наприклад, ATtiny2313V), із значно нжчою напругою живлення .

Деякі мікросхеми мають декілька входів живлення, тобі на всі входи Vcc треба подати напругу живлення, а входи GND треба заземлити. Велика кількість виводів зроблена з метою рівномірного розподілу струму при подічі струму на кристалу. Підключати потрібно всі виводи Vcc і GND.

Окремі питання викликають лінії AGND і AVCC – це аналогова земля і живлення для АЦП. АЦП – точний вимірювач напруги, тому його бажано живити через додаткові фільтри, щоб перешкоди, які не рідкісні в колі живлення, не впливали на якість вимірювання. З цією метою в точних схемах проводять поділ землі на цифрову й аналогову (вони повинні бути з’єднані тільки в одній точці), а на AVCC подається напруга через фільтруючий дросель. Якщо не використовувати АЦП або не робити точні вимірювання, то

допустимо на AVCC подати ті ж 5 В, що і на Vcc, а AGND з'єднати з GND. Залишати непідключені виводі AVCC якщо АЦП не потрібно не можна, так як лінії AVCC можуть живити і інші блоки, наприклад у ATmega32 від них живиться також порт А.

Проста схема підключення мікроконтролера AVR наведена на рис. 186:

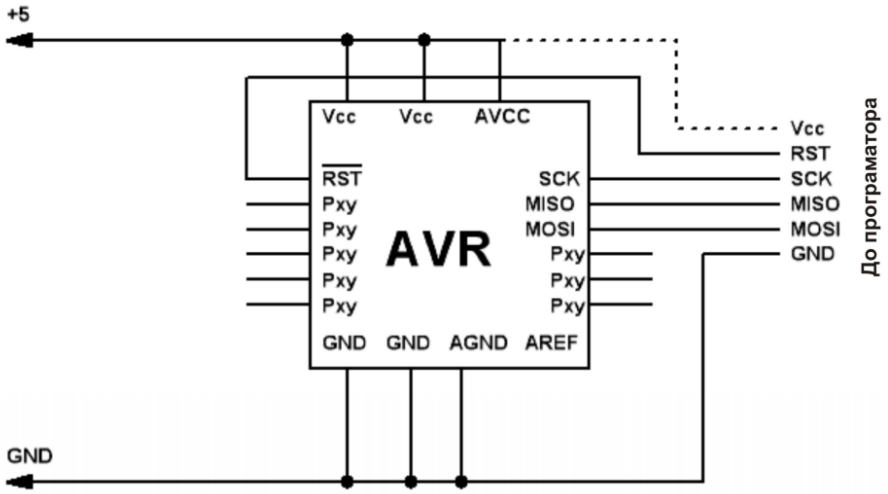


Рисунок 186. Найпростіша схема підключення мікроконтролерів AVR

Провід Vcc до програматора показаний пунктиром, він не обов'язковий. На практиці використовують іншу схему (рис. 187).

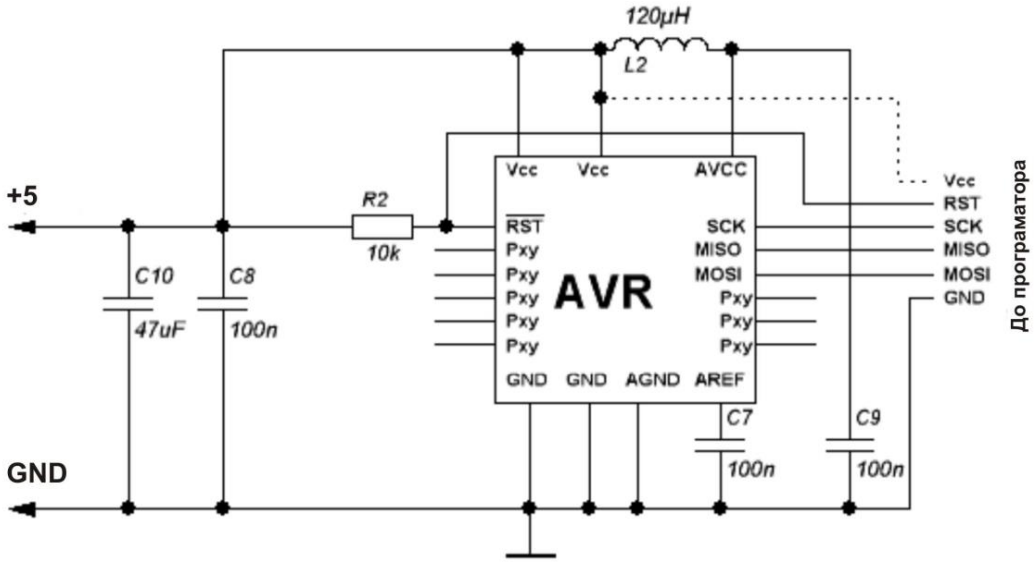


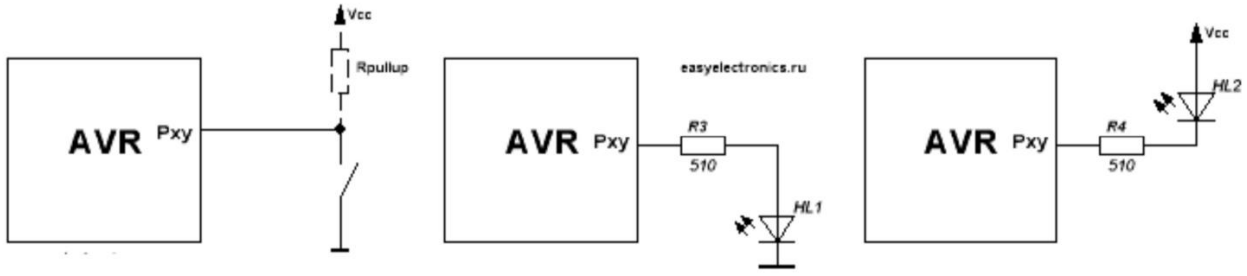
Рисунок 187. Практична схема під'єднання мікроконтролерів AVR до живлення

В порівнянні з попередньою схемою додався дросель в коло живлення AVCC, а також конденсатори. Рекомендується ставити керамічний конденсатор на 100 нф між Vcc і GND для кожної мікросхеми (а якщо у мікросхеми багато виводів живлення і землі, то між кожним живленням і кожною землею) якомога ближче до виводів живлення – він згладить короткі імпульсні перешкоди в шині живлення, які викликані роботою цифрових схем. Конденсатор на 47 мкф у колі живлення згладить великі кидки напруги.

Конденсатор між AVcc і GND додатково забезпечить живлення на АЦП. Вхід AREF – це вхід опорної напруги АЦП. Зазвичай, використовується або внутрішнє джерело опорної напруги на 2,56 В, або напруга на AVCC. Тому на лінію AREF рекомендується ставити конденсатор, що трохи поліпшить якість опорної напруги АЦП (а від якості цієї напруги залежить адекватність показань на виході АЦП).

3.14.2. Підключення до мікроконтролера світлодіода і кнопки.

Кнопка і світлодіод підключаються натупним чином (рис. 188):



а) б) в)

Рисунок 188. Підключення до МК кнопки (а) та світлодіода (б, в)

Для кнопки треба обрати вивід вводу-виводу і підключити його через кнопку на землю (рис. 188 а). Сам же вивід потрібно запрограмувати як вхід з «підтяжкою» (DDRxy = 0, PORTxy = 1). Тоді, коли кнопка не натиснута, завдяки внутрішньому «підтягуючому» резистору на вході буде високий рівень напруги, а біт PINxy при читанні буде сприйматися як «1». Якщо кнопку

натиснути, то вхід буде заземлено, а напруга на ньому впаде до нуля і з PINxy буде вводитися «0».

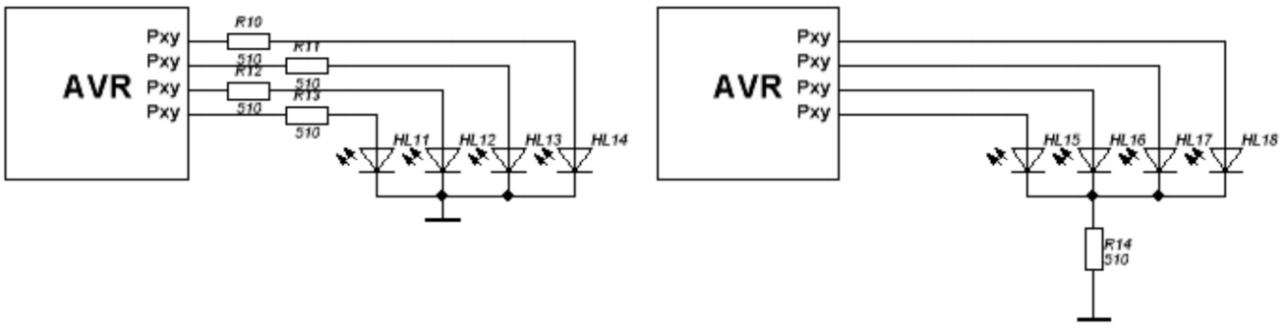
За «нулями» в бітах регістру PINxy дізнаємося, що кнопки натиснуті. Пунктиром показаний додатковий «підтягуючий» резистор. Незважаючи на те, що всередині AVR на лінію порту підключено «підтягуючий» резистор, він високоомний –100 кОм. А, значить, напругу на вході можна сприйняти як «землю» завдяки перешкоді або наведенню, що викличе помилкове спрацювання. А ще ці внутрішні «підтягуючі» резистори часто горять від наведень.

Світлодіод підключається до ліній порту двома способами:

- за схемою «порт-земля» (рис. 188 б), або
- «порт-живлення» (рис. 188 в).

У першому випадку для запалювання діода потрібно вивести в порт «1» – високий рівень напруги (приблизно дорівнює Vcc). У другому випадку – вивести в порт логічний «0» – низький рівень (близький до нуля).

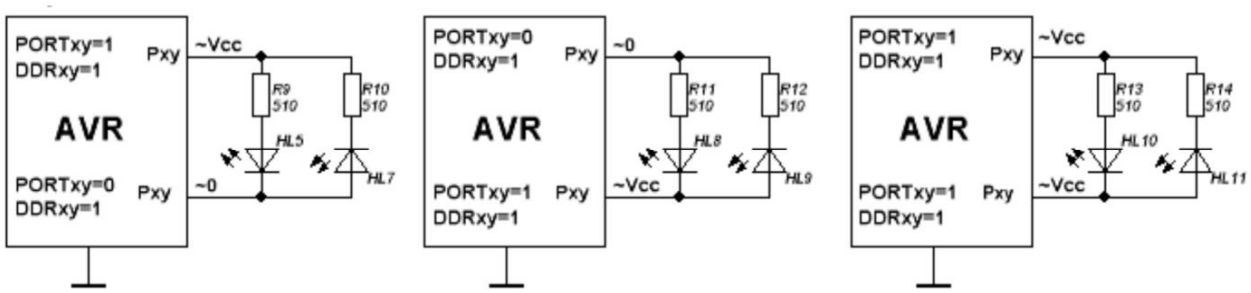
Вивід порту для роботи з світлодіодом потрібно запрограмувати на вивід (DDRxy = 1), і тоді, залежно від значення в PORTxy, на лінії виводу буде або високий, або низький рівень напруги. Світлодіод потрібно підключати через резистор.



а) б)

Рисунок 189. Підключення до ліній портів МК світлодіодів:
а) рекомендується, б) не бажане

Якщо потрібно підключити кілька світлодіодів, то до кожного послідовно вмикається резистор (рис. 189 а). Варіант включення світлодіодів (рис. 189 б) не дає економію виводів, зате дозволяє спростити розведення друкованої плати, однак ускладює програмне забеспечення, так як для рівномірного свічення світлодіодів необхідно буде застосовувати динамічну індикацію.



а) б) в)

Рисунок 190. Підключення світлодіодів за схемою «порт-порт»:
а) світиться лівий світлодіод,
б) світиться правий, в) не світиться жоден

У схемі на рис. 190, виводячи на один з виводів то «0», то «1», пропускаємо струм то в один, то в інший бік. В результаті горить то один, то інший світлодіод. Для погашення обох необхідно перевести обидва виводи в якийсь один стан: «11» або «00». Два діоди відразу не запаляться, але можна зробити динамічну індикацію – якщо їх швидко перемикати, то око не помітить мерегтіння, для нього вони будуть обидва засвіченими (рис. 191). А додавши третю лінію, можна до трьох виводів під’єднати до шести світлодіодів за цим же принципом. Від резисторів можна позбавитись, якщо вивід, де має бути викокий рівень, переводити у режим вводу із підтяжкою. Тоді струм буде обмежуватись внутрішнім опором підтяжки. Однак цей метод не є рекомендованим так так опір підтяжки коливається у широких межах і є значно більшим ніж типовий опір обмежуючого резистора.

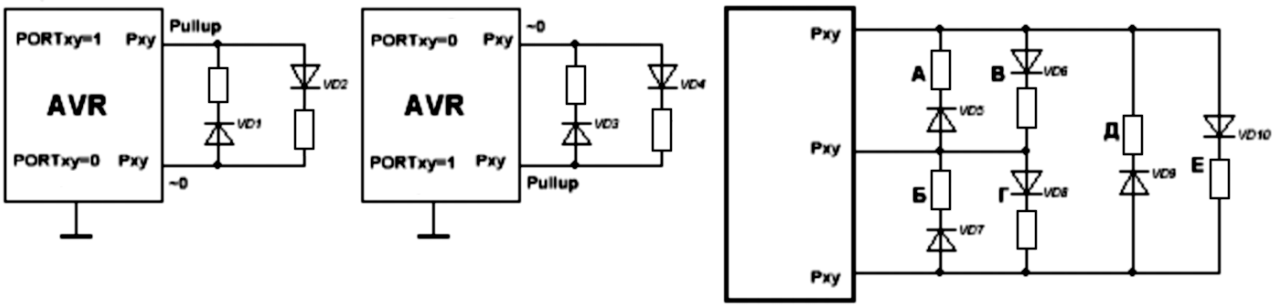


Рисунок 191. Під'єднання до виводів МК до шести світлодіодів

3.14.3. Підключення реле.

