МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет нафти і газу

Кафедра інформаційно-телекомунікаційних технологій та систем

О. В. Євчук

МІКРОКОНТРОЛЕРИ

КОНСПЕКТ ЛЕКЦІЙ

Івано-Франківськ
2019

Рецензент:

Николайчук М.Я. кандидат технічних наук, доцент кафедри інформаційно-телекомунікаційних технологій та систем Івано-Франківського національного технічного університету нафти і газу

Євчук О. В.

Є-17 Мікроконтролери

Конспект лекцій. Івано-Франківськ

ІФНТУНГ, 2019. 147 с.

Конспект лекцій містить матеріал для проведення лекційних занять з дисципліни «Цифрова та аналогова схемотехніка». Розроблений відповідно до робочої програми навчальної дисципліни, чинного навчального плану. Може бути використаний студентами денної та заочної форм навчання.

Призначено для підготовки бакалаврів за спеціальністю 151 – «Автоматизація та комп’ютерно-інтегровані технології».

ВСТУП

Сучасні мікроконтролери все більшою мірою стають невід'ємною частиною нашого життя. Вони все частіше зустрічаються в конструкціях сучасних телефонів, телевізорів, автомагнітол, пральних машин та іншої побутової техніки, збагачуючи останню різноманітними додатковими функціями, які суттєво покращують її експлуатаційні характеристики. Використання мікроконтролерів, крім того, дозволяє зменшити масо-габаритні та енергетичні показники розроблюваного обладнання. Тому безсумнівним є використання мікроконтролерів в промисловому обладнанні. Метою вивчення даного курсу є освоєння базових моделей мікроконтролерів серій КР1816ВЕ51 (аналог Intel 8051 або MSC-51) та AVR мікроконтролерів серій AT90 та ATMega —що дозволить майбутнім інженерам в подальшому здійснювати самостійну розробку приладів на їх базі.

В загальному ***Мікроконтролер (або Мікро-ЕОМ)*** це функціонально завершена інтегральна схема (ІС), яка має всі основні вузли та складові мікропроцесорної системи загального призначення—оперативний запам'ятовуючий пристрій (ОЗП), постійний запам'ятовуючий пристрій (ПЗП), арифметико-логічний пристрій (АЛП), пристрої вводу-виводу (ПВВ), та систему переривань. Очевидно, що в порівнянні із сучасними мікропроцесорними системами, які містять в своєму складі десятки ІС, мікроконтролерні системи поступаються показниками швидкодії, обсягом оперативної пам'яті, та обсягом пам'яті програм, однак переважають їх за показниками енергозбереження та масо-габаритними показниками.

Використовуючи мікроконтролери можна створювати "інтелектуальні" пристрої для задач, які не потребують високої швидкодії— з часом алгоритмічної реакції на події більшому ніж 0.1...3 сек. У випадку недостатності обчислювальних можливостей для вирішення поставленої задачі, можна застосувати прийом багатоконтролерності, розбивши поставлену задачу на декілька підзадач і виділивши для неї окремий мікроконтролер. У випадку, якщо загальна вартість кількох контролерів буде перевищувати вартість більш продуктивної моделі, слід розглянути можливість використання останньої.

Сучасний процес розробки мікроконтролерних засобів передбачає наступні фази:

1. Аналіз поставленої задачі, попередній вибір мікроконтролера, розробку блок-схеми апаратної частини пристрою та алгоритму функціонування програмного забезпечення для нього.
2. Попередню реалізацію програмного забезпечення, використовуючи крос-компілятори (компілятори, що генерують код процесора для деякої платформи, відмінної від тієї, на якій функціонують).
3. Попереднє відлагоджування програмного забезпечення з використанням програмних симуляторів (програми що імітують роботу досліджуваного контролера на ПЕОМ) або апаратних емуляторів (тестових плат, які містять досліджуваний контролер) і мають змогу передавати відлагоджувальну інформацію до управляючої ПЕОМ.
4. У випадку, якщо в результаті попередньої симуляції програми встановлено, що контролер встигає виконати поставлену задачу у відведений для цього час, здійснюється розробка і виготовлення реального прототипу пристрою. В іншому випадку розглядається можливість оптимізації алгоритму по швидкості, і якщо такі можливості вичерпані переходять до більш швидкісної моделі мікроконтролера. При виготовленні плати-прототипу пристрою використовують друковані плати та елементи макетування.
5. Тестування плати-прототипу. Для цього на базі основного, створюється окреме програмне забезпечення, яке дає змогу перевірити функціонування всіх вузлів плати. Принцип дії такого програмного забезпечення – генерація тестових сигналів, які можна контролювати за допомогою осцилографа, у відповідь на зовнішні команди. При здійсненні тестування плати всі виявлені недоліки схеми усуваються шляхом впаювання додаткових провідників, або шляхом усування існуючих.

6. У випадку правильного функціонування плати, здійснюють відлагоджування програмного забезпечення на ній, та попереднє випробовування приладу.

7. З врахуванням знайдених помилок, здійснюють чистове проектування та виготовлення апаратних засобів розроблюваного приладу.

Вважається, що використання мікроконтролерних засобів є доцільним, якщо при цьому зменшується кількість мікросхем жорсткої логіки (тригерів, регістрів) на 30, або таке використання надасть приладу додаткових сервісних функцій, які неможливо реалізувати іншим чином. Проте слід зауважити, що розробка з використанням мікроконтролерів, передбачає наявність додаткових апаратних засобів (програматорів – пристроїв, що здійснюють запис виконавчого коду в ПЗП контролера або в зовнішній ПЗП), а також вимагає від інженера знання схемотехніки, конструювання радіоелектронної апаратури (РЕА) та навичок програмування. Крім того, процес відлагоджування мікроконтролерних пристроїв є більш складним у порівнянні із пристроями, що виконані на жорсткій логіці. Враховуючи це, вибір способу вирішення поставленої задачі залишається в компетенції інженера – розробника, а задача студента – максимально засвоїти розглянутий нижче матеріал, оскільки це сприятиме полегшенню його творчої праці в майбутньому .

1. Мікроконтролери серії 8051

1.1 Загальна структура мікроконтролера 8051

Основою структури МК MCS-51 є 8-розрядна внутрішня шина даних (рис.1.1), до якої під'єднані :

- блок управління БУ із лічильником команд PC, дешифратором команд ДшК та покажчиком даних DPTR;
- арифметико-логічний пристрій CPU;
- резидентна (внутрішня) пам'ять програм ROM;
- внутрішня оперативна пам'ять RAM (256 байт), до складу якої входять :

резидентна пам'ять даних РПД (128 байт) із 4 банками по 8 регістрів оперативного призначення РОП, регістри спеціальних функцій РСФ;

- чотири 8-розрядні порти P0...P3;
- два 16-розрядні таймери-лічильники подій T/C1, T/C2;
- блок послідовного інтерфейсу Serial Port.

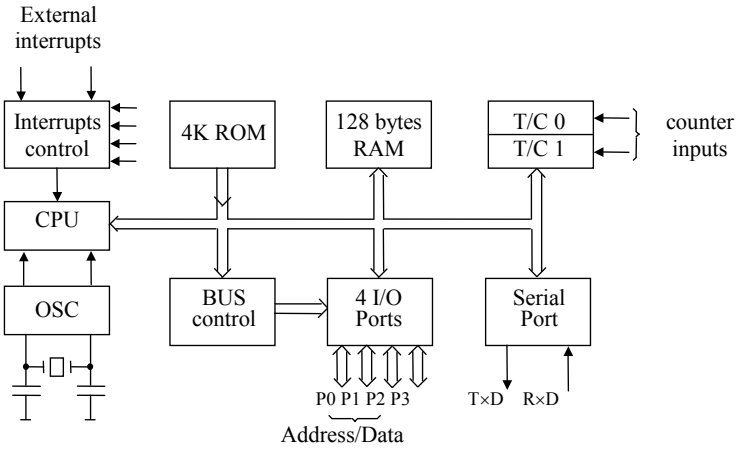


Рисунок 1.1 - Структура МК MCS-51

В CPU здійснюється : 1) формування лічильником команд PC поточної 16-розрядної адреси пам'яті програм та дешифрація коду поточної команди в ДшК; 2) формування 8-розрядної адреси внутрішньої оперативної пам'яті; 3) формування 16-розрядної адреси зовнішньої пам'яті даних у покажчику даних DPTR; 4) формування синхросигналів, що забезпечують координацію роботи вузлів МК.

Регістр PSW

CY	AC	F0	RS1	RS0	OV	-	P

Флаг	Функція
CY	Флаг переносу
AC	Флаг додаткового переносу
F0	Флаг користувача
RS1	Біт 1 вибору банку регістрів оперативного призначення.
RS0	Біт 0 вибору банку регістрів оперативного призначення.
OV	Флаг переповнення
-	Не використовується
P	Флаг парності. Встановлюється/скидається в кожному машинному циклі для визначення парної/непарної кількості розрядів акумулятора, що встановлені в стан "1"

Паралельний 8-розрядний CPU забезпечує виконання арифметико-логічних операцій над двійковими та двійково-десятковими числами. CPU може оперувати з чотирма форматами даних - бітами, тетрадами, байтами, двобайтовими словами. Можливість довільного застосування великої кількості однобітових операцій дозволяє широко використовувати МК в системах автоматики. Основним джерелом операнда і місцем фіксації результату при виконанні арифметико-логічних операцій є 8-розрядний акумулятор. Разом із тим, деякі операції (інкрементування/декрементування, порівняння, виконання логічних функцій) здійснюються і без використання акумулятора. Ознаки, що формуються в CPU після виконання арифметико-логічних команд, фіксуються в регістрі слова стану PSW. Флаг переносу CY, крім звичайних функцій продовжувача акумулятора в командах додавання, віднімання і зсуву, виконує функції акумулятора в командах, що оперують із однобітовими операндами. Флаг OV фіксує арифметичне переповнення в операціях із знаковими числами, що дозволяє використовувати в повному обсязі властивості арифметики в додаткових кодах. Флаг OV встановлюється в 1 також при діленні на нуль і у випадку одержання результату множення, що перевищує 255. (Флаг CY при виконанні операцій множення та ділення скидається в нуль). Флаг FO, що фіксує будь-яку ознаку, обрану користувачем, може бути програмно встановлений в "1", скинутий в 0, проінвертований чи перевірений.

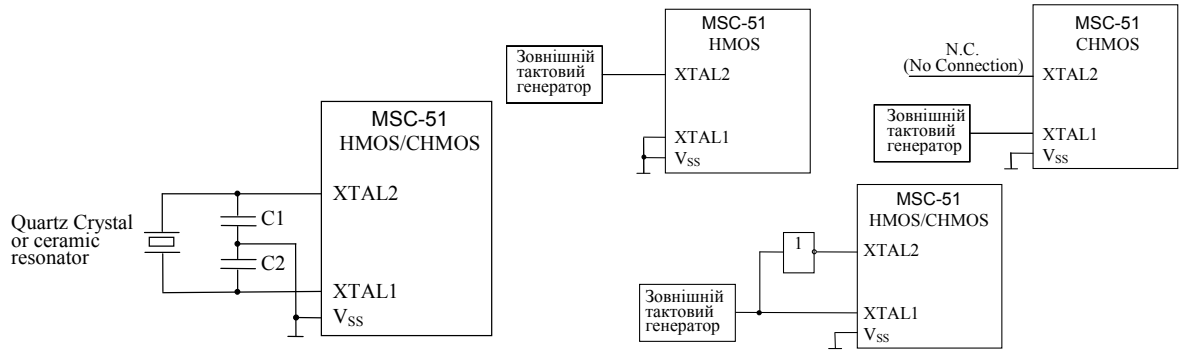
Бітами RS1, RS0 вибираються комірки пам’яті, які будуть служити активним банком регістрового файлу (регістри R0…R7):

RS1	RS0	Банк	Адреса
0	0	0	00H…07H
0	1	1	08H…0FH
1	0	2	10H…17H
1	1	3	18H…1FH

Особливістю МК MCS-51 є те, що акумулятор не відноситься до регістрів оперативного призначення, а разом із регістром PSW входить до складу регістрів спеціальних функцій.

1.2 Синхронізація мікроконтролера

Джерелом синхронізації для МК може бути зовнішній тактовий генератор або під’єднаний до виводів XTAL1, XTAL2 зовнішній кварцовий резонатор.



а - використання внутрішнього тактового генератора

б - використання зовнішнього тактового генератора

Рисунок 1.2 - Способи тактування МК

Для різних версій МК частота тактового генератора може знаходиться в межах від 3,5 до 33 мГц (в режимі програмування тактова частота повинна бути в межах від 4 до 6 МГц). Приклади використання внутрішнього і зовнішнього тактового генератора наведені на рис.1.2.

1.3 СТРУКТУРА ПАМ'ЯТІ

Пам'ять МК складається з п'яти окремих фізичних адресних просторів, що функціонально розділені як програмно - за рахунок різних способів адресації, так і апаратно - шляхом керування різними сигналами запису та читання :

- пам'ять програм внутрішня до 4 Кбайт,
- пам'ять програм зовнішня - до 64 К,
- пам'ять даних резидентна - 128 байт,
- регістри спеціальних функцій,
- пам'ять даних зовнішня - до 64 К.

Адресний простір пам'яті програм складається з внутрішньої та зовнішньої частин. МК MCS-51 не виходить за межі внутрішньої пам'яті програм, якщо адреса будь-якої команди не перевищує 0FFF. Пам'ять програм має 16-розрядну адресну шину, її комірки адресуються за допомогою лічильника команд PC або команд, що формують пряму 8/16-розрядну адресу. Комірки 0000...0023h в пам'яті програм зарезервовані для звертання до підпрограм обслуговування переривань.

Внутрішня і зовнішня пам'яті програм утворюють єдиний адресний простір ємністю до 64 Кбайт і не розрізняються програмістом, але обов'язково відокремлюються на етапі проектування апаратних засобів.

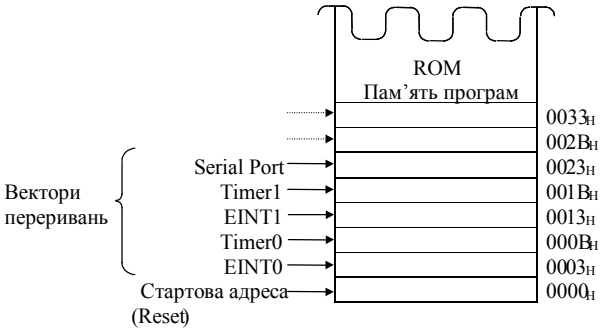


Рисунок 1.3 - Розподіл початкових адрес пам'яті МК

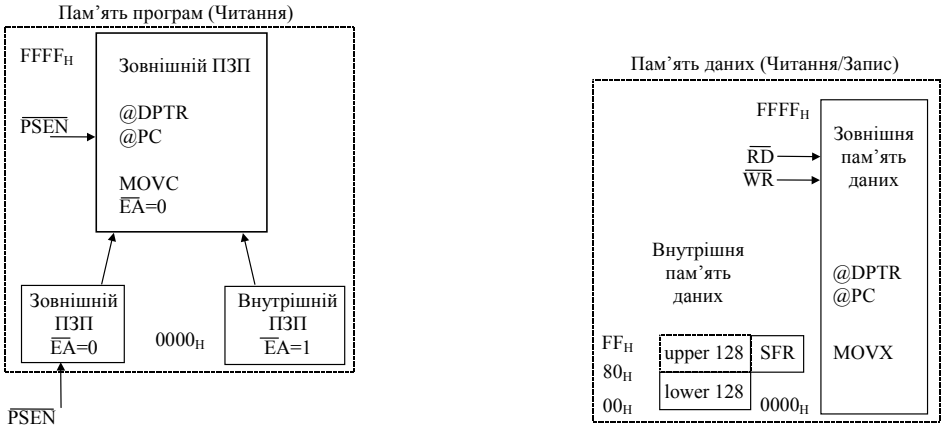


Рисунок 1.4 - Організація пам'яті мікроконтролера MCS-51

7FH 30H	Побайтно-адресований простір ОЗП (direct, indirect)							
2FH	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H
2EH	77H	76H	75H	74H	73H	72H	71H	70H
	Побітно- адресований простір ОЗП							

	(direct)							
21H	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
20H	07H	06H	05H	04H	03H	02H	01H	00H
1FH 18H	RB3							
17H 10H	RB2							
0FH 08H	RB1							
07H 00H	← SP після RESET RB0(R7+R0)							

Рисунок 1.5 - Нижні 128 байт внутрішнього ОЗП.

Внутрішній ОЗП поділений на дві області - резидентна пам'ять даних з адресами 70...7F та регістри спеціального функцій SFR з адресами 80...FF (рис.1.5). В свою чергу, резидентна пам'ять даних утворена трьома частинами :
- 4 банки регістрів оперативного призначення з адресами 00...1F,
- 16 байтових комірок з адресами 20...2F, що одночасно утворюють область з 128 однобітових комірок окремої адресації з адресами 00...7F;
- 80 байтових комірок ОЗП з адресами 30...7F.

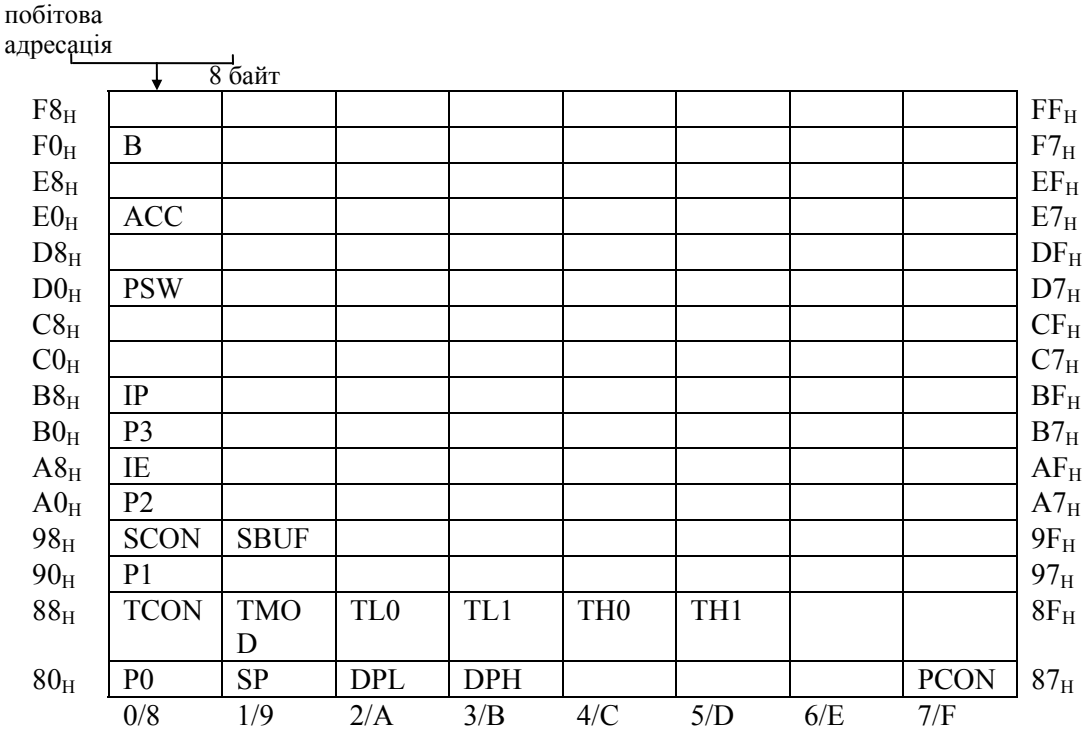


Рисунок 1.6 - Розміщення регистрів спеціальних функцій в просторі SFR

Ім'я SFR	Адреса SFR	Адреса і мнемоніка бітів								Значення після RST
P0	80_H	87_H P0.7	86_H P0.6	85_H P0.5	84_H P0.4	83_H P0.3	82_H P0.2	81_H P0.1	80_H P0.0	FF_H
TCON	88_H	$8F_H$ TF1	$8E_H$ TR1	$8D_H$ TF0	$8C_H$ TR0	$8B_H$ IF1	$8A_H$ IT1	89_H IE0	88_H IT0	00_H

P1	90_H	97_H P1.7	96_H P1.6	95_H P1.5	94_H P1.4	93_H P1.3	92_H P1.2	91_H P1.1	90_H P1.0	FF_H
SCON	98_H	$9F_H$ SM0	$9E_H$ SM1	$9D_H$ SM2	$9C_H$ REN	$9B_H$ TB8	$9A_H$ RB8	99_H TI	98_H RI	00_H
P2	$A0_H$	$A7_H$ P2.7	$A6_H$ P2.6	$A5_H$ P2.5	$A4_H$ P2.4	$A3_H$ P2.3	$A2_H$ P2.2	$A1_H$ P2.1	$A0_H$ P2.0	FF_H
IE	$A8_H$	AF_H EA	AE_H —	AD_H —	AC_H ES	AB_H ET1	AA_H EX1	$A9_H$ ET0	$A8_H$ EX0	$0XX00000_B$
P3	$B0_H$	$B7_H$ P3.7	$B6_H$ P3.6	$B5_H$ P3.5	$B4_H$ P3.4	$B3_H$ P3.3	$B2_H$ P3.2	$B1_H$ P3.1	$B0_H$ P3.0	FF_H
IP	$B8_H$	BF_H —	BE_H —	BD_H —	BC_H PS	BB_H PT1	BA_H PX1	$B9_H$ PT0	$B8_H$ PX0	$XXX00000_B$
PSW	$D0_H$	$D7_H$ CY	$D6_H$ AC	$D5_H$ F0	$D4_H$ RS1	$D3_H$ RS0	$D2_H$ OV	$D1_H$ —	$D0_H$ P	00_H
A	$E0_H$	$E7_H$ —	$E6_H$ —	$E5_H$ —	$E4_H$ —	$E3_H$ —	$E2_H$ —	$E1_H$ —	$E0_H$ —	00_H
B	$F0_H$	$F7_H$ —	$F6_H$ —	$F5_H$ —	$F4_H$ —	$F3_H$ —	$F2_H$ —	$F1_H$ —	$F0_H$ —	00_H

Рисунок 1.7 - Регістри SFR при побітній адресації.

Ім’я SFR	Адреса SFR	Мнемоніка бітів								Значення після RST
SP	81_H	—	—	—	—	—	—	—	—	00000111_B
DPL	82_H	—	—	—	—	—	—	—	—	00_H
DPH	83_H	—	—	—	—	—	—	—	—	00_H
PCON	87_H	SMOD SMOD	— —	— —	— —	— GF1	— GF0	— PD	— IDL	$0XXXXXXX_B$ $0XXX0000_B$
TMOD	89_H	GATE1		M1.1	M0.1	GATE0		M1.0	M0.0	00_H
TL0	$8A_H$	—	—	—	—	—	—	—	—	00_H
TL1	$8B_H$	—	—	—	—	—	—	—	—	00_H
TH0	$8C_H$	—	—	—	—	—	—	—	—	00_H
TH1	$8D_H$	—	—	—	—	—	—	—	—	00_H
SBUF	99_H	—	—	—	—	—	—	—	—	$XXXXXXXX_B$

Рисунок 1.8 - Регістри SFR при байтовій адресації

Регістр B застосовується при виконанні команд множення і ділення, а також використовується як розширювач акумулятора при обробці двобайтових чисел. 8-розрядний регістр покажчику стеку SP інкрементується при занесенні даних до стеку командами PUSH і CALL. Стек може бути організований в будь-якому місці резидентної пам'яті даних. Після системного скидання SP встановлюється на адресу 07, що призводить до організації області стеку, починаючи з адреси 08 (банк 2 РОП).Покажчик даних DPTR, що містить два байтових регістри: DPH - старший і DPL - молодший, призначений для зберігання 16-розрядної адреси комірки зовнішньої пам'яті даних. DPTR може використовуватись як 16-бітовий регістр або два незалежних байтових регістри. Інші регістри спеціальних функцій забезпечують роботу окремих вузлів МК і детально розглядаються під час аналізу відповідних блоків МК.

По аналогії із другою частиною резидентної пам'яті даних, що дозволяє звертатись як до байтових комірок, так і до окремих бітів, одинадцять регістрів спеціальних функцій також мають подвійну - байтову і бітову адресацію.

1.4 Система команд

Система команд МК містить 111 базових команд, серед яких 50 - однобайтові, 45 - двобайтові, 16 - трьохбайтові. Більшість команд (37 однобайтових, 27 двобайтових) виконуються за один машинний цикл, 45 команд (10 одно-, 19 дво-, 16 трьохбайтових) виконуються за 2 машинних цикли, команди множення MUL та ділення DIV здійснюються на протязі чотирьох машинних циклів.

Використовуються 11 способів адресації - 4 основні: безпосередня, пряма, непряма, а також комбіновані: безпосередня з прямою, пряма з прямою, пряма з непрямою, тощо. Завдяки зміні режимів адресації загальна кількість команд збільшується до 255 з 256 можливих при однобайтовому коді команд. Важлива особливість: бітові операнди і регістри спеціальних функцій підлягають тільки прямій адресації - на відміну від інших комірок пам'яті, для яких, крім прямої, можлива непряма адресація через регістри R0, R1, DPTR та лічильник команд PC.

1.4.1 Команди передачі даних

Переважна більшість команд передачі даних оперує всередині внутрішнього ОЗП - з акумулятором A, регістрами оперативного призначення Rn (від R0 до R7), комірками внутрішнього ОЗП, адреси яких (ad, add або ads - від 00 до FF) прямо вказуються в команді, комірками резидентної пам'яті даних, адреси яких (від 00 до 7F) вказуються в першому або другому регістрах оперативного призначення Ri з обраного банку.

Команди передачі даних, які використовують внутрішній ОЗП

Мнемоніка	Операція	Режими адресації			
		Dir	Ind	Reg	Imm
MOV A, <src>	A=<src>	x	x	x	x
MOV <dest>, A	<dest>=A	x	x	x	
MOV <dest>, <src>	<dest>=<src>	x	x	x	x
MOV DPTR, #data$_{16}$	DPTR=16-bit immediate constant				x
PUSH <src>	INC SP; MOV "@SP",<src>	x			
POP <dst>	MOV <dst>, "@SP"; DEC SP	x			
XCH A, <byte>	обмін даними між ACC і <byte>	x	x	x	
XCHD A, @Ri	обмін молодшими тетрадами між ACC і @Ri		x		

Команди передачі даних,які використовують зовнішню пам'ять даних (ОЗП)

Мнемоніка	Операція	Розрядність даних
MOVX A, @Ri	Read external RAM @Ri	8 біт
MOVX @Ri, A	Write external RAM @Ri	8 біт
MOVX A, @DPTR	Read external RAM @DPTR	16 біт
MOVX @DPTR, A	Write external RAM @DPTR	16 біт

Команди для роботи з таблицями

Мнемоніка	Операція
MOVC A, @A+DPTR	Read Program Memory at (A+DPTR)

MOVC A, @A+PC	Read Program Memory at (A+PC)

1.4.2 Команди арифметичних операцій

Мнемоніка	Операція	Режими адресації			
		Dir	Ind	Reg	Imm
ADD A, <byte>	A=A+<byte>	x	x	x	x
ADDC A, <byte>	A=A+<byte>+C	x	x	x	x
SUBB A, <byte>	A=A-<byte>-C	x	x	x	x
INC A	A=A+1	Accumulator only			
INC <byte>	<byte>=<byte>-1	x	x	x	
INC DPTR	DPTR=DPTR+1	Data Pointer only			
DEC A	A=A-1	Accumulator only			
DEC <byte>	<byte>=<byte>-1	x	x	x	
MUL A,B	B:A=B×A	ACC and B only			
DIV A,B	A=Int[A/B], B=Mod[A/B]	ACC and B only			
DA A	двійкова корекція	Accumulator only			

В групі команд арифметичних операцій необхідно виділити команду беззнакового множення вмісту акумулятора і регістру B: MUL A,B із фіксацією результату в регістрі B (старший байт) і акумуляторі (молодший байт),а також команду беззнакового ділення вмісту акумулятора на вміст регістру B із фіксацією частки в акумуляторі, залишку - в регістрі B.
Відмітимо, що для арифметичних операцій додавання та віднімання обов'язковим є зберігання в акумуляторі одного з операндів, а далі результату операції.

1.4.3 Логічні команди

Мнемоніка	Операція	Режими адресації			
		Dir	Ind	Reg	Imm
ANL A, <byte>	A=A.AND.<byte>	x	x	x	x
ANL <byte>, A	<byte>=<byte>.AND.A	x			
ANL <byte>, #data	<byte>=<byte>.AND.#data	x			
ORL A, <byte>	A=A.OR.<byte>	x	x	x	x
ORL<byte>, A	<byte>=<byte>.OR.A	x			
ORL <byte>, #data	<byte>=<byte>.OR.#data	x			
XRL A, <byte>	A=A.XOR.<byte>	x	x	x	x
XRL <byte>, A	<byte>=<byte>.XOR.A	x			
XRL <byte>, #data	<byte>=<byte>.XOR.#data	x			
CLR A	$A=00_H$	Accumulator only			
CPL A	A=.NOT.A	Accumulator only			
RL A	зсув ACC вліво на 1 біт	Accumulator only			
RLC A	зсув акумулятора вліво через перенiс	Accumulator only			
RRA	зсув ACC вправо на 1 біт	Accumulator only			
RRC A	зсув акумулятора вправо через перенiс	Accumulator only			
SWAP A	обмін тетрад в акумуляторі	Accumulator only			

На відміну від арифметичних операцій, логічні функції І, АБО, Виключаюче АБО можуть здійснюватись без участі акумулятора, а їх результат може бути сформований у будь-якій комірці внутрішнього ОЗП, пряма адреса якої (від 00 до 7F) вказується в команді (що дуже зручно для маскування операндів при роботі з портами).

Візуальні схеми роботи логічних зсувів:

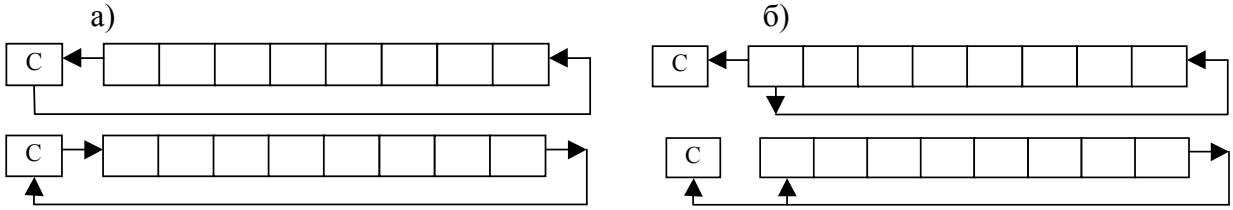


Рисунок 1.9 - Зсув : а) через флаг переносу - RLC,RRC; б) циклічний - RL,RR.

1.4.4 Операції з бітами

Операції з бітами введені в набір інструкцій виходячи із специфіки роботи однокристальних ЕОМ. Як правило основна задача МК в автоматичних системах керування - сканувати вхідні контакти в очікуванні управляючого сигналу, формувати складні управляючі послідовності при програмній емуляції послідовних інтерфейсів, тощо. Оскільки вихідні контакти МК сформовані у вигляді груп 8-бітових портів, то постає потреба ефективного оперування із окремими бітами, при чому дуже часто різні біти одного порту можуть бути різноспрямовані та призначені для виконання різних функцій вводу і виводу.
МК MCS-51 надає користувачу широкий набір операцій з бітами. Це логічне "І" біту та флагу С, пересилка бітів через флаг С, встановлення біту С, операції умовних переходів.
Нижче наведений список побітових операцій:

Мнемоніка	Операція
ANL C, bit	C=C.AND.bit
ANL C, /bit	C=C.AND.(.NOT.bit)
ORL C, bit	C=C.OR.bit
ORL C, /bit	C=C.OR.(.NOT.bit)
MOV C, bit	C=bit
MOV bit, C	bit=C
CLRC	C=0
CLR bit	bit=0
SETB C	C=1
SETB bit	bit=1
CPL C	C=.NOT.C
CPL bit	bit=.NOT.bit
JC rel	Jump if C=1
JNC rel	Jump if C=0
JB bit, rel	Jump if bit=1
JNB bit, rel	Jump if bit=0
JBC bit, rel	Jump if bit=1; CLR bit

1.4.5 Команди безумовних переходів

Передача управління всередині програми може здійснюватись в межах повної пам'яті програм - командами довгого переходу, в межах 2048-байтової сторінки пам'яті програм - командами

абсолютного переходу, а також командами відносного переходу - в межах -127...+127 байт відносно адреси команди, наступної за командами відносного переходу. Безумовний довгий перехід реалізовано трьома командами - однобайтовою командою непрямої адресації JMP @(A+DPTR) і двома трьохбайтовими командами прямої адресації, другий та третій байти яких представлені 16-розрядною адресою переходу, а мнемоніка містить букву L (Long) : LJMP - довгий перехід, LCALL - довгий виклик підпрограми. По командах JMP @(A+DPTR) та LJMP лічильник команд PC завантажується відповідно значенням A+DPTR та прямою адресою, вказаною в команді. По команді LCALL лічильник команд тричі інкрементується, вміст PC заноситься до стеку (молодший байт до комірки з адресою SP+1, старший байт до комірки з адресою SP+2), покажчик стеку SP збільшується на 2, до лічильника команд PC заноситься 16-розрядна адреса, що вказана в команді.

Двобайтові команди безумовного абсолютного переходу містять 11-розрядну адресу прямого переходу, розміщену в другому байті команди (8 молодших бітів адреси) та на місці старших трьох бітів першого байту команди (3 старші біти адреси). Мнемоніка команд абсолютного переходу починається буквою A (Absolute) : AJMP - абсолютний перехід, ACALL - абсолютний виклик підпрограми. При виконанні команд 11 молодших бітів лічильника команд PC заміняються 11-ма бітами адреси, що вказані в команді. По команді ACALL перед тим попередньо виконується подвійне інкрементування лічильника PC, занесення його вмісту до стеку аналогічно команді LCALL та подвійне інкрементування покажчика стеку SP.

Мнемоніка	Операція
JMP addr	перехід на адресу addr
JMP @A+DPTR	перехід на адресу A+DPTR
CALL addr	виклик підпрограми за адресою addr
RET	повернення з підпрограми
RETI	повернення з переривання
NOP	No operation

1.4.6 Команди умовних переходів

Мнемоніка	Операція	Режими адресації			
		Dir	Ind	Reg	Imm
JZ rel	JMP if A=0	Accumulator only			
JNZ rel	JMP if A≠0	Accumulator only			
DJNZ <byte>, rel	декремент і перехід якщо не нуль	x		x	
CJNE A, <byte>, rel	JMP if A≠<byte>	x			x
CJNE <byte>#data,rel	JMP if <byte>≠ #data		x	x	

Відносний перехід здійснюють всі команди умовної передачі управління, а також команда безумовного переходу SJMP (Short - короткий). Всі команди відносного переходу двобайтові - другий байт містить відносну адресу переходу rel, що представлена 8-розрядним знаковим числом. При виконанні команд відносного переходу відбувається збільшення лічильника команд PC на 2, після чого до вмісту PC додається значення rel. Крім аналізу флагів регістру PSW та бітів внутрішнього ОЗП в командах умовного переходу може здійснюватись попереднє декрементування (буква D у мнемоніці команди) або порівняння (буква C у мнемоніці команди) операндів.

Команди повертання з підпрограм RET, RETI реалізують завантаження лічильнику команд PC вмістом двох комірок верхівки стеку (комірка з адресою SP завантажується до старшої половини PC, комірка з адресою SP-1 - до молодшої) та зменшення на 2 вмісту покажчика SP.

Виконання команди RETI припиняє обслуговування переривання, якщо воно мало місце, при використанні команди RET для обслуговування переривання вважається, що обслуговування переривання триває.

1.5 Призначення виводів мікроконтролера

Призначення однофункціональних виводів:

XTAL1,XTAL2 - виводи для підключення зовнішнього синхрогенератора,

Vss - загальний вивід,

Vcc - живлення +5 В,

Призначення багатофункціональних виводів:

1) основне призначення:

Reset - початкове встановлення,

P0.0-P0.7, P1.0-P1.7, P2.0-P2.7,P3.0-P3.7 - двонаправлені порти,

$\overline{PME}$ - вивід дозволу читання із зовнішньої пам'яті програм,

$ALE / \overline{PROG}$ - вивід дозволу фіксації адреси у зовнішній пам'яті,

$DEMA / Vpp$ - вивід режиму доступу тільки до зовнішньої пам'яті програм (DEMA=0),

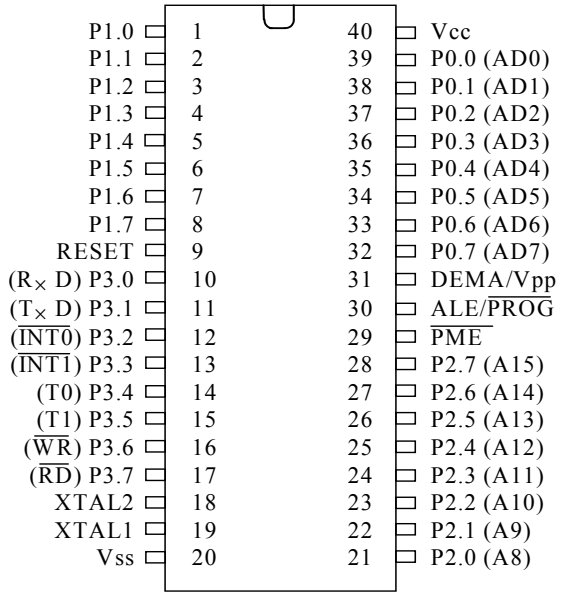


Рисунок 1.10

2) альтернативне додаткове призначення:

RESET- вхід напруги живлення при зменшеному енергоспоживанні,

P2.0-P2.7 - старший байт адреси звертання до зовнішньої пам'яті,

P0.0-P0.7 - молодший байт адреси звертання до зовнішньої пам'яті та байт даних, що зчитується із зовнішньої пам'яті або записується до зовнішньої пам'яті даних,

P3.0 - вхід послідовного інтерфейсу,

P3.1 - вихід послідовного інтерфейсу,

P3.2 - вхід зовнішнього переривання 0,

P3.3 - вхід зовнішнього переривання 1,

P3.4 - зовнішній вхід таймеру 0,

P3.5 - зовнішній вхід таймеру 1,

P3.6 - вихід дозволу запису даних до зовнішніх пристроїв через порт P0 при виконанні команд MOVX @Ri,A і MOVX @ DPTR,A,

P3.7 - вихід дозволу зчитування даних із зовнішніх пристроїв через порт P0 при виконанні команд MOVX A,@Ri і MOVX A,@ DPTR,

3) альтернативне при програмуванні внутрішнього ПЗП :

P1.0-P1.7 - молодший байт адреси внутрішнього ПЗП (PROM0-PROM7),

P2.0-P2.3 - старша тетрада адреси внутрішнього ПЗП (PROM8-PROM11),

P0.0-P0.7 - байт даних, що записується до ПЗП.

Порти P0,P1,P2,P3 є двонаправленими портами вводу-виводу і призначені для забезпечення обміну інформацією мікропроцесора із зовнішніми пристроями, утворюючи 32 лінії вводу-виводу. Кожен з портів містить фіксатор і буфер. Фіксатор являє собою 8-розрядний регістр, що має байтову і бітову адресації для встановлення розрядів програмним шляхом. Апаратним шляхом, наприклад через зовнішні виводи портів, встановлення розрядів фіксатора неможливе, через що в портах фіксуються тільки вихідні дані, але не фіксуються вхідні.

В режимі роботи із зовнішньою пам’яттю програм і даних порт P0 та задіяні лінії порту P2 не можуть бути застосовані для поширення вводу і виводу. При будь-якому звертанні до зовнішньої пам'яті регістр P0 завантажується байтом FF, втрачаючи дані, що раніше могли

бути занесені. Для використання лінії порту в якості входу відповідний біт фіксатора повинен бути попередньо встановлений в 1. Після сигналу скидання фіксатори портів P0 - P3 встановлюються у стан готовності до приймання вхідної інформації без додаткових зусиль. Будь-яку лінію вводу-виводу можна перевірити за допомогою команд умовного переходу: JB bit,rel; JNB bit,rel; JBC bit,rel.

Для зчитування інформації з портів є два способи : в командах, де порт виступає тільки операндом-джерелом, а результат формується в будь-якому іншому місці, дані знімаються безпосередньо з виводів портів. (Наприклад, ADD A,P1 - вміст акумулятора сумується з вмістом порту P1 із збереженням суми в акумуляторі). У випадках, де регістр порту є одночасно операндом-джерелом і місцем формування результату операції, реалізується зчитування вмісту регістру порту, зміна операнду, після чого результат перезаписують до регістру. Таким чином, ввід сигналу здійснюється не із зовнішніх виводів порту, а з регістра-фіксатора, що дозволяє уникнути неправильного зчитування раніш виведеної інформації.

Нижче наведені команди, що реалізують зчитування даних не з виводів, а з фіксатора порту :

ANL ad,A	ANL ad, #d	INC ad	JBC bit,rel	CLR bit
ORL ad,A	ORL ad, #d	DEC ad	MOV bit,C	SETB bit
XRL ad,A	XRL ad, #d	DJNZ ad,rel	CPL bit	

1.6 Робота з пам’яттю програм та даних

Режим роботи із внутрішньою пам’яттю програм встановлюється одиничним рівнем на виводі DEMA. По сигналу скидання і обнулення лічильника команд PC виконання програми, що зберігається в пам'яті, починається з команди, розміщеної за адресою 0000. В режимі роботи із внутрішньою пам’яттю порти P0 і P2 можна використовувати як порти вводу-виводу завдяки передачі адресів і даних пам'яті програм по внутрішнім магістралям мікроконтролера. Способи підключення пам’яті програм і даних стандартні, і їх слід дотримуватись при проектуванні пристроїв.

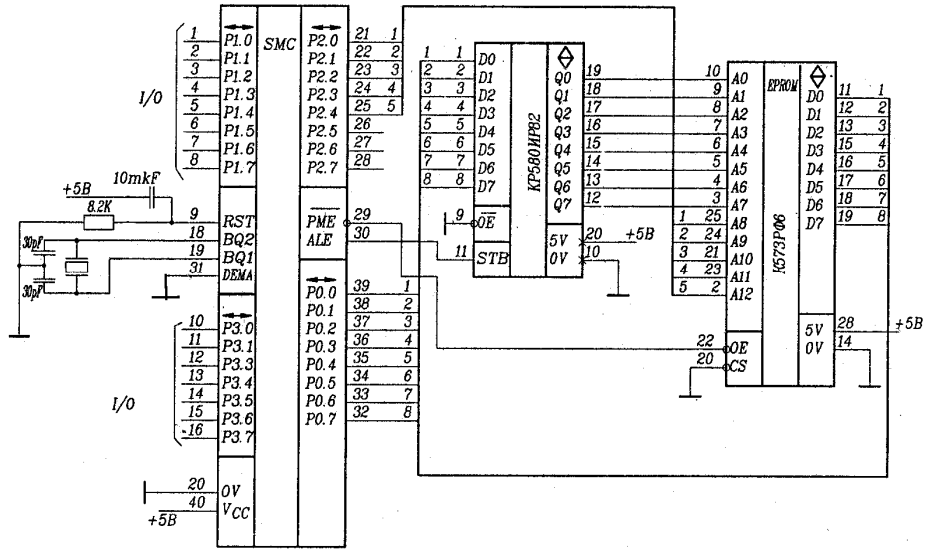


Рисунок 1.11 - Використання зовнішньої пам’яті програм

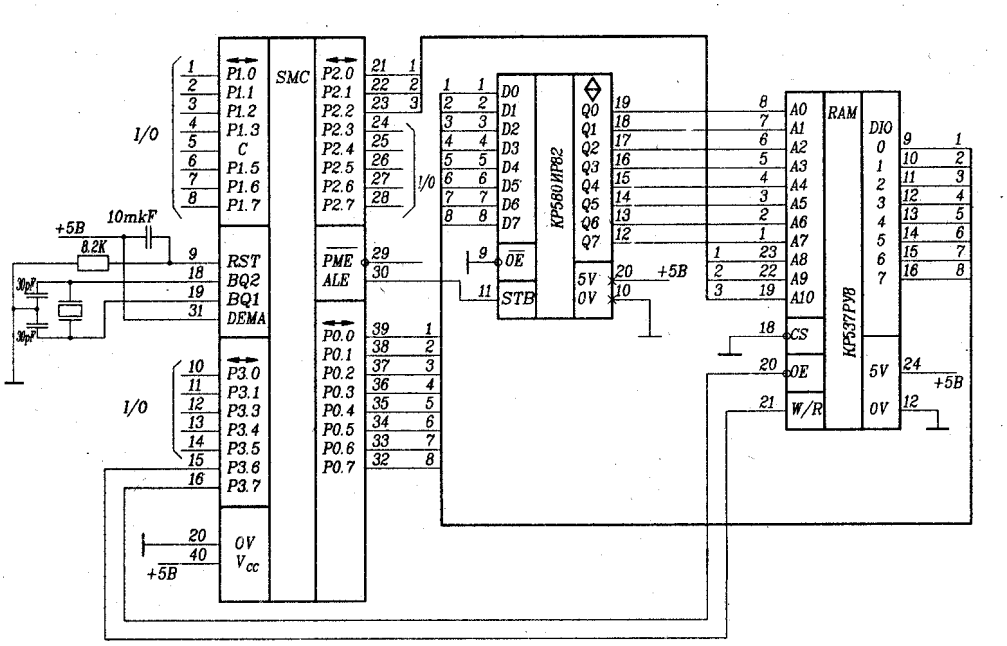


Рисунок 1.12 - Використання зовнішньої пам'яті даних

Режим роботи із зовнішньою пам'яттю програм встановлюється нульовим рівнем на виводі DEMA і застосовується при налагоджуванні програм та контролюванні роботи процесора, а також у випадку, коли внутрішньої пам'яті програм недостатньо (при суміщенні внутрішньої та зовнішньої пам'яті на вивід DEMA подається напруга високого рівня). Формування молодших розрядів адреси (A7-A0) до зовнішньої пам'яті і приймання коду команди із зовнішньої пам'яті здійснюється через порт P0 в режимі часового мультиплексування: спочатку з комірки пам'яті із адресою, встановленою в попередньому циклі, зчитується байт команди, виводи порту PO переходять в Z-стан (фаза S1F2), у фазах S2F1, S2F2 на виводах порту PO формується молодший байт адреси комірки зовнішньої пам'яті програм із наступним байтом команди, після чого виводи порту знову переходять у Z-стан (фази S3F1,S3F2). В залежності від структури команди формування байтів команди та молодших байтів наступної адреси у станах S4,S5 може і не відбуватись. Старші розряди адреси (A15-A8) видаються через порт P2. При цьому адреса фіксується заднім фронтом сигналу ALE, а команди приймаються по сигналу PME. В режимі роботи із внутрішньою пам'яттю програм сигнал PME не формується.

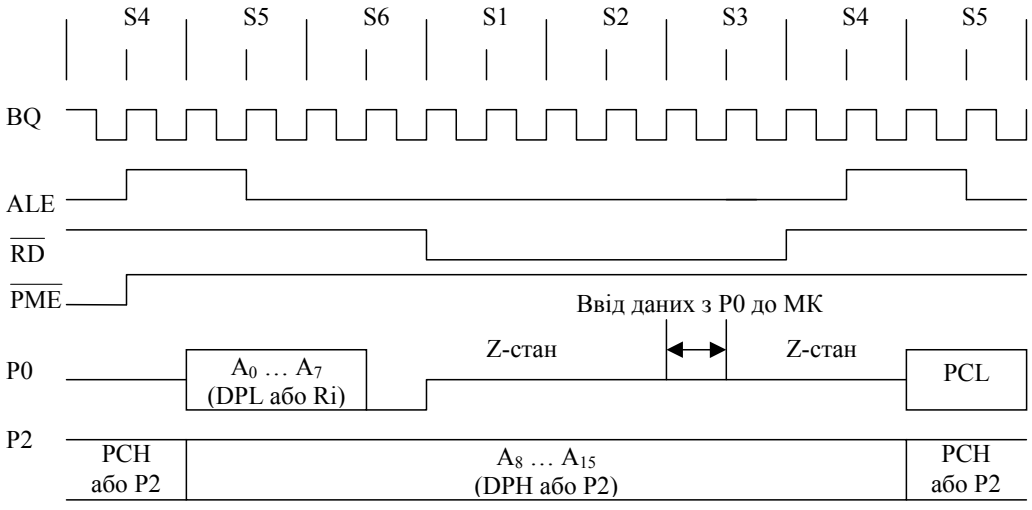


Рисунок 1.13 - Цикл читання із зовнішньої пам´яті даних

Звертання до зовнішньої пам'яті даних можливе при використанні 16-розрядної адреси (команди MOVX @DPTR,A, MOVX A,@DPTR) або 8-розрядної адреси (команди MOVX @Ri,A, MOVX A,@Ri).

Виконання однобайтових команд MOVX здійснюється на протязі двох циклів роботи процесора. У початковій фазі зчитується код команди MOVX, після чого виводи порту P0 переходять у Z-стан до закінчення фази S4F2. У фазах S5F1, S5F2, S6F1 на виводах порту формується молодший байт DPTR (для команд MOVX @DPTR) або стан регістру Ri (для команд MOVX @Ri).В цей час на виводах порту P2 сформовано старший байт DPTR або попереднє значення регістру P2 (для команд MOVX @DPTR, MOVX @Ri відповідно). Фронтом спаду сигналу ALE сформована адреса повинна бути зафіксована в зовнішніх буферних регістрах і подана на адресні входи мікросхеми ОЗП, що використовується в якості зовнішньої пам'яті даних.

В другому циклі виконання команд MOVX на протязі станів S1,S2,S3 реалізується формування сигналів читання (RD-для команд MOVX A,@DPTR, MOVX A,@Ri) або запису (WR - для команд MOVX @DPTR,A, MOVX @Ri,A). Звертання до зовнішньої пам'яті даних можливе лише за відсутності сигналу ALE, завдяки чому перший сигнал ALE у другому циклі команд MOVX (фази S1F2, S2F1) не формується. Читання даних із зовнішньої пам'яті здійснюється не на протязі дії всього сигналу RD, а тільки у фазі S3F1. Дані для запису до зовнішньої пам'яті даних формуються на виводах порту P0 на протязі всієї дії сигналу WR.

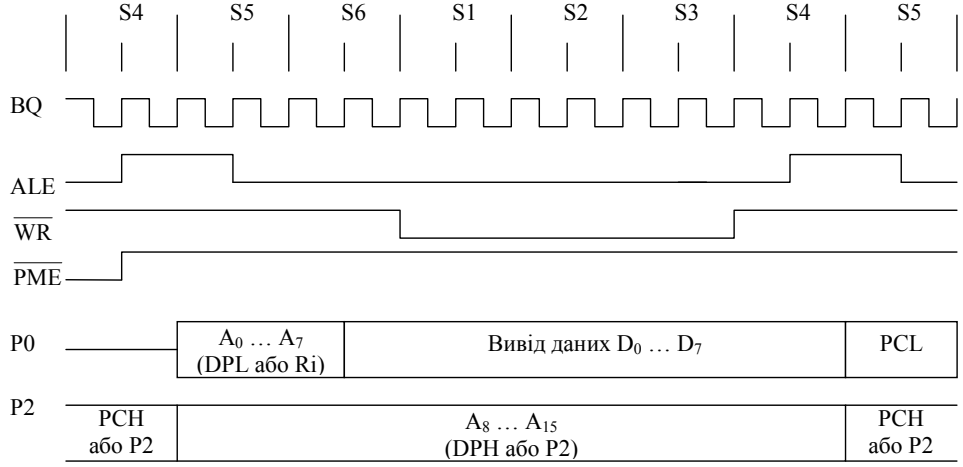
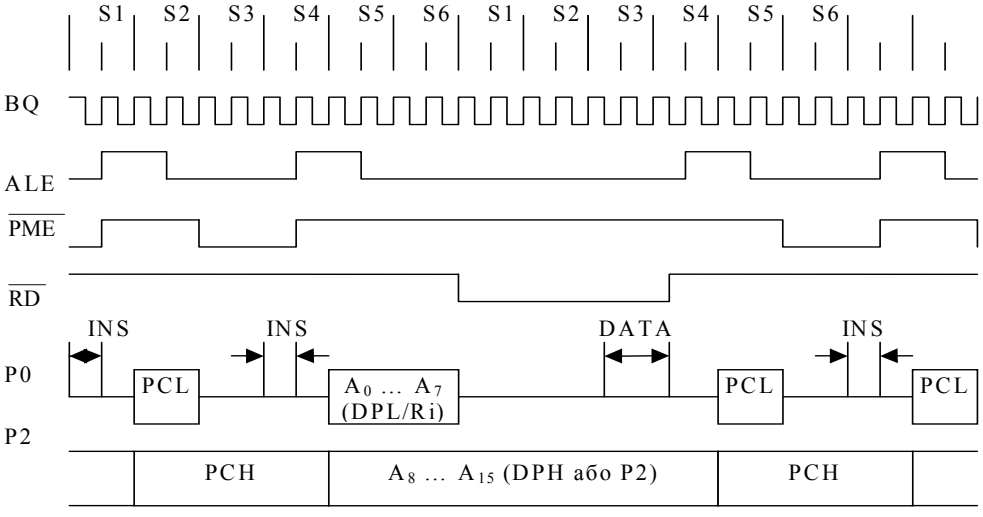


Рисунок 1.14 - Цикл запису у зовнішню пам´ять даних

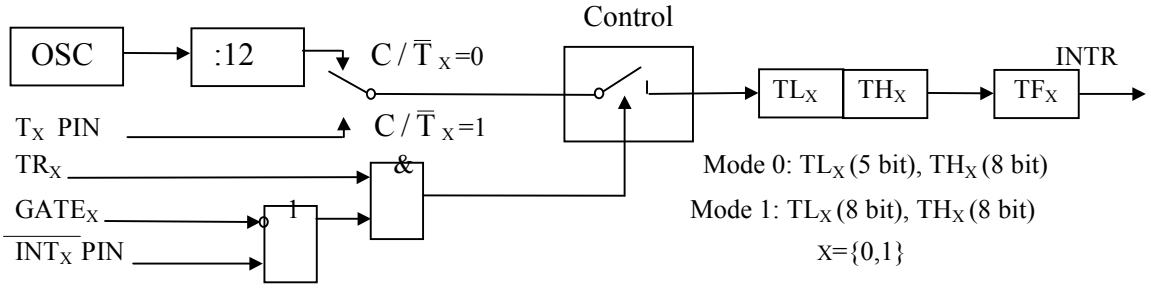


INS – ввід байта коду команди з P0
DATA - ввід байта даних з P0

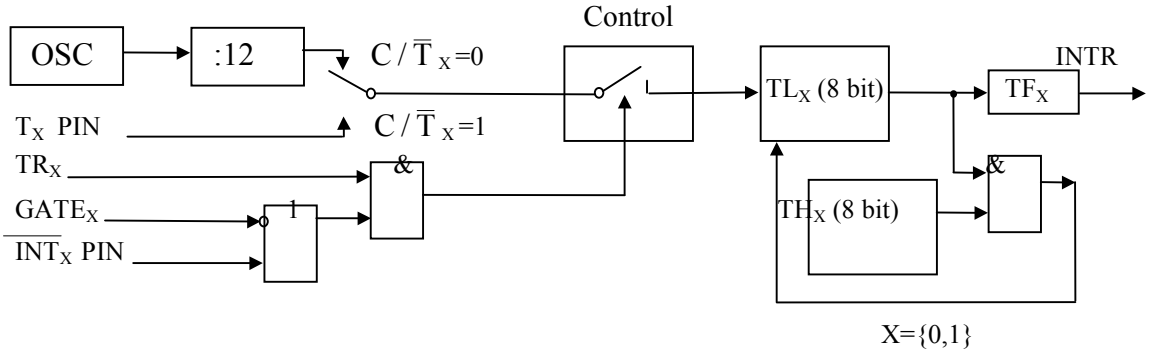
Рисунок 1.15 - Робота з зовнішньою пам´яттю програм з виконанням команди MOVX

1.7 Робота з таймерами MCS-51.

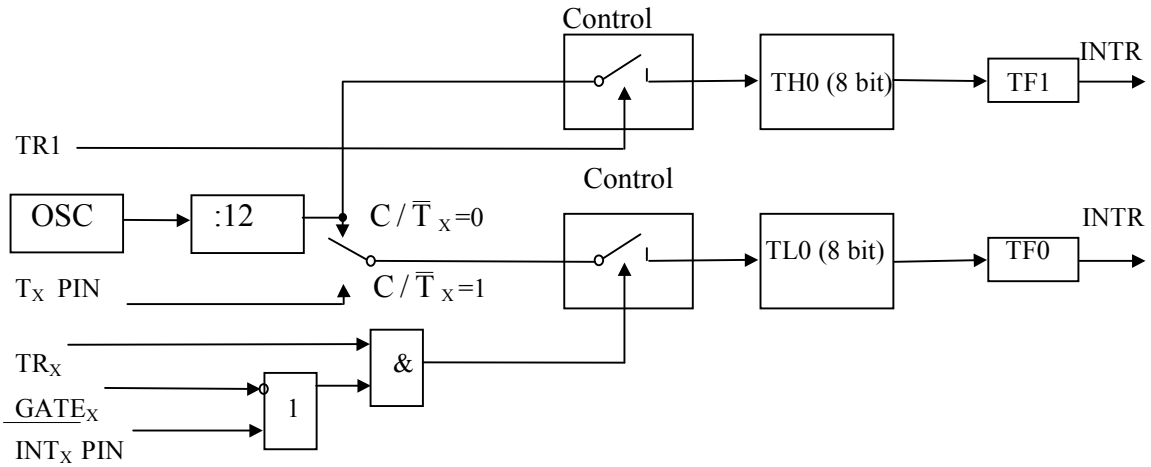
Мікроконтролер має 2 таймери-лічильники ТС0 і ТС1. Кожен із ТЛ складається з двох однобайтових регістрів TL і TH. Керування ТЛ здійснюється за допомогою регістрів TMOD, TCON. Можливі 4 режими роботи таймерів-лічильників:



а - логіка роботи Т/С0 и Т/С1 в режимах 0 і 1



б - логіка роботи Т/С0 и Т/С1 в режимі 2



в - логіка роботи Т/С0 в режимі 3

Рисунок 1.16 - Режими роботи таймерів

Деталізація регістрів:

TF1	TR1	TF0	IE1	IT1	IE0	IT0

— регістр TCON

TF1 - флаг переповнення таймера 1 TR1 - флаг дозволу таймера 1 (TR1)
TF0 - флаг переповнення таймера 0 TR0 - флаг дозволу таймера 0

IE1 - флаг переривання INT1 (1 - по спаду , 0 - по рівню 0)
IT1 - флаг вибору режиму переривання INT1
IE0 - флаг переривання INT0
IT0 - флаг вибору режиму переривання INT0

GATE	C/T1	M 1,1	M 0,1	GATE 0	CTO	M 1,0	M0,1

- регістр TMOD

GATE - флаг дозволу таймеру від зовнішнього пристрою по входу INT
C/T - флаг призначення пристрою (1 - лічильник, 0 - таймер)
M1 - M0 - флаг режиму роботи
(00 - режим 0,
01 - режим 1,
10 - режим 2,
11 - режим 3)

режим 0: TL - дільник на 32, TH - 8 - розрядний ТЛ.

режим 1: 16 - розрядний ТЛ (TH іTL з'єднані послідовно).

