АРХІТЕКТУРА І ХАРАКТЕРИСТИКИ AVR-МІКРОКОНТРОЛЕРІВ

1. Характеристики AVR-мікроконтролерів

Перші мікроконтролери AVR були розроблені в дослідницькому центрі Atmel в Норвегії групою інженерів до складу якої входили Alf Bogen та Vergard Wollan. Ініціали їх імен та посилання на тип архітектури (Risc architecture) і сформували назву "AVR". Перший AVR мікроконтролер AT90S1200 був випущений у 1996 – 1997 р.

Сьогодні AVR мікроконтролери за показниками «ціна – швидкодія - енергоспоживання» вважають одними з найкращих серед 8 бітових контролерів з RISC архітектурою. Об'єм їх продажу подвоюється щороку. По інформації, що наведена в огляді фірми Atmel Product Selection Guide, Volume 11, 2015, відзначається, що на сьогоднішній день випущено біля 7 мільярдів AVR мікроконтролерів.

Сфери застосування AVR мікроконтролерів надзвичайно широкі – від найпростіших приладів до складних систем збору обробки інформації та керування, що застосовуються у побуті та промисловості.

Перший офіційний каталог мікроконтролерів фірми Atmel було випущено в 1997 році. До складу цього каталогу ввійшли мікроконтролери першого сімейства Classic. А вже в 1999 році у другому випуску каталогу були приведені дані мікроконтролерів трьох сімейств: Classic, Mega, Tiny.

З 2008 р. фірма Atmel почала серійний випуск нового сімейства XMega 8 бітових AVR мікроконтролерів.

Сьогодні фірмою Atmel виробляються 8, 8/16 та 32 розрядні AVR мікроконтролери. Всього випускається 294 різних типів AVR мікроконтролерів.

Сімейства AVR мікроконтролерів мають наступні загальні властивості:

- це 8 (16- або 32) розрядні RISC мікроконтролери загального призначення з Гарвардською архітектурою з фізично та логічно розділеними шинами даних та адресними просторами пам’яті програм та даних;
- мають вбудовану пам'ять програм Flash – типу, статичну SRAM пам’ять даних та енергонезалежну EEPROM пам’ять даних. Підвищена ємність вбудованої пам’яті даних дозволяє використовувати для розробки прикладних програм мову високого рівня;
- продуктивність мікроконтролера сягає 1 MIPS на частоті 1МГц;
- мають 32 регістри загального призначення, які утворюють файловий регістр. За участю цих регістрів виконуються операції в арифметично логічному пристрої. Таке збільшення кількості регістрів загального призначення дозволяє використовувати їх як для зберігання змінних, так і для виконання операцій над ними, що суттєво підвищує ефективність програм;
- в AVR мікроконтролерах використовується 1-рівневий конвеєр по роботі з пам’яттю програм, який дозволяє на 1 циклі мікроконтролера виконувати поточну команду та робити вибірку коду наступної команди;
- передбачена можливість як зовнішнього так і внутрішньо схемного програмування мікроконтролера;
- пам'ять програм розподілена на два сектори – для прикладної програми (application sector) та для програми завантажувача (boot sector) мікроконтролера. Існує можливість перерозподілу ємності цих сегментів. Сектор завантаження дозволяє реалізувати функцію самопрограмування;
- наявність спеціальних засобів (фюзесів) програмування структури мікроконтролера.

Сучасна класифікація сімейств 8-розрядних AVR мікроконтролерів представлена у каталогах трьома напрямками Tiny, Mega, XMega. Проте на офіційному сайті фірми Atmel визначено більшу кількість сімейств:

- tinyAVR- 38 різних 8 розрядних мікроконтролерів;
- megaAVR – 105 різних 8 розрядних мікроконтролерів;
- AVR XMEGA – 46 різних 8/16 розрядних мікроконтролерів;
- Battery Management MCUs – 5 різних 8 розрядних мікроконтролерів;
- Automotive AVR MCUs – 38 різних 8 розрядних мікроконтролерів;
- 32 біт AVR UC3 - 62 різних 32 розрядних мікроконтролерів.

Вбудоване ядро AVR мікроконтролерів використовується також спільно з FPGA програмованими матрицями в ASIC платформах.

Також існує можливість підключення ядра AVR мікроконтролера при розробці систем на кристалі SoC на базі FPGA матриць. Існує можливість отримання опису ядра з використанням ресурсів сайту www.opencores.org.

Досить давно випускають як спеціалізовані мікросхеми, так і засоби для емуляції таких систем, наприклад ATasicICE POD.

Огляд максимальних ресурсів AVR мікроконтролерів

До ресурсів мікроконтролерів зазвичай включають дані про фізичні характеристики корпусів, діапазони напруги живлення та робочої температури, максимальну робочу частоту, кількість розрядів шини даних, об'єми пам'яті програм, статичної та енергонезалежної пам'яті даних. Вказують на кількість ліній вводу/виводу та дають опис кількості вбудованих контролерів, таких як таймери-лічильники, контролер PTC ємнісної Touch панелі, периферійних контролерів USART, SPI, TWI, USB, PWM, Ethernet, CAN. Вказують дані по групі аналогових контролерів компаратора, аналогово-цифрового та цифро-аналогового перетворювачів. Максимальні ресурси мікроконтролерів змінюються у залежності від типу сімейства.

У табл. 1 відображені максимальні діапазони по ресурсам всіх AVR мікроконтролерів.

Таблиця 1

№	Назва ресурсу	Значення, діапазон
1.	Розрядність процесорного ядра (CPU)	8 / 16 / 32
2.	Кількість виводів корпусу мікроконтролера	6 – 144
3.	Максимальна робоча частота	1-84 МГц
4.	Об'єм вбудованої пам'яті програм FLASH	0,5 – 512 кБайт
5.	Об'єм статичної пам'яті даних SRAM	0,03 - 128 кБайт
6.	Об'єм енергонезалежної пам'яті даних EEPROM	0 – 4096 Байт
7.	Максимальна кількість ліній вводу/виводу	4 – 123
8.	Напруга живлення	0,7 – 25 В
9.	Діапазон робочих температур	-40 - +150 ºС
10.	Кількість каналів Touch Channels (контролер PTC)	0 – 56
11.	Кількість таймерів лічильників (8 / 16 біт)	1 – 10
12.	Сторожовий таймер (WDT)	1
13.	Незалежний RTC генератор 32кГц	0 – 1
14.	Аналоговий компаратор	1 – 8
15.	Температурний сенсор	0 – 1
16.	Розрядність аналогово - цифрового перетворювача	8 – 10 - 12
17.	Кількість каналів аналогово-цифрового перетворювача	4 – 28
18.	Цифро аналоговий перетворювач DAC	0 – 4
19.	Контролер UART	0 – 8
20.	Контролер SPI	0 – 12
21.	Контролер TWI	0 – 4
22.	Контролер USB	0 –Device - OTG
23.	Контролер PWM	0 – 36
24.	Контролер Ethernet	0 – 1
25.	Контролер CAN	0 – 2

Сімейство AVR мікроконтролерів CLASSIC

До першої групи AVR мікроконтролерів, які започаткували сімейство CLASSIC, входило 4 мікроконтролера. Найбільш цікавим був мікроконтролер AT90S8515. Фірма Atmel створювала цей мікроконтролер як перехідний від сімейства MCS-51 до AVR. Типологічно, за розміщенням ліній портів, живлення, скидання та генератора, він повністю повторював

розповсюджений мікроконтролер AT89C51. Це дозволило розробникам без зміни печатних вузлів переключитися на новий тип сімейства мікроконтролерів. Перші моделі мікроконтролерів характеризувалися значною кількістю помилок, які відзначалися в Errata - листах технічної документації. Поступово у нових версіях мікроконтролерів ці помилки виправлялися. Для позначення мікроконтролерів цього сімейства використовувався префікс «AT90». Розвиток цього сімейства характеризувався появою нових мікросхем з розширеними можливостями, в тому числі і спеціалізованих, наприклад для роботи з CAN шиною (AT90CAN), USB шиною (AT90USB), PWM формувачем імпульсів керування трифазових інверторів напруги (AT90PWM).

В цьому сімействі створювалися мікроконтролери як з розширеними можливостями (наприклад введено аналогово цифровий перетворювач в AT90S4433, AT90S8535), так і з обмеженими можливостями, наприклад, AT90S2323, AT90S2343 випущено в корпусі DIP8. По завершенню розвитку цього напрямку до сімейства входило 17 різних типів мікроконтролерів.

Поступово фірма Atmel відійшла від напрямку розвитку сімейства CLASSIC та почала розвивати два інших сімейства - з обмеженими апаратними можливостями AT tiny та розширеними можливостями ATmega. При цьому мікроконтролери сімейства CLASSIC розподілили між новими сімействами.

Сімейство AVR мікроконтролерів tinyAVR

Мікроконтролери сімейства tinyAVR розроблені для систем де суттєвими є розміри та вартість. В початкових мікросхемах цього сімейства зменшували кількість вбудованих ресурсів, у порівнянні з сімейством ATmega. Проте на певному етапі була проведена розробка мікросхем ресурсні показники яких перевищили показники деяких мікроконтролерів ATmega.

Базовими рисами цього сімейства є компактність та потужність споживання. Мікроконтролери мають таку само продуктивність, як і мікроконтролери сімейства ATmega. Мікроконтролери можуть працювати в широкому діапазоні напруги живлення 0,7 – 5,5 В та температури зовнішнього середовища від -40 до 125°С, з робочою частотою до 20 МГц та продуктивністю до 20MIPS. Випускаються у корпусах які мають 6 – 32 виводи. Мають вбудовану FLESH пам'ять програм 0,5-8 кБайт, статичну SRAM та енергонезалежну EEPROM пам'ять даних, супервізор напруги живлення. Характеризуються високим ступенем інтеграції. Кожен контакт мікроконтролера, за винятком ліній подачі живлення, має значну кількість альтернативних функцій. До складу навіть найпростіших мікроконтролерів (TINY5, TINY10), що випускаються у корпусі з 6 виводами (SOT23-6) входять такі складні вузли як аналогово-цифровий перетворювач з 4 входами.

Проте деякі типи мікроконтролерів мають надзвичайно малу ємність SRAM пам'яті даних (TINY4, TINY5), що не дозволяє ефективно використовувати мову Сі для програмування. Розробку прикладних програм у цьому випадку доцільно проводити з використанням мови асемблеру. Деякі мікроконтролери мають у своєму складі підвищуючий стабілізатор напруги, який дає можливість працювати з напругою джерела живлення до 0,7В. У табл. 2 відображені максимальні діапазони значень для ресурсів AVR мікроконтролерів tinyAVR.

Таблиця 2

№	Назва ресурсу	Значення, діапазон
1.	Розрядність процесорного ядра (CPU)	8
2.	Кількість виводів корпусу мікроконтролера	6 – 32
3.	Максимальна робоча частота	4-20 МГц
4.	Об’єм вбудованої пам’яті програм FLASH	0,5 – 16 кБайт
5.	Об’єм статичної пам’яті даних SRAM	0,03 - 1 кБайт
6.	Об’єм енергонезалежної пам’яті даних EEPROM	0 – 512 Байт
7.	Максимальна кількість ліній вводу/виводу	4 – 28
8.	Напруга живлення	0,7 – 5,5 В

		Продовження табл.. 2.2
9.	Діапазон робочих температур	-40 - +125 ºС
10.	Кількість каналів Touch Channels (контролер PTC)	0 – 12
11.	Кількість таймерів лічильників (8 / 16 біт)	1 – 3
12.	Сторожовий таймер (WDT)	1
13.	Незалежний RTC генератор 32кГц	0 - 1
14.	Аналоговий компаратор	1 - 2
15.	Температурний сенсор	0 - 1
16.	Розрядність аналогово - цифрового перетворювача	8 – 10
17.	Кількість каналів аналогово-цифрового перетворювача	4 - 28
18.	Контролер UART	0 - 2
19.	Контролер SPI	0 - 2
20.	Контролер TWI	0 - 1
21.	Контролер PWM	0 - 9

У табл. 3 відображені характеристики деяких мікроконтролерів сімейства tinyAVR.