При включенні обмотки реле за схемою рис. 192 паралельно обмотці вмикається діод. При відкритому транзисторі діод включений зустрічно напрузі й струм через нього не тече. А при виключенні транзистора напруга на індуктивності обмотки буде мати протилежний напрямок і струм замкнеться через діод.

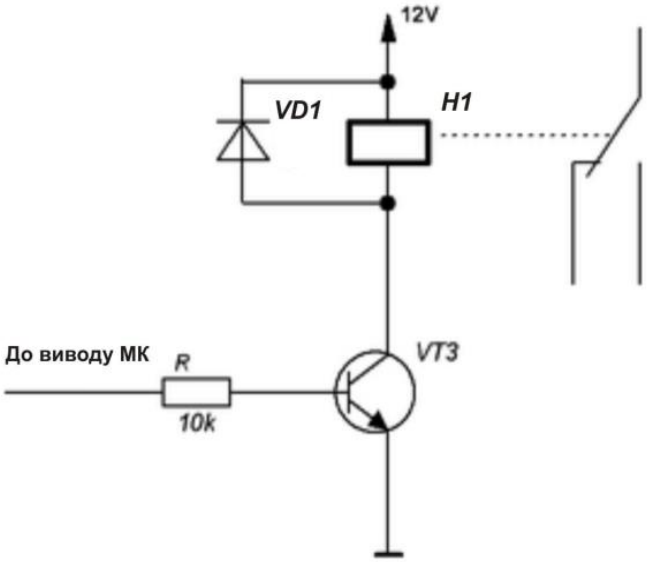


Рисунок 192. Ввімкнення діода паралельно обмотці реле

Правда, при цьому кидки напруги негативно позначаються на стабільності мережі живлення пристрою, тому є сенс біля котушок між плюсом і мінусом живлення ввімкнути електролітичний конденсатор (рис. 193). Він прийме на себе більшу частину пульсації напруги. Реле мають досить великий струм спрацювання, а струм утримання якоря менший разів у три.

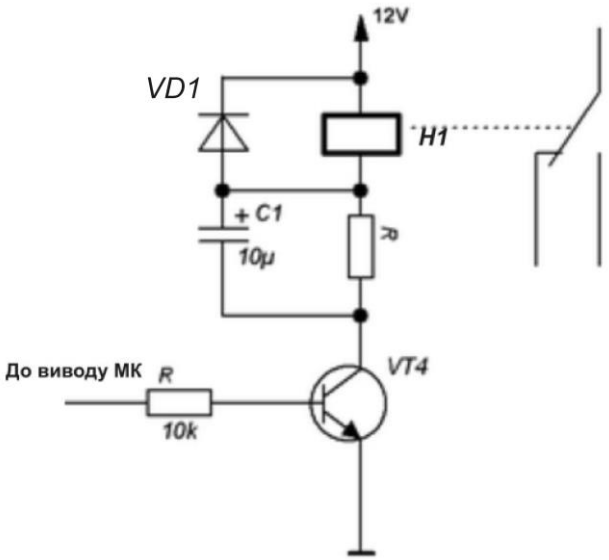


Рисунок 193. Використання конденсатора в схемі ввімкнення реле

При відкриванні транзистора конденсатор С2 ще не заряджений, а, значить, в момент його заряду він являє собою майже коротке замикання і струм через обмотку проходить без обмежень. Недовго, але цього вистачає для зрушення якоря реле з місця. Далі конденсатор зарядиться і струм через нього протікати перестане. А реле буде живитися через резистор, що обмежує струм. Резистор і конденсатор слід підбирати таким чином, щоб струм спрацювання реле був достатнім.

Після закриття транзистора конденсатор розряджається через резистор.

3.14.4. Підключення транзисторних ключів.

Польовий транзистор працює як звичайний транзистор – порівняно невеликим сигналом на затворі керуємо потужним струмом, який протікає через канал цього транзистора. Але на відміну від біполярних транзисторів управління здійснюється не струмом, а напругою. Таким чином, на затвор транзистора подається напруга, а його вхідний струм буде практично нульовим. Тому необхідна потужність управління цим транзистором буде малою, він споживає її тільки в момент перемикання, коли відбувається заряд і розряд конденсатора переходу затвор-витік.

Навантаження включається в коло стикування.

Однією з проблем стикування МОН транзистора і мікроконтролера (або цифрової схеми) є те, що для повноцінного відкривання до повного насичення цього транзистора треба подати на затвор досить велику напругу. Зазвичай, це близько 10 В, а на виході МК можна отримати максимум 5 В.

Тут варіантів три:

- на малопотужному біполярному транзисторі (VT1) зібрати ключ, який подає вже значно більшу напругу на затвор МОН-транзистора (рис. 194);
- застосувати спеціальну мікросхему-драйвер, яка сама сформує потрібний керуючий сигнал і узгодить рівні напруги межд контролером і транзистором. Потрібно тільки пам’ятати, що є драйвери верхнього і нижнього плеча (або півмостові) (рис. 195). Вибір драйвера залежить від схеми включення навантаження і комутуючого транзистора;
- застосувати транзистор з малою напругою відкривання (із серії IRL630A або подібної), в якого відкриваюча напруга прив’язана до логічного рівня напруги.

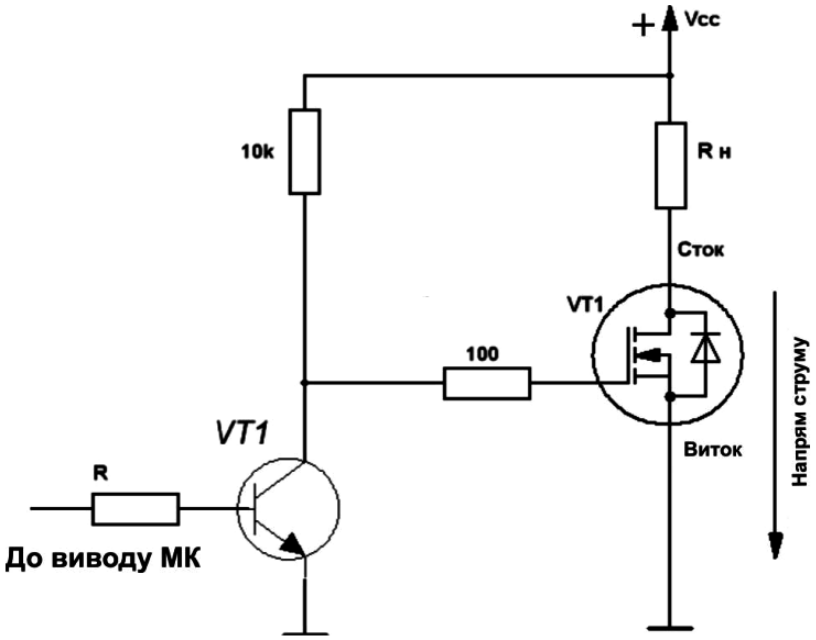


Рисунок 194. Збільшення напруги на затворі МОН-транзистора ввімкненням ключа на біполярному транзисторі

Типовий приклад драйвера – це, наприклад, мікросхема IR2117.

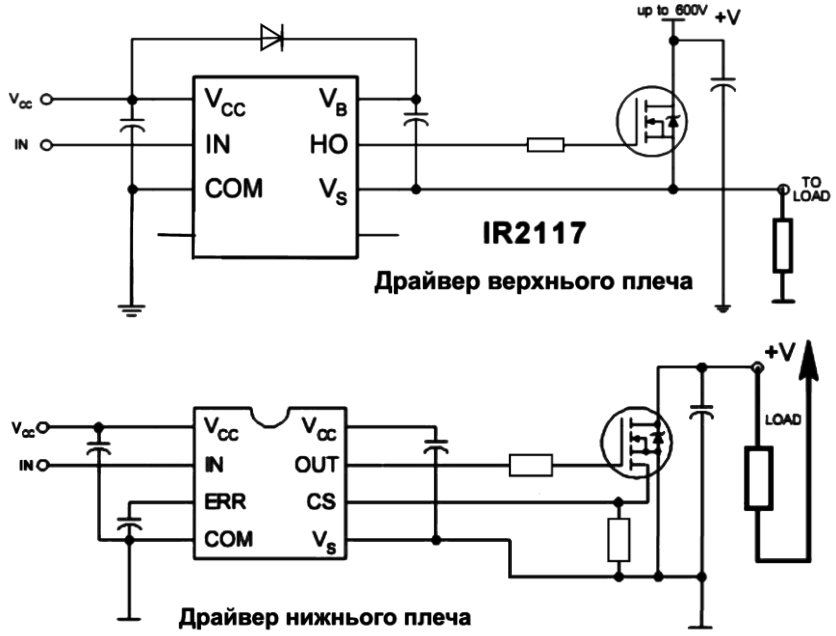


Рисунок 195. Керування МОН-транзистором за допомогою драйвера

Правильніше все ж ставити драйвер, тому що крім основних функцій формування керуючих сигналів він в якості додаткових функцій забезпечує струмовий захист, захист від пробою, перенапруги, оптимізує швидкість комутації.

3.14.5. Підключення симистора.

Якщо навантаження підключається до кола змінної напруги, тоді доцільно використовувати симистори або тиристори (рис. 196).

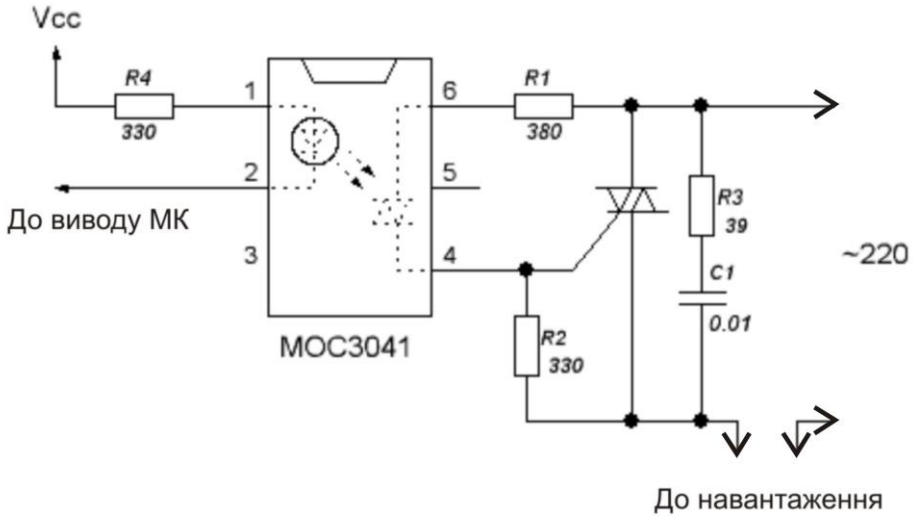


Рисунок 196. Керування комутацією навантаження в колі змінної напруги через симистор

Якщо на керуючий вхід тиристору не подавати струм відкривання, то тиристор не пропустить струм навіть в прямому напрямку. Але варто подати хоч короткий імпульс, як він відразу відкривається і залишається відкритим до тих пір, поки є пряма напруга. Якщо напругу зняти або поміняти її полярність, то тиристор закриється. Полярність керуючої напруги переважно повинна збігатися з полярністю напруги на аноді тиристора.

Симистор же на позитивній півхвилі синусоїди змінної напруги пропускає струм в одному напрямку, на негативній – в протилежному. Причому, пропускає струм тільки за наявності керуючого сигналу. Якщо сигнал керування зняти, то на наступному ж періоді обидва тиристори, що входять до складу симистора, закриються і коло розімкнеться.

Але потрібно враховувати, що комутується силове високовольтне коло 220 В, а контролер низьковольтний, працює від п’яти вольт. Тому потрібно виконати гальванічну розв’язку, тобто роз’єднати потенціали різної величини, щоб між високовольтною та низьковольтною частинами не було прямого електричного з’єднання. Для цього використовується симисторний оптоелектронний драйвер (MOC3041). У самому оптодрайвері сигнал керування подається світлодіодом, а, значить, можна заживити його від виводу мікроконтролера.

3.14.6. Підключення клавіатури.

3.14.6.1. Підключення клавіатури до МК по трьох проводах з використанням зсувних регістрів.

Нехай потрібно підключити клавіатуру до контролера з обмеженою кількістю вільних виводів.

Підключення клавіатури здійснюється по трьох сигнальних провідниках.

Додаткові елементи: зсувні регістри SN74198N і кілька резисторів. Максимальна кількість кнопок обмежується лише максимально допустимим часом на сканування клавіатури. Додаванням додаткових зсувних регістрів, можна збільшити кількість кнопок до необхідного значення. Єдиним обмеженням буде лише пропорційно зростаючий час сканування клавіатури.

Зсувні регістри використовують для підключення світлодіодів, семисегментних індикаторів і т.п. при невеликій кількості виводів мікроконтролера. У цьому

випадку дані будуть передаватися не від мікроконтролера, а до нього.

Блок-схема роботи даного пристрою показана на рис. 197. Кнопки блока кнопок одним виводом підключені до землі, а іншим – до відповідного входу зсувного регістра. Зсувний регістр здійснює перетворення паралельного вхідного сигналу в послідовний вихідний. У процесі сканування клавіатури він буде отримувати одночасно вісім значень з блока кнопок і відсилати у вигляді послідовних даних у мікроконтролер.