режим 2: TL - 8- розрядний автозавантажуваний ТЛ (при кожному переповненні до ТЛ перезавантажується код з регістру TH).

режим 3: для таймера TC1 - робота зупинена, для таймера TC0: TL0 - 8-розрядний ТЛ, що керується флагами TMOD3-TMOD0, TH0 - 8 - розрядний таймер (не лічильник), що керується флагами TMOD7-TMOD4).

В режимі таймера стан ТЛ змінюється з частотою F/12, де F – частота кварцового генератора, тобто таймер підраховує кількість циклів виконання програми. В режимі лічильника стан ТЛ змінюється по спаду сигналу на вході T0(T1). При GATE=0 запуск ТЛ здійснюється зразу при встановленні TR=1, тобто програмним шляхом. При GATE=1 запуск ТЛ здійснюється також при TR=1, але обов'язково при наявності одиничного рівня на зовнішньому вході INT, тобто ТЛ запускається зовнішніми пристроями.

1.8 Система переривань

Існують зовнішні переривання (по входах INT0 і INT1) і внутрішні переривання (від TC0, TC1, та блоку послідовного інтерфейсу: TI - переривання по закінченню передавання байта, RI - переривання по закінченню приймання байта).

Система переривань керується двома регістрами: IE і IP.

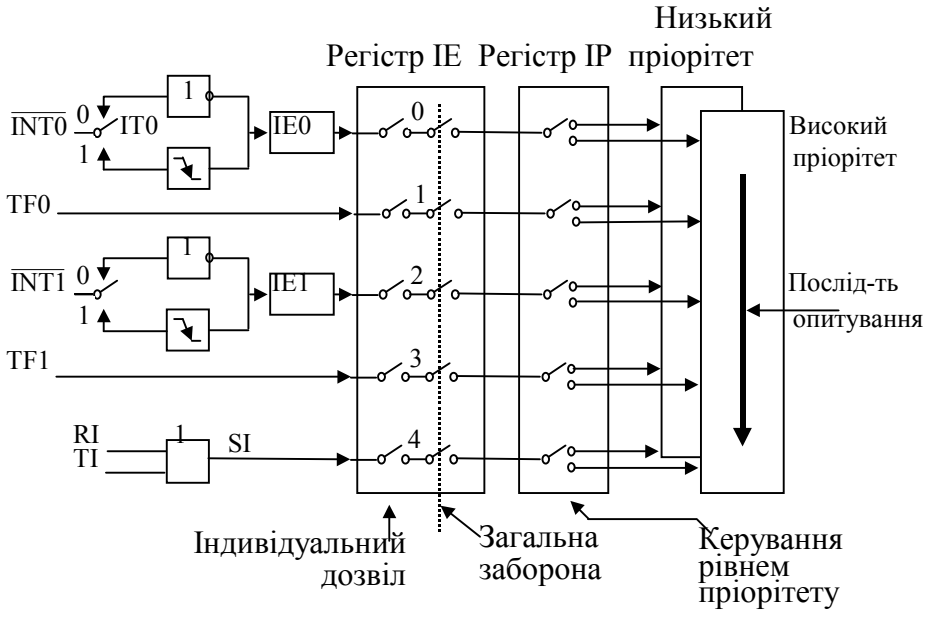
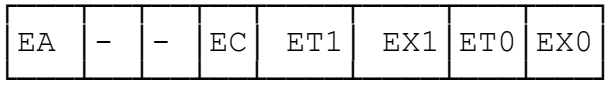
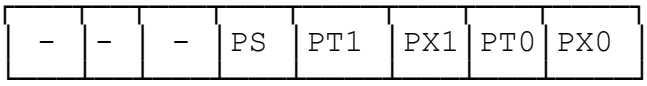


Рисунок 1.17 - Логіка роботи блоку переривань.

Біти регістру IE служать для дозволу/заборони переривань.



EA - флаг заборони всіх переривань.
ES - флаг дозволу переривань від послідовного інтерфейсу (TI або RI).
ET1 - флаг дозволу переривань від таймеру TC1.
EX1 - флаг дозволу зовнішніх переривань INT1.
ET0 - флаг дозволу переривань від таймеру TC0.
EX0 - флаг дозволу зовнішніх переривань INT0.



PS - флаг пріоритету переривань від ПІ (TI або RI) .
PT1- флаг пріоритету переривань від таймеру TC1.
PX1 - флаг пріоритету зовнішніх переривань INT1.
PT0 - флаг пріоритету переривань від таймеру TC0.
PX1 - флаг пріоритету зовнішніх переривань INT0.

При рівності пріоритетів переривань в регістрі IP обробка переривань здійснюється в наступному порядку:
1) INT0
2) TF0
3) INT1
4) TF1
5) TI, RI

Адреси векторів переривань (тобто адреси команд переходу на підпрограму обслуговування переривання):

1.Зовнішнє переривання INT0 - 0003.
2.Переривання таймеру TC0 - 000B.
3.Зовнішнє переривання INT1 - 0010.
4.Переривання таймеру TC1 - 001B.
5.Переривання послідовного інтерфейсу TI або RI - 0023.

Флаги переривань розміщені: по входах INT0, INT1 - в регістрі TCON, від TC0, TC1 - в регістрі TCON, від послідовного інтерфейсу - в регістрі SCON.
Активним рівнем сигналу на входах INT0, INT1 може бути:
1) фронт спаду (якщо флаг IT0, IT1 регістру TCON дорівнює 1);
2) нульовий рівень (якщо флаг IT0, IT1 = 0);
В другому випадку МК після визначення переривання по входах INT0, INT1 повинен (якщо це можливо) забезпечити встановлення в 1 сигналу INT0, INT1 для попередження помилкового багаторазового переходу на обробку переривань.
Перехід на підпрограму обробки переривань супроводжується наступними діями:
1. Занесення до стеку лічильника команд, де зберігається адреса команди наступної за поточною, яка буде виконуватись після повернення з підпрограми;
2. Блокування всіх переривань поточного та нижніх рівнів пріорітетів;
3. Відбувається або не відбувається скидання власного флага переривання.

Флаги переривань від TC0 і TC1 скидаються автоматично при переході на підпрограму обробки переривань. Флаги переривань по входах INT0, INT1 скидаються автоматично у випадку активного сигналу на цих входах у вигляді фронту спаду. При активному сигналі у

вигляді нульового рівня флагці IE0 і IE1 повинні скидатись програмно всередині підпрограми обробки переривань. Флаги переривання встановлюються незалежно від заборони переривання регістром IE. Якщо флаг був встановлений при забороненому перериванні, то при наступному дозволі цього переривання може відбутися помилковий перехід на його обробку, хоча переривання давно зняте. Повертання з підпрограми переривання здійснюється командою RETI по якій:

1. Із стеку до лічильника команд повертається адреса наступної команди (так само як і після команди RET - повертання із звичайної підпрограми).
2. Зняття блокування переривань поточного і нижчих рівнів.

1.9 Блок послідовного інтерфейсу

Блок послідовного інтерфейсу (БПІ) призначений для обміну інформацією з іншими мікроконтролерами при роботі в багатопроцесорних мережах. Блок ПІ має два регістра: SBUF - буферний і SCON - регістр керування. Крім того, задіяний старший біт SMOD регістру PCON.

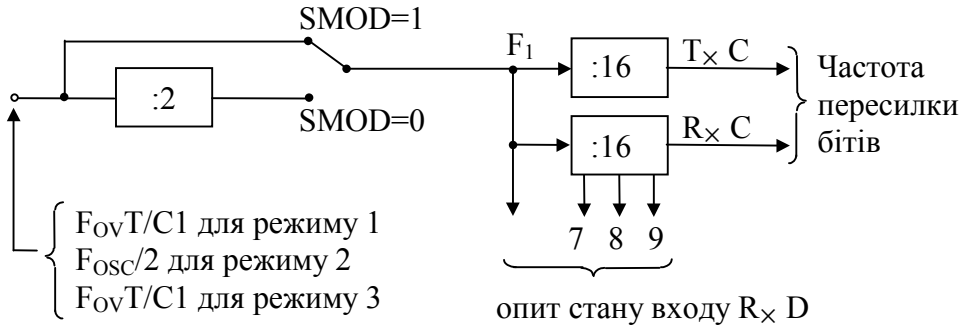


Рисунок 1.18 - Формування робочої частоти передачі/прийому БПІ

Регістр SBUF призначений для тимчасового зберігання байту, що передається або приймається. Команда, що записує байт до регістру SBUF, автоматично реалізує початок передавання цього байту молодшими бітами вперед.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

— регістр SCON

SM0, SM1 - флаги режиму роботи ПІ (00- режим 0; 01- режим 1;10- режим 2; 11- режим 3).

режим 0: 8 біт даних передається/приймається через вивід RxD, на виводі TxD формуються імпульси зсуву, що супроводжують кожний біт із частотою F/12 (F - частота резонатора).

режим 1: 10 біт (старт - біт 0, 8 біт даних, стоп - біт 1) передаються через RxD / приймаються через TxD, швидкість обміну визначається таймером TC1.

режим 2 : 11 біт (старт - біт 0, 8 біт даних, 9- ий додатковий програмований біт даних, стоп - біт 1) передаються/приймаються через виводи RxD/TxD, частота обміну дорівнює F/32*12 (при SMOD =1) або F/64*12 (при SMOD=0).

режим 3: аналогічний режиму 2, але частота обміну задається таймером TC1.

SM2 - флаг заборони приймання даних, в яких 9 - й біт дорівнює 0 .
REN - флаг дозволу приймання даних.
TB8 - 9-й біт, що буде передаватись в режимі 2 і 3.
RB8 - 9-й біт, що приймається в режимі 2 і 3.

TI - флаг переривання передавача, встановлюється при закінченні передавання байту.
RI - флаг переривання приймача , встановлюється при закінченні приймання байту.

В режимах 1, 3 частота обміну залежить від частоти переповнення таймеру TC1 Fov і дорівнює (SMOD+1)/32*Fov.
Стандартні швидкості передавання сигналів по послідовному інтерфейсу RS232 - 1200, 2400, 4800, 9600, 19200 бод. В режимі 2 МК51 працює з частотою 31250 Гц, тобто цей режим може бути використаний тільки в нестандартних локальних мережах. В режимах 1 та 3 швидкість передачі визначається частотою переповнення таймера 1: Fov=(F/12)/(256 - TH) і може бути встановлена рівною одній із стандартних при умові, що частота тактування мікроконтролера складає 11.059 МГц. При іншій частоті тактування допустиме значення похибки задання швидкості обміну отримується переважно тільки для низьких швидкостей.

1.10 Режими роботи МК

1. Програмування внутрішньої пам'яті програм .
2. Верифікація ВПП - після програмування в режимі верифікації існує можливість перевірки занесеної інформації до кожної комірки пам'яті програм.
3. Захист ВПП від несанкціонованого доступу шляхом запису спеціального біту захисту. Після операції захисту читання ВПП можливе тільки на мікросхемотехнічному рівні, тобто шляхом зрізання кристалу.
4. Скидання - початкова ініціалізація контролера. Скидаються в нуль всі регістри спеціальних функцій, крім регістру SP (початкове значення 07), порти P0, P1, P2, P3 встановлюються в значення FF. Значення ВПД є довільним.
5. Режим холостого ходу. В цей режим МК переводиться встановленням в 1 біту IDL в регістрі PCON. При цьому продовжує працювати внутрішній тактовий генератор, значення всіх регістрів та комірок пам'яті зберігається. Виведення з режиму холостого ходу здійснюється сигналом RESET або надходженням сигналу на вхід зовнішнього переривання.

SMOD	-	-	-	GF1	GF0	PD	IDL

— регістр PCON

SMOD - флаг швидкості передавання ПІ.
GF1, GF0 - флаги користувача загального призначення.
PD - флаг зменшеного енергоспоживання.
IDL - флаг холостого ходу.

6.Перехід в режим зменшеного енергоспоживання здійснюється встановленням в 1 біту PD. В цьому режимі мікроконтролер споживає в 20 раз менший струм, напруга живлення - 2.5 - 3В, працює тактовий генератор. З режиму зменшеного енергоспоживання МК виводиться сигналом RESET, внаслідок чого втрачається інформація в регістрах спеціальних функцій, але не змінюється стан ВПД, тобто перед переведенням МК в режим зменшеного енергоспоживання вміст регістрів спеціальних функцій при необхідності слід здублювати у ВПД.

1.11 Розробка програм для MCS-51

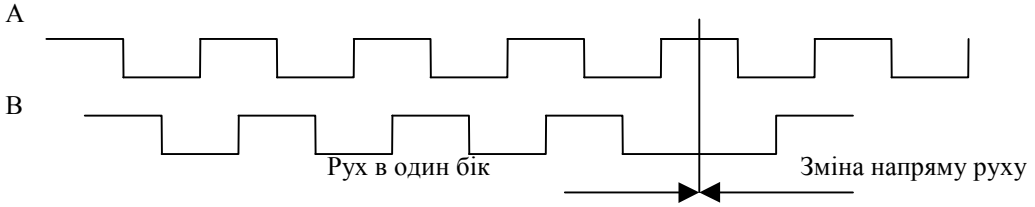
Розробка програмного забезпечення для мікроконтролерів здійснюється за допомогою мови Асемблера та мов високого рівня: (наприклад Сі або Паскаль). Внаслідок обмеженої швидкодії базових моделей МК (за сучасними вимогами), розробку програмного забезпечення рекомендується проводити мовою Асемблера. Тільки для простих задач і для задач, які не вимагають високої швидкодії (час обробки даних >1 сек), але які одночасно вимагають деяких математичних обчислень, можна використовувати мови високого рівня.
Для прикладу, розглянемо побудову інтелектуального вимірювача лінійних переміщень на базі МК MCS-51. Як базову модель виберемо МК Atmel 89C51, з максимальною тактовою частотою 24 МГц.

Загальні задачі МК:

- вимірювання переміщення із врахуванням його напряму.
- вимірювання часових інтервалів з роздільною здатністю < 10 мс.
- визначення стартової точки - початку мірного інтервалу
- передача даних до ЕОМ по послідовному каналу

Прецизійне вимірювання переміщення зручно здійснити за допомогою оптичного кодера, який формує на виході два зсунуті по фазі на 90^0 сигнали напругою +5В. Тип такого давача: А178-А5. Кількість імпульсів при одному оберті ротора на 360^0 складає 1024 імп.

Форма вихідних сигналів, при роботі А178-А5 наступна:



Видно, що по фронту сигнала фази А при русі в один бік можна прочитати лог. "0" фази В, а при зміні напрямку руху - лог."1" фази В. Визначення часових інтервалів може бути реалізоване за допомогою внутрішнього таймера, а визначення стартової точки за допомогою геркона1. Передача даних у послідовному форматі можлива за допомогою використання SPI-подібного послідовного інтерфейсу. Приймання даних на ЕОМ можливе за допомогою емуляції цього інтерфейсу LPT-портом ЕОМ.

Реалізація алгоритму роботи здійснена мовою Асемблера (Franklin IDE).

```
 NAME Sensor_Position;
       STACK        SEGMENT IDATA; Визначення розміщення стеку
                    RSEG STACK;        в сегменті внутрішніх даних
                    DS 10;             розмір стеку
DOUT        EQU P1.1;     Призначення контакту для виводу послід. даних
DRDY        EQU P1.4;     Призначення контакту для сигналу готовності даних
SCLK        EQU P1.2;     Призначення контакту для сигналу тактування даних
Dir         EQU F0;       Призначення ознаки напряму
DIR_Pin     EQU P3.2;     Контакт для сигналу апаратного визначення напряму
      VAR  SEGMENT DATA;      Сегмент даних
       RSEG VAR;              Опис змінних
CX    : DS 1;                 Змінні розміром 1 байт
Tm0   : DS 1
Tm1   : DS 1
status  : DS 1;               Змінна статусу програми
dirChg: DS 1

CSEG AT 000H                  ; Сегмент коду. Розміщення основної програми
LJMP MAIN_PROC                ; Перехід до основного програмного циклу
MAIN_PROC SEGMENT CODE
RSEG MAIN_PROC                ;Старт після RESET
USING 0                       ;Використовувати банк 0
  MOV SP,#STACK-1             ;Завантажити регістр покажчика стеку
```

1 Геркон - "ГЕРметичний КОНтакт" - вид вимикача, що являє собою контакти, запаяні в скляну колбу, заповнену інертним газом. Включення вимикача відбувається під дією магнітного поля. Герметичність пристрою забезпечує його стійкість до дії шкідливих атмосферних проявів.

```
  MOV status,#00              ;Заборонити випадкові переривання

  SETB DIR                    ; Встановити признак напрямку руху пристрою - ВГОРУ
  MOV dirChg,#00h             ; Встановити признак зміни напряму руху
  CLR TR1                     ; Заборонити роботу таймера 1
  CLR TR0                     ; Заборонити роботу таймера 0
  MOV TMOD,#01010001B ; T1-лічильник, T0-таймер в РЕЖИМІ 1
  MOV TH1,#00H                ; Початкове значення старшої тетради T1
  MOV TL1,#00H                ; Початкове значення молодшої тетради T1
  CALL Init_Timer             ; Виклик процедури ініціалізації таймера
  MOV Tm0,#00                 ; Ініціалізуємо змінні поточного часу
  MOV Tm1,#00
  SETB EA                     ;Загальний дозвіл переривань
  SETB EX1                    ;Дозвіл переривання...int1
  SETB EX0                    ;Дозвіл переривання...int0
  SETB IT1                    ;Переривання 1 по фронту імпульса
  SETB IT0                    ;Переривання 0 по фронту імпульса
  SETB ET1                    ;Дозвіл переривання T1
  SETB ET0                    ;Дозвіл переривання T0
  SETB TR1                    ;TC1 - дозвіл лічильника
  SETB TR0                    ;TC0 - дозвіл таймера
LOOP:                         ; Основний цикл
          JNB dir_pin,DownL
    MOV a,dirChg
    CJNE a,#00,DownL          ;Якщо зафіксовані зміни ... оминути…
    SETB dir                  ;dir=1 -> Рух вгору
          CLR TR1                     ;Зупинка підрахунку імпульсів
    CLR DRDY                  ;Дані не готові - сигнал для ЕОМ
    CLR C                     ;Очистимо флаг переносу
    MOV A,R2
    SUBB A,TL1
    MOV R2,A
    MOV A,R3
    SUBB A,TH1
    MOV R3,A                  ;(R3 R2)=(R3 R2)-(TH1 TL1)
    MOV DirChg,#01h           ;Зміна напрямку руху
    MOV TH1,#00
    MOV TL1,#01               ;Додаткове значення до лічильника імпульсів
    SETB TR1                  ;Старт лічби
    SETB DRDY                 ;Дані готові для передачі до ЕОМ!
DownL:
    MOV a,status;             Аналіз поточного статусу
    CJNE a,#01,LOOP           ;Якщо програмний_режим=1 тоді основний цикл
  ;-----------Передача даних, якщо наявний запит даних
    MOV A,R1                  ;                 R1R0- час в мілісекундах
          CALL SEND_BYTE
    MOV A,R0
    CALL SEND_BYTE            ;Завершимо видачу 16-розрядного значення часу
    MOV A,R3                        ;R3R2- переміщення в квантах
    CALL SEND_BYTE
    MOV A,R2
    CALL SEND_BYTE            ;Завершимо видачу 16-розрядного значення  переміщення
    CLR DRDY    ; Дані НЕ ГОТОВІ - сигнал для РС
```

```
    JMP LOOP                ; Кінець основного циклу


;--------------- ПІДПРОГРАМА ПЕРЕДАЧІ БАЙТУ ДО  PC  -----
SEND_BYTE:
            MOV CX,#08D ; Кількість бітів у байті - 8
LOOP_B:
 JB SCLK,$ ;Дочекаємось 0 тактового імпульсу ($-Адреса розміщення команди)
 JNB SCLK,$          ; Дочекаємось 1 тактового імпульсу
 RLC A               ; Зсунемо байт даних через флаг переносу- в С біт даних
 JNC ZERO            ; якщо біт даних=0 тоді перейдемо….
 SETB DOUT           ; ...інакше 1 на контакті видачі даних
 JMP NEXT    ; оминемо обнулення DOUT
ZERO: CLR DOUT    ;…до 0 на контакті видачі даних
NEXT: DJNZ CX, LOOP_B; обробимо всі біти
 RET;           повернемось із підпрограми
;--------------- ПІДПРОГРАМА ІНІЦІАЛІЗАЦІЇ ТАЙМЕРА Т0 -------
Init_timer:
CLR TR0             ;  Заборона таймера
MOV TH0,#0faH       ;  1 мсек для кварцу 14.318Mhz
MOV TL0,#0deH       ;  ---//----
SETB TR0            ; - Дозвіл роботи таймера ТС0
RET                 ;  Повернення з підпрограми
;Визначення векторів переривань
;---------------INT0--------
 CSEG AT 003H
 LJMP I_Int0        ; Визначає напрям руху пристрою
;---------------INT1--------
 CSEG AT 013H
 LJMP I_Int1         ; Обробка зовнішнього ЗАПИТУ ДАНИХ - передача даних
;---------------TC0---------
CSEG AT 000Bh
LJMP I_T0
;--Підпрограма обробки переривання по переповненню таймера 0-
I_T0_SEG SEGMENT CODE
RSEG I_T0_SEG
USING 0
I_T0:
CLR DRDY            ; Дані не готові ! Попередження можливому читанню даних.
CLR EA              ; Заборона переривань
MOV A,Tm0
ADD a,#01           ; Збільшуємо на 1 змінну часу
MOV Tm0,a
MOV a,Tm1
ADDC a,#00
MOV Tm1,a           ; Збільшуємо на 1 16-розрядну змінну часу
CALL init_Timer     ; Переініціалізуємо таймер
SETB DRDY           ; Дані готові
SETB EA             ; Дозвіл переривань
RETI;               ; Повернення з переривання
;---------------------------- Визначає рух ВНИЗ
I_Int0_SEG SEGMENT CODE
RSEG I_Int0_SEG
```

```
USING 0
I_Int0:
CLR TR1                 ; Зупинка таймера
CLR DRDY                ; ДАНІ НЕ ГОТОВІ !
MOV A,TL1
ADD A,R2
MOV R2,A
MOV A,TH1
ADDC A,R3
MOV R3,A                ; (R3 R2)=(R3 R2)+(TH1 TL1)
CLR Dir                 ;  Напрям руху = ВНИЗ
MOV DirChg,#00          ;  Дозвіл контролювання напряму
MOV TH1,#00             ;
MOV TL1,#01             ;  Початкові значення для лічильника
SETB TR1                ;Старт лічби
SETB DRDY               ;Дані готові для передачі
RETI                    ;Вихід з підпрограми
;---------------------- Підпрограма обробки запиту даних
I_Int1_SEG SEGMENT CODE
RSEG I_Int1_SEG
USING 0
I_INT1:
  RDCounter:
MOV a,TH1
MOV R2,TL1
CJNE A,TH1,RDCounter
JNB DIR,SK1
ORL A,#128d             ;Старший біт даних - біт напрямку руху
SK1:   MOV R3,A                 ;R3:R2 - значення переміщення
  RDTime:
MOV A,TM1
MOV R1,TM0
CJNE A,TM1,RDTime
MOV R0,A                ;R1:R0 - Поточний час
MOV status,#01d         ;Наявний виклик Int 1 = зовнішній запит даних!
RETI                    ;Повернення із підпрограми

END ;                           Кінець асемблер-програми
```

2. 16-розрядні мікроконтролери серії XA

2.1 Структура CPU

CPU мікроконтролера XA складається з наступних функціональних блоків:

1) блок вибірки і декодування команд
2) блок виконання команд
3) 16-розрядний арифметико-логічний пристрій
4) контролер виключень
5) регістри загального призначення та спеціальних функцій
6) осцилятор
7) контролер переривань
8) пам’ять програм
9) пам’ять даних
10) інтерфейс з зовнішньою шиною
11) порти вводу-виводу

Блоки 1-5 входять до ядра CPU, що наявне у всіх контролерах XA незалежно від модифікації. Блоки 7-11 можуть бути відсутні в окремих модифікаціях контролера.

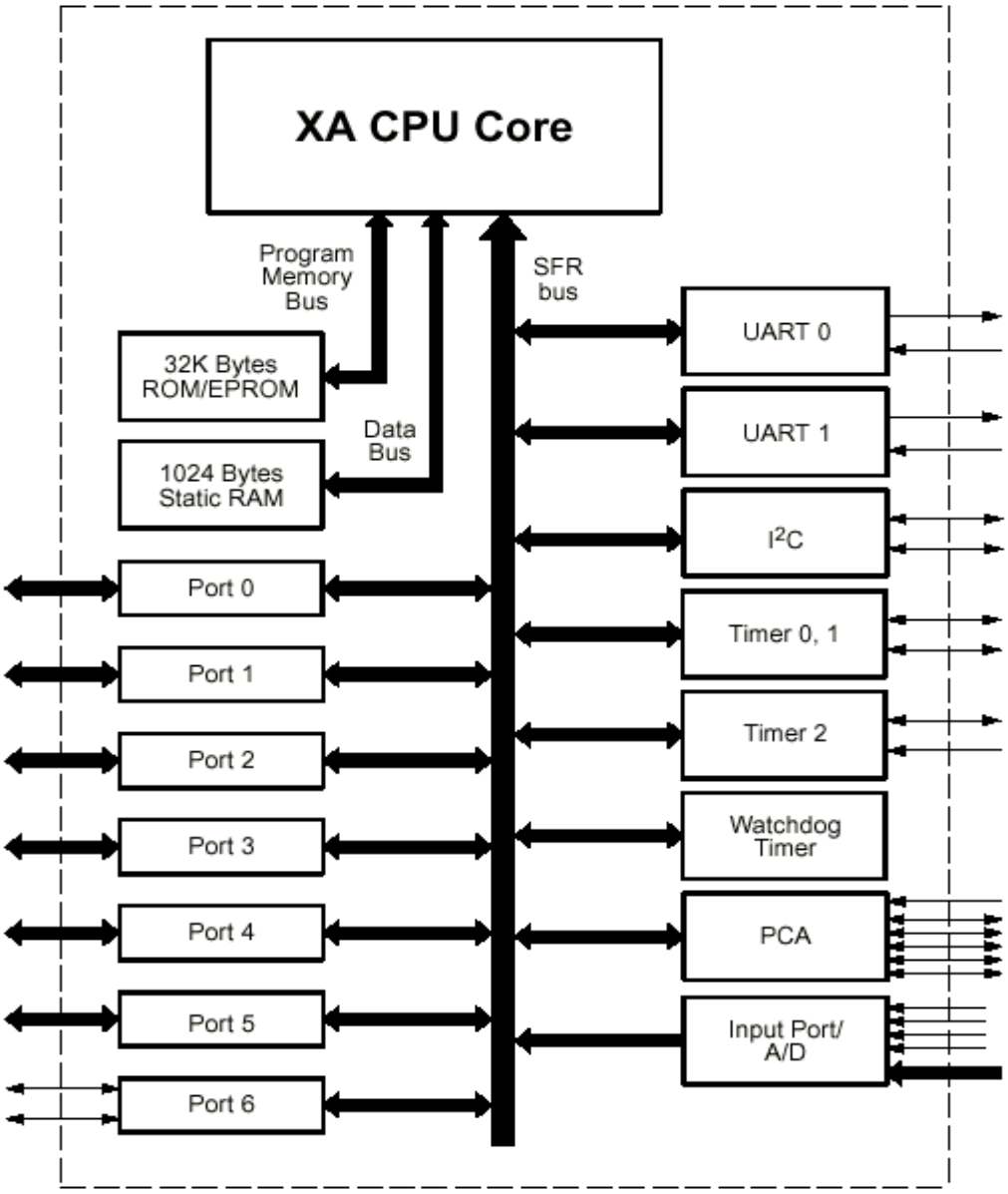


Рисунок 2.1 - Структура XA-S3

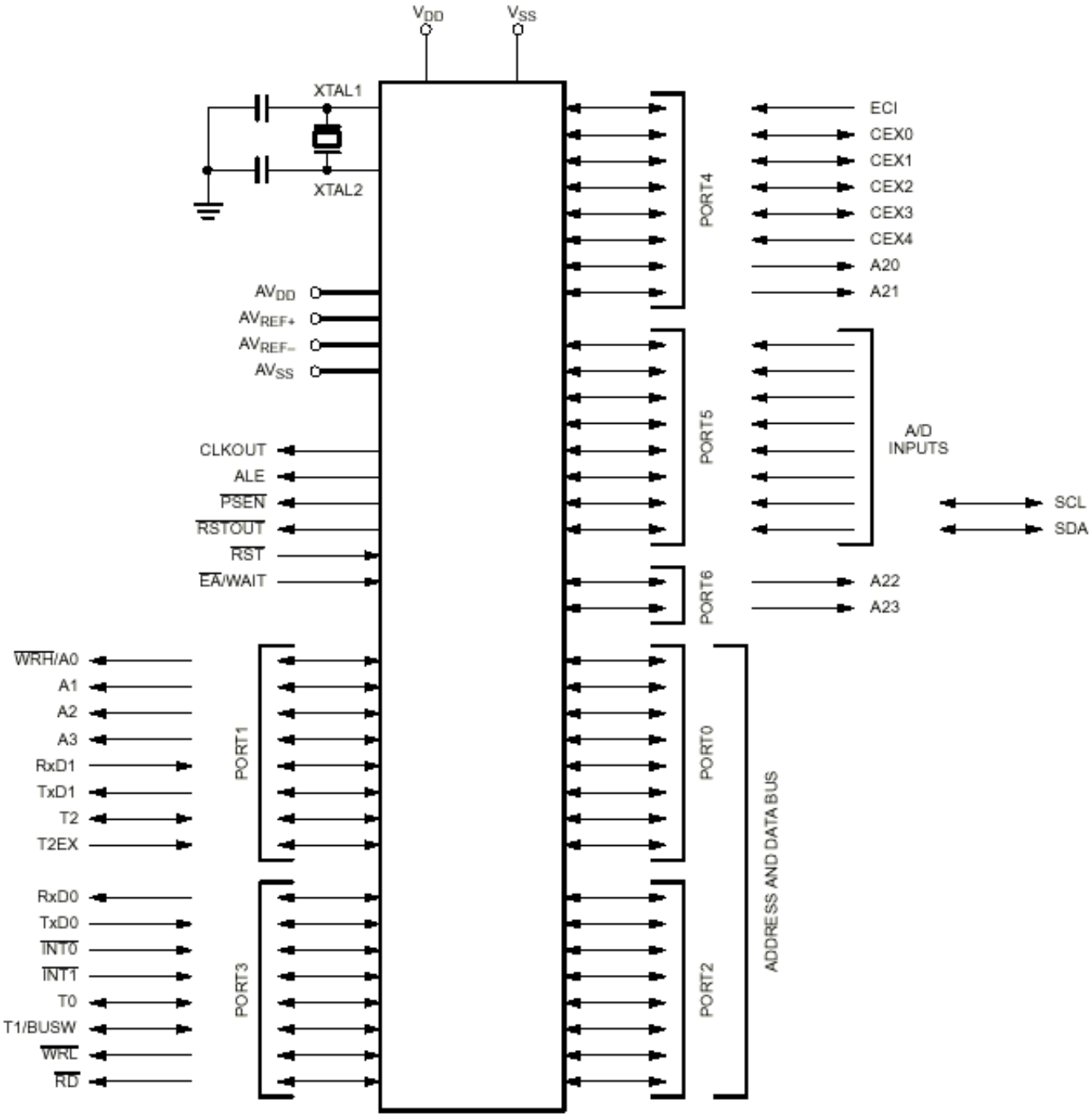


Рисунок 2.2 - Призначення виводів XA-S3

2.1.1 Регістр статуса PSW

Регістр статуса є двобайтовим. У молодшому байті містяться флаги, які відображають результати виконання команд, у старшому – біти, що встановлюють режими роботи контролера. В режимі користувача у старшому байті доступними для запису є тільки біти вибору банку регістрів (інші біти доступні тільки в системному режимі).

Примітка. XA містить також регістр спеціальних функцій PSW51, який аналогічний регістру PSW контролера 80C51.

Молодший байт PSW (адреса 400h)

C	AC	-	-	-	V	N	Z

C – флаг переносу, відображає результати арифметичних і логічних операцій (ADD, ADDC, CMP, CJNE, SUB, SBB, ASL, ASR, LSR, RLC, RRC). Скидається після виконання команд множення і ділення.
AC – флаг додаткового переносу.
V – флаг переповнення. Встановлюється по результатах виконання операцій ADD, ADDC, CMP, NEG, SUB, SBB. Встановлюється в 1: при діленні на 0; якщо результат ділення має більшу розрядність, ніж регістр результату; якщо результат множення має більшу розрядність, ніж множники.
N – флаг знаку.
Z – флаг нуля.
Флаги N і Z встановлюються по результатах виконання арифметичних операцій та команд пересилки даних (крім PUSH, POP, SEXT, LEA та XCH).

Старший байт PSW (адреса 401h)

SM	TM	RS1	RS0	IM3	IM2	IM1	IM0

SM – флаг вибору системного режиму (SM=1) або режиму користувача (SM=0). У режимі користувача обмежений доступ до деяких регістрів та областей пам’яті. Наприклад, біти SM, TM і IM3:IM0 можуть бути змінені тільки в системному режимі.
TM – флаг покрокового режиму (використовується при відладці)
RS1:RS0 – біти вибору банку регістрів:

RS1	RS0	банк
0	0	банк 0
0	1	банк 1
1	0	банк 2
1	1	банк 3

IM3:IM0 – біти пріоритету виконуваної програми. При виникненні маскованого переривання, яке є дозволеним, ці біти порівнюються з значенням пріоритету переривання, і якщо пріоритет переривання є більшим, то виконується перехід на підпрограму обробки переривання. В противному випадку переривання може бути оброблене після зниження рівня пріоритету поточної програми.

При скиданні в PSW завантажується значення, яке знаходиться у векторі переривання за адресою 0 у пам’яті програм. Рекомендується задавати значення бітів SM=1 (системний режим) та IM3:IM0 = 1111 (тобто ініціалізація – процес з найвищим пріоритетом). Після виконання дій, пов’язаних з ініціалізацією, пріоритет, як правило, можна зменшити, щоб дозволити виконання інших задач.

2.1.2 Регістр конфігурації SCR *(адреса 440h)*

-	-	-	-	PT1	PT0	CM	PZ

PZ – біт вибору режиму адресації. При PZ=1 вибраний режим 16-бітної адресації (режим “нульової сторінки”). Цей режим доцільно задавати, якщо не потрібно використовувати більше, ніж 64 Кбайт пам’яті. При PZ=0 вибраний режим 24-бітної адресації.
CM – біт встановлення режиму сумісності з 80C51. Якщо цей біт встановлений, то 32 байти пам’яті даних в кожному сегменті заміняються чотирма банками регістрів, і при непрямій адресації при посиланні на R0 використовується R0L, при посиланні на R1 використовується R0H.

PT1 та PT0 задають значення коефіцієнта подільника частоти осцилятора Fosc для тактування периферійних пристроїв (таймерів, послідовного інтерфейсу):

PT1	PT0	Частота тактування
0	0	Fosc/4
0	1	Fosc/16
1	0	Fosc/64
1	1	зарезервовано

2.1.3 Стек

Стеків є два – системний стек та стек користувача. Стек користувача може розміщуватися у довільному просторі пам'яті даних, системний стек – тільки в нульовому сегменті. При стековій адресації використовується сегментний регістр DS та покажчик стеку SP, який знаходиться в регістрі R7. У кожен момент часу доступний тільки один з покажчиків стеку – користувача USP (user stack pointer) або системного стеку SSP (system stack pointer), в залежності від того, в якому режимі працює контролер. У режимі користувача не можна змінювати значення регістру DS, тобто зміна сегменту, в якому розташовується стек, можлива тільки при переключенні задач в системному режимі. Таким чином, кожна задача працює в окремому сегменті і має власний стек.

На відміну від стека мікроконтролера 80C51, при занесенні даних у стек покажчик стеку зменшується на 2, а при читанні даних зі стеку – збільшується на 2.

Стек оперує з двобайтовими даними. Якщо у команді PUSH вказується однобайтовий операнд, наприклад, R1L, то у стек за адресою SP+1 записується значення з регістру R1L, а за адресою SP+2 нічого не записується, однак покажчик стеку все рівно зменшується на 2. Якщо однобайтовий операнд вказується у команді POP, наприклад, R1H, то до R1H записується молодший байт (який розміщувався у стеку за адресою SP), і покажчик стеку збільшується на 2.

Якщо покажчик стеку SP містить непарне значення, молодший біт ігнорується, тобто відбувається звертання до слова, розташованого починаючи з меншої парної адреси.

Команди PUSH і POP не впливають на стан флагів регістру PSW.

Переповнення стеку фіксується у випадку, коли покажчик стеку декрементується від 80h до 7Eh. При цьому генерується переривання типу виключення з рівнем пріоритету 2 (адреса вектора переривання 000Ch). Резерв 64 байт дозволяє використовувати стек у підпрограмі обробки переривання. Антипереповнення стеку автоматично не визначається.

При скиданні обидва покажчики стеку (USP і SSP) ініціалізуються значенням 0100h. Такий варіант розміщення стеків доцільно використовувати тільки в тому випадку, коли один з стеків взагалі не використовується. Оскільки системний стек завжди задіяний при обробці переривань, то, як правило, після скидання слід ініціалізувати USP іншим значенням.

2.1.4 Порти вводу-виводу

Портів вводу-виводу є 7 (P0…P6). Порти P0…P5 8-розрядні, у порта P6 задіяні лише 2 лінії.
Для встановлення конфігурації виводів порту використовуються регістри PnCFGA і PnCFGB (тобто для порта P0 - регістри P0CFGA і P0CFGB, для P1 - P1CFGA і P1CFGB і т.д.). Кожному виводу порта відповідає один з бітів у обох регістрах. При скиданні всі виводи конфігуруються як квазідвонаправлені.

PnCFGA	PnCFGB	Тип виводу
0	0	Відкритий стік
0	1	Квазідвонаправлений
1	0	Відключений (Z - стан)
1	1	Двотактний

2.2 Організація пам’яті XA

Простір пам’яті XA має Гарвардську архітектуру, тобто пам’ять програм і пам’ять даних організовані в різних адресних просторах. Підтримується 16 Мбайт пам’яті програм та даних (24-бітна адресація).

XA може працювати з різними типами пам’яті: програмна пам’ять може бути програмованою, перепрограмованою, з одноразовим програмуванням, флеш, з програмуванням маскою; пам’ять даних - ОЗП, перепрограмованою або флеш.

2.2.1 Регістри

Архітектура XA включає 16 двобайтових регістрів. В основній моделі XA застосовуються тільки регістри R0-R7. Ці регістри доступні для використання без будь-яких обмежень за виключенням R7, який є покажчиком стеку. Таким чином, користувач має в розпорядженні 7 “акумуляторів”, над якими можна здійснювати будь-які операції.

Додаткові регістри R8-R15 зарезервовані і можуть використовуватися в окремих модифікаціях XA. Від R0-R7 вони відрізняються відсутністю побайтового доступу та неможливістю використання в якості покажчиків.

Регістри є незалежними від усіх інших просторів пам’яті.

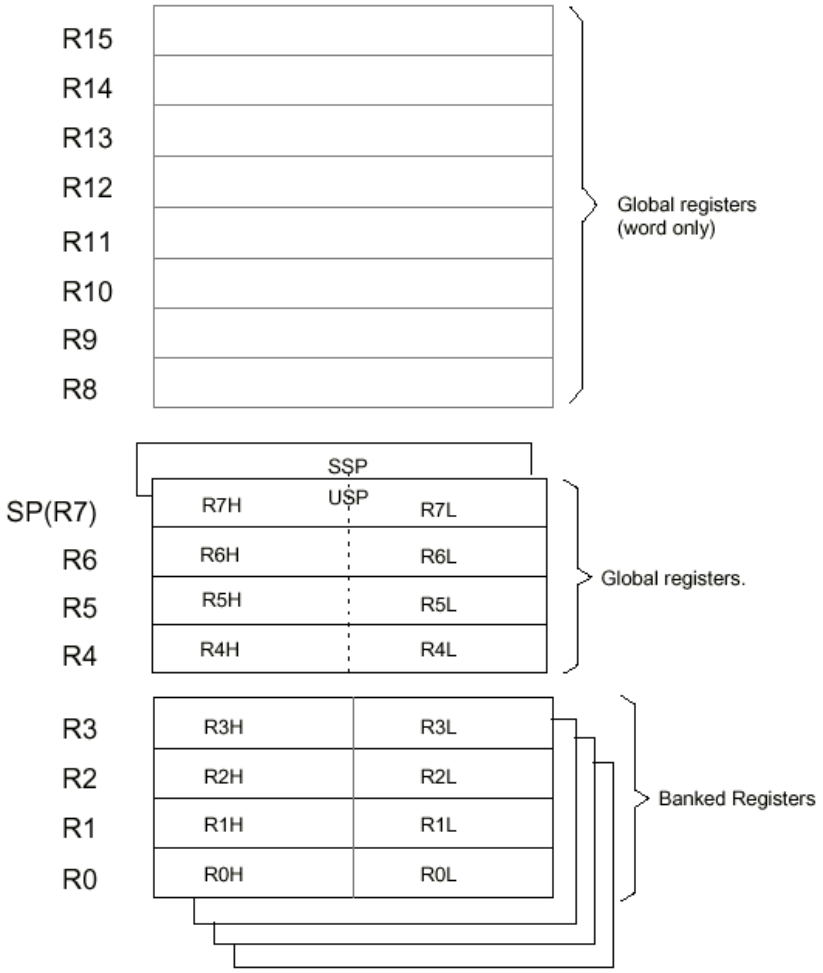


Рисунок 2.3 – Регістри XA

При звертанні до регістрів можлива побітова та побайтова адресація. Наприклад, R1 – це посилання на двобайтовий регістр, R1H – на старший байт, R1L – на молодший байт, R1.2 – на 2-й біт регістру R1. Бітова адресація регістрів показана на рис.2.4

R15	FF	FE	FD	FC	FB	FA	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0
R14	EF	EE	ED	EC	EB	EA	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
							•									
							•									
							•									
R7	7F	7E	7D	7C	7B	7A	79	78	77	76	75	74	73	72	71	70
R6	6F	6E	6D	6C	6B	6A	69	68	67	66	65	64	63	62	61	60
R5	5F	5E	5D	5C	5B	5A	59	58	57	56	55	54	53	52	51	50
R4	4F	4E	4D	4C	4B	4A	49	48	47	46	45	44	43	42	41	40
R3	3F	3E	3D	3C	3B	3A	39	38	37	36	35	34	33	32	31	30
R2	2F	2E	2D	2C	2B	2A	29	28	27	26	25	24	23	22	21	20
R1	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
R0	0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00

Рисунок 2.4 – Бітова адресація регістрів

32-розрядні регістрові пари утворюються з двох сусідніх регістрів і використовуються для 32-бітних зсувів, множення та ділення. Регістрова пара адресується через регістр з парним номером, який містить молодші байти подвійного слова. Тобто регістровими парами є: (R0,R1), (R2,R3), (R4,R5) та (R6,R7).
Регістри R0-R3 містяться в 4 банках. Активний банк вибирається бітами RS1 та RS0 регістру PSW:

RS1	RS0	банк
0	0	банк 0
0	1	банк 1
1	0	банк 2
1	1	банк 3

Ці біти доступні для запису як в системному режимі, так і в режимі користувача. Крім того, при перериваннях забезпечується автоматичне переключення банків регістрів.

2.2.2 Пам’ять даних

Пам’ять поділяється на пам’ять програм і пам’ять даних. Вони можуть бути внутрішніми або зовнішніми, залежно від модифікації контролера та конкретного застосування.
Пам’ять даних займає область адрес 0 – FFFFFFh і розділена на 256 сегментів по 64 Кбайт. Зовнішня пам’ять даних починається з першої адреси, наступної за останньою адресою у внутрішній пам’яті даних. Як мінімум 512 Кбайт внутрішнього ОЗП доступні у всіх моделях XA.
Останні 16 сегментів (адреси від F0:0000 до FF:FFFF) як у пам’яті програм, так і в пам’яті даних зарезервовані для спеціальних функцій в залежності від модифікації контролера.
При звертанні до слів адреса повинна бути парною. Якщо задана непарна адреса, буде зчитане слово, що починається з попередньої парної адреси.
Якщо деяка область адрес зовнішньої пам’яті перекривається внутрішньою, то при звертанні перевага надається внутрішній пам’яті, і зовнішня пам’ять доступна тільки при використанні команди MOVX. Для неперекриваючихся областей використання MOVX не обов’язкове.
Для утворення 24-бітної адреси використовуються сегментні регістри, де містяться старші 8 біт адреси. Сегментних регістрів два – DS (Data Segment Register) і ES (Extra Segment Register). Кожному з регістрів-покажчиків R0-R6 відповідає один з сегментних регістрів відповідно до бітів в регістрі SSEL (рис.2.5). 0 відповідає регістру DS (встановлюється при скиданні), 1 – ES. Внаслідок сегментної адресації простір пам’яті може розглядатись як 256 сегментів по 64 Кбайт. (рис. 2.6).

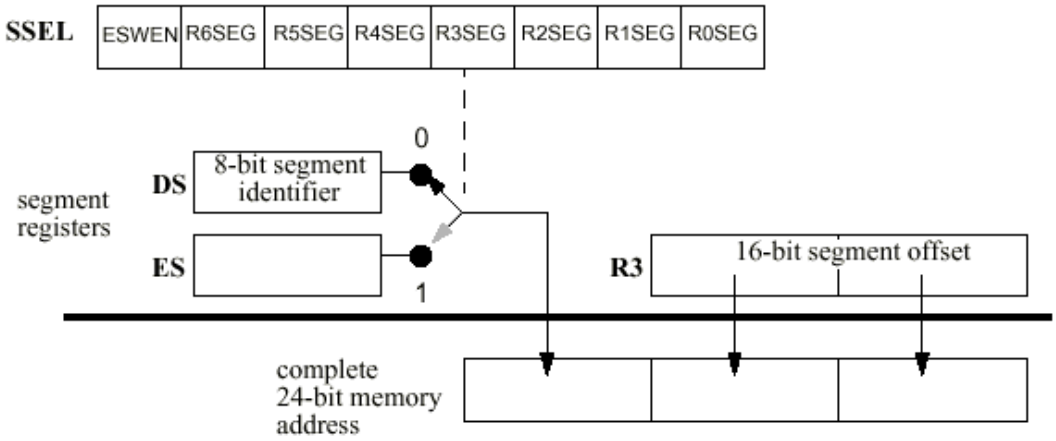


Рисунок 2.5 – Механізм утворення 24-розрядної адреси

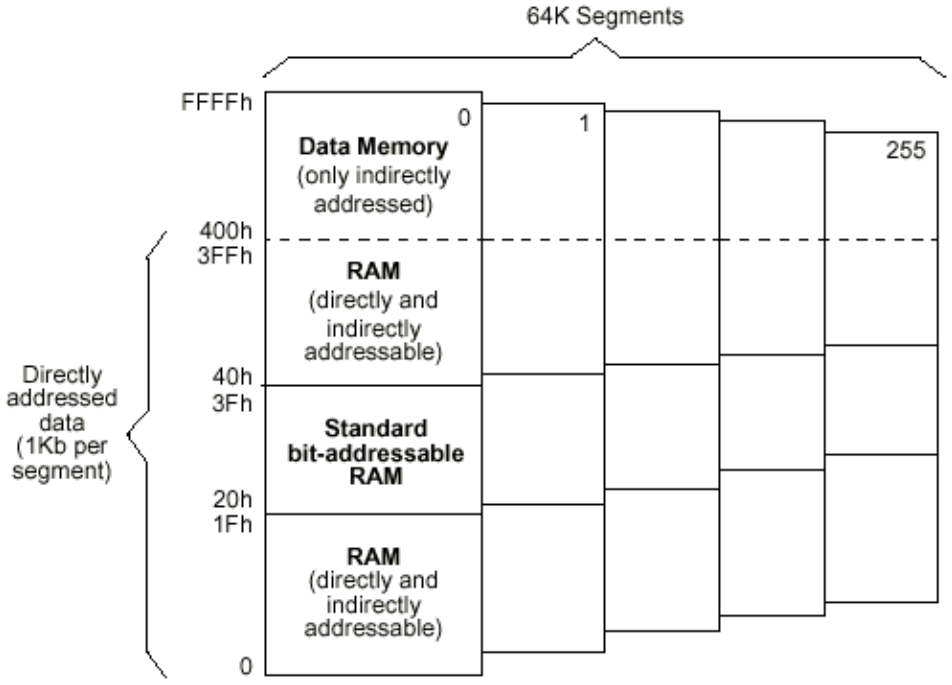


Рисунок 2.6 – Сегментація пам’яті

Біт ESWEN (7-й біт регістру SSEL) може бути змінений тільки в системному режимі. Він дозволяє (1) або забороняє (0) права запису до сегменту, адресованого через ES, у режимі користувача. При скиданні він встановлюється в 0.

2.2.3 Способи адресації

Непрямий. 24-бітна адреса утворюється конкатенацією вмісту 8-бітного сегментного регістру та 16-бітного покажчика в одному з регістрів.
Непрямий зі зміщенням (рис.2.7а). Одно- або двобайтове знакове зміщення додається до вмісту вказаного регістра і результат конкатенується з вмістом сегментного регістру. Цей спосіб забезпечує доступ до структур даних, коли регістр-покажчик містить початкову адресу структури, а також передачу параметрів через стек.

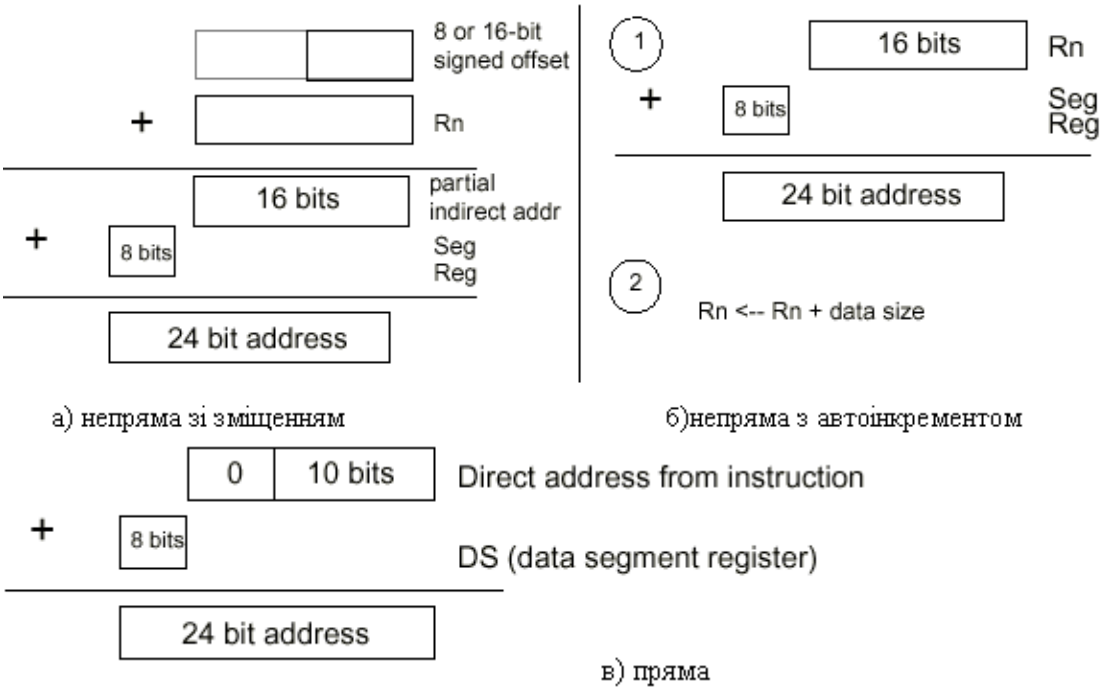


Рисунок 2.7 – Утворення 24-розрядної адреси при різних способах адресації

Непрямий з автоінкрементом (рис.2.7б). Адреса формується як при непрямій адресації, і регістр покажчика автоматично інкрементується. Якщо операнд однобайтовий, додається 1; якщо двобайтовий – 2. Цей метод зручний для обробки масивів даних , менших за 64 Кбайт. Для більших масивів необхідно змінювати вміст сегментного регістру при досягненні кінця сегмента.
Прямий (рис.2.7в). Перший кілобайт в кожному сегменті може адресуватися безпосередньо в команді. При цьому завжди використовується сегментний регістр DS. 10-бітна адреса, вказана в команді, доповнюється нулями до 16 біт і конкатенується з вмістом DS.

2.2.4 Регістри спеціальних функцій

1 Кбайт в області адрес 400h – 7FFh зарезервований для регістрів спеціальних функцій (РСФ). Ця область використовує адреси прямоадресованої пам’яті, але являє собою логічну область, що не пов’язана з схемою сегментації пам’яті.
Область 600h – 7FFh призначена для зовнішніх регістрів. Це забезпечує швидкий доступ до зовнішніх пристроїв без необхідності створення покажчика при кожному звертанні.
Регістри за адресами 400h – 43Fh є біт-адресованими.
У програмі РСФ повинні адресуватись тільки символьно, оскільки адреси регістрів можуть відрізнятися у різних модифікаціях.
Доступ до РСФ не залежить від значень сегментних регістрів, оскільки при виконанні команди використовуються тільки молодші 10 біт адреси.
Для РСФ не допускається непряма адресація. При використанні непрямої адресації відбувається звертання до пам’яті даних.
Порядок доступу до зовнішніх РСФ залежить від конкретної модифікації контролера.

2.2.5 Бітова адресація

Крім регістрів R0 – R15 та РСФ за адресами 400h – 43Fh біт-адресованими є також 32 байти в кожному сегменті пам’яті даних (DS) починаючи з адреси 20h. Утворення адреси показано на рис. 2.8, область біт-адресованої пам’яті – на рис.2.9.

Рисунок 2.8 – Бітова адресація

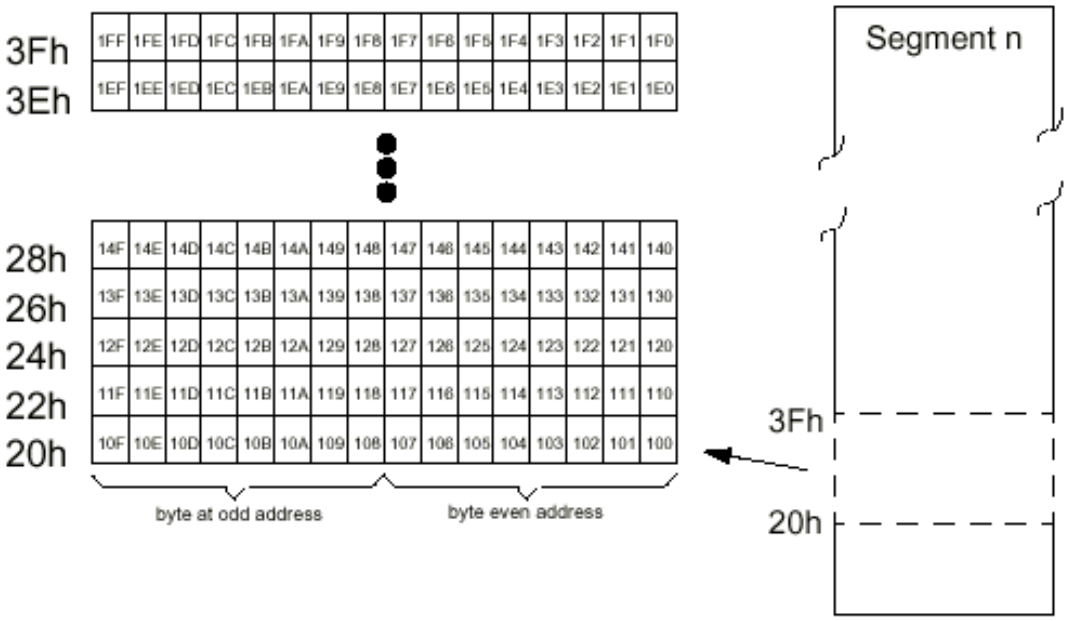


Рисунок 2.9 – Біт-адресована область пам’яті

2.2.6 Пам’ять програм

Пам’ять програм займає область адрес 0 – FFFFFFh і розділена на 256 сегментів по 64Кбайт. Зовнішня пам’ять програм починається з першої адреси, наступної за останньою адресою у внутрішній пам’яті програм.

Оскільки довжина команд змінюється від 1 до 6 байт, команди можуть розміщуватися як за парними, так і за непарними адресами, за виключенням команд переходів, які повинні розміщуватись за парними адресами. При перекриванні областей адрес внутрішньої та зовнішньої пам’яті відбувається звертання до внутрішньої пам’яті.

Архітектура XA підтримує зберігання констант у пам’яті програм і забезпечує доступ до них командою MOVC. Стандартний формат MOVC використовує один з 16-бітних регістрів як покажчик, додаючи його вміст до старших 8 біт лічильника команд PC (якщо відповідний біт SSEL = 0) або сегментного регістру CS (якщо відповідний біт SSEL = 1).

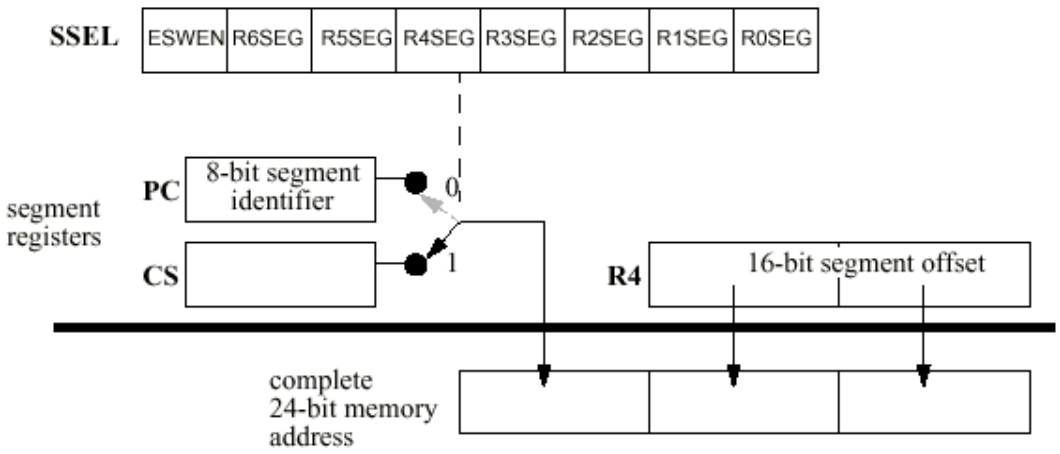


Рисунок 2.10 – Адресація у пам’яті програм

2.3 Переривання

Переривання XA поділяються на:
- виключення (exception interrupts)
- події (event interrupts)
- програмні переривання(software interrupts)
- (trap interrupts)

У пам'яті програм за адресами 000h – 11Bh розміщується таблиця векторів переривань. Таблиця може містити до 71 вектора. Кожен вектор займає 4 байти і включає 16-розрядну адресу підпрограми обробки переривання та нове значення регістру ознак PSW. Оскільки адреса 16-розрядна, перша команда підпрограми обробки повинна розташовуватись в нульовому сегменті. Ця адреса повинна бути парною. Перших 16 векторів зарезервовані для виключень (з них використовуються тільки 7), наступні 16 – для переривань типу Trap, наступні 32 – для переривань типу подій та останні 7 – для програмних переривань.