Таблиця 2.3

Тип	**Виводи**	**Flash (КВ)**	**SRAM (В)**	**EEPROM(В)**	**I/Q**	**ADC**	**Корпус**
TINY 4	4/6	0.5	32	N/A	4		SOT23-6,UDFN-8
TINY 5	4/6	0.5	32	N/A	4	4 x 8bit	SOT23-6,UDFN-8
TINY 9	4/6	1	32	N/A	4		SOT23-6,UDFN-8
TINY 10	4/6	1	32	N/A	4	4 x 8bit	SOT23-6,UDFN-8
TINY 13A	8/10/12	1	64	64	6	4 x 10bit	PDIP-8, SOIC-8, QFN-10,QFN-20
TINY 20	14/15/20	2	128	N/A	10/12	4 x 10bit	SOIC-14, TSSOP-14, WCCSP-12, UFBGA-15, VQFN-20
TINY 24A	14/15/20	2	128	128	12	12 x 10bit	SOIC-14, PDIP-14, UFBGA-15, QFN-20 MLF-20, VQFN-20
TINY 25	8/20	2	128	128	6	4 x 10bit	PDIP-8, SOIC-8, QFN-20,QFN-20
TINY 40	20	4	256	N/A	18	8 x 10bit	SOIC-14, TSOP-20, VQFN-20
TINY 44A	14	4	256	256	12	8 x10 bit	SOIC-14, PDIP-14, UFBGA-15, QFN-20 MLF-20, VQFN-20
TINY 45	8/20	4	256	256	6	4 x 10bit	PDIP-8, SOIC-8, TSOP-8,QFN-20
TINY 48	28/32	8	256	64	24/28	10bit	PDIP-28,QFN-28, QFP-32,QFN-32
TINY 84A	14	8	512	512	12	8 x 10bit	SOIC-14, PDIP-14, UFBGA-15, QFN-20,MLF-20, VQFN-20

Сімейство AVR мікроконтролерів megaAVR

На іншому полюсі AVR, мікроконтролерів за рівнем інтеграції і можливостей, знаходиться сімейство megaAVR**.** Для мікроконтролерів цієї групи характерні наступні властивості:

- великий об'єм Flash пам'яті програм від 4 до 256 Кбайт);
- режим самопрограмування, забезпечений можливістю програмування з Boot-сектору пам’яті програм за допомогою програми-завантаження;
- вбудований апаратний помножувач, що підтримує множення 8 розрядних чисел;
- розширені набори вбудованої периферії;
- широкий набір спеціальних мікроконтролерних функцій, у тому числі: до шести режимів енергозбереження і можливість програмної установки тактової частоти;
- розширення системи команд до 130 – 133;
- організація в нових моделях інтерфейсу граничного сканування (IEEE 1149. 1/ JTAG), що підтримує вбудоване налагодження і забезпечує ще один шлях програмування Flash і EEPROM пам'яті, фюзесів конфігурації мікроконтролера та бітів блокування вмісту пам’яті;
- спеціальні мікроконтролерні функції, що забезпечують високу усталеність роботи апаратних і програмних засобів при випадкових змінах напруги живлення;
- ємність вбудованої пам’яті програм знаходиться у межах від 4 до 256 кБайт.

Мікроконтролери цього сімейства характеризуються типовою продуктивністю виконання програм 1,0 MIPS/МГц можуть працювати на частоті генератора до 20МГц.. Області застосування мікроконтролерів цієї групи – в системах загального призначення для реалізації функцій збору інформації, її обробки та керування периферійним обладнанням.

У табл. 4 відображені максимальні діапазони значень для вбудованих ресурсів AVR мікроконтролерів megaAVR.

Таблиця 4

№	Назва ресурсу	Значення, діапазон
1.	Розрядність процесорного ядра (CPU)	8
2.	Кількість виводів корпусу мікроконтролера	20 - 100
3.	Максимальна робоча частота	16-20 МГц
4.	Об’єм вбудованої пам’яті програм FLASH	4 – 256 кБайт
5.	Об’єм статичної пам’яті даних SRAM	0,25 - 16 кБайт
6.	Об’єм енергонезалежної пам’яті даних EEPROM	256 – 4096 Байт
7.	Максимальна кількість ліній вводу/виводу	19 - 86
8.	Напруга живлення	1,8 – 5,5 В
9.	Діапазон робочих температур	-40 - +125 ºС
10.	Кількість каналів Touch Channels (контролер PTC)	8 - 32
11.	Кількість таймерів лічильників (8 / 16 біт)	1 - 6
12.	Сторожовий таймер (WDT)	1
13.	Незалежний RTC генератор 32кГц	0 - 1
14.	Аналоговий компаратор	1 - 4
15.	Температурний сенсор	0 - 1
16.	Розрядність аналогово - цифрового перетворювача	10
17.	Кількість каналів аналогово-цифрового перетворювача	8 - 16
18.	Цифро аналоговий перетворювач DAC	0 - 1
19.	Контролер UART	0 - 4
20.	Контролер SPI	1 - 5
21.	Контролер TWI	0 - 2
22.	Контролер USB	0 –Device - OTG
23.	Контролер PWM	3 - 15
24.	Контролер CAN	0 - 1

У табл. 5 відображені характеристики деяких мікроконтролерів сімейства megaAVR.

Таблиця 5

Серія	Виводи	Flash (KB)	SRAM (KB)	EEPROM (KB)	I/O	ADC	Корпус
MEGA 16A	40/44	16	1	0.5	32	8 x 10bit	PDIP-40, MLF-44, QFP-44
MEGA 32A	40/44	32	2	1	32	8 x 10bit	PDIP-40, MLF-44, QFP-44

Продовження табл.. 2.5

MEGA 48PB	32	4	0.5	0.25	27	8 x 10bit	MLF-32, QFP-32
MEGA 64A	64	64	2	N/A	53	8 x 10bit	MLF-64, QFP-64
MEGA 88PB	32	8	1	0.5	23	8 x 10bit	MLF-32, QFP-32
MEGA 162	40/44	16	1	0.5	35	N/A	PDIP-40,MLF-44, QFP-44
MEGA 164PA	40/44/49	16	1	0.5	32	8 x 10bit	PDIP-40,MLF-44, QFP-44
MEGA 168PB	32	16	1	0.5	27	8 x 10bit	MLF-32, QFP-32
MEGA 169PA	64	16	1	0.5	54	8 x 10bit	MLF-64,QFP-64, FN-64
MEGA 324PB	44	32	2	1	39	8 x 10bit	MLF-44, QFP-44
MEGA 328PB	32	32	2	1	27	8 x 10bit	MLF-32, QFP-32
MEGA 329PA	64	32	2	1	54	8 x 10bit	MLF-64, QFP-64
MEGA 640	100	64	8	4	86	16 x 10bit	BGA-100, QFP-100
MEGA 344PA	40/44	64	4	2	32	8 x 10bit	MLF-64, QFP-64
MEGA 349P	64	64	4	2	54	8 x 10bit	PDIP-40, MLF-44, QFP-44
MEGA 1280	100	128	8	4	86	16 x 10bit	BGA-100, QFP-100
MEGA 1281	64	128	8	4	54	16 x 10bit	MLF-64, QFP-64
MEGA 1284	40/44	128	16	4	32	8 x 10bit	PDIP-40, MLF-44, QFP-44
MEGA 2560	100	256	8	4	86	16 x 10bit	BGA-100, QFP-100
MEGA 2561	64	256	8	4	54	8 x 10bit	MLF-64, QFP-64
MEGA 3290P	100	32	2	1	69	8 x 10bit	QFP-100
MEGA 6490P	100	64	4	2	69	8 x 10bit	QFP-100
MEGA 8515	40/44	8	0.5	0.5	35	N/A	PDIP-40,MLF-44, QFP-44
MEGA 8535	40/44	8	0.5	0.5	32	8 x 10bit	PDIP-40,MLF-44, QFP-44

Сімейство AVR мікроконтролерів Xmega

Мікроконтролери AVR сімейства Xmega відносяться до 8/16 розрядних та характеризуються вищою складністю та ступенем інтеграції,

у порівнянні з megaAVR. Їх доцільно використовувати в системах загального призначення для яких суттєвими є складність системних вимог та швидкість математичної обробки сигналів.

Для цих мікроконтролерів характерні наступні властивості:

- великий обсяг вбудованої FLESH пам'яті програм від 16 до 384 кБайт;
- типова продуктивність 1,0 MIPS/МГц та максимальна робоча частота до 32 МГц;
- наявність аналогових пристроїв обробки інформації з підвищеною точністю (аналогово цифровий та цифро аналоговий перетворювачі 12 бітні) та швидкодією. Для аналогово цифрового перетворювача максимальна швидкодія досягає 4 MSPS;
- система обробки подій спрощує взаємодію периферійних ресурсів мікроконтролера. Всі периферійні пристрої можуть використовувати механізм прямого доступу до пам'яті (DMA);
- високий рівень інтеграції, наприклад, можлива реалізація до 32 каналів PWM, до 8 вбудованих контролерів UART, можливість використання USB контролера;
- високий рівень кількості виконуваних паралельно задач. Система переривань охоплює до 122 джерел;
- значна кількість вбудованих 16 бітних таймерів (до 8);
- наявність вбудованих криптоблоків алгоритмів AES (Advanced Encryption Standard) та DES (Data Encryption Standard);
- на відміну від контролерів megaAVR забезпечена сумісність різних мікроконтролерів по програмному коду. Це дає можливість створення бібліотек функцій та використання їх в різноманітних проектах.

Мікроконтролери сімейства Xmega у залежності від складності поділяються на 5 серій – A, B, C, D, E. Найбільш складними та насиченими апаратними ресурсами є мікроконтролери A групи. Їх

доцільно використовувати у складних пристроях які характеризуються найбільш жорсткими функціональними вимогами та швидкодією. Мікроконтролери B серії більш прості проте до їх складу входить інтегрований контролер рідкокристалічного дисплею. Серія C включає мікроконтролери початкового рівня, проте до їх складу входить повношвидкісний USB порт. Серія D включає мікроконтролери початкового рівня які характеризуються малим енергоспоживання. Вони рекомендуються фірмою Atmel для технічних рішень критичних з точки зору споживання енергії. Серія E характеризується найменшим корпусом. Мікроконтролери цієї серії призначені для використання в системах із жорсткими вимогами до габаритів. Загальні характеристики A-E серій Xmega мікроконтролерів приведено у табл. 6.

Таблиця 6

Серія	**Виводи**	**Flash (KB)**	**SRAM (KB)**	**EEPROM (KB)**	**I/O**	**ADC**	**Корпус**
A1U	100	64/128	4/8	2	78	2-16 x 12bit 2msps	TQFP-100, BGA-100, VFBGA-100
A2U	64	64/128 192/256	4/8/16	2/4	50	2-16 x 12bit 2msps	TQFP-64, QFN-64
A4U	44	16/32 64/128	2/4/8	1/2	34	12 x 12bit 2msps	QFP-44, QFN-44, VFBGA-49
B1	100	64/128	4/8	2	53	2-8 x 12bit	TQFP-100, VFBGA-100
B3	64	64/128	4/8	2	36	1-8 x 12bit	TQFP-64, QFN-64, DRQFN-64
C3	64	32/64 128/192 256/384	2/4/8/16	4/8/16	50	1-16 x 12bit	TQFP-64, QFN-64
C4	44	16/32	2/4	1	34	1-16 x 12bit	TQFP-44, QFN-44, VFBGA-49
D3	64	32/64 128/192 256/38	4/8 16/32	1/2/4	50	1-16 x 12bit	TQFP-64, QFN-64
D4	44	16/32 64/128	1/2/4/8	1/2/4	34	1-12 x 12bit	TQFP-44, VFBGA-49
E5	32	8/16/32	1/2/4	0.5/1	26	1-16 x 12bit	QFP-44, QFN-44,UQFN-49

У табл. 7 приведено параметри деяких мікроконтролерів Xmega.

Таблиця 7.