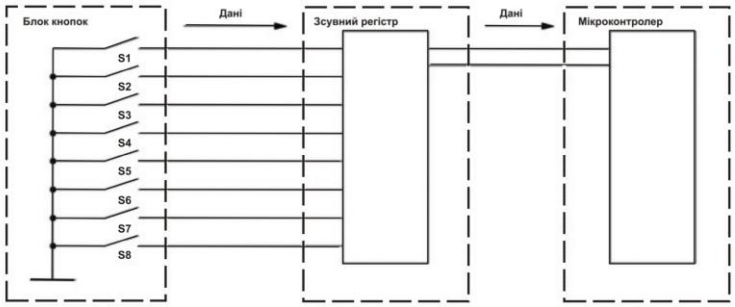


Рисунок 197. Блок-схема підключення клавіатури з використанням зсувного регістра: блок кнопок, зсувний регістр, мікроконтролер

Для цього можуть використовуватися послідовні стандарти UART або по SPI. До зсувного регістра ставляться вимоги наявності паралельний входу і послідовного виходу. Мікроконтролер керує регістром SN74198N. Він визначає, коли регістр має отримати дані з блока кнопок і передати їх до мікроконтролера. Схема електрична принципова такого під'єднання клавіатури наведена на рис. 198.

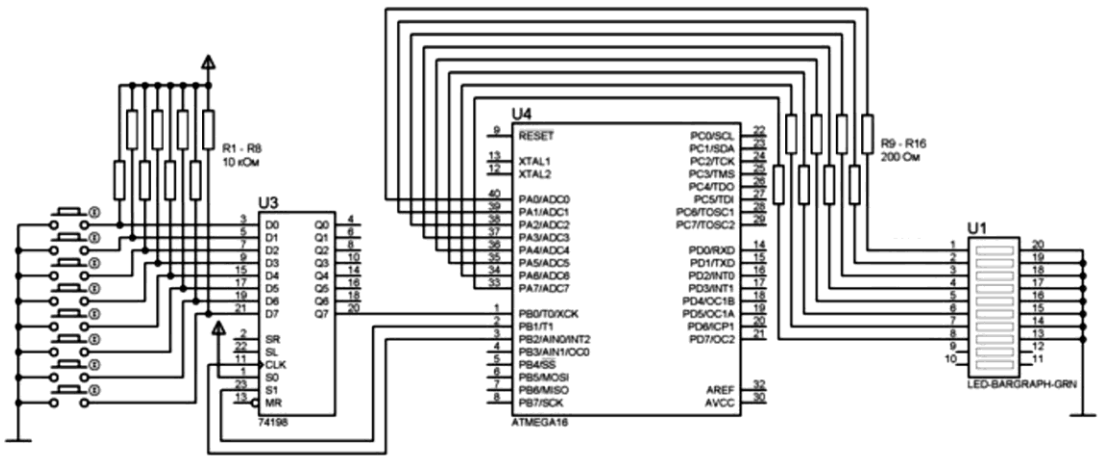


Рисунок 198. Принципова схема підключення клавіатури з використанням зсувних регістрів

Схема спрощена, показано тільки підключення клавіатури. Живлення та інші додаткові елементи обв'язки контролера не показані. Кнопки клавіатури одним виводом підключені до входу зсувного регістра. Іншим – до загального проводу. «Підтягуючі» резистори R1-R8 створюють високий логічний рівень на вході зсувного регістра, коли відповідна йому кнопка розімкнута.

Як тільки кнопка замикається, на вході зсувного регістра утворюється логічний нуль, так як він виявляється безпосередньо підключений до загального проводу. Значення резистора в 10 кОм не дає протікати великому струму, поки кнопка замкнута, і забезпечує високий логічний рівень, поки кнопка розімкнута.

Зсувний регістр має вхідний порт (D0-D7), вихідний порт (Q0-Q7) і сигнали управління (SR, SL, CLK, S0, S1, MR). Вхідний і вихідний порт можна використовувати як в паралельному режимі, так і в послідовному.

Призначення ліній регістра:

• D0-D7 – входи, на який подаються 8 сигналів від кнопок;

• Q0-Q7 – паралельні виходи. Використовується лише один вивід – Q7. Решта в нашому випадку не потрібні;

• CLK – тактовий вхід. Всі операції виконуються регістром по наростаючому фронту імпульса на цьому вході;

• S0 і S1. Цей регістр може мати як послідовний (у нашому випадку), так і паралельний вихід. Коли на обидва входи подано високий логічний рівень, можна завантажити на вихід ті дані, які в даний момент знаходяться на входах (D0-D7) в момент появи імпульсу на вході CLK. Відразу ж після цього на виходах з'являться ті ж значення, що і на входах.

Якщо ж на вхід S1 подати логічний «0», а на S0 – логічну «1», то при наявності імпульсу на вході CLK дані на вихідному порту зсунуться в бік старшого біта (біт з виходу Q0 перейде на вихід Q1, а біт, який раніше був на місці Q1, перейде на Q2 і т.д.).

Щоб не втрачати ці дані їх потрібно зчитувати і передавати в мікроконтролер. Так як біти рухаються в бік старшого розряду (Q7), то і зчитувати послідовні дані слід з нього.

3.14.6.2. Алгоритм роботи клавіатури.

1. S0 і S1 переводяться у високий рівень, далі натискаються кнопки клавіатури. На виходах Q0-Q7 фіксується необхідний байт (8 біт).

2. Скидається в «0» S1 (S0 залишається у високому рівні завжди, тому його можна безпосередньо підключити до живлення).

3. Зчитується біт з виводу Q7.

4. На вхід CLK подається імпульс. Біт, який раніше був на виводі Q7, втрачається, а на зміну йому приходить біт, який раніше знаходився на виводі Q6.

5. Повертаємося на пункт 3 і повторюємо наступні операції ще 7 разів, тому що потрібно зчитати всі вісім бітів.

Таким чином, використовуючи 3 виводи мікроконтролера, підключених до зсувного регістра (CLK, S1, Q7), опитуються всі 8 кнопок клавіатури.

3.14.7. Використання аналогового компаратора.

Майже в кожному контролері AVR є аналоговий компаратор. Він дозволяє порівнювати два аналогових сигнали.

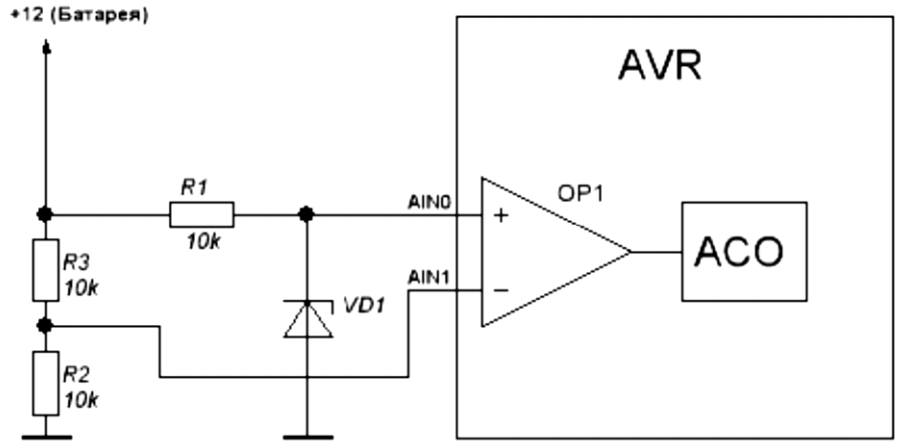


Рисунок 199. Використання аналогового компаратора

Застосувати його можна, наприклад, щоб відстежувати рівень заряду акумулятора від зменшення напруги. Схема (рис. 199) включає стабілітрон, що створює опорну напругу, яка завжди є однаковою, а напруга з резистивного дільника залежить від вхідної напруги.

Наприклад, на вхід подається 8 В. Зі стабілітрона з напругою стабілізації 3,3 В знімається завжди одна і та ж напруга – 3,3 В. А з симетричного резистивного подільника знімається половина напруги, тобто 4 В, яка більша ніж 3,3 (3,3 – 4 = -0,7 результат менший від нуля). На виході компаратора отримаємо «0».

Якщо вхідна напруга знизиться до 6 вольт, то з подільника надійде вже 3 В, а з опорного стабілітрона, як і раніше, 3,3 В. Тепер на вході компаратора різниця напруг буде більшою нуля, а значить на виході компаратора буде «1».

3.14.8. Підключення зовнішньої пам’яті.

Мікроконтролери AVR виробництва компанії Atmel мають вбудовану статичну оперативну пам’ять з довільним доступом (SRAM). Однак при розробленні деяких типів мікропроцесорних схем цього обсягу може бути недостатньо.

Наприклад, якщо програма має справу з великими обсягами даних або управляє операційною системою реального часу, то об’єм SRAM дуже швидко вичерпається. Для вирішення цієї проблеми можна використовувати інтерфейс зовнішньої пам’яті, який дозволяє розширити об’єм пам’яті SRAM до 64 КБ (рис. 200), котрий є в деяких контролерах сімейства ATMega, наприклад, в ATMega128.

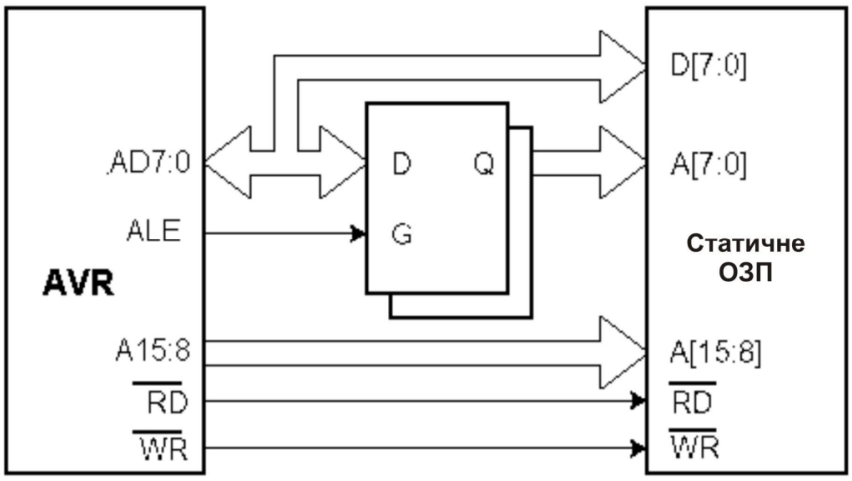


Рисунок 200. Підключення зовніщнього статичного ОЗП

Для підключення додаткової зовнішньої пам’яті можна використати схему розширення (рис. 201). Схема містить встановлену мікросхему SRAM AS6C6264, мікросхему для управління пам’яттю 74HC573 (восьмирозрядний регістр) та підключається дооднойменних виводів мікроконтролера сімейства ATMega, що має можливітсть використовувати зовнішню пам’ять даних (наприклад до ATMega128).

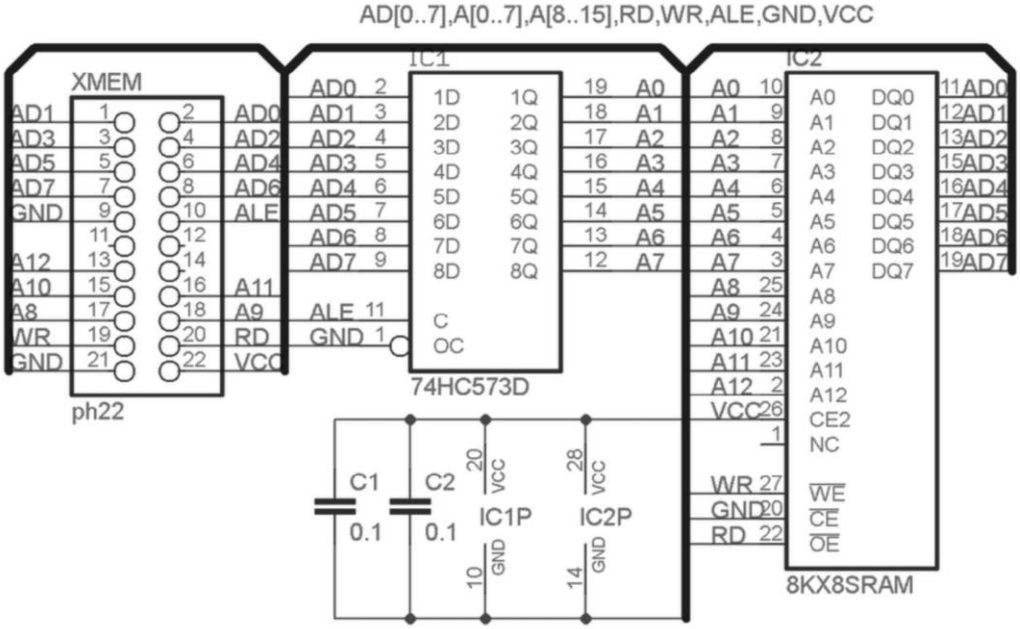


Рисунок 201. Принципова схема модуля розширення оперативної пам’яті контролера

Характеристики інтерфейсу зовнішньої пам’яті ATMega128 дозволяють використовувати його не тільки для підключення до зовнішнього статичного ОЗП або флеш-пам’яті, але й в якості інтерфейсу з зовнішніми периферійними пристроями, наприклад, ЖК-дисплеями, АЦП і ЦАП.

Його основними особливостями є:

можливість задавання чотирьох різних за тривалістю станів очікування, в т.ч. без стану очікування;

можливість встановлення різних станів очікування для різних секторів зовнішньої пам’яті (розмір сектора конфігурується);

можливість вибору кількості задіяних розрядів у старшому адресному байті;

• пристрій запам’ятовування стану шини для мінімізації споживання струму (опціонально).

Інтерфейс XRAM характеризується високою швидкодією, тому фіксація адреси повинна виконуватися на частотах понад 8 МГц при 4 В і 4 МГц при 2,7 В.

До основних параметрів, що характеризують фіксацію адреси, відносяться:

• Тривалість затримки на поширення сигналу зі входу D на вихід Q.

• Час установки даних перед тим, як G дорівнюватиме «0».

• Час утримання даних (адреси) після установки низького рівня на вході G.

Інтерфейс XRAM мікроконтролера складається з:

• AD [0..7] (виводи порту A) – мультиплексована молодша шина адреси / шина даних

• A [8..15] (виводи порту C) – старша шина адреси (з конфігурованим числом розрядів).

• ALE (вивід PG2) – строб адреси зовнішньої пам’яті.