При обробці будь-якого переривання у системний стек заноситься адреса наступної команди та значення регістру ознак PSW. З вектора переривань у пам'яті програм зчитується нове значення PSW та адреса підпрограми обробки переривання, і відбувається перехід на цю адресу. Нове значення PSW може мати інші біти SM, RP1:RP0 та IM3:IM0, тобто обробка переривання може відбуватися в іншому режимі та з іншим рівнем пріоритету, ніж у поточної задачі, а також з використанням іншого банку регістрів. Для повернення з підпрограми служить команда RETI, по якій зі стеку читаються попередньо занесені значення адреси та PSW і продовжується виконання перерваної задачі. Як правило, обробка переривань здійснюється в системному режимі, оскільки виконання команди RETI в режимі користувача спричиняє переривання типу виключення.

Виключення виникають при подіях значної важливості і обробляються відразу, незалежно від рівня пріоритету виконуваної задачі. Якщо одночасно виникають кілька виключень, вони обробляються згідно з пріоритетом:

Виключення	Адреса вектора переривання	Пріоритет
Точка переривання	0004h	0
Покроковий режим	0008h	1
Переповнення стеку	000Ch	2
Ділення на нуль	0010h	3
RETI в режимі користувача	0014h	4
(зарезервовано)	0018h	5
NMI	009Ch	6
Скидання	0000h	7

Джерелом переривань типу **подій** є, як правило, периферійні пристрої (таймер, UART та ін.) або сигнал на вході зовнішнього переривання. Ці переривання обробляються тільки в тому випадку, коли вони дозволені і їх рівень пріоритету вищий, ніж у поточної задачі.

Для дозволу і заборони переривань служить регістр IE (interrupt enable). Якщо біт EA в цьому регістрі дорівнює 0, то всі переривання заборонені. Якщо EA = 1, то дозвіл/заборона переривання визначається значенням відповідного біту в регістрі IE (1 – дозвіл, 0 – заборона).

Пріоритет переривання визначається регістром пріоритету. Якщо одночасно виникають кілька переривань з однаковим пріоритетом, вони обробляються згідно наступного порядку (для модифікації XA-S3):

Переривання	Флаг перери-вання	Адреса вектора переривання	Біт дозволу	Біти вибору пріоритету	Порядок обробки
Зовнішнє переривання 0	IE0	0080–0083	EX0	IPA0.2–0(PX0)	2
Переповнення таймера 0	TF0	0084–0087	ET0	IPA0.6–4(PT0)	3
Зовнішнє переривання 1	IE1	0088–008B	EX1	IPA1.2–0(PX1)	4
Переповнення таймера 1	TF1	008C–008F	ET1	IPA1.6–4(PT1)	5
Переповнення таймера 2	TF2(EXF2)	0090–0093	ET2	IPA2.2–0(PT2)	6
Переповнення таймера PCA	CCF0–CCF4,CF	0094–0097	EPC	IPA2.6–4(PPC)	7
Переривання АЦП	ADINT	0098–009B	EAD	IPA3.2–0(PAD)	8
Переривання приймача UART0	RI_0	00A0–00A3	ERI0	IPA4.2–0(PRI0)	9
Переривання передавача UART0	TI_0	00A4–00A7	ETI0	IPA4.6–4(PTI0)	10
Переривання приймача UART1	RI_1	00A8–00AB	ERI1	IPA5.2–0(PRI1)	11
Переривання передавача UART1	TI_1	00AC–00AF	ETI1	IPA5.6–4(PTI1)	12
PCA0	CCF0	00C0–00C3	EC0	IPB0.2–0(PC0)	17
PCA1	CCF1	00C4–00C7	EC1	IPB0.6–4(PC1)	18
PCA2	CCF2	00C8–00CB	EC2	IPB1.2–0(PC2)	19
PCA3	CCF3	00CC–00CF	EC3	IPB1.6–4(PC3)	20
PCA4	CCF4	00D0–00D3	EC4	IPB2.2–0(PC4)	21
Переривання I^2C	SI	00D4–00D7	EI2	IPB2.6–4(PI2)	22

Програмні переривання викликаються програмним встановленням біту запиту в регістрі SWR (адреса 42Ah). У цьому регістрі 7 бітів запиту, що відповідають 7 перериванням. Переривання обробляється тільки в тому випадку, коли його рівень пріоритету вищий, ніж у поточної задачі та якщо встановлений біт дозволу у регістрі SWE (адреса 47Ah).

Програмні переривання можуть використовуватись для виконання фрагментів підпрограм обробки інших переривань з нижчим рівнем пріоритету, ніж на початку підпрограми. Наприклад, якщо у деякій підпрограмі обробки лише частина коду повинна бути виконана негайно, а виконання решти коду можна завершити пізніше, то цей код доцільно викликати за допомогою програмного переривання з нижчим рівнем пріоритету, щоб дозволити на протязі його виконання обробку інших переривань (з пріоритетом вищим, ніж у відповідного програмного переривання, але нижчим, ніж у критичної ділянки коду, що була виконана перед цим).

Програмне переривання	**Адреса вектора переривання**	**Пріоритет**
SWI1	0100h	1
SWI2	0104h	2
SWI3	0108h	3
SWI4	010Ch	4
SWI5	0110h	5
SWI6	0114h	6
SWI7	0118h	7

Переривання типу **trap** виникають при виконанні команди TRAP. Визначено 16 таких переривань. Вони можуть застосовуватись як механізм виклику системних функцій та для переключення між системним режимом і режимом користувача.

2.4 Таймери XA

XA має 2 16-розрядні таймери-лічильники 0 та 1 та таймер 2 - 16-розрядний реверсивний таймер-лічильник.
Таймери-лічильники можуть виконувати наступні функції:
- вимірювати інтервал часу або тривалість імпульсу;
- рахувати зовнішні події;
- генерувати запит на переривання;
- генерувати широтно-модульовані імпульсні послідовності.

Всі таймери можуть працювати у режимі таймера або лічильника, що визначається бітом C/T у регістрі TnCON.Частота тактування таймерів T0, T1, T2 в режимі таймера визначається бітами PT1, PT0 в регістрі SCR і може складати від Fosc/4 до Fosc/64, де Fosc – частота осцилятора.
В режимі лічильника інкрементування таймера відбувається по спаду імпульсу (від 1 до 0) на зовнішньому вході. Входи читаються кожні 2 цикли, тому для ідентифікації переходу з 1 в 0 необхідно як мінімум 4 цикли, тобто частота підрахунку не може бути вищою за Fosc/4.
При переповненні таймера/лічильника встановлюється флаг TF в регістрі TnCON.
Якщо в режимі таймера встановлений біт TnOE, то на виводі, який в режимі лічильника використовується для підрахунку подій, генерується широтно-модульована імпульсна послідовність.

2.4.1 Таймери 0 та 1

Режим роботи таймерів визначається регістром TMOD, який є повністю аналогічним відповідному регістру мікроконтролера 80C51. Режими, що встановлюються бітами M1:M0, відповідають режимам 80C51, за виключенням *режиму 0*, який для XA є режимом 16-розрядного таймера/лічильника з автозавантаженням. У цьому режимі використовуються 2 додаткові регістри RTH:RTL, значення яких завантажується у регістри таймера TH:TL при переповненні таймера.
Частота переповнення у режимі 0:

$$Timer\_rate = Fosc/ (N*(65536 – Reload\_Value))$$

де N – значення коефіцієнта попереднього масштабування, Reload_Value = RTH:RTL.

Режим 1 – 16-розрядний таймер\лічильник без автозавантаження.

Режим 2 – 8-розрядний таймер\лічильник з автозавантаженням. Для автозавантаження замість регістра TH (як у 80C51) використовується регістр RTL.
Частота переповнення у режимі 2:

$$Timer\_rate = Fosc/ (N*(256 – RTL))$$

де N – значення коефіцієнта попереднього масштабування.

Режим 3 – два 8-розрядні таймери (тільки для таймера 0). Таймер 2 у цьому режимі зупиняється, як при встановленні TR1 = 0. Біти TR1 і TF1 використовуються 8-розрядним таймером TH0. Тому, коли таймер 0 знаходиться в режимі 3, таймер 1 може включатися і виключатися шляхом включення і виключення режиму 3, або використовуватися для задання швидкості передачі послідовного порту, або для виконання будь-якої задачі, що не потребує переривання.Управляючий регістр TCON також є повністю аналогічним відповідному регістру мікроконтролера 80C51.
Входи, які використовуються у режимі лічильника для підрахунку подій (T0(P3.4) для таймера 0 та T1(P3.5) для таймера 1), можуть бути використані в XA як виходи для генерації широтно-імпульсних сигналів. Якщо встановлений біт TnOE в регістрі TSTAT, то на цих виходах генерується ШІМ-сигнал з тривалістю одиничного рівня, що визначається регістром автозавантаження, та частотою, що визначається значенням поділювача та регістра автозавантаження і може складати від Fosc/8 до Fosc/8 388 608 (для частоти осцилятора 30МГц це становить 3.58 Гц … 3.75 МГц).

Регістр TSTAT (адреса 411h)

-	-	-	-	-	T1OE	-	T0OE

T0OE – біт дозволу генерування широтно-імпульсних сигналів на виводі T0
T1OE – біт дозволу генерування широтно-імпульсних сигналів на виводі T1

Регістр TMOD (адреса 45Ch)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
для таймера 1				для таймера 0			

GATE – біт дозволу таймера по входу INT: при GATE = 1 для дозволу таймера необхідний не тільки TR = 1, але і одиничний рівень на вході INT; при GATE = 0 – тільки TR = 1.
C/T – біт задання режиму таймера (0) або лічильника (1)
M1:M0 – біти задання режиму роботи

Регістр TCON (адреса 410h)

IT0	IE0	IT1	IE1	TR0	TF0	TR1	TF1

TFn – флаг переривання таймера n
TRn – біт дозволу роботи таймера n
IEn – флаг зовнішнього переривання по входу INTn
ITn – біт вибору переривання INTn (0 – по фронту спаду, 1 – по нульовому рівню)

2.4.2 Таймер 2

Таймер 2 може працювати в одному з трьох режимів: з автозавантаженням (з прямим або зворотнім рахунком), в режимі захоплення та для задання швидкості передачі UART. Встановлення бітів для вибору режиму показано в таблиці.

TR2	CP/RL2	RCLK+TCLK	DCEN	Режим
0	x	x	x	таймер відключений
1	0	0	0	таймер\лічильник з автозавантаженням, прямий рахунок
1	0	0	1	таймер\лічильник з автозавантаженням, прямий або зворотній рахунок в залежності від стану входу T2EX
1	1	0	x	таймер\лічильник в режимі захоплення
1	x	1	x	Задання частоти передачі

Переривання таймера 2 генерується або при переповненні, або при антипереповненні в режимі зворотнього рахунку, або по фронту спаду на вході T2EX, якщо в регістрі T2CON біт EXEN2 = 1. Всім цим подіям відповідає єдиний вектор переривання за адресою 0090h. Переривання може бути масковане бітом ET2 в регістрі IE.

Регістр T2CON (адреса 418h)

TF2	EXF2	RCLK0	TCLK0	EXEN2	TR2	C/T2	CP/RL2

TF2 – флаг переповнення. Повинен скидатись програмно. Не встановлюється в режимах задання швидкості передачі та генерування імпульсної послідовності.
EXF2 – флаг, що встановлюється при переході від 1 до 0 на виводі T2EX при EXEN2 = 1. Повинен скидатись програмно.
RCLK0 – флаг тактування прийому UART0
TCLK0 – флаг тактування передачі UART0.

EXEN2 – біт дозволу захоплення/автозавантаження по фронту спаду на виводі T2EX.
TR2 – біт дозволу таймера 2 (0 – виключений. 1 – включений)
C/T2 – вибір режиму: 0 – таймер, 1 - лічильник
CP/RL2 – флаг захоплення(1) \автозавантаження(0) (захоплення/автозавантаження відбувається тільки при EXEN2 = 1 та RCLK=0 і TCLK=0)

Регістр T2MOD (адреса 419h)

-	-	RCLK1	TCLK1	-	-	T2OE	DCEN

RCLK1 – флаг тактування прийому UART1
TCLK1 – флаг тактування передачі UART1
T2OE – флаг дозволу генерування імпульсної послідовності на виводі T2
DCEN – флаг напряму рахунку в режимі автозавантаження (0 – прямий, 1 – в залежності від входу T2EX)

Режим захоплення встановлений, якщо у регістрі T2CON біт CP/RL2 = 1. Якщо в регістрі T2CON біт EXEN2 = 0, таймер працює як звичайний 16-розрядний таймер/лічильник. Якщо EXEN2 = 1, то при появі фронту спаду на вході T2EX поточне значення регістрів TH2:TL2 переписується в регістри RCAP2H:RCAP2L і встановлюється флаг переривання EXF2 в регістрі T2CON.

В *режимі з автозавантаженням* при переповненні таймера у регістри TH2:TL2 записується значення з регістрів T2CAPH:T2CAPL. Якщо при цьому встановлений біт EXEN2 в регістрі T2CON, то перезавантаження відбувається також при появі фронту спаду на вході T2EX (при цьому встановлюється біт EXF2 в регістрі T2CON). У цьому режимі можна встановити напрям рахунку за допомогою біту DCEN (down counter enable) в регістрі T2MOD. Якщо DCEN = 0, рахунок прямий (таймер рахує від менших до більших значень). Якщо DCEN = 0, напрям рахунку залежить від стану входу T2EX: якщо на цьому вході одиничний рівень, рахунок прямий, якщо нульовий – зворотній. При зворотньому рахунку значення TH2:TL2 порівнюється зі значенням T2CAPH:T2CAPL і при їх рівності (антипереповненні) таймер завантажується значенням FFFFh. При цьому, як і при переповненні, встановлюється флаг переривання TF2. Флаг EXF2 змінює значення при кожному переповненні/антипереповненні таймера і може бути при необхідності використаний як його 17-й біт.

Режим задання частоти передачі вибраний, якщо встановлені біти TCLKn і (або) RCLKn у регістрі T2CON або T2MOD. Швидкості прийому і передавання можуть бути різними.

Генерування імпульсної послідовності з щілинністю 2 на виводі P1.6 відбувається, якщо встановлений біт T2OE в регістрі T2MOD. Біт C/T2 у регістрі TCON при цьому повинен бути нульовим. Частота послідовності складатиме TCLK/ (2*(65536 – TCAP2H:TCAP2L)). Переповнення таймера у цьому випадку не спричиняє переривання. Можна використовувати таймер 2 одночасно для задання швидкості передачі та для генерації імпульсної послідовності (швидкість передачі при цьому буде у 8 разів менша, ніж частота послідовності).

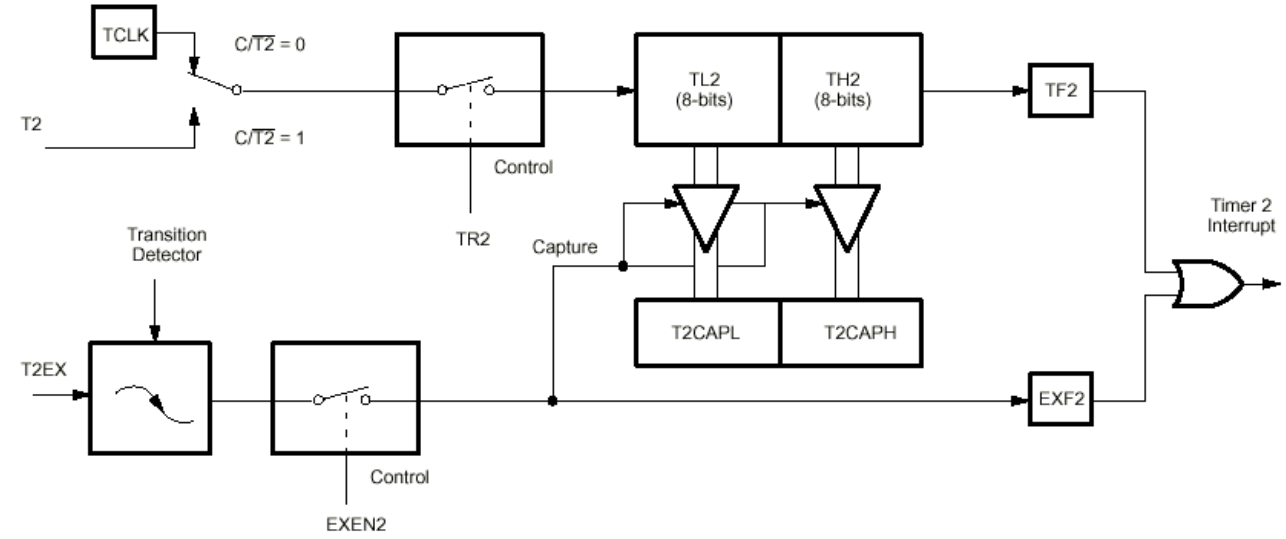


Рисунок 2.11 - Таймер 2 в режимі захоплення

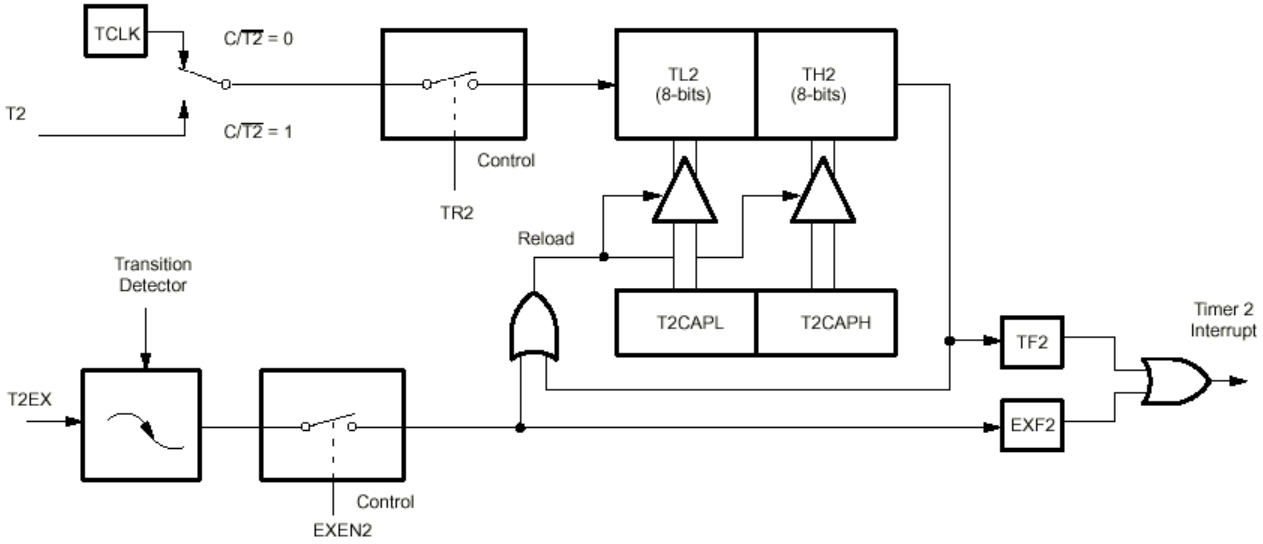


Рисунок 2.12 -Таймер 2 в режимі автозавантаження при DCEN = 0

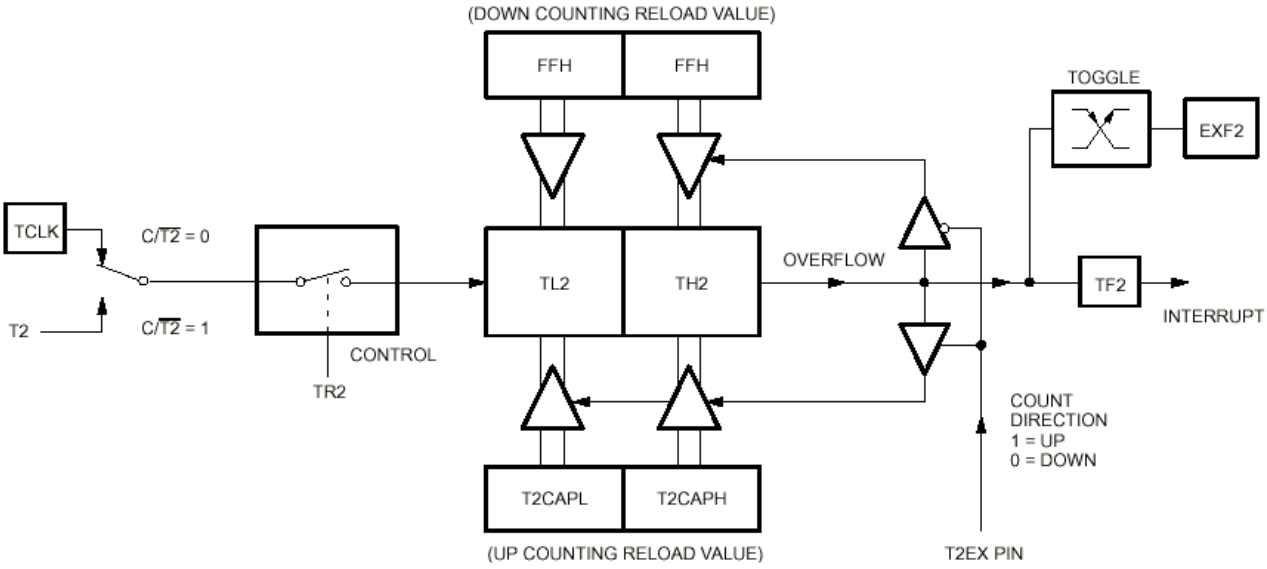


Рисунок 2.13 - Таймер 2 в режимі автозавантаження при DCEN = 1

2.4.3 Масив програмованих лічильників (PCA – Programmable counter array) XA-S3

Масив програмованих лічильників – це 16-розрядний таймер-лічильник з 5 модулями захоплення/порівняння.

Для запуску PCA необхідно встановити біт CR у регістрі CCON.

Джерело тактування таймера задається бітами CPS1 та CPS0 у регістрі CMOD.

Кожен з п'яти модулів може працювати в чотирьох режимах: захоплення по фронту наростання/спаду, програмний таймер, високошвидкісний вивід або широтно-імпульсний модулятор. Кожен модуль пов'язаний із одним з виводів порту 4 (модуль 0 – P4.1 (CEX0), 1 – P4.2 (CEX1) і т.д.) та з регістрами CCAPnH:CCAPnL. Для встановлення режиму роботи модулів використовуються регістри CCAPMn.

Переривання PCA можуть генеруватися по переповненню таймера-лічильника (якщо встановлений біт ECF у регістрі CMOD) або по виконанню умови захоплення/порівняння для будь-якого з модулів (якщо встановлений біт ECCFn у регістрі CCAPMn)

Модуль 4 може бути відконфігурований для роботи в режимі таймера-сторожа (WDT) встановленням біту WDTE у регістрі CMOD. У цьому режимі, як і в режимі порівняння, відбувається порівняння регістрів CH:CL з CCAPnH:CCAPnL, і у випадку співпадіння відбувається скидання (вивід RST при цьому не встановлюється в 0). Щоб уникнути скидання, можливі 3 шляхи:

- періодично змінювати значення CCAPnH:CCAPnL, щоб вони ніколи не були рівні CH:CL;
- періодично змінювати значення CH:CL, , щоб вони ніколи не були рівні CCAPnH:CCAPnL;
- заборонити WDT скиданням біту WDTE перед співпадінням, а потім дозволити його знову.

Регістри управління PCA:

1. Регістр CMOD (адреса 490h)

CIDL	WDTE	-	-	-	CPS1	CPS0	ECF

CIDL – біт дозволу(0) чи заборони (1) роботи PCA у режимі зменшеного енергоспоживання
WDTE – біт дозволу таймера-сторожа (WDT) у модулі 4
CPS1, CPS0 – біти вибору джерела тактування:

CPS1	CPS0	Джерело тактування
0	x	TCLK (Osc/4, Osc/16, Osc/64)
1	0	Переповнення таймера 0
1	1	вхід зовнішнього тактування ECI

ECF – біт дозволу (1) переривання по переповненню.

2. Регістр CCON (адреса 41Ah)

CF	CR	-	CCF4	CCF3	CCF2	CCF1	CCF0

CF – флаг переривання по переповненню таймера. Повинен скидатися програмно.
CR – біт дозволу роботи PCA
CCF0 – CCF4 – флаги переривання модулів захоплення-порівняння 0 – 4. Повинні скидатися програмно.

3. Регістри CCAPMn (адреси 491h - 495h)

-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

ECOMn – біт дозволу компаратора
CAPPn – біт дозволу захоплення по фронту наростання
CAPNn – біт дозволу захоплення по фронту спаду
MATn – дозвіл встановлення флагу переривання CCFn по співпадінню таймера-лічильника з регістром модуля порівняння
TOGn – дозвіл зміни виходу CEXn по співпадінню таймера-лічильника з регістром модуля порівняння
PWMn – біт режиму широтно-імпульсної модуляції
ECCFn – біт дозволу переривання модуля захоплення/порівняння

Режими роботи модуля CCAPn в залежності від бітів регістру CCAPMn

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	Режим
0	0	0	0	0	0	0	Модуль не працює
x	1	0	0	0	0	x	Захоплення по фронту наростання на CEXn
x	0	1	0	0	0	x	Захоплення по фронту спадання на CEXn
x	1	1	0	0	0	x	Захоплення по будь-якому фронту на CEXn
1	0	0	1	0	0	x	Програмний таймер
1	0	0	1	1	0	x	Високошвидкісний вихід
1	0	0	0	0	1	0	8-розрядна ШІМ
1	0	0	1	x	0	x	Таймер-сторож

У режимі *захоплення* один з бітів CAPN, CAPP (або обидва) повинен бути встановлений в 1. При CAPP = 1 захоплення відбувається по фронту наростання, при CAPN = 1 – по фронту спаду, при CAPP = 1 і CAPN = 1 – по будь-якому фронту. При появі відповідного фронту на вході CEXn значення регістрів таймера-лічильника CH:CL апаратно переписується у регістри CCAPnH:CCAPnL. При цьому встановлюється флаг CCFn і генерується переривання, якщо воно дозволене бітом ECCFn.

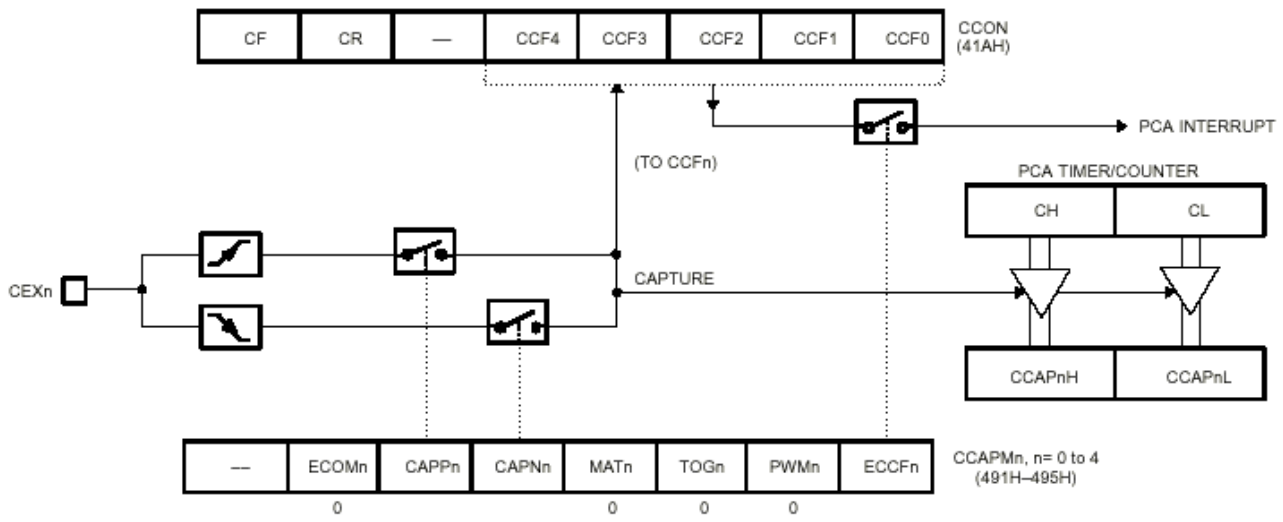


Рисунок 2.14 - Режим захоплення

Режим *порівняння* задається встановленням бітів ECOMn та MATn в регістрі CCAPMn. Значення регістрів таймера-лічильника CH:CL порівнюється з CCAPnH:CCAPnL і генерується переривання, якщо воно дозволене бітом ECCFn.

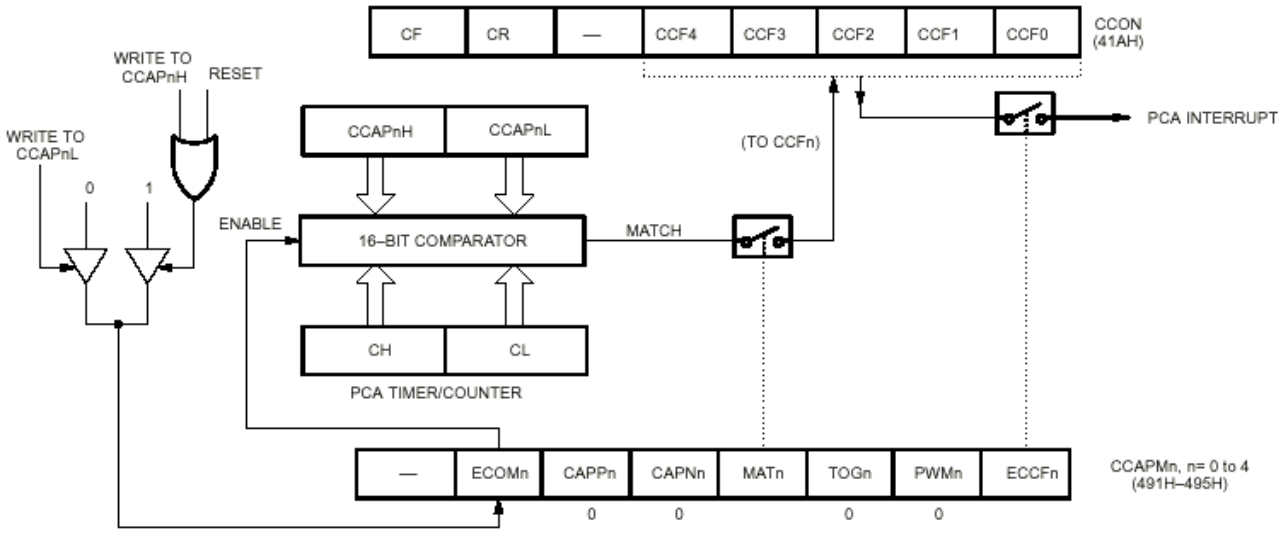


Рисунок 2.15 - Режим порівняння

У режимі *високошвидкісного виходу* при співпадінні регістрів таймера-лічильника CH:CL з CCAPnH:CCAPnL відбувається зміна стану виводу CEXn. Для роботи в цьому режимі біти TOGn, MATn і ECOMn повинні бути рівні 1.

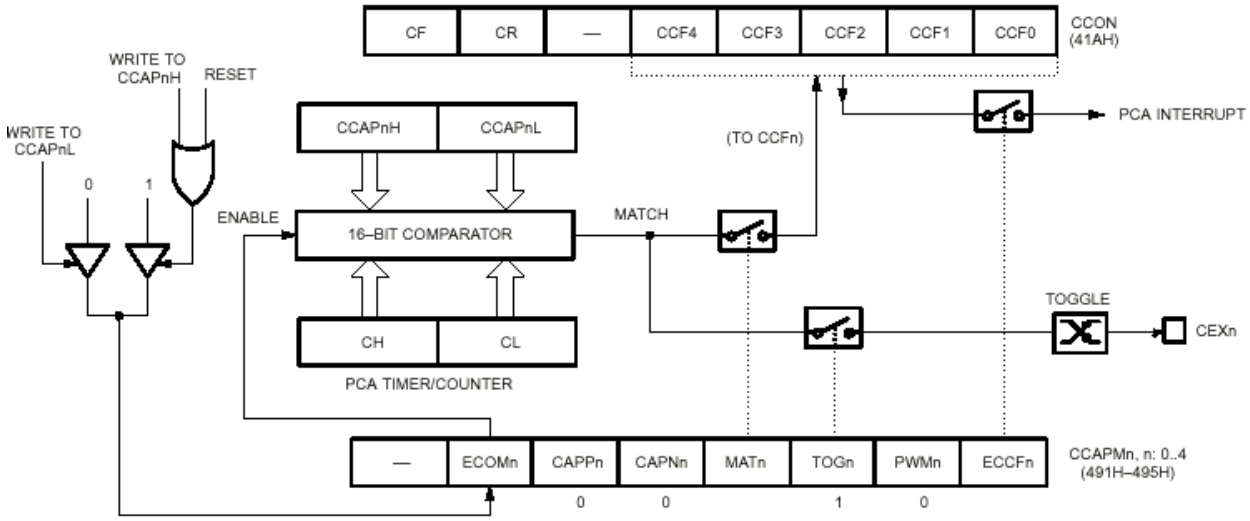


Рисунок 2.16 - Режим високошвидкісного виходу

Для задання режиму *широтно-імпульсної модуляції* повинні бути встановлені біти PWMn і ECOMn. Частота ШІМ-сигналу визначається джерелом тактування, а тривалість одиничного рівня – регістром CCAPLn. Якщо в режимі ШІМ одночасно працюють кілька модулів, частота ШІМ для усіх модулів буде однаковою, оскільки вони спільно використовують один таймер-лічильник. Формування ШІМ-сигналу відбувається наступним чином: поки значення в регістрі таймера CL менше, ніж CCAPLn, на виводі CEXn нульовий рівень, коли більше або рівне – одиничний. При переповненні CL регістр CCAPLn перезавантажується значенням з CCAPHn.

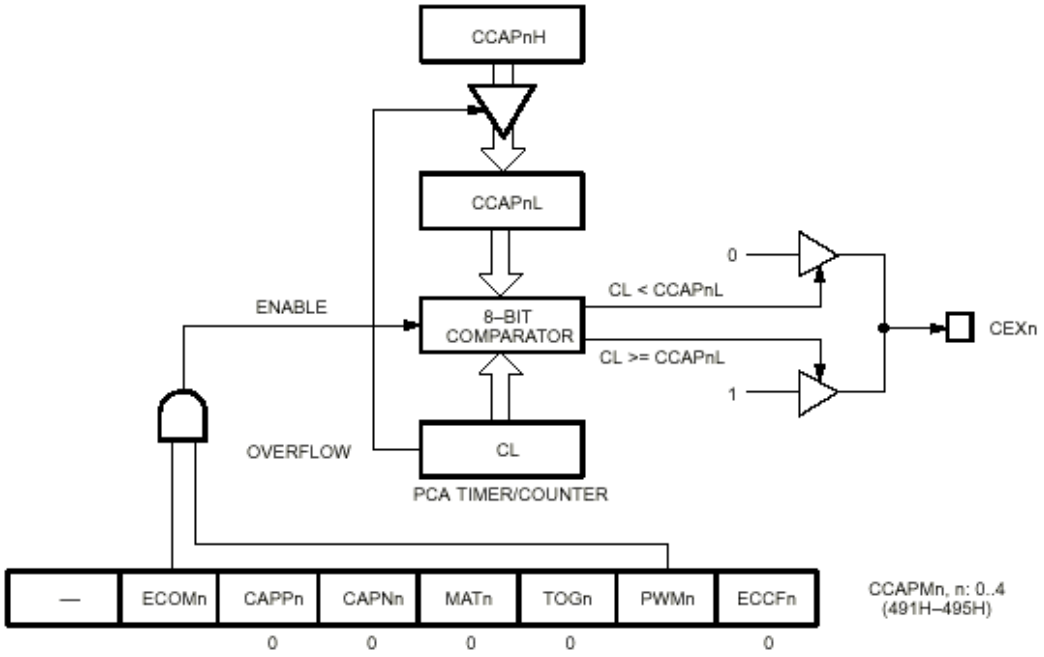


Рисунок 2.17 - Режим широтно-імпульсної модуляції

2.4.4 Таймер-сторож (WDT – watchdog timer)

Таймер WDT призначений для захисту системи від виконання некоректного коду шляхом скидання основного процесора при переповненні WDT, яке відбувається, якщо програма не перезавантажила таймер до виникнення переповнення. Таймер-сторож запускається після будь-якого скидання і повинен бути відключений програмно, якщо у його використанні немає необхідності.

WDT складається з 13-розрядного масштабувача та 8-розрядного таймера. Джерело тактування для WDT те саме, що і для таймерів 0,1,2, тобто може складати Osc/4 … Osc/64. Таймер декрементується через кожні N × P тактів, де N – коефіцієнт ділення подільника осцилятора, P – коефіцієнт поділювача WDT, що задається бітами PRE2:PRE0 у регістрі WDCON.

Регістр WDCON (адреса 41Fh)

PRE2	PRE1	PRE0	-	-	WDRUN	WDTOF	-

PRE2:PRE0 – біти для задання коефіцієнта ділення подільника частоти:

PRE2	PRE1	PRE0	коефіцієнт подільника
0	0	0	32
0	0	1	64
0	1	0	128
0	1	1	256
1	0	0	512
1	0	1	1024
1	1	0	2048
1	1	1	4096

WDRUN – біт запуску WDT

WDTOF – флаг, який встановлюється при антипереповненні WDT

Відразу після зовнішнього скидання WDT починає працювати (біт WDRUN = 1); поділювач та регістр автозавантаження WDL (адреса 45Fh) завантажуються нульовими

значеннями; скидається флаг WDTOF; коефіцієнт поділювача WDT встановлюється у максимальне значення.

При антипереповненні таймер перезавантажується 8-розрядним значенням з регістру WDL, встановлюється флаг WDTOF та відбувається скидання контролера. Якщо у послідовність команд, які виконуються відразу після скидання, включити перевірку флага WDTOF, можна визначити, з якої причини відбулося скидання.

Час спрацювання WDT: $t_D = t_{OSC} \times N \times P \times (W + 1)$,

де t_{OSC} – тривалість одного такту осцилятора, N – коефіцієнт ділення масштабувача осцилятора, P – коефіцієнт ділення масштабувача WDT, W – значення регістру автозавантаження WDL

Мінімальний час спрацювання $t_{MIN} = t_{OSC} \times 4 \times 32$ (W = 0, N = 4)

Максимальний час спрацювання $t_{MAX} = t_{OSC} \times 64 \times 4096 \times 256$ (W = 255, N = 64)

Щоб уникнути спрацювання WDT, необхідно завантажити регістри WFEED1, WFEED2 значеннями A5h та 5Ah відповідно, тобто виконати наступну послідовність команд:

```
clr ea                  ; заборона переривань
mov.b wfeed1,#A5h       ; “feed watchdog”
mov.b wfeed2,#5Ah       ; (дослівно “погодувати сторожового пса”)
setb ea                 ; відновлення дозволу переривань
```

Програмне забезпечення повинне бути написане таким чином, щоб ця операція відбувалась через кожні t_D секунд. Не рекомендується включати її в цикли чи підпрограми.

Цю ж послідовність команд слід виконати після присвоєння нового значення регістру WDCON.

Для відключення WDT відразу після скидання необхідно включити в ініціалізуючий код наступну послідовність команд:

```
mov.b wdcon,#0          ; відключення WDT
mov.b wfeed1,#A5h
mov.b wfeed2,#5Ah
```

2.5 Послідовний інтерфейс UART

Мікроконтролер XA містить 2 блоки послідовного інтерфейсу UART0 та UART1.

Регістри управління SCONn аналогічні регістру SCON мікроконтролера 80C51. У регістрі SnSTAT знаходяться флаги, що встановлюються при виникненні помилкових ситуацій (не прийнятий стоп-біт, прийнятий нульовий байт, прийнятий байт до зчитування з буфера попереднього байта).

Режими роботи блоку послідовного інтерфейсу аналогічні режимам UART мікроконтролера 80C51, за виключенням швидкостей передачі.

Швидкість передачі у режимах 0 та 2 визначається осцилятором, 1 та 3 – частотою переповнення таймера 1 або 2. По замовчуванню обидва блоки тактуються таймером 1. Таймер 2 можна вибрати для задання швидкості передачі UART0 шляхом встановлення бітів R0CLK,T0CLK у регістрі T2CON або UART1 шляхом встановлення бітів R1CLK,T1CLK у регістрі T2MOD.

Регістр передачі і регістр прийому апаратно розділені, але доступ до обох забезпечується через регістр SnBUF.

SM0:SM1		Режим	Опис	Частота передачі
0	0	0	8 біт передаються через RxD. На TxD формується тактова послідовність.	Osc/16
0	1	1	10 біт (старт-біт, 8 біт даних, стоп-біт) передаються через TxD / приймаються через RxD	$\frac{\text{Osc}}{16 \cdot \text{N} \cdot (\text{Timer\_Range} - \text{Reload\_Value})}$, де N – коефіцієнт масштабувача, Timer_Range = 256 для таймера 1 в режимі 2, 65536 для таймера 1 в режимі 0 та для таймера 2
1	0	2	11 біт (старт-біт, 9 біт даних, стоп-біт) передаються через TxD / приймаються через RxD	Osc/32
1	1	3	Як в режимі 2	Як в режимі 1

Передавач двічі буферизований, що дозволяє зменшити до мінімуму інтервал часу між передаванням двох байтів. Для використання цієї особливості флаг передавача TI встановлюється не тільки по закінченні передавання байта, а і в тому випадку, якщо відбувається запис в регістр SnBUF, коли UART не передає дані (тобто другий буфер вільний і можна відразу записати в SnBUF наступний байт для передачі). Однак, якщо UART орієнтований на передачу окремих байтів, а не багатобайтових повідомлень, то для кожного байта фактично буде генеруватись два переривання передавача. Щоб уникнути подвійного буферування, флаг TI слід скидати відразу після запису байта в регістр SnBUF. Якщо при цьому є можливість виникнення переривань вищого пріоритету, доцільно заборонити всі переривання перед записом в SnBUF і відновити дозвіл після скидання TI.

При роботі в режимах 2 та 3 слід враховувати, що біт TB8 не є двічі буферизованим. Тому, якщо у цих режимах передається неперервна послідовність байт з різними значеннями в 9-му біті, слід забороняти подвійне буферування, як вказано вище.

При роботі в мультипроцесорних системах можна задавати режим автоматичного розпізнавання адрес встановленням біту SM2 у регістрі SnCON. Флаг переривання приймача у цьому режимі встановлюється тільки тоді, коли дев'ятий біт даних дорівнює 1 (що вказує на передавання байта адреси, а не байта даних), і отримана адреса співпадає з власною адресою приймача, яка знаходиться у регістрі SADDR. Регістр SADEN використовується для задання маски, у якій нульові біти означають, що відповідний біт адреси може бути довільним. Це дозволяє адресувати кілька керованих пристроїв однією адресою. При скиданні до SADDR та SADEN записуються нульові значення, що означає відсутність автоматичного розпізнавання адрес, тобто флаг RI буде встановлюватись при прийомі будь-яких даних навіть при SM2 = 1.

Регістр SnCON (адреса 420h, 424h)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0, SM1 – біти вибору режиму

SM2 – біт дозволу мультипроцесорного зв'язку у режимах 2 та 3. При SM2 = 1 флаг RI не встановлюється, якщо у прийнятих даних 9-й біт (RB8) дорівнює 0 або якщо в режимі 1 не був прийнятий стоп-біт.

REN – біт дозволу послідовного прийому.

TB8 – дев'ятий біт даних, що передається в режимах 2 та 3. Встановлюється і скидається програмно.

RB8– дев'ятий біт даних, що приймається в режимах 2 та 3. В режимі 1 при SM2 = 0 в RB8 записується стоп-біт. В режимі 0 не використовується.

TI – флаг переривання передавача. Встановлюється по закінченні передавання байта, а також при записі в регістр SnBUF, коли UART не передає дані. Скидається програмно.

RI – флаг переривання приймача. Встановлюється по закінченні прийому байта. Скидається програмно.

Регістр SnSTAT (адреса 421h, 425h)

-	-	-	-	FEn	BRn	OEn	STINTn

FEn – флаг помилки, встановлюється, якщо приймач не прийняв стоп-біт.
BRn – флаг, який встановлюється, якщо всі біти у прийнятому байті нульові.
OEn – флаг, який встановлюється, якщо прийнятий новий байт до того, як значення у буфері прийому буде зчитано програмно і скинутий флаг RI.
STINTn – флаг, який повинен бути встановлений, щоб генерувалось переривання по прийому байта при встановленні будь-якого з флагів FEn, BRn, OEn.
Всі флаги в регістрі SnSTAT повинні скидатись програмно.

2.6 Інтерфейс I²C XA-S3

Інтерфейс I²C мікроконтролера XA-S3 відповідає специфікації 100 кГц, хоча може використовувати і швидкості передачі до 400 кГц.

I²C використовує для передачі дві лінії – SCL (serial clock line) – вивід тактування (P5.6) і SDA (serial data line) – вивід даних (P5.7). Перед початком роботи необхідно встановити для цих виводів режим виводів з відкритим стоком (P5CFGA.6 = P5CFGB.6 = P5CFGA.7 = P5CFGB.7 = 0).

При передаванні даних один з пристроїв є керуючим, інший – керованим. Як керуючих, так і керованих пристроїв може бути кілька. Кожен пристрій має 7-розрядну адресу. Керуючий пристрій ініціює передачу і генерує сигнал тактування.

Формат передачі:

Джерело	m	m	m	s	m/s	s/m	m/s	s/m	m
	START	Адреса керованого	R/W	A	Дані	A	Дані	A	STOP
		7 біт	1 біт		8 біт		8 біт		

m (master) - керуючий
s (slave) - керований

START – фронт спаду на лінії SDA при одиничному рівні на SCL.
R/W – біт читання/запису (читання – 1, запис – 0)
A (acknowledgement) – підтвердження. Нульовий рівень на SDA означає підтвердження, одиничний – не-підтвердження.
STOP– фронт наростання на лінії SDA при одиничному рівні на SCL.

Передається спочатку сигнал START, потім 7-розрядна адреса керованого пристрою, якому призначаються дані, та біт R/W . Керований пристрій аналізує адресу, і якщо вона дорівнює його власній або якщо це адреса глобального виклику (тобто прийнятий байт 00000000), генерує підтвердження. Якщо керуючий не отримує сигналу підтвердження, він припиняє передачу даних. При отриманні підтвердження прийому адреси керуючим передаються (при R/W = 0) або приймаються (при R/W = 1) байти даних, після кожного з яких слідує такт підтвердження прийому. Сигнал підтвердження генерується тим пристроєм, який приймає дані. По завершенні передачі керуючим пристроєм генерується сигнал STOP, який вказує на звільнення шини.

Для роботи з I²C використовуються наступні регістри:
I2CON – регістр управління;
I2STA – регістр статуса;
I2DAT – регістр даних;
I2ADR – регістр адреси.

Регістр I2ADR містить у старших бітах 7-розрядну адресу пристрою, а в молодшому – біт глобального виклику GC (global call). Якщо цей біт встановлений в 1, то ХА буде генерувати підтвердження у випадку глобального виклику, тобто прийняття адреси 0000 000 з бітом R/W=0.

Регістр I2CON

CR2	ENA	STA	STO	SI	AA	CR1	CR0

CR2,CR1,CR0 – біти, що задають швидкість передачі при роботі в режимі керуючого. Варіанти швидкостей при різних частотах осцилятора приведені в таблиці. Робота з швидкостями до 100 кГц відповідає специфікації I^2C 100 кГц, від 100 до 400 кГц - не повністю відповідає специфікації 400 кГц, але може використовуватися, якщо потрібна висока швидкість передачі.

CR2:CR0	Коеф. поділювача	Швидкість передачі при частотах осцилятора, МГц					
		8	12	16	20	24	30
000	20	(400)	-	-	-	-	-
001	40	(200)	(300)	(400)	-	-	-
010	68	(116.65)	(176.46)	(235.29)	(294.12)	(352.94)	-
011	88	90.91	(136.36)	(181.82)	(227.27)	(272.73)	(340.91)
100	160	50	75	100	(125)	(150)	(187.5)
101	272	29.41	44.12	58.82	73.53	88.24	(110.29)
110	352	22.73	34.09	45.45	56.82	68.18	85.23
111	(Таймер 1)	(Тайм. 1)	(Тайм. 1)	(Таймер 1)	(Таймер 1)	(Тайм/ 1)	(Тайм. 1)

При використанні таймера 1 швидкість передачі є вдвічі меншою, ніж частота переповнення таймера.

ENA - біт дозволу I^2C.
Якщо ENA = 0, то вхідні сигнали на SDA і SCL ігноруються, пристрій знаходиться в режимі неадресованого керованого, біт STO скидається. В цьому випадку виводи P5.6 і P5.7 можуть використовуватись як виводи з відкритим стоком.
При ENA = 1 виводи P5.6 і P5.7 необхідно встановити в 1.
ENA не слід використовувати для тимчасового відключення від шини I^2C, оскільки при скиданні ENA втрачається інформація про стан шини. Для цього служить флаг АА.

STA – старт-флаг. Цей флаг необхідно встановити в 1 для переходу в режим керуючого. I^2C перевіряє стан шини, і якщо шина вільна, генерує сигнал START , якщо зайнята – чекає на сигнал STOP, тобто звільнення шини, і генерує сигнал START після затримки на півперіода тактової послідовності.
Якщо STA встановлений, коли I^2C вже знаходиться в режимі керуючого, сигнал START генерується повторно. STA може бути встановлений в будь-який час, в т.ч. коли I^2C знаходиться в режимі адресованого керованого.

STO – стоп-флаг. Коли він встановлюється в 1 в режимі керуючого, на шину передається сигнал STOP. Як тільки STOP з'являється на шині, флаг STO скидається апаратно.
В режимі керованого STO може бути встановлений для виправлення помилки. В цьому випадку на шину не передається STOP, але апаратна частина спрацьовує так само, як при прийнятті сигнала STOP, і переключається в режим неадресованого керованого приймача.

SI – флаг переривання послідовного порту. Він встановлюється апаратно при кожному новому стані I^2C (крім стану F8h, який вказує на відсутність коректної інформації про стан). Якщо при

цьому встановлені флаги дозволу EA (дозвіл всіх переривань) та EI2 (дозвіл переривання I^2C), то генерується переривання I^2C. При встановленні SI припиняється послідовна передача даних, і на SCL встановлюється нульовий рівень.

AA – флаг підтвердження. Встановлення AA викликає генерування підтвердження (нульовий рівень на SDA) на протязі такту підтвердження на SCL для таких подій:
- прийнята адреса керованого;
- прийнята адреса глобального виклику, якщо біт GC(general call) = 1;
- прийнятий байт даних в режимі керуючого-приймача;
- прийнятий байт даних в режимі адресованого керованого приймача.

При AA = 0 генерується не-підтвердження (одиничний рівень на SDA) на протязі такту підтвердження на SCL для таких подій:
- прийнятий байт даних в режимі керуючого-приймача;
- прийнятий байт даних в режимі адресованого керованого приймача.

Якщо I^2C знаходиться в режимі адресованого керованого передавача, то відбудеться перехід у стан C8h після передачі останнього байту даних. Коли SI скидається, I^2C виходить зі стану C8h, переходить в стан неадресованого керованого приймача, і SDA залишається в одиничному стані. У стані C8h AA може бути знову встановлений для розпізнавання наступної адреси.
Якщо I^2C знаходиться в режимі неадресованого керованого, то прийнята адреса ігнорується, підтвердження не генерується і переривання не виникає. Таким чином, апаратна частина може бути ізольована від шини I^2C, хоча стан шини контролюється, тобто приймаються сигнали START, STOP і послідовні дані. Розпізнавання адреси може бути відновлене у будь-який час встановленням флагу AA. Якщо це відбудеться у момент, коли адреса вже частково прийнята, розпізнавання адреси відбудеться у кінці передавання байта.

Регістр I2STA – це однобайтовий регістр, у якому 3 молодші біти не використовуються (нульові), а 5 старших містять код стану I^2C. Всіх можливих кодів 26. Код F8h означає, що інформація про стан відсутня. При зміні I2STA генерується переривання I^2C (встановлюється біт SI в регістрі I2CON).

2.7 Аналого-цифровий перетворювач XA-S3

До складу XA-S3 входить 8-канальний 8-розрядний АЦП. Є можливість здійснення 10-розрядного АЦ перетворення. Вхідна напруга може бути в межах 0…AVdd (максимум 3.3 В). Входи АЦП містяться у порті 5. Виводи повинні бути відконфігуровані як високоомні виводи Для використання АЦП необхідно подати відповідні сигнали на входи AVdd, AVss, AVref+, AVref- (живлення, опорні напруги).

Регістр ADCON(адреса 43Eh)

-	-	-	-	ADRES	ADMOD	ADSST	ADINT

ADRES – біт вибору розрядності АЦ перетворення: 0 – 8-розрядне, 1 – 10-розрядне.
ADMOD – біт вибору режиму перетворення: 1 –неперервне сканування, 0 – одиночне сканування.
ADSST – біт старта і статуса
ADINT – флаг завершення АЦ перетворення. Повинен скидатись програмно.

АЦП підтримує режими неперервного та одиночного перетворення. В обох режимах перетворення може виконуватись для одного чи кількох каналів. Щоб дозволити виконання перетворення по деякому каналу, необхідно встановити біт дозволу, номер якого в регістрі ADCS відповідає номеру канала. Результати перетворення зберігаються у регістрах ADRSHn (по одному регістру на канал). У режимі 10-розрядного перетворення два молодші біти

результату зберігаються в регістрі ADRSL (один для всіх каналів) і повинні бути прочитані до початку наступного перетворення.

Щоб почати АЦ перетворення, необхідно встановити біт ADSST в регістрі ADCON. В режимі одиночного сканування відбувається одне АЦ перетворення для всіх каналів, які дозволені бітами регістру ADCS. В режимі неперервного сканування перетворення відбувається неперервно.

По закінченні одного перетворення для всіх каналів встановлюється флаг ADINT і генерується переривання, якщо воно дозволене і його пріоритет вищий, ніж у поточної задачі.

Для задання часу вибірки та перетворення використовується регістр ADCFG, у якого 4 старші біти зарезервовані, а значення 4 молодших визначають часові характеристики перетворення:

ADCFG.3 ... 0	Максимальна частота осцилятора (МГц)		Час перетворення				Час вибірки (в тактах)
	8 бітний режим	10 бітний режим	в тактах		в мкс для макс. частоти осцилятора		
			8 біт	10 біт	8 біт	10 біт	
0h(0000)	6.66	6.66	72	88	10.81	13.21	4
1h(0001)	10	8	76	92	7.6	9.2	6
2h(0010)	11.11	8	80	96	7.2	8.64	8
3h(0011)	13.33	12	96	116	7.2	8.7	8
4h(0100)	16.66	12	100	120	6.0	7.2	10
5h(0101)	20	12	104	124	5.2	6.2	12
6h(0110)	20	12	116	136	5.8	6.8	24
7h(0111)	22.2	12	108	128	4.86	5.77	14
8h(1000)	23.3	13	124	148	5.32	6.35	14
9h(1001)	26.6	13	128	152	4.81	5.71	16
Ah(1010)	30	13	132	156	4.4	5.2	18
Bh(1011)1	30	13	146	170	4.87	5.67	32
Ch(1100)	–	13	136	160	4.25	5.0	20
Dh(1101)	–	16	152	180	4.56	5.41	20
Eh(1110)	–	20	172	204	4.7	5.57	22
Fh(1111)	–	20	176	208	4.4	5.2	24

В 10-бітному режимі похибка від теплових шумів при одиночному АЦ перетворенні складає ±1,25 молодшого розряду. Щоб виключити цю похибку, рекомендується, якщо це можливо, здійснювати усереднення по 16 значеннях.

2.8 Режими роботи XA

2.8.1 Скидання

При скиданні відбувається наступна послідовність подій:

- генерується зовнішній апаратний сигнал скидання (нульовий рівень на виводі RST);
- виконується апаратна ініціалізація регістрів;
- на виводі RST встановлюється одиничний рівень;
- визначається конфігурація пам'яті та зовнішньої шини;
- генерується переривання скидання;
- починає виконуватись програма.

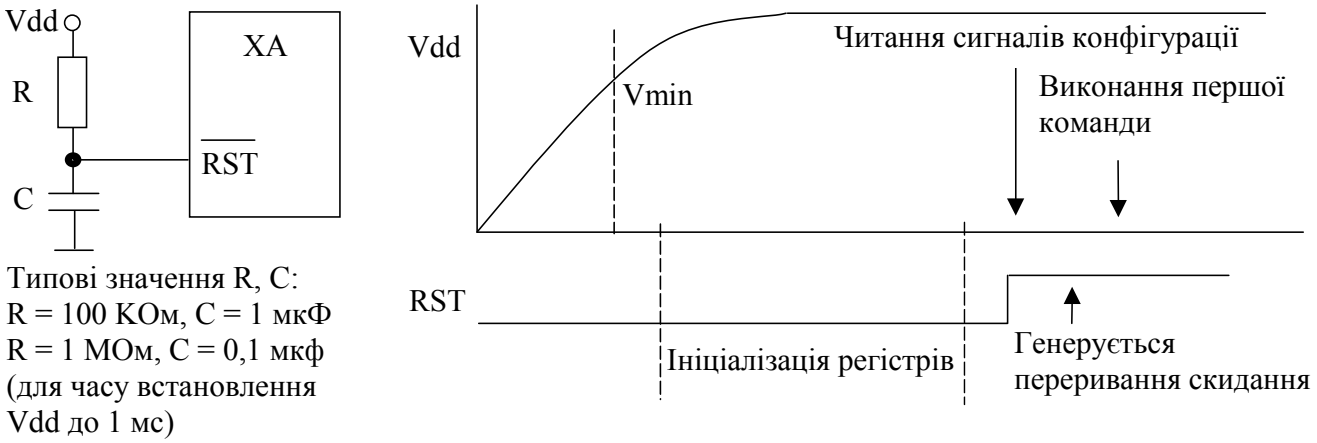


Рисунок 2.18 - Схема апаратного скиду XA та відповідні йому часові діаграми

При ініціалізації регістрів нульові значення записуються у всі сегментні регістри, регістр SSEL, регістри загального призначення, біти PZ, CM і EA, обидва покажчики стеку ініціалізуються значенням 0100h та всі виводи портів конфігуруються як входи (квазідвонаправлені виводи зі значенням в регістрі порту FFh). До більшості регістрів спеціальних функцій записуються нульові значення.

Нульовий рівень на виводі RST повинен тривати не менше 10мс, що забезпечує стабілізацію осцилятора та ідентифікацію контролером режиму скидання. По фронту наростання на виводі RST читаються значення входів EA та BUSW, що визначають конфігурацію пам'яті та зовнішньої шини. Якщо EA = 0, XA буде працювати з зовнішньою пам'яттю програм, якщо EA = 1 – з внутрішньою. Якщо BUSW = 0, розрядність зовнішньої шини 8 біт, якщо BUSW = 1 – 16 біт.

Відразу після встановлення одиничного рівня на RST генерується переривання скидання. При цьому з вектора переривання за адресою 0 у пам'яті програм зчитується значення регістру ознак PSW та адреса першої команди. Значення молодшого байта PSW, як правило, не має значення і може бути 0 або довільним. Значення старшого байту визначає режим (системний чи користувача) та пріоритет коду, що буде виконуватись. Рекомендується встановлювати при скиданні системний режим (PSWH.7 = 1) та найвищий рівень пріоритету 15 (PSWH.3 … PSWH.0 = 1).