Тип	Status (b)	Flash (KB)	Boot code (Bytes)	EEPROM (KB)	SRAM (KB)	I/O pins	16-bit Timers	PWM (channel)	SPI	TWI(I2C)	USART	12-bit ADC (ch.)	12-bit DAC (ch.)	Analog Comp.	Interrupts	Interrupts Ext.
ATxmega 64A1	I	64	4	2	4	78	8	24	4	4	8	2x8	2x2	4	122	78
ATxmega 128A1	I	128	8	2	8	78	8	24	4	4	8	2x8	2x2	4	122	78
ATxmega 192A1	I	192	8	4	16	78	8	24	4	4	8	2x8	2x2	4	122	78
ATxmega 256A1	I	256	8	4	16	78	8	24	4	4	8	2x8	2x2	4	122	78
ATxmega 384A1	I	384	8	4	34	78	8	24	4	4	8	2x8	2x2	4	102	78
ATxmega 64A3	I	64	4	2	4	50	7	22	3	2	7	2x8	1x2	4	102	50
ATxmega 128A3	I	128	8	2	8	50	7	22	3	2	7	2x8	1x2	4	102	50
ATxmega 192A3	I	192	8	4	16	50	7	22	3	2	7	2x8	1x2	4	102	50
ATxmega 256A3B	I	256	8	4	16	49	7	22	2	2	6	2x8	1x2	4	102	49
ATxmega 256A3	I	256	8	4	16	50	7	22	3	2	7	2x8	1x2	4	102	50
ATxmega 16A4	F	16	4	1	2	34	5	16	2	2	5	1x12	1x2	2	77	34
ATxmega 32A4	F	32	4	1	4	34	5	16	2	2	5	1x12	1x2	2	77	34
ATxmega 64A4	F	64	4	2	4	34	5	16	2	2	5	1x12	1x2	2	77	34
ATxmega 128A4	F	128	4	2	8	34	5	16	2	2	5	1x12	1x2	2	77	34
ATxmega 64D3	F	64	4	2	4	50	5	18	2	1	3	1x16		2	67	50
ATxmega 128D3	F	128	8	2	8	50	5	18	2	1	3	1x16		2	67	50
ATxmega 192D3	F	192	8	2	16	50	5	18	2	1	3	1x16		2	67	50
ATxmega 256D3	F	256	8	4	16	50	5	18	2	1	3	1x16		2	67	50
ATxmega 16D4	F	16	4	1	2	34	4	14	2	1	2	1x12		2	55	34
ATxmega 32D4	F	32	4	1	4	34	4	14	2	1	2	1x12		2	55	34

Всі МК містять 16 розрядний RTC, окрім ATxmega 256A3B, який має 32 розрядний; Vcc Range – 16 - 32V, Clock Speed - 32 MHz. МК ATxmegaxxAx мають 8 каналів Event і 4 канали DMA, ATxmegaxxDx – 4 канали Event і не мають DMA.

Сімейство AVR мікроконтролерів Battery Management MCUs

Мікроконтролери цього сімейства доцільно використовувати в системах які працюють спільно з літій-іонними батареями. Вони дозволяють визначати рівень заряду акумуляторної батареї, яка має від 1 до 4 елементів. На базі цих мікроконтролерів може бути створена система з лічильником кулонів, захистом від короткого замикання та перегріву. Можливе балансування рівнів заряду окремих акумуляторів в батареї. Особливістю мікроконтролерів є те, що вони можуть працювати в широкому діапазоні робочої напруги від 1,8 до 25 В. Мають досить велику ємність пам’яті програм від 8 до 40 кБайт, типову для AVR мікроконтролерів продуктивність 1,0 MIPS/МГц та максимальну робочу частоту до 8 МГц. Мікроконтролери мають засоби, які дозволяють проводити аутентифікацію акумуляторів, що унеможливлює використання в системах не оригінальних батарей у випадках такої необхідності.

Сімейство AVR мікроконтролерів Automotive AVR MCUs

Це сімейство 8 бітових AVR мікроконтролерів розроблено для використання в автомобільній електроніці. Мікроконтролери можуть працювати в розширеному діапазоні температур від – 40 до 150°C, вирізняються широким набором периферійних функцій та характеризуються підвищеною надійністю. Мають типову для AVR мікроконтролерів продуктивність 1,0 MIPS/МГц та максимальну робочу частоту до 32 МГц.

Фірма пропонує готові рішення по технології SIP (System-in-Package, "система в корпусі"), які поєднують AVR мікроконтролер, інтерфейси LIN

та CAN, стабілізатор напруги та супервізор напруги, блок що дозволяє проводити обчислення з плаваючою комою (FPU), механізм захисту коду FlashVault™, інтерфейси Ethernet, USB OTG. До складу цього сімейства входять мікроконтролери решти сімейств, наприклад, AT90CAN32, ATtiny44, ATmega88, ATxmega 6403, AT32UC3C0512.

На базі таких мікроконтролерів можливе створення компонентів автомобіля для кузова та салону, захисту та доступу до автомобіля, інформаційно-розважальної системи на базі сенсорних рішень, компонентів для моторного відсіку.

2. Архітектура AVR мікроконтролера Mega 16

Узагальнена архітектура AVR мікроконтролера Mega 16 наведена на рис. 1.

Мікроконтролер містить:

- Flash Program Memory – флеш-пам‘ять програм (ПЗП);
- Program Counter - програмний лічильник;
- Instruction Register – регістр інструкцій;
- Instruction Decoder – пристрій декодування інструкцій;
- Control Lines – шина керування мікроконтролера;
- Data Bus 8-bit – шина даних мікроконтролера;
- EEPROM – енергонезалежна пам‘ять даних з байтовим доступом;
- Data SRAM – ОЗП статичного типу;
- ALU - арифметико-логічний пристрій (АЛП);
- 32x8 General Purpose Register - регістри загального призначення які формують файловий регістр;
- Status and Control – регістр статусу мікроконтролера (SREG);
- Interrupt unit - модуль переривань;
- SPI (Serial Peripheral Interface) - послідовний периферійний

інтерфейс, та послідовний інтерфейс програмування;

- WDT (Watchdog Timer) - сторожjвий таймер;
- Analog Comparator - аналоговий компаратор;
- I/O lines – 8 бітові порти введення-виведення;
- I/O module 1 - I/O module n - модулі вводу виведення інформації до яких належать таймери, аналогово цифровий перетворювач (АЦП), контролер TWI, UART універсальний асинхронний приймач-передавач.

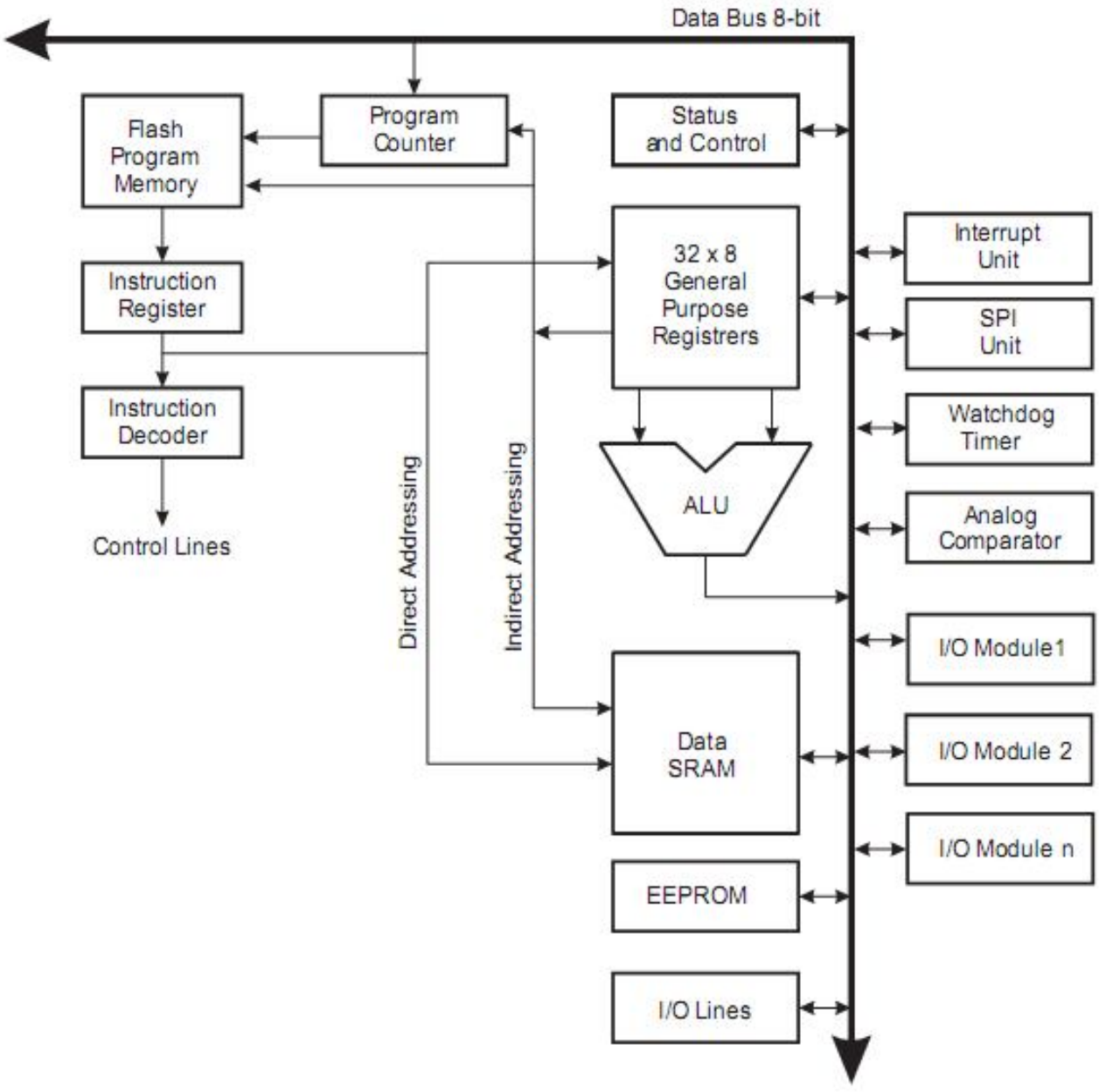


Рис. 1 Архітектура *AVR* мікроконтролера Mega 16

Основою архітектури ядра AVR мікроконтролера є Гарвардський процесор, регістрова пам’ять (регістри загального призначення та регістри введення-виведення), пам'ять програм та пам'ять даних. Склад та кількість периферійних пристроїв (таймерів, портів введення-виведення, послідовних інтерфейсів, АЦП) залежить від конкретної моделі мікроконтролера.

Гарвардський процесор реалізує повний логічний і фізичний розподіл не тільки адресних просторів, але й інформаційних шин для звертання до пам'яті програм і до пам'яті даних, причому способи адресування і доступу до цих масивів пам'яті також різні. Подібна побудова забезпечує істотне підвищення продуктивності мікроконтролера. Процесор працює одночасно як із пам'яттю програм, так і з пам'яттю даних; розрядність шини пам'яті програм розширена до 16 біт. В AVR мікроконтролерах використовується технологія однорівневої конвеєризації, унаслідок чого цикл "вибірка - виконання" команди помітно скорочений.

Послідовність виконання команд в конвеєрі та обробка їх а АЛП наведено на рис. 2 та 3 відповідно.

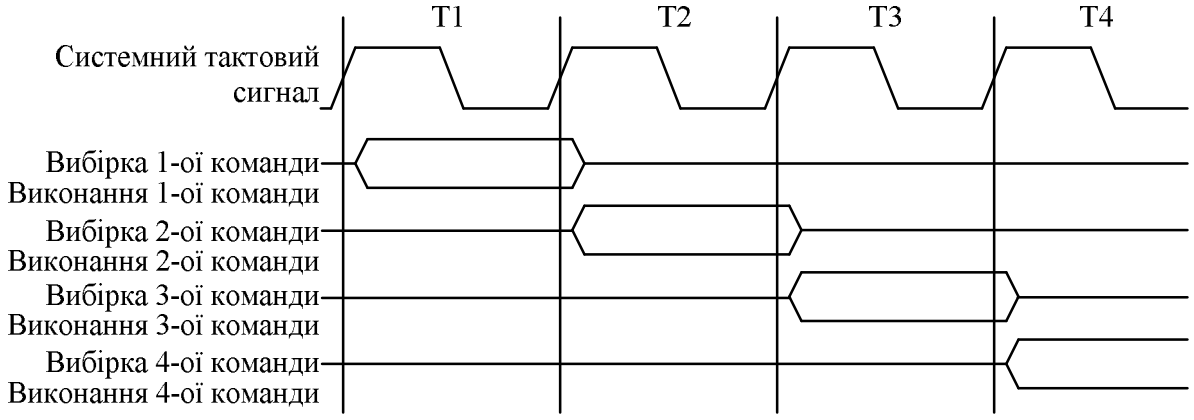


Рис. 2 Послідовність виконання команд в конвеєрі

Під час першого машинного циклу T1 (рис. 2) відбувається вибірка команди з пам'яті програм і її декодування. Під час другого циклу T2 ця

команда виконується, а паралельно відбувається вибірка й декодування другої команди, і так далі.