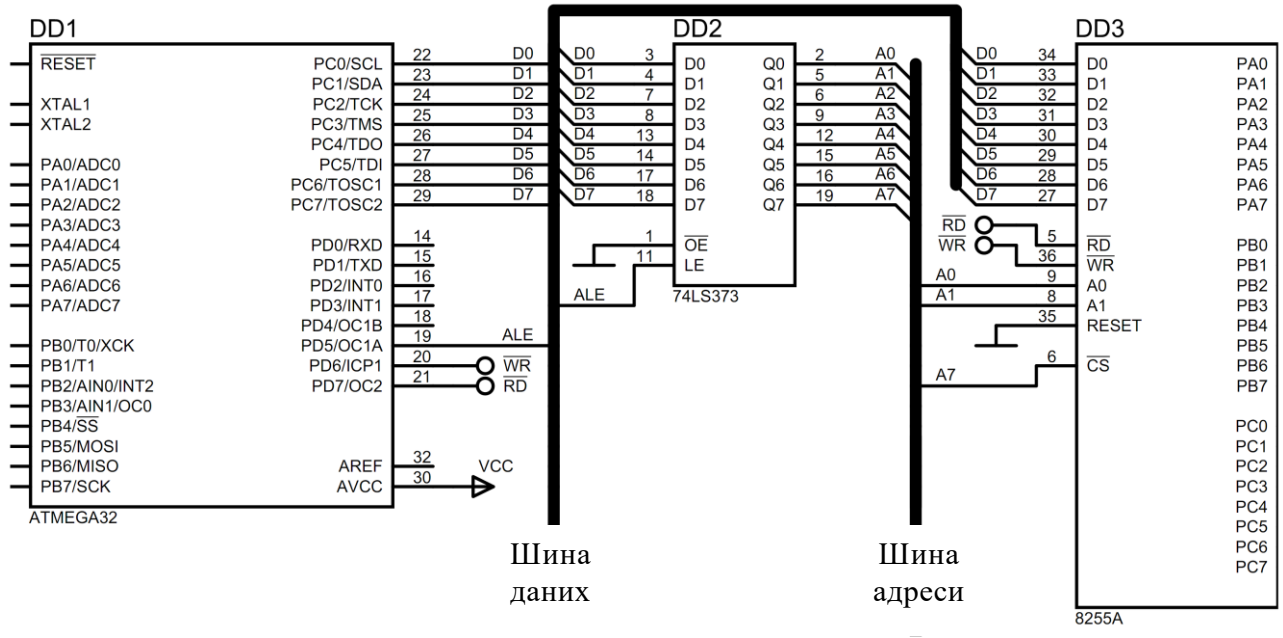
• RD (вивід PG1) – інвертований строб читання із зовнішньої пам’яті.

• WR (вивід PG0) – інв.ертований строб записування в зовнішню пам’ять.

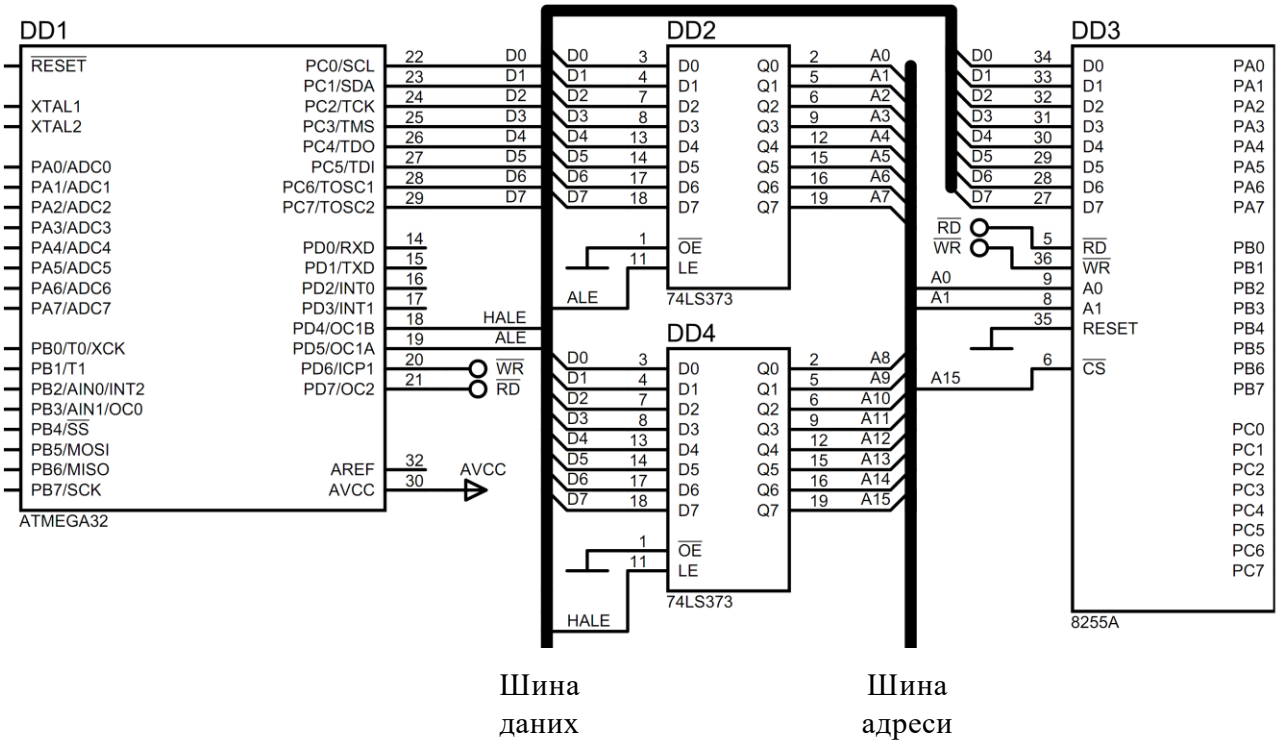
AS6C6264 – мікросхема КМОН статичної оперативної пам’яті з довільним доступом 65536 біт (8192 слова по 8 біт). Регістр-засувка використовується для роботи з мультиплексованою молодшою шиною адреси і даних.

Для доступу до комірок пам’яті необхідно вказати 16-бітову адресу до 8-бітної комірки, тому використовуємо лише 8 КБ. Тобто, використовуватися будуть тільки 13 адресних ліній.

У тих випадках коли мікроконтролер не має інтерфейсу зовнішньої пам’яті даних (як наприклад, ATMega32) шину у стилі MCS80 чи MCS85 процесорів можна створити програмним шляхом, використовуючи лінії портів як шину даних та шину керування. Причому за рахунок високої швидкодії контролера можна отримати швидкість обміну по крайній мірі не нижче ніж швидкість обміну оригінального i8085.



а)



б)

Рисунок 202. Створення шини розширення у стилі MCS80(85) із використанням мультиплексування адреси

Для зменшення кількості задіяних у інтерфейсі ліній варто повністю мультиплексувати необхідні лінії адресної шини із шиною даних. Приклади

схемних рішень для створеної шини у стилі MCS80(85) показаний на рисунку 202. Вибір ліній управління та порту даних не є принциповим. Лінії відповідних портів можуть бути замінені на інші так, щоб лінії D0-D7 під'єднувались до одного порту, а лінії керування можуть бути обрані із всіх можливих вільних ліній.

У схемі на рисунку 202 а для обміну даними спочатку за допомогою PD5 встановлюється високий рівень на лінії ALE. Далі на порт С і лінії D0-D7 видається адреса обміну. Сигнал ALE переводиться у пасивний стан і у регістрі DD2 фіксується адреса обміну. Потім у режимі запису на порт С і лінії D0-D7 видається код, що необхідно записати, подається активній рівень на лінію WR і після деякої затримки він знімається. На тому запис завершений. У режимі читання після переводу сигналу ALE у пасивний стан лінії порту даних (PC0-PC7) переводяться на ввід, і з деякою затримкою активізується сигнал читання. Через певний час, коли дані будуть готові, відбувається читання порту даних, далі сигнал читання знімається, порт переводиться на вивід, чим і завершується обмін даними.

Схема на рисунку 202 б працює аналогічно, але формуються дві частини адреси: старша і молодша, котрі передаються послідовно через загальний порт С і фіксуються за допомогою сигналів HALE та ALE, відповідно.

Приклад коду для обміну даними із зовнішніми пристроями із використанням приведених вище схем показаний далі. Для використання 16 бітової адресації необхідно використовувати функції WriteByte та ReadByte, а для 8 бітової адресації – функції WriteByte8 та ReadByte8. У всіх функціях Addr – адреса відповідної комірки, а Data – дані, що записуються.

```
#define sbit(x,PORT) ((PORT) |= (1<<x))
#define cbit(x,PORT) ((PORT) &= ~(1<<x))
#define pin(x,PIN)   ((PIN)  & (1<<x))
// Визначення сигналів обміну
// HALE - Сигнал фіксації старшої половини адреси і
//   команди роботи з ним
#define sHAle sbit(4,PORTD)
```

```
#define cHAle cbit(4,PORTD)
// ALE - Сигнал фіксації молодшої половини адреси і
//   команди роботи з ним
#define sAle sbit(5,PORTD)
#define cAle cbit(5,PORTD)
// WR - Сигнал запису і команди роботи з ним
#define sWr  sbit(6,PORTD)
#define cWr  cbit(6,PORTD)
// RD - Сигнал читання і команди роботи з ним
#define sRd  sbit(7,PORTD)
#define cRd  cbit(7,PORTD)
// I/O BUS - Шина даних і регістр напрямку передачі
#define  Out  PORTC
#define  In   PINC
#define  IOCL DDRC
//Версія для  16 бітової шини адреси
//Підпрограма виводу байта
inline void WriteByte(unsigned int Addr,
                      unsigned char Data){
   sHAle;            //Встановити лінію HALE у високий
   Out  = (Addr>>8);//Вивести старшу половину адреси
   sAle;             //Встановити  лінію ALE у високий
   cHAle;            //Встановити  лінію HALE у
                     //   низький  - зафіксувати
                     //   старшу половину адреси
   Out  = (unsigned char)(Addr&0xFF);
                     //Вивести молодшу половину адреси
   cAle;             //Встановити  лінію ALE у низький
                     //   зафіксувати молодшу половину
                     //   адреси
    Out  = Data;     //Вивести дані
    cWr;             //Скинути сигнал WR. Почати запис
    asm("nop");      //Затримка для встановлення стану
                     //   зовнішнього пристрою
    sWr;             //Встановити сигнал WR
}                    //   закінчити запис
//Підпрограма вводу байта
inline  unsigned char ReadByte(unsigned int Addr){
   sHAle;            //Встановити лінію HALE у високий
   Out  = (Addr>>8);//Вивести старшу половину адреси
   sAle;             //Встановити лінію ALE у високий
   cHAle;            //Встановити лінію HALE у низький
                     //   зафіксувати старшу половину
                     //   адреси
   Out  = (unsigned char)(Addr&0xFF);
```

```
                        //Вивести молодшу половину
                        // адреси
      cAle;             //Встановити  лінію ALE у низький
                        //   зафіксувати молодшу половину
                        //   адреси
      IOCL = 0x00;      //Перевести лінії на ввід
      asm("nop");       //Затримка для встановлення
                        //   стану лінії
      cRd;              //Скинути сигнал RD
                        //   почати читання
      asm("nop");       //Затримка для встановлення стану
                        //   зовнішнього пристрою і
                        //   видачі даних
      unsigned char inb = In;
                        //Читання виданих даних
      sRd;              //Встановлення сигналу RD кінець
                        //   читання
      IOCL = 0xFF;      //Перевести лінії знову на вивід
      return inb;       //Повернення зчитаних даних.
   }

   //Версія для 8 бітової шини адреси
   inline void WriteByte8(unsigned int Addr, unsigned
char Data){
      sAle;             //Встановити  лінію ALE у високий
      Out  = Addr;      //Вивести адресу
      cAle;             //Встановити  лінію ALE у низький
                        //   зафіксувати адресу
      Out  = Data;      //Вивести дані
      cWr;              //Скинути сигнал WR почати запис
      asm("nop");       //Затримка для встановлення стану
                        //   зовнішнього пристрою
      sWr;              //Встановити сигнал WR
   }                    //   закінчити запис
   //Підпрограма вводу байта із використанням 8 бітової
   //   адреси
   inline  unsigned char ReadByte8(unsigned int Addr){
      sAle;             //Встановити  лінію ALE у високий
      Out  = (unsigned char)(Addr&0xFF);
                        //Вивести молодшу половину адреси
      cAle;             //Встановити  лінію ALE у низький
                        ///   зафіксувати адресу
      IOCL = 0x00;      //Перевести лінії на ввід
      asm("nop");       //Затримка для встановлення
                        //   стану лінії
```

```
    cRd;                //Скинути сигнал RD почати
                        //   читання
    asm("nop");         //Затримка для встановлення стану
                        //   зовнішнього пристрою і
                        //   видачі даних
    unsigned char inb = In;
                        //Читання виданих даних
    sRd;                //Встановлення сигналу RD
                        //   кінець читання
    IOCL = 0xFF;        //Перевести лінії знову на вивід
    return inb;         //Повернення зчитаних даних.
}
```

3.14.9. Підключення світлодіодних багаторозрядних цифрових семисегментних індикаторів до мікроконтролерів AVR ATtiny/ATmega.

Кожен цифровий розряд індикатора CA56-12/CC56-12 є групою світлодіодів, з'єднаних між собою одним з виводів. Для індикатора CA загальним є анод світлодіодів, для CC – катод. Другі виводи (катоди і аноди відповідно) з'єднані між собою для відповідних сегментів усіх чотирьох цифрових розрядів. Нумерація виводів і схема відповідностей спільних виводів для вибору розряду й вибору сегмента представлені на рисунку 203.