Перша команда, яка буде виконуватись після скидання, повинна розміщуватись в нульовому сегменті, оскільки вектор скидання містить 16-розрядну адресу, а значення сегментного регістру CS після скидання є нульовим. Оскільки за адресами 000h – 11Bh розміщується таблиця векторів переривань, програмний код можна розміщувати починаючи з адреси, більшої 11Bh.

Приклад ініціалізуючого коду:

```
code_seg        ; кодовий сегмент
org 0h          ; адреса 0 у пам’яті програм
dw 8F00h        ;слово для завантаження PSW ( системний режим, найвищ. пріоритет)
dw start        ; адреса першої команди в кодовому сегменті

org 120h        ; адреса 120h у пам’яті програм
start:
…               ; початок програми
```

Рекомендується першою командою виконати завантаження регістра системної конфігурації SCR, щоб визначити режими роботи XA (режим нульової сторінки, режим сумісності з 80C51, коефіцієнт подільника осцилятора), після чого ініціалізувати покажчики стеків.

Скидання може бути викликане не тільки апаратно, а і програмно (командою RESET), а також відбувається при антипереповненні таймера-сторожа (WDT).

Хоча скидання обробляється як переривання, послідовність команд ініціалізуючого коду не можна завершувати командою RETI, оскільки покажчик стеку міг бути змінений і виконання RETI може призвести до непередбачуваних наслідків.

2.8.2 Управління енергоспоживанням

XA може працювати в холостому режимі та в режимі зменшеного енергоспоживання. Перехід у ці режими здійснюється встановленням біту PD (power-down mode – режим зменшеного енергоспоживання) або IDL (idle mode – холостий режим) в регістрі PCON. Якщо встановлені обидва біти, XA переходить в режим зменшеного енергоспоживання.

В холостому режимі продовжує працювати осцилятор та деякі периферійні пристрої. Вихід з цього режиму відбувається при виникненні переривання від периферійних пристроїв.

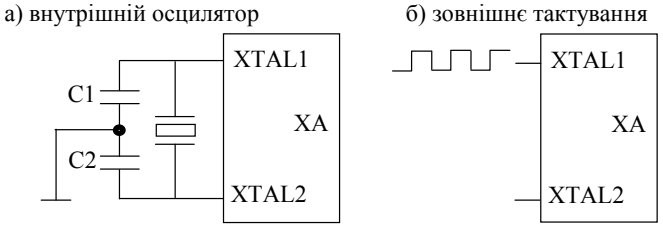
В режимі зменшеного енергоспоживання відключаються всі блоки, включаючи осцилятор. Вміст внутрішніх регістрів, регістрів спеціальних функцій та внутрішнього ОЗП зберігається. Вихід з режиму можливий через скидання або зовнішнє переривання. В останньому випадку зовнішнє переривання повинне бути дозволеним і генеруватися по рівню, а не по фронту. Щоб забезпечити стабілізацію осцилятора при переході в робочий режим через зовнішнє переривання, виконання програми починається тільки після інтервалу часу, рівного 9,892 тактам осцилятора.

Регістр PCON (адреса 404h)

-	-	-	-	-	-	PD	IDL

2.9 Синхронізація

Для тактування XA можна використовувати як внутрішній осцилятор, так і зовнішнє джерело тактування. У першому випадку до виводів XTAL1 та XTAL2 під’єднується кварцевий або керамічний резонатор.



Типові значення C1, C2:
для частот до 25 Мгц: 28 … 34 пФ
для частот > 25 Мгц: 18 … 24 пФ

Рисунок 2.19 - Схеми тактування XA

2.10 Система команд XA

Команди пересилки даних	
MOV	пересилка байтів або слів
MOVC	пересилка до регістру байта або слова з пам’яті команд
MOVS	пересилка знакової 4-бітової константи (-8 <= x <= 8) до регістру
MOVX	пересилка до регістру з зовнішньої пам’яті даних
PUSH	занесення байта або слова до стеку
PUSHU	занесення байта або слова до стеку користувача
POP	читання зі стеку байта або слова
POPU	читання зі стеку користувача байта або слова
XCH	обмін байтів або слів
LEA	завантажити 16-бітову виконавчу адресу
Арифметичні операції	
ADD	додавання байтів або слів
ADDC	додавання байтів або слів з врахуванням флага переносу
ADDS	додавання знакової 4-бітової константи (-8 <= x <= 8)
CMP	порівняння байтів, слів
DA	двійкова корекція байтів для двійково-десяткового формату BCD
DIV16	ділення знакових 16 біт / 8 біт
DIV32	ділення знакових 32 біт / 16 біт
DIVU8	ділення беззнакових 8 біт / 8 біт
DIVU16	ділення беззнакових 16 біт / 8 біт
DIVU32	ділення беззнакових 32 біт / 16 біт
MUL16	множення знакових 16 біт * 16 біт
MULU8	множення беззнакових 8 біт * 8 біт
MULU16	множення беззнакових 16 біт * 16 біт
NEG	зміна знаку
SEXT	розширення знаку
SUB	віднімання байтів, слів
SUBB	віднімання байтів, слів з врахуванням флага переносу
Логічні операції	
AND	логічне І байтів або слів
CPL	побітова інверсія регістра
OR	логічне АБО байтів або слів
XOR	ВИКЛЮЧАЮЧЕ АБО байтів або слів
Зсуви	
RL	зсув вліво (MSB -> LSB) байта або слова
RR	зсув вправо (LSB -> MSB) байта або слова
RLC	зсув вліво через переніс байта або слова
RRC	зсув вправо через переніс байта або слова
LSR	зсув вправо (з заповненням 0) байта, слова, подвійного слова
ASL	зсув вліво (з заповненням 0) байта, слова, подвійного слова
ASR	зсув вправо (з розширенням знаку) байта, слова, подвійного слова
NORM	нормалізація (зсув до появи 1 в старшому біті) байта, слова, подвійного слова

Бітові команди	
ANL	логічне І біту і переносу
CLR	обнулення біт
MOV	перенесення біту до (з) флагу переносу
ORL	логічне АБО біту і переносу
SETB	встановлення біту в 1
Команди переходів	
BCC	перехід якщо флаг переносу 0
BCS	перехід якщо флаг переносу 1
BEQ	перехід якщо нуль
BNE	перехід якщо не нуль
BG	перехід якщо більше (для беззнакових)
BL	перехід якщо менше (для беззнакових)
BGT	перехід якщо більше (для знакових)
BGE	перехід якщо більше або рівне (для знакових)
BLT	перехід якщо менше (для знакових)
BLE	перехід якщо менше або рівне (для знакових)
BMI	перехід якщо результат від'ємний
BPL	перехід якщо результат додатній
BNV	перехід якщо немає переповнення
BOV	перехід якщо переповнення
JNZ	перехід якщо акумулятор (R4L) не дорівнює нулю
JZ	перехід якщо акумулятор (R4L) дорівнює нулю
CJNE	порівняння і перехід якщо не рівне
DJNZ	декремент і перехід якщо нуль
JB	перехід якщо біт дорівнює 1
JBC	перехід якщо біт дорівнює 1 з наступним обнуленням
JNB	перехід якщо біт дорівнює 0
BR	безумовний перехід (+-256 байт)
JMP	безумовний перехід
FJMP	міжсегментний перехід
CALL	виклик підпрограми
FCALL	міжсегментний виклик підпрограми
RET	повернення з підпрограми
RETI	повернення з переривання
Виключення / Переривання	
BKPT	точка переривання (для відладки)
RESET	скидання
TRAP	1 з 16 програмних переривань (для виклику системних функцій в мультизадачних системах)
NOP	(НЕМАЄ ОПЕРАЦІЇ)

3. 8-розрядні мікроконтролери ATmega603, ATmega103

3.1 Особливості архітектури

• Використана AVR розширена RISC архітектура

• Потужний набір з 121 команди, більшість яких виконується за один машинний цикл

• Ємність внутрісистемно програмованої Flash пам'яті 64 Кбайт (ATmega603/L) і 128 Кбайт (ATmega 103/L), 1000 циклів стирання/запису

• SPI інтерфейс внутрісистемного програмування

• Ємність вбудованої EEPROM 2 Кбайт (ATmega 603/L) і 4 Кбайт (ATmega 603/L), 100000 циклів стирання/запису

• Вбудована RAM ємністю 4 Кбайт

• 32 8-розрядних регістра загального призначення, набір регістрів керування периферією

• 32 програмовані лінії I/O, 8 ліній виходу, 8 ліній входу

• Програмовані послідовні UART і SPI інтерфейси

• Діапазон напруг живлення від 2,7 В до 6,0 В (ATmega603L/ Atmega103L) і від 4,0 В до 6,0 В (ATmega603/ Atmega103)

• Діапазон тактових частот від 0 до 4 МГц (ATmega603L/ Atmega103L) і від 0 до 6 МГц (ATmega603/ Atmega103)

• Продуктивність до 6 MIPS при частоті 6 МГц

• Вбудована система реального часу з окремим генератором

• Два 8-розрядних таймери/лічильника з окремим переддільником та ШІМ

•16-розрядний таймер/лічильник з окремим переддільником, режимами захоплення/порівняння і подвійним ШІМ з розрядністю 8, 9 чи 10 розрядів

• Програмований сторожовий таймер з вбудованим генератором

• Вбудований аналоговий компаратор

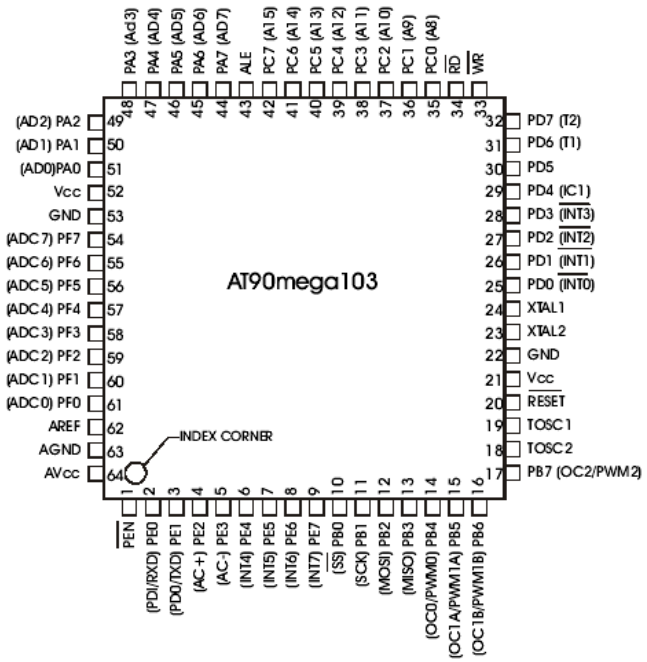
• 8-канальний 10-розрядний аналого-цифровий перетворювач

• Режими енергозбереження Idle, Power Save і Power Down

• Програмне встановлення тактової частоти

• Програмне блокування захисту програмних засобів

Розташування виводів:



3.2 Опис

Прилади ATmega603/103 є 8-розрядними CMOS мікроконтролерами з AVR удосконаленою RISC архітектурою. Виконуючи більшість команд за один тактовий цикл, мікроконтролери ATmega603/103 забезпечують продуктивність 1 MIPS на кожен мегагерц тактової частоти, що дозволяє розроблювачам оптимізувати споживання, що залежить в основному від тактової частоти.

AVR ядро базується на удосконаленій RISC архітектурі, з реєстровим файлом швидкого доступу, що містить 32 регістра загального призначення, безпосередньо зв'язаних з арифметико-логічним пристроєм (ALU), і потужною системою команд. За один тактовий цикл із реєстрового файлу витягаються два операнда, виконується команда і результат записується в регістр призначення. Така високоефективна архітектура забезпечує продуктивність майже в десять разів більшу, ніж стандартні CISC мікроконтролери.

Мікроконтролери ATmega603/103 мають у своєму розпорядженні наступні можливості: 64/128 Кбайт внутрісистемно програмованої Flash пам'яті програм, 2/4 Кбайт EEPROM даних, 4 Кбайт SRAM даних, 32 лінії I/O загального призначення, 8 ліній входу, 8 ліній виходу, 32 робочих регістра загального призначення, 4 гнучких таймери/лічильника з режимами порівняння, PWM і UART, програмувальний сторожовий таймер з вбудованим власним генератором, послідовний SPI порт і три програмно встановлюваних режими енергозбереження. У режимі Idle зупиняється центральний процесор, але продовжують працювати SRAM, таймери/лічильники, порт SPI і система переривань. У режимі Power Down зберігається вміст регістрів, але зупиняється тактовий генератор і до надходження сигналу чи переривання апаратного блоку забороняється виконання усіх функцій мікроконтролера. У режимі Power Save усі пристрої знаходяться в режимі «сну», але генератор продовжує працювати, забезпечуючи збереження тимчасової бази.

Прилади виготовляються по технології енергонезалежної пам'яті фірми Atmel. Вбудована ISP Flash пам'ять програм може бути перепрограмована безпосередньо в системі, з використанням послідовного SPI інтерфейсу, чи за допомогою звичайних програматорів енергонезалежної пам'яті. Об'єднавши 8-розрядне RISC CPU з внутрісистемно програмованою Flash пам'яттю великого об'єму, фірма створила сімейство потужних мікроконтролерів, що забезпечує реалізацію недорогих і дуже зручних рішень для великої кількості вбудовуваних застосувань. Сімейство ATmega603/103 підтримується великою кількістю засобів розробки програм та систем, що включають: С-компілятори, макроассемблери, відлаштувальники/симулятори програм, внутрісхемні емулятори та відлаштувальні пристрої.

Порівняння ATmega603 і ATmega103

Мікроконтролер ATmega603 оснащений внутрішньо-системно програмованою Flash пам'яттю ємністю 64 Кбайт, 2 Кбайт EEPROM і 4 Кбайт SRAM і не виконує команду ELPM. Мікроконтролер ATmega103 оснащений внутрісисгемно програмованою Flash пам'яттю ємністю 128 Кбайт, 4 Кбайт EEPROM і 4 Кбайт SRAM. У систему команд цього мікроконтролера включена команд ELPM, необхідна для забезпечення неперервного табличного пошуку у старшій половині адрес Flash пам'яті.

Відмінності в обсязі пам'яті цих двох приладів:

Тип пристрою	Об'єм Flash пам'яті	Об'єм EEPROM	Об'єм SRAM
ATmega603	64 Кбайт	2 Кбайт	4 Кбайт
ATmega103	128 Кбайт	4 Кбайт	4 Кбайт

3.3 Призначення виводів

VCC Напруга живлення

GND Земля

Port A (PA7..PA0) 8-розрядний двонаправлений порт I/O. До виходів порту можуть бути

підключені вбудовані навантажувальні резистори (окремо до кожного розряду). Вихідні буфери забезпечують струм, що втікає, в 20 мА і здатні напряму керувати LED індикатором. При використанні виводів порту як входи й установці зовнішнім сигналом у низький стан, так буде випливати тільки при підключених вбудованих навантажувальних резисторах.

Порт А, при наявності зовнішньої SRAM, використовується в якості мультиплексованої шини адреси/даних.

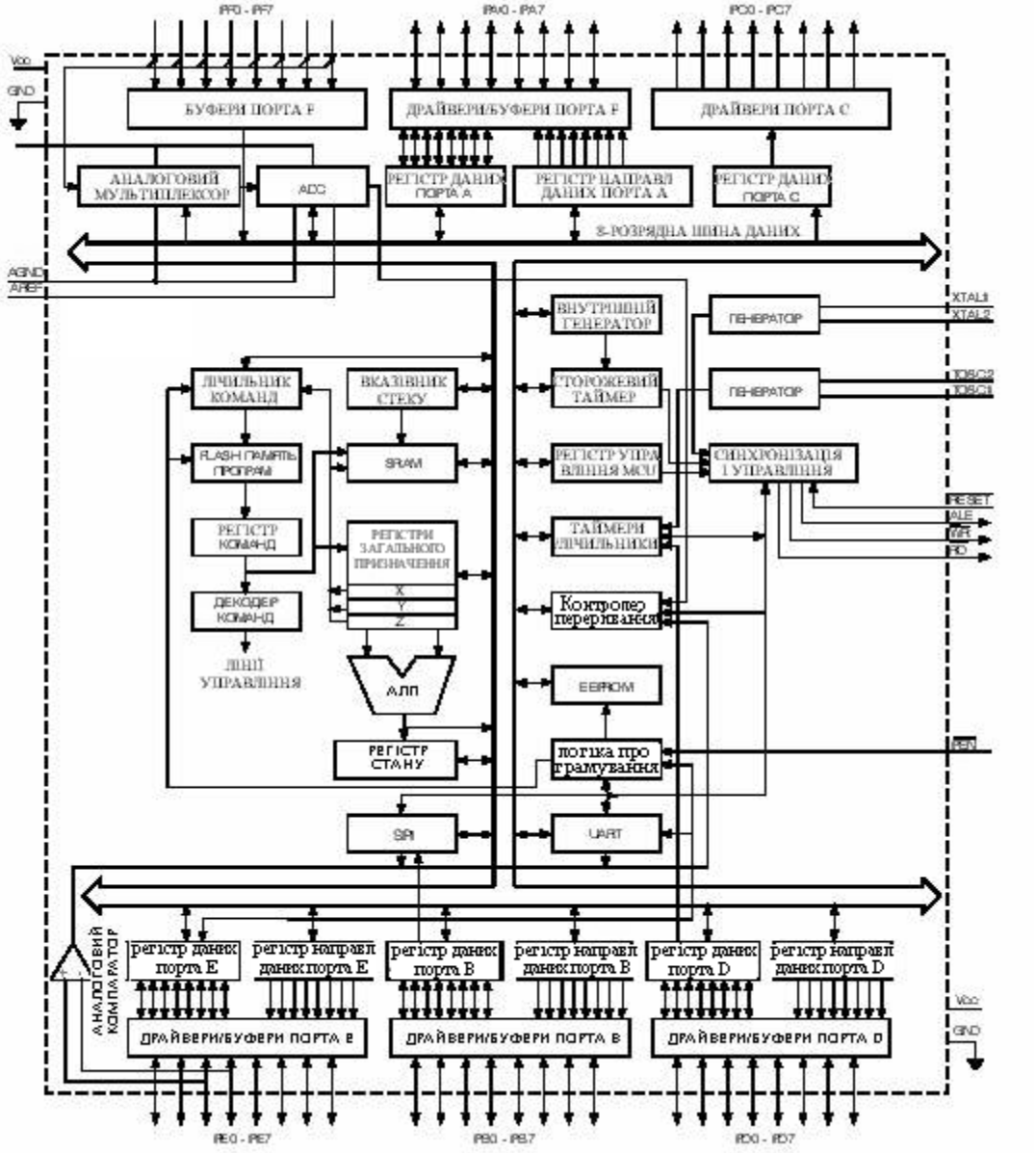


Рисунок 3.1 - Блок схема мікроконтролерів ATmega603/103

PortB(PB7..PB0) 8-розрядний двонаправлений порт I/O з вбудованими навантажувальними резисторами. Вихідні буфери забезпечують струм, що втікає, в 20 мА. При використанні виводів порту в якості входів й установці зовнішнім сигналом у низький стан, струм буде витікати тільки при підключених вбудованих навантажувальних резисторах. Порт В використовується також при реалізації різних спеціальних функцій.

PortC(PC7..PC0) 8-розрядний порт виходу. Вихідні буфери забезпечують струм, що втікає, в 20 мА. Порт С використовуються також як виходи адреси при використанні зовнішньої SRAM.

PortD(PD7..PD0) 8-розрядний двонаправлений порт I/O з вбудованими навантажувальними резисторами. Вихідні буфери забезпечують вхідний струм 20 мА. При використанні виводів порту як входів й установці зовнішнім сигналом у низький стан, струм буде витікати тільки при підключених вбудованих навантажувальних резисторах.

PortE(PE7..PE0) 8-розрядний двонаправлений порт I/O з вбудованими навантажувальними резисторами. Вихідні буфери забезпечують струм, що втікає, в 20 ма. При використанні виводів порту як входів й установці зовнішнім сигналом у низький стан, струм, що витікає через них, забезпечується тільки при підключених вбудованих навантажувальних резисторах.

PortF(PF7..PF0) 8-розрядний порт входу. Входи порту використовуються також як аналогові входи аналого-цифрового перетворювача.

RESET Вхід скидання. Для виконання скидання необхідно утримувати низький рівень на вході протягом двох машинних циклів.

XTAL1 Вхід інвертуючого підсилювача генератора і вхід схеми вбудованого генератора тактової частоти.

XTAL2 Вихід інвертуючого підсилювача генератора.

TOSC1 Вхід інвертуючого підсилювача генератора таймера/лічильника.

TOSC2 Вихід інвертуючого підсилювача генератора таймера/лічильника.

WR Строб запису зовнішньої SRAM.

RD Строб читання зовнішньої SRAM.

ALE Строб дозволу фіксації адреси, використовуваний для дозволу зовнішньої пам'яті. Строб ALE використовується для фіксації молодшого байта адреси в засувках адреси протягом циклу звертання, протягом циклу звертання, при звертанні до даних використовуються виводи AD0-AD7.

AVCC Напруга живлення аналого-цифрового перетворювача. Вивід приєднується до зовнішнього VCC через низькочастотний фільтр.

AREF Вхід аналогової напруги порівняння для аналого-цифрового перетворювача. На цей вивід, для забезпечення роботи аналого-цифрового перетворювача, подається напруга в діапазоні між AGND і AVCC.

AGND Цей вивід повинен бути під'єднаний до окремої аналогової землі, якщо плата оснащена нею. В іншому випадку вивід приєднується до загальної землі.

PEN Вивід дозволу програмування в низьковольтному послідовному режимі програмування. При утриманні цього виводу на низькому рівні під час скидання по включенні живлення, прилад перейде в режим програмування по послідовному каналу.

3.4 Тактовий генератор

XTAL1 і XTAL2 є входом і виходом, відповідно, інвертуючого підсилювача, який з використанням кварцового кристалу чи керамічного резонатора працює як вбудований генератор, як показано на Рис.3.2. При використанні зовнішнього джерела тактової частоти вивід XTAL2 повинен залишитися вільним, сигнал подається на вивід XTAL1, як показано на Рис.3.3. Кварцовий кристал генератора таймера приєднується безпосередньо до виводів OSC1 і OSC2. Зовнішні конденсатори не потрібні. Генератор оптимізований під годинний кварц із частотою 32,768 КГц. Зовнішній тактовий сигнал, що подається на ці виводи, надходить на підсилювач зі смугою пропускання 256 КГц. У такий спосіб частота зовнішнього сигналу повинна знаходитися в діапазоні від 0 до 256 КГц.

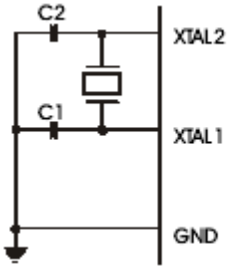


Рисунок 3.2 – Під'єднання тактового генератора

Рисунок 3.3 – Під'єднання зовнішнього джерела тактового сигналу

3.5 Архітектура мікроконтролерів ATmega 603/103

Файл регістрів швидкого доступу, містить 32 8-розрядних робочих регістра загального призначення, зв'язаних безпосередньо з ALU. За один тактовий цикл із файлу регістрів вибираються два операнда, виконується операція і результат знову повертається у файл регістрів.

Шість з 32 регістрів можуть бути використані як три 16-розрядних регістри покажчика непрямої адресації адресного простору даних, що забезпечують ефективне обчислення адрес. Один з цих покажчиків адреси використовується також, як покажчик адреси для функції неперервного перегляду таблиць. Ці 16-розрядні додаткові регістри позначаються X-регістр, Y-регістр і Z-регістр.

ALU підтримує арифметичні і логічні операції між регістрами чи між константою і регістром. Виконуються в ALU і операції з окремими регістрами. На Рис. 3.4 показана AVR розширена RISC архітектура мікроконтролерів ATmega603/103.

В доповнення до операцій з регістрами, регістровий файл може використовуватися і для звичайної адресації пам'яті. Це пояснюється тим, що файл регістрів розташовується по 32 самими молодшими адресами простору даних, і до них можна звертатися як до звичайних комірок пам'яті.

Простір пам'яті I/O містить 64 адреси периферійних функцій CPU таких як: регістри керування, таймери/лічильники, аналогово-цифрові перетворювачі й інші I/O функції. До пам'яті I/O можна звертатися безпосередньо чи як до комірок простору пам'яті відповідним адресам регістра файлів $20-$5F. У мікроконтролерах AVR використані принципи Гарвардської архітектури – окремі пам'ять і шини для програм і даних. При роботі з пам'яттю програм використовується однорівневий конвеєр – у той час, як одна команда виконується, наступна команда вибирається з пам'яті програм. Такий прийом дозволяє виконувати команду в кожному тактовому циклі. Пам'яттю програм є внутрісистемно програмована Flash пам'ять. За малим виключенням AVR команди мають формат одного 16-розрядного слова, у зв'язку з чим кожна адреса пам'яті програм містить одну 16-розрядну команду.

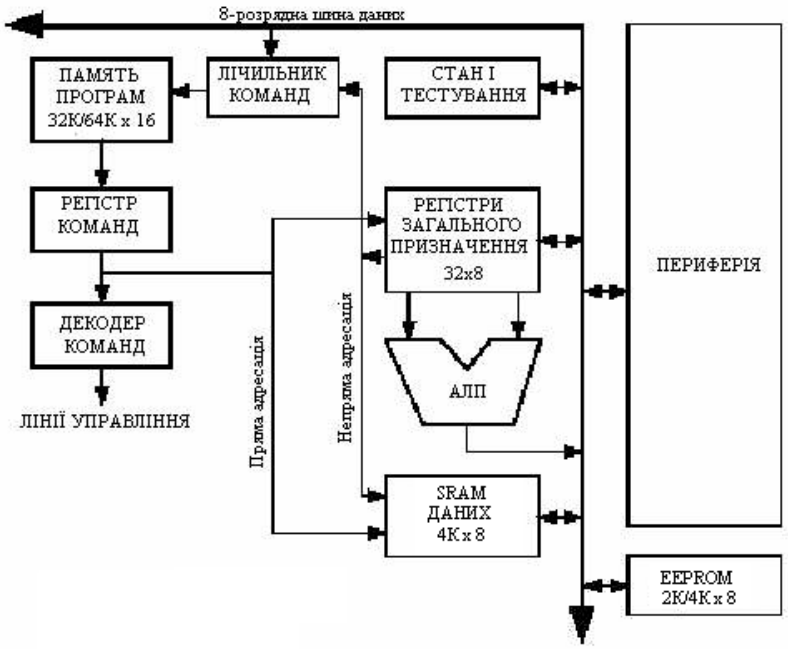


Рисунок 3.4 – Розширена RISC архітектура мікроконтролерів ATmega603/103

У процесі обробки переривань і викликів підпрограм адреса повернення лічильника команд (PC) зберігається в стеці. Стек розміщається в SRAM даних і, отже, розмір стека обмежений тільки загальним розміром SRAM і рівнем її використання. Усі користувацькі

програми в підпрограмах повернення (попередньо, коли підпрограми чи переривання будуть виконуватися) повинні ініціалізувати покажчик стека (SP). 16-розрядний покажчик стека, з можливістю читання/запису розташовується в просторі I/O.

AVR архітектура підтримує п’ять різних режимів адресації 4000 байт SRAM даних.

Гнучкий модуль обробки переривань має в просторі I/O свій керуючий регістр із додатковим бітом дозволу глобального переривання в регістрі статусу. Усі переривання мають свої вектори переривання в таблиці векторів переривання, розташовуваної на початку пам’яті програм. Пріоритети переривань відповідають положенню векторів переривань – переривання з найменшою адресою вектора має найвищий пріоритет.

Усі простори пам’яті AVR архітектури лінійні та регулярні.

3.5.1 Файл регістрів загального призначення

На Рис. 3.5 представлена структура 32 регістрів загального призначення.

Усі регістрові команди звертаються безпосередньо до регістрів протягом одного тактового циклу. Виключенням є п’ять логічних та арифметичних операцій з константами (SBCI, SUBI, CPI і ANDI) і операція ORI між константою і вмістом регістра, і команда безпосереднього завантаження константи LDI. Ці команди використовують другу половину регістрів регістрового файлу - R16..R31.

Самі загальні команди SBC, SUB, CP, AND і OR і всі інші операції між двома регістрами чи з одним регістром використовують для запису результату регістровий файл.

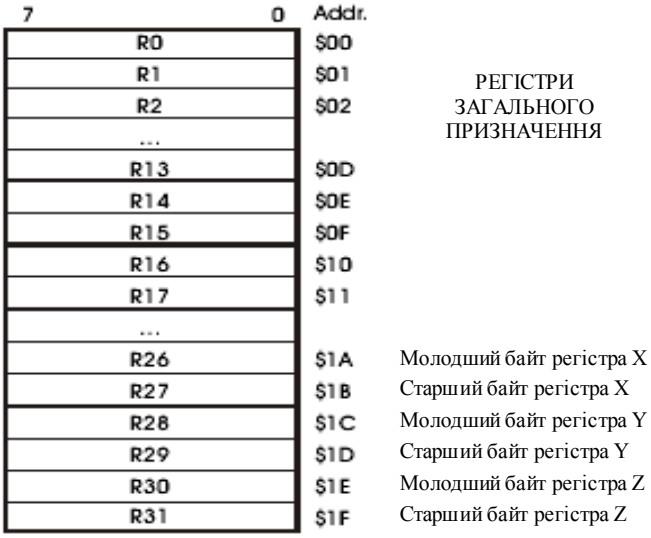


Рисунок 3.5 – Регістри загального призначення CPU мікроконтролерів AVR

Як показано на рис. 3.5, кожному регістру відповідає адреса пам’яті даних, що відображає їх у перших 32 комірках користувацького простору даних. Хоча вони не використовуються як фізичні комірки SRAM, така організація пам’яті забезпечує гнучке звертання до регістрів, оскільки X, Y і Z регістри можуть бути використані для індексації будь-якого регістра у файлі.

SRAM даних має об’єм 4 Кбайт і займає адресний простір від $ 0060 до $0FFF.

3.5.2 Регістри X,Y,Z

Шість регістрів (з R26 по R31) регістрового файлу, крім звичайних для інших регістрів функцій, виконують функцію 16-розрядних регістрів покажчиків адреси при непрямій адресації SRAM. Ці три регістри непрямої адресації визначаються як регістри X,Y і Z.

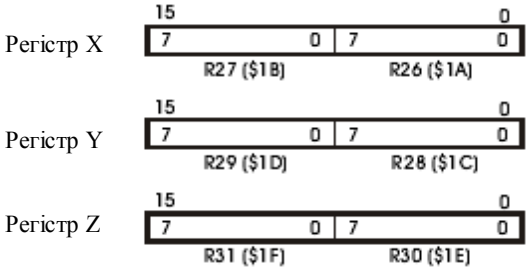


Рисунок 3.6 – Регістри X, Y і Z

В різноманітних режимах адресації ці регістри виконують функції фіксованого зміщення, автоматичного інкременту та декременту (див. опис команд).

3.5.3 ALU - Арифметико-логічний пристрій

Високопродуктивне AVR ALU з'єднано безпосередньо з усіма 32 швидкодіючими регістрами загального призначення. За один тактовий цикл ALU виконує операцію між регістрами цього регістрового файлу. Операції ALU підрозділяються на три основні категорії: арифметичні, логічні й операції над бітами.

3.5.4 Внутрісистемно програмована Flash пам'ять програм

Коди програм мікроконтролерів ATmega603/103 записуються в 64/128 Кбайт вбудованої внутрісистемно програмованої Flash пам'яті. Оскільки всі команди мають формат одного чи двох 16-розрядних слів, то і пам'ять програм має організацію 32/64Кх16. Flash пам'ять забезпечує не менш 1000 циклів стирання/запису.

Таблиці констант можуть бути розміщені в будь-якім місці всього простору пам'яті програм (див. опис команд LPM (Load Program Memory) - Завантажити байт пам'яті програм і ELPM (Extended Load Program Memory) - Завантажити байт пам'яті програм у розширеному режимі).

3.5.5 Конфігурація пам'яті

Мікроконтролери ATmega603/103 підтримують дві конфігурації:

Конфігурація	***Вбудована SRAM даних***	***Зовнішня SRAM даних***
A	4000 байт	Немає
B	4000 байт	До 64 Кбайт (1)

Примітка 1. ***З 64 Кбайт зовнішньої пам'яті будуть доступні 60 Кбайт***

По першим 4096 адресам пам'яті даних розміщаються регістровий файл, простір пам'яті I/O і вбудована SRAM даних. З них перші 96 адрес займають регістровий файл і простір пам'яті I/O, у наступних 4000 адрес розміщується вбудована SRAM.

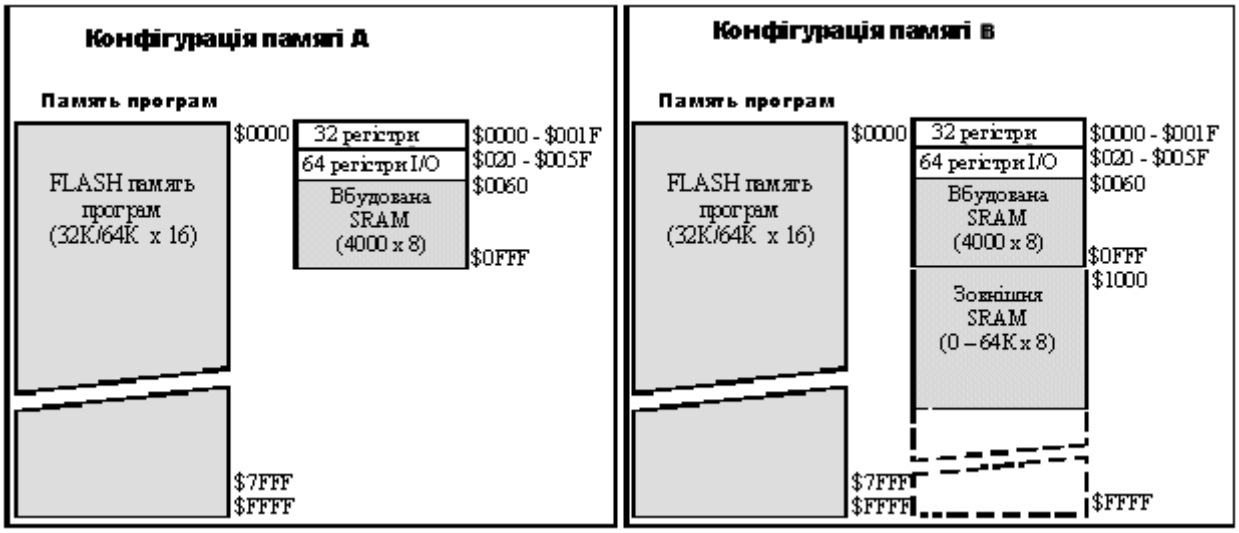


Рисунок 3.7 – Конфігурація пам'яті

Мікроконтролери конфігурації B дозволяють використовувати додаткову зовнішню пам'ять даних. Зовнішня пам'ять буде адресуватися залишковим до 64K простором адрес, тобто він буде починатися слідом за простором адрес вбудованої SRAM. При використанні зовнішньої SRAM ємністю 64K будуть загублені 4K зовнішньої пам'яті, оскільки адреси цього обсягу будуть зайняті вбудованою пам'яттю. При звертанні по адресах пам'яті даних за межами вбудованої SRAM використовуються ті ж команди, що й для звертання до вбудованої SRAM. При звертанні до вбудованої пам'яті дані виводи стробів керування зовнішньою пам'яттю даних (RD і WR) залишаються неактивними підчас усього циклу звертання.

Робота зовнішньої SRAM дозволяється установкою біта SRE у регістрі MCUCR. У порівнянні зі звертанням до вбудованої пам'яті даних, звертання до зовнішньої пам'яті даних вимагає додаткового циклу на кожен байт. Це означає, що для виконання команд LD, ST, IDS, STS, PUSH і POP потрібен додатковий тактовий цикл. Якщо стек розміщений у зовнішньої SRAM, то переривання, виклик підпрограм і повернення зажадають два додаткових цикли, оскільки в стеці буде опускатися і підніматися вміст двобайтового лічильника команд. Якщо інтерфейс із зовнішньої SRAM використовується зі станом очікування, то на кожен байт необхідно ще два додаткових тактових цикли. Це приводить до наступного ефекту. Командам пересилання даних необхідно два додаткових тактових цикли, тоді як при обробці переривання, виклику підпрограм і при поверненні з підпрограм буде потрібно на чотири тактових цикли більше, ніж це зазначено в описі системи команд.

При адресації пам'яті даних використовуються п'ять режимів адресації: безпосередня адресація, непряма зі зсувом, непряма, непряма з переддекрементом і непряма з постдекрементом. Регістри з R26 по R31 реєстрового файлу працюють як X, Y та Z регістри покажчики непрямої адресації.

Непрямій адресації зі зсувом доступні 63 адреси відносно базових адрес, що знаходяться в регістрах Y чи Z. При використанні непрямої адресації з автоматичним переддекрементом і постдекрементом автоматично декрементуються та інкрементуються адреси, записані в регістри X, Y та Z. Усіма цими режимами перекривається весь адресний простір даних, включаючи 32 регістра загального призначення і 64 регістра I/O. Докладний опис усіх режимів адресації приведено в наступному розділі.

3.5.6 Режими адресації пам'яті програм і даних

При звертанні до Flash пам'яті програм і пам'яті даних (SRAM, регістровому файлу і пам'яті I/O) AVR Enhanced RISC мікроконтролерами ATmega603/103 використовуються потужні й ефективні режими адресації. У даному розділі описуються режими адресації,

підтримувані AVR архітектурою. На малюнках OP означає частину слова команди, що відповідає операційному коду.

Безпосередня адресація, одиночний регістр Rd

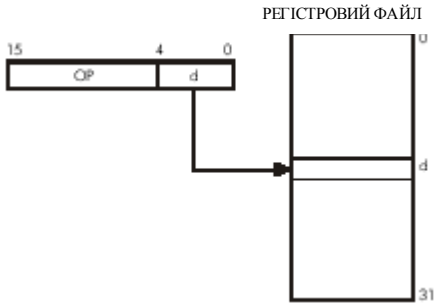


Рисунок 3.8 – Безпосередня адресація одного регістра

Операнд міститься у регістрі d (Rd).

Безпосередня адресація, два регістри Rd та Rr

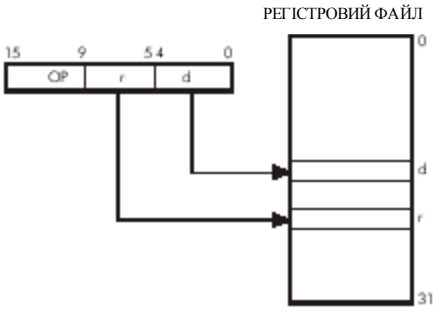


Рисунок 3.9 – Безпосередня регістрова адресація двох регістрів

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

Безпосередня адресація I/O

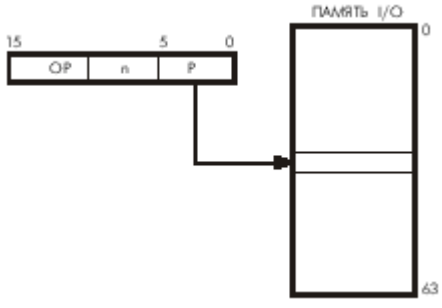


Рисунок 3.10 – Безпосередня адресація I/O

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

Безпосередня адресація даних

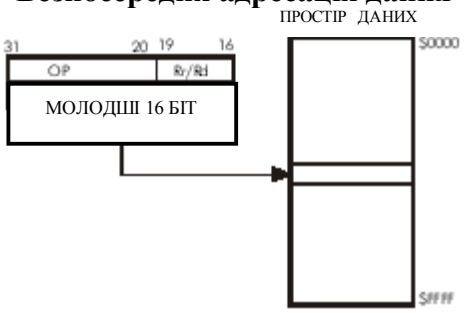


Рисунок 3.11 - Безпосередня адресація даних

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

Непряма адресація даних із зміщенням

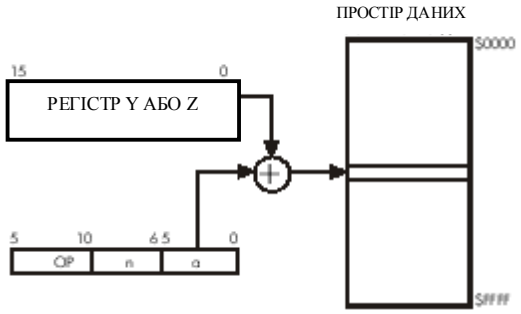


Рисунок 3.12 - Непряма адресація даних із зміщенням

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

Непряма адресація даних

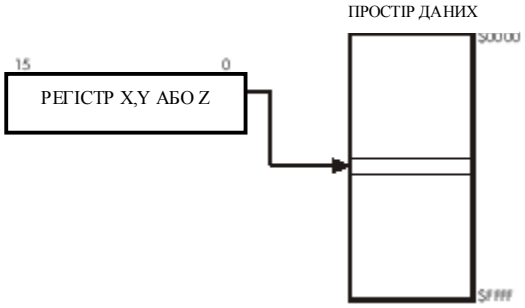


Рисунок 3.13 - Непряма адресація даних

Адреса операнда міститься в регістрі X, Y чи Z.

Непряма адресація даних із переддекрементом

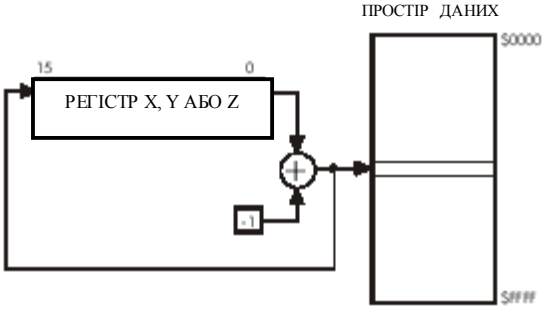


Рисунок 3.14 – Непряма адресація даних з переддекрементом

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

Непряма адресація даних з постінкрементом

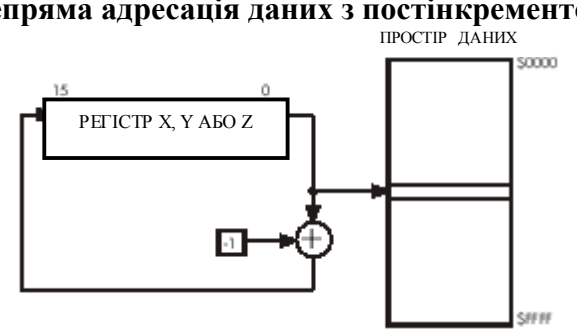


Рисунок 3.15 – Непряма адресація даних з постінкрементом

Після виконання операції регістр X, У чи Z інкрементується. Адресою операнда є вміст X, У чи Z регістра попередній інкрементуванню.

Адресація констант з використанням команд LPM і ELPM

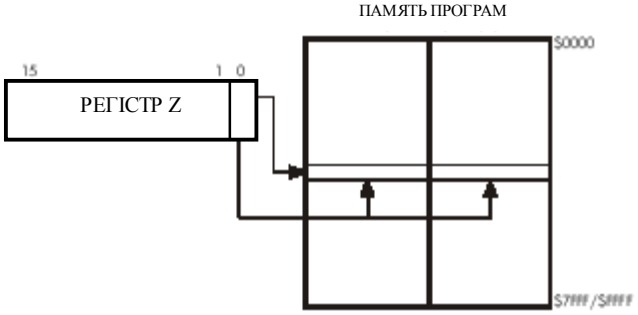


Рисунок 3.16 – Адресація константи в пам'яті програм

Адреса байта константи визначається вмістом регістра Z. Старші 15 бітів визначають слово адреси (від 0 до 32К). Стан молодшого біта визначає вибір молодшого байта (LSB = 0) чи старшого байта (LSB = 1]. При використанні команди ELPM молодший біт (RAM Page) регістра Z - RAMPZ використовується для вибору сторінки пам'яті (RAMPZO = 0: молодша сторінка, RAMPZO = 1: старша сторінка). Команда ELPM не використовується мікроконтролером ATmega603.

Безпосередня адресація пам'яті програм, команди JMP TA CALL

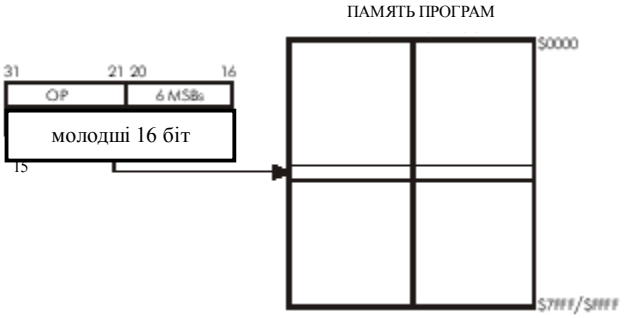


Рисунок 3.17 – Безпосередня адресація пам'яті програм

Виконання програми продовжується з адреси, записаної безпосередньо в адресі команди.

Непряма адресація пам’яеі програм,команди IJMP TA ICALL

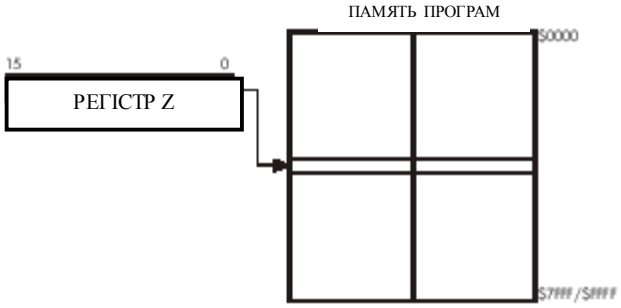


Рисунок 3.18 – Непряма адресація пам’яті програм

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

Відносна адресація пам’яті програм, команди RJMP TA RCALL

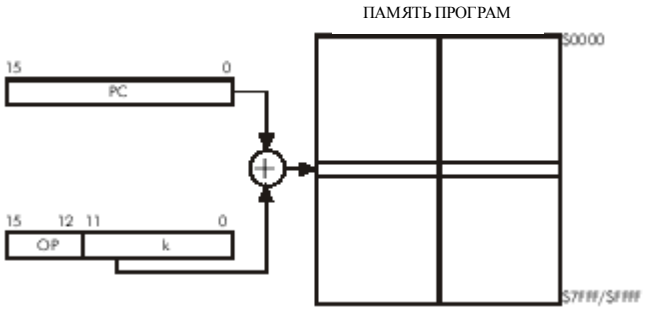


Рисунок 3.19 – Відносна адресація пам’яті команд

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

3.5.7 EEPROM пам’ять даних

EEPROM пам'ять даних організована як окремий простір даних з можливістю зчитування і запису окремого байта. EEPROM забезпечує 100000 циклів стирання/запису. Взаємодія між EEPROM і CPU визначається регістром адреси EEPROM, регістром даних EEPROM і регістром керування EEPROM.

3.5.8 Час звертання до пам’яті і тактування виконання команд

AVR CPU тактується системним тактовим сигналом System Clock Ø, формованим за допомогою зовнішнього кварцового кристала. Внутрішнє ділення не використовується.

На рис.3.20 представлений процес паралельних вибірки і виконання команд, забезпечувані Гарвардською архітектурою, і концепція регістрового файлу швидкого доступу. Це базовий принцип конвеєрної обробки, що забезпечує питому продуктивність 1 MIPS/МГц при відповідних результатах вартості функції, кількості функцій на один такт і кількості функцій на одиницю споживаної потужності.

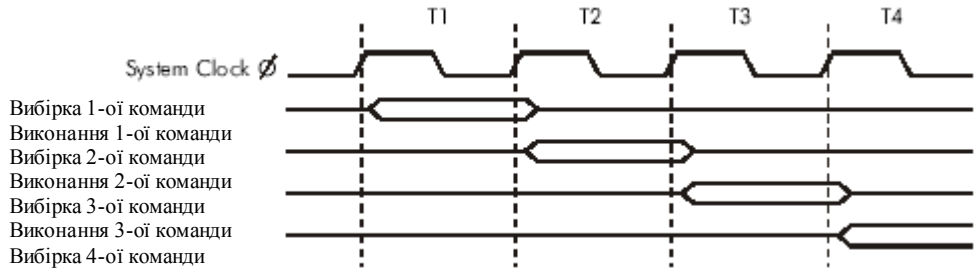


Рисунок 3.20 – Паралельні вибірка та виконання команд

На рис. 3.21 представлений принцип внутрішнього тактування регістрового файлу. Протягом одного тактового циклу виконання операції ALU використовує два операнда регістрів і результат повертає в регістр призначення.

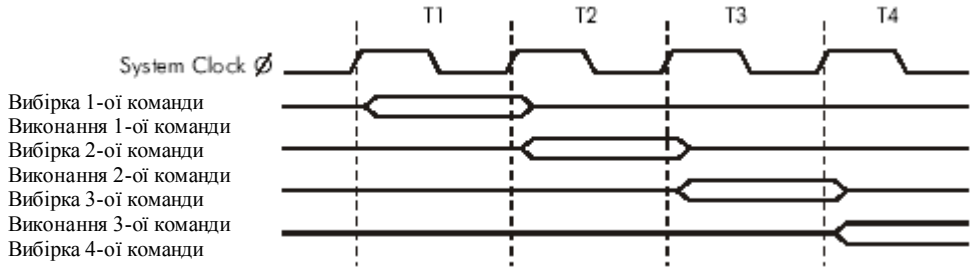


Рисунок 3.21 – Одноциклова робота ALU

На рис. 3.22 показано звертання до вбудованої SRAM даних за два тактових цикли.

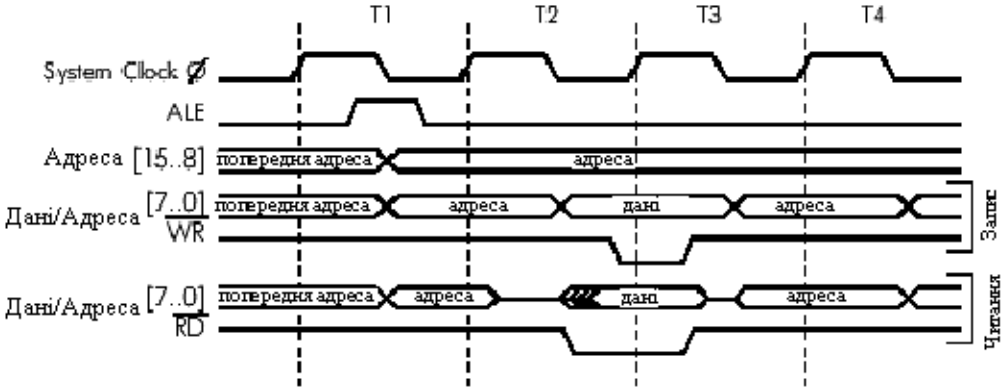


Рисунок 3.22 – Цикли звертання до зовнішньої SRAM даних без стану очікування(Wait State)

На рис. 3.23 показано звернення до зовнішньої SRAM даних при встановленому біті стану очікування (Wait State active).

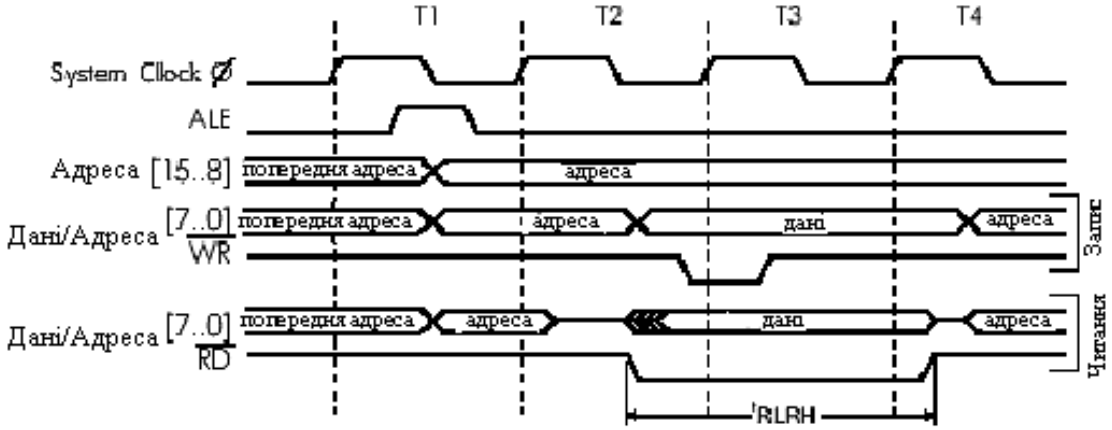


Рисунок 3.23 - Цикли звертання до зовнішньої SRAM даних із станом очікування

3.5.9 Пам’ять вводу/виводу (I/O)

Опис простору I/O мікроконтролерів ATmega603/103 представлений в Табл. 3.1

Таблиця 3.1- *Простір I/O мікроконтролерів ATmega603/103*

Адреса I/O (адреса SRAM)	Позначення	Функція
$3F($5F)	SREG	Регістр статусу (Status REGister)
$3E ($5E)	SPH	Верхній байт покажчика стека (Stack Pointer High)
$3D ($5D)	SPL	Нижній байт покажчика стека (Slack Pointer Low)
$3C ($5C)	XDIV	Регістр керування розподілом тактової частоти (XTAL Divide Control Register)
$3B ($5B)	RAMPZ	Регістр вибору сторінки Z RAM (RAM Page Z Select Register)
$3A ($5A)	EICR	Регістр керування зовнішніми перериваннями (External Interrupt Control Register)
$39 ($59)	EIMSK	Регістр масок зовнішніх переривань (External Interrupt MaSK register)
$38 ($58)	EIFR	Регістр флагів зовнішніх переривань (External Interrupt Flag Register)
$37 ($57)	TIMSK	Регістр масок переривань по таймерам/лічильникам (Timer/Interrupt MaSK register)
$36 ($56)	TIFR	Регістр флагів переривань по таймерам/лічильникам (Timer/Counter Interrupt Flag register)
$35 ($55)	MCUCR	Регістр керування MCU (MCU General Control Register)
$34 ($54)	MCUSR	Регістр статусу MCU (MCU Status Register)
$33 ($53)	TCCR0	Регістр керування таймером/лічильником0 (Timer/Counter0 Control Register)
$32 ($55)	TCNT0	Таймер/лічильник0 (Timer/Counter0 (8-bit))
$31 ($51)	OCR0	Регістр порівняння виходу таймера/лічильника0 (Timer/Counter0 Output Compare Register)
$30 ($50)	ASSR	Регістр статусу асинхронного режиму (Asynchronous Mode Status Register)
$2F ($4F)	TCCR1A	Керуючий регістр А таймера/лічильника1 (Timer/Counterl Control Register A)
$2E ($4E)	TCCR1B	Керуючий регістр В таймера/лічильника1 (Timer/Counter1 Control Register A)
$2D ($4D)	TCNT1H	Старший байт таймера/лічильника1 (Timer/Counterl High Byte)
$2C ($4C)	TCNT1L	Молодший байт таймера/лічильника1 (Timer/Counter1 Low Byte)
$2B ($4B)	OCR1AH	Старший байт регістра А порівняння виходу таймера/лічильника 1 (Timer/Counter1 Output Compare Register A High Byte)
$2A ($4A)	OCR1AL	Молодший байт регістра А порівняння виходу таймера/лічильника1 (Timer/Counter1 Output Compare Register A Low Byte)
$29 ($49)	OCR1BH	Старший байт регістра В порівняння виходу таймера/лічильника1 (Timer/Counterl Output Compare Register B High Byte)

Продовження таблиці 3.1

$28($48)	OCR1BL	Молодший байт регістра B порівняння виходу таймера/лічильника1 (Timer/Counter1 Output Compare Register B Low Byte)
$27 ($47)	ICR1H	Старший байт регістра захоплення таймера/лічильника1 (Timer/Counterl Input Capture Register High Byte)
$56 ($46)	ICR1L	Молодший байт регістра захоплення таймера/лічильника1 (Timer/Counterl Input Capture Register Low Byte)
$25 ($45)	TCCR2	Регістр керування таймером/лічильником2 (Timer/Counter2 Control Register)
$24 ($44)	TCNT2	Таймер/лічильник 2 (Timer/Counter2 (8-bit))
$23 ($43)	OCR2	Регістр порівняння виходу таймера /лічильника2 (Timer/Counter2 Output Compare Register)
$21 ($41)	WDTCR	Регістр керування сторожовим таймером (Watchdog Timer Control Register)
$1F ($3F)	EEARH	Старший байт регістра адреси EEPROM (EEPROM Address Register High)
$1E ($3E)	EEARL	Молодший байт регістра адреси EEPROM (EERPOM Address Register Low)
$1D ($3D)	EEDR	Регістр даних EEPROM (EEPROM Data Register)
$1B ($3B)	PORTA	Регістр даних порту A (Data Register, Port A)
$1A($3A)	DDRA	Регістр напрямку даних порту A (Data Direction Register, Port A)
$19 ($39)	PINA	Виводи входів порту A (Input Pins, Port A)
$18 ($38)	PORTB	Регістр даних порту B (Data Register, Port B)
$17 ($37)	DDRB	Регістр напрямку даних порту B (Data Direction Register, Port B)
$16 ($36)	PINB	Виводи входів порту B (Input Pins, Port B)
$15 ($35)	PORTC	Регістр даних порту C (Data Register, Port C)
$15 ($35)	PORTD	Регістр даних порту D (Data Register, Port D)
$11 ($31)	DDRD	Регістр напрямку даних порту D (Data Direction Register, PortD)
$10 ($30)	FIND	Виводи входів порту D (Input Pins, Port D)
$OF ($2F)	SPDR	Регістр даних SPI I/O (SPI I/O Data Register)
$OE ($2E)	SPSR	Регістр статусу SPI (SPI Status Register)
$OD ($2D)	SPCR	Регістр керування SPI (SPI Control Register)
$OB ($2B)	USR	Регістр статусу DART (UART Status Register)
$OA($2A)	UCR	Регістр керування UART (UART Control Register)
$09 ($29)	UBRR	Регістр керування швидкістю UART (UART Baud Rate Register)
$08 ($28)	ACSR	Регістр статусу та керування аналогового компаратора (Analog Comparator Control and Status Register)
$07 ($27)	ADMUX	Регістр вибору мультиплексора ADC (ADC Multiplexer Select Register)
$06 ($26)	ADCSR	Регістр статусу і керування ADC (ADC Control and Status Register)
$05 ($25)	ADCH	Старший байт регістра даних ADC (ADC Data Register High)
$04 ($24)	ADCL	Молодший байт регістра даних ADC (ADC Data Register Low)

Продовження таблиці 3.1

$03 ($23)	PORTE	Регістр даних порту E (Data Register, Port E)
$02 ($22)	DDRE	Регістр напрямку даних порту E (Data Direction Register, Port E)
$01 ($21)	PINE	Виводи входів порту E (Input Pins, Port E)
$00 ($20)	PINF	Виводи входів порту F (Input Pins, Port F)

Примітки: зарезервовані і не використовувані комірки в таблиці не показані. Усі засоби I/O і периферії мікроконтролерів ATmega603/103 розміщені в просторі I/O. При використанні IN і OUT використовуються адреси регістрів I/O з $00 по $3F. Оскільки регістри I/O представлені в адресному просторі SRAM, то до них можна адресуватися як до звичайних осередків SRAM з адресами з $20 по $5F. Адреса SRAM отримується простим додаванням $20 до безпосередньої адреси I/O. Адреса SRAM, по всьому документі, приведена у круглих дужках після безпосередньої адреси I/O. Регістри I/O, у межах адрес від $00 ($20) до $1F ($3F), по-бітово адресуються командами SBI і CBI. Стан кожного окремого біта цих регістрів може бути перевірений командами SBIS і SBIC.