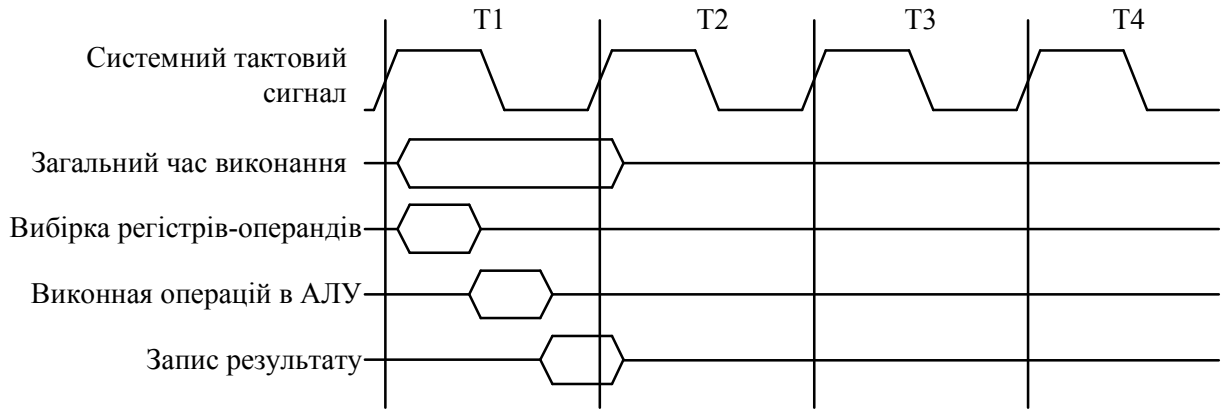


Рис. 3 Функціонування АЛП

Завдяки підключенню АЛП безпосередньо до регістрового файлу він виконує одну команду (читання вмісту двох регістрів, виконання операції й запис результату в регістр - приймач) за один такт, як показано на рис. 3.

У результаті фактичний час виконання кожної команди виходить рівним одному машинному циклу. Таке рішення дозволяє досягати продуктивності до 1 MIPS на МГц. Продуктивність AVR мікроконтролера суттєво підвищується також за рахунок того, що АЛП може виконувати операції з вмістом 32 регістрів загального призначення. Ці регістри можуть виконувати функції як регістрів загального призначення, так і регістрів ОЗП. Це суттєво скорочує часові втрати які пов’язані з пересиланням інформації.

Для порівняння, у мікроконтролерів сімейства MCS-51 коротка команда виконується за 12 тактів генератора (1 машинний цикл), протягом якого процесор послідовно зчитує код операції і виконує її. У PIC-контролерах фірми Microchip, де вже реалізований конвеєр, коротка команда виконується протягом 8 періодів тактової частоти (2 машинних цикли).

Розмір лічильника команд становить від 9 до 12 розрядів, у залежності від обсягу пам'яті. При цьому лічильник команд недоступний програмі безпосередньо (як регістр). При нормальному виконанні програми вміст лічильника команд автоматично збільшується на 1 (або на 2, залежно від команди) у кожному машинному циклі. Цей порядок порушується при виконанні команд переходу, виклику й повернення з підпрограм, а також при виникненні переривань.

Програмна модель AVR мікроконтролера

Програмна модель зображена на рис. 4.

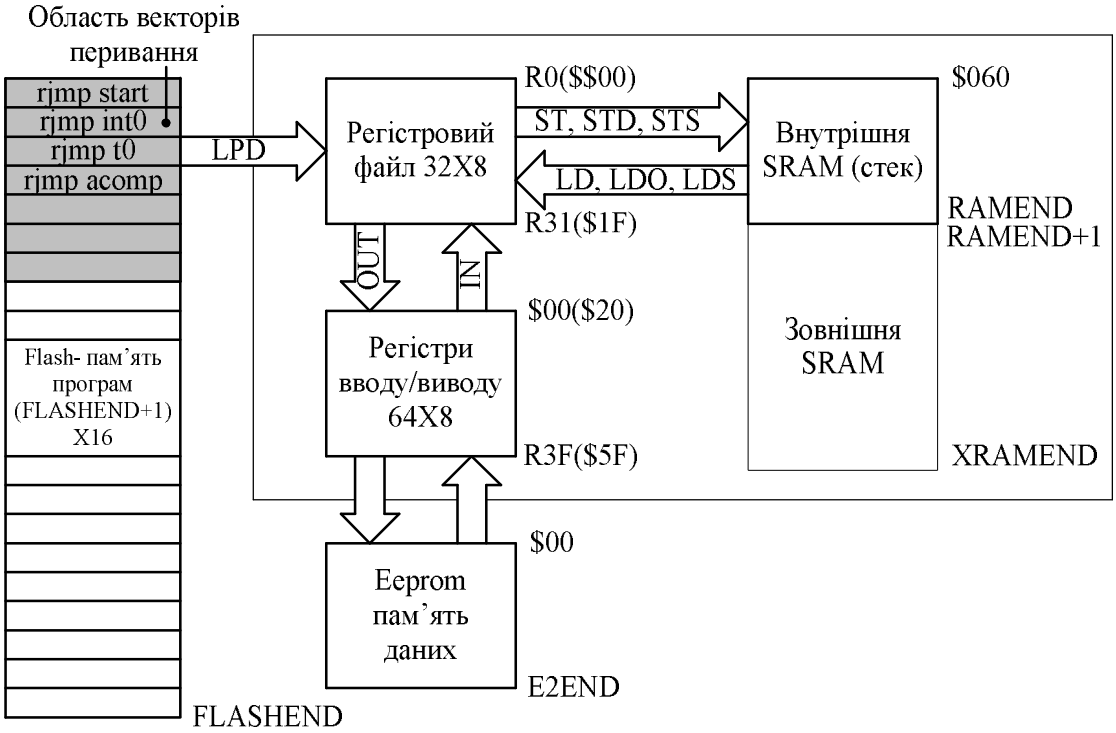


Рис. 4 Програмна модель AVR-мікроконтролерів

Являє собою сукупність програмно доступних ресурсів, які використовуються наступним чином:

- для збереження робочої програми та програми завантаження мікроконтролера (FLASH пам'ять програм, Application та Boot сектори);

- для збереження динамічних змінних (внутрішня та зовнішня SRAM);
- для збереження енергонезалежних змінних (EEPROM пам'ять даних);
- регістрів вводу/виводу, за допомогою яких здійснюється керування вбудованими ресурсами мікроконтролера;
- регістрового файлу, що використовується для обробки інформації в АЛП.

Арифметико-логічний пристрій

До складу мікроконтролера ATmega16 входить 8-бітовий арифметико-логічний пристрій (АЛП), який безпосередньо пов’язаний з 32 регістрами, що входять до складу регістрового файлу. Арифметико-логічний пристрій здатен виконувати три типи операцій: арифметичні, логічні та бітові. Операції можуть виконуються між елементами регістрового файлу, або цими регістрами та константами. АЛП може виконувати операції над числами представленими у трьох форматах: бітовому, байтовому та двобайтовому (операції додавання та віднімання). Більшість операцій виконуються за один такт генератора. Особливістю АЛП є присутність вбудованого апаратного помножувача байтових чисел.

В результаті виконання кожної операції в АЛП модифікується вміст регістру статусу SREG. Регістр SREG відноситься до групи регістрів введення/виведення та доступний для читання та запису інформації.

На рис.2.5 приведено бітову упаковку регістру SREG.

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

Рис 5

Біти цього регістру мають наступне призначення:

- I (SREG.7) – загальний дозвіл переривань.

Мікроконтролер має дворівневу систему дозволів переривань. Індивідуальні дозволи переривань встановлюються у відповідних регістрах масок. Для встановлення загального дозволу роботи системи переривань використовується біт «I». Для дозволу переривань необхідно встановити цей біт «I = 1». У разі виконання підпрограми переривань біт апаратно скидається «I = 0». Апаратне поновлення (I = 1) відбувається після виконання команди повернення (RETI) з підпрограми переривань. Такий алгоритм маніпуляцій бітом «I» забороняє використання алгоритмів з вкладеними перериваннями. Проте можливий програмний дозвіл вбудованих переривань, за рахунок повторного примусового програмного встановлення дозволу (I = 1) у програмах переривання;

- T (SREG.6) – комірка зберігання біту, що копіюється.

Існує дві інструкції, які дозволяють оперативно зберегти (BST) або передати (BLD) біти до будь якого файлового регістру;

- H (SREG.5) – прапор половинного переносу.

Цей біт встановлюється (H = 1), якщо відбувся перенос, або позика при операціях з тетрадами. Цей біт дозволяє працювати з двійково-десятково (BCD) упакованими числами;

- S (SREG.4) – прапор знаку.

Цей прапор визначається як результат операції XOR між прапорами від'ємного результату «N» та переповнення розрядної сітки знакового результату «V». Зокрема прапор встановлюється (S=1), якщо результат арифметичної операції менше нуля;

- V (SREG.3) – прапор переповнення додаткового коду.

Прапор встановлюється (V = 1) при переповненні розрядної сітки знакового результату. Використовується при роботі з числами із знаком;

- N (SREG.2) – прапор від'ємного результату.

Прапор встановлюється (N = 1) якщо результат арифметичної або логічної операції над числами із знаком негативний. Тобто «7» розряд результату операції дорівнює «1»;

- Z (SREG.1) – прапор нуля.

Прапор встановлюється (Z = 1) якщо результат арифметичної або логічної операції дорівнює нулю;

- C (SREG.0) – прапор переносу.

Прапор встановлюється (C = 1) якщо результат арифметичної або логічної операції виходить за межі байту.

Особливістю регістру статусу є те, що він автоматично не зберігається в стеку при виконанні звичайних підпрограм та підпрограм переривань. Якщо в таких підпрограмах використовуються операції з АЛП, то вміст регістру статусу необхідно зберігати з використанням програмних засобів..

Ознаки результату операції використовуються для виконання подальших арифметико-логічних операцій або команд умовних переходів.

Після включення живлення, а також після скидання мікроконтролера, в залежності від налаштування фюзесів, до лічильника програм автоматично завантажується початкова адреса Application сектору (0x000) або Boot сектору (розміщення та об’єм можуть модифікуватися за допомогою фюзесів). За цією адресою розташовується команда переходу до фрагменту ініціалізації (Application сектор), або програми завантаження (Boot сектор).

При виникненні переривання в лічильник команд завантажується адреса відповідного вектора переривання. Область векторів переривань може розташовуватися, в залежності від налаштувань мікроконтролера, як в Application, так і в Boot секторах. Якщо в програмі використовуються переривання, за адресами векторів повинні розміщатися команди переходу до відповідних підпрограм обробки переривань.

3. Схеми синхронізації та скидання мікроконтролера

Система формування тактових імпульсів мікроконтролера

Структурна схема системи формування тактових імпульсів мікроконтролера ATmega16 зображена на рис. 6.

У мікроконтролера ATmega16 може формуватися 5 тактових послідовностей:

- CLK_{CPU} – базова тактова послідовність, що забезпечує роботу арифметико-логічного пристрою та роботу з пам'яттю мікроконтролера;
- CLK_{IO} – тактова послідовність, що використовується модулями вводу/виводу, таймерами, послідовними інтерфейсними блоками, системою переривань мікроконтролера;
- CLK_{FLASH} – тактова послідовність, що використовується для керування роботою Flesh інтерфейсу;
- CLK_{ADC} – тактова послідовність, що використовується аналогово-цифровим перетворювачем (АЦП). Введення окремої лінії тактування дозволяє заблокувати решту тактових послідовностей для зменшення впливу завад при роботі АЦП;
- CLK_{ASY} – асинхронна тактова послідовність, що формується таймером/лічильником Т2 та дозволяє виконувати ряд програмних завдань в режимі реального часу, навіть при заблокованій послідовності CLK_{CPU};

Існує можливість блокування тактових послідовностей, що забезпечує зменшення енергоспоживання мікроконтролера.

У мікроконтролера mega16 існує 7 джерел тактових послідовностей. П'ять з цих джерел формують базові послідовності, що використовуються безпосередньо при виконанні програми. До них відносяться наступні джерела:

- зовнішній RC- генератор;
- зовнішня тактова послідовність;

- кристалічний резонатор;
- низькочастотний кристалічний резонатор;
- внутрішній калібрований RC-генератор.