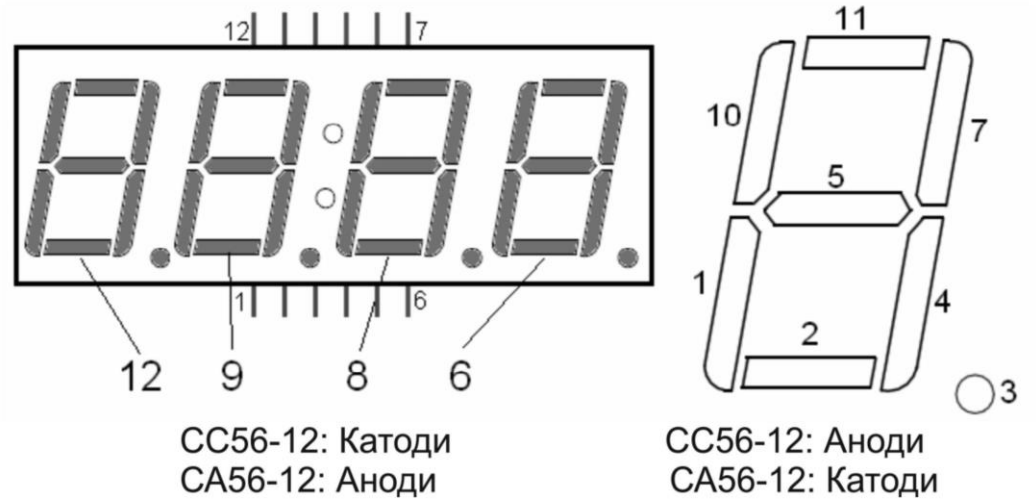


Рисунок 203. Нумерація виводів індикаторів CA56-12/CC56-12

Для того, щоб вивести інформацію на індикатор, необхідно з великою частотою поперемінно запалювати кожен з розрядів. Зручно поперемінно

подавати відповідно мінус або плюс одночасно на всі входи сегментів, які необхідно запалити в обраному розряді. Зручно підключити всі аноди на один з портів мікроконтролера, а катоди – на інший.

Оскільки світлодіоди індикатора розраховані на меншу напругу, ніж ту, що, як правило, використовується, необхідно використовувати обмежуючі струм резистори. Оскільки одночасно запалюються всі сегменти одного розряду, необхідно використовувати резистори на кожен вивід сегмента.

Розрахунок резистора можна провести за формулою

$$R = (U0 - ULED) / ILED,$$

де U0 – напруга на лінії живлення світло діодів, ULED – пряма напруга, розрахована для світлодіодів (з специфікації на індикатор); ILED – розрахунковий струм для світлодіода.

Наприклад, специфікація на індикатор CC56-12GWA визначає напругу світлодіодів, як 2,2 В при струмі 20 мА. Специфікація допускає використання більш високого значення струму (до 140 мА) в імпульсному режимі – імпульсами не більше 0,1 мс (100 мікросекунд) зі щільністю не менше 10.

Проведемо розрахунок резистора для напруги живлення 5 В і струму 20 мА при напрузі на світлодіоді 2,2 В:

$$R = (5В – 2.2В) / 0.02А = 140Ом.$$

Підійдуть резистори найближчого доступного, але не меншого номіналу, наприклад 150, 160, або 180 Ом, тому що невелике зниження струму не сильно помітно позначається на яскравості індикатора.

Залежно від варіанта підключення, слід обмежити струм також відповідно до характеристик використовуваних елементів, або, навпаки, допустити більший струм в імпульсному режимі.

Пряме підключення.

Однім із варіантів підключення є пряме підключення світлодіодів до

портів мікроконтролера (рис. 204). Через обмеження потужності, яку мікроконтролер може забезпечити на портах, робочий струм світлодіодів повинен бути обмежений. При цьому варіанті світлодіоди працюють у чверть потужності й яскравість світіння індикатора може бути недостатньою.

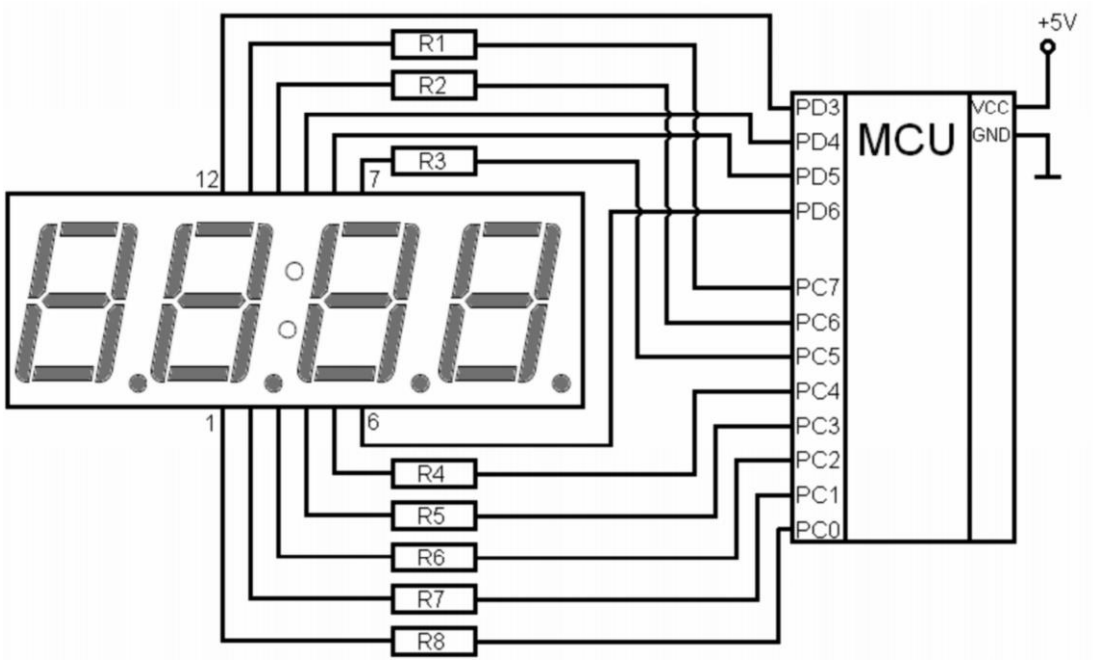


Рисунок 204. Пряме підключення семисегментного індикатора до МК AVR

Для портів ATmega і ATtiny вхідний або вихідний струм одного виводу не повинен перевищувати 40 мА. Так як вісім світлодіодів підключаються на один вивід, що вибирає розряд, то струм на кожному зі світлодіодів не повинен перевищувати 40/8 = 5 мА. Обмеження струму 40 мА на один вивід позначено як гранично допустимий. Перевищення цього струму може призвести до виходу з ладу обладнання. Для деяких моделей мікроконтролерів AVR гранично допустимий струм може бути ще менший.

Для приблизного розрахунку можна скористатися вищенаведеною формулою, прийнявши в якості значення ILED струм 0,005 А. Для точнішого розрахунку необхідно скористатися графіками залежності напруги від струму зі специфікацій на індикатор і мікроконтролер.

Конфігурація портів в ATtiny і ATmega при прямому підключенні.

Порти вводу-виводу мікроконтролерів, залежно від значень відповідних розрядів регістрів DDRx і PORTx, можуть працювати в різних режимах.

Отже, запалювання світлодіода буде проводитися наступним чином: з боку анода (для CC56 – це порт вибору сегментів, для CA56 – це порт вибору розрядів) порт повинен бути налаштований завжди на вивід: DDRx.n = 1. Виставляючи відповідний біт регістра PORTx, даний вивід буде або підключатися на мінус («землю»), або на плюс.

Таблиця 66. Програмування портів

DDRx.n	PORTx.n	Напрям	Стан виводу
0	0	Ввід	Вивід знаходиться у вільному стані (tri-state)
0	1	Ввід	Вивід підключено на VCC (лінію живлення) через вбудований підтягуючий резистор 20–50 кОм
1	0	Вивід	Вивід 0
1	1	Вивід	Вивід 1

З боку катода (для CC56 – це порт вибору розрядів, для CA56 – це порт вибору сегментів) вивід повинен бути налаштований в нуль: PORTx.n = 0.

Вибираючи відповідним бітом регістра DDRx напрямок порту, даний вивід буде або залишатися у вільному стані, або замикатися на мінус («землю»).

3.15. Програмування мікроконтролерів AVR із використанням програмного симулятора AVR Simulator IDE.

AVR Simulator IDE забезпечує користувачів мікроконтролерів Atmel зручним графічним середовищем розробки для Windows з інтегрованим симулятором (емулятором), базовим AVR-компілятором, асемблером, дизасемблером та налагоджувачем. AVR Simulator IDE підтримує 8-розрядні мікроконтролери від Atmel, архітектурні продукти AVR і сімейство 90S (вибрані моделі ATmega, ATtiny, AT90S).

AVR Simulator IDE призначений до використання для 8-бітних мікроконтролерів мікропроцесорної версії MegaAVR та tinyAVR, а також повного сімейства 90S (вибрані моделі ATmega, ATtiny, AT90S).

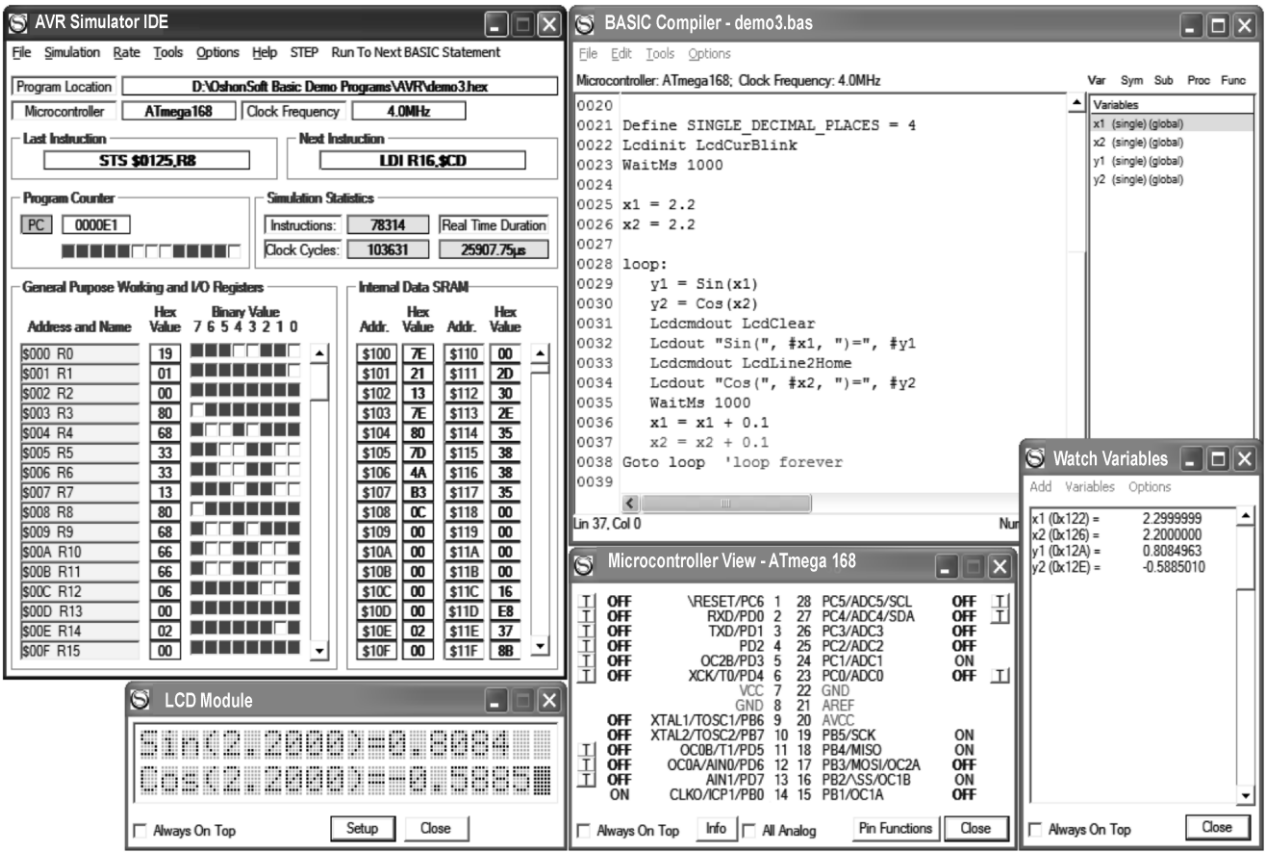


Рисунок 205. Вигляд головного вікна симулятора

У головному вікні програми (рис. 205) відображається стан внутрішніх регістрів загального користування та регістрів вводу/виводу, SRAM внутрішніх даних та лічильника програм, мнемоніка останньої виконуваної інструкції, мнемоніка наступної інструкції, що буде виконуватися, цикли та інструкції лічильника і тривалість імітації в режимі реального часу.

3.15.1. Основні можливості програми AVR Simulator IDE.

- Основний інтерфейс, що показує внутрішню архітектуру мікроконтролера.
- Редактор пам’яті програм FLASH, редактор пам’яті даних EEPROM, редактор простору SRAM.

- Інтерфейс з’єднань мікроконтролера для моделювання цифрових входів/виходів та аналогових входів.
- Змінна швидкість моделювання, моделювання статистики.
- Поточний менеджер для кодування з підтримкою точок зупинки.
- Асемблер AVR, диспетчер AVR.
- Потужний AVR Basic-компілятор з інтелектуальним основним редактором джерела (рис. 206).