3.5.10 Регістр статусу - SREG

Регістр статусу SREG - розміщений у просторі I/O за адресою $3F ($5F) і його біти визначаються як:

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

• Bit 7 - 1: Global Interrupt Enable – Дозвіл глобального переривання

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

• Bit 6 - T: Bit Copy Storage - Біт збереження копії

Команди копіювання біта BLD (Bit Loa) і BST (Bil STore) використовують біт T як біт джерело і біт призначення при операціям з бітами. Командою BST біт регістра регістрового файлу копіюється в біт T, командою BLD біт T копіюється в регістр регістрового файлу.

• Bit 5 - H: Half Carry Flag – Флаг напівпереносу

Флаг напівпереносу вказує на напівперенос у ряді арифметичних операцій. Більш докладна інформація приведена в описі системи команд.

• Bit 4 - S: Sign Bit, S = N O V - Біт знаку

Біт S завжди знаходиться в стані, обумовленому логічним що виключає ЧИ (exclusive OR) між флагом негативного значення N і доповненням до двох флага переповнення V. Більш докладна інформація приведена в описі системи команд.

• Bit 3 - V: Two's Complement Overflow Flag - Доповнення до двох флага переповнення

Доповнення до двох флага V підтримує арифметику доповнення до двох. Більш докладна інформація приведена в описі системи команд.

• Bit 2 - N: Negative Flag – Флаг негативного значення

Флаг негативного значення N вказує на негативний результат ряду арифметичних та логічних операцій. Більш докладна інформація приведена в описі системи команд.

• Bit 1 - Z: Zero Flag - Флаг нульового значення

Флаг нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій. Більш докладна інформація приведена в описі системи команд.

• **Bit 0 - З: Carry Flag - Флаг переносу**

Флаг переносу С вказує на перенос в арифметичних і логічних операціях. Більш докладна інформація приведена в описі системи команд.

3.5.11 ПОКАЖЧИК НА СТЕК (Stack Pointer) - SP

Мікроконтролери AVR мають 16-розрядний покажчик на стек, розміщений у двох регістрах простору I/O по адресах $3E ($5E) і $3D ($5D). Оскільки мікроконтролери ATmega603/103 підтримують об'єм SRAM до 64 Кбайт, то використовуються всі 16 розрядів покажчика стека.

Покажчик на стек вказує на область у SRAM даних, у якій розміщаються стеки підпрограм і переривань. Об'єм стека в SRAM даних повинний задаватися програмою перед кожним викликом підпрограми й обробкою дозволеного переривання. Покажчик стеку декрементується на одиницю, при кожнім занесенні командою PUSH даних у стек, і на дві одиниці при занесенні даних у стек підпрограмою CALL і перериванням.

Покажчик стеку інкрементується на одиницю, при витягуванні даних зі стеку командою POP, і на дві одиниці при витягуванні даних зі стеку при поверненні з підпрограми (RET) чи поверненні з переривання (IRET).

Біти	15	14	13	12	11	10	9	8	
$3E ($5E)	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Читання/Запис	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	
Початковий стан	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

3.5.12 РЕГІСТР ВИБОРУ СТОРІНКИ Z RAM – RAMPZ

Біти	7	6	5	4	3	2	1	0	
$3FB{$5B}	-	-	-	-	-	-	-	RAMPZ0	RAMPZ
Читання/Запис	R	R	R	R	R	R	R	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Регістр RAMPZ використовується звичайно для визначення до якої сторінки RAM, ємністю 64К, можливе звертання за допомогою покажчика Z. Оскільки мікроконтролери ATmega603/103 не підтримують SRAM з обсягом понад 64К, цей регістр використовується тільки для вибору сторінки в пам'яті програм при використанні команди ELPM. Різні установки біта RAMPZO призводять до наступного ефекту:

RAMPZO = 0: Команді ELPM доступна пам'ять програм з адресами від $0000 до S7FFF (молодші 64 Кбайт).

RAMPZO = 1: Команді ELPM доступна пам'ять програм з адресами від $8000 до SFFFF (старші 64 Кбайт).

Відзначимо, що на LPM не впливає установка RAMPZ.

Мікроконтролер ATmega603 не містить регістра RAMPZ і не має команд ELPM. Команда LPM здатна перекрити весь простір пам'яті програм мікроконтролера ATmega603.

3.5.13 РЕГІСТР КЕРУВАННЯ MCU - MCU Control Register - MCUCR

Біти регістра керування MCU керують виконанням основних функцій MCU.

Біти	7	6	5	4	3	2	1	0	
$3B {$5B}	SRE	SRW	SE	SM1	SM0	-	-	-	MCUCR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R	R	R	
Початковий стан	0	0	0	0	0	0	0	0	

• ВВ 7 - SRE: External SRAM Enable - Дозвіл зовнішньої SRAM

Встановлений у 1 біт SRE дозволяє звертання до зовнішньої SRAM даних і переводить роботу виводів ADO-7 (Порт А), А8-15 (Порт С), WR і RD на виконання альтернативної функції. Потім біт SRE перенастроює установки напрямків будь-яких виводів у відповідних регістрах напрямку даних. Очищення біта SRE (установка в 0) забороняє звертання до зовнішньої SRAM і відновлює нормальні установки напрямків виводів і даних.

• Bit 6 - SRW: External SRAM Walt State - Режим чекання зовнішньої SRAM

При встановленому в 1 біті SRW до циклу звертання до зовнішньої SRAM додається один цикл чекання. При скинутому в 0 біті SRW звертання до зовнішнього SRAM виконується по трьоцикловій схемі (див. рис. 3.22 і рис. 3.23)

• Bit 5 - SE: Sleep Enable - Дозвіл режиму Sleep

Встановлений у 1 біт SE дозволяє переведення MCU у режим sleep по команді SLEEP. Щоб виключити переведення MCU у незапрограмований режим sleep, рекомендується встановлювати біт SE безпосередньо перед виконанням команди SLEEP.

• Bits 4,3 - SM1/SM0: Sleep Mode Select bits 1 and 0 - Біти вибору режиму Sleep

Дані біти дозволяють вибрати один із трьох можливих режимів sleep, як показано в таблиці 3.2.

Таблиця 3.2– Вибір режиму Sleep

SM1	SM0	Sleep Mode
0	0	Режим Idle
0	1	Зарезервовано
1	0	Режим Power Down
1	1	Режим Power Save

• Bits 2..0 - Res: Reserved bits - Зарезервовані біти

Ці біти зарезервовані і при зчитуванні завжди будуть показувати стан 0

3.5.14 РЕГІСТР КЕРУВАННЯ КОЕФІЦІЄНТОМ ПОДІЛУ ЧАСТОТИ КВАРЦОВОГО ГЕНЕРАТОРА XDIV

Регістр XDIV використовується для встановлення коефіцієнта поділу частоти кварцового генератора в діапазоні від 1 до 129.

Біти	7	6	5	4	3	2	1	0	
$3C {$5C}	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0	XDIV
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

• Bit 7 - XDIVEN: XTAL Divide Enable - Дозвіл поділу частоти XTAL

При встановленому в 1 біті XDIVEN тактова частота CPU і всієї периферії поділяється відповідно до встановлених бітів XDIV6 - XDIVO коефіцієнтом поділу. Така можливість може бути використана для зниження споживання, за умови допустимості зниження обчислювальної потужності. Цей біт може бути встановлений і очищений у процесі виконання програми тоді, коли це визначається застосуванням.

• Bits 6..0 - XPIV6..XDIVO: XTAL Divide Select Bits 6 – 0 - Біти вибору коефіцієнта поділу

Ці біти встановлюють коефіцієнт поділу тактової частоти при встановленому біті XDIVEN. Якщо десяткове значення цих семи бітів позначити через d, то результуюча тактова частота CPU буде визначатися по формулі

$$f\ clk = \frac{XTAL}{(129-d)}$$

Стан цих бітів можна змінити тільки коли біт XDIVEN скинутий (у стані 0). При встановленому біті XDIVEN, записане одночасно в біти XDIV6..XDIV0 значення буде визначати коефіцієнт поділу. При скиданні біта XDIVEN записані в біти XDIV6..XDIV0 значення ігноруються. Оскільки дільник поділяє тактову частоту, що надходить на MCU, то і на периферійні пристрої надходить тактова частота з тим же коефіцієнтом поділу.

3.5.15 Обробка переривань і скидання

Мікроконтролери ATmega603/103 використовують 23 джерела переривання. Ці переривання і вектор скидання розташовують окремими програмними векторами в просторі пам'яті програм. Кожному перериванню привласнений свій біт дозволу, який повинний бути встановлений разом з бітом I регістра статусу. Молодші адреси простору пам'яті програм автоматично визначаються як вектори скидання і переривань.

Повний перелік векторів представлений у таблиці 3.3. Перелік представляє також рівень пріоритету для кожного переривання. Переривання з молодшими адресами мають більший рівень пріоритету. RESET має найвищий рівень пріоритету, наступним є INTO - Запит зовнішнього переривання 0 і т.д.

Таблиця 3.3- Вектори скидання і переривань

Номер вектора	Адреса	Джерело	Опис переривання
1	$0000	RESET	Скидання по виводу і сторожовому таймері (Hardware Pin and Watchdog Reset)
2	$0002	INTO	Запит зовнішнього переривання 0 (External Interrupt Request 0)
3	$0004	INT1	Запит зовнішнього переривання 1 (External Interrupt Request 1)
4	$0006	INT2	Запит зовнішнього переривання 2 (External Interrupt Request 2)
5	$0008	INT3	Запит зовнішнього переривання 3 (External Interrupt Request 3)
6	$000A	INT4	Запит зовнішнього переривання 4 (External Interrupt Request 4)
7	$000C	INT5	Запит зовнішнього переривання 5 (External Interrupt Request 5)
8	$000E	INT6	Запит зовнішнього переривання 6 (External Interrupt Request 6)
9	$0010	INT7	Запит зовнішнього переривання 7 (External Interrupt Request 7)
10	$0012	TIMER2 COMP	Збіг при порівнянні таймера/лічильника2 (Timer/Counter2 Compare Match)
11	$0014	TIMER2 OVF	Переповнення таймера/лічильника2 (Timer/Counter2 Overflow)
12	$0016	TIMER 1 CAPT	Захоплення таймера/лічильника1 (Timer/Counterl Capture Event)
13	$0018	TIMER1 COMPA	Збіг А при порівнянні таймера/лічильника1 (Timer/Counterl Compare Match A)

Продовження таблиці 3.3

14	$001A	TIMER1	Збіг В при порівнянні таймера/лічильника1 (Timer/Counterl

		COMPB	Compare Match B)
15	$001C	TIMER1 OVF	Переповнення таймера/лічильника1 (Timer/Counterl Overflow)
16	$001E	TIMER0 COMP	Збіг при порівнянні таймера/лічильника0 (Timer/Counter0 Compare Match)
17	$0020	TIMER0 OVF	Переповнення таймера/лічильника0 (Timer/Counter0 Overflow)
18	$0022	SPI, STC	Завершення пересилання SPI (SPI Serial Transfer Complete)
19	$0024	UART, RX	Завершення прийому UART (UART, Rx Complete)
20	$0026	UART, UDRE	Регістр даних UART порожній (UART Data Register Empty)
21	$0028	UART, TX	Завершення передачі UART (UART, Tx Complete)
22	$002A	ADC	Завершення ADC перетворення (ADC Conversion Complete)
23	$002C	EE READY	Готовність EEPROM (EEPROM Ready)
24	$002E	ANALOG COMP	Спрацьовування аналогового компаратора (Analog Comparator)

Найбільш часто використовувані установки адрес векторів скидання та переривань:

```
$0000 jmp RESET ; Обробка перезапуска процесора (Reset Handler)
$0002 jmp EXT_INT0 ; Обробка зовнішнього переривання  IRQ0 (IRQ0Handler)
$0004 jmp EXT_INT1 ; Обробка зовнішнього переривання  IRQ1 (IRQ1 Handler)
$0006 jmp EXT_INT2 ; Обробка зовнішнього переривання  IRQ2 (IRQ2 Handler)
$0008 jmp EXT_INT3 ; Обробка зовнішнього переривання  IRQ3 (IRQ3 Handler)
$000A jmp EXT_INT4 ; Обробка зовнішнього переривання  IRQ4 (IRQ4 Handler)
$000C jmp EXT_INT5 ; Обробка зовнішнього переривання  IRQ5 (IRQ5 Handler)
$000E jmp EXT_INT6 ; Обробка зовнішнього переривання  IRQ6 (IRQ6 Handler)
$0010 jmp EXT_INT7 ;  Обробка зовнішнього переривання ¡ IRQ7 (IRQ7 Handler)
$0012 jmp TIM2_COMP ; Обробка порівняння Таймера 2 (Timer2 Compare Handler)
$0014 jmp TIM2_OVF ; Обробка переповнення Таймера 2 (Timer2 Overflow Handler)
$0016 jmp TIM1_CAPT ; Обробка захоплення Таймера 1 (Timer1 Capture Handler)
$0018 jmp TIM1_COMPA ; Обробка порівняння  A Таймера  1 (Timer1 CompareA Handler)
$001A jmp TIM1_COMPB ; Обробка порівняння  B Таймера  1 (Timer1 CompareB Handler)
$001C jmp TIM1_OVF ; Обробка переповнення Таймера  1 (Timer1 Overflow Handler)
$001E jmp TIM0_COMP ; Обробка порівняння Таймера 0 (Timer0 Compare Handler)
$0020 jmp TIM0_OVF ; Обробка переповнення Таймера 0 (Timer0 Overflow Handler)
$0022 jmp SPI_STC ; Обробка завершення пересилки  SPI (SPI Transfer Complete Handler)
$0024 jmp UART_RXC ; Обробка завершення прийому  UART (UART RX Complete Handler)
$0026 jmp UART_DRE ; Обробка порожнього регістра даних (UDR Empty Handler)
$0028 jmp UART_TXC ; Обробка завершення пересилки  UART (UART TX Complete Handler)
$002A jmp ADC ; Обробка завершення перетворення ADC (ADC Conversion Complete Handler)
$002C jmp EE_RDY ; Обробка готовності EEPROM (EEPROM Ready Handler)
$002E jmp ANA_COMP ; Обробка спрацьовування аналогового компаратора (Analog
Comparator Handler)
;
$0030 MAIN: <instr> xxx ; Початок основної програми (Main program start)
... ... ... ...
```

3.5.16 ДЖЕРЕЛА СКИДАННЯ

Мікроконтролери ATmega603/103 розпоряджаються трьома джерелами сигналу скидання:

• Скидання по включенню живлення (Power-On Reset). MCU скидається при підключенні живлення до виводів VCC і GND.

• Зовнішнє скидання (External Reset). MCU скидається, якщо низький рівень присутній на вході більш двох циклів XTAL.

• Скидання по сторожовому таймері (Watchdog Reset). MCU скидається, якщо минає період сторожового таймера і сторожовий таймер дозволений.

Протягом скидання всі регістри I/O, за винятком регістра статусу MCU, встановлюються в їхні початкові стани і програма починає роботу з адреси $0000. По цій адресі повинна знаходитися команда JMP - команда абсолютного переходу до підпрограми обробки скидання. Якщо програма ніколи не дозволяє переривання, то вектори переривань не використовуються і по цих адресах можуть розташовуватися коди програми.

Таблиця 3.4- Тимчасові й електричні параметри схеми скидання (V_{cc} = 5 В)

Позн.	Параметр	Умови	Мін	Тип	Макс	Од. виміру
V_{POT}	Гранична напруга скидання по включенню живлення		1,8	2	2,2	V
V_{RST}	Гранична напруга скидання по виводу RESET			V_{cc} /2		V
V_{BO}	Скидання по зниженню напруги живлення			2.5		V
T_{TOUT}	Період затримки сигналу скидання	SUT1/0 = 00 SUT1/0 = 01 SUT1/0 = 10 SUT1 /0=11	 0.4 3.2 128	5 0.5 4.0 160	 0.6 48 192	Циклів CPU мс мс мс

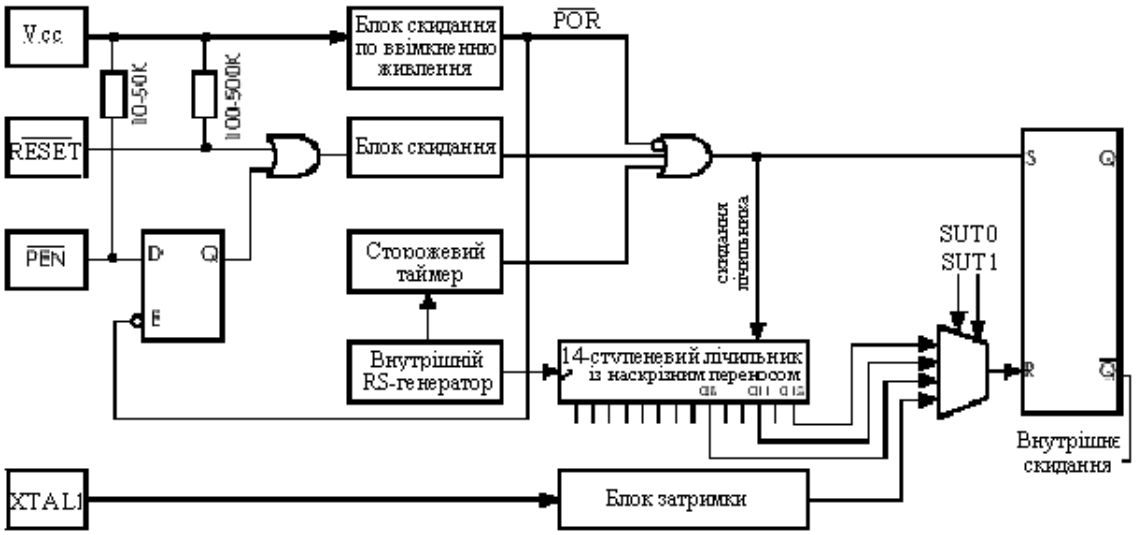


Рисунок 3.24 – Логіка скидання

Перезапуск після включення живлення

Схема скидання по включенню живлення (Power-On Reset - POR) забезпечує запуск мікроконтролера тільки по досягненні напругою Vcc безпечного рівня. Як показано на

рис.3.25, вбудований таймер, тактований вбудованим генератором сторожового таймера, утримує запуск MCU на якийсь час після досягнення граничної напруги включення живлення Vpot, що не залежить від швидкості наростання напруги Vcc (див. рис. 3.26).

У таблиці 3.4 показані установки бітів SUT1 і SUTO, що використовуються для установки тривалості періоду затримки процедури запуску. Користувачу надається можливість вибору затримки часу запуску. Установка SUT 1/0 = 00, при якій MCU запускається через 5 тактових циклів, використовується при використанні зовнішнього тактового сигналу, подаваного на вивід XTAL1. Така установка забезпечує швидкий запуск із режимів power down чи power save, за умови наявності тактового сигналу в цих режимах. Подробиці в розділі **Програмування**.

Якщо вбудована затримка запуску достатня, то RESET може бути приєднаний до Vcc чи безпосередньо через зовнішній навантажувальний резистор. Утриманням виводу на низькому рівні, під час подачі напруги, період скидання по включенню живлення може бути збільшений. Приклад такого тактування приведений на. рис. 3.27.

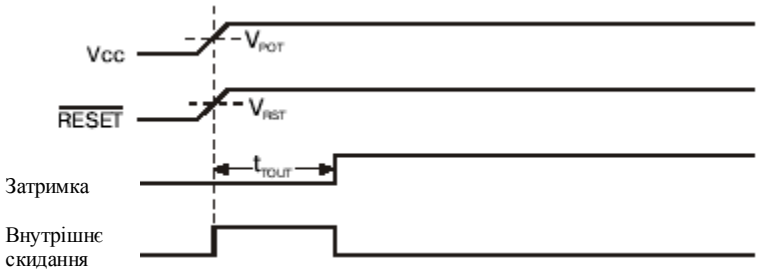


Рисунок 3.25 – Початковий запуск MCU. Вивід RESET підключений до Vcc, швидке наростання Vcc

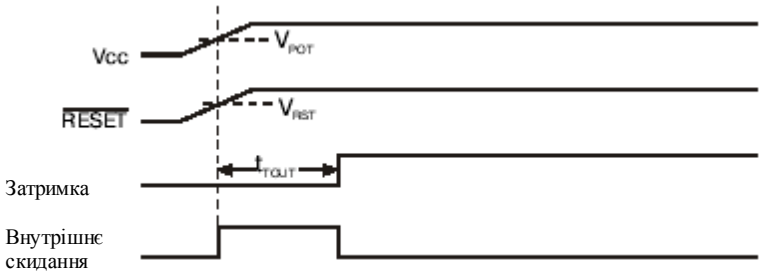


Рисунок 3.26 – Початковий запуск MCU. Вивід RESET підключений до Vcc, повільне наростання Vcc

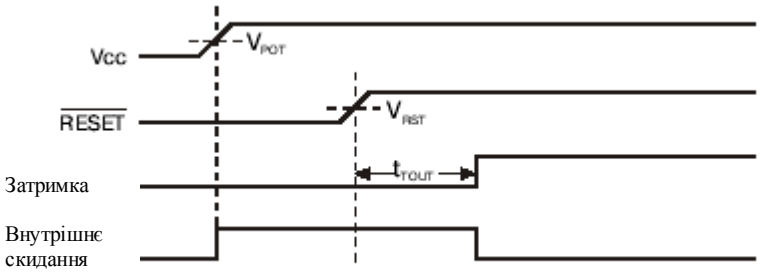


Рисунок 3.27 – Початковий запуск MCU. Зовнішнє керування станом виводу RESET

Зовнішнє керування скиданням

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

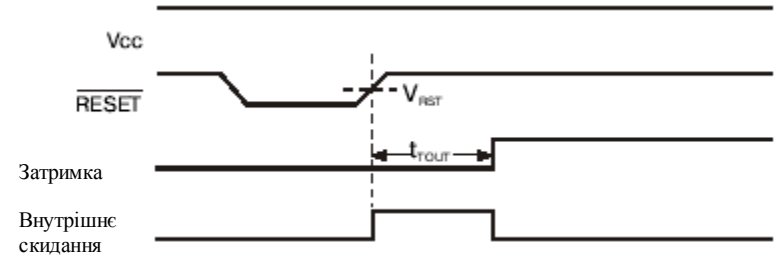


Рисунок 3.28 – Зовнішнє скидання під час роботи мікроконтролера

Скидання за сторожовим таймером

По закінченні часу, обумовленого сторожовим таймером, таймер формує короткий, тривалістю в один цикл XTAL, імпульс скидання. По фронті цього імпульсу, що падає, таймер затримки починає відлік t TOUT У розділі **Сторожовий таймер** (Watchdog Timer) приводиться докладний опис роботи сторожового таймера.

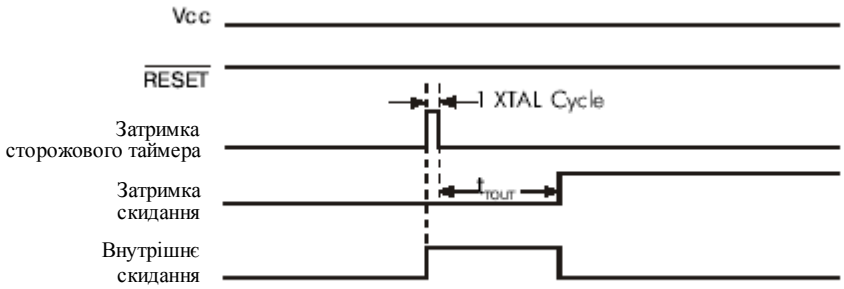


Рисунок 3.29 – Скидання сторожовим таймером під час роботи мікроконтролера

Регістр статусу mcu - mcusr – (mcu status register)

Регістр статусу MCU видає інформацію про джерело, що викликало скидання MCU.

Біт	7	6	5	4	3	2	1	0	
$34{$54}	-	-	-	-	-	-	EXTRF	PORF	MCUSR
Читання/Запис	R	R	R	R	R	R	R/W	R/W	
Початкове значення	0	0	0	0	0	0	див. опис бітів		

• Bits 7..2 - Res: Reserved Bits – Зарезервовані біти

Ці біти зарезервовані і при зчитуванні завжди покажуть стан 0.

• Bit 1 - EXTRF: External Rasat Flag – Флаг зовнішнього скидання

Після подачі напруги живлення стан цього біта невизначено (X). Біт встановлюється в стан 1 зовнішнім скиданням. Скидання по сторожовому таймері залишає цей біт незмінним.

• Bit 0 - PORF: Power On Rasat Flng - Флаг включення живлення

Даний біт установлюється скиданням по подачі живлення. Скидання по сторожовому таймері чи по зовнішньому скиданню залишають цей біт незмінним.

У таблиці показаний вплив трьох режимів скидання на стани бітів PORF і EXTRF.

Таблиця 3.5- Стан PORF і EXTRF бітів після скидання

Джерело скидання	PORF	EXTRF
Скидання по подачі живлення	1	Невизначений стан
Зовнішнє скидання	Не міняється	1
Скидання по сторожовому таймері	Не міняється	Не міняється

Для визначення режиму скидання, на підставі стану даних бітів PORF і EXTRF, користувацьке програмне забезпечення повинне скидати біти PORF і EXTRF як тільки це дозволить програма. Перш, ніж їх очистити необхідно провести перевірку цих бітів. Якщо біт був очищений до зовнішнього скидання чи скидання по сторожовому таймері, то джерело скидання може бути визначене по наступній таблиці істиності:

Таблиця 3.6- Визначення джерела скидання

PORF	*EXTRF*	*Джерело скидання*
0	0	Скидання по сторожовому таймері
0	1	Зовнішнє скидання
1	0	Скидання по подачі живлення
1	1	Скидання по подачі живлення

3.6 Обробка переривань

Мікроконтролери ATmega603/103 містять два спеціальних 8-розрядних регістри масок переривань: регістр масок зовнішніх переривань EIMSK (External Interrupt Mask) і регістр масок переривань по таймеру/лічильнику TIMSK (Timer/Counter Interrupt Mask). Крім того, у регістрах керування периферією можуть бути організовані й інші біти дозволу і біти масок.

При виникненні переривання біт I дозволу глобального переривання (Global Interrupt Enable) очищається і всі інші переривання забороняються. Користувацьке ПО, для того, щоб дозволити вкладені переривання, може установити біт I усередині підпрограми обробки переривання. Вихід з підпрограми обробки переривання відбувається по команді RETI, при цьому біт I встановлюється в стан 1. Коли лічильник команд вказує вектор підпрограми обробки переривання, відповідний флаг, що викликав переривання, апаратно очищається. Деякі флаги переривань можна очистити, записавши у відповідний біт(и) флага, що очищається, логічну одиницю.

РЕГІСТР МАСОК ЗОВНІШНІХ ПЕРЕРИВАНЬ - EIMSK - (EXTERNAL INTERRUPT MASK REGISTER)

Біт	7	6	5	4	3	2	1	0	
$39 {$59}	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Читання/Запис R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bits 7..4 - INT7 - INT4: External Interrupt Request 7-4 Enable - Дозвіл запиту зовнішніх переривань з 4 по 7

При встановлених бітах INT7- INT4 і встановленому біті I регістра статусу (SREG) дозволяються переривання по відповідним виводам входів сигналів переривань. Біт керування розпізнаванням переривання регістра керування зовнішніми перериваннями EICR (External Interrupt Control Register) визначає спрацьовування по наростаючому чи падаючому фронті чи

по логічному рівні. Активація кожного з цих виводів викликає запит переривання навіть якщо вивід буде дозволений як вихід. Це забезпечує можливість організації програмного переривання.

• Bits 3..0 - INT3 – INT0: External Interrupt Request 3-0 Enable - Дозвіл запиту зовнішніх переривань з 0 по 3

При встановлених бітах INT3- INTO і встановленому біті I регістра статусу (SREG) дозволяються переривання по відповідних входах переривань. Зовнішні переривання завжди викликають переривання низьким рівнем. Активація кожного з цих виводів викликає запит переривання навіть якщо вивід буде дозволений як вихід. Це забезпечує можливість організації програмного переривання. Запит переривання по логічному рівню, якщо він дозволений, буде генерувати запит переривання доти, поки на вході буде знаходитися низький рівень.

РЕГІСТР ФЛАГІВ ЗОВНІШНІХ ПЕРЕРИВАНЬ - EIFR (EXTERNAL INTERRUPT FLAG REGISTER)

Біт	7	6	5	4	3	2	1	0	
$38 {$58}	INTF7	INTF6	INTF5	INTF4	-	-	-	-	EIFR
Читання/Запис	R/W	R/W	R/W	R/W	R	R	R	R	
Початкове значення	0	0	0	0	0	0	0	0	

• Bits 7..4 - INTF7 - INTF4: External Interrupt 7-4 Flags - Флаги зовнішніх переривань з 4 по 7

У випадку надходження запиту на переривання на якийсь із виводів INT7 - INT4, буде встановлений у 1 відповідний флаг переривання (INTF7 - INTF4). Якщо біт I регістра SREG і відповідний біт дозволу (INT7 - INT4) у EIMSK будуть установлені, то MCLJ перейде до вектора переривання. По завершенню підпрограми переривання флаг очищається. Крім того, його можна очистити, записавши в нього логічну 1.

• Bits 3..0 - Res: Reserved Bits - Зарезервовані біти

Ці біти зарезервовані і при зчитуванні завжди покажуть стан 0.

РЕГІСТР КЕРУВАННЯ ЗОВНІШНІМИ ПЕРЕРИВАННЯМИ - EICR (EXTERNAL INTERRUPT CONTROL REGISTER)

Біт	7	6	5	4	3	2	1	0	
$3A {$5A}	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bits 7..0 - ISCX1, ISCX0: External Interrupt 7-4 Sense Control bits - Біти керування розпізнаванням зовнішніх переривань з 4 по 7

Зовнішні переривання 7-4 активуються по виводах INT7 - INT4, якщо встановлений флаг I у SREG і встановлена відповідна маска в EIMSK. Запит переривання по логічному рівню чи фронту визначається в наступній таблиці:

Таблиця 3.7- Керування розпізнаванням переривання

ISCX1	ISCX0	Опис
0	0	Запит переривання генерується низьким рівнем на INTX і я генерується низьким рівнем на INTX.
0	1	Зарезервований
1	0	Запит переривання генерується фронтом спаду на INTX ія генерується спадаючим фронтом на INTX.
1	1	Запит переривання генерується фронтом наростання на INTX

Примітка: X може бути рівним 7, 6, 5 чи 4

При зміні бітів ISC11/ISC10 переривання повинне бути заборонене шляхом очищення біта дозволу в регістрі GIMSK. В іншому випадку може відбутися переривання. Запит переривання по логічному рівні, якщо він дозволений, буде генерувати запит переривання доти, поки на вході буде знаходитися низький рівень.

РЕГІСТР МАСОК ПЕРЕРИВАННЯ ПО ТАЙМЕРАМ/ЛІЧИЛЬНИКАМ -TIMSK (TIMER/COUNTER INTERRUPT MASK REGISTER)

Біт	7	6	5	4	3	2	1	0	
$37 {$57}	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• **Bit 7 - OCIE2: Timer/Counter2 Output Compare Interrupt Enable - Дозвіл переривання по збігу таймера/лічильника2**

При встановленому біті OCIE2 і встановленому біті I регістра статусу дозволяється переривання по збігу вмісту регістра порівняння і стани таймера/лічильника2. Відповідне переривання (з вектором $0012) виконується якщо відбудеться збіг при порівнянні вмісту регістра порівняння і стани таймера/лічильника2. У регістрі флагів переривання TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг збігу таймера/лічильника2.

• Bit 6 - TOIE2: Timer/Counted Overflow Intsrrupt Enable - Дозвіл переривання по переповненню таймера/лічильника2

При встановленому біті TOIE2 і встановленому біті I регістра статусу дозволяється переривання по переповненню таймера/лічильника2. Відповідне переривання (з вектором $0014) виконується якщо відбудеться переповнення таймера/лічильника2. У регістрі флагів TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг переповнення таймера/лічильника2.

• Bit 5 – TICIE1; Timer/Counter1 Input Capture Interrupt Enable - Дозвіл переривання по захопленню таймера/лічильника1

При встановленому біті TICIE1 і встановленому біті I регістра статусу дозволяється переривання по захопленню таймера/лічильника1. Відповідне переривання (з вектором $0016) виконується якщо відбудеться запуск захоплення по виводу 29, PD4(IC1). У регістрі флагів TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг захоплення таймера/лічильника1.

• Bit 4 - OCI1 A: Timer/Counter I Output Compare Match Interrupt Enable - Дозвіл переривання по збігу регістра A з таймером/лічильником1

При встановленому біті OCIE1A і встановленому біті I регістра статусу дозволяється переривання по збігу регістра A с станом таймера/лічильника1. Відповідне переривання (з вектором $0018) виконується якщо відбудеться збіг вмісту регістра A порівняння виходу зі

станом таймера/лічильника1. У регістрі флагів TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг збігу регістра А с таймером/лічильником1.

• Bit3 - OCIE1B; Timer/Counter 1 Output Compares Match Interrupt Enable - Дозвіл переривання по збігу регістра В з таймером/лічильником1

При встановленому біті OCIE1 В и встановленому біті I регістра статусу дозволяється переривання по збігу регістра В з станом таймера/лічильника1. Відповідне переривання (з вектором $001A) виконується якщо відбудеться збіг вмісту регістра В порівняння виходу зі станом таймера/лічильника1. У регістрі флагів TIFR (Timer/Counter Interrupt Flag Register) встановлюється флаг збігу регістра В з таймером/лічильником1.

• Bit 2 - TOIE1: Timer/Counter1 Overflow Interrupt Enable - Дозвіл переривання по переповненню таймера/лічильника1

При встановленому біті OCIE1B и встановленому біті I регістра статусу дозволяється переривання по переповненню таймера/лічильника1. Відповідне переривання (з вектором $001C) виконується якщо відбудеться переповнення таймера/лічильника1. У регістрі флагів TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг переповнення таймера/лічильника1. При перебуванні таймера/лічильника1 у PWM режимі флаг переповнення лічильника встановлюється коли лічильник змінить напрямок рахунку при $0000.

• Bit 1 – OCIE0: Tlmer/Counter0 Output Compare Interrupt Enable - Дозвіл переривання по збігу таймера/лічильника0

При встановленому біті OCIE0 і встановленому біті I регістра статусу дозволяється переривання по збігу вмісту регістра порівняння і стану таймера/лічильника0. Відповідне переривання (з вектором $001E) виконується якщо відбудеться збіг при порівнянні вмісту регістра порівняння і стану таймера/лічильника0. У регістрі флагів переривання TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг збігу таймера/лічильника0.

• Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable - Дозвіл переривання по переповненню таймера/лічильника0

При встановленому біті TOIE0 і встановленому біті I регістра статусу дозволяється переривання по переповненню таймера/лічильника0. Відповідне переривання (з вектором $0020) виконується якщо відбудеться переповнення таймера/лічильника0. У регістрі флагів TIFR (Timer/Counter Interrupt Flag Register) установлюється флаг переповнення таймера/лічильника0.

РЕГІСТР ФЛАГІВ ПЕРЕРИВАНЬ ПО ТАЙМЕРАХ/ЛІЧИЛЬНИКАХ TIFR (TIMER/COUNTER INTERRUPT FLAG REGISTER)

Біт	7	6	5	4	3	2	1	0	
$36 {$56}	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bit 7 – OCF2: Output Compare Flag 2: - Флаг 2 збігу таймера/лічильника2 і даних OCR2

Біт OCF2 установлюється при збігу стану таймера/лічильника 2 і вмісту регістра OCR2 (Output Compare Register 2). Біт OCF2 апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1. При встановленому біті I у регістрі SREG, установлених OCIE2 (Timer/Counter2 Output Compare Interrupt Enable) і OCF2 виконується переривання по збігу виходу таймера/лічильника 2.

• Bit 6-TOV2: Timer/Counter2 Ovarflow Flag - Флаг переповнення таймера/лічильника2

Біт TOV2 установлюється при переповненні таймера/лічильника2. Він апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у

флаг логічної 1. При встановленому біті I у регістрі SREG, установлених OCIE2 (Timer/Counter2 Overflow Interrupt Enable) і TOV2 виконується переривання по переповненню таймера/лічильника2. У режимі PWM цей біт установлюється при зміні напрямку рахунку при $00.

• Bit 5 - ICF1: Input Capture Flag 1 - Флаг 1 захоплення входу

Біт ICF1 встановлюється у випадку захоплення входу, що показує, що стан таймера/лічильника1 переслано у вхідний регістр захоплення ICR1. Біт очищається апаратно при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1.

• Bit 4 – OCF1A: Output Compare Flag 1A - Флаг 1A збігу виходу

Біт OCF1A установлюється при збігу стану таймера/лічильника1 і вмісту регістра OCR1A (Output Compare Register 1A). Біт OCF1A апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1. При встановленому біті I у регістрі SREG, установлених OCIE1A (Timer/Counter 1 Compare Interrupt Enable) і OCF1A виконується переривання по збігу виходу таймера/лічильника1.

• Bit 3 – OCF1B: Output Compare Flag 1B - Флаг 1B збігу виходу

Біт OCF1B установлюється при збігу стану таймера/лічильника1 і вмісту регістра OCR1B (Output Compare Register IB). Біт OCF1B апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1. При встановленому біті I у регістрі SREG, установлених OCIE1 B (Timer/Counter1 Compare Interrupt Enable) і OCF1B виконується переривання по збігу виходу таймера/лічильника1.

• Bit 2 - TOV1: Timer/Counter1 Overflow Flag - Флаг переповнення таймера/лічильника1

Біт TOV1 установлюється при переповненні таймера/лічильника1. Він апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1. При встановленому біті I у регістрі SREG, установлених TOIE1 (Timer/Counterl Overflow Interrupt Enable) і TOV1 виконується переривання по переповненню таймера/лічильника1. У режимі PWM цей біт установлюється при зміні таймером/лічильником1 напрямку рахунка при $0000.

• Bit 1 – OCF0: Output Compare Flag 0 - Флаг 0 збігу виходу

Біт OCF0 установлюється при збігу стану таймера/лічильника0 і вмісту регістра OCR0 (Output Compare Register 0). Біт OCF0 апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1. При встановленому біті I у регістрі SREG, установлених OCIE0(Timer/Counter0 Compare Interrupt Enable) і OCF0 виконується переривання по збігу виходу таймера/лічильника0.

• Bit 0 - TOV1: Timer/Counter1 Overflow Flag - Флаг переповнення таймера/лічильника0

Біт TOV0 установлюється при переповненні таймера/лічильника0. Він апаратно очищається при обробці відповідного вектора переривання. Можливе очищення біта записом у флаг логічної 1. При встановленому біті I у регістрі SREG, установлених TOIE0 (Timer/Counterl Overflow Interrupt Enable) і TOV0 виконується переривання по переповненню таймера/лічильника0. У режимі PWM цей біт установлюється при зміні

Час реакції на переривання

Відгук на виконання всіх дозволених переривань AVR складає мінімум 4 тактових цикли. Протягом 4 тактових циклів після установки флага переривання виконується перехід за адресою вектора переривання для виконання підпрограми переривання. Протягом цих 4 циклів вміст лічильника команд (2 байти) опускається в стек і покажчик стека декрементується на 2. Вектор вказує перехід у підпрограму обробки переривання і цей перехід займає 3 тактових цикли. Якщо переривання виникне під час виконання багатоциклової команди, то команда завершується до початку обслуговування переривання. Повернення з підпрограми обробки переривання (як і виклик підпрограми) займає 4 тактових цикли. Протягом цих 4 циклів стан лічильника команд (2 байти) витягається зі стека і покажчик стека інкрементується на 2. Коли

AVR виходить з переривання, він завжди повертається в основну програму і виконує ще одну команду, перш, ніж почати обслуговування якогось відкладеного переривання.

Відзначимо, що регістр статусу SREG не обробляється апаратними засобами AVR, ні для переривань, ні для підпрограм.

При обробці підпрограм переривань, що вимагають збереження в SREG, запис повинен виконуватися програмними засобами користувача. Для переривань, що запускаються статичними подіями (наприклад збіг вмісту регістра порівняння 1A з станом таймера/лічильника1) флаг переривання встановлюється в момент настання події. Якщо флаг очищений, але умови виникнення переривання продовжують існувати, флаг не буде встановлюватися доти, поки ця подія не наступить знову.

3.7 Режими енергозбереження (Sleep Modes)

Для переведення в кожній із трьох режимів енергозбереження біт SE у регістрі MCUCR повинний бути встановлений у стан 1. Біти SM1 і SM0 регістра MCUCR визначають який з режимів Idle, Power Down чи Power Save буде запущений командою SLEEP.

При виникненні дозволеного переривання під час перебування MCU у режимі енергозбереження, MCU активується, виконує підпрограму обробки переривання і продовжує роботу до наступної команди SLEEP. Якщо під час режиму енергозбереження відбувається скидання, MCU активується і починає роботу з вектора скидання. Вміст регістрового файлу, SRAM і пам’яті I/O у процесі активації не змінюється.

Відзначимо, що якщо для повернення з режиму енергозбереження Power Down чи Power Save, використовується запуск переривання за рівнем, то низький рівень повинний утримуватися трохи довше, ніж час затримки скидання t TOUT, інакше мікроконтролер не активується.

РЕЖИМ IDLE

Якщо біти SM1/SM0 знаходяться в стані 00 команда SLEEP переводить MCU

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

РЕЖИМ POWER DOWN

При установці бітів SM1/SM0 у стан 10 команда SLEEP переводить MCU у режим Power Down. У цьому режимі зупиняється зовнішній генератор. Користувач може дозволити роботу сторожового таймера. Якщо сторожовий таймер дозволений, то активація MCU відбудеться по завершенні встановленого в сторожовому таймері періоду часу. Якщо зовнішнє джерело тактового сигналу підключене до виводу XTAL1, то активація MCU з режиму Power Down може відбуватися без затримки, зазвичай необхідної для стабілізації XTAL генератора. Режим такої активації дозволяється програмуванням перемичок SUT0/SUT1 у Flash пам’яті. Див. розділ **Режим паралельного програмування.**

РЕЖИМ POWER SAVE

При установці бітів SM1/SM0 у стан 11 команда SLEEP переводить MCU у режим Power Save. Цей режим, за одним виключенням, аналогічний режиму Power Down. Якщо таймер/лічильник0 тактується асинхронно, тобто біт ASO у регістрі ASSR установлений, таймер/лічильник0 буде працювати в режимі Power Save.

MCU буде активуватися перериваннями по переповненню чи збігу виходу таймера/лічильника0.

3.8 Таймери/лічильники

Мікроконтролери ATmega603/103 оснащені трьома таймерами/лічильниками загального призначення – двома 8-розрядними й одним 16-розрядним. Таймер/лічильник0, на додаток до звичайного режиму, може тактуватися асинхронно від зовнішнього генератора. Цей генератор оптимізований під використання кварцового кристала на частоту 32768 КГц, що дозволяє використовувати таймер/лічильник0 як годинник реального часу (Real Time Clock – RTC).

Таймер/лічильник0 оснащений своїм власним попереднім дільником. Таймери/лічильники 1 і 2 використовують виходи ступеней поділу загального 10-розрядного попереднього дільника. Ці два таймери/лічильника можна використовувати як таймери з вбудованою тимчасовою базою чи як лічильники, що переключаються за станом на зовнішньому виводі.

3.8.1 Попередні подільники таймерів/лічильників

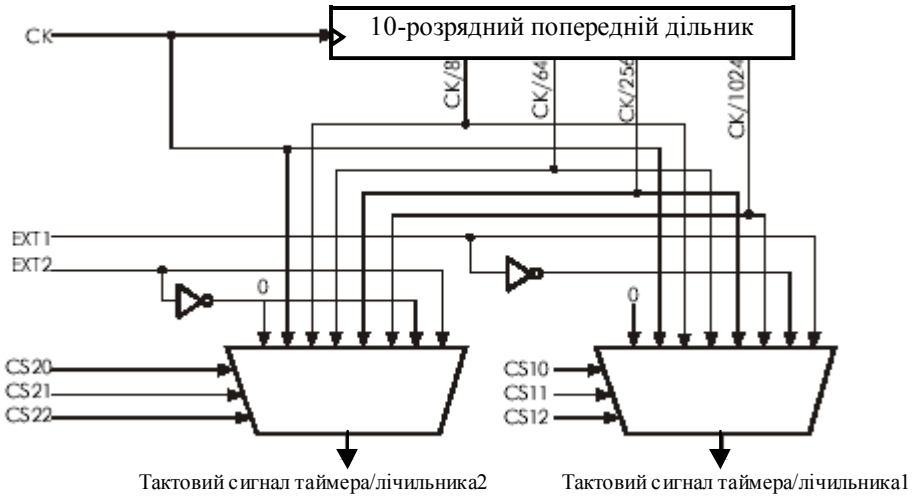


Рисунок 3.30 – Попередній дільник таймера/лічильника1 та таймера/лічильника2

Попередній дільник таймерів/лічильників 1 і 2 містить чотири ступені ділення: СК/8, СК/64, СК/256 та СК/1024, де СК – вхідний тактовий сигнал. Крім того, в якості джерел тактових сигналів можуть бути використані сигнали від зовнішніх джерел, тактовий сигнал СК та нульовий сигнал (stop).

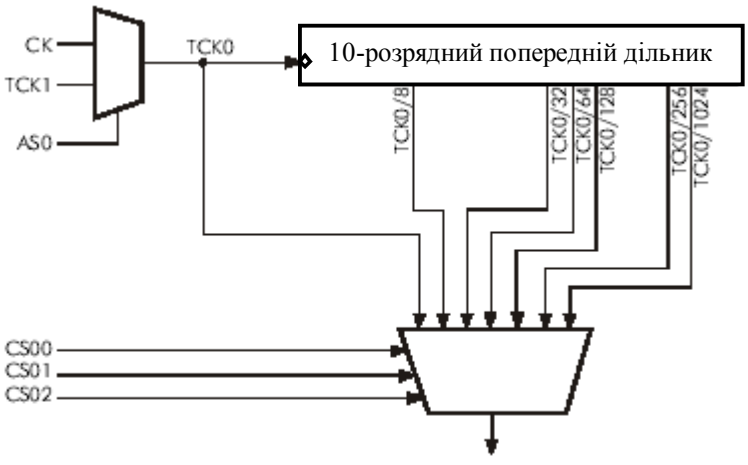


Рисунок 3.31 – Попередній дільник таймера/лічильника0

Тактовий сигнал таймера/лічильника0 позначений ТСК0. Цей тактовий сигнал за замовчуванням підключений до основного тактового сигналу системи СК. При установці біта ASO у регістрі ASSR таймер/лічильника буде асинхронно тактуватися сигналом з виводу TOSC1, що дозволяє використовувати таймер/лічильник0 як годинник реального часу (RTC). Генератор оптимізований під використання кварцового кристала з частотою 32768 КГц, що під'єднується між виводами TOSC1 і TOSC2.

3.8.2 8-розрядні таймери/лічильники T/C0 і T/C2

8-розрядний таймер/лічильник0 одержує тактовий сигнал чи безпосередньо від ТСК0 чи після проходження його через попередній дільник.

8-розрядний таймер/лічильник2 одержує тактовий сигнал безпосередньо від СК, після проходження його через попередній чи дільник від зовнішнього виводу. Обидва таймери/лічильника можуть бути зупинені, як це показано в описі регістрів керування таймерами/лічильниками TCCR0 і TCCR2.

У регістрі флагів переривання таймерів/лічильників TIFR зберігаються різні флаги стану регістрів (переповнення, збігу при порівнянні і захоплення події). Установки керуючих сигналів зберігаються в регістрах керування таймерами/ лічильниками TCCR0 і TCCR2. Установка дозволу/заборони переривань виробляється в регістрі масок переривань таймерів/лічильників TIMSK.

При тактуванні таймера/лічильника 2 зовнішнім тактовим сигналом цей сигнал синхронізується з тактовою частотою CPU. Для забезпечення правильної синхронізації зовнішнього сигналу необхідно, щоб мінімальний час між двома вхідними тактовими циклами був не менш одного циклу внутрішнього тактового сигналу CPU. Зовнішній тактовий сигнал синхронізується наростаючим фронтом внутрішнього тактового сигналу CPU.

Точність і дозвіл 8-розрядних таймерів/лічильників росте зі зменшенням коефіцієнта попереднього поділу. Аналогічним чином високий коефіцієнт попереднього поділу зручно використовувати при реалізації функцій з низькою швидкодією чи точної синхронізації дій, що рідко відбуваються.

Обидва таймери/лічильника підтримують дві функції порівняння виходу, використовуючи регістри порівняння виходу OCR0 і OCR2 як джерела даних, порівнюваних із вмістом таймерів/лічильників. У функції порівняння виходу входить і опція очищення лічильника при збігу і формування, при збігу, сигналу на виводах порівняння виходу - PB4(OC0/PWM0) і PB7(OC2/PWM2).

Таймери/лічильники 0 і 2 можна використовувати як 8-розрядні широтно-імпульсні модулятори (PWM). У цьому режимі таймер/лічильник, разом з регістром збігу виходу працюють як автономний ШІМ з центрованими імпульсами і без помилкових викидів. Докладніше ця функція описана в розділі **Таймери/лічильники О та 2 у ШІМ режимі.**

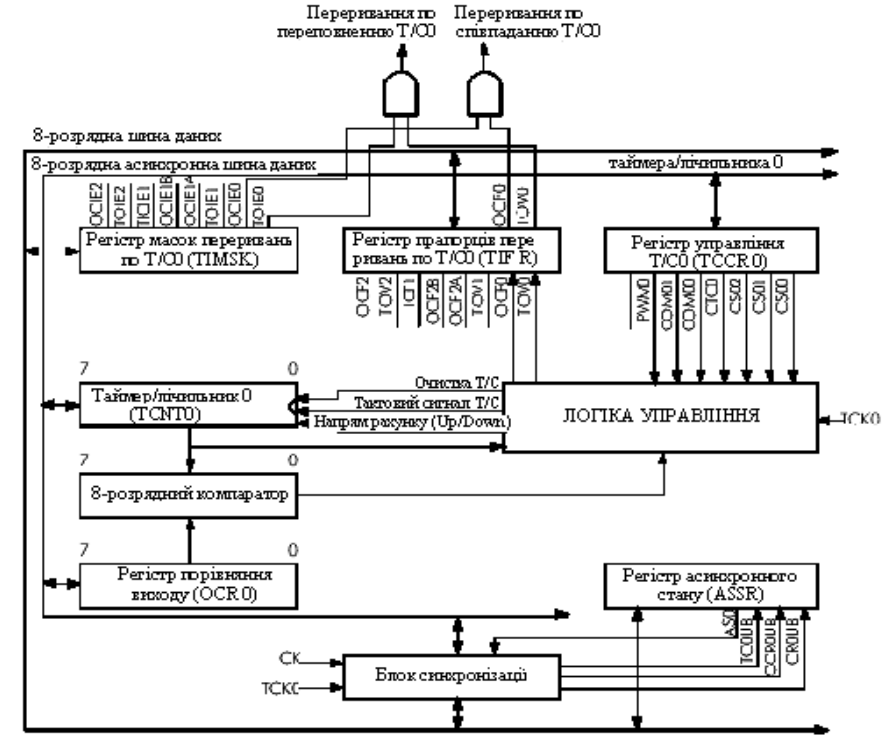


Рисунок 3.32 – Блок-схема таймера/лічильника0

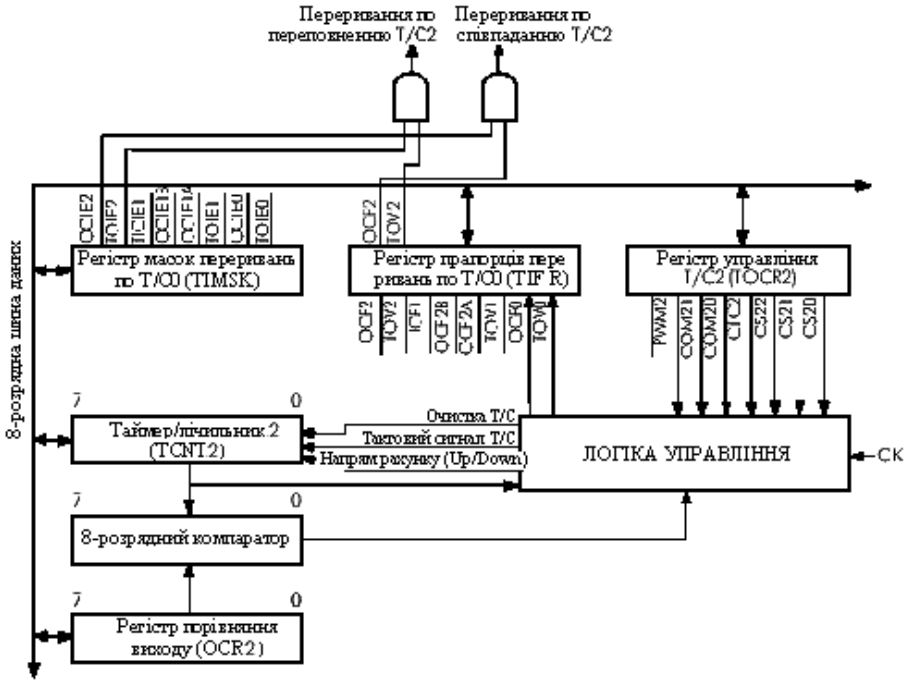


Рисунок 3.33 – Блок-схема таймера/лічильника2

РЕГІСТР КЕРУВАННЯ ТАЙМЕРОМ/ЛІЧИЛЬНИКОМ0 – TCCR0 – (THE TIMER/COUNTER0 CONTROL REGISTER)

Біти	7	6	5	4	3	2	1	0	
$33 {$53}	-	PWM0	COM01	COM00	CTC0	CS02	CS01	CS00	TCCR0
Читання/Запис	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

РЕГІСТР КЕРУВАННЯ ТАЙМЕРОМ/ЛІЧИЛЬНИКОМ2 – TCCR2 – (THE TIMER/COUNTER2 CONTROL REGISTER

Біти	7	6	5	4	3	2	1	0	
$25 {$45}	-	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	TCCR2
Читання/Запис	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

• **Bit 7 - Res; Reserved Bit - Зарезервований біт**

Даний біт у мікроконтролерах ATmega603/103 зарезервований і при зчитуванні завжди 0.

• **Bit 6 – PWM0 / PWM2: Pulse Width Modulator Enable - Дозвіл широтно-імпульсного модулятора**

Встановлений у стан 1 біт дозволяє режим ШІМ для таймерів/лічильників 0 чи 2. Режим докладно описаний у розділі **Таймери/лічильники 0 і 2 у ШІМ режимі.**

• **Bits 5,4 - COM01, COM00 / COM21, COM20: Compare Output Mode, bits 1 and 0 - Режим порівняння виходу, біти 1і 0**

Керуючі біти COMnl і COMn0 установлюють стан вихідних виводів PB4(OC0/PWM0) чи PB7(OC2/PWM2) після збігу в таймері/лічильнику2 Оскільки це альтернативна функція виводів порту I/O, то відповідний біт напрямку виводу повинний бути встановлений у стан 1. Керуюча конфігурація показана в таблиці 3.8.

Таблиця 3.8- Вибір режиму порівняння

COMnl	COMn0	Опис
0	0	Таймер /лічильник n від'єднаний від вихідного виводу OCn/PWMn.
0	1	Переключення вихідної лінії OCn/PWMn.
1	0	Очищення вихідної лінії OCn/PWMn (установка в стан 0).
1	1	Установка вихідної лінії OCn/PWMn (установка у стан 1).

Примітки: n = 0 або 2.

У ШІМ режимі функції цих бітів відрізняються. Докладний опис приведений у таблиці 3.11. При зміні бітів COMnl/COMn переривання в порівнянні виходу повинне бути заборонено очищенням його біта дозволу переривання в регістрі TIMSK. У противному випадку при зміні стану біта може відбутися переривання.

• **Bit 3 – CTC0 / CTC2: Clear Timor/Counter on Compara match - Очистити таймер/ лічильник при збігу**

При встановленому в стан 1 біті CTC0 чи CTC2 таймер/лічильник скидається у стан $00 протягом одного тактового циклу CPU після настання збігу. Якщо біт керування скинутий, то таймер продовжує рахувати і не використовується в процедурі порівняння. Оскільки факт збігу детектується в тактовому циклі CPU наступним за збігом, то ця функція буде поводитися трохи по іншому, якщо коефіцієнт попереднього поділу буде більше 1. Якщо використовується коефіцієнт попереднього поділу рівний 1 і в регістр порівняння A встановлено вміст C, то таймер буде продовжувати рахунок так як це робиться при встановленому CTC0/2.

... | C-1 | C | C+1 | 0 | 1 | ...

Якщо встановлений коефіцієнт поділу 8, таймер буде рахувати аналогічно наступній послідовності:

... |C-1,C-1,C-1,C-1,C-1,C-1,C-1,C-1|C,C,C,C,C,C,C,C|C+1,0,0,0,0,0,0,0,0|1,1,1, …

У ШІМ режимі стан цього біта значення не має.

• Bits 2,1,0 - CS02, CS01. CS00 / CS22, CS21, CS20: Clock Select bits 2, 1 and 0 - Біти вибору тактової частоти

Біти 2, 1 і 0 вибори тактової частоти таймера/лічильника2 підключають вихід визначеної ступіні попереднього дільника.

Таблиця 3.9- Вибір коефіцієнта розподілу попереднього дільника таймера/лічильника0

CS02	CS01	Cs00	Опис
0	0	0	Таймер/лічильник0 зупинений.
0	0	1	ТСК0.
0	1	0	ТСК0/8.
0	1	1	ТСК0/32.
1	0	0	ТСК0/64.
1	0	1	ТСК0/128.
1	1	0	ТСКО/256.
1	1	1	ТСКО/1024.

Таблиця 3.10- Вибір коефіцієнта поділу попереднього дільника таймера/лічильника2

CS22	CS21	CS20	Опис
0	0	0	Таймер/лічильник2 зупинений.
0	0	1	СК.
0	1	0	СК/8.
0	1	1	СК/64.
1	0	0	СК/256.
1	0	1	СК/1024.
1	1	0	Зовнішній вивід PD7(T2), фронт спадання.
1	1	1	Зовнішній вивід PD7(T2), фронт наростання.

Умова Stop забезпечує реалізацію функції дозволу/заборони таймера.

Режим розподілу СК реалізується безпосереднім поділом тактової частоти СК. Якщо для тактування таймера/лічильника2 використовується зовнішнє джерело, то переключення на виводі PD7/(T2) будуть впливати на лічильник, навіть якщо цей вивід сконфігурований як вихід.

ТАЙМЕР/ЛІЧИЛЬНИК0 – TCNT0 – (TIMER/COUNTER0)

Біти	7	6	5	4	3	2	1	0	
$32 {$42}	MSB							LSB	TCNT0
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

ТАЙМЕР/ЛІЧИЛЬНИК2 – TCNT2 – (TIMER/COUNTER2)

Біти	7	6	5	4	3	2	1	0	
$24 {$44}	MSB							LSB	TCNT2
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Вміст цих 8-розрядних регістрів є станом таймерів/лічильників. Обидва таймери/лічильника реалізовані як лічильники по наростанню чи реверсивні (у ШІМ режимі)

лічильники з можливістю читання/запису. Якщо в таймер/лічильник записане деяке значення й обране джерело тактового сигналу, то він продовжить відлік починаючи з записаного значення, з тактовою частотою лічильника.