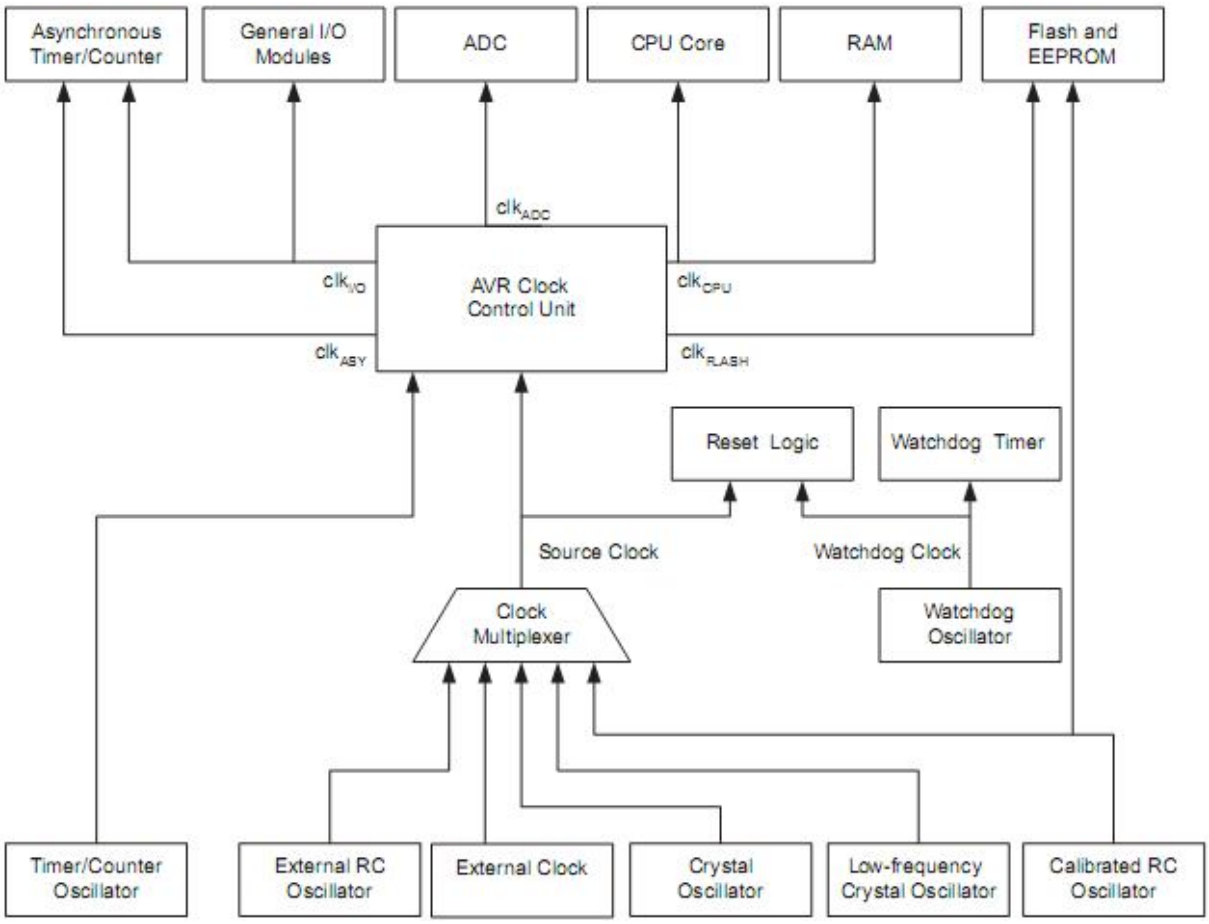


Рис. 6 Система формування тактових імпульсів мікроконтролера ATmega16

Додаткові два джерела є асинхронними, по відношенню до генераторів базових послідовностей, та можуть застосовуватись для формування вторинних функцій мікроконтролера:

- асинхронний генератор тактової послідовності таймера/лічильника T2. Використовує незалежний кварцовий резонатор. Цей генератор адаптовано для частоти резонатора 32768Гц;
- незалежний асинхронний генератор сторожового таймера.

Дистрибутив мікроконтролера поступає від виробника налаштованим для роботи з внутрішнім каліброваним RC- генератором частотою 1МГц.

Підключення певних джерел тактових сигналів проводиться на етапі програмування мікроконтролера за рахунок встановлення фюзесів CKSEL0 – CKSEL3 та SUT0 – SUT1. На рис. 2.7 приведено заставку програматора PonyProg, що відповідає режиму програмування біт конфігурації та захисту.

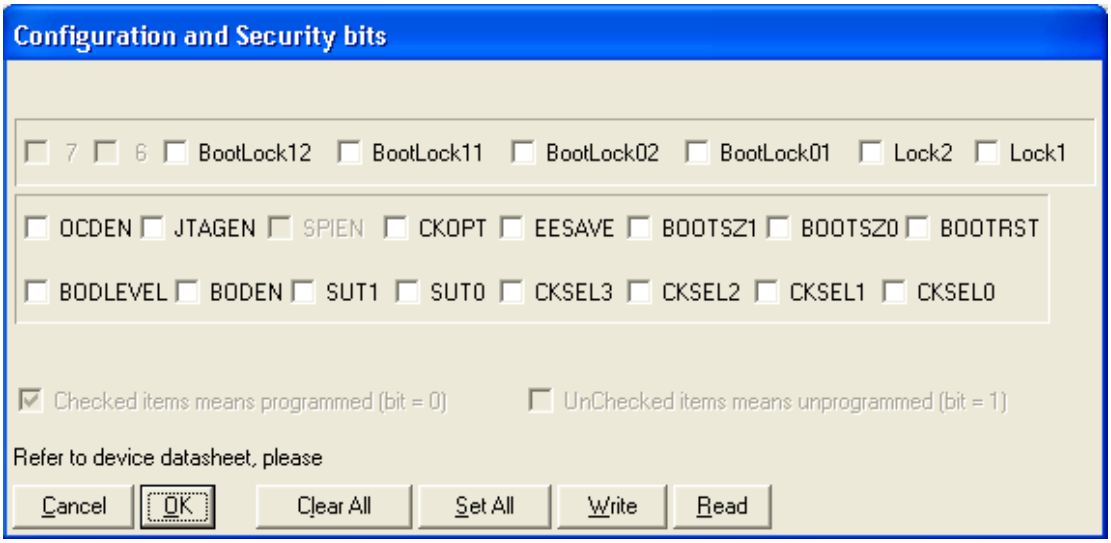


Рис 7

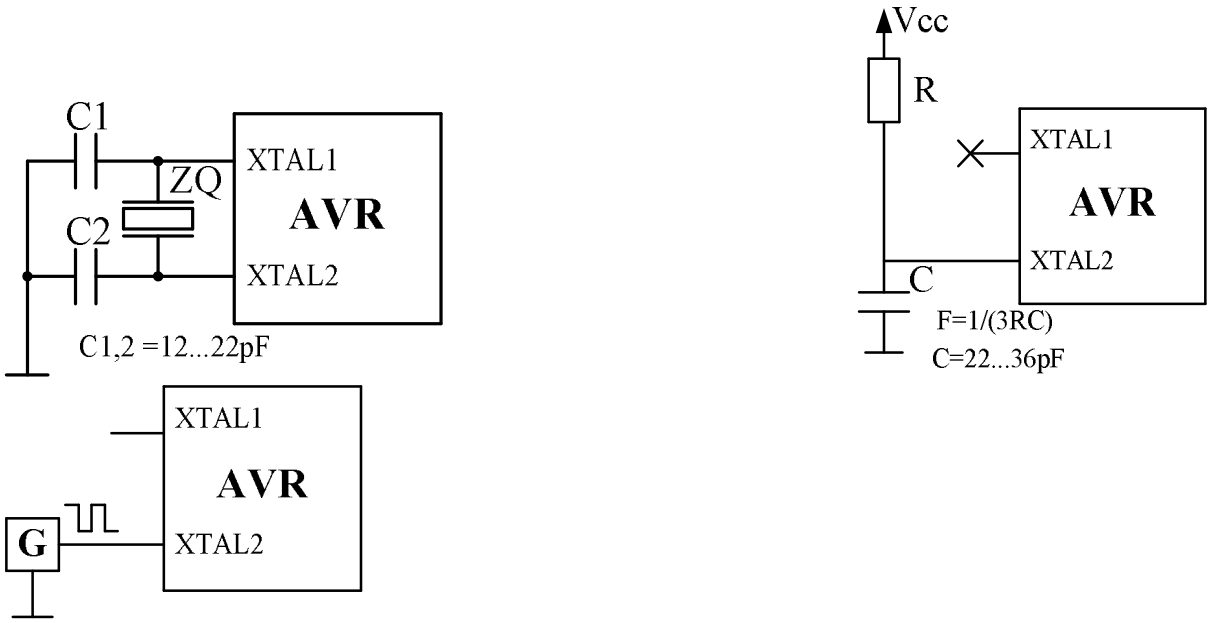
У табл. 8 приведено відповідність фюзесів програмування CKSEL0 – CKSEL3 відповідним джерелам тактових сигналів.

Таблиця 8

ДЖЕРЕЛА ТАКТОВИХ СИГНАЛІВ	CKSEL3...0
Зовнішній кристалічний / керамічний резонатор	1111 - 1010
Зовнішній низькочастотний кварц	1001
Зовнішній *RC* генератор	1000 - 0101
Внутрішній *RC* генератор	0100 - 0101
Зовнішнє джерело сигналів	0000

Схеми підключення джерел тактових імпульсів наведено на рис. 8. Конкретні значення фюзесів CKSEL0 – CKSEL3 та SUT0 – SUT1 визначаються у технічному описі мікроконтролера, згідно з обраним джерелом тактових імпульсів та режимом роботи.

У разі вибору зовнішнього керамічного/кристалічного резонатора значення фюзесів CKSEL0 – CKSEL3 обирають згідно табл. 9.



а) б) с)

Рис. 8. Схеми синхронізації (а - зовнішній кварцовий резонатор, б –RC ланцюг, в - зовнішній тактовий генератор

Таблиця 9

CKOPT	**CKSEL3...1**	**Діапазон частот (МГц)**	**Діапазон номіналів C1 та C2 (пФ)**
1	101[1]	0.4 – 0.9	-
1	110	0.9 – 3.0	12 – 22
1	111	0.3 – 8.0	12 – 22
0	101, 110, 111	1.0 ≤	12 – 22

Біти SUT0 – SUT1 та CKSEL0 обираються згідно табл. 10.

Таблиця 10

CKSEL0	**SUT1…0**	**Час запуску після виключення або режиму енергозбереження**	**Додаткова затримка після перезапуску Vcc = 5.0 V**
0	00	258 CK[1]	4.1 ms
0	01	258 CK[1]	65 ms
0	10	1K CK[2]	-
0	11	1K CK[2]	4.1 ms
1	00	1K CK[2]	65 ms
1	01	16K CK	-
1	10	16K CK	4.1 ms
1	11	16K CK	65 ms

При роботі з внутрішнім *RC*-генератором номінальне значення частоти встановлюється у відповідності з даними табл. 11.

Таблиця 11

CKSEL3...0	Номінальне значення частоти (МГц)
101[1]	1.0.
0010	2.0
0011	4.0
0100	8.0

За допомогою калібрувального регістру OSCCAL встановлене значення частоти внутрішнього *RC*-генератора можна корегувати. На рис. 9 приведено бітову упаковку цього регістру, а у табл. 11 вказані орієнтовні значення вмісту цього регістру для ряду робочих частот.

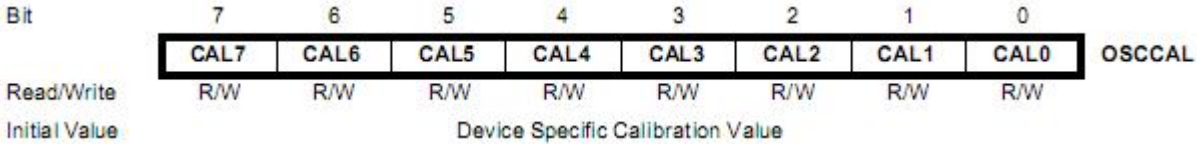


Рис. 9

Таблиця 11

Значення OSCCAL	Мінімальна частота у відсотках від номінальної частоти (%)	Максимальна частота у відсотках від номінальної частоти (%)
$00	50	100
$7F	75	150
$FF	100	200

Система скидання

Система скидання забезпечує початкове дистрибутивне налаштування основних вбудованих ресурсів мікроконтролера.

Розрізняють два типа скидання мікроконтролера – холодне та гаряче.

При реалізації холодного скидання регістри керування мікроконтролера переводяться в дистрибутивний стан. У вказівник пам’яті програми заноситься адреса вектора скидання. У регістри SRAM пам’яті даних заноситься довільне значення. Такий тип скидання використовується при включенні мікроконтролера.

При реалізації гарячого скидання мікроконтролер знаходиться у ввімкненому стані. Як і в попередньому випадку регістри керування мікроконтролера переводяться в дистрибутивний стан. Проте зберігається інформація у регістрах SRAM пам’яті. Структурну схему системи скидання зображено на рис. 10.