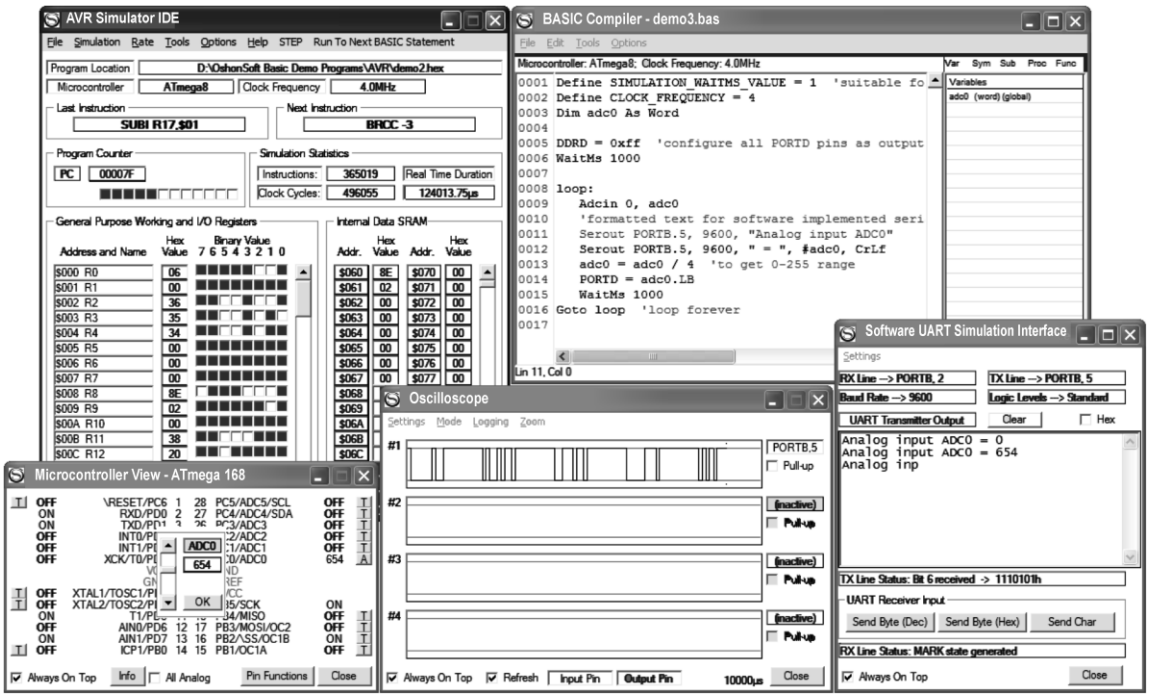


Рисунок 206. Вигляд симулятора з полем компілятора Basic, апаратними виводами контролера, осцилографом, полем послідовного інтерфейсу

Основні програмні можливості компілятора AVR.

- три основних цілі типи даних (1-бітний, 1 байт, 2 байти).
- 4-байтовий (32-розрядний) довгий цілий тип даних з 32-розрядною арифметикою.
- 4-байтовий (32-розрядний) тип даних з єдиною точністю з плаваючою точкою з одиничними точністю математичних функцій, масивів.
- тип рядків даних з великим набором пов’язаних з рядками функцій.
- всі стандартні елементи базової мови, підтримка структурованої мови (процедури та функції), підтримка керування інтерфейсу Modbus master/slave.

Реалізація карт MMC/SD/SDSC/SDHC (з підтримкою файлової системи FAT16 та підтримкою файлової системи FAT32), підтримка мовлення на високому рівні для використання внутрішньої пам'яті EEPROM, використання внутрішнього модуля A/D конвертора, використання переривань, послідовний зв'язок за допомогою внутрішнього обладнання UART, програмне забезпечення UART, зв'язок I2C з зовнішніми пристроями I2C, зв'язок послідовного периферійного інтерфейсу (SPI), інтерфейсні символьні РК-дисплеї, взаємодія графічних РК-дисплеїв з точковою матрицею 128x64, сервоприводом R/C, керуванням кроковим двигуном, пристроями 1-провідного зв'язку, DS 18S20;

- Термінал послідовного порту ПК для зв'язку з реальними пристроями, підключеними до послідовного порту.
- Інтерфейс моделювання РК-модуля для персональних РК-модулів.
- Графічний інтерфейс моделювання РК-модуля для графічних РК-модулів 128x64.
- Інтерфейс моделювання ступеневої моторної фази для візуалізації керування кроковим двигуном.
- Модуль моделювання для зовнішніх EEPROM I2C від сімейства 24 C.
- Апаратне моделювання інтерфейсу UART.
- Програмний інтерфейс моделювання UART для програмного забезпечення, що виконує UART процедуру.
- Осцилограф (з функцією масштабування) та інструменти для моделювання сигналів.
- 7-сегментний світлодіодний інтерфейс.
- DS18S20/DS18B20 цифровий термометр, інструмент моделювання.
- Пристрій моделювання Modbus (майстер/підлеглий).
- Підтримка зовнішніх модулів моделювання.
- Великі можливості програми, кольорові теми, інше.

Деякі моделі мікроконтролерів у даний час мають обмежену підтримку.

3.15.2. Призначення опцій симулятора.

Меню «Файл».

● Очистити пам’ять.

Ця команда скине симулятор до початкового стану та очистить робочі буфери пам’яті програми FLASH та пам’яті EEPROM.

● Програма завантаження.

Ця команда завантажує файл програми у буфер пам’яті програми AVR Simulator FLASH. Файл програми має бути у форматі Intel HEX. Розширення за умовчанням – HEX. Після завантаження файл програми можна швидко перезавантажити, натиснувши на його поле розташування в основному інтерфейсі програми або за допомогою комбінації клавіш SPACE.

● Зберегти пам’ять.

За допомогою цієї команди можна зберегти вміст робочого буфера пам’яті програми FLASH у файл HEX.

Меню моделювання.

● Початок.

AVR Simulator IDE входить у режим імітації й починає виконання інструкцій, починаючи з вектора скидання в пам’яті програми.

● Крок.

Ця команда вмикається, якщо вибрано швидкість моделювання за кроком. Наступна інструкція виконується при кожному натисканні клавіші F2.

● СТОП.

AVR Simulator IDE виходить з режиму симуляції та представляє інформацію про загальну кількість виконаних інструкцій, тривалість імітації та загальну тривалість симуляції в режимі реального часу в тактових циклах та μs на рівні мікроконтролера AVR.

● Виконати наступну BASIC команду.

Ця команда доступна для програм, створених інтегрованим BASIC-компілятором. Почнеться швидке моделювання, доки не буде досягнуто наступної BASIC-команди, а потім автоматично переключиться на режим крок за кроком. Комбінація клавіш F4. Якщо доступна швидкість моделювання за

кроком, то ця команда також відображатиметься як новий елемент головного меню на інтерфейсі програми.

Меню режимів роботи.

Це дозволяє користувачеві змінити швидкість моделювання. Воно також доступне під час моделювання.

● Крок за кроком.

Інтервал між послідовними інструкціями визначається за бажанням користувача. Коли симулятор знаходиться в режимі крок за кроком, можна змінити значення в усіх регістрах загального користування та регістрі вводу-виводу, лічильника програм та регістра SRAM, натиснувши відповідне поле імені або значення у інтерфейсі програми. Значення в регістрах GPWR та вводу-виводу може бути змінено альтернативно шляхом перемикання окремих бітів графічного представлення. Ця функція може бути використана також з іншими функціями моделювання. Коли вибирається цей режим симуляції, на інтерфейсі програми з’явиться новий елемент головного меню «STEP». Це дозволить легко отримати доступ до команди «Крок» у меню «Симуляція».

● Повільно.

Інтервал становить 1500 мсек.

● Нормальний.

Інтервал становить 250 мсек.

● Швидко.

Інтервал становить близько 50 мсек.

● Надзвичайно швидкий.

Інтервал дуже короткий і лінійно залежить від загальної продуктивності комп’ютера.

● Остаточний.

Головне вікно тренажера не постійно оновлюється після кожної симульованої інструкції, що значно покращує продуктивність моделювання. Інтервал оновлення можна змінити, скориставшись командою «Змінити інтервал оновлення інтервалів» у меню «Параметри».

Меню «Інструменти».

● BASIC-компілятор.

Буде відкрито вікно редактора компілятора BASIC. Детальніше інформація доступна в довідковому посібнику BASIC Compiler. Доступ до нього можна отримати в меню «Довідка» головного вікна програми або у вікні редактора BASIC.

● Вид мікроконтролера.

Ця команда відкриє вікно з вибраним типом мікроконтролера AVR. Логічні стани всіх ліній вводу-виводу відображаються графічно. Їх можна вручну змінити на вхідних лініях, натиснувши відповідні кнопки перемикання. Для того, щоб використовувати лінію як аналогову, на ній потрібно клацнути правою кнопкою миші. Після цього можна буде змінити аналогове значення, застосоване до лінії, використовуючи аналоговий слайдер. Якщо це вікно буде відкритим, його буде оновлено під час моделювання.

● Альтернативний переглядач регістрів.

Ця команда відкриває альтернативне вікно перегляду для загальних робочих регістрів і регістрів вводу-виводу, які можуть бути змінені.

● Програмований редактор пам’яті.

Це доступ до редактора буфера пам’яті програми AVR Simulator FLASH. Можна вручну ввести командний код інструкції, натиснувши на лінію відображення цільового розташування.

● Реєстр пам’яті EEPROM.

Це відкриває в AVR Simulator буферний редактор пам’яті даних EEPROM. Якщо це вікно буде відкрито, його буде оновлено під час моделювання. Значення в конкретному місці пам’яті даних можна змінити, натиснувши на нього.

● Стековий редактор систем SRAM.

Ця команда відкриє редактор для стека області внутрішніх даних SRAM. Якщо це вікно буде відкрито, його буде оновлено під час моделювання. Вона також може змінювати значення покажчика стека та значення, що зберігаються в області стека внутрішніх даних SRAM.

● Асемблер.

Ця команда запускає інтегрований асемблер. Вихідні файли Assembler можна редагувати та збирати в одному графічному середовищі. Розширення за умовчанням – ASM. Після успішного процесу складання створюються два нові файли. Один із розширенням HEX, що є файлом програми в форматі Intel HEX, який можна завантажити в пам'ять програми мікроконтролера, а інший – з розширенням LST, що є асемблерним виходом списку. Цей асемблер є відмінним рішенням для складання вихідних файлів розміром до 20K. Для більших файлів процес збирання може тривати певний час, але у файлі не існує жодних обмежень. Їх обмежує лише те, що підтримуються директори асортименту ORG, .EQU, .DB, .DW та .END.

- Дизасемблер.

AVR Simulator IDE має внутрішній дизасемблер, який запускається цією командою. Процес розбирання автоматично ініціюється відкриттям цього вікна. Дисасемблер завжди починається з місця розташування вектора скидання. Після завершення операції disassembler покаже файл вихідного списку. Сформований запис може бути збережений на диску. Користувачеві буде запропоновано ввести ім'я вихідного файлу. Розширення за умовчанням – LST.

- Менеджер точок зупинки.

Ця команда запускає інтегрований налагоджувач, який може використовуватися для налагодження та моніторингу виконання програми. Файл списків налагоджувачів програми в пам'яті може бути створений внутрішнім дизасемблером. Можна визначити до 10 точок зупинки, натиснувши на окремі рядки у списку завантажених програм. Якщо симуляція починається в режимах більшої швидкості, він автоматично перемикається в режим «Крок за кроком», коли виходить один із цих точок зупинки. Точки зупинки позначені червоними точками, а поточне значення регістра ПК позначено жовтою стрілкою. Існує можливість зберегти вказівник ПК під час моделювання.

- Спеціальні точки зупинки.

Цей інструмент моделювання забезпечує функцію для визначення спеціальних точок зупинки, які змінять швидкість моделювання в режимі «Крок за кроком», коли значення попередньо визначеного регістра змінилося

або досягне заздалегідь заданого значення. Можна визначити до 5 спеціальних точок зупинки, які одночасно активні.

● 8-розрядний світлодіодний рядок.

Цей простий модуль може бути використаний для приєднання до восьми світлодіодів до контактів мікроконтролера. Колір кожного світлодіода можна змінити, натиснувши поле кольорів світлодіодів, а призначення пікселів можна змінити, натиснувши мітку, що показує поточний вибір виводу.

● Матриця клавіатури.

Це ще один простий модуль моделювання для матриці клавіатури до 4x4. Призначення ліній можна змінити, натиснувши відповідні їх мітки.

● РК-модуль.

Ця команда запускає інтегрований симулятор РК-модуля. Перш ніж він може бути використаний для моделювання, користувач повинен налаштувати параметри інтерфейсу в діалоговому вікні «Налаштування».

● Графічний РК-модуль 128x64.

Ця команда запускає інтегрований графічний симулятор РК-модуля 128x64. Перш ніж він може бути використаний для моделювання, користувач повинен налаштувати параметри інтерфейсу в діалоговому вікні «Налаштування».

● Моделювання крокового двигуна.

Цей інструмент моделювання покаже спрощену (2-полюсну) графічну подачу однополярних фаз крокових двигунів як в автоматичних, так і в напівавтоматичних режимах. Під час імітаційного поля Step Counter буде відображатися поточна абсолютна позиція ротора, виміряна ступенями від першого визначеного положення ротора.

● I2C EEPROM.

Ця команда запускає інтегрований симуляційний модуль для зовнішніх EEPROM I2C з сімейства 24C.

● Апаратне моделювання інтерфейсу UART.

Ця команда відкриває інтерфейс для апаратного UART симулятора.

● Термінал послідовного порту ПК.

Це інтегрований інструмент, незалежний від симулятора. Цей термінал підключений до послідовного порту комп'ютера і може бути використаний для зв'язку з реальним мікроконтролером для тестування процедур послідовного зв'язку. Номер порту команд і швидкість передавання даних можна встановити за допомогою відповідних команд з меню.

● Програмне забезпечення UART Simulation Interface.

Ця команда відкриває інтерфейс для програмного UART симулятора. Це серійний пристрій, який може взаємодіяти із запущеним програмним забезпеченням мікроконтролера, що реалізує процедури UART (висловлювання SERIN, SERININV, SEROUT та SEROUTINV), використовуючи послідовну комунікацію в режимі реального часу, що моделюється на контактах RX і TX.

● Осцилограф.