РЕГІСТР ПОРІВНЯННЯ ВИХОДУ ТАЙМЕРА/ЛІЧИЛЬНИКА 0 - OCRO – (TIMER/COUNTER0 OUTPUT COMPARE REGISTER)

Біти	7	6	5	4	3	2	1	0	
$31 {$51}	MSB							LSB	OCR0
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

РЕГІСТР ПОРІВНЯННЯ ВИХОДУ ТАЙМЕРА/ЛІЧИЛЬНИКА2 - OCR2 - (TIMER/COUNTER2 OUTPUT COMPARE REGISTER)

Біти	7	6	5	4	3	2	1	0	
$23 {$43}	MSB							LSB	OCR2
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

Регістри порівняння виходів є 8-розрядними регістрами з можливістю читання/запису. Виконання процедури порівняння визначається регістрами TCCR0 і TCCR2. Збіг при порівнянні відбудеться тільки тоді, коли таймер/лічильник дорахує до значення вмісту OCR. Програмний запис того самого значення в таймер/лічильник і в регістр порівняння виходу не приведе до формування збігу при порівнянні.

Збіг при порівнянні приведе до установки флага переривання по збігу протягом тактового циклу CPU наступного за збігом. Необхідно приймати запобіжного заходу при роботі таймера/лічильника0 в асинхронному режимі, тобто встановлювати в стан 1 біт AC0 у регістрі ASSR. При записі в регістр OCR0 значення, пересилається в регістр по TCK0 такту, що випливає за операцією запису.

3.8.3 Таймери/лічильники 0 і 2 у ШІМ режимі

При встановленому ШІМ режимі таймер/лічильник і регістр порівняння виходу (OCR0 чи OCR2) формують 8-розрядний, без помилкових викидів і з правильною фазою ШІМ сигнал з виходом через PB4(OC0/PWM0) чи PB7(OC2/PWM2) виводи. Таймер/лічильник працює як реверсивний лічильник, що рахує від $00 до $FF, після чого він рахує в зворотну сторону до нуля і тільки після цього починає новий цикл. Коли стан лічильника збігається з вмістом регістра порівняння виходу виводи PB4(OC0/PWM0) чи PB7(OC2/PWM2) встановлюються чи очищаються, відповідно до встановленого, у регістрах керування таймерами/лічильниками TCCR0 і TCCR2, бітами COM01/COM00 чи COM21/COM20. Див. таблицю 3.11.

У ШІМ режимі при записі вміст регістра порівняння виходу пересилається на тимчасове збереження. Вміст фіксується при досягненні таймером/лічильником стану $FF. Такий прийом охороняє від появи ШІМ імпульсів збільшеної ширини (помилкових викидів) у випадку несинхронного запису OCR0 чи OCR2. Приклад див. на рис.3.34.

У проміжок часу між операціями запису і фіксації зчитування з OCR0 чи OCR2 приведе до зчитування з місця тимчасового збереження. Це означає, що найчастіше при читанні значення уставки зчитування буде здійснюватися з OCR0/2. При стані регістра OCR $00 чи $FF вихід ШІМ буде утримуватися

Таблиця 3.11– Вибір режиму порівняння в ШІМ режимі

COMn1	COMn0	Ефект, що створює на вивід Compare/PWM
0	0	Не під'єднаний.
0	1	Не під'єднаний.
1	0	Очищення при збігу, рахунок по нарощуванню. Установка при збігу, рахунок по спаданню (неінвертуючий ШІМ).
1	1	Очищення при збігу, рахунок по спаданню. Установка при збігу, рахунок по нарощуванню (інвертуючий ШІМ).

У ШІМ режимі флаг переповнення таймера (TOV0 чи TOV2) установлюється при зміні напрямку рахунку при $00. Переривання по переповненню таймерів 0 і 2 працюють так само, як і в нормальному режимі таймерів/лічильників, тобто вони спрацьовують коли TOV0 чи TOV2 установлені, і дозволені переривання по переповненню таймера і глобальному перериванню. Це відноситься також до флагів порівняння виходу таймерів

Частота ШІМ буде відповідати тактовій частоті таймера діленої на 510.

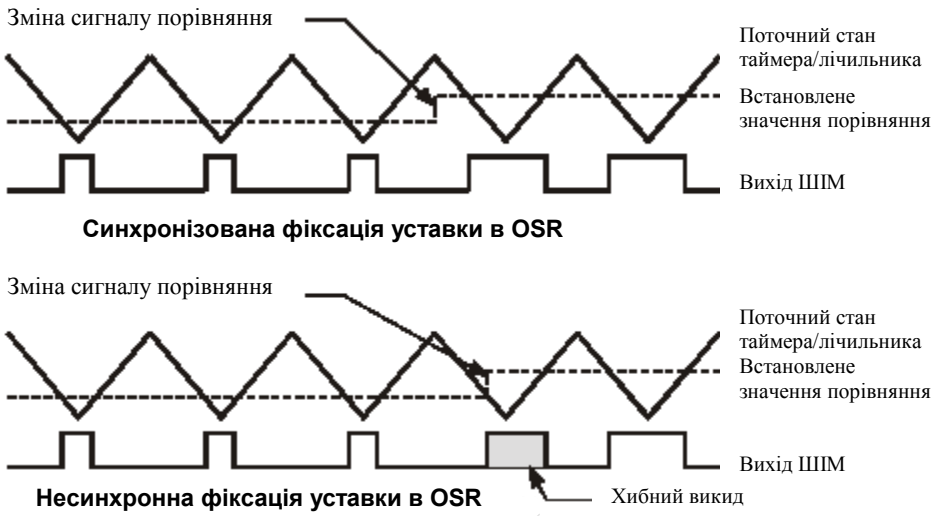


Рисунок 3.34 – Ефект несинхронної фіксації OCR

Таблиця 3.12– Стан ШІМ виходів при OCRn=$00 чи $FF

COMn1	COMn0	OCRn	Вихід PWMn
1	0	$00	L – низький рівень
1	0	$FF	H – високий рівень
1	1	$00	H – високий рівень
1	1	$FF	L – низький рівень

Примітки: n = 0 або 2.

3.8.4 Асинхронна робота таймера/лічильника0

При синхронній роботі таймера/лічильника0 всі операції і тактування ідентичні роботі таймера/лічильника2. Однак асинхронна робота має деякі особливості.

• **Це важливо! При переключенні між асинхронним і синхронним тактуванням таймера/лічильника0 регістри таймера TCNT0, OCR0 і TCCR0 можуть бути ушкоджені**. Безпечне переключення виконується наступною послідовністю дій:

1. Забороняються переривання OCIE0 і TOIE0 таймера0.
2. Відповідною установкою ASO вибирається джерело тактового сигналу.
3. У TCNT0, OCR0 і TCCR0 записуються нові значення.
4. Якщо виконується переключення в асинхронний режим: очікувати TCN0UB, OCR0UB і TCR0UB.

5. Дозволити переривання, якщо це необхідно.

• Генератор оптимізований під використання годинникового кристала з частотою 32,768КГц. Зовнішній тактовий сигнал, поданий на цей вивід, проходить через той самий підсилювач зі смугою пропускання 256 КГц. Таким чином, зовнішній тактовий сигнал повинний бути в діапазоні від 0 Гц до 256 КГц. Частота зовнішнього тактового сигналу, поданого на вивід TOSC1, не повинна перевищувати однієї четвертої від тактової частоти CPU. Відмітимо, що тактова частота CPU може бути нижче частоти XTAL, якщо дозволений поділ частоти XTAL.

• При запису в один з регістрів TCNT0, OCR0 або TCCR0, записувана величина пересилається в регістр тимчасового зберігання і фіксується після двох позитивних фронтів TOSC1. Користувач не повинний записувати нове значення поки вміст регістра тимчасового зберігання не буде передано за призначенням. Кожний з вказаних регістрів має свій власний регістр тимчасового зберігання, це означає, приміром, що запис у TCNT0 не спотвориться при запису в OCR0. Для того, щоб переконатися у виконанні пересилання в регістр призначення використовується регістр статусу асинхронного режиму (Asynchronous Status Register - ASSR).

• При введенні режиму Sleep після запису в регістри TCNT0, OCR0 чи TCCR0 користувач повинний очікувати, поки записуваний регістр не буде обновлений, якщо таймер/лічильник0 використовується для активації приладу. В іншому випадку MCU перейде в режим Sleep перш, ніж зміни зроблять якийсь ефект. Це особливо важливо, якщо для активації приладу використовується переривання за порівнянням виходу0; порівняння виходу забороняється під час запису в OCR0 чи TCNT0. Якщо цикл запису не завершений (тобто користувач введе режим Sleep перш, ніж біт OCROUB буде скинутий у 0) прилад ніколи не одержить збігу при порівнянні і MCU не буде активований. Якщо таймер/лічильник0 використовується для активації приладу з режиму Power Save і якщо користувач має намір відновити режим Power Save, то необхідно вжити запобіжного заходу - для скидання логіки переривання необхідний один цикл TOSC1. Якщо час між активацією і відновленням режиму Power Save менше одного циклу TOSC1 переривання не відбудеться і прилад не буде активований Якщо користувач сумнівається в тому, що проміжок часу перед відновленням режиму Power Save достатній, необхідно використовувати наступний алгоритм:

1. Записати значення в TCCR0, TCNT0 чи OCR0.

2. Очікувати, поки відповідний флаг зайнятого відновлення в регістрі ASSR не буде скинутий у 0.

3. Увести режим Power Save

• Генератор частоти 32гц таймера/лічильника0 працює завжди, за виключенням режиму Power Down. При відновленні живлення чи активації з режиму Power Down користувач повинен пам’ятати про те, що генератору для стабілізації необхідний час порядку однієї секунди. Користувачу рекомендується виждати не менш однієї секунди, перш ніж використовувати таймер/лічильник0 після подачі живлення чи активації з режиму Power Down.

• Опис активації приладу з режиму Power Save при асинхронному тактуванні таймера. Коли умови переривання виконані, на наступному циклі тактової частоти таймера починається процес активації, тобто стан таймера повинен збільшитися як мінімум на одиницю, перш, ніж процесор зможе прочитати стан лічильника. Флаги переривань обновляються за три тактових цикли процесора після запуску тактової частоти процесора. Протягом цих циклів процесор виконує команди, але умови переривання ще не зчитувані і підпрограма обробки переривання не може почати виконання.

Під час асинхронної роботи синхронізація флагів переривань асинхронного таймера займає три тактових цикли процесора плюс один цикл таймера. У такий спосіб вміст таймера збільшується мінімум на одиницю, перш ніж процесор зможе прочитати вміст таймера, що викликав установку флага переривання. Вивід порівняння виходу змінює стан відповідно до тактового сигналу таймера і не синхронізований з тактовою частотою процесора.

РЕГІСТР СТАТУСУ АСИНХРОННОГО РЕЖИМУ - ASSR - (ASYNCHRONOUS STATUS REGISTER)

Біти	7	6	5	4	3	2	1	0	
$30 {$50}	-	-	-	-	AS0	TCN0UB	OCR0UB	TDR0UB	ASSR
Читання/Запис	R	R	R	R	R/W	R	R	R	
Початковий стан	0	0	0	0	0	0	0	0	

• Bit 7..4 – Res: Reserved Bits – Зарезервовані біти

Ці біти в мікроконтролерах ATmega603/103 зарезервовані і при зчитуванні завжди будуть показувати 0.

• Bit 3-AS0: Asynchronous Timer/Counter0 - Асинхронний режим таймера/лічильника0 При встановленому в стан 1 біті таймер/лічильник0 тактується сигналом виводу TOSC1. При очищеному біті таймер/лічильник0 тактується внутрішнім тактовим сигналом СК. При зміні стану цього біта вміст TCNT0 може бути ушкоджено.

• Bit 2 – TCM0UB: Timer/Counter0 Update Busy - Таймер/лічильник0 зайнятий для відновлення

Біт встановлюється в стан 1 при роботі таймера/лічильника 0 в асинхронному режимі і записаному TCNT0. При відновленні записаного в TCNT0 значення вмістом регістра тимчасового збереження біт апаратно очищається.

Біт в логічному стані 0 означає, що TCNT0 готовий до обновлення новим значенням.

• Bit 1 – OCR0UB Output Compare Register0 Update Busy - Порівняння виходу регістра0 зайнято для відновлення

Біт встановлюється в стан 1 при роботі таймера/лічильника0 в асинхронному режимі і записаному OCR0. При відновленні записаного в OCR0 значення вмістом регістра тимчасового збереження біт апаратно очищається.

Біт у логічному стані 0 означає, що OCR0 готовий до обновлення новим значенням.

• Bit 0 – TCR0UB: Timer/Counter Control Register Update Busy - Регістр керування таймера/лічильника0 зайнятий для відновлення

Біт встановлюється в стан 1 при роботі таймера/лічильника0 в асинхронному режимі і записаному TCCR0. При відновленні записаного в TCCR0 значення вмістом регістра тимчасового збереження біт апаратно очищається.

Біт у логічному стані 0 означає, що TCCR0 готовий до обновлення новим значенням. Якщо запис виконується в кожній із трьох регістрів таймера /лічильника0 коли їхні флаги зайнятості для відновлення встановлені, то записуване значення може бути ушкоджене і привести до небажаного переривання.

Зчитування станів TCNT0, OCR0 і TCCR0 відрізняється При читанні стану TCNT0 зчитується дійсний вміст, при читанні станів OCR0 чи TCCR0 зчитується вміст регістрів тимчасового збереження.

3.8.5 16-розрядний таймер/лічильник1

Блок-схема таймера/лічильника1 приведена на рис. 3.35.

16-розрядний таймер/лічильник1 може одержувати тактовий сигнал від СК, СК після попереднього дільника і від зовнішнього виводу. Крім того, його можна зупинити, як показано в описі регістра керування таймером/лічильником1 - TCCR1B (Timer/Counter1 Control Register). У регістрах керування TCCR1A і TCCR1B знаходяться різні флаги, що вказують на переповнення, збіг при порівнянні і випадки захоплення подій. У регістрі масок переривань TIMSK (Timer/Counter Interrupt Mask Register) установлюються дозволи/заборони переривань таймера/лічильника1. При зовнішньому тактуванні таймера/лічильника1 зовнішній сигнал синхронізується частотою тактового генератора CPU. Для правильної роботи таймера/лічильника1 по зовнішньому тактовому сигналі мінімальний час між двома переключеннями зовнішнього тактового сигналу повинний бути не менш одного періоду

тактового сигналу CPU. Синхронізація зовнішнього тактового сигналу ведеться наростаючим фронтом внутрішнього тактового сигналу CPU.

Найкращі точність і роздільну здатність 16-розрядний таймер/лічильник1 забезпечує при найменшому коефіцієнті попереднього поділу. З іншого боку, високий коефіцієнт попереднього поділу зручний при реалізації таймером/лічильником1 низькошвидкісних функцій чи точної синхронізації дій, які рідко відбуваються. Таймер/лічильник1 підтримує дві функцій порівняння виходу, використовуючи регістр1 порівняння виходів A и B – OCR1A і OCR1B як джерела даних, порівнюваних з вмістом таймера/лічильника1. Функції порівняння виходу включають очищення лічильника по збігу порівняння і вплив на виводи порівняння виходу при обох збігах порівняння.

Таймер/лічильник1 може бути використаний у якості 8, 9 чи 10-розрядного широтно-імпульсного модулятора. У цьому режимі лічильник і регістри OCR1A/OCR1B працюють як здвоєний самостійний ШІМ із зцентрованими імпульсами, без формування помилкових імпульсів. Див. розділ **Таймер/лічильник1 у ШІМ режимі**, де докладно описана ця функція.

Функція захоплення входу таймера/лічильника1 забезпечує захоплення вмісту таймера/лічильника1 у регістр захоплення входу, що запускається зовнішньою подією на виводу входу захоплення PD4/(IC1). Реальні установки захоплення події визначаються регістром керування таймером/лічильником1 TCCR1B (Timer/Counter1 Control Register).

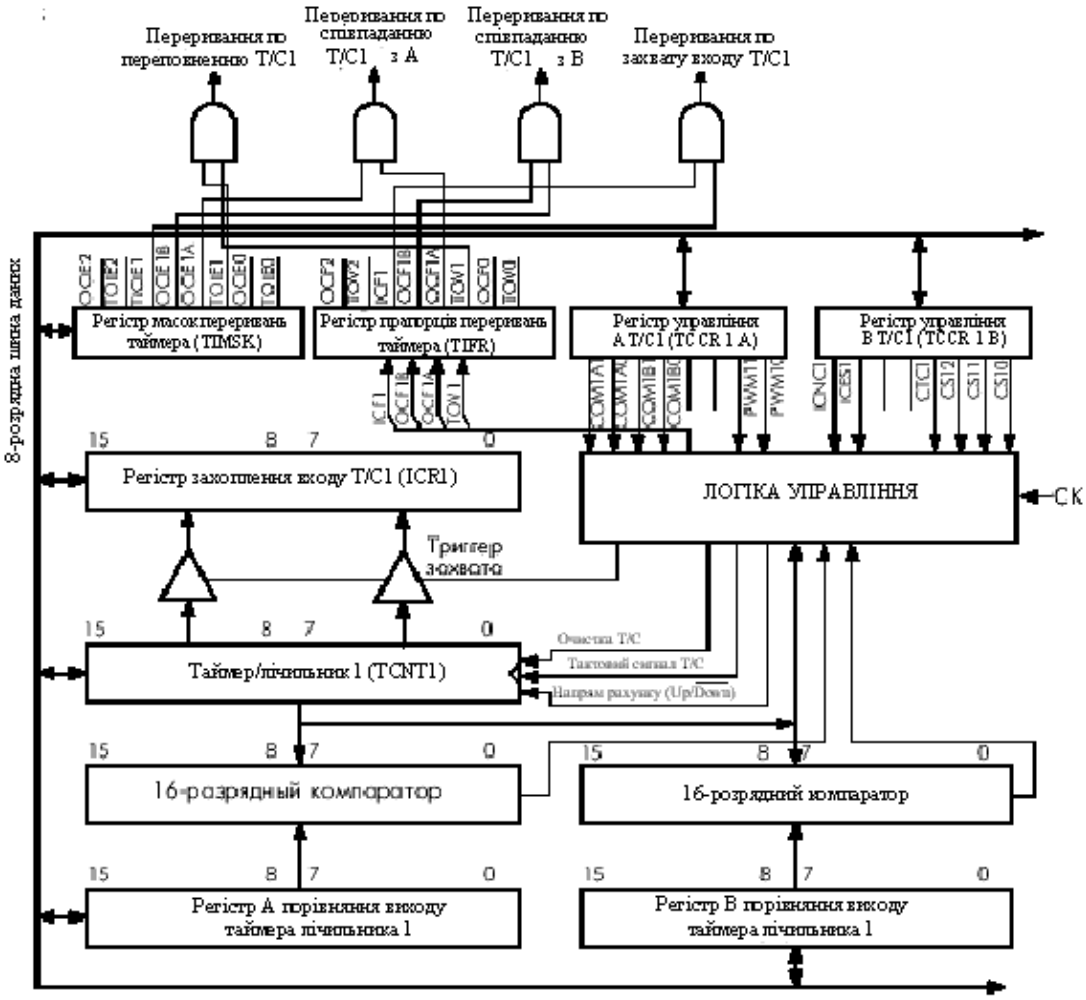


Рисунок 3.35 – Блок-схема таймера/лічильника1

Крім того, для переключення входу захоплення може бути використаний аналоговий компаратор. Докладніше дана функція описана в розділі **Аналоговий компаратор.**

Якщо дозволена функція зменшення шуму, дійсні умови переключення події захоплення тестуються чотирма вибірками, перш ніж захоплення буде активоване. Тестування сигналу на вхідному висновку виробляється з частотою XTAL.

РЕГІСТР КЕРУВАННЯ А ТАЙМЕРА/ЛІЧИЛЬНИКА1 - TCCR1A - (TIMER/COUNTER1 CONTROL REGISTER A)

Біти	7	6	5	4	3	2	1	0	
$2F {$4F}	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	TCCR1A
Читання/Запис	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

• **Bits 7,6 - COM1A1. COM1A0: Compare Output ModelA. bits 1 and 0 – Режим1А порівняння виходу, біти 1 і 0**

Керуючі біти COM1A1 і COM1AT визначають характер сигналу виходу, наступного за збігом при порівнянні таймера/лічильника1. Сигнал виходу надходить на вивід OC1A (Oulput CompareA). Оскільки це альтернативна функція порту I/O, то відповідний біт керування напрямком повинний бути встановлений у 1 (вивід працює на вихід). Конфігурування керування представлено в таблиці 3.13.

Таблиця 3.13–Вибір режиму порівняння 1

COM1X1	COM1X0	Опис
0	0	Таймер/лічильник1 відключений від виводу виходу OC1X.
0	1	Перемикання вихідної лінії OC1X.
1	0	Очищення вихідної лінії OC1X (на лінії низький рівень).
1	1	Установка вихідної лінії OC1X (на лінії високий рівень).

Примітка: X=A або B

• **Bit 5, 4 - COM1B1, COM1B0: Compare Output Mode 1B, bits 1 and 0 - Режим 1В порівняння виходу, біти 1 і 0**

Керуючі біти COM1B1 і COM1B0 визначають характер сигналу виходу, що випливає за збігом при порівнянні таймера/лічильника1. Сигнал виходу надходить на вивід OC1B (Output CompareB). Оскільки це альтернативна функція порту I/O, то відповідний біт керування напрямком повинний бути встановлений у 1 (вивід працює на вихід). Конфігурування керування представлено в таблиці 3.13.

При зміні бітів COM1X1/COM1X0 переривання в порівнянні виходу1 повинні бути заборонені очищенням бітів дозволу переривання в регістрі TIMSK. У противному випадку при зміні бітів може відбутися переривання.

• **Bits 3..2 - Res: Reserved bits - Зарезервовані біти**

Ці біти в мікроконтролерах ATmega603/103 зарезервовані і при зчитуванні завжди будуть 0.

• **Bits 1..0- PWM11, PWM10: Pulse Width Modulator Select Bits - Біти вибору режиму ШІМ**

Дані біти визначають установку режиму ШІМ, як це показано в таблиці 3.14. Подробиці див. у розділі **Таймер/лічильник1 у ШІМ режимі.**

Таблиця 3.14– Вибір ШІМ режиму

PWM11	PWM10	Опис
0	0	Робота таймера/лічильника1 в ШІМ режимі заборонена.
0	1	Робота таймера/лічильника1 в 8-розрядному ШІМ режимі.
1	0	Робота таймера/лічильника1 в 9-розрядному ШІМ режимі.
1	1	Робота таймера/лічильника1 в 10-розрядному ШІМ режимі.

РЕГІСТР КЕРУВАННЯ В ТАЙМЕРА/ЛІЧИЛЬНИКА1 – TCCR1B – (TIMER/COUNTER1 CONTROL REGISTER B)

Біти	7	6	5	4	3	2	1	0	
$2E {$4E}	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	TCCR1B
Читання/Запис	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

• Bit 7 - ICNC1: Input Copturel Noise Canceler (4 CKs) - Установка режиму придушення шуму на вході захоплення 1

При скинутому в стан 0 біті ICNC1 функція придушення шуму вхідного тригера захоплення заборонена. Вхід захоплення переключається по першому наростаючому/ спадаючому фронту, що надійшов на вивід входу захоплення PD4(ICl). При встановленому в стан 1 біті ICNC1 виконуються чотири послідовних опитування стану виводу PD4(ICl) і всі чотири вибірки повинні мати однаковий (високий/низький), обумовлений бітом ICES1, рівень. Частота опитування відповідає частоті XTAL.

• Bit 6 - ICES1: Input Copturel Edge Select – Вибір фронту спрацювання на вході захоплення 1

При скинутому в стан 0 біті ICE51 вміст таймера/лічильника1, по спадаючому фронту на виводі входу захоплення PD4(IC1), пересилається в регістр захоплення входу ICR1. При встановленому в 1 біті ICES1 вміст таймера/лічильника1 пересилається в регістр захоплення входу ICR1 по наростаючому фронті на виводі входу захоплення PD4(IC1).

• Bits 5.4 - Res: Reserved bits - Зарезервовані біти

Ці біти в мікроконтролерах ATmega603/103 зарезервовані і при зчитуванні завжди будугь 0.

• Bit 3 – CTC1: Clear Timer/Counter1 on Compare Match – Очищення таймера/лічильника1 по збігу

При встановленому в стан 1 біті CTC1 таймер/лічильник1 скидається в стан $0000 протягом тактового циклу, що випливає за збігом при порівнянні. Якщо біт CTC1 очищений, таймер/лічильник1 продовжує відлік і не реагує на збіг при порівнянні. Оскільки збіг при порівнянні детектується протягом тактового циклу CPU наступного за збігом, то поводження функції буде відрізнятися при установці коефіцієнта попереднього поділу таймера/лічильника1 більшому за 1. При коефіцієнті попереднього поділу 1 і встановленому в регістрі порівняння стані С таймер буде рахувати відповідно до установки CTC1:

...|C-1|C|C+1|0|1| ...

При встановленому коефіцієнті попереднього поділу 8 таймер буде рахувати так:

...|C-1,C-1,C-1,C-1,C-1,C-1,C-1,C-1|C,C,C,C,C,C,C,C|C+1,0,0,0,0,0,0,0,0| ...

У ШІМ режимі стан біту CTC1 значення не має.

• Bits 2,1,0 - CS12, CS11, CS10: Clock Select1, bit 2,1 and 0 - Вибір джерела тактової частоти, біти 2, 1 і 0

Установкою стану даних бітів здійснюється вибір джерела тактового сигналу (у тому числі коефіцієнта попереднього поділу).

Stop умова виконує функцію дозволу/заборони таймера/лічильника1. У режимах з попереднім поділом на відповідний коефіцієнт поділяється частота СК тактового генератора.

При використанні зовнішнього тактування необхідно виконати відповідні установки в регістрі керування напрямом (очищення переводить вивід в режим входу).

Таблиця 3.15 – Вибір джерела тактового сигналу таймера/лічильника1

CS12	CS11	CS10	Опис
0	0	0	Stop умова – таймер/лічильник1 зупинено.
0	0	1	СК.
0	1	0	СК/8.
0	1	1	СК/64.
1	0	0	СК/256.
1	0	1	СК/1024.
1	1	0	Зовнішній тактуючий сигнал на виводі T1, фронт наростання.
1	1	1	Зовнішній тактуючий сигнал на виводі T1, фронт спадання.

ТАЙМЕР/ЛІЧИЛЬНИК1 – TCNT1H ТА TCNT1L

Біти	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LCB	TCNT1L
	7	6	5	4	3	2	1	0	
Читання/Запис	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	
Початковий стан	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

16-розрядний регістр містить поточне значення 16-розрядного таймера/лічильника1. Для того, щоб CPU могло зчитувати/записувати і старший, і молодший байти цього регістра одночасно, звертання реалізоване за допомогою 8-розрядного регістра тимчасового збереження (TEMP). Цей регістр використовується також при звертанні до OCR1A, OCR1B і ICR1. Якщо основна програма і підпрограми обробки переривань використовують звертання до регістрів за допомогою TEMP, то переривання повинні бути заборонені на час звертання з основної програми.

• Запис у таймер/лічильник1 - TCNT1

Коли CPU робить запис у старший байт (TCNT1H) записувані дані розміщаються в регістрі TEMP. Потім, коли CPU робить запис у молодший байт (TCNT1L) дані молодшого байта поєднуються з байтом даних регістра TEMP і всі 16 бітів одночасно переписуються в регістр таймера/лічильника TCNT1. Отже, при 16-розрядних операціях звертання до старшого байта (TCNT1H) повинне виконуватися першим. При використанні таймера/лічильника1 у якості 8-роэрядного таймера досить робити запис тільки молодшого байта

• Читання таймера/лічильника1 - TCNT1

Коли CPU зчитує молодший байт (TCNT1L), то вміст TCNT1L направляється безпосередньо в CPU, вміст старшого байта (TCNT1H) розміщається в регістрі TEMP і при зчитуванні CPU старшого байта (TCNT1H) його вміст CPU приймає з регістра TEMP. Отже, при 16-розрядних операціях першим повинне виконуватися звертання до молодшого байта (TCNT1L). При використанні таймера/лічильника1 в якості 8-розрядного таймера достатньо здійснювати запис тільки молодшого байта.

Таймер/лічильник виконаний у вигляді лічильника з нарощуванням чи реверсивного лічильника (в ШІМ режимі) і можливістю читання/запису. Якщо в таймер/лічильник1 занесено деяке значення і вибране джерело тактового сигналу, то таймер/лічильник1 продовжить відлік через один тактовий цикл після установки в ньому записаного значення.

РЕГІСТР ПОРІВНЯННЯ А ВИХОДУ ТАЙМЕРА/ЛІЧИЛЬНИКА1 - OCR1AH ТА OCR1AL – (TIMER/COUNTER1 OUTPUT COMPARE REGISTER)

Біти	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	7	6	5	4	3	2	1	0	
Читання/Запис	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	
Початковий стан	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

РЕГІСТР ПОРІВНЯННЯ В ВИХОДУ ТАЙМЕРА/ЛІЧИЛЬНИКА1 OCR1BH ТА OCR1BL – (TIMER/COUNTER1 OUTPUT COMPARE REGISTER)

Біти	15	14	13	12	11	10	9	8	
$29 ($49)	MSB								OCR1BH
$28 ($48)								LSB	OCR1BL
	7	6	5	4	3	2	1	0	
Читання/Запис	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	
Початковий стан	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

16-розрядні регістри порівняння виходу забезпечують і читання і запис.

Регістри порівняння виходу таймера/лічильника1 зберігають дані постійно порівнювані зі станом таймера/лічильника1. Дія, що запускається збігом при порівнянні, визначається вмістом регістра керування і стани таймера/лічильника1. Збіг при порівнянні може відбутися тільки якщо таймер/лічильник1 дорахує до значення вмісту OCR. Якщо в TCNT1 і OCR1A чи OCR1B програмно будуть записані однакові значення, то значення, то збіг при порівнянні згенеровано не буде.

Збіг при порівнянні встановлює флаг переривання по збігу в тактовому циклі CPU наступному за самим збігом.

Оскільки регістри порівняння виходу OCR1A і OCR1B є 16-розрядними, то для забезпечення одночасного занесення старшого і молодшого байтів даних у регістри OCR1A/B використовується регістр тимчасового збереження TEMP. Коли CPU записує старший байт, то дані тимчасово зберігаються в регістрі TEMP. Коли ж CPU записує молодший байт у OCR1AL чи OCR1BL, то одночасний вміст регістра OCR1BH переписується в OCR1AH чи OCR1BH. Отже, при 16-розрядних операціях старші байти регістрів OCR1A/B повинні записуватися першими.

Крім того, регістр TEMP використовується при звертанні до TCNT1 і ICR1. Якщо основна програма і підпрограми обробки переривань, використовують звертання до регістрів за допомогою TEMP, то переривання повинні бути заборонені на час звертання з основної програми.

РЕГІСТР ЗАХОПЛЕННЯ ВХОДУ ТАЙМЕРА/ЛІЧИЛЬНИКА1 - ICR1H І ICR1L - (TIMER/COUNTER1 INPUT CAPTURE REGISTER)

Біти	15	14	13	12	11	10	9	8	
$27 ($47)	MSB								ICR1H
$26 ($46)								LSB	ICR1L
	7	6	5	4	3	2	1	0	
Читання/Запис	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Початковий стан	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

16-розрядний регістр захоплення входу забезпечує тільки читання вмісту.

При виявленні на виході захоплення входу PD4(ICl) наростаючого чи спадаючого фронтусигналу (обумовленого установкою ICES1) поточний стан таймера/лічильника1 пересилається в регістр захоплення входу ICR1. Одночасно встановлюється в стан 1 флаг захоплення входу ICF1.

Оскільки регістр захоплення входу є 16-розрядним, то для забезпечення одночасного читання старшого і молодшого байтів даних регістра ICR1 використовується регістр тимчасового збереження TEMP. При зчитуванні CPU даних молодшого байта вміст ICR1L пересилається в CPU, а вміст старшого байта ICR1H розміщається в регістрі TEMP, читання старшого байта полягає в переносі в CPU вмісту регістра тимчасового збереження TEMP. Отже, при читанні всього 16-розрядного регістра операцію читання необхідно починати з молодшого байта ICR1L. Регістр TEMP використовується також при звертанні до TCNT1, OCR1A і OCR1B. Якщо основна програма і підпрограми обробки переривань, використовують звертання до регістрів за допомогою TEMP, то переривання повинні бути заборонені на час звертання з основної програми.

3.8.6 Таймер/лічильник1 у ШІМ режимі

При встановленому ШІМ режимі таймер/лічильник1 і регістри порівняння виходу A та B (OCR1A і OCR1B), утворять здвоєний 8, 9 чи 10-розрядний автономний генератор ШІМ із правильним чергуванням фаз, відсутністю помилкових викидів і виходами на виводи PD5(OC1A) і OC1B. Таймер/лічильник1 працює як реверсивний лічильник, що рахує від 30000 до TOP (див таблицю 3.16), при якому напрямок рахунку міняється і відлік ведеться до нуля, після чого цикл повторюється Коли стан лічильника збіжиться з вмістом 10 молодших бітів OCR1A чи OCR1B, виводи PD5(OC1A)/OC1B встановлюються чи очищаються, відповідно до установок бітів COM1A1/COM1A0 чи COM1B1/COM1B0 у регістрі керування таймером/лічильником1 TCCR1A. Подробиці див. у таблиці 3.17.

Таблиця 3.16– TOP значення таймера та частота ШІМ

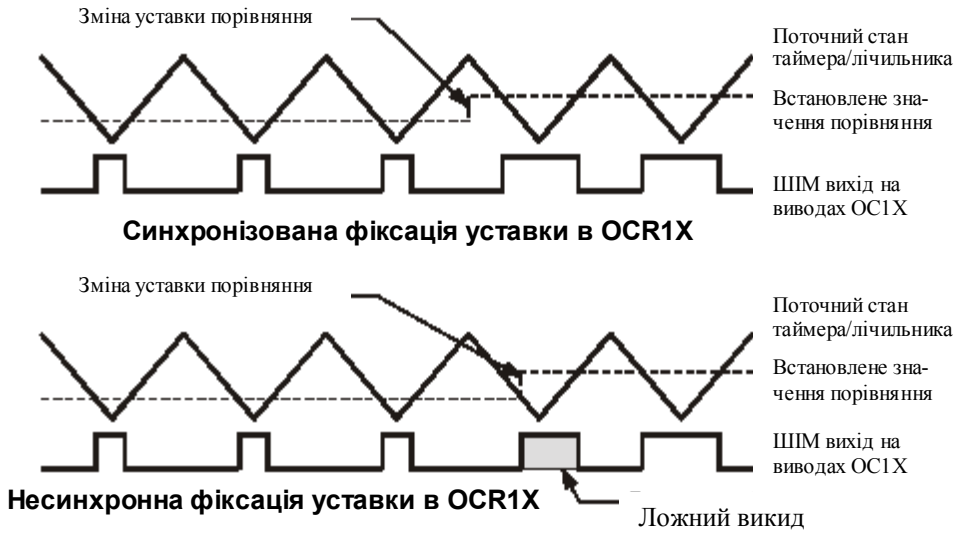
Дозвіл ШІМ	TOP значення таймера	Частота ШІМ
8-розрядний	$00FF(255)	f TC1/510
9-розрядний	$01FF(511)	f TC1/1022
10-розрядний	$03FF(1023)	f TC1/2046

Таблиця 3.17– Вибір режиму порівняння1 в ШІМ режимі

COM1X1	COM1X0	Вихідний сигнал на OCX1
0	0	Не підключений.
0	1	Не підключений.
1	0	Очищається по збігу при рахунку вверх. Встановлюється по збігу при рахунку вниз (не інвертований ШІМ).
1	1	Очищається по збігу при рахунку вниз. Встановлюється по збігу при рахунку вверх (інвертований ШІМ).

Примітка: X=A або B

Відзначимо, що в ШІМ режимі молодші 10 розрядів OCR1A/OCR1B, при записі, пересилаються в комірки тимчасового збереження. Вони фіксуються по досягненні таймером/лічильником1 значення TOP. Таким способом забезпечується захист від появи розширених ШІМ імпульсів {помилкових викидів - glitches) при несинхронному записі OCR1A/OCR1B. Див приклад на рис. 3.36.



Примітка: X=A або B

Рисунок 3.36 – Ефект несинхронної фіксації OCR1

При читанні OCR1A чи OCR1B, у проміжку часу між записом і фіксацією, буде зчитано вміст комірки тимчасового збереження. Це означає, що більш раннє записане значення завжди буде зчитуватися з OCR1A/B.

Коли OCR1 містить 30000 чи TOP, вивід OC1A/OC1B залишається на низькому рівні, відповідно установкам COM1A1/COM1A0 чи COM1B1/COM1B0. Це відображено в таблиці 3.18.

Таблиця 3.18– Стан виходів в ШІМ режимі при OCR1X=$0000 чи TOP

COM1X1	COM1X0	OCR1X	Стан виводів OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

Примітка: X = A чи B

У ШІМ режимі флаг переповнення таймера1 (TOV1) встановлюється при зміні напрямку рахунка по досягненні значення $0000. Переривання по переповненню таймера1 працює так само як і в звичайному режимі таймера/лічильника, тобто воно виконується, коли TOV1 встановлено і дозволені переривання по переповненню таймера1 і глобальні переривання. Це відноситься і до флагів порівняння виходу таймера1 і до переривань.

3.9 Сторожовий таймер (Watchdog Timer)

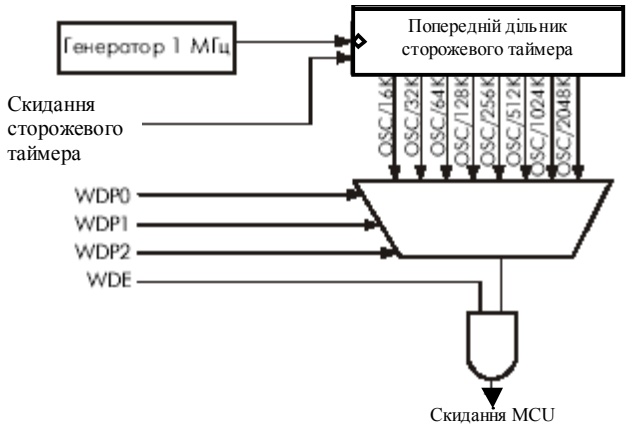


Рисунок 3.37 – Сторожовий таймер

Сторожовий таймер тактується окремим вбудованим генератором, що працює з частотою 1 МГц при типовій напрузі живлення Vcc=5В (див. типові значення при інших значеннях Vcc). Установкою коефіцієнта попереднього поділу можна змінювати тривалість інтервала до скидання по сторожовому таймері від 16 тис. до 2048 тис. циклів (від 16 до 2048 мс). Команда WDR (Watchdog Reset) скидає сторожовий таймер.

З моменту скидання сторожового таймера можна встановити вісім періодів тривалості тактового сигналу, установлюючи, у такий спосіб тривалість періоду скидання. Якщо період скидання завершується (протягом цього періоду не надійшов сигнал скидання сторожового таймера), то мікроконтролер ATmega603/103 скидається і його робота продовжується по вектору скидання. Подробиці тактування скидання по сторожовому таймері див. вище в розділі **Скидання за сторожовим таймером.**

При дозволі сторожового таймера його стан невідомий і перш, ніж дозволяти сторожовий таймер, необхідно виконати команду WDR. В іншому випадку прилад може бути скинутий швидше, ніж буде виконана команда WDR, прописана після дозволу. Для запобігання випадкової заборони, заборона сторожового таймера повинна супроводжуватися спеціальною процедурою вимикання. Подробиці в описі регістра керування сторожовим таймером.

РЕГІСТР КЕРУВАННЯ СТОРОЖОВИМ ТАЙМЕРОМ - WDTCH - (WATCHDOG TIMER CONTROL REGISTER)

Біти	7	6	5	4	3	2	1	0	
$21 ($41)	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Читання/Запис	R	R	R	R/W	R/W	R/W	R/W	R/W	
Початковий стан	0	0	0	0	0	0	0	0	

• Bits 7..5 – Res: Reserved bits – Зарезервовані біти

Ці біти в мікроконтролерах ATmega603/103 зарезервовані і при зчитуванні завжди будуть 0.

• Bit 4 – WDTOE: Watch Dog Turn Off Enable - Дозвіл відключення сторожового таймера

Дані біти повинні бути встановлені в стан 1 при очищенні біта WDE. В іншому випадку сторожовий таймер не буде заборонений. Установлений біт апаратно очищається після чотирьох тактових циклів. Див опис біта WDE у процедурі заборони сторожового таймера.

• Bit 3 – WDE: Watch Dog Enable - Дозвіл сторожового таймера

Якщо біт WDE встановлений у стан 1 (сторожовий таймер дозволений) і якщо біт WDE очищений, то функціонування сторожового таймера заборонено. Біт WDE може бути очищений тільки якщо встановлений біт WDTOE. Для заборони дозволеного сторожового таймера необхідно виконати наступну процедуру:

1. В одній операції записати логічну 1 у WDTOE і WDE. Логічна 1 повинна бути записана в WDE навіть якщо цей біт був установлений перед початком операції заборони сторожового таймера.

2. За час наступних чотирьох тактових циклів записати логічний 0 у WDE. Сторожовий таймер буде заборонений.

• **Bits 2..0 - WDP2, WDP1, WDPO: Watch Dog Timer Prescaler 2, 1 and 0 – біти встановлення коефіцієнта попереднього поділу сторожового таймера**

Стан бітів WDP2, WDP1 і WDPO визначають коефіцієнт попереднього поділу тактової частоти дозволеного сторожового таймера. Коефіцієнти і відповідні їм проміжки часу представлені в табл. 3.19.

Таблиця 3.19– Вибір коефіцієнта попереднього поділу тактової частоти сторожового таймера

WDP2	WDP1	WDP0	Тривалість циклу сторожового таймера
0	*0*	*0*	*16 тис. циклів*
0	*0*	*1*	*32 тис. циклів*
0	*1*	*0*	*64 тис. циклів*
0	*1*	*1*	*128 тис. циклів*
1	*0*	*0*	*256 тис. циклів*
1	*0*	*1*	*512 тис. циклів*
1	*1*	*0*	*1024 тис. циклів*
1	*1*	*1*	*2048 тис. циклів*

3.10 Робота з енергонезалежною пам'яттю (EEPROM)

Звертання до EEPROM при читанні/запису

Звертання до EEPROM відбувається за допомогою регістрів, розташованих у просторі I/O. Час звертання при записі складає від 2.5 до 4 мс, у залежності від напруги VCC. Однак існує спеціальна функція, що дозволяє програмі користувача виявляти момент, коли можна починати запис наступного байта - для індикації моменту готовності EEPROM до запису нових даних може бути встановлене спеціальне переривання по завершенню запису EEPROM (EEPROM Write Complete). Випадковий запис у EEPROM запобігається виконанням спеціальної процедури, показаної докладніше в описі регістра керування EEPROM.

Після процедури запису в EEPROM CPU, перш ніж почати виконання наступної команди, зупиняється на два тактових цикли. При читанні EEPROM CPU зупиняється на 4 тактових цикли.

РЕГІСТР АДРЕСИ EEPROM - EEARH, EEARL –
(EEPROM Address Register)

Біт	15	14	13	12	11	10	9	8	
$1F {$3F}	-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
$1E {$3E}	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Читання/Запис	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

За допомогою регістрів адреси EEARH і EEARL визначається адреса в просторі адрес EEPROM ємністю 2 К/4 К. Байти даних EEPROM адресуються лінійно в межах від 0 до

2047/4095. Мікроконтролер ATmega603 оснащений EEPROM з адресним простором 2 К та біт I/O EEAR11 при читанні завжди буде в стані 0.

РЕГІСТР ДАНИХ EEPROM – EEDR – (EEPROM Data Register)

Біт	7	6	5	4	3	2	1	0	
$1D {$3D}	MSB							LSB	EEDR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bits 7..0 - EEDR7..0: EEPROM Data – Дані EEPROM

У процесі виконання операції запису в регістрі EEDR містяться дані, записувані в EEPROM за адресою, що задається регістром EEAR. При читанні регістр EEDR містить дані, що зчитуються з EEPROM за адресою, що визначається EEAR.

РЕГІСТР КЕРУВАННЯ EEPROM – EECR – (EEPROM Control Register)

Біт	7	6	5	4	3	2	1	0	
$1C {$3C}	-	-	-	-	EERIE	EEMWE	EEWE	EERE	EECR
Читання/Запис	R	R	R	R	R	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bits 7..4 - Res: Reserved bits - Зарезервовані біти

Ці біти в мікроконтролерах ATmega603/103 зарезервовані і при зчитуванні завжди покажуть стан 0.

• Bit 3 - EERIE: EEPROM Ready Interrupt Enable - Дозвіл переривання по готовності EEPROM

При встановлених у стан 1 біті EERIE та I-біті регістра SREG дозволяється переривання по готовності EEPROM. При очищеному біті EERIE переривання заборонене. Запит переривання по готовності EEPROM при очищеному біті EEWE генерується безперервно.

• Bit 2 - EEMWE: EEPROM Master Write Enable - Керування дозволом запису EEPROM

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

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

Сигнал дозволу запису EEPROM EEWE (EEPROM Write Enable Signal) є стробом запису в EEPROM. Запис установлених даних по встановленій адресі EEPROM виконується по установці біту EEWE. При цьому біт EEMWE обов'язково повинний бути в стані 1, інакше запис не відбудеться. Процес запису EEPROM виконується наступною процедурою (черговість виконання пунктів 2 і 3 значення не має):

1. Почекати скидання біта EEWE у стан 0.
2. Записати нову адресу в EEAR (при необхідності).
3. Записати нові дані (при необхідності).
4. Встановити у стан 1 біт EEMWE регістра EECR.
5. Протягом чотирьох тактових циклів після установки EEMWE встановити у стан 1 біт EEWE.

Після закінчення часу запису (типове значення 2,5 мс при Vcc=5 В чи 4 мс при Vcc=2,7 В) біт EEWE апаратно очищається. Користувацьке програмне забезпечення може тестувати стан цього біта для визначення моменту скидання його в 0, щоб почати запис наступного байта. Після установки біта EEWE CPU, перш ніж почати виконання наступної команди, зупиняється на два тактових цикли.

• Bit 0 - EERE: EEPROM Read Enable - Дозвіл читання EEPROM

Сигнал дозволу читання EERE (EEPROM Read Enable Signal) є стробом читання EEPROM. Біт EERE повинний бути встановлений по встановленні в регістрі EEAR необхідної

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

Захист EEPROM від руйнування

Вміст EEPROM може бути зруйнований при зниженні напруги VCC до рівня, при якому CPU і EEPROM працюють неправильно. Для вирішення цієї проблеми використовуються ті ж прийоми, що використовуються для забезпечення цілісності даних у EEPROM системних плат.

Руйнування даних EEPROM, при занадто низькій напрузі живлення, відбувається в двох випадках. По-перше, для правильного виконання послідовності операції запису необхідно, щоб напруга живлення була не нижчою за рівень, що гарантує правильне їх виконання. По-друге, саме CPU, при занадто низькій напрузі живлення, може неправильно виконувати команди. Руйнування даних легко уникнути, якщо дотримуватись наступних рекомендацій (досить виконання однієї з трьох):

1. Утримання сигналу скидання в активному (низькому) стані під час зниження напруги харчування. Найкраще це реалізовувати зовнішньою схемою захисту від зниження напруги, яку часто називають Brown-Out Detector (BOD).

2. Утримання ядра AVR мікроконтролера в Power Down Sleep режимі в період зниження напруги VCC. Це запобіжить неправильному декодуванню і виконанню команд CPU, що захистить регістри EEPROM від випадкових записів.

3. Збереження констант у Flash пам’яті, якщо немає необхідності змінювати їх програмно. Flash пам’ять не змінюється CPU і, отже, не може бути пошкоджена.

3.11 Послідовний периферійний інтерфейс – SPI (Serial Peripheral Interface)

Послідовний периферійний інтерфейс (SPI) забезпечує високошвидкісний синхронний обмін даними між мікроконтролерами ATmega603/103 і периферійними пристроями чи між декількома мікроконтролерами ATmega603/103.

Основні характеристики SPI інтерфейсу:

- Повнодуплексний 3-провідний синхронний обмін даними.
- Режим роботи ведучий/ведений.
- Обмін даними з переданими першими старшим чи молодшим бітами.
- Чотири програмувальні швидкості обміну даними.
- Флаг переривання по закінченню передачі.
- Активація з Idle режиму (тільки в режимі веденого).

З’єднання між ведучим і веденим CPU, що використовують SPI інтерфейс, показані на рис. 3.39. Вивід PB1(SCK) є виходом тактового сигналу ведучого мікроконтролера і входом тактового сигналу веденого. По запису ведучим CPU даних у SPI регістр починає працювати тактовий генератор SPI і записані дані зсуваються через вивід виходу PB2(MOSI) ведучого мікроконтролера на вивід входу PB2(MOSI) веденого мікроконтролера. Після зсуву одного байта тактовий генератор SPI зупиняється, установлюючи флаг закінчення передачі (SPIF). Якщо в регістрі SPCR буде встановлений біт дозволу переривання SPI (SPIE), то відбудеться запит переривання. Вхід вибору веденого PB0($\overline{SS}$), для вибору індивідуального SPI пристрою в якості веденого, встановлюється на низький рівень. При установці високого рівня на виводі PB0($\overline{SS}$) порт SPI деактивується і вивід PB2(MOSI) може бути використаний як вивід входу. Режим ведучий/ведений може бути встановлений і програмним способом установкою чи очищенням біта MSTR у регістрі керування SPI.

Два регістри зсуву ведучого і веденого мікроконтролерів можна розглядати як один рознесений 16-розрядний циклічний регістр зсуву. Див. рис. 3.38. При зсуві даних з ведучого мікроконтролера у ведений одночасно відбувається зсув даних з веденого мікроконтролера у ведучий, тобто протягом одного циклу зсуву відбувається обмін даними між ведучим і веденим мікроконтролерами.

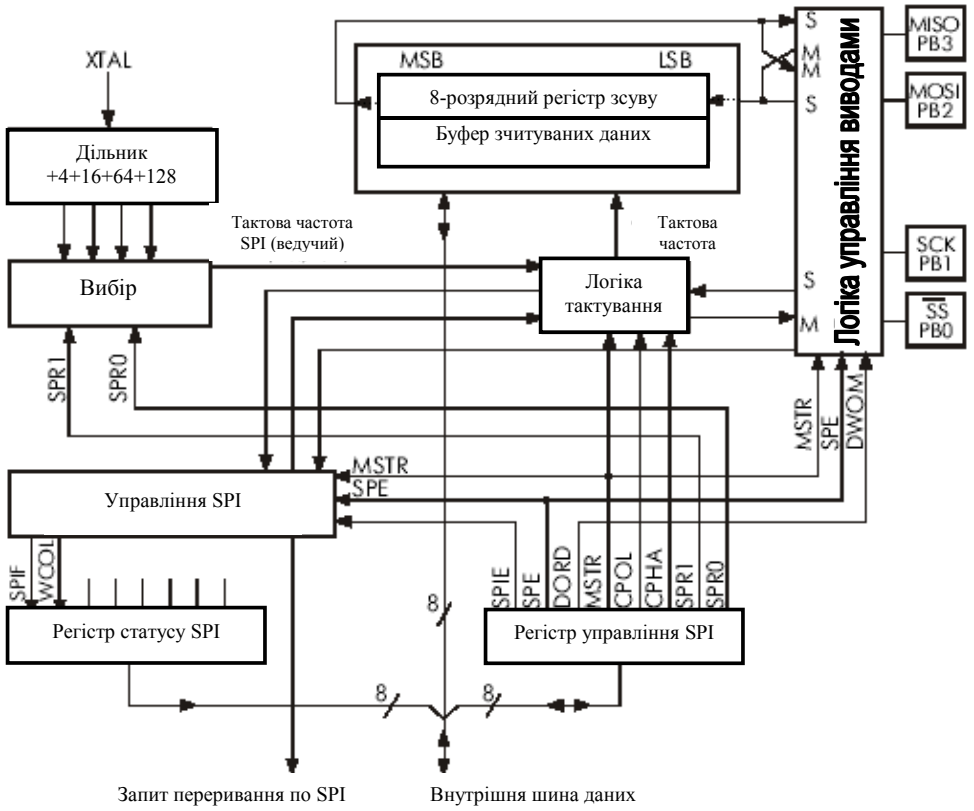


Рисунок 3.38 – Блок-схема SPI

У системі організоване одиночне буферування передаючої сторони і подвійне буферування на прийомній стороні. Це означає, що передані символи не можуть бути записані в регістр даних SPI перш, ніж буде цілком завершений цикл зсуву.

З іншого боку, при прийомі даних прийнятий символ повинний бути зчитаним з регістра даних SPI перш, ніж буде завершений прийом наступного символу, у противному випадку попередній символ буде загублений.

При дозволеному SPI напрямку даних виводів MOSI, MISO, SCK і $\overline{SS}$ настроюються у відповідності з наступною таблицею:

Таблиця 3.20– Настройка виводів SPI

Вивід	Напрям, ведучий SPI	Напрям, ведений SPI
MOSI	Визначається користувачем	Вхід
MISO	Вхід	Визначається користувачем
SCK	Визначається користувачем	Вхід
SS	Визначається користувачем	Вхід

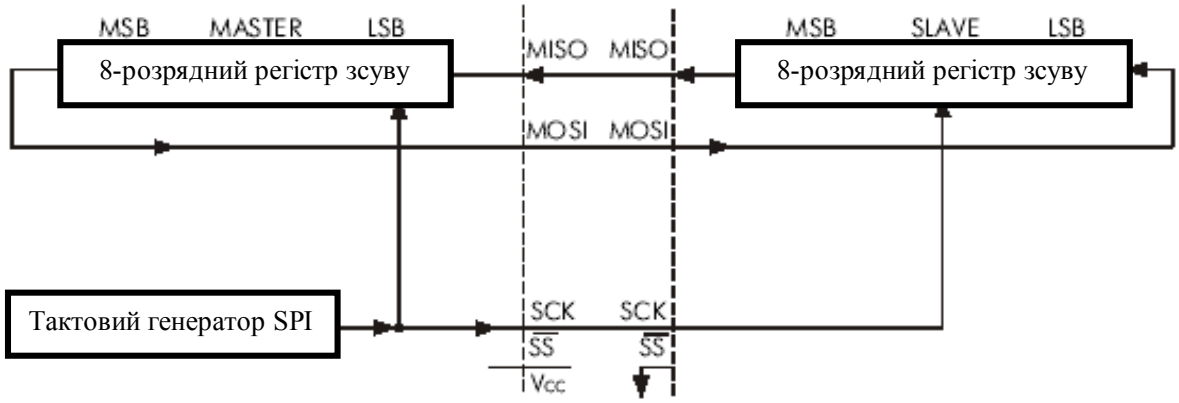


Рисунок 3.39 – з'єднання ведучого та веденого SPI

ФУНКЦІОНУВАННЯ ВХОДУ $\overline{SS}$

При роботі SPI ведучим (біт MSTR регістра SPCR встановлений), користувач має можливість встановити напрямок роботи виводу $\overline{SS}$. Якщо вивід $\overline{SS}$ сконфігурований як вихід, то вивід є виводом загального призначення і він не активується системою SPI. Якщо ж вивід $\overline{SS}$ сконфігурований як вхід, то для забезпечення роботи ведучого SPI він повинен утримуватися на високому рівні. Якщо, у режимі ведучого, вивід $\overline{SS}$ є входом і зовнішньою периферійною схемою на нього поданий низький рівень, то SPI сприйме його як звертання іншого ведучого SPI до себе як до веденого. Щоб уникнути конфліктної ситуації на шині, система SPI виконує наступні дії:

1. Біт MSTR у регістрі SPCR очищається і SPI система стає веденою. Результатом цього є те, що MOSI і SCK виводи стають входами.

2. Встановлюється флаг SPIF регістра SPSR і, якщо дозволене переривання SPI, почнеться виконання підпрограми обробки переривання. Таким чином, коли керований перериванням передавальний SPI використовується у ведучому режимі, і існує імовірність подачі на вивід $\overline{SS}$ керуючого сигналу низького рівня, переривання повинне завжди перевіряти чи встановлений ще біт MSTR. Якщо ж біт MSTR був очищений вибором режиму веденого, то він повинен бути встановлений користувачем.

Якщо ж SPI працює в режимі веденого, то вивід $\overline{SS}$ постійно працює на вхід. Якщо на вивід $\overline{SS}$ поданий низький рівень, то SPI активується і MISO, якщо це визначено користувачем, стає виходом. Всі інші виводи є входами. Якщо вивід $\overline{SS}$ утримується на високому рівні, то всі виводи є входами, SPI пасивний, що означає, що він не буде одержувати вхідних даних.

Існує чотири варіанти комбінації фази і полярності SCK щодо послідовних даних, обумовлені керуючими бітами CPHA і CPOL. Формати передачі даних SPI показані на рис. 3.40 і 3.41.

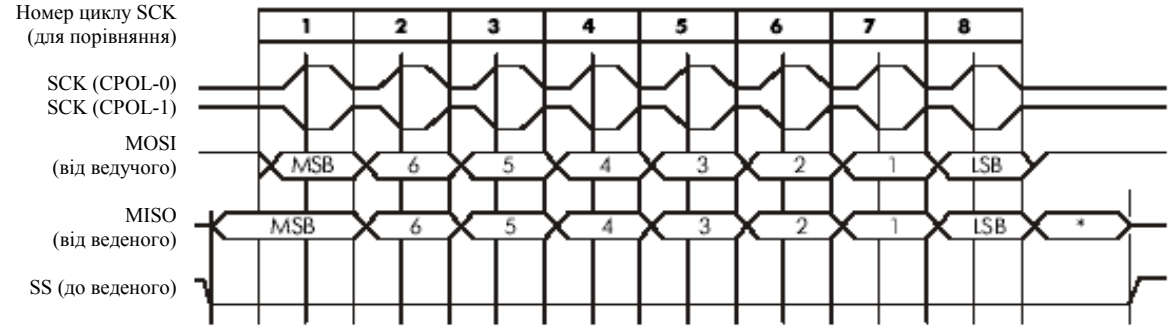


Рисунок 3.40 - Формат передачі SPI при CPHA=0

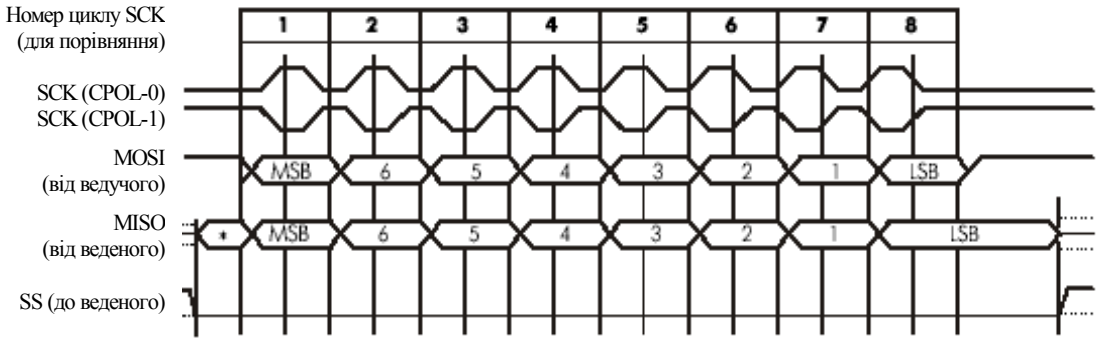


Рисунок 3.41 - Формат передачі SPI при CPHA=1

РЕГІСТР КЕРУВАННЯ SPI – SPCR – (Control Register)

Бит	7	6	5	4	3	2	1	0	
$0D ($2D)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

• Bit 7 - SPIE: SPI Interrupt Enable - Дозвіл переривання SPI

Установка біта SPIE у стан 1 приводить до установки біта SPIF регістра SPSR і, при дозволі глобального переривання, до виконання переривання SPI.