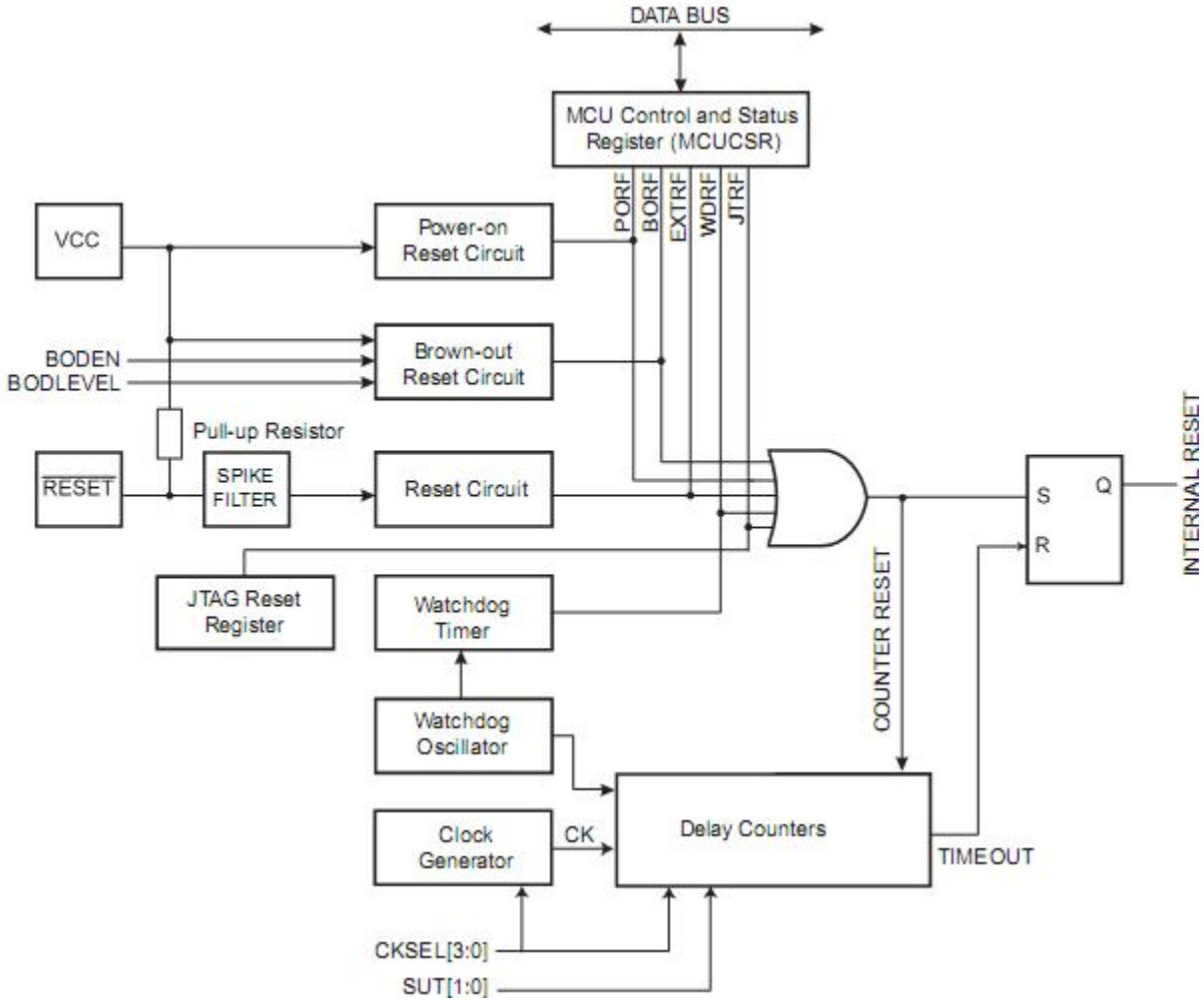


Рис 10

Адреса вектора скидання визначається значенням фюзесу BOOTRST (рис. 7). Якщо цей фюзес не запрограмований, то вектор скидання

розташовується на початку пам’яті програм (за адресою 0x0000). У випадку його програмування розташування вектора скидання визначається двома фюзесами, що визначають розміри BOOT- сектору – BOOTSZ0 та BOOTSZ1. Можливі чотири варіанти налаштування мікроконтролера Mega16 з розташуванням вектора скидання за адресами 0x1C00, 0x1E00, 0x1F00, 0x1F80.

Джерела скидання

Мікроконтролер ATmega16 має 5 незалежних джерел скидання:

- скидання при подачі напруги джерела живлення (Power-on-Reset);
- зовнішнє скидання;
- скидання, що обумовлене дією сторожового таймера;
- скидання, що обумовлене дією супервізора напруги живлення;
- скидання по інтерфейсу JTAG.

Скидання при подачі напруги живлення

На рис 11 зображено часові діаграми, що пояснюють дію такого типу скидання.

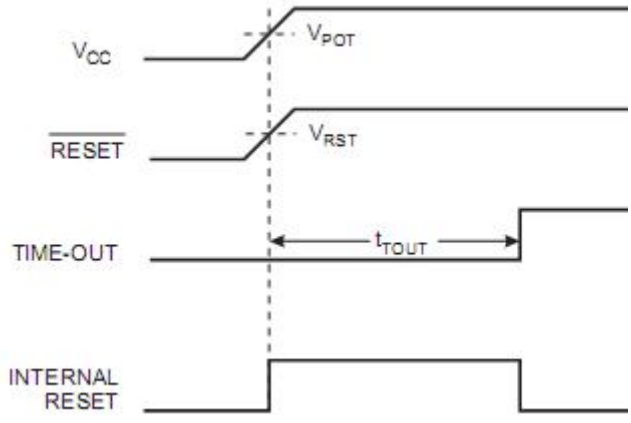


Рис. 11

При подачі напруги живлення проводиться аналіз величини напруги. У разі досягнення цією напругою рівня U_{POT} вмикається внутрішній

генератор затримки, який формує тривалість імпульсу скидання (t_{TOUT}). Сигнал внутрішнього скидання формується на інтервалі часу, що відповідає напрузі живлення меншій за U_{POT}, та додатковому інтервалі, який формує генератор затримки. Тривалість інтервалу затримки можна регулювати за рахунок зміни фюзесів програмування SUT0 – SUT1 (табл. 10). Таке налаштування необхідне для узгодження динамічних властивостей джерела живлення мікроконтролера та системі скидання.

Зовнішнє скидання

На рис. 12 зображено часові діаграми, що пояснюють дію зовнішнього типу скидання.

Зовнішнє скидання викликається внаслідок подачі на вивід *RST* мікроконтролера сигналу низького рівня. Після досягнення позитивним фронтом сигналу скидання рівня U_{RST} вмикається внутрішній генератор затримки, який формує додаткову тривалість імпульсу скидання (t_{TOUT}).

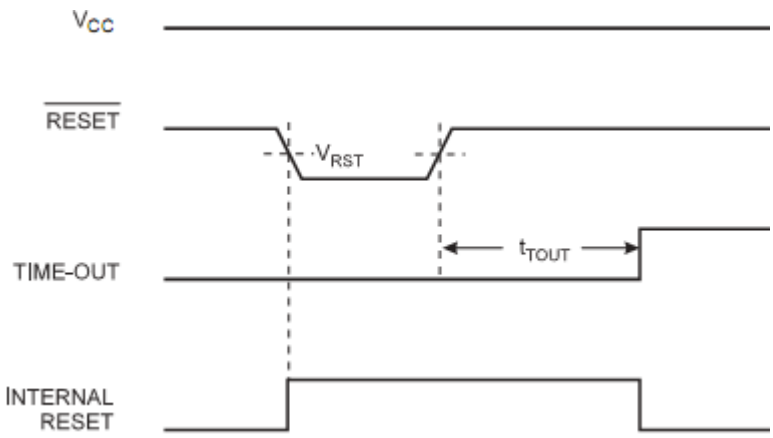


Рис 12

Схема формування сигнала зовнішнього скидання приведена на рис. 13. Зазначимо, що хоча в МК **AVR** є своя внутрішня схема скидання, а сигнал **RESET** підтянутий резистором в 100кОм к **Vcc**, але внаслідок чутливості такої схеми до завад.. рекомендується **RST** подтягнути до лінії живлення резистором в 10к.

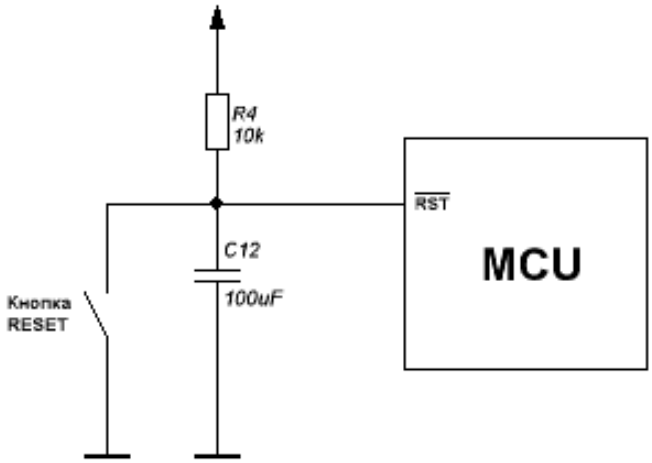


Рис. 13

Скидання під дією супервізора напруги джерела живлення

Супервізор напруги використовується для того, щоб уникнути некоректної роботи мікроконтролера у разі провалів напруги живлення. У мікроконтролера використовується вбудований супервізор напруги (BOD-Brown-Out-Detection) який при програмуванні фюзесу BODEN (рис. 7) може вмикатися. За рахунок відповідного програмування фюзесу BODLEVEL можлива зміна рівня напруги спрацювання супервізора. Для не програмованого фюзесу (BODLEVEL=1) напруга спрацювання складає 2,7В. У разі його програмування (BODLEVEL=0) напруга спрацювання складає 4,2В. На рис 14 зображено часові діаграми, що пояснюють дію супервізора напруги.

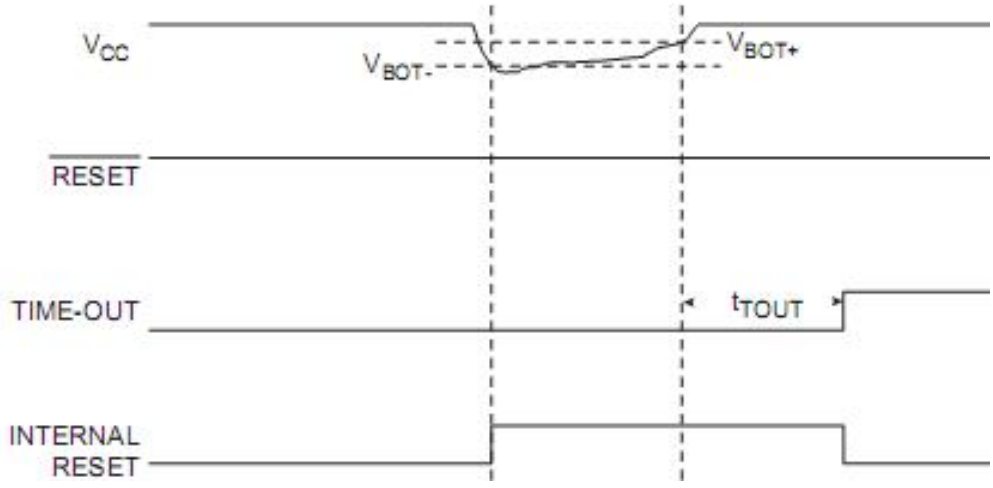


Рис. 14

У разі зменшення напруги живлення до рівня U_{BOT-} починає формуватися імпульс внутрішнього скидання. Якщо напруга живлення підвищується до рівня U_{BOT+} включається внутрішній генератор затримки, який формує додаткову тривалість імпульсу скидання (t_{TOUT}).

Такий варіант скидання мікроконтролера застосовується у разі використання внутрішньої енергонезалежної EEPROM - пам'яті даних. Він забезпечує захист від спотворення даних.

Скидання під дією сторожового таймера

Сторожовий таймер WDT є неодмінним атрибутом усіх сучасних мікроконтролерів. Він використовується для захисту від збоїв програми. При включенні сторожового таймеру через запрограмовані інтервали часу формуються імпульси скидання мікроконтролера. Формування таких імпульсів уникають за рахунок програмного скидання лічильника інтервалу часу. Сторожовий таймер має свій власний *RC*-генератор, що працює навіть під час знаходження мікроконтролера в режимі Power Down. Типове значення частоти дорівнює 1 МГц, при напрузі живлення Vcc = 5.0 В, і 350 кГц, при Vcc = 3.0 В.