Це дуже корисний інтегрований інструмент для відстеження рівня логіки на шпильках мікроконтролера під час моделювання. Це чотириканальний цифровий осцилограф. Користувач може призначати цільові штифти на канали осцилографа та змінювати довжину інтервалу відображення за допомогою команд в меню «Налаштування». Вхідні та вихідні контакти розфарбовані різними кольорами, які вибираються користувачем. Якщо для каналу осцилографа вибрано параметр Pull-up, при переході від вихідного сигналу до вхідного сигналу, шпильку буде встановлено високий рівень логіки за замовчуванням.

● Генератор сигналів.

За допомогою цього інструменту моделювання користувач може визначити до чотирьох незалежних генераторів безперервних імпульсів з вибраними цільовими штифтами, періодами імпульсів та робочими циклами. Відповідна анімація покаже поточну фазу кожного покоління імпульсів.

● 7-сегментний світлодіодний дисплей панелі.

Цей інтегрований інструмент моделювання дозволяє користувачеві визначати зв'язок з чотирма 7-сегментними світлодіодними індикаторами. Визначення з'єднання включає призначення призначень для всіх сегментів відображення та одну лінію включення для керування вибором дисплея, коли використовується кілька дисплеїв з паралельним підключенням сегментів –

мультиплексування. Активні рівні для всіх ліній з'єднання можна перевернути, щоб відповідати вимогам до обладнання. Існує також можливість змінити колір, який використовується для фарбування активних світлодіодів на дисплеях та опції Keep Last Display, щоб імітувати повільний ефект відгуку очей для програм, що використовують мультиплексування дисплеїв.

● Можливість дивитися змінні.

Під час моделювання програм, написаних із використанням інтегрованого Basic-компілятора, цей інструмент може використовуватися для перегляду поточних значень усіх змінних, оголошених в імітованій програмі. Ця функція корисна для моніторингу пам'яті для моделювання програмних файлів, які не компілюються в інтегрованому Basic-компіляторі. Змінні, додані користувачем, будуть запам'ятовуватись між сеансами, якщо той самий файл програми завантажений у симулятор. Змінні зі списку годинників можна легко видалити за допомогою команди Delete Variable, тому список може містити лише змінні, що мають особливий інтерес.

Інші команди та параметри включають:

- зміни значення змінної (також може бути запущений одним натисканням кнопки миші на змінній зі списку);
- значення показів HEX;
- підтвердження видалення.

● Цифровий термометр DS1820.

Це інструмент для моделювання програм, що підтримують зв'язок з пристроєм DS18S20 або DS18B20 за допомогою 1-провідного протоколу. Він показує внутрішню пам'ять пристрою ROM і SRAM, а також функції функціонального генератора CRC. Користувач може змінити тип пристрою, встановити 1-провідний режим інтерфейсу, температуру, яка буде вимірюватися пристроєм, разом із часом перетворення температури, який буде використовуватися для моделювання. Модуль не імітує всі доступні команди ROM та Function. Список імітованих команд можна переглянути, натиснувши кнопку інформації.

● Модуль моделювання пристрою.

За допомогою цього інструменту моделювання користувач може відслідковувати і втручатися в прошивку Modbus, що працює в симуляторі, а також створеного інтегрованим компілятором Basic. Детальніша інформація доступна в Довіднику по основному компілятору.

● Зовнішні модулі.

Цей інструмент повинен використовуватися для встановлення інтерфейсу автоматизації з максимум п’ятьма зовнішніми клієнтськими/серверними модулями. Потрібно ввести назву класу зовнішнього пристрою у формі ApplicationName.ObjectName, щоб встановити з ним зв’язок. Зовнішні клієнтські/серверні програми будуть запускатися та завершуються автоматично за допомогою IDE для програмування AVR Simulator. Додаткова інформація доступна в Посібнику з зовнішніх модулів. Доступ до нього можна отримати в меню «Довідка» головного вікна програми.

Меню опцій.

● Виберіть мікроконтроллер.

Ця команда використовується для зміни моделі мікроконтролера, яка буде використовуватися в IDE. Це призведе до скидання програми до початкового стану.

● Зміна частоти годинника.

Ця команда дозволяє користувачеві змінювати параметр частоти, який використовується для розрахунку тривалості реального часу симуляції. Введене значення в МГц запам’ятовується для майбутніх сеансів. Цей параметр також використовується багатьма операторами в комбінаторі BASIC, які використовують деяку форму процедур синхронізації (WaitMs, WaitUs, Serin, Serout, ...). Значення за замовчуванням становить 4 МГц.

● Конфігурація панелі ярликів.

Ця команда вибору відкриває простий у використанні інтерфейс для вмикання та налаштування панелі затишних комбінацій клавіш у головному вікні IDE для легкого доступу до найчастіше використовуваних команд меню. Панель може містити до трьох рядків ярликів пункту меню. Усі основні елементи меню вікна IDE доступні для розміщення на панелі.

● Зберегти позиції.

За допомогою цієї опції позиції вікон на екрані будуть запам’ятовуватися.

● Перелік реєстру вводу/виводу спочатку.

Вибір цієї опції призведе до інверсного порядку відображення списку реєстрів загального призначення та вводу/виводу.

● Зберегти завжди зверху.

Якщо вибрано цей параметр, буде замінений параметр Завжди вгорі для всіх вікон із цією функцією.

● Автоматичні параметри запуску.

За допомогою цієї утиліти користувачі можуть визначати дії, які будуть виконуватися при запуску додатка. Ці дії включають в себе автоматичне відкриття різних інструментів та моделюючих інтерфейсів з меню «Інструменти», автоматичне завантаження останніх використаних файлів у симуляторі, асемблере й базовому компіляторі.

● Скидання статистики моделювання.

Ця команда встановить нульовий тактовий цикл, лічильник інструкцій та тривалість імітації в режимі реального часу. Ця команда може бути використана для визначення тривалості конкретної частини симульованої програми.

● Змінити час записування EEPROM.

Ця команда використовується для зміни кількості мікросекунд, які будуть використовуватися для інтервалу записування EEPROM під час моделювання. Значення за замовчуванням становить 3400 мікросекунд.

● Змінити час конвертації A/D.

Ця команда використовується для зміни кількості мікросекунд, які будуть використовуватися для часу перетворення A / D під час моделювання. Значення за замовчуванням – 25 мікросекунд.

● Змінити час передавання/прийманняUART

Ця команда використовується для зміни кількості мікросекунд, які будуть використовуватися для часу передавання/приймання UART за допомогою внутрішнього апаратного симулятора UART. Реальне значення залежить від заданої швидкості передавання. Значення за замовчуванням становить 1000 мікросекунд.

● Змінити непрограмироване значення FLASH / EEPROM.

Ця команда використовується для зміни нестандартних бітових значень для пам’яті програми FLASH та пам’яті EEPROM даних від 1 до 0 та навпаки від 0 до 1. Якщо непрограмоване значення бітового значення становить 1, то непрограмоване розташування FLASH і непрограмоване розташування EEPROM міститиме шістнадцяткове значення FF. Якщо бітове значення становить 0, то вся непрограмована пам’ять заповнюється нулями.

● Компактний перегляд мікроконтролерів.

Якщо цей параметр увімкнено, вікно Microcontroller View буде відображатися у компактніші формі.

● Моделювання нескінченних циклів зупинки.

Перевірка цієї опції змусить симулятор автоматично зупинити симуляцію при зустрічі нескінченного циклу.

● Основна програма відстеження.

В даний час симульована основна заява з’явиться в основному вікні компілятора, коли цей параметр увімкнено.

● Показати вікна підтвердження.

Якщо цей параметр увімкнено, буде показано вікно підтвердження, що показує результати операцій, і буде потрібно закрити відповідь користувача.

● Зберігати вхідні стану на початку моделювання.

Якщо цей параметр увімкнено, то стани цифрових та аналогових входів у вікні перегляду мікроконтролера не будуть скидатися до вимкненого цифрового стану та нульового аналогового значення при повторному запуску симуляції.

● Використовувати напругу для аналогових входів.

Використовуйте цю опцію, якщо ви хочете бачити значення напруги (0,00V-5,00 В) замість вихідного аналогового значення (0-1023) для аналогових вхідних станів у вікнах перегляду мікроконтролера.

● Постійний аналоговий введення повзунка оновлення.

Якщо цей параметр увімкнено, тоді значення аналогового вводу будуть постійно оновлюватися за допомогою прокрутки слайдера аналогового вводу в вікні вікна перегляду мікроконтролера. В іншому випадку оновлення

відбудеться після закриття аналогового спливаючого слайдера.

● Змінити інтервал оновлення останньої ставки.

Ця команда дозволяє користувачеві змінювати інтервал оновлення (у мілісекундах) для основного інтерфейсу симуляції, коли симуляція працює на кінцевій швидкості. Однак його вартість значно не впливає на продуктивність моделювання. Значення за замовчуванням – 500 мс.

● Налаштування редактора.

За допомогою цього інструмента настроювання можна змінювати різні властивості основних редакторів кодів і асемблерів коду.

● Змінити колірну тему.

Ця команда відкриє діалогове вікно із насиченим списком доступних кольорових тем, завдяки чому користувач може змінити зовнішній вигляд програми.

Меню «Довідка».

● Довідкові теми.

Ця команда буде відображати довідкові теми. Цей файл довідки містить загальну інформацію про програму з описом всіх елементів меню.

Вікно довідки переглядача містить навігаційну панель, в якій показані теми та підтеми відображуваного файлу довідки. Клацніть правою кнопкою миші, на панелі навігації з’явиться спливаюче меню із пунктом Показати всі підтеми та Приховати всі підтеми.

Одним натисканням на елемент з панелі навігації буде переміщено фокус на панель дисплея до відповідної позиції.

Двічі клацніть по предмету, який буде показано/сховати його підтеми. На дисплеї відображається вміст завантаженого файлу довідки. Клацніть правою кнопкою миші, щоб відобразити спливаюче меню із різними параметрами та командами, включаючи копіювання, копіювання RTF, копіювання HTML, друк, збільшення шрифту, зменшення шрифту, скидання шрифтів, завжди на вершині. Вікно довідки переглядача змінювати розмір і буде пам’ятати як його позицію, так і розмір. Вертикальний сепаратор між панелями навігації та дисплеями є рухомим, і його позиція також буде збережена після того, як глядачем буде закрита.

● Довідковий посібник для базового компілятора.

Довідник з базового компілятора можна буде переглянути у переглядачі довідки.

● Зовнішні модулі вручну.

Посібник з зовнішніх модулів можна буде переглянути у переглядачі довідки.

● Перевірити наявність оновлень.

Цей інструмент дозволить користувачеві встановити зв’язок з веб-сайтом OshonSoft.com, щоб перевірити наявність нового завантаження програмного забезпечення. Файл журналу версії відображатиметься після отримання відповіді від веб-сайту.

● Інтерфейс звіту про помилку.

Цей інтерфейс слід використовувати для відправки звітів про можливі помилки в програмне забезпечення на OshonSoft.com. Крім написаної частини користувача повний звіт містить частину, яка створюється програмним забезпеченням (системний звіт).

● Про...

Ця команда покаже основну інформацію про пакет програмного забезпечення.

● Переглянути інформацію про ліцензію.

Ця команда покаже інформацію про встановлену ліцензію на програмне забезпечення. Кнопка «Інформація» відображатиме інформацію про модулі додаткового підключення, увімкнуті ліцензією.

Особливості.

● Інструкція SPM не моделюється даним симулятором.

● Сторожовий таймер не моделюється (інструкція WDR виконується як NOP)

● Режим вимкнення живлення не моделюється (інструкція SLEEP зупинить симуляцію).

Перелік периферійних пристроїв, які моделюються симулятором.

- Цифровий вхід/вихід.
- Пам’ять EEPROM даних.

- Переривання.
- Зовнішні переривання INT0, INT1.
- Модуль перетворювача A/D.
- Модуль аналогового компаратора.
- Модуль USART.
- Модуль таймера / Counter0 (з PWM).

3.15.3. Послідовність роботи з симулятором при виконанні програм.

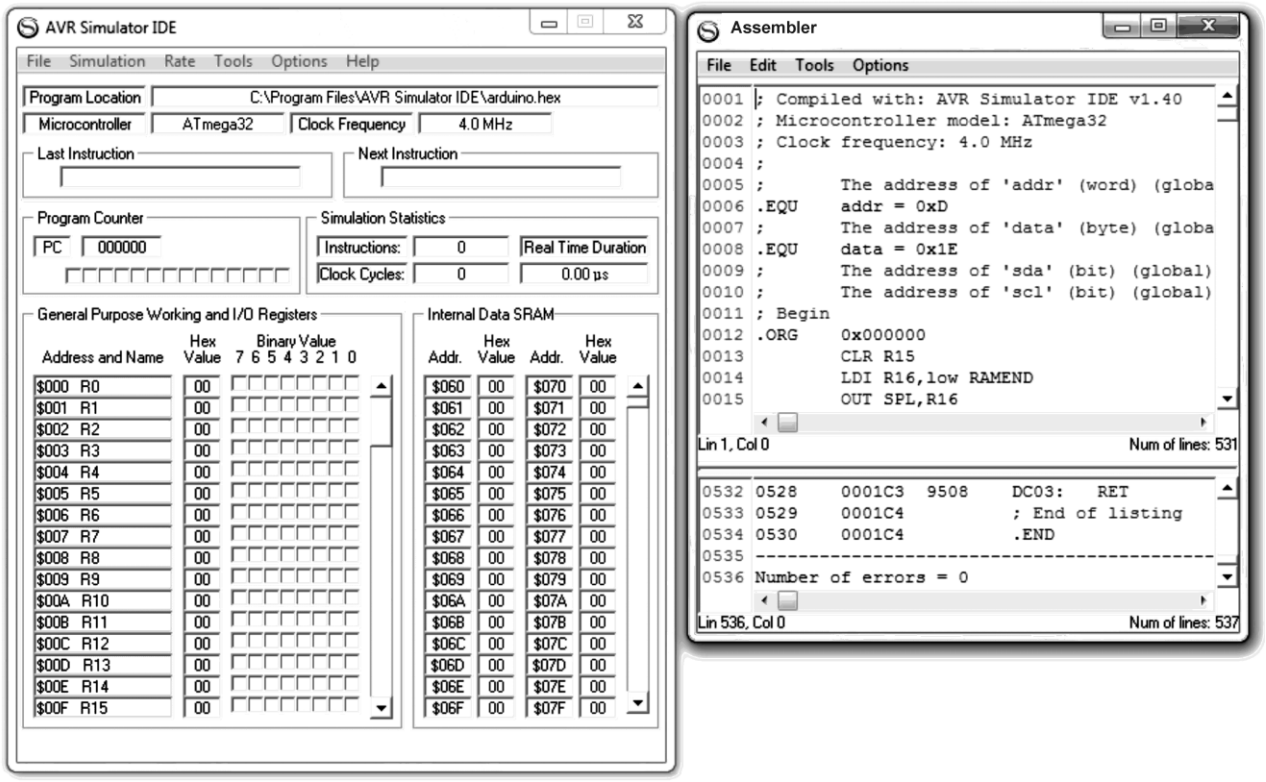


Рисунок 207. Вигляд симулятора з програмою

Виконаємо програму в AVR Simulator ID, для чого необхідно:

1. Запустити AVR Simulator IDE.
2. Натиснути Options/ Select Microcontroller.
3. Вибрати мікроконтролер Atmega32 і натиснути кнопку Select.
4. У папці «Program Files» вашого комп'ютера знайти папку «AVR Simulator IDE» з інсталяцією симулятора.
5. Відкрити файл «demo.asm» і скопіювати його вміст.
6. Натиснути Tools і у вікні, що розкриється, вибрати «Assembler».