• Bit 6 - SPE: SPI Enable - Дозвіл SPI

Установка біта SPE у стан 1 дозволяє підключення $\overline{SS}$, MOSI, MISO і SCK до виводів PB4, PB5, PB6 і PB7.

• Bit 5 - DORD: Data Order - Порядок даних

При встановленому в стан 1 біті DORD передача слова даних відбувається LSB вперед. При очищеному біті DORD першим передається MSB слова даних.

• Bit 4 - MSTR: Master/Slave Select - Вибір режиму ведучий/ведений

При встановленому в стан 1 біті MSTR SPI працює у ведучому режимі і при очищеному біті у веденому режимі. Якщо $\overline{SS}$ сконфігурований як вхід і на нього подано низький рівень при встановленому MSTR, то MSTR буде скинутий і буде встановлено біт SPIF у регістрі SPSR. Щоб знову дозволити ведучий режим SPI, користувач повинен встановити MSTR.

• Bit 3 - CPOL: Clock Polarity - Полярність тактового сигналу

SCK у режимі очікування знаходиться на високому рівні при встановленому в стан 1 біті CPOL і на низькому рівні при скинутому біті CPOL. Див. рис. 3.40 і 3.41.

• Bit 2 - CPHA : Clock Phase - Фаза тактового сигналу

Робота цього біту відображена на рис. 3.40 і 3.41.

• Bits 1, 0 - SPR1, SPRO: SP1 Clock Rate Select 1 and 0 - Вибір частоти тактового сигналу, біти 1 і 0

Цих два біти керують частотою тактового сигналу приладу, що працює у ведучому режимі. У веденому режимі стани бітів впливу не чинять. Стани бітів і встановлюваний коефіцієнт розподілу частоти fCL показані в таблиці:

Таблиця 3.21- ***Співвідношення між SCK і частотою генератора***

SPR1	SPR0	Тактова частота SCK
0	0	fα/4
0	1	fα/16
1	0	fα/64
1	1	fα/128

РЕГІСТР СТАТУСУ SPI – SPSR – (Status Register)

Біт	7	6	5	4	3	2	1	0	
$0E {$2E}	SPIF	WCOL	-	-	-	-	-	-	SPSR
Читання/Запис	R	R	R	R	R	R	R	R	
Початкове значення	0	0	0	0	0	0	0	0	

• Bit 7 - SPIF: SPI Interrupt Flag - Флаг переривання по SPI

По завершенні обміну послідовними даними біт SPIF встановлюється в стан 1 і, якщо біт SPIE у регістрі SPCR встановлений і дозволене глобальне переривання, генерується сигнал переривання. Біт SPIF очищається апаратно при виконанні підпрограми обробки відповідного вектора переривання. Біт SPIF може бути очищений також при першому зчитуванні стану регістра статусу SPI, із встановленим бітом SPIF, з наступним звертанням до регістра даних SPI (SPDR).

• Bit 6 - WCOL: Write Collision Flag - Флаг помилки при записі

Біт WCOL встановлюється в стан 1, якщо в процесі передачі даних виконувався запис у регістр даних (SPDR). Читання вмісту регістра даних, як і запис у нього, виконані під час пересилання даних, можуть привести до невірного результату. Біт WCOL (і біт SPIF) апаратно очищаються (скидаються в стан 0) при першому зчитуванні регістра статусу SPI, із установленим WCOL, з наступним звертанням до регістра даних SPI (SPDR).

• Bit 5..0 - Res: Reserved bits - Зарезервовані біти

Ці біти в мікроконтролерах Atmega603/103 зарезервовані і при зчитуванні завжди покажуть стан 0.

РЕГІСТР ДАНИХ SPI – SPDR – (SPI Data Register)

Біт	7	6	5	4	3	2	1	0	
$0F {$2F}	MSB							LSB	SPDR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

Регістр даних SPI представляє собою регістр із можливістю читання/запису і призначений для пересилання даних між регістровим файлом і регістром зсуву SPI. Запис у регістр SPDR ініціює передачу даних, зчитування регістра приводить до читання регістра зсуву прийому.

3.12 UART – універсальний асинхронний прийомопередавач

Мікроконтролери ATmega603/103 оснащені повнодуплексними й універсальними прийомопередавачами (UART). Їхні основні можливості наступні:

- Генератор забезпечує будь-яку швидкість передачі інформації в бодах.
- Висока швидкість передачі при низькій частоті XTAL.
- 8-розрядний чи 9-розрядний формати даних.
- Фільтрація шуму.
- Виявлення переповнення.
- Виявлення помилок формування кадрів.
- Детектування біту помилкового старту.
- Три окремих переривання: по завершенні передачі (TX Complete), по пустому регістру даних, що передаються (TX Data Register Empty), та по завершенні прийому (RX Complete).

Передача даних

Блок-схема передавача UART показана на рис. 3.42.

Передача даних ініціюється записом переданих даних у регістр даних I/O UART (UDR). Дані пересилаються з UDR у регістр зсуву передачі в наступних випадках:

• Новий символ записаний у UDR після того, як був виведений з регістра стоповий біт попереднього символу. Регістр зсуву завантажується негайно.

• Новий символ записаний у UDR перш, ніж був виведений стоповий біт попереднього символу. Регістр зсуву завантажується після виходу стопового біту переданого символу, що знаходився в регістрі зсуву.

Якщо із 10 (11)-розрядного регістра зсуву передачі виведена вся інформація (регістр зсуву передачі порожній) дані з UDR пересилаються в регістр зсуву. У цей час встановлюється біт UDRE (UART Data Register Empty) регістра статусу UART (USR). При встановленому в стан 1 біті UDRE UART готовий прийняти наступний символ. Запис в UDR очищає біт UDRE. У той самий час, коли дані пересилаються з UDR у 10 (11)-розрядний регістр зсуву, біт 0 регістра зсуву скидається в стан 0 (стан 0 - стартовий біт) а біт 9 чи 10 встановлюється в стан 1 (стан 1 - стоповий біт). Якщо в регістрі керування UART (UCR) установлений біт CHR9 (тобто обраний режим 9-розрядного слова даних), то біт TXB8 регістра UCR пересилається в біт 9 регістра зсуву передачі.

Відразу після пересилання даних у регістр зсуву тактом бод-генератора стартовий біт зсувається на вивід TXD. За ним слідує LSB даних. Коли буде виданий стоповий біт, регістр зсуву завантажується новою порцією даних, якщо вона була записана в UDR під час передачі. У процесі завантаження біт UDRE знаходиться у встановленому стані. Якщо ж нові дані не будуть завантажені в UDR до видачі стопового біта, флаг UDRE залишається встановленим. У такому випадку, після того, як стоповий біт буде присутній на виводі TXD протягом одного такту, у регістрі статусу UART (USR) встановлюється флаг завершення передачі TXC (TX Complete Flag).

Встановлений у стан 1 біт TXEN регістра UCR дозволяє передачу UART. При очищеному біті TXEN (скинутому в стан 0) вивід PE1 може бути використаний як вивід I/O загального призначення. При встановленому біті TXEN передавач UART підключається до PE1 і використовує його в якості виводу виходa, незалежно від установки біта DDE1 в DDRE.

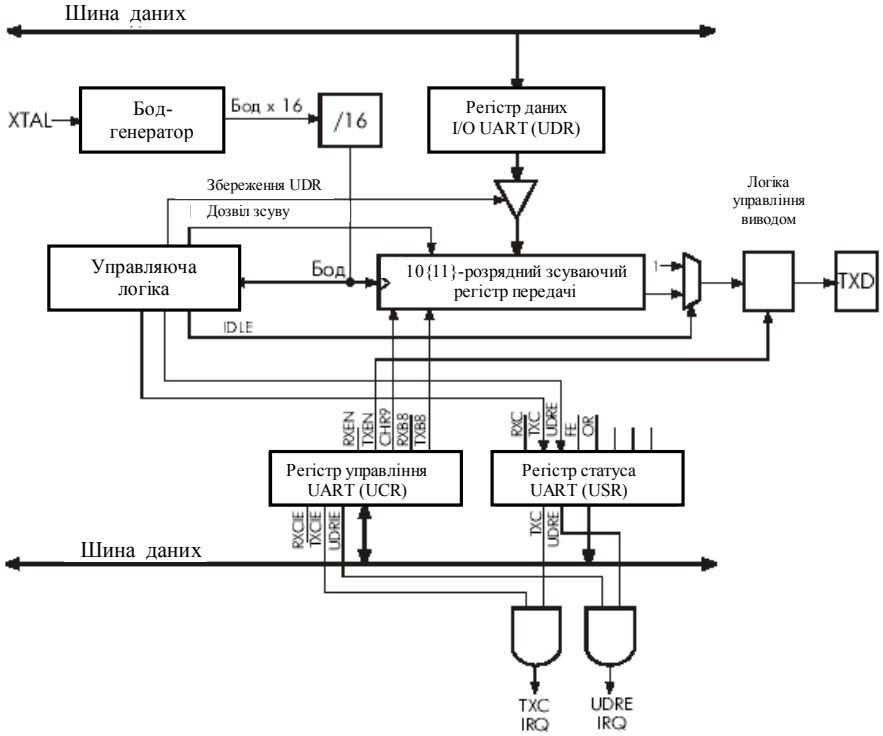


Рисунок 3.42 – Передавач UART

Прийом даних

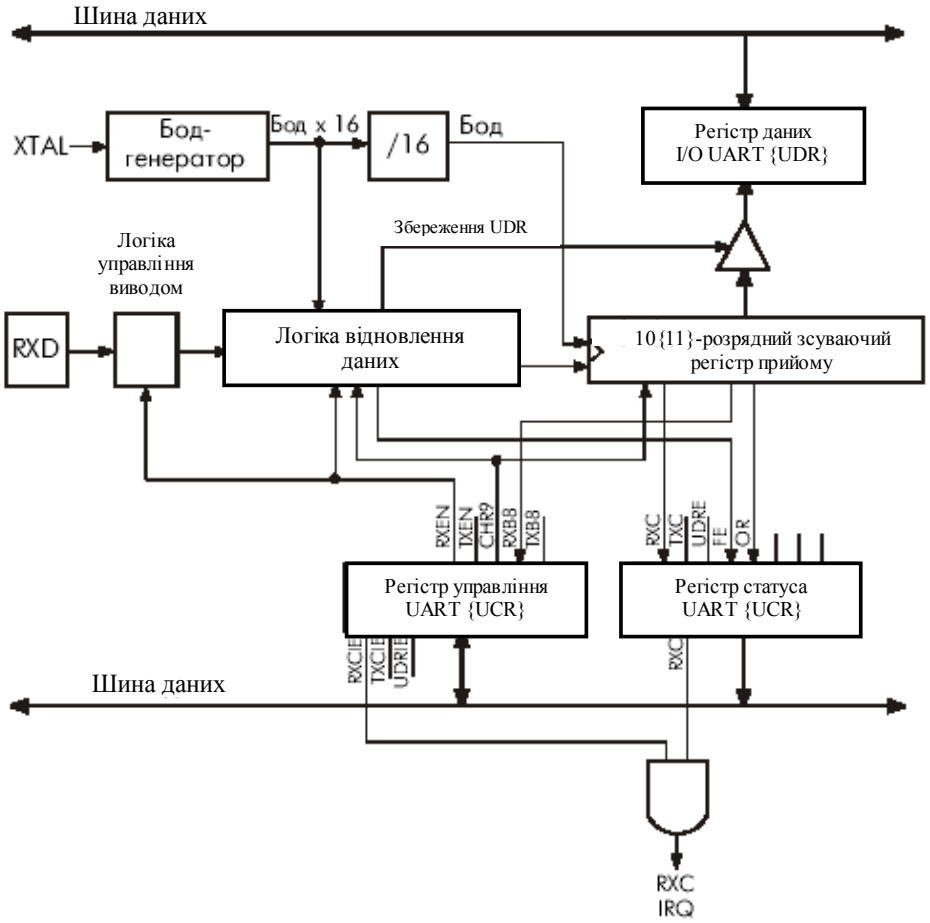


Рисунок 3.43 – Приймач UART

Логіка відновлення даних робить вибірку станів вивода RXD з частотою в 16 разів більшою, ніж частота бодів. При знаходженні лінії в пасивному стані одиночна вибірка нульового логічного рівня буде інтерпретуватися як падаючий фронт стартового біта і буде запущена послідовність детектування стартового біта. Вважається, що перша вибірка знайшла перший нульовий логічний рівень ймовірного стартового біта. На вибірках 8, 9 і 10 приймач знову тестує висновок RXD на зміну логічних станів. Якщо дві чи більш з цих трьох вибірок знайдуть логічні 1, то даний ймовірний стартовий біт відкидається як шумовий сплеск і приймач почне виявляти й аналізувати наступні переходи з 1 в 0.

Якщо ж був виявлений дійсний стартовий біт, то починає вироблятися вибірка наступних за стартовим бітом інформаційних бітів. Ці біти також тестуються на вибірках 8, 9 і 10. Логічний стан біта приймається по двом і більше (із трьох) однаковим станам вибірок. Всі біти вводяться в регістр зсуву приймача з тим значенням, яке було визначено тестуванням вибірок. Тестування вибірок бітів символів, що приймаються, показано на рис. 3.44.

Рисунок 3.44 – Тестування вибірок даних, що приймаються

При поступленні стопового біта необхідно, щоб не менше двох вибірок із трьох підтвердили прийом стопового біту (показали високий рівень). Якщо ж дві чи більше вибірок покажуть стан 0, то при пересилці прийнятого байта в UDR, в регістрі статусу UART (USR) встановлюється біт помилки кадру FE (Framing Error). Для виявлення помилки кадру

користувач перед читанням регістру UDR повинен перевіряти стан біта FE. Флаг FE очищується при зчитуванні вмісту регістра даних UART(UDR). Незалежно від того, прийнятий правильний стоповий біт чи ні, дані пересилаються в регістр UDR і встановлюється флаг RXC в регістрі статусу UART (USR). Регістр UDR фактично є двома фізично окремими регістрами, один з яких служить для передачі даних і другий для прийому. При зчитуванні UDR звертання ведеться до регістру прийому даних, при запису звертання ведеться до регістру передачі. Якщо обраний режим обміну 9-розрядними словами даних (встановлений біт CHR9 регістру UCR), при пересилці даних в UDR біт RXB8 регістру UCR завантажується в біт 9 регістра зсуву передачі. Якщо після отримання символу до регістру UDR не було звертання, починаючи з останнього прийому, в регістрі UCR встановлюється флаг переповнення (OR). Це означає, що нові дані, які пересилаються в регістр зсуву, не можуть бути передані в UDR і втрачені. Біт OR буферований і доступний тоді, коли в UDR читається байт достовірних даних. Користувачу, для виявлення переповнення, необхідно завжди перевіряти флаг OR після зчитування вмісту регістра UDR.

При очищеному (скинутому в логічний стан 0) біті RXEN регістра UCR приймач заборонений. Це означає, що вивід PE0 може використовуватися в якості виводу I/O загального призначення. При встановленому біті RXEN, приймач UART під'єднується до виводу PE0, який працює як вивід входу, незалежно від установки біта DDE0 в DDRE.

При установці UART виводу PE0 на роботу в якості входу, біт PORTE0 може використовуватися для керування навантажувальним резистором виводу.

РЕГІСТР ДАНИХ UART – UDR – (UART I/O Data Register)

Бит	7	6	5	4	3	2	1	0	
$0D ($2D)	MSB							LSB	UDR
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

У дійсності регістр UDR є двома фізично розділеними регістрами - регістром передачі даних і регістром прийому даних, що використовують ті самі адреси I/O. При записі в регістр запис здійснюється в регістр передачі даних UART, при читанні відбувається читання вмісту регістра прийому даних UART.

РЕГІСТР СТАТУСУ UART – USR – (UART Status Register)

Біт	7	6	5	4	3	2	1	0	
$0B {$2B}	RXC	TXC	UDRE	FE	DOR	-	-	-	USR
Читання/Запис	R	R/W	R	R	R	R	R	R	
Початкове значення	0	0	0	0	0	0	0	0	

Регістр USR забезпечує тільки читання інформації про стан UART.

• Bit 7 - RXC: UART Receive Complete - Прийом завершений

Даний біт встановлюється в стан 1 при пересиланні прийнятого символу із регістра зсуву прийому в UDR. Біт встановлюється незалежно від відсутності чи наявності помилок прийому кадру. При встановленому в UCR біті RXCIE і встановленому біті RXC виконується переривання по завершенню прийому UART. Біт RXC очищається при зчитуванні UDR. При прийомі даних ініційованому перериванням, підпрограма обробки переривання по завершенню прийому UART повинна зчитати UDR, для того, щоб очистити RXC, інакше по закінченні підпрограми обробки переривання відбудеться нове переривання.

• Bit 6 - TXC: UART Transmit Complete - Передача завершена

Даний біт встановлюється в стан 1, коли весь символ (включаючи стоповий біт) виведений із регістра зсуву передачі і в UDR не записані нові дані. Цей флаг використовується при напівдуплексному зв'язному інтерфейсі, коли устаткування передачі повинне установити режим прийому і звільнити комунікаційну шину відразу після завершення передачі.

При встановленому в регістрі UCR біті TXCIE установка TXC приведе до виконання переривання по завершенню передачі UART. Флаг TXC очищається апаратно при виконанні обробки відповідного вектора переривання. Очистити біт TXC можна записом у біт логічної 1.

• Bit 5 - UDRE: UART Data Register Empty - Регістр даних порожній

Даний біт встановлюється в стан 1, коли символ, записаний в UDR, пересилається в регістр зсуву передачі. Установка цього біта означає, що передавач готовий до одержання нового символу для передачі.

Коли біт UDRIE в UCR установлений, до тих пір, поки встановлений UDRE, виконується переривання по завершенню передачі UART. Біт UDRE очищається при записі в UDR. При прийомі даних ініційованому перериванням, підпрограма обробки переривання по порожньому регістрі даних UART повинна зчитати UDR, для того, щоб очистити UDRE, інакше по закінченні підпрограми переривання відбудеться нове переривання. Під час скидання біт UDRE встановлюється в стан 1 з тим, щоб індиціювати готовність передавача.

• Bit 4 - FE: Framing Error - Помилка кадру

Даний біт встановлюється в стан 1 при виявленні умов помилкового прийому кадру, тобто коли стоповий біт вхідного символу в стані 0. Біт FE очищається при прийомі стопового біта з логічним рівнем 1.

• Bit 3 - DOR: Data QverRun - Переповнення даних

Біт DOR встановлюється в стан 1 при виявленні умов переповнення, тобто коли символ, що вже знаходиться в регістрі UDR, не зчитаний перед пересиланням нового символу з регістра зсуву прийому. Біт DOR буферований, що означає, що він буде залишатися встановленим, поки не будуть зчитані правильні дані з UDR. Біт DOR очищається (скидається в 0), коли дані прийняті і переслані в UDR.

• Bits 2..0 - Res: Reserved bits - Зарезервовані біти

Ці біти в мікроконтролерах Atmega603/103 зарезервовані і при зчитуванні завжди покажуть стан 0.

РЕГІСТР КЕРУВАННЯ UART – UCR – (UART Control Register)

Біт	7	6	5	4	3	2	1	0	
$0A {$2A}	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	UCR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bit 7 - RXCIE: RX Complete Interrupt Enable - Дозвіл переривання по завершенню прийому

При встановленому в стан 1 біті RXCIE і встановленому дозволі глобального переривання установка біта RXC у регістрі USR приведе до виконання переривання по завершенню прийому.

• Bit 6 - TXCIE: TX Complete Interrupt Enable - Дозвіл переривання по завершенню передачі

При встановленому в стан 1 біті TXCIE і встановленому дозволі глобального переривання установка біта TXC у регістрі USR приведе до виконання переривання по завершенню передачі.

• Bit 5 - UDRIE: UART Data Register Empty Interrupt Enable - Дозвіл переривання по порожньому регістрі даних

При встановленому в стан 1 біті UDRIE і встановленому дозволі глобального переривання установка бита UDRE у регістрі USR приведе до виконання переривання по порожньому регістрі даних UART.

• Bit 4 - RXEN: Receiver Enable - Дозвіл приймача

Встановлений у стан 1 біт RXEN дозволяє приймач UART. Якщо приймач заборонений, то флаги статусу TXC, DOR і FE установити неможливо. Якщо ці флаги встановлені, то очищення біта RXEN не призведе до очищення цих флагів.

• Bit 3 - TXEN: Transmitter Enable - Дозвіл передавача

Встановлений у стан 1 біт TXEN дозволяє передавач UART. При забороні передавача під час передачі символу, передавач не буде заблокований перш, ніж будуть цілком передані символ в регістрі зсуву плюс будь-який, що знаходиться в UDR, наступний символ.

• Bit 2 - CHR9: 9 Bit Characters - Режим 9-розрядних символів

При встановленому в стан 1 біті CHR9 передаються і приймаються 9-розрядні символи плюс стартовий і стоповий біти. Дев'яті біти читаються і записуються з використанням бітів RXB8 і TXB8 (відповідно) регістра UCR. Дев'ятий біт даних може використовуватися як додатковий стоповий біт чи біт контролю парності.

• Bit 1 - RXB8: Receive Potq Bit 8 - Прийом 8-розрядних даних

При встановленому в стан 1 біті CHR9 біт RXB8 є дев'ятим бітом даних прийнятого символу.

• Bit 0 - TXB8: Transmit Data Bit 8 - Передача 8-розрядних даних

При встановленому в стан 1 біті CHR9 біт TXB8 є дев'ятим бітом даних переданого символу.

БОД-ГЕНЕРАТОР (Baud Rate Generator)

Бод-генератор являє собою дільник, що генерує імпульси передачі з частотою, що визначається за виразом:

$$BAUD = \frac{f_{CX}}{16(UBRR + 1)},$$

де:

- BAUD - частота в бодах,
- f_{CK} - частота кварцового генератора,
- UBRR - вміст регістра UBRR (Baud Rate register = 0 - 255).

При використанні стандартних кварцових кристалів, найбільш часто використовувані швидкості передачі в бодах можуть бути отримані установками UBRR, представленими в Табл. 3.24. При установках UBRR, зазначених у таблиці, реальні швидкості в бодах будуть мати відмінності, менші 2% від стандартних швидкостей.

Таблиця 3.22 – установки UBRR для одержання стандартних швидкостей передачі

Швид-кість (бод)	1 МГц	помилка, %	1,8432 МГц	помилка, %	2 МГц	помилка, %	2,4576 МГц	помилка, %
2400	UBRR= 25	0,2	UBRR= 47	0,0	UBRR= 51	0,2	UBRR= 63	0,0
4800	UBRR= 12	0,2	UBRR= 23	0,0	UBRR= 25	0,2	UBRR= 31	0,0
9600	UBRR= 6	7,5	UBRR= 11	0,0	UBRR= 12	0,2	UBRR= 15	0,0
14400	UBRR= 3	7,8	UBRR= 7	0,0	UBRR= 8	3,7	UBRR= 10	3,1
19200	UBRR= 2	7,8	UBRR= 5	0,0	UBRR= 6	7,5	UBRR= 7	0,0
28800	UBRR= 1	7,8	UBRR= 3	0,0	UBRR= 3	7,8	UBRR= 4	6,3
38400	UBRR= 1	22,9	UBRR= 2	0,0	UBRR= 2	7,8	UBRR= 3	0,0
57600	UBRR= 0	7,8	UBRR= 1	0,0	UBRR= 1	7,8	UBRR= 2	12,5
76800	UBRR= 0	22,9	UBRR= 1	33,3	UBRR= 1	22,9	UBRR= 1	0,0
115200	UBRR= 0	84,3	UBRR= 0	0,0	UBRR= 0	7,8	UBRR= 0	25,0

Швид-кість (бод)	3,2768 МГц	пом., %	3,6864 МГц	поми., %	4 МГц	помилка, %	4,608 МГц	помилка, %
2400	UBRR= 84	0,4	UBRR= 95	0,0	UBRR= 103	0,2	UBRR= 119	0,0
4800	UBRR= 42	0,8	UBRR= 47	0,0	UBRR= 51	0,2	UBRR= 59	0,0
9600	UBRR= 20	1,6	UBRR= 23	0,0	UBRR= 25	0,2	UBRR= 29	0,0
14400	UBRR= 13	1,6	UBRR= 15	0,0	UBRR= 16	2,1	UBRR= 19	0,0
19200	UBRR= 10	3,1	UBRR= 11	0,0	UBRR= 12	0,2	UBRR= 14	0,0

Продовження таблиці 3.22

28800	UBRR= 6	1,6	UBRR= 7	0,0	UBRR= 8	3,7	UBRR= 9	0,0
38400	UBRR= 4	6,3	UBRR= 5	0,0	UBRR= 6	7,5	UBRR= 7	6,7
57600	UBRR= 3	12,5	UBRR= 3	0,0	UBRR= 3	7,8	UBRR= 4	0,0
76800	UBRR= 2	12,5	UBRR= 2	0,0	UBRR= 2	7,8	UBRR= 3	6,7
115200	UBRR= 1	12,5	UBRR= 1	0,0	UBRR= 1	7,8	UBRR= 2	20,0

Швид-кість (бод)	7,3728 МГц	помилка, %	8 МГц	помилка, %	9,216 МГц	помилка, %	11,059 МГц	помилка, %
2400	UBRR= 191	0,0	UBRR= 207	0,2	UBRR= 239	0,0	UBRR= 287	-
4800	UBRR= 95	0,0	UBRR= 103	0,2	UBRR= 119	0,0	UBRR= 143	0,0
9600	UBRR= 47	0,0	UBRR= 51	0,2	UBRR= 59	0,0	UBRR= 71	0,0
14400	UBRR= 31	0,0	UBRR= 34	0,8	UBRR= 39	0,0	UBRR= 47	0,0
19200	UBRR= 23	0,0	UBRR= 25	0,2	UBRR= 29	0,0	UBRR= 35	0,0
28800	UBRR= 15	0,0	UBRR= 16	2,1	UBRR= 19	0,0	UBRR= 23	0,0
38400	UBRR= 11	0,0	UBRR= 12	0,2	UBRR= 14	0,0	UBRR= 17	0,0
57600	UBRR= 7	0,0	UBRR= 8	3,7	UBRR= 9	0,0	UBRR= 11	0,0
76800	UBRR= 5	0,0	UBRR= 6	7,5	UBRR= 7	6,7	UBRR= 8	0,0
115200	UBRR= 3	0,0	UBRR= 3	7, 8	UBRR= 4	0,0	UBRR= 5	0,0

РЕГІСТР БОД-ГЕНЕРАТОРА UART – UBRR – (UART Baud Rate Register)

Біт	7	6	5	4	3	2	1	0	
$09 {$29}	MSB							LSB	UBRR
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

Регістр UBRR є 8-розрядним регістром, з можливістю читання/запису, що визначає швидкість UART відповідно до установок табл. 3.22.

3.13 Аналоговий компаратор

Аналоговий компаратор порівнює рівні напруг на позитивному виводі PE2 (AC+) і негативному виводі PE3 (AC-). При напрузі на позитивному виводі PE2 (AC+) більшому, ніж напруга на негативному виводі PE3 (AC-), вихід аналогового компаратора ACO встановлюється в стан 1. Вихід компаратора може бути використаний для керування входом захоплення таймера/лічильника 1. Крім того, компаратор може формувати свій запит переривання. Користувач може задати формування запиту на переривання по наростаючому чи падаючому фронту чи по переключенню. Блок-схема аналогового компаратора показана на рис. 3.43.

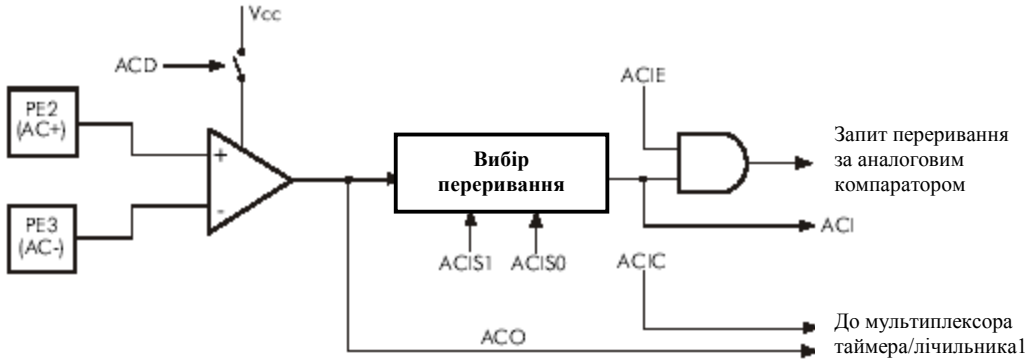


Рисунок 3.45 – Блок-схема аналогового компаратора

РЕГІСТР СТАТУСУ І КЕРУВАННЯ АНАЛОГОВОГО КОМПАРАТОРА – ACSR – (The Analog Comparator Control and Status Register)

Біт	7	6	5	4	3	2	1	0	
$08 {$28}	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Читання/Запис	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bit 7 - ACP: Analog Comparator Disable - Заборона аналогового компаратора

При встановленому в стан 1 біті ACD аналоговий компаратор заборонений. Для вимикання аналогового компаратора установку донного біта можна робити в будь-який час. Відключення аналогового компаратора дозволяє знизити споживання в активному і Idle режимах. При зміні стану біта ACD необхідно забороняти переривання по аналоговому компараторі очищенням біта ACIE у регістрі ACSR. У противному випадку при зміні стану біта ACD може відбутися переривання.

• Bit 6 - Res: Reserved bit - Зарезервований біт

Цей біт у мікроконтролерах Atmega603/103 зарезервований і при зчитуванні завжди покаже стан 0.

• Bit 5 - ACO: Analog Comparator Output - Вихід аналогового компаратора

Біт ACO зв’язаний безпосередньо з виходом компаратора.

• Bit 4 - ACI: Analog Comparator Interrupt Flag - Флаг переривання по аналоговому компаратору

Даний біт встановлюється в стан 1 у випадку формування компаратором переривання, обумовленого ACIS1 і ACISO. Підпрограма обробки переривання по аналоговому компаратору буде виконуватися при встановленому біті ACIE і встановленому біті глобального переривання в регістрі SREG. Біт ACI очищається апаратно при виконанні відповідного вектора обробки переривання. Біт ACI можна очистити, також, записом у флаг логічної 1.

Відзначимо однак, що при модифікації інших бітів регістра ACSR командами SBI чи CBI біт ACI буде очищений, якщо він був установлений перед цими операціями.

• Bit 3 - ACIE: Analog Comparator Interrupt Enable - Дозвіл переривання по аналоговому компараторі

При встановленому біті ACIE і встановленому біті глобального переривання регістра SREG активується переривання по аналоговому компараторі. При скинутому біті ACIE переривання заборонене.

• Bit 2 - ACIC: Analog Comparator Input Capture enable - Дозвіл входу захоплення аналогового компаратора

Встановлений у стан 1 біт ACIC дозволяє спрацьовування функції захоплення входу таймера/лічильника 1 по переключенню аналогового компаратора. У цьому випадку вихід аналогового компаратора приєднується безпосередньо до вхідного ланцюга логіки захоплення входу, що забезпечує використання функцій пониження шуму і вибору виду спрацьовування переривання по захопленню входу таймера/лічильника 1. При очищеному біті ACIC з’єднання немає. Для запуску переривання по захопленню входу таймера/лічильника 1 біт TICIE1 у регістрі масок переривань TIMSK повинний бути встановлений у стан 1.

• Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select - Вибір режиму переривання по аналоговому компаратору

Ці біти визначають характер події компаратора, при якому запускається переривання за аналоговим компаратором. Варіанти установок показані в Таблиці 3.23.

Таблиця 3.23 – Установки бітів ACIS1, ACIS0

ACIS1	ACIS0	Режим переривання
0	0	Переривання по переключенню виходу компаратора
0	1	Зарезервовано

1	0	Переривання по падаючому фронті на виході компаратора
1	1	Переривання по наростаючому фронті на виході компаратора

Примітка: При зміні стану бітів ACIS1/ACIS0 переривання по аналоговому компаратору повинне бути заборонено очищенням біта дозволу переривання в регістрі ACSR. У противному випадку при зміні стану бітів може відбутися переривання.

3.14 Аналого-цифровий перетворювач

Основні технічні характеристики:

- Роздільне здатність 10 розрядів.
- Точність +1/2 LSB.
- Час перетворення 70..280 мс.
- 8 мультиплексованих каналів входу.
- Режими циклічного й одноразового перетворення.
- Переривання по завершенню ADC перетворення.
- Пристрій зменшення шумів Sleep режиму.

Мікроконтролери ATmega603/103 оснащені 10-розрядними ADC послідовного наближення. ADC приєднаний до 8-канального аналогового мультиплексора, що дозволяє використовувати будь-який вивід порту F як вхід ADC. ADC містить підсилювач вибірки/збереження, що утримує напругу входу ADC під час перетворення на незмінному рівні. Блок-схема ADC представлена на рис. 3.46. Для живлення ADC використовуються два окремих виводи: AVCC та AGND. Вивід AGND повинен бути приєднаний до GND і напруга AVCC не повинна відрізнятися від напруги VCC більше, ніж на 0,4 В. Способи підключення цих виводів див. у розділі Технологія зменшення шуму ADC.

Зовнішня напруга порівняння подається на вивід AREF і повинна бути в діапазоні від 1,7 В до AVCC.

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

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

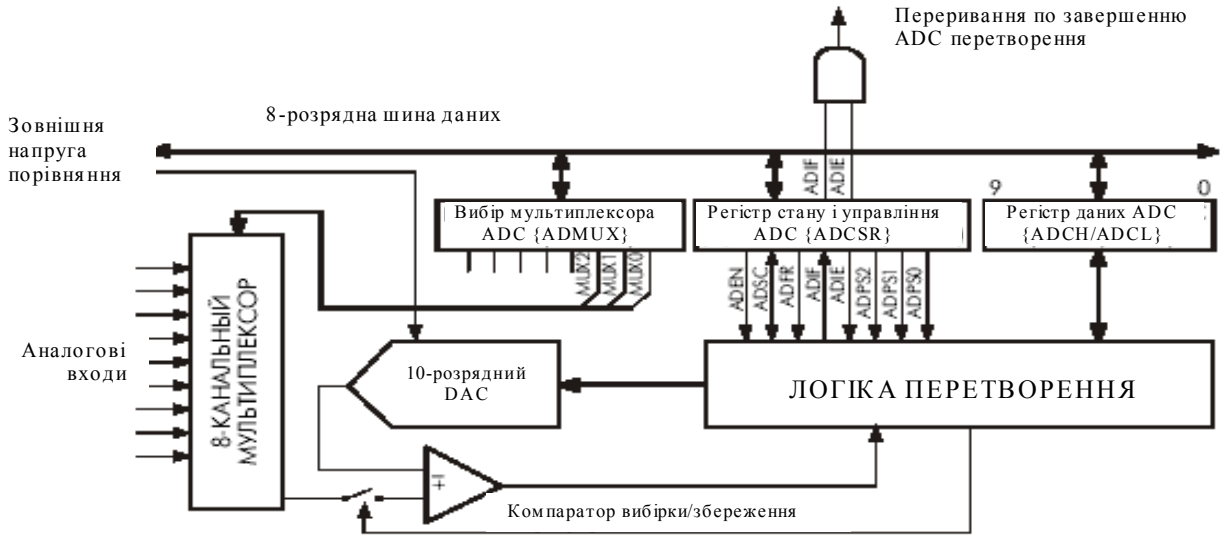


Рисунок 3.46 – Блок-схема аналого-цифрового перетворювача

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

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

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

Попередній поділ частоти

ADC працює з тактовою частотою в діапазоні від 50 до 300 кГц. У режимі циклічного перетворення для перетворення необхідно 14 тактових циклів, тобто перетворення виконується за час від 70 до 280 мс. У режимі одноразового перетворення перетворення виконується за 15 тактових циклів. Якщо тактова частота вийде за зазначені межі, то правильність результату не гарантується. Біти ADPS0 - ADPS2 використовуються для забезпечення необхідного діапазону тактової частоти ADC при частоті XTAL понад 100 кГц.

Функція зменшення шуму АЦП

Функція зменшення шуму забезпечує включення Idle режиму на час виконання перетворення, що дозволяє знизити шуми, створювані ядром CPU. Для реалізації цієї функції необхідно виконати наступну процедуру:

1. Відключити ADC очищенням біта ADEN.
2. Включити ADC і одночасно запустити перетворення установкою бітів ADEN і ADSC. У такий спосіб запускається порожнє перетворення, за яким піде робоче перетворення.
3. Протягом 14 тактових циклів порожнього перетворення ADC увести Idle режим.
4. Якщо перш, ніж буде завершене робоче перетворення, не відбудеться іншого переривання, то переривання ADC активує MCU і буде виконана підпрограма обробки переривання по завершенню перетворення ADC.

РЕГІСТР ВИБОРУ МУЛЬТИПЛЕКСОРА ADC – ADMUX (ADC Multiplexer Select Register)

Біт	7	6	5	4	3	2	1	0	
$07 {$27}	-	-	-	-	-	MUX2	MUX1	MUX0	ADMUX
Читання/Запис	R	R	R	R	R	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bits 7..3 - Res: Reserved Bits - Зарезервовані біти

Ці біти в мікроконтролерах ATmega603/103 зарезервовані і при зчитуванні завжди покажуть стан 0.

• Bits 2..0 - MUX2..MUX0: Analog Channel Select Bits 2-0 - Біти вибору аналогового каналу

Стан даних бітів визначає який із восьми аналогових каналів (0 - 7) буде підключений до ADC.

РЕГІСТР КЕРУВАННЯ І СТАНУ ADC – ADCSR (ADC Control and Status Register)

Біт	7	6	5	4	3	2	1	0	
$06 {$26}	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSR
Читання/Запис	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

• Bit 7 – ADEN: ADC Enable – Дозвіл ADC

Установка даного біту в стан 1 дозволяє ADC. Очищення біту забороняє ADC. Заборона ADC в процесі перетворення припиняє перетворення.

• Bit 6 – ADSC: ADC Start Conversion – Запуск перетворення ADC

В режимі одноразового перетворення для запуску кожного циклу перетворення необхідно встановлювати біт ADSC в стан 1. В циклічному режимі біт ADSC встановлюється в стан 1 тільки при запуску першого циклу перетворення. Кожен раз після першої установки біта ADSC, виконаної після дозволу ADC чи одночасно з дозволом ADC, буде виконуватися порожнє перетворення, що передує активованому перетворенню. Це порожнє перетворення активує ADC.

ADSC буде зберігати стан 1 протягом усього циклу перетворення і скидається по завершенні перетворення. При виконанні порожнього перетворення, що передує активованому, біт ADSC залишається встановленим до завершення активованого перетворення. Запис 0 у цей біт ефекту не робить.

• Bit 5 - ADFR: ADC Free Run Select - Установка циклічного режиму роботи ADC

При встановленому в стан 1 біті ADFR ADC буде працювати в циклічному режимі. У цьому режимі ADC робить вибірки і звертання до регістрів безупинно (одне за іншим). Очищення біта приводить до припинення циклічного режиму.

• Bit 4 - ADIF: ADC Interrupt Flag - Флаг переривання ADC

Даний біт встановлюється в стан 1 по завершенню перетворення і відновлення регістрів даних. Переривання по завершенню перетворення ADC виконується, якщо в стан 1 установлений біт ADIE і I-біт регістра SREG. Біт ADIF скидається апаратно при виконанні підпрограми обробки відповідного вектора переривання. Крім того, біт ADIF може бути очищений записом у флаг логічної 1. Цього необхідно остерігатися при читанні-модифікації-запису ADCSR, оскільки може бути заборонене відкладене переривання. Це застосовувано й у випадках використання команд SBI і CBI.

• Bit 3 - ADIE: ADC Interrupt Enable - Дозвіл переривання ADC

При встановлених у стан 1 біті ADIE та I-біті регістра SREG активується переривання по завершенню перетворення ADC.

• Bits 2..0 – ADPS2..ADPS0: ADC Prescaler Select Bits - Вибір коефіцієнту попереднього поділу

Дані біти визначають коефіцієнт поділу частоти XTAL для одержання необхідної тактової частоти ADC.

Таблиця 3.24 - Вибір коефіцієнта попереднього поділу

ADPS2	ADPS1	ADPS0	Кефіцієнт ділення
0	0	0	Без ділення
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

РЕГІСТРИ ДАНИХ ADC – ADCL та ADCH – (ADC Data Register)

Біт	15	14	13	12	11	10	9	8	
$05 {$25}	-	-	-	-	-	-	ADC9	ADC8	ADCH
$04 {$24}	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Читання/Запис	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Початкове значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

По завершенні циклу перетворення результат перетворення розміщується в цих двох регістрах. Важливо, щоб в циклічному режимі зчитувались обидва регістри і щоб регістр ADCL зчитувався перед зчитуванням ADCH.

Сканування аналогових каналів

Оскільки зміна аналогових каналів здійснюється після завершення циклу перетворення в циклічному режимі зміна каналів (сканування каналів) може проходити без переривання перетворювача. Зазвичай для виконання зміни каналу виконується переривання по завершенні перетворення. Однак користувачу необхідно взяти до уваги наступну думку: переривання активується зразу по готовності результату до зчитування. В циклічному режимі наступне перетворення починається через один тактовий цикл ADC після активації переривання. Якщо вміст ADMUX буде змінений на протязі цього одного тактового циклу, то нові установки будуть задіяні при початку нового перетворення. Якщо ж зміна стану ADMUX пройде пізніше цього тактового циклу, то при активованому перетворенні будуть використовуватися попередні установки.

Технологія зменшення шуму АЦП

Цифрові схеми самого мікроконтролера ATmega 603/103 і зовнішні цифрові схеми генерують електромагнітне випромінювання, яке може впливати на точність вимірювань аналогових сигналів. Якщо точність перетворення є визначальною, то можна використовувати наступні прийоми:

- Аналогова частина ATmega 603/103 і всі аналогові компоненти пристрою повинні мати на друкованій платі окрему аналогову землю. Ця аналогова земля повинна мати з'єднання з цифровою землею в одній точці друкованої плати.
- Провідники аналогових сигналів необхідно робити якомога коротшими, намагатися проводити їх поверх аналогової землі і, по можливості, якомога далі від доріжок високошвидкісних цифрових сигналів.
- Вивід AVCC мікроконтролерів ATmega 603/103 повинен підключатися до напруги живлення VCC через RC ланку, як показано на рис. 3.47.

Для зменшення шуму CPU можна використовувати функцію зменшення шуму ADC.

Якщо якісь виводи PORTF використовуються в якості цифрових входів, то важливо, щоб в процесі перетворення на цих виводах не проходили перемикання.

Таблиця 3.25-Характеристики ADC по постійному струму

Познач.	Параметр	Умови	Min	Typ	Max	Од. виміру
	Дозвіл			10		Біти
	Інтегральна нелінійність	VREF>2V		0.2	0.5	LSB
	Диференційна нелінійність	VREF>2V		0.2	0.5	LSB
	Помилка нуля (зміщення)			1		LSB
	Час перетворення		70		280	мкс
	Тактова частота		50		200	кГц
AV_{CC}	Аналогова напруга живлення		V_{CC}-0.3(1)		V_{CC}+0.3 (2)	В

V_{REF}	Напруга порівняння		AGND		AV_{CC}	В

Продовження таблиці 3.25

R_{REF}	Вхідний опір входу порівняння		6	10	13	кОм
R_{AIN}	Вхідний опір аналогового входу			100		МОм

Примітка: 1. Мінімальне при AVCC=2,7 В 2. Максимальне при AVCC=6,0 В

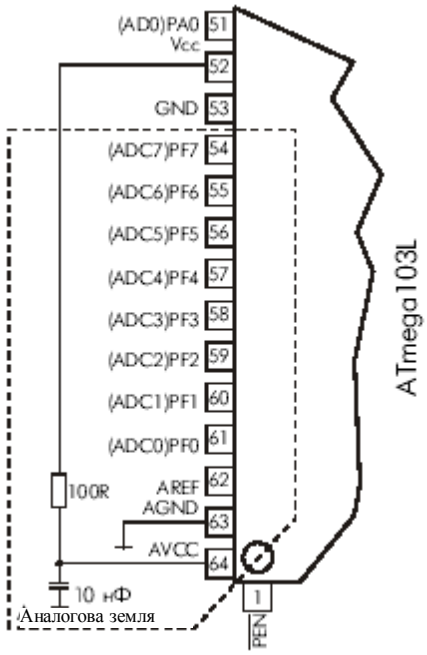


Рисунок 3.47 – Підключення напруги живлення ADC

3.15 Порти ВВОДУ-ВИВОДУ (I/O)

3.15.1 Порт A

Порт A є 8-розрядним двонаправленим I/O портом і оснащений вбудованими навантажувальними резисторами.Взаємодія з портом A здійснюється трьома розташованими в просторі I/O пам'яті даних регістрами: регістром даних - PORTA, $1B($3B), регістром напрямку даних - DDRA, $1A($3A) і регістром адрес виводів входу - PINA, $19($39). Регістр адрес виводів входу порту A забезпечує можливість тільки читання, регістри даних і напрямку даних порту A забезпечують можливість і читання, і запису. Усі виводи порту A оснащені індивідуально під'єднуваними вбудованими навантажувальними резисторами.

Вихідні буфери виводів порту A забезпечують струм, що втікає, до 40 мА, що досить для прямого керування LED дисплеями. Якщо виводи з PA0 по PA7 використовуються як входи і зовнішнім сигналом утримуються на низькому рівні, то струм, що витікає, забезпечується підключенням внутрішніх навантажувальних резисторів. Виводи порту A можуть виконувати, додаткову до основної функції, функцію забезпечення взаємодії з зовнішньою додатковою SRAM даних - вони можуть бути сконфігуровані як молодші розряди шини адреси/даних зовнішньої SRAM даних.

Додаткова функція включається установкою біта SRE (дозвіл зовнішньої SRAM) у регістрі керування MCU (MCUCR), при цьому установки регістра напрямку даних ігноруються.

РЕГІСТР ДАНИХ ПОРТА А – PORTA

Біт	7	6	5	4	3	2	1	0	
$1B {$3B}	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР НАПРЯМКУ ДАНИХ ПОРТА А – DDRA

Біт	7	6	5	4	3	2	1	0	
$1A {$3A}	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР ВИВОДІВ ВХОДУ ПОРТА А – PINA

Біт	7	6	5	4	3	2	1	0	
$19 {$39}	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Читання/Запис	R	R	R	R	R	R	R	R	
Початкове значення	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

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

РОБОТА ПОРТА А В ЯКОСТІ ЦИФРОВОГО I/O ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

Усі 8 бітів порту А при їхньому використанні в якості цифрового I/O працюють однаково.

Таблиця 3.26 - Вплив бітів DDAn на характер роботи виводів порту А

DDAn	PORTAn	I/O	Навантажуючий резистор	Опис
0	0	Вхід	Не під'єднаний	Третій стан (Hi-Z)
0	1	Вхід	Під'єднаний	При низькому рівні PAn забезпечують витікаючий струм
1	0	Вихід	Не під'єднаний	Низький рівень, двотактний вихід
1	1	Вихід	Не під'єднаний	Високий рівень, двотактний вихід

Примітка: PAn - виводи **I/O** загального призначення, n=7,6, ... 1,0 - номера виводів порту А.

Біти DDAn регістра DDRA визначають напрямок роботи відповідного виводу. При встановленому в стан 1 біті DDAn вивід PAn конфігурується як вивід виходу. При очищеному біті DDAn (скинутому в 0) вивід PAn конфігурується як вивід входу.

Якщо біт PORTAn встановлений у стан Т, коли відповідний вивід сконфігуровано як вхід, то навантажувальний MOS резистор активується (підключається). Для відключення навантажувального резистора біт PORTAn необхідно очистити або ж необхідно сконфігурувати вивід як вихід.

3.15.2 Порт В

Порт В є 8-розрядним двонаправленим I/O портом і оснащений вбудованими навантажувальними резисторами.

Взаємодія з портом В здійснюється трьома розташованими в просторі I/O пам'яті даних регістрами: регістром даних - PORTB, $18($38), регістром напрямку даних – DDRB, $l7($37) і регістром адрес виводів входу – PINB, $16($36). Регістр адрес виводів входу порту В забезпечує можливість тільки читання, регістри даних і напрямку даних порту В забезпечують можливість і читання, і запису. Усі виводи порту В оснащені індивідуально під'єднуваними вбудованими навантажувальними резисторами.

Вихідні буфери виводів порту В забезпечують струм, що втікає, до 40 мА, що достатньо для прямого керування LED дисплеями. Якщо виводи з PB0 по PB7 використовуються як

входи і зовнішнім сигналом утримуються на низькому рівні, то струм, що витікає, забезпечується підключенням внутрішніх навантажувальних резисторів. Виводи порту B можуть виконувати, додатково до основної функції, функції, представлені в таблиці 3.27.

Таблиця 3.27 - Додаткові функції виводів порту B

Вивід порта	Додаткова функція
PB0	Вхід вибору ведомого – SS {SPI Slave Select input}
PB1	Тактовий сигнал послідовної SPI-шини – SCK {SPI Bus Serial Clock}
PB2	Установка Ведучий вихід/Ведомий вхід SPI шини-MOSI {SPI Bus Master Output/Slave Input}
PB3	Установка Ведучий вхід/Ведомий вихід SPI шини-MISO {SPI Bus Master Input/Slave Output}
PB4	Порівняння виходу і PWM вихід таймера/лічильника0 – OC0A/PWM0A{Output Compare and PWM Output for Timer/Counter0}
PB5	Порівняння виходу і PWM вихід A таймера/лічильника1 – OC1A/PWM1A{Output Compare and PWM Output A for Timer/Counter1}
PB6	Порівняння виходу і PWM вихід B таймера/лічильника1 – OC1B/PWM1B{Output Compare and PWM Output B for Timer/Counter1}
PB7	Порівняння виходу і PWM вихід таймера/лічильника2 – OC2/PWM2{Output Compare and PWM Output for Timer/Counter2}

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

РЕГІСТР ДАНИХ ПОРТА B – PORTB

Біт	7	6	5	4	3	2	1	0	
$18 {$38}	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР НАПРЯМКУ ДАНИХ ПОРТА B – DDRB

Біт	7	6	5	4	3	2	1	0	
$17 {$37}	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР ВИВОДІВ ВХОДУ ПОРТА B – PINB – (PORT B Input Pins Address)

Біт	7	6	5	4	3	2	1	0	
$16 {$36}	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Читання/Запис	R	R	R	R	R	R	R	R	
Початкове значення	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

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

РОБОТА ПОРТА B У ЯКОСТІ ЦИФРОВОГО I/O ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

Усі 8 бітів порту B при їхньому використанні в якості цифрового I/O працюють однаково.

Таблиця 3.28 - Вплив бітів DDBn на характер роботи виводів порту B

DDBn	PORT Bn	I/O	Навантажуючий резистор	Опис
0	0	Вхід	Не під'єднаний	Третій стан (Hi-Z)
0	1	Вхід	Під'єднаний	При низькому рівні PBn забезпечують витікаючий струм
1	0	Вихід	Не під'єднаний	Низький рівень, двотактний вихід
1	1	Вихід	Не під'єднаний	Високий рівень, двотактний вихід

Примітка: PBn - виводи I/O загального призначення, n=7,6. ... 1,0 - номера виводів порту B.

Біти DDBn регістра DDRB визначають напрямок роботи відповідного виводу. При встановленому в стан 1 біті DDBn вивід PBn конфігурується як вивід виходу. При очищеному біті DDBn (скинутому в 0) вивід PBn конфігурується як вивід входу.

Якщо біт PORTBn встановлений у стан 1, коли відповідний вивід сконфігуровано як вхід, то навантажувальний MOS резистор активується (підключається). Для відключення навантажувального резистора біт PORTBn необхідно очистити або ж необхідно сконфігурувати вивід як вихід.

ДОДАТКОВІ ФУНКЦІЇ ВИВОДІВ ПОРТА B

Додаткові функції виводів порту B наступні:

• OC2/PWM2, Біт 7

Вивід порівняння виходу таймера/лічильника 2 (OC2) чи PWM вихід таймера/лічильника 2, який знаходиться в PWM режимі. Для виконання цих функцій вивід повинен бути сконфігурований як вихід.

• OC1B/PWM1B, Біт 6

Вивід порівняння виходу B таймера/лічильника 1 (OC1B) чи PWM вихід B таймера/лічильника 1, що знаходиться в PWM режимі. Для виконання цих функцій вивід повинен бути сконфігурований як вихід.

• OC1A/PWM1A, Біт 5

Вивід порівняння виходу A таймера/лічильника 1 (OC1A) чи PWM вихід A таймера/лічильника 1, що знаходиться в PWM режимі. Для виконання цих функцій вивід повинен бути сконфігурований як вихід.

• OC0/PWM0, Біт 4

Вивід порівняння виходу таймера/лічильника 0 (OC0) чи PWM вихід таймера/лічильника 0, що знаходиться в PWM режимі. Для виконання цих функцій вивід повинен бути сконфігурований як вихід.

• MISO - PORTB, Біт 3

Визначає вивід SPI каналу як вхід даних у режимі ведучого чи як вихід даних у режимі веденого. При дозволі SPI як ведучий, цей вивід конфігурується як вхід, незалежно від установки біту DDB3.

При дозволі SPI як веденого, напрямок даних на цьому виводі керується бітом DDB3 і, якщо вивід визначений як вхід, підключення навантажувального резистора керується бітом PORTB3. Подробиці див. в описі SPI порту.

• MOSI – PORTB, Біт 2

Вивід SPI каналу, обумовлений у ведучому режимі SPI як вхід даних і як вихід даних у веденому режимі. При SPI дозволеному як ведений, цей біт конфігурується як вхід, незалежно від стану біта DDB2. При дозволі SPI як ведучого напрямок даних на цьому виводі керується бітом DDB2 і, якщо вивід визначений як вхід, підключення навантажувального резистора керується бітом PORTB2. Подробиці див. в описі SPI порту.

• SCK - PORTB, Біт 1

Вихід тактового сигналу у ведучому режимі SPI каналу, вхід тактового сигналу у веденому режимі SPI каналу. При SPI дозволеному як ведений, цей біт конфігурується як вхід, незалежно від стану біта DDB1. При дозволі SPI як ведучого напрямок даних на цьому виводі

керується бітом DDB1 і, якщо вивід визначений як вхід, підключення навантажувального резистора керується бітом PORTB1. Подробиці див. в описі SPI порту.

• $\overline{SS}$ - PORTB, Біт 0

Вхід вибору порту як веденого. При SPI дозволеному як ведений, цей біт конфігурується як вхід, незалежно від стану біта DDB0. Як ведений SPI активується, коли на цей вхід поданий низький рівень. При SPI дозволеному в якості ведучого, напрямок даних на цьому виводі керується станом біта DDB0. Якщо вивід визначений як вхід, підключення навантажувального резистора керується бітом PORTB0. Подробиці див. в описі SPI порту.

3.15.3 Порт C

Порт C являє собою 8-розрядний вихідний порт. Крім основної функції виводи порту C виконують додаткову функцію забезпечення взаємодії із зовнішньою додатковою SRAM. При використанні зовнішньої SRAM через виводи порту C виводиться старший байт адреси зовнішньої SRAM.

РЕГІСТР ДАНИХ ПОРТА C – PORTC – (PORT C DATA REGISTER)

Біт	7	6	5	4	3	2	1	0	
$15 {$35}	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

3.15.4 Порт D

Порт D є 8-розрядним двонаправленим I/O портом і оснащений вбудованими навантажувальними резисторами.

Взаємодія з портом D здійснюється трьома розташованими в просторі I/O пам'яті даних регістрами: регістром даних - PORTD, $12($32), регістром напрямку даних - DDRD, $11($31) і регістром адрес виводів входу - PIND, $10($30). Регістр адрес виводів входу порту D забезпечує можливість тільки читання, регістри даних і напрямку даних порту D забезпечують можливість і читання, і запису.

Вихідні буфери виводів порту D забезпечують струм, що втікає, до 40 мА. Якщо виводи з PD0 по PD7 використовуються як входи і зовнішнім сигналом утримуються на низькому рівні, то струм, що витікає, забезпечується підключенням навантажувальних резисторів. Виводи порту D можуть виконувати додаткові до основної функції, представлені в Таблиці 3.29.
При використанні виводів порту для додаткових функцій, їхнє функціонування визначається установками регістрів DDRD і PORTD.

Таблиця 3.29– Додаткові функції виводів порту D

Вивід порта	Додаткова функція
PD0	Вхід зовнішнього переривання0 – INT0 – {External Interrupt0 Input}
PD1	Вхід зовнішнього переривання0 – INT1 – {External Interrupt1 Input}
PD2	Вхід зовнішнього переривання0 – INT2 – {External Interrupt2 Input}
PD3	Вхід зовнішнього переривання0 – INT3 – {External Interrupt3 Input}
PD4	Вхід тригера захоплення таймера/лічильника1 – IC1 – (Timer/Counter1 Input Capture Trigger)
PD6	Вхід тактового сигнала таймера/лічильника1 – T1 – (Timer/Counter1 Clock Input)
PD7	Вхід тактового сигнала таймера/лічильника2 – T2 – (Timer/Counter2 Clock Input)

РЕГІСТР ДАНИХ ПОРТА D – PORTD

Біт	7	6	5	4	3	2	1	0	
$12 {$32}	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР НАПРЯМКУ ДАНИХ ПОРТА D – DDRD

Біт	7	6	5	4	3	2	1	0	
$11 {$31}	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР АДРЕСИ ВИВОДІВ ВХОДУ ПОРТА D – PIND

Біт	7	6	5	4	3	2	1	0	
$10 {$30}	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Читання/Запис	R	R	R	R	R	R	R	R	
Початкове значення	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

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

РОБОТА ПОРТА D У ЯКОСТІ ЦИФРОВОГО I/O ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

Виводи порту PDn є виводами I/O загального призначення. Стани бітів DDDn регістра DDRD визначають напрямок роботи цих виводів. При встановленому в стан 1 біті DDDn вивід PDn конфігурується як вивід виходу, скидання біта DDDn у стан 0 конфігурує вивід PDn як вивід входу. При установці виводу PDn у стан 1, якщо він сконфігурований як вхід, активується MOS навантажувальний резистор. Для відключення навантажувального резистора вивід PDn повинен бути очищений (скинутий у стан 0) чи ж повинен бути сконфігурований як вивід виходу.

Таблиця 3.30– Вплив стану бітів DDDn на виводи порту D

DDDn	PORTDn	I/O	Навантажуючий резистор	Опис
0	0	Вхід	Від'єднаний	Третій стан (Hi-Z)
0	1	Вхід	Під'єднаний	При низькому рівні PDn забезпечують витікаючий струм
1	0	Вихід	Від'єднаний	Низький рівень, двотактний вихід
1	1	Вихід	Від'єднаний	Високий рівень, двотактний вихід

ДОДАТКОВІ ФУНКЦІЇ ВИВОДІВ ПОРТА D

$\overline{INT0}$.. $\overline{INT3}$ - PORTD, Біти 0..3

Виводи зовнішніх переривань з 0 по 3. Виводи PD0 – PD3 можуть бути використані як зовнішні активні низькі рівнем джерела переривань MCU. Активація вбудованих навантажувальних MOS резисторів описана вище. Дозвіл джерел переривань і інших подробиць описані в розділі присвяченому перериванням.