Структурну схему сторожового таймера зображено на рис. 15, а на рис. 16 зображено часові діаграми, що пояснюють дію цього типу скидання.

У разі спрацювання системи скидання сторожового таймера формується короткий імпульс запуску ($WDT_{TIME-OUT}$) внутрішнього генератора затримки та починає формуватися внутрішній імпульс скидання. Цей імпульс скидання завершується в кінці інтервалу додаткової тривалості (t_{TOUT}). Тривалість інтервалу повторних спрацювань сторожового таймера встановлюється за допомогою регістра керування WDTCR (рис. 17) згідно з даними табл. 12.

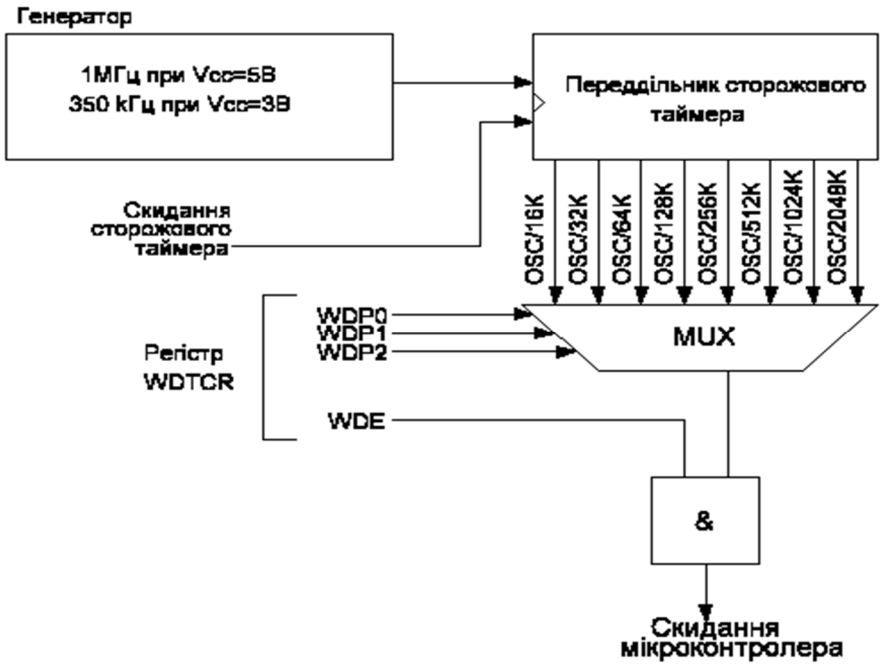


Рис. 2. Структурна схема сторожового таймера

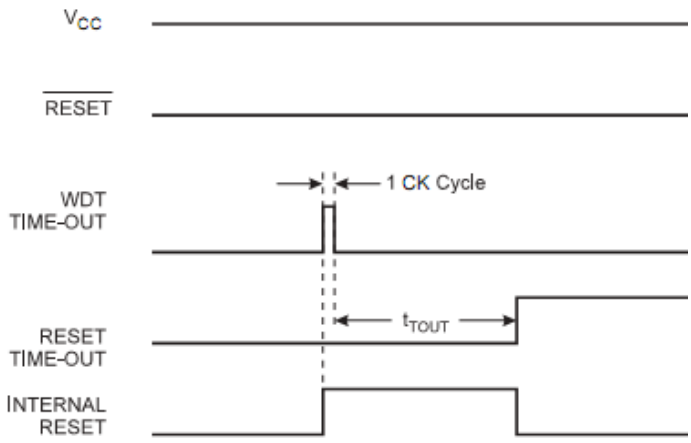


Рис.2.16

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис 17

Для вмикання/вимикання сторожового таймера використовують два розряди регістра WDTCR – WDE І WDTOE. Якщо розряд WDE встановлений в «1», сторожовий таймер ввімкнено, якщо скинуто в «0» -

вимкнений. Безпосередньо перед ввімкненням таймера рекомендується також виконувати його скидання командою WDR. Вимкення сторожового таймера (скидання розряду WDE) можна здійснити тільки при встановленому розряді WDTOE.

Причому через 4 машинні цикли після установки в «1» цей розяд апаратно скидається, завдяки чому фактично зникає можливість випадкового вимкнення сторожового таймера.

Для виключення сторожового таймера рекомендується наступна послідовність дій:

- однією командою записати логічну «1» в розряди WDE I WDTOE;
- протягом наступних чотирьох машинних циклів записати «0» в розряд WDE.

Період затримки спрацювання сторожового таймера задається з допомогою розрядів WDP2…..WDP0 регістру WDTCR згідно з табл. 12.

Щоб уникнути випадкового скидання мікроконтролера при зміні періоду сторожового таймера, необхідно перед записом розрядів WDP2:WDP0 або заборонити роботу сторожового таймера або скинути його.

Таблиця 12

WDP2	**WDP1**	**WDP0**	**Кількість періодів генератора WDT**	**Час спрацьовування WDT при Vcc=3.0V**	**Час спрацьовування WDT при Vcc=5.0V**
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

У системі команд асемблеру AVR мікроконтролерів використовується команда «wdr» для скидання лічильника сторожового таймера в початковий стан. В інтегрованому середовищі *Image Craft ICC v*7, для цієї мети використовується макрос _WDR().

Нижче, у прикладі 1 приведено функцію вимикання сторожового таймера.

***Приклад* 1**

```
      //---------- Функція вимикання сторожового таймера
void WDT_OFF(void)
{
      _WDR();  // скидання лічильника сторожового таймера
WDTCR |= (1<<WDTOE) | (1<<WDE);
      WDTCR = 0x00;
      }
//---------------------------
```

ОРГАНІЗАЦІЯ ПАМ`ЯТІ AVR МІКРОКОНТРОЛЕРІВ

1. Організація пам`яті програм та даних

Пам’ять мікроконтролерів AVR організована за Гарвардською архітектурою, в якій розділені адресні простори пам’яті програм та пам’яті даних, а також і шини доступу до них. Загальна карта розподілу пам’яті мікроконтролера відповідає програмній моделі, яку зображено на рис. 4.

Пам’ять програм

Пам’ять програм (FLASH) призначена для зберігання команд, що керують роботою мікроконтролера. В пам’яті програм зберігаються також різні константи, що не змінюються під час роботи програми.

Усі AVR мікроконтролери мають вбудовану флеш-пам'ять програм, що може бути завантажена як за допомогою звичайного паралельного програматора, так і за допомогою SPI-інтерфейсу, безпосередньо на цільовій платі. Число циклів перезапису залежить від типу мікроконтролера та знаходиться у діапазоні від 1000 до 25000. Час зберігання інформації (Retention Time) також залежить від типу мікроконтролера і для сучасних мікросхем доходить до 25 років.

Пам’ять програм виконана на базі пам’яті флеш-типу з електричним стиранням інформації. Оскільки довжина команди складає 16 біт (деяких – 32 біта), пам’ять програм має 16-розрядну організацію 8К*16 біт. Для адресації пам’яті програм використовується лічильник команд (PC – Program Counter). Його розрядність відповідає об’єму пам’яті програм, для мікроконтролера Atmega16 складає 13 біт. Лічильник команд вказує на адресу двобайтових слів. Відповідно мітки у програмах, які написані мовою асемблеру, також розраховані на таке пакування слів команд. Записи констант в таблиці мають байтове пакування. Тому при завантаженні констант, що розміщуються в таблицях, необхідно подвоювати адресу мітки, щоб переходити від адреси двобайтових слів до

фізичної адреси байта. При написанні програм мовою Сі ця особливість враховується автоматично при компіляції.

На рис. 1 зображено діаграму розподілу пам’яті програм.

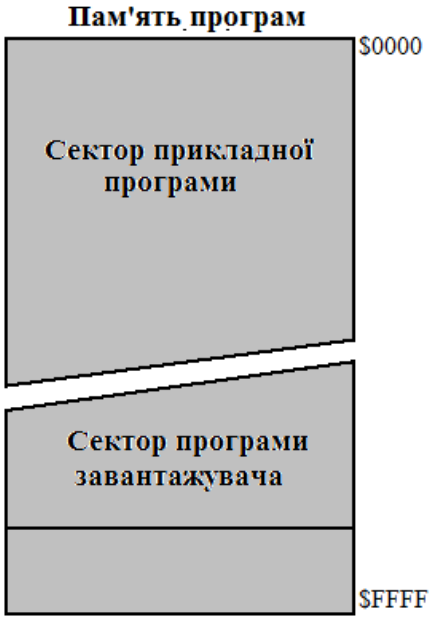


Рис 1

Пам'ять програм розділена на дві частини – сектор програми завантаження (Boot), та сектор прикладної програми. У Boot – секторі може розташовуватися програма, що дозволяє проводити самопрограмування, як області прикладної програми, так і частин програми завантаження мікроконтролера. Перерозподіл розмірів секторів відбувається під час програмування, за рахунок встановлення фюзесів BOOTSZ0, BOOTSZ1.

За адресою $000 пам’яті програм знаходиться вектор скидання. Після ініціалізації (скидання) мікроконтролера виконання програми починається з цієї адреси. Існує можливість переносу адреси вектора скидання на початок Boot-сектора (встановлюється фюзес програмування BOOTRST). Для мікроконтролерів з об’ємом пам’яті програм що перевищує 8 кБайт, за цією адресою розміщують команду переходу rjmp до частини ініціалізації

програми). Починаючи з адреси $002 у мікроконтролера ATmega16 розташовується таблиця векторів переривань.

Використовують наступні способи ініціалізації констант у пам’яті програм.

Якщо адреса розміщення неважлива, то застосовують варіанти, що наведені у прикладі 1.

***Приклад* 1**

```
const int  Con1  = 1;
const char Con_Mas1[] = {1,2,3,4,5,6,7,8,9,10};
```

Якщо необхідно жорстко визначити адресу, то використовують варіант опису, який приведено у прикладі 3.2. Недоліком такого варіанту є можливість перетину фрагментів основної програми з кодами команд та описом констант. У випадку такої події у вікні повідомлень деяких Сі-компіляторів може бути відсутня інформація про помилку. У пам’яті програм розміщуються ті коди, які визначені останніми.

***Приклад* 2**

```
#pragma abs_address:0x1B0
const char Titul[] = "Designed by _______09.2016";
#pragma abs_address:0x1E0
const char Name[] = "Serial number";
#pragma abs_address:0x1F0
const unsigned int Serial_number =0x0000;
#pragma end_abs_address
```

2. Пам’ять даних AVR мікроконтролерів

Пам’ять даних складається з двох областей:

- статичний запам’ятовуючий пристрій (SRAM);
- енергонезалежна пам’ять даних на основі EEPROM.

Кожна з цих областей розміщена в своєму адресному просторі, мають власні вказівники та шини адреси.

На рис. 2 зображено структуру адресного простору SRAM – області. У складі області виділяють наступні сегменти:

- регістровий файл (R0 – R31);

- регістри вводу/виводу (РВВ) 64 - 128, у залежності від моделі мікроконтролера;
- регістри оперативного запам’ятовуючого пристрою (ОЗП).

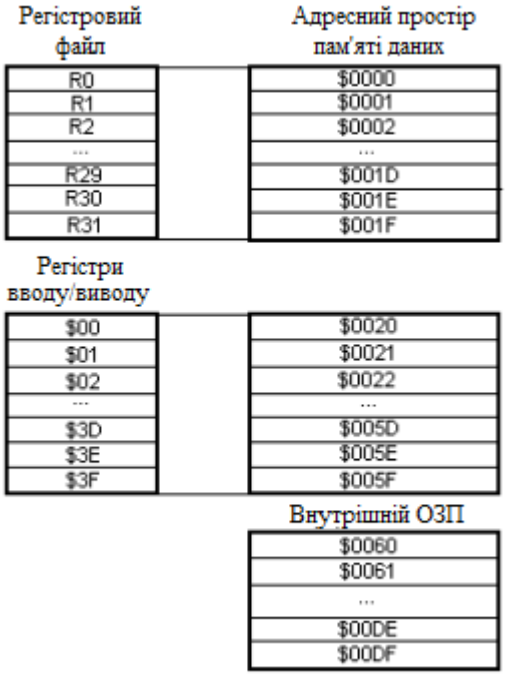


Рис. 2.

Статичний запам’ятовуючий пристрій (*SRAM*)

Регістровий файл включає 32 регістра загального призначення, що мають символічні імена R0 – R31.