Відкриється вікно компілятора «Assembler» (рис. 207).

7. Вставити скопійований раніше файл «demo.asm» у вікно «Assembler».

8. Натиснути Tools і у вікні, що розкриється – Assemble. В нижній половині вікна Assembler з'явиться лістинг відкомпільованого файлу. Одночасно буде створено об'єктний файл «demo.hex».

9. Вибрати File/Load Program і завантажити створений файл «demo.hex».

10. Натиснути Tools/LCD Module (відкриється вікно з LCD дисплеєм).

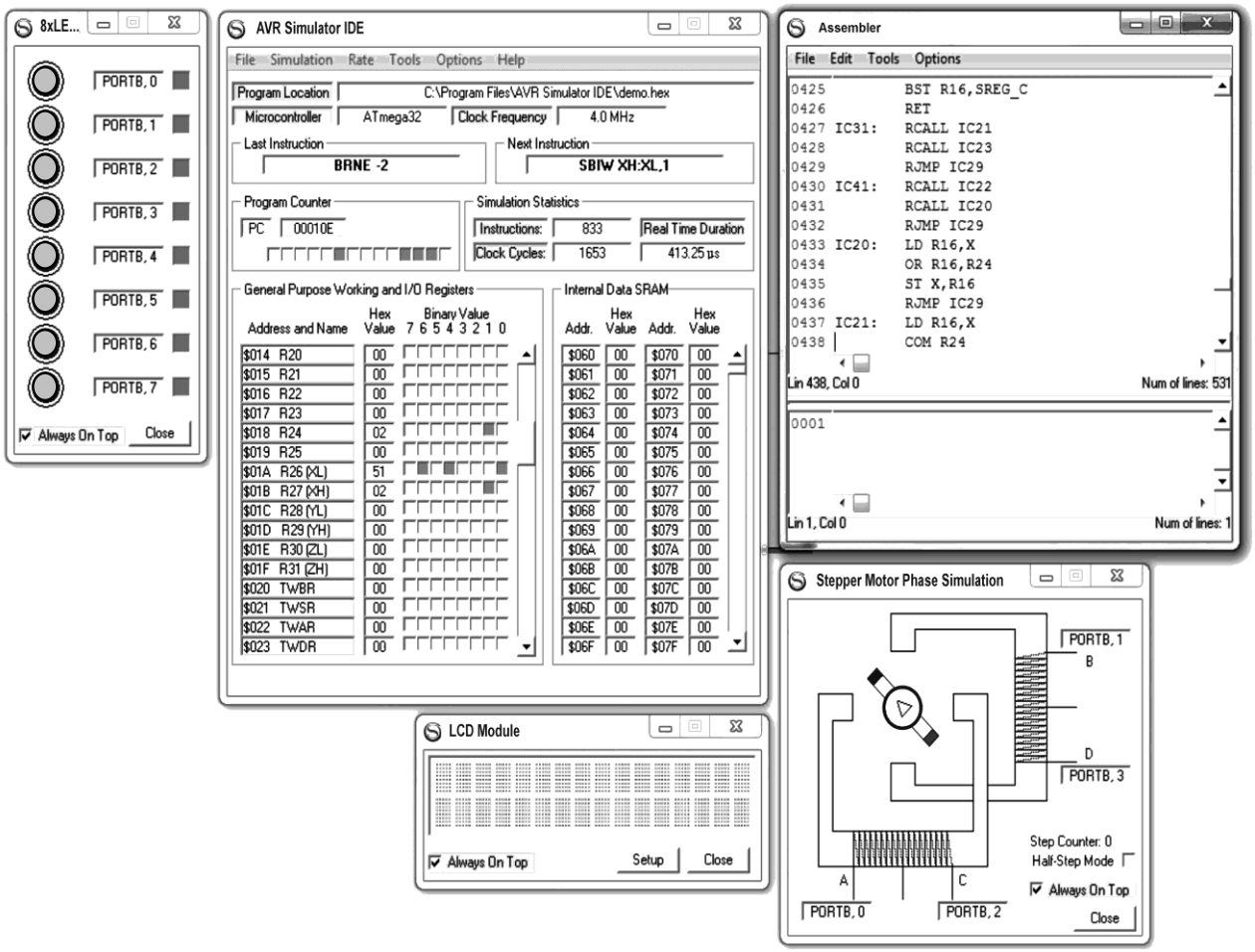


Рисунок 208. Вигляд симулятора з завантаженою програмою, вікнами: LCD Module, Stepper Motor Phase Simulation та дисплеєм 8xLED Board

11. Натиснути Tools/8xLED Board (відкриється вікно з дисплеєм, що містить вісім світлодіодів).

12. Натиснути Tools/Stepper Motor Phase Simulation (відкриється вікно симулятором уніполярного крокового двигуна).

13. Якщо програма буде виконуватися в кроковому режимі, то в основному вікні симулятора натиснути Rate/Step By Step, а далі вибрати опцію Simulation і натиснути Start. Симулятор готовий до виконання програми в кроковому режимі. Для виконання наступної команди програми потрібно натиснути на закладку STEP, яка з'явиться справа від закладки HELP вгорі основного вікна симулятора після вибору крокового режиму його роботи.

14. Для виконання програми в автоматичному режимі потрібно вибрати Rate/Extremely Fast simulation rate, а далі вибрати опцію Simulation і натиснути Start.

15. Щоб зупинити виконання програми, потрібно натиснути Simulation/Stop.

Уміст регістрів контролера, які використовуються при виконанні команд програми, знайти в області регістрів загального призначення та регістрів керування Adress and Name, яка розташована в лівій нижній частині основного вікна симулятора (виділені рожевим кольором). Усі регістри восьмирозрядні.

У процесі виконання програми по зміні кольору комірок видно, вміст яких регістрів змінюється. Забарвлення комірки відповідного розряду регістра помаранчевим кольором означає наявність «1», білим – «0».

В правій нижній частині основного вікна симулятора розташована область внутрішньої пам'яті даних Internal Data SRAM.

Вигляд стимулятора з виведенням поточної інформації на дисплеї в процесі виконання програми показано на рис. 208.

3.15.4. Контрольні запитання.

1. Основні технічні характеристики мікроконтролерів.
2. Система команд.
3. Проектування мікропроцесорних систем на базі ATMega32.
4. Контролер ATMega32. Основні характеристики.
5. Архітектура контролера ATMega32. Регістри контролера, їх призначення. Регістр стану.
6. Організація резидентної пам'яті даних та пам'яті програм ATMega32.

7. Система переривань.
8. Переривання від таймерів.
9. Переривання від портів.
10. Переривання від EEPROM.
11. Стек і повернення з підпрограм.
12. Порти вводу-виводу.
13. Регістри портів.
14. Схема ліній портів, конфігурування портів на ввід чи вивід.

Список літератури

1. Сташин В. В., Урсов А. В., Мологонцева О. Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990. 224 с.
2. Handbook of Microcontrollers/Predko Michael. NYc. McGraw-Hill. 1998. 861 p.
3. Зубчук В. И., Сигорский В. П., Шкуро А. Н. Справочник по цифровой схемотехнике. К.: Техніка, 1990. 448 с.
4. Артюхов В. Г., Будняк А. А., Лапий В. Ю., Молявко С. М., Петренко А. И. Проектирование микропроцессорной электронно-вычислительной аппаратуры: справочник. К.: Техніка, 1988. 263 с.
5. Казаринов Ю. М., Номоконов В. Н., Подклетнов Г. С., Филиппов Ф.В. Микропроцессорный комплект К1810. Структура, программирование, применение. М.: Высшая школа, 1990. 269 с.
6. Якименко Ю. І., Терещенко Т. О., Сокол Є. І., Жуйков В. Я., Петергера Ю. С. Мікропроцесорна техніка; за ред. Т. О. Терещенко. К.: Кондор, 2004. 440 с.
7. Бродин В. Б., Калинин А. В. Системы на микроконтролерах и БИС программируемой логики. М.: ЭКОМ, 2002. 400 с.
8. Practical Interfacing Techniques for Microprocessor Systems. James Coffron. NJ. Prentice Hall 1983. 401 p.
9. Боборыкин А. В., Липовецкий Г. П., Литвинский Г. В., Оксинь О. Н., Прохорчик С. В., Проценко Л. В., Пертенко Н. В., Сергеев А. А., Сивобород П. В. Однокристальные микроЭВМ. М.: МИКАП, 1994. 400 с.
10. Ульрих В. А. Микроконтроллеры PIC16X7XX: 2-е изд., перераб и доп. СПб.: Наука и техника, 2002. 320 с.
11. Les microcontroleurs PIC - Applications: Applications Broché. Christian Tavernier. Paris. Dunod. 2002. 304 p.
12. PIC16F84 Data Sheet. 18-pin 8-Bit CMOS EEPROM Microcontroller. Microchip Technology Inc. 1996. 108 с. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/30445c.pdf.

13. PIC16F84A Data Sheet 18-pin Enhanced FLASH/EEPROM 8-bit Microcontroller. Microchip Technology Inc. 2001. 86 c. URL: http://ww1. microchip.com/downloads/en/DeviceDoc/35007b.pdf.
14. Евстифеев В. А. Микроконтроллеры AVR семейства Mega. Руководство пользователя. М.: Изд. дом «Додека – XXI», 2007.
15. Данилин А. Программа-симулятор PIC Simulator IDE. Современная электроника. 2006. № 4. С. 68–76.
16. Messen, Steuern und Regeln mit AVR-Controllern. Trampert W. Munich, Franzis'Verlag GmbH 2010. 256 p.
17. Проектування мікропроцесорних систем керування: навчальний посібник/ Медвідь В.Р, Пісьціо В.П., Козбур І.Р. Тернопіль: Вид-во ТНТУ ім. Івана Пулюя, 2015. 354 с.
18. Кирик В. В. Мікропроцесорні системи та промислові контролери: Навчальний посібник. Київ: АМУ, 2010. 72 с.
19. Мілих В. І., Шавьолкін О. О. Електротехніка, електроніка та мікропроцесорна техніка: підручник; за ред. В. І. Мілих. 2-е вид. К.: Каравела, 2008. 688 с.
20. Бойко В. І., Гуржій А. М., Жуйков В. Я. та ін.Схемотехніка электронних схем: У 3 кн. Кн.3 Мікропроцесори та мікроконтролери: підручник. 2-ге вид., допов. і переробл. К.: Вища шк., 2004. 399 с.
21. Павельчак А. Г., Самотий В. В., Яцук Ю. В. Програмування мікроконтролерів систем автоматики: конспект лекцій. Львів: Львівська політехніка, 2012. 143 с. 8. ATmega48A/PA/88A/PA/168A/PA/328/P DATASHEET – Atmel Corporation. 657 с.
22. Ляшенко О., Мартинюк О. Моделювання та дослідження електронних пристроїв: навч. посібник. Луцьк: Східноєвроп. нац. ун-т ім. Лесі Українки, 2013. 217 с.
23. Цирульник С. М., Азаров О. Д., Крупельницький Л. В., Трояновська Т. І. Мікропроцесорна техніка: навчальний посібник. Вінниця: ВНТУ, 2017. 123 с.

24. Kunikowski Wojciech, Czerwiński Ernest, Olejnik Paweł, Awrejcewicz. An Overview of ATmega AVR Microcontrollers Used in Scientific Research and Industrial Applications. Pomiary Automatyka Robotyka. 2015, Par. 19, P. 15-20. DOI: 10.14313/PAR_215/15.
25. Dogan Ibrahim, Chapter 1. Microcomputer systems, Arm-Based Microcontroller Multitasking Projects. 2021. P. 1–12. DOI: 10.1016/B978-0-12- 821227-1.00001-3.
26. Ng T. S. Microcontroller. Real Time Control Engineering. 2016. Vol. 65. DOI: 10.1007/978-981-10-1509-0_4.

Для нотаток

Навчально-методична література

І.Р. Козбур, П.О. Марущак, В.Р. Медвідь, В.Б. Савків, В.П. Пісьціо

Проектування мікропроцесорних систем керування

Навчальний посібник

Формат 60х90/16. Обл. вид. арк. 10,20. Тираж 100 пр. Зам. № 3567

Видавництво Тернопільського національного
технічного університету імені Івана Пулюя.
46001, м. Тернопіль, вул. Руська, 56.
Свідоцтво суб'єкта видавничої справи ДК № 4226 від 08.12.11.