IC1 - PORTD, Біт 4

Вивід захоплення входу таймера/лічильника 1. При надходженні на вивід наростаючого чи падаючого фронту (залежить від установки) вміст таймера/лічильника 1 пересилається в регістр захоплення входу таймера/лічильника 1. Для забезпечення реалізації даної функції вивід повинен бути сконфігурований як вхід (DDD4 повинний бути скинутий у стан 0). Більш

докладно дана функція описана в описі таймера/лічильника 1. Активація вбудованого навантажувального MOS резистора описано вище.

T1 - PORTD, Біт 6

T1 - вивід таймера/лічильника 1. Подробиці в описі таймера/лічильника 1.

T2 - PORTD, Біт 7

T2 - вивід таймера/лічильника 2. Подробиці в описі таймера/лічильника 2.

3.15.5 Порт E

Порт E є 8-розрядним двонаправленим I/O портом і оснащений вбудованими навантажувальними резисторами.

Взаємодія з портом E здійснюється трьома розташованими в просторі I/O пам'яті даних регістрів: регістра даних - PORTE, $03($23), регістра напрямку даних – DDRE, $02($22) і регістра адрес виводів входу - PINE, $01($21). Регістр адрес виводів входу порту E забезпечує можливість тільки читання, регістри даних і напрямку даних порту E забезпечують можливість і читання, і запису.

Вихідні буфери виводів порту E забезпечують струм, що втікає, до 40 мА. Якщо виводи з PE0 по PE7 використовуються як входи і зовнішнім сигналом утримуються на низькому рівні, то струм, що витікає, забезпечується підключенням навантажувальних резисторів. Виводи порту E можуть виконувати додаткові до основних функції, представлені в Таблиці 3.31.

Таблиця 3.31 – Додаткові функції виводів порту E

Вивід порта	Додаткова функція
PE0	Вхід програмованих даних чи прийом UART (PDI/RXD)
PE1	Вихід програмованих даних чи передача UART (PDI/TXD)
PE2	Позитивний вхід аналогового компаратора (AC+)
PE3	Негативний вхід аналогового компаратора (AC-)
PE4	Вхід зовнішнього переривання (INT4)
PE5	Вхід зовнішнього переривання (INT5)
PE6	Вхід зовнішнього переривання (INT6)
PE7	Вхід зовнішнього переривання (INT7)

При використанні виводів порту для додаткових функцій, їх функціонування визначається установками регістрів DDRE і PORTE.

РЕГІСТР ДАНИХ ПОРТА E – PORTE

Біт	7	6	5	4	3	2	1	0	
$03 {$03}	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	PORTE
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР НАПРЯМКУ ДАНИХ ПОРТА E – DDRE

Біт	7	6	5	4	3	2	1	0	
$02 {$22}	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	DDRE
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

РЕГІСТР АДРЕСИ ВИВОДІВ ВХОДУ ПОРТА Е – PINE

Біт	7	6	5	4	3	2	1	0	
$01 {$21}	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	PINE
Читання/Запис	R	R	R	R	R	R	R	R	R
Початкове значення	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

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

РОБОТА ПОРТА Е В ЯКОСТІ ЦИФРОВОГО I/O ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

Виводи порту PEn є виводами I/O загального призначення. Стани бітів DDEn регістра DDRE визначають напрямок роботи цих виводів. При встановленому в стан 1 біті DDEn вивід PEn конфігурується як вивід виходу, скидання біта DDEn у стан 0 конфігурує вивід PEn як вивід входу. При установці вивід в PEn стан 1, якщо він сконфігурований як вхід, активується MOS навантажувальний резистор. Для відключення навантажувального резистора вивід PEn повинен бути очищений (скинутий у стан 0) чи ж повинний бути сконфігурований як вивід виходу.

Таблиця 3.32- Вплив стану бітів DDEn на виводи порту Е

DDEn	PORTDn	I/O	Навантажуючий резистор	Опис
0	0	Вхід	Від’єднаний	Третій стан (Hi-Z)
0	1	Вхід	Під’єднаний	При низькому рівні PDn забезпечують витікаючий струм
1	0	Вихід	Від’єднаний	Низький рівень, двотактний вихід
1	1	Вихід	Від’єднаний	Високий рівень, двотактний вихід

ДОДАТКОВІ ФУНКЦІЇ ВИВОДІВ ПОРТА Е

PDI/RXD – PORTE, Біт 0

PDI - вхід послідовних програмованих даних. У процесі послідовного завантаження даних програми цей вивід використовується для введення даних в ATmega603/103.

RXD - вхід прийому UART.

PDO/TXD - PORTE, Біт 1

PDO вихід послідовно програмованих даних. У процесі послідовного завантаження даних програми цей вивід використовується для виведення даних ATmega603/103.

TXD - вихід передавача UART.

AC+ - PORTE, Біт 3

AC+ - позитивний вхід аналогового компаратора. Даний вивід з’єднаний безпосередньо з позитивним входом аналогового компаратора.

AC- - PORTE, Біт 4

AC- - негативний вхід аналогового компаратора. Даний вивід з’єднаний безпосередньо з негативним входом аналогового компаратора.

INT4 .. INT7 – PORTD, Біти з 4 по 7

INT4 .. INT7 - джерела зовнішніх переривань з 4 по *7*. Виводи з 4 по 7 можуть використовуватися як джерела зовнішніх переривань MCU. Переривання на цих виводах можуть запускатися по низькому рівню на вході чи по наростаючому, чи падаючому фронту сигналу. Активація вбудованих навантажувальних MOS резисторів описано вище.

Способи дозволу джерел переривань і деталізація переривань приведені в описі переривань.

3.15.6 Порт F

Порт F є 8-розрядним портом. У просторі пам'яті I/O цьому порту відповідають тільки PINF, $00($20) - виводи входу порту F. Усі входи порту F з'єднані з аналоговим мультиплексором приєднаним, у свою чергу, до аналогово-цифрового перетворювача. Вивод порту F, крім виконання функцій входів мультиплексора, можуть бути використані і як цифрові входи, що дозволяє користувачу в один і той же час використовувати частину виводів порту F як цифрові входи і частину, що залишилася, як аналогові входи.

АДРЕСИ ВИВОДІВ ВХОДІВ ПОРТА F – PINF

Біт	7	6	5	4	3	2	1	0	
$00 {$20}	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	PINF
Читання/Запис	R	R	R	R	R	R	R	R	
Початкове значення	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

PINF - адреса виводів входу порту E не є регістром у повному змісті цього слова, ці адреси забезпечують зчитування фізичного стану кожного виводу порту.

3.16 Програмування пам'яті програм мікроконтролера

Біти блокування програмування пам'яті

MCU мікроконтролерів ATmega603/103 оснащено двома бітами, які можуть бути залишені незапрограмованими (у стані 1) чи запрограмованими (стан 0). Вплив станів бітів на роботу мікроконтролерів показано в Таблиці 3.33.

Таблиця 3.33– Режими захисту бітів блокування

Біти блокування програмування			Тип захисту
Режим	LB1	LB2	
1	1	1	Немає блокування програмування
2	0	1	Програмування Flash і EEPROM заборонено
3	0	0	Те саме що і режим 2, але заборонена і перевірка

Примітка: Біти блокування, при виконанні операції очищення кристала, можуть бути тільки стерті (стан 1).

Біти-запобіжники

Мікроконтролери ATmega603/103 оснащені чотирма бітами-запобіжниками SPIEN, SUT1, SUT0 і EESAVE. При запрограмованому в стан 0 біт SPIEN дозволяється послідовне завантаження програми. За замовчуванням біт SPIEN знаходиться в стані 0, у режимі послідовного програмування він недоступний і при виконанні операції очищення кристала його стан не змінюватися. Біти-запобіжники SUT1 і SUT0 визначають тривалість циклу запуску MCU. За замовчуванням ці біти не запрограмовані (стан бітів 11) і задають тривалість циклу запуску в 16 мс.

При запрограмованому біті-запобіжнику EESAVE пам'ять EEPROM попередньо обробляється циклом очищення кристала. За замовчуванням біт-запобіжник EESAVE не запрограмований (стан 1), він не може бути запрограмований, якщо хоча б один біт блокування запрограмований.

Байти сигнатури (коди ідентифікації)

Усі мікроконтролери фірми Atmel оснащені трьома байтами коду сигнатури, що дозволяють ідентифікувати прилад. Цей код може бути зчитаний і в послідовному, і в паралельному режимах. Ці три байти розташовані в окремому адресному просторі.

Для мікроконтролера ATmega603 це:

1. $00: $1E (показує що прилад виготовлений фірмою Atmel)
2. S01: $06 (показує що прилад оснащений 64 Кбайтами Flash пам'яті)
3. $02: S01 (якщо за адресою $01 знаходиться вміст $06, те це мікроконтролер ATmega603)

Для мікроконтролера ATmega103 це:

1. $00: $1E (показує що прилад виготовлений фірмою Atmel)
2. $01: $01 (показує що прилад оснащений 128 Кбайтами Flash пам'яті)
3. $02: $01 (якщо за адресою $01 знаходиться вміст $01, то це мікроконтролер ATmega103)

Програмування Flash і EEPROM пам'яті

Мікроконтролери ATmega603/103 оснащені програмованою внутрішньо-системно Flash пам'яттю, ємністю 64/128 Кбайт, і 2/4 Кбайтами EEPROM пам'яті даних. Після виготовлення мікроконтролерів, вбудована Flash пам'ять програм, і EEPROM пам'ять даних знаходяться в очищеному стані (тобто вміст - $FF) і готова до програмування. Прилади підтримують режим високовольтного (12 В) паралельного програмування і режим низьковольтного послідовного програмування. Напруга програмування 12 В використовується тільки якщо програмування дозволене, в іншому випадку струм по цьому виводу не споживається. Режим послідовного програмування є звичайним способом завантаження програм і даних у мікроконтролери, що знаходяться безпосередньо в системі користувача.

Матриця пам'яті, програм мікроконтролерів ATmega603/103 організована з 256/512 сторінок по 256 байт кожна. При програмуванні Flash пам'яті дані програми фіксуються в буфері сторінки, що дозволяє програмувати відразу цілу сторінку даних програми в кожному з режимів програмування.

Матриця EEPROM пам'яті даних мікроконтролерів програмується по-байтово (байт-за-байтом) у всіх режимах програмування. У послідовному режимі програмування вбудована функція самотактування EEPROM виконує автоматичне попереднє очищення кожного програмованого байта.

3.17 Гранично допустимі режими експлуатації

Діапазон робочих температур	від -40ºС до +105ºС
Діапазон температур зберігання	від -65ºС до +150ºС
Напруга на будь-якому виході відносно землі	від –1,0 В до +7,0 В
Максимальна робоча напруга	6,6 В
Споживання по постійному струму	300 мА

Примітки: Впливи за межами вказаних в таблиці режимів можуть призвести до виходу приладу з ладу. Вказані режими є стресовими і постійна робота приладів в цих та інших режимах за межами, вказаними в специфікаціях, недопустима. Тривала дія граничних режимів може призвести до зниження надійності приладів.

3.18 Характеристики за постійним струмом

TA = від -40ºC до 85ºC, VCC = від 2,7 В до 6,0 В

Познач.	Параметр	Умови	Мін	Тип	Макс	Од. вимір.
V_{IL}	Вхідна напруга низького рівня		-0,5		$0,3V_{CC}$	В
V_{IL1}	Вхідна напруга низького рівня	XTAL	-0,5		$0,2V_{CC}$	В
V_{IH}	Вхідна напруга високого рівня	Виключаючи (XTAL, RESET)	$0,6V_{CC}$		V_{CC}+0,5	В
V_{IH1}	Вхідна напруга високого рівня	XTAL	$0,8V_{CC}$		V_{CC}+0,5	В
V_{IH2}	Вхідна напруга високого рівня	RESET	V_{CC}		V_{CC}+0,5	В
V_{OL}	Вхідна напруга низького рівня. Порти A, B, C, D	I_{OL}=20 мА, V_{CC}=5 В I_{OL}=10 мА, V_{CC}=3 В			0,6 0,5	В
V_{OH}	Вхідна напруга високого рівня. Порти A, B, C, D	I_{OH}=-3 мА, V_{CC}=5 В I_{OH}=-1,5 мА, V_{CC}=3 В	4,2 2,3			В
I_{IL}	Вхідний струм витоку виводу I/O	V_{CC}=6 В, низький рівень	-8,0		8,0	мкА
I_{IH}	Вхідний струм витоку виводу I/O	V_{CC}=6 В, високий рівень	-8,0		8,0	мкА
RRST	Навантажувальний резистор скидання		100		500	кОм
$R_{I/O}$	Навантажувальний резистор I/O		35		120	кОм
I_{CC}	Споживаний струм в режимах:	Активний, 4 МГц, V_{CC}=3 В (2)			3,0	мА
		Idle, 4 МГц, V_{CC}=3 В		1,0	1,2	мА
		Power Down, 4 МГц, V_{CC}=3 В, WDT дозволений		8,5	15	мкА
		Power Down, 4 МГц, V_{CC}=3 В, WDT заборонений		< 1	2,0	мкА
V_{ACIO}	Напруга зміщення входу аналогового компаратора	V_{CC}=5 В			40	мВ
I_{ACLK}	Витік по входу аналогового компаратора	V_{CC}=5 В, V_{IN}=V_{CC}/2	-50		50	нА
t_{ACPD}	Затримка аналогового компаратора	V_{CC}=2,7 В, V_{CC}=4,0 В		750 500		нс

Примітки:

1. В режимі, що встановився, (не в перехідному) значення IOL повинні зовнішніми засобами обмежуватися на рівні:

Максимальний IOL на кожному виводі порту - 10 мА

Максимальний IOL по всім виводам виходу - 300 мА

Порт A - 26 мА

Порти A, B, D - 15мА

У випадку перевищення тестових величин IOL величина VOL може перевищити відповідні значення. Не гарантується вихідний струм виводу більший вказаного.

2. При тактовій частоті XTAL = 4 МГц тактова частота шини теж 4 МГц.

Таблиця 3.34– Характеристики зовнішньої пам'яті даних, напруга живлення від 4,0 до 6,0 В, стан очікування 1 цикл

	Позн.	Параметр	Генератор 8 МГц		Настроюваний генератор		Од. вимір.
			Мін	Макс	Мін	Макс	
0	$1/t_{CLCL}$	Частота генератора			0,0	8,0	МГц
10	t_{RIDV}	Низьке читання до дійсних даних		195,0		$2,0t_{CLCL}$-55,0	нс
12	t_{RLRH}	Ширина імпульсу RD	230,0		$2,0t_{CLCL}$-20,0		нс
15	t_{DVWH}	Дійсність даних до високого WR	220,0		$2,0t_{CLCL}$-30,0		нс
16	t_{WLWH}	Ширина імпульсу WR	167,5		$1,5t_{CLCL}$-20,0		нс

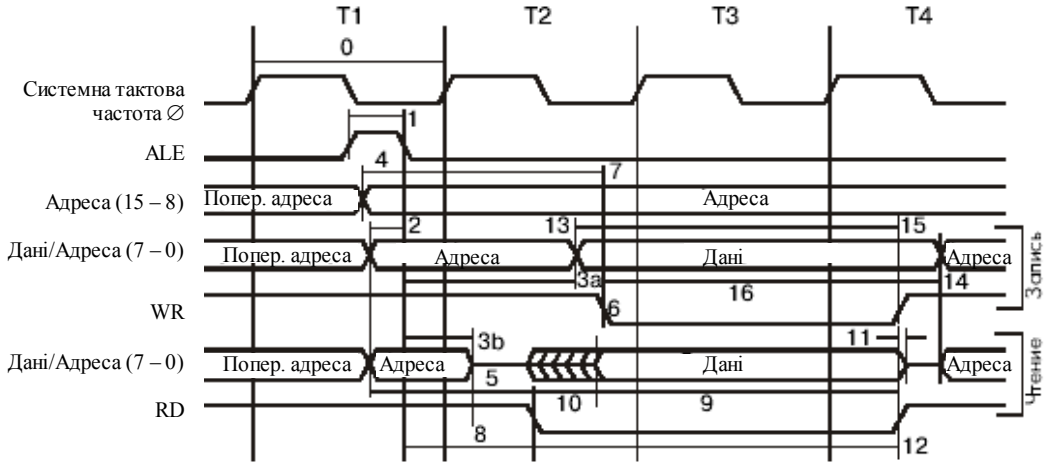


Рисунок 3.48 – Часові діаграми тактування зовнішньої пам'яті

Примітка: Тактовий цикл T3 присутній тільки в тому випадку, коли дозволений стан очікування зовнішньої SRAM.

Таблиця 3.35– Характеристики зовнішньої пам'яті даних, напр. живл. від 2,7 до 6,0 В, без стану очікування

	Позн.	Параметр	Генератор 8 МГц		Настроюваний генератор		Од. вимір.
			Мін	Макс	Мін	Макс	
0	$1/t_{CLCL}$	Частота генератора			0,0	4,0	МГц
1	t_{LHLL}	Ширина імпульсу ALE	70,0		$0,5t_{CLCL}$-55,0		нс
2	t_{AVLL}	Дійсність адреси A до низького ALE	60,0		$0,5t_{CLCL}$-65,0		нс
3a	t_{LLAX_ST}	Утримання адреси після низького ALE, команди ST/STD/STS	130,0		$0,5t_{CLCL}$-65,0		нс
3b	t_{LLAX_LD}	Утримання адреси після низького ALE, команди LD/LDD/LDS	15,0		15,0		нс

Продовження таблиці 3.35

4	t_{AVLLC}	Дійсність адреси С до низького ALE	60,0		$0,5t_{CLCL}$-65,0		нс
5	t_{AVRL}	Дійсність адреси С до низького RD	200,0		$1,0t_{CLCL}$-50,0		нс
6	t_{AVWL}	Дійсність адреси С до низького WR	325,0		$1,5t_{CLCL}$-50,0		нс
7	t_{LLWL}	Низький ALE до низького WR	230,0	270	$1,0t_{CLCL}$-20,0	$1,0t_{CLCL}$+20,0	нс
8	t_{LLRL}	Низький ALE до низького RD	105,0	145,0	$0,5t_{CLCL}$-20,0	$0,5t_{CLCL}$+20,0	нс
9	t_{DVRH}	Установка даних до високого RD	95,0		95,0		нс
10	t_{RIDV}	Низьке читання до дійсних даних		170,0		$1,0t_{CLCL}$-80,0	нс
11	t_{RHDX}	Утримання даних після високого RD	0,0		0,0		нс
12	t_{RLRH}	Ширина імпульсу RD	230,0		$1,0t_{CLCL}$-20,0		нс
13	t_{DVWL}	Установка даних до низького WR	70,0		$0,5t_{CLCL}$-55,0		нс
14	t_{WHDX}	Утримання даних після високого WR	0,0		0,0		нс
15	t_{DVWH}	Дійсність даних до високого WR	210,0		$1,0t_{CLCL}$-40,0		нс
16	t_{WLWH}	Ширина імпульсу WR	105,0		$0,5t_{CLCL}$-20,0		нс

Таблиця 3.36-Типи виконання та маркування мікроконтролерів

Fроб, МГц	Напруга жив., В	Код для замовлення	Тип корпусу	Діапазон робочих температур
4	2,7 – 6,0	ATmega603L-4AC	64A	Комерційний (від 0ºС до 70ºС)
		ATmega603L-4AI	64A	Промисловий (від -40ºС до 85ºС)
6	4,0 – 6,0	ATmega603L-6AC	64A	Комерційний (від 0ºС до 70ºС)
		ATmega603L-6AI	64A	Промисловий (від -40ºС до 85ºС)
4	2,7 – 6,0	ATmega103L-4AC	64A	Комерційний (від 0ºС до 70ºС)
		ATmega103L-4AI	64A	Промисловий (від -40ºС до 85ºС)
6	4,0 – 6,0	ATmega103L-6AC	64A	Комерційний (від 0ºС до 70ºС)
		ATmega103L-6AI	64A	Промисловий (від -40ºС до 85ºС)

Тип корпусу	
64A	64-вивідний тонкий (1,0 мм) квадратний пластмасовий корпус TQFP (Plastic Gull Wing Quad Flat Package)

Таблиця 3.37 - Регістри мікроконтролерів ATmega603/103

Адреса	Познач.	Біт 7	Біт 6	Біт 5	Біт 4	Біт 3	Біт 2	Біт 1	Біт 0
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
$3C ($5C)	XDIV	XDIV EN	XDIV 6	XDIV 5	XDIV4	XDIV 3	XDIV 2	XDIV 1	XDI V0
$3B ($5B)	RAMPZ	-	-	-	-	-	-	-	RAM PX0
$3A ($5A)	EICR	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC4 0
$39 ($59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
$38 ($58)	EIFR	INTF7	INTF6	INTF5	INTF4				
$37 ($57)	TIMSK	OCIE2	TOIE2	TICIE 1	OCIE1 A	OCIE 1B	TOIE 1	OCIE 0	TOIE 0
$36 ($56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1 B	TOV1	OCF0	TOV 0
$35 ($55)	MCUCR	SRE	SRW	SE	SM1	SM0	-	-	-
$34 ($54)	MCUSR	-	-	-	-	-	-	EXTR F	POR F
$33 ($53)	TCCR0	-	PWM 0	COM0 1	COM00	CTC0	CS02	CS01	CS00
$32 ($52)	TCNT0	Таймер/лічильник 0 (8-розрядний)							
$31 ($51)	OCR0	Регістр порівняння виходу Таймера/лічильника 0							
$30 ($50)	ASSR	-	-	-	-	AS0	TCN0 UB	OCR0 UB	TCR0 UB
$2F ($4F)	TCCR1A	COM1 A1	COM1 A0	COM1 B1	COM1B 0	-	-	PWM 11	PWM 10
$2E ($4E)	TCCR1B	ICNC 1	ICES1	-	-	CTC1	CS12	CS11	CS10
$2D ($4D)	TCNT1H	Старший байт регістра лічильника Таймера/лічильника 1							
$2C ($4C)	TCNT1L	Молодший байт регістра лічильника Таймера/лічильника 1							
$2B ($4B)	OCR1AH	Старший байт регістра порівняння А виходу Таймера/лічильника 1							
$2A ($4A)	OCR1AL	Молодший байт регістра порівняння А виходу Таймера/лічильника 1							
$29 ($49)	OCR1BH	Старший байт регістра порівняння В виходу Таймера/лічильника 1							
$28 ($48)	OCR1BL	Молодший байт регістра порівняння В виходу Таймера/лічильника 1							
$27 ($47)	ICR1H	Старший байт регістру захоплення входу Таймера/лічильника 1							
$26 ($46)	ICR1L	Молодший байт регістру захоплення входу Таймера/лічильника 1							
$25 ($45)	TCCR2	-	PWM 2	COM2 1	COM20	CTC2	CS22	CS21	CS20
$24 ($44)	TCNT2	Таймер/лічильник 2 (8-розрядний)							
$23 ($43)	OCR2	Регістр порівняння виходу Таймера/лічильника 2							
$21 ($41)	WDTCR	-	-	-	WDTO E	WDE	WDP2	WDP 1	WDP 0
$1F ($3F)	EEARH	-	-	-	-	EEAR 11	EEAR 10	EEAR 9	EEA R8
$1E ($3E)	EEARL	Молодший байт адреси EEPROM							
$1D ($3D)	EEDR	Регістр даних EEPROM							
$1C ($3C)	EECR	-	-	-	-	EERI E	EEM WE	EEW E	EER E
$1B ($3B)	PORTA	PORT A7	PORT A6	PORT A5	PORTA 4	PORT A3	PORT A2	PORT A1	POR TA0
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA 1	DDA 0
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA 3	PINA 2	PINA 1	PINA 0

Продовження таблиці 3.37

$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
$0F ($2F)	SPDR	Регістр даних SPI							
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	-
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
$0C ($2C)	UDR	Регістр даних UART I/O							
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8
$09 ($29)	UBRR	Регістр керування швидкістю UART							
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
$07 ($27)	ADMUX	-	-	-	-	-	MUX2	MUX1	MUX0
$06 ($26)	ADCSR	ADEN	ADCS	ADRF	ADIF	ADIE	ADPS2	ADPS1	ADPS0
$05 ($25)	ADCH	-	-	-	-	-	-	ADC9	ADC8
$04 ($24)	ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
$03 ($23)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
$02 ($22)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
$01 ($21)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
$00 ($20)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0

ІНФОРМАЦІЯ ПО КОРПУСУ

64A, 64-виводний тонкий (1,0 мм) квадратний пластмасовий корпус TQFP (Plastic Gull Wing Quad Flat Package)

Розміри в міліметрах і (дюймах).

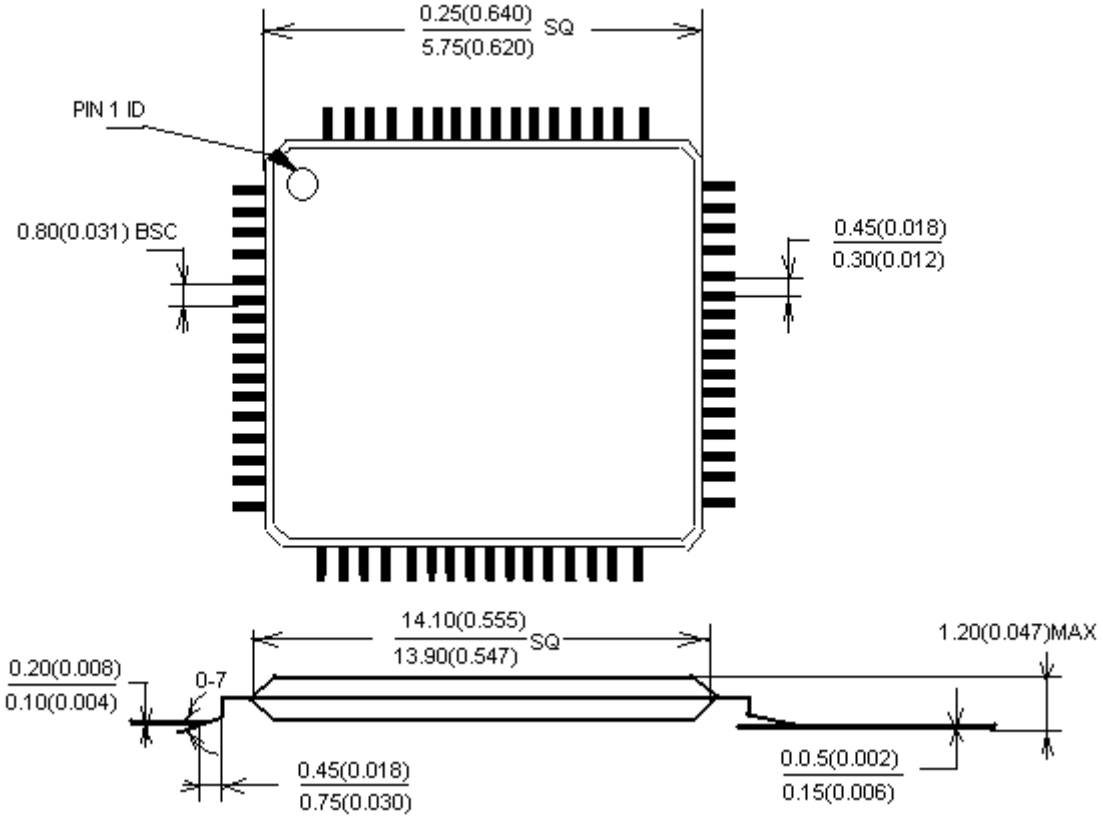


Рисунок 3.49 – конструкція корпусу мікроконтролера

3.19 Програмування мікроконтролерів з використанням мов високого рівня

Продуктивність та апаратні засоби сучасних мікроконтролерів є співрозмірними і в більшості випадків переважають аналогічні можливості персональних обчислювальних систем початку 90-х років XX століття. Крім того, конкуренція зі сторони фірм-виробників змушує шукати шляхів зменшення часу розробки своєї продукції, яка в свою чергу морально застаріває вже через 3-4 роки. Тому прийнятний ще 10 років тому час на розробку нової продукції (1-3 роки) при терміні експлуатації продукції 10-20 років значно скоротився і складає тепер 1-5 місяців. Розроблені вироби, як правило, використовуються в одиничних екземплярах, рідко доходячи до дрібносерійного виробництва. За таких умов програмування за допомогою мови Асемблера стає економічно невиправданим, оскільки потребує тривалого часу і зусиль. Тому на сучасному етапі все більшого значення набувають тенденції використання мов високого рівня при розробці автоматизованих систем різноманітного призначення. В якості мов високого рівня використовують переважно різноманітні компілятори мов Сі, Паскаль, Бейсик, Форт. Найбільшого розповсюдження для мікроконтролерів набула мова Сі. Проте при написанні програм мовою Сі часто виникає бажання використати прийоми, заборонені концепцією структурного програмування і притаманні мовам низького рівня (примусові виходи з циклу, переходи по мітках, передача даних через покажчики, тощо), що призводить до погіршення сприймання тексту програми як

самим автором так і іншими, і це стає джерелом додаткових помилок. При виборі програмного середовища розробки слід зважати на наступні фактори:

- наявність програмних засобів відладки програм (засоби трасування, відлагоджувачі), сумісних із даним компілятором; У випадку, якщо ці програми різних виробників, можлива неповна сумісність цих програмних продуктів.
- імовірність неправильної компіляції. Вибрати надійний компілятор можна виходячи з критерію його "популярності" у користувачів та наявність їх схвальних відгуків, про що можна легко дізнатись за допомогою мережі Інтернет. Важливу роль відіграє також час присутності програми на ринку. Регулярне оновлення програмного забезпечення та виправлення помилок свідчить про "серйозність" програмного забезпечення.
- ціна програмного продукту та умови поставки та технічної підтримки. Наявність "піратських" версій програмного забезпечення дозволяє вивчити його можливості та прийняти рішення про доцільність його придбання. Крім того, це свідчить про підвищену зацікавленість програмним забезпеченням зі сторони потенційних споживачів, а значить і про його реальну вартість. У випадку відсутності "піратських" версій програмного забезпечення найбільш зручним є використання компіляторів, які допускають повне короткотермінове користування (звичайно протягом 1-го місяця.) За цей час можна вивчити його та реалізувати нескладний проект, не порушуючи ліцензійних угод. Використання обмежених демо-версій (обмеження по розміру генерованого програмного коду на рівні 1-2 кБайт) слід вважати незручним. Це пов’язано з тим, що основні труднощі та помилки чи неточності компіляції виявляються, як правило, про роботі з масштабними проектами.
- знання мови програмування. Оскільки сучасні мови програмування високого рівня в багатьох своїх рисах подібні і реалізують однакові підходи у програмуванні, доцільно все ж вивчити нову мову, у випадку якщо вона добре відповідає вищевикладеним положенням. Як показує практика, для цього достатньо від 3 до 30 днів в залежності від складності задачі та бажання програміста.

При цьому необхідно зауважити, що використання мов високого рівня дозволяє легко розв’язувати тільки задачі, які не вимагають високої швидкодії та максимального використання апаратних засобів, проте потребують значної кількості обчислень. У випадку необхідності використань значної кількості побітових операцій та для досягнення максимальної продуктивності слід використовувати програмування мовою Асемблера. Слід зауважити також те, що у випадку, якщо компілятор дозволяє використання "асемблерних вставок", тоді слід фрагменти програми, критичні до швидкості виконання, переписати на асемблері. Як правило така технологія приносить значні успіхи, оскільки такі програми по швидкості не поступаються "повністю асемблерним", хоч і поступаються їм своїм розміром. У випадку використання AVR мікроконтролерів, найбільш вживаними є Сі - компілятори ICC (ImageCraft C Compiler), та CodeVision C, які однак не мають власних симулюючих засобів і в той чи інший спосіб є сумісними із стандартним симулятором-відлагоджувачем Astudio 3.55 фірми Atmel.

Як приклад, розглянемо середовище програмування E-Lab Pascal, яке містить в своєму складі наступні основні програми:

- Генератор шаблонів програм
- Текстовий редактор
- Компілятор з мови E-Lab Pascal, який генерує асемблерний код із Паскаль-програми користувача
- Компілятор Асемблера
- Симулятор-відлагоджувач, який дозволяє перевіряти логіку роботи програм включно з часовими залежностями, на ЕОМ типу IBM PC. Симулятор підтримує також стандартні апаратні засоби (символьний та графічний РКІ, клавіатура, 7-ми сегментний індикатор, АЦП, аналоговий компаратор, тощо)

За логікою роботи, мова програмування подібна до компілятора Turbo Pascal фірми Borland, тому перехід на E-Lab Pascal для осіб знайомих мовою Turbo Pascal труднощів не викликає. Навпаки, викликає повагу вилучення надмірних "BEGIN" та введення в синтаксис "ENDIF, ENFOR" замість безликих "END",(як у "Модулі-2") що значно полегшує загальне сприйняття написаних програм.

3.20 Короткий огляд E-Lab Pascal

Інтегроване середовище призначене для створення та відпрацювання програм, призначених для функціонування на мікроконтролерах серії AT90 (AVR). Мова програмування функціонально подібна до стандарту Pascal та Modula-2.

Основні символи мови:

Літери **A..Z, a..z** та **_** (підкреслювання). Цифри **0..9**. Спецсимволи: ***/=<>()[]{}|.,**
Різниці між великими та малими літерами немає.
Оператор присвоювання: **:=**
Операції відношення: **<> <= >=**
Коментарі: **(*** та ***)** можна використати **{** та **}**. Можна використати коментар в Сі-стилі, використовуючи **//**

Зарезервовані слова:

Зарезервовані слова є складовою частиною компілятора і не можуть бути змінені.
ABS, AND, ARRAY, ASM, BEGIN, BREAK, CASE, CONST, CONTINUE, DIV, DO, DOWNTO, ELSE, ELSIF, END, ENDFOR, ENDCASE, ENDWHILE, ENDIF, EXIT, FOR,FORWARD,FUNCTION, GOTO, IF, IN, LABEL, MOD, NOT, OF, OR, PROCEDURE, PROGRAM, RECORD, REPEAT, ROR, ROL, SHL, SHR, STRING, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH, XOR

Стандартні ідентифікатори:

Стандартні ідентифікатори визначають типи, константи, змінні, процедури і функції. Ці ідентифікатори не можуть бути змінені або перевизначені.
FALSE, TRUE, NIL, CHAR, BOOLEAN, INTEGER, BYTE, LONGINT, WORD, LONGWORD, FLOAT, POINTER, SIZEOF, ADDR, @, INC, DEC, MOVE, LENGTH, COPY, INTTOSTR, BYTETOSTR, INTTOHEX, STRTOINT, LO, HI, LOWORD, HIWORD, INSERT, DELETE, UPCASE, POS

Програмний рядок:

Максимальна довжина програмного рядка складає 250 символів.

Стандартні типи даних:

BOOLEAN	8bit, true..false
BYTE	8bit, 0..255
CHAR	8bit, chr(0..255)
BIT	1bit, true..false, 0..1
ENUM	8bit, 0..255
BITSET	8/16bit
STRING	обмежується 255 символами
TABLE	255 елементів, 1 вимір
ARRAY	32000 елементів, 4 виміри
RECORD	визначає комплексний тип за бажанням користувача
POINTER	16bit, @var, nil
WORD	16bit, 0..65535,
INTEGER	16bit, -32767..32768
LONGWORD	32bit, 0..4294967295
LONGINT	32bit, -32767..32768
FLOAT	24bit, 10e-18..10e18
SEMAPHORE	8bit, байт
PIPE	**..**255 елементів, byte, word, int, float і тд.
SYSTIMER	16bit, word
SYSTIMER8	8bit, byte

PIDCONTROL Special Record. pFactor, iFactor, dFactor, sFactor, Actual, Nominal, Execute

Оператори:

NOT a:= not a; інвертована змінна a
DIV a:= a div b; Ділення
MOD a:= a mod 5; Залишок від ділення
AND a:= a and $0f; And mask
OR a:= a or $30; Or mask
XOR a:= a xor b; Xor
SHL a:= a shl 5; Зсув вліво
SHLA a:= a shla 5; арифметичний зсув вліво
SHR a:= a shr b; зсув вправо
SHRA a:= a shra b; арифметичний зсув вправо
ROL a:= a rol 4; ротація вліво
ROR a:= a ror x; ротація вправо
IN якщо ch входить в множину [’a’..’z’] тоді
+ a:=a +5; додавання
- a:=a -b; віднімання
***** a:=a *b; множення
/ a:= a / 5.5; Ділення з плаваючою комою
@ p:= @a; операція взяття адреси змінної
^ x:= p^; Змінна як покажчик
% x:= %01100101; binary constant
$ x:= $00FF; Hexadecimal constant
# x:= #13; Перетворення 10-го значення в Char (13 -> CR)

Ключові слова:

PROGRAM Старт основної програми
DEVICE Визначення типу процесора та обладнання
IMPORT Імпорт системних функцій
FROM Longs, Float, Processes, Tasks, Pipes, Pids
SYSTEM Longs, Float, Processes, Tasks, Pipes, Pids
DEFINE Параметри для імпортованих функцій
IMPLEMENTATION Старт програми
TYPE Початок визначення типу
CONST Початок визначення констант
STRUCTCONST Початок визначення структурованих констант
VAR Початок визначення змінних
FORWARD Майбутнє визначення процедур та функцій
PROCEDURE Визначення процедур
FUNCTION Визначення функцій
PROCESS Визначення процесу
TASK Task declaration
INTERRUPT Визначення переривання
BEGIN Старт процедури, процесу, задачі, гілки програми
RETURN Зупинка і вихід в процедурі/функції
END Кінець процедури чи гілки програми.
ASM Початок асемблерної вставки
ENDASM Кінець асемблерної вставки
IF Умовний оператор. IF Вираз потребує наступних THEN і ENDIF
THEN IF a >b then a:=b;EndIf;
ELSE IF a >b then a:= b else b:= a; EndIf;
ELSIF IF a >bthen .. elsif b =athen ; EndIf;
ENDIF End of IF statements
LABEL Визначення мітки для оператора GOTO
GOTO Абсолютний перехід в межах блоку
CASE початок CASE виразу Вибір x із…
ENDCASE Кінець CASE виразу

FOR Початок FOR - циклу For x:= 0 to 10 do
TO Напрям FOR циклу: вгору i:=i+1
DOWNTO Напрям FOR циклу: вниз i:=i-1
BY Крок зміни параметру для циклу FOR
ENDFOR Кінець циклу FOR
WHILE Початок циклу While
ENDWHILE Кінець циклу While
REPEAT Початок циклу repeat
UNTIL Кінець циклу repeat
BREAK Примусова зупинка циклів FOR, WHILE або REPEAT
CONTINUE Перехід на початок циклу FOR, WHILE або REPEAT
LOOP Початок нескінченного циклу loop
ENDLOOP Кінець нескінченного циклу loop
EXITLOOP Вихід з нескінченного циклу loop

```
Loop
...
If a >bthen exitLoop; EndIf;
...
EndLoop;
```

Системні бібліотеки

Стандартна бібліотека для більшості випадків відповідає стандарту мови Паскаль, і додатково включає в себе деякі корисні математичні функції (піднесення до степеня, десяткові логарифми, роботу з бітами, часові затримки, тощо.)

TRUE Наперед визначена константа
FALSE Наперед визначена константа
NIL Наперед визначена константа для типу pointer
PI Наперед визначена константа для функції circle
SYSTEM_INIT Процедура для ініціалізації системного обладнання
SYSTEM_RESET Reset системи і рестарт
LO молодший байт 16bit значення a:= lo(i)
LOWORD молодше слово 32bit значення a:= LoWord(ii)
HI старший байт 16bit значення a:= hi(i)
HIWORD старше слово 32bit значення a:= HiWord(ii)
ABS Абсолютне значення типу Integer, Longint, Float
ADDR Адреса змінної, результат - безтиповий покажчик
INCL встановити біт: Incl(a, 5);
EXCL скинути біт: Excl(w, a);
TOGGLE змінити біт на протилежний: Toggle(x, a);
BIT перевірити біт: if Bit(a, 0) then.. ; Endif;
SETBIT set/reset бітт: SetBit(a, true);
INC збільшити змінну на 1: inc(a);
DEC зменшити змінну на 1: dec(b);
SWAP обміняти LoByte/HiByte або півбайта: x:= Swap(b);
ODD перевірити значення на непарність: if ODD(x) then.. ;
ORD код символа: x:= ord(ch);
LOWER менше значення для 2-х значень: x:= lower(y, z);
HIGHER більше значення для 2-х значень: x:= higher(y, z);
WITHIN число a перевір. на попадання в межі: x:= WithIn(low, a, high); і повертається одне з 3-х значень в залежності від результату перевірки.
MIN найменше можливе для даного типу: x:= min(a);
MAX найбільше можливе для даного типу: x:= max(a);
BOOLEAN перетворення аргументу до Boolean : bo:= boolean(x);
BYTE перетворення аргументу до Byte : b:= byte(x);
CHAR перетворення аргументу до Char : ch:= char(x);
WORD перетворення аргументу до Word : w:= word(x);
INTEGER перетворення аргументу до Integer : i:= integer(x);

LONGWORD перетворення аргументу до LongWord: ww:=longword(x);
LONGINT перетворення аргументу до LongInt : Li:= LongInt(x);
FLOAT перетворення аргументу до Float : f:= Float(x);
POINTER перетворення аргументу до Pointer : p:= Pointer(x);
VAL перетворення рядкового типу до чисельного Val(S; var V; var err: Err);
UPCASE зміна символу до верхнього регістру: ch:= UpCase(ch);
UPPERCASE зміна символу в рядку до верхнього регістру: st:= UpperCase(st);
LOWCASE зміна символу до нижнього регістру: ch:= LowCase(ch);
LOWERCASE зміна символу в рядку до нижнього регістру: st:= LowerCase(st);
LENGTH довжина рядка: x:= Length(st1);
SIZEOF пам'ять необхідна для об'єкта : x:= SizeOf(word);
COPY копіювання частини рядка до іншого рядка

GETTABLE читати член таблиці : x:= GetTable(Table1, 5);
SETTABLE писати член таблиці: SetTable(Table1, 5, x);
FLUSHBUFFER записати буфер в флеш : FlushBuffer([RxBuffer, TxBuffer]);
COPYBLOCK копіювання пам'яті: CopyBlock(src, dest, len);
FILLBLOCK Заповнити масив пам'яті : FillBlock(dest, len, x);
RANDOM Псевдо-випадкові числа: x:= Random;
SQR Квадрат числа: x:= sqr(a);
SQRT Квадратний корінь числа: x:= sqrt(a);
POW Степінь числа : f:= pow(x, y);
POW10 Значення 10^x : f:= pow10(x);
EXP значення e^x : f:= exp(x);
LOG2 Двійковий логарифм x: f:= log2(x);
LOG10 10 логарифм x: f:= log10(x);
DEGTORAD перетворення градуси -> рад. : r:= DegToRad(w);
RADTODEG перетворення радіани -> градуси: w:= RadToDeg(r);
ARCTAN обчислення ArcTan: w:= ArcTan(r);
TAN тригонометрична функція : t:= Tan(w);
SIN тригонометрична функція: s:= Sin(w);
COS тригонометрична функція: c:= Cos(w);
TRUNC ціла частина типу Float : i:= trunc(f);
ROUND округлене до більшого цілого число типу Float : i:= round(f);
FRAC дробова частина від числа типу Float : f:= frac(f);
INT дробова частина від числа типу Float : f:= int(f);
BYTETOSTR перетворення числового значення до рядкового типу.
LONGTOSTR перетворення числового довгого значення до рядкового типу.
FLOATTOSTR перетв. числового значення з плаваючою комою до рядкового типу.
BYTETOHEX, перетворення числового значення до 16-го рядка.
INTTOHEX перетворення числового значення до 16-го рядка.
LONGTOHEX перетворення числового значення до 16-го рядка.
STRTOINT, перетворення рядка до числового значення
STRTOFLOAT перетворення рядка до числового значення
WRITE Писати символ або рядок
READ Читати символ або рядок
MDELAY Програмна затримка в мілісек (1..high(word)) точність < 20%
UDELAY Програмна затримка в мксек x 10 (1..high(byte)) точність < 20%
SDELAY Програмна затримка в CPU циклів (1..high(byte))
ISSYSTIMERZERO Boolean, true if a Systimer набуває значення = 0
SETSYSTIMER Завантажує Systimer with початковим значення
GETSYSTIMER Повертає актуальне значення Systimer-a
RESETTIMER Повертає Systimer в 0
ENABLEINTS Дозволяє глобальні переривання
DISABLEINTS Забороняє глобальні переривання
CPUSLEEP CPU переходить в 'sleep'-режим.
POWERSAVE CPU переходить в режим збереження потужності.
WATCHDOGSTART Ініціалізує WatchDog, при потребі

WATCHDOGSTOP Зупиняє WatchDog
WATCHDOGTRIG Перекидає стан WatchDog, якщо він дозволений
WATCHDOGFLAG Boolean, є true, якщо WatchDog-TimeOut.
RUNERR Boolean, є true, якщо run-time помилка.

Функції, які залежать від апаратного забезпечення:

PROCCLOCK Тактова частота процесора в Гц
STACKSIZE Розмір стеку для основної програми в байтах
FRAMESIZE Розмір фрейму для процедур в байтах
TASKSTACK Розмір стеку для задач в байтах
RunTimeErr Визначення Range/StackCheck Procedure.
SYSTICK Контрольоване таймером переривання для функцій часу мультизадачності, процесів
SWITCHPORT1 визначає Portadr для усунення "брязкоту" контактів Port
PORT_STABLE1 Змінна, Port без "брязкоту" контактів
INP_STABLE1 Функція (SwitchP1): if INP_STABLE1(1) then..
INP_RAISE1 Функція (SwitchP1): if INP_RAISE1(0) then..
SWITCHPORT2 визначає Portadr для усунення "брязкоту" контактів Port
PORT_STABLE2 Змінна, Port без "брязкоту" контактів
INP_STABLE2 Функція (SwitchP2): if INP_STABLE2(1) then..
INP_RAISE2 Функція (SwitchP2): if INP_RAISE2(0) then..
PWMPORT1 Імпорт і визначення PWM каналу 1
PWMPORT2 Імпорт і визначення PWM каналу 2
LCDPORT Імпорт і визначення порта РКІ (LCD)
LCDTYPE Імпорт і визначення типу контролера
LCDROWS Визначення рядків РКІ дисплея
LCDCOLUMNS Визначення стовпчиків РКІ дисплея
LCDOUT Писати в РКІ дисплей
LCDINP Читати з РКІ дисплея
LCDCTRL Писати в РКІ порт управління
LCDSTAT Читати з РКІ порта стану
LCDUPPER Вибір 1-го РКІ контролера
LCDLOWER Вибір 2-го РКІ контролера .. підтримується вивід на два РКІ
LCDOFF Вимкнути дисплей
LCDHOME Курсор в позицію 0, 0
LCDCLR Стерти РКІ і Курсор в позицію 0, 0
LCDCURSOR Вибір режиму курсора
LCDXY Курсор в позицію x, y
LCDCHARSET Визначити власні символи для РКІ
LCDSETUP Власна ініціалізація контролера РКІ
DISP7SPORT Імпорт і визначення 7-сегм. світлодіодного дисплея
DISPDIGITS Визначення позиції для 7-сегм. світлодіодного дисплея
DISPOUT Писати в 7-сегм. світлодіодного диспл.
DISPPOS Позиція курсора
DISPBLINK Режим "Блимання" дисплея
DISPDIGBLINK "Блимання" знаку
DISPCLEAR Повна очистка дисплею
DISPCLEOL Очистка дисплею від курсора до кінця лінії
SERPORT Визначення та імпорт послідовного інтерфейсу
RXBUFFER Визначення довжини RX-буфера і сторінки пам’яті
TXBUFFER Визначення довжини TX-буфера і сторінки пам’яті
SERINP Зчитує дані з послідовного інтерфейсу або буфера
SEROUT Записує дані до послідовного інтерфейсу або буфера
SERSTAT повертає стан послідовного інтерфейсу
SERPORT2 Визначає та імпортує 2-й послідовн. інтерфейс
RXBUFFER2 Визначає довжину RX-буфера і сторінки пам’яті
TXBUFFER2 Визначає довжину TX-буфера і сторінки пам’яті
SERINP2 Читає послідовний інтерфейс або буфер

SEROUT2 Записує в послідовний інтерфейс або буфер
SERSTAT2 повертає стан послідовного інтерфейсу
SPIPORT Визначає режим Master/Slave SPI-інтерфейс
SPIORDER Визначає порядок передачі LSB/MSB SPI-інтерфейсу
SPICPOL Визначає полярність біту
SPICPHA Визначає фазу передачі SPI-інтерфейсу
SPIPRESC Визначає подільник/швидкість SPI-інтерфейсу для Master-режиму
ADCPORT Імпорт АЦП
ADCCHANS Визначає необхідний канал АЦП
ADCPRESC Визначає необхідний час перетворення АЦП
GETADC Повернення даних АЦП
I2CPORT Визначає та імпортує I2C-інтерфейс
I2CSTAT Стан I2C-інтерфейсу
I2CINP Зчитування даних через I2C-інтерфейс
I2COUT Запис даних через I2C-інтерфейс
STEPPORT Визначає та імпортує інтерфейс крокового двигуна
STEPPERON Вмикає живлення на кроковий двигун
STEPPEROFF Вимикає живлення від крокового двигуна
STEPONECW Один крок за часовою стрілкою
STEPONECCW Один крок проти часової стрілки
STEPRAMPCW Прискорений крок за часовою стрілкою
STEPRAMPCCW Прискорений крок проти часової стрілки
STEPDESTCW n кроків за часовою стрілкою
STEPDESTCCW n кроків проти часової стрілки
EEPROM Імпортує зчитування та запис Eeprom

Каркас будь-якої програми будується за допомогою додаткової програми – генератора шаблонів (Application Wizard). Заповнивши всі необхідні поля діалогових вікон, у відповідності до апаратури яка використовується в задачі, користувач може отримати шаблон програми з розрахованими коефіцієнтами швидкості передачі послідовного порта, встановленими параметрами таймерів, АЦП, описами необхідних функцій та процедур, тощо. Для одержання готової програми необхідно заповнити тіла процедур та функцій необхідним кодом, і дописати додаткові фрагменти програми, якщо необхідно. При цьому набір функцій апаратно-залежної бібліотеки дозволяє програмно реалізувати засоби, відсутні в мікроконтролерах (наприклад, інтерфейс I^2C). Наявність засобів відлагоджування програм дозволяє симулювати роботу мікроконтролерів і що особливо важливо – у поєднанні із іншими апаратними засобами: символьними РКІ, графічними РКІ, АЦП, клавіатурою, тощо.

Для ілюстрації можливостей використання E-Lab Pascal, наведемо програму, яка після виводу короткого повідомлення одразу після включення живлення здійснює послідовне вимірювання напруг в першому та другому каналі вбудованого АЦП мікроконтролера 90S8535 та виводить їх на 6-ти розрядний 7-ми сегментний світлодіодний (LED) індикатор.

```
program ADCtest_AVR;
{$NOSHADOW}
{ $S+ Stack check}        {switch not used}
{ $R+ Range check}        {switch not used}
{ $W+ Warnings}           {Warnings off}
// Розділ описів програми
Device = 90S8535; // Виберемо тип процесора
Import SysTick, ADCPort, Disp7sPort; // Імпорт системного таймера, АЦП, 7-сегм. індикатора
From System Import ;  // …Вкажемо звідки це все імпортуємо…

Define
      ProcClock   = 8000000;       {Гц – частота кварцу}
      SysTick     = 4;             {мілісек – системний квант часу}
```

```
      StackSize   = $0020, iData; // Розмір стеку
      FrameSize   = $0010, iData; // Розмір фрейму
      ADCchans    = 4, iData;//  [n], iData; = [m,n], iData; Кількість каналів та розташування даних
      ADCpresc    = 128; // Подільник частоти АЦП
      Disp7sPort  = PortC;    // Вибір порта підключення індикатора
      DispDigits  = 6, iData;    // Кількість розрядів індикатора

Implementation       // Розділ реалізації програми
     {$IDATA}       // Розміщення в внутрішній пам'яті процесора
{------------------------------------------------------------}
{ Type Declarations – тут слід описати можливі типи даних, яких в даному разі немає}
type
{------------------------------------------------------------}
{ Const Declarations– тут слід описати можливі константи, яких в даному разі немає }
{------------------------------------------------------------}
{ Var Declarations  - опис глобальних змінних}
var
   value      : word;
{------------------------------------------------------------}
{ functions – опис функцій та процедур користувача, яких в даному разі немає }
{------------------------------------------------------------}
{ Main Program –Основна програма }
{$IDATA}
 begin
  EnableInts;                          // Дозвіл переривань
  Write(DispOut, 'E-LAB');             // Виведемо повідомлення на дисплей
  mDelay(1000);                        // Потримаємо його протягом 1 сек.
  DispOut(#10);                        // Тепер очистимо дисплей
loop                                   // Нескінченний цикл
   Value:= GetAdc(1) and $3ff;       // Зчитаємо дані з АЦП –1 фільтруючи надлишкові розряди
   Write(DispOut, '1='+IntToStr(Value));    // Виведемо на дисплей значення
   mDelay(1000);                       // Почекаємо 1 сек.
   DispOut(#10);                       // Тепер очистимо дисплей
   Value:= GetAdc(2) and $3ff;         // канал 2 АЦП фільтрація з надлишку 10-розрядів
   Write(DispOut, '2='+IntToStr(Value));      // Виведемо на дисплей значення
   mDelay(1000);                       // Почекаємо 1 сек.
   DispOut(#10);                              // Тепер очистимо дисплей
endloop;                                 // Повторимо знову все спочатку
end ADCtest_AVR.
```

3.21 Особливості програмування

Використання E-Lab Pascal надає переваги при написанні коротких програм на зразок наведеної вище, побудованих на принципі "один контролер - одна задача". Такий принцип дозволяє легко відлагоджувати програми для кожного мікроконтролерного вузла та будувати багатоконтролерні системи. Такий підхід забезпечує високу швидкість виконання проектів, високу надійність роботи програм за дещо надмірних апаратних витрат. У випадку, якщо з тих, чи інших причин, програмування безпосередньо мовою E-Lab Pascal не забезпечує необхідних характеристик швидкодії, можна використовувати вбудований асемблер. Як асемблерний ідентифікується код, обмежений інструкціями : ASM;…. ENDASM; між якими слід вставити необхідні асемблерні команди. Дуже часто виникає потреба використання змінних E-Lab Pascal в асемблерному коді. Для цього слід проаналізувати файл лістінгу *.lst проекту, і знайти асемблерний аналог зарезервованої змінної та її тип даних. Наприклад для

глобальної змінної **j1** компілятор генерує кодове ім'я **asd.j1** досупне з будь-якого місця програми. У випадку передачі даних в функції та процедури використовується фреймовий механізм передачі. *Фрейм - це виділена область оперативної пам'яті, що характеризується своїм покажчиком на вершину*. Механізм дії аналогічний стековій пам'яті, за винятком того, що дії з фреймом здійснюються програмно. **Вершина_фрейму-1** вказує на останню змінну в списку і міститься в регістрі Y. При цьому не робиться різниці між змінними-параметрами, які передаються до процедур та функцій та локальними змінними функцій чи процедур.

Розглянемо приклад. Нехай існує деяка процедура ASDx описана на Паскалі наступним чином (рядки додатково пронумеровані):

```
{47}   procedure ASDx(x1:byte;var y:word);
{48}     var agt:byte;
{49}           begin
{50}             agt:=5;
{51}             x1:=agt;
{52}           end;
```

Після компіляції, можна отримати наступний асемблерний код, який в даному випадку для зрозумілості містить коментарі компілятора та додаткові коментарі:

```
.FUNC   ASDX, 47, 00020h
ASD.ASDX:
                ;SYM    X1, 00003h, 0000Dh, 3 //03h - зміщення відносно 1-ї змінн.
                ;SYM    Y, 00001h, 0800Eh, 3
                ;SYM    AGT, 00000h, 0000Dh, 3
                SBIW     _FRAMEPTR, 1           //_FRAMEPTR:= _FRAMEPTR-1
                .BLOCK  49
                .LINE   50
                MOV      _ACCCLO, _FRAMEPTR // молодший байт :=Lo(Y)
                MOV      _ACCCHI, _FPTRHI       // старший байт Z:=Hi(Y)
                LDI      _ACCA, 005h         // _ACCA:=5
                ST       Z, _ACCA// Зберегти за адресою Z (agt:=5;)
                .LINE   51
                MOV      _ACCCLO, _FRAMEPTR
                MOV      _ACCCHI, _FPTRHI
                ADIW     _ACCCLO, 000003h // Додати зміщення змінної X1
                PUSH     _ACCCLO // Зберегти Z в стеку
                PUSH     _ACCCHI  //------//-------
                MOV      _ACCCLO, _FRAMEPTR
                MOV      _ACCCHI, _FPTRHI // Z:=ADDR(agt)
                LD       _ACCA, Z //  _ACCA:=agt
                POP      _ACCCHI // Відновити Z зі стеку
                POP      _ACCCLO //----------\\----------
                ST       Z+, _ACCA // x1:=agt
                .ENDBLOCK 52
ASD.ASDX_X:
                .LINE   52
                RET             // Повернення із підпрограми
                .ENDFUNC  52
```

Визначення початкових адрес змінних, які передаються до функцій чи процедур потребує обчислення їх положення відносно початкового покажчика фрейму шляхом додавання їх зміщення (значення генерується компілятором з директивою ;SYM) до адреси останньої змінної в списку, який передається.

Використовуючи таку техніку, можна формувати звернення до змінних в процедурах. Повернення результату виконання функцій в мові E-Lab Pascal здійснюється за допомогою оператора **return(…);** В асемблерному варіанті результат, який необхідно повернути, присвоюється змінним _ACCA, _ACCB, _ACCALO, _ACCAHI і тп. в залежності від оголошеного типу функції.

Виходячи з вищенаведеного матеріалу видно, що надмірне зловживання процедурами та функціями призводить до збільшення асемблерного коду, до переобтяження робочої програми додатковими обчисленнями, а значить до зниження її ефективності. Тому практика використання "тисячі дрібних процедур" призводить до зменшення швидкодії робочої програми. Аналогічний ефект дає використання рекурсії - виклику функції з самої себе. (Класичний приклад рекурсії - обчислення факторіалу: **n!=(n-1)!*n**.) Виконання такого алгоритму безпосередньо, швидко призведе до переповнення оперативної пам’яті, ліміт якого (для AVR-контролерів) відчутний в значній мірі.

Література

1. Боборикин А.В., Липовецкий Г.П. Литвинский Г.В., и др. Однокристальные микроЭВМ. М.: МИКАП ,1994, - 400с.
2. 8-bit Microcontroller with 4K Bytes Programmable Flash. AT89C51. // www.atmel.com
3. XA-S3 XA 16-bit microcontroller // www.philips.com
4. 8-bit Microcontroller with 4K/8K Bytes In-System Programmable Flash. AT90S4434, AT90LS4434,AT90S8535, AT90LS8535. // www.atmel.com
5. 8-bit Microcontroller with 64K/128K Bytes In-System Programmable Flash. ATMega103, AtMega106. // www.atmel.com