В області регістрів вводу/виводу розташовані службові регістри, а також регістри керування пристроями мікроконтролера. Розміри області РВВ залежать від типу мікроконтролера та складають від 64 до 128 байт. Для мікроконтролера Atmega16 РВВ має об’єм у 64 байти.

Для зберігання змінних разом з регістрами файлового регістру також можуть використовуватися регістри ОЗП. Цей сегмент пам’яті використовується також для створення стекової області.

Для зберігання даних, які можуть змінюватися в процесі налагодження та функціонування готової системи, може використовуватися енергонезалежна пам’ять даних - EEPROM. Ця пам’ять

розташована в окремому адресному просторі. Доступ до неї здійснюється за допомогою спеціальних РВВ.

Регістри загального призначення (регістровий файл) та регістри ОЗП SRAM області можуть використовуватися для зберігання змінних.

Крім того регістри SRAM області використовуються для створення стеку до якого записуються адреси точок повернення у разі використання функцій та підпрограм. До стеку можуть також записуватися значення змінних.

Шість із 32-х регістрів файлу можуть використовуватися як три 16-розрядних покажчики адреси при непрямому адресуванні даних (рис. 3). Один із цих покажчиків (*Z Pointer*) застосовується також для доступу до даних, записаних у пам'яті програм мікроконтролера. Використання трьох 16-бітних покажчиків (*X, Y* і *Z Pointers*) істотно підвищує швидкість пересилання даних при роботі прикладної програми.

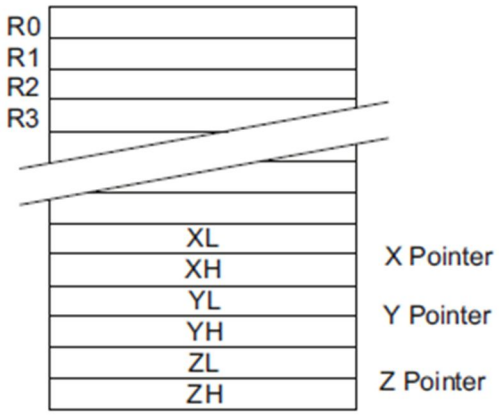


Рис. 3. Регістровий файл

У мікроконтролерах AVR усі 32 РЗП безпосередньо доступні АЛП на відміну від мікроконтролерів інших фірм, в яких є лише один такий регістр – акумулятор. Завдяки цьому будь-який РЗП може використовуватись в усіх командах і як операнд-джерело, і як операнд-передавач. Виняток складають арифметичні та логічні команд, що

виконують дії між константою та регістром (SBCI, SUBI, CPI, ANDI, ORI), команда завантаження константи у регістр (LDI) та команди операцій з бітами (SBR, CBR). Ці команди можуть звертатися лише до другої половини файлового регістру (*R*16…*R*32). Команди 16-розрядного додавання з константою ADIW і віднімання константи SBIW у якості першого операнду використовують тільки регістри *R*24, *R*26, *R*28, *R*30.

Для резервування місця для змінної необхідно вказати її тип та ім'я.

Ініціалізація дистрибутивного значення змінної відбувається після присвоєння їй певного значення. Якщо операція ініціалізації у програмі відсутня, то змінним автоматично присвоюється нулеві значення (компілятор ImageCraft ICC).

***Приклад* 3**

```
unsigned char  var1;     //резервування місця для змінної
unsigned int   var2=9;   //ініціалізація змінної
```

Місце розташування змінної у файловому регістрі чи SRAM-сегменті залежить від її типу модифікаторів доступу до змінної (register, volatile).

Якщо при визначенні змінної тип не вказано, або використано модифікатор register, то резервується місце у сегменті файлового регістру.

У разі застосування модифікатору volatile використовуються файлові регістри R0/R1/R2/R3/R4/R5/R6/R7/R8/R9/R24/R25/R26/R27/R30/R31. Зміст такої змінної може змінюватися у функції, що викликається. Така змінна не запам`ятовується та не поновлюється. Розташована змінна у SRAM-сегменті.

У мікроконтролері Atmega16 регістри вводу/виводу розташовуються у просторі розміром 64 байта.

Для обміну інформацією з регістрами вводу виводу використовують способи, що наведені у прикладі 4.

***Приклад* .4**

```
unsigned char TMP;  //визначення змінної TMP
```

```
PORTA=4;              //запис у регістр PORTA числа «4»
TMP=SREG;             //читання регістру вводу/виводу SREG
```

Енергонезалежна пам’ять даних

Пам’ять EEPROM розташована в своєму адресному просторі та має лінійну організацію. Спеціальних команд звертання до EEPROM немає.

Ініціалізація області EEPROM відбувається з використанням pragma-визначень (компілятор ImageCraft ICC). Для ініціалізації EEPROM визначають символічні імена відповідних змінних та їх вміст. У прикладі 5 проілюстрована ініціалізація EEPROM.

Приклад **5**

```
#pragma data:eeprom
unsigned char ee_var1 = 10;
unsigned int  ee_var2 = 0x100;
#pragma data:data
```

Після компіляції проекту у цьому випадку генерується вихідний файл <output file>.eep.

Читання й запис комірок EEPROM виконується за допомогою регістрів введення/виведення - EEAR (регістр адреси), EEDR (регістр даних) і EECR (регістр керування).

До регістру адреси завантажується адреса комірки, до якої будуть звертатися. У залежності від ємності цього сегменту пам’яті регістр адреси має різну кількість байт. Так для мікроконтролера ATmega16 регістр адреси складається з двох байт – EEARH та EEARL (рис. 4.)

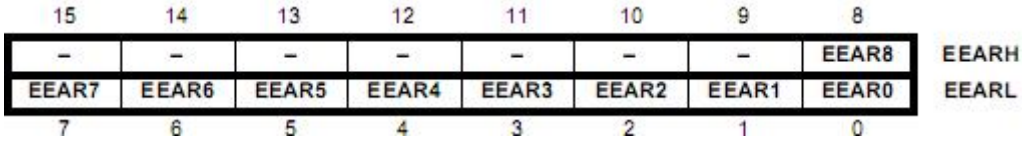


Рис. 4.

Регістр даних EEPROM - пам’яті, називається EEDR. При запису в цей регістр завантажуються дані, які повинні розміщатися в EEPROM за адресою, що знаходиться в регістрах EEARH:EEARL. При зчитуванні в

цей регістр поступають дані записані в EEPROM за адресою, що знаходиться в регістрах EEARH:EEARL (рис. 5).

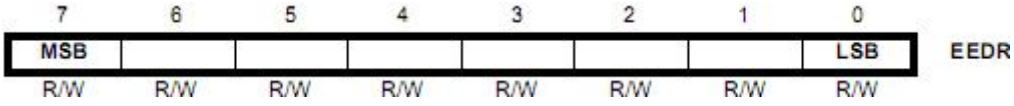


Рис. 5.

Регістр керування використовується для управління доступом до EEPROM - пам’яті. Цей регістр називається EECR (рис. 3.6).

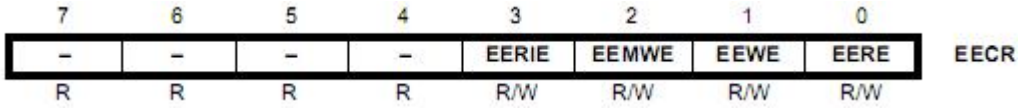


Рис. 6.

Окремі біти цього регістру мають наступне значення:

- EERE - дозвіл читання з EEPROM;
- EEWE - дозвіл запису до EEPROM;
- EEMWE – керування дозволом запису до EEPROM;
- EERIE - дозвіл переривання від EEPROM.

Для звертання до змінних, що розташовані у EEPROM області, можна використовувати бібліотечні функції та макроси, або необхідно розробити власні функції.

Якщо використовуються бібліотечні функції, то для їх виклику необхідно підключити до проекту вкладений файл “eeprom.h” (компілятор ImageCraft ICC). Для запису/читання змінних int-типу використовують функції:

- EEPROM_WRITE(int location, object) – запис двох байт.

Наприклад,

```
unsigned int i;
EEPROM_WRITE(0x1, i);    // запис  змінної “i”за адресою 0x1
```

- EEPROM_READ(int location, object) – читання двох байт.

Наприклад,

```
unsigned int i;
EEPROM_READ(0x1, i);     // читання 2 байт у змінну "i"
```

Для запису/читання змінних char-типу використовують внутрішні функції:

- unsigned char EEPROMread(int location) – читання байту з EEPROM за адресою location;
- int EEPROMwrite(int location, unsigned char byte) – запис байту за адресою location та повернення «0» у разі успіху.

У деяких випадках можлива розробка власних функцій читання/запису для роботи з EEPROM пам'яттю. Наприклад, бібліотечні функції не підтримують роботу з змінними, формат яких складає 4 байти. У прикладах 6, 7 приведені функцій запису/читання байту, які описані в технічному описі мікроконтролера ATmega8.

***Приклад* 6**

```
;-----------------------------------------------------
;Функція запису у EEPROM
;-----------------------------------------------------
void EEPROM_write (unsigned int uiAddress, unsigned char
ucData)
{
while (EECR & (1<<EEWE));     //очікування закінчення
//попереднього звертання
     EEAR = uiAddress;        //встановлення адреси EEPROM
     EEDR = ucData;           //встановлення даних EEPROM
     EECR |= (1<<EEMWE);      //запуск майстер - стробу
     EECR |= (1<<EEWE));      //запуск запису
}
```

***Приклад* 7**

```
;-----------------------------------------------------
;Функція читання з EEPROM
;-----------------------------------------------------
unsigned char EEPROM_read (unsigned int uiAddress)
{
while (EECR & (1<<EEWE));     //очікування закінчення
//попереднього звертання
EEAR = uiAddress;        //встановлення адреси EEPROM
EECR |= (1<<EERE);       //дозвіл читання
```

```
return EEDR;                //повернення даних
}
```

У прикладі 8 приведено програму в якій проводиться ініціалізація константи у FLASH - сегменті та змінних у SRAM-області і EEPROM-сегменті. Для розробки програми використовувався засіб Application Builder інтегрованого середовища ImageCraft ICC.

***Приклад* 8**

У наведеному нижче прикладі проводиться така ініціалізація змінних:

- ee_var1 та ee_var2 визначаються у EEPROM-сегменті;
- c_var1 визначається у FLASH (ROM) - сегменті;
- tmp_SRAM визначається у SRAM-області.

З використанням бібліотечних функцій видобуваються значення змінних, що розташовані у EEPROM-сегменті .

Проводиться математична обробка змінних у відповідності з наступним виразом

```
result = c_var1*(tmp_ee1+tmp_ee2)-tmp_SRAM.
```

```
//-------------------------------------------------------------
--
//ICC-AVR application builder: 11.02.2016 11:45:40
// Target: M16
// Crystal: 10.000Mhz
//-------------------------------------------------------------
--
#include <iom16v.h>
#include <macros.h>
#include <eeprom.h>
//-------------------------------------------------------------
--//ініціалізація змінних у EEPROM
#pragma data:eeprom
unsigned char ee_var1 = 100;
unsigned int  ee_var2 = 0x100;

#pragma data:data
//-------------------------------------------------------------
//ініціалізація константи  у FLASH (ROM)
const char c_var1 = 2;
//-------------------------------------------------------------
void port_init(void)
{
PORTA = 0xC0;               //ініціалізація портів
DDRA  = 0xF0;
PORTB = 0x00;
DDRB  = 0x00;
```

```
PORTC = 0x00;
DDRC  = 0x00;
PORTD = 0x00;
DDRD  = 0x00;
}
//-----------------------------------------------------------
--
Void init_devices(void)
{
CLI();                      //disableallinterrupts
port_init();                //ініціалізація портів
MCUCR = 0x00;
GICR  = 0x00;
TIMSK = 0x00;               //timerinterruptsources
SEI();                      //re-enableinterrupts
}
//-----------------------------------------------------------
--
//Головна функція
//------------------
void main(void)
{
//--------------------------
init_devices();          //Ініціалізація мікроконтролера

//--------------------------
while(1)             //Головний цикл
 {
unsigned int result,tmp_ee1,tmp_ee2;
unsigned int tmp_SRAM = 10;
tmp_ee1 = EEPROMread(ee_var1);  //читання char-змінної ee_var1
EEPROM_READ((int)&ee_var2,tmp_ee2);//читання        int-змінної
ee_var2
result=c_var1*(tmp_ee1+tmp_ee2)-tmp_SRAM;         //математична
обробка
};
//--------------------------
}

//-----------------------------------------------------------
```