

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни " Комп'ютерна схемотехніка " для
студентів фаху 8.091501 «Комп'ютерні системи та мережі» та
7.091503 „Спеціалізовані комп'ютерні системи” усіх форм навчання

Частина I Мікроконтролери AVR

2016

Методичні вказівки до лабораторних робіт з дисципліни "Комп'ютерна схемотехніка " для студентів фаху 8.091501 «Комп'ютерні системи та мережі» та 7.091503 „Спеціалізовані комп'ютерні системи” усіх форм навчання. Частина I Мікроконтролери AVR /Укл.: С.О.Сгадов–Запоріжжя: ЗНТУ, 2016. – 58 с.

Укладачі: С.О. Сгадов , ст. викладач,

Рецензент: Р.К. Кудерметов, доцент, к.т.н.

Відповідальний за випуск: С.О. Сгадов, ст. викладач

Затверджено
на засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол №3 від 18.10.2016

Рекомендовано до видання НМК
ФКНТ
Протокол № 2
від “20” 10 2016.

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ	5
ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ..	5
1 ЛАБОРАТОРНА РОБОТА №1 ВИВЧЕННЯ СТЕНДА ТА КОМАНД ОДНОКРИСТАЛЬНОЇ ЕОМ AT90USB162	6
1.1 Порядок виконання лабораторної роботи	6
1.2 Теоретичні відомості.....	6
1.3 Контрольні питання.....	10
2 ЛАБОРАТОРНА РОБОТА №2 ВИКОРИСТАННЯ ДИНАМІЧНОГО ІНДИКАТОРА	12
2.1 Порядок виконання лабораторної роботи.....	12
2.2 Короткі теоретичні відомості	12
2.3 Приклад підпрограми декодування	14
2.3 Варіанти індивідуальних завдань	14
2.4 Контрольні питання.....	15
3 ЛАБОРАТОРНА РОБОТА №3 СИСТЕМА ПЕРЕРИВАНЬ. ОПИТУВАННЯ ДИСКРЕТНИХ ДАТЧИКІВ	17
3.1 Порядок виконання лабораторної роботи.....	17
3.2 Короткі теоретичні відомості	17
3.3 Приклад налаштування INT1 та INT5 на падаючий фронт	26
3.4 Варіанти індивідуальних завдань	26
3.5 Контрольні питання.....	27
4 Лабораторная работа №4 ИЗУЧЕНИЕ ТАЙМЕРОВ МИКРОКОНТРОЛЛЕРА AVR.....	28
5 ЛАБОРАТОРНА РОБОТА №5 ГЕНЕРАЦІЯ ШІМ.....	42
5.1 Порядок виконання лабораторної роботи.....	42
5.2 Короткі теоретичні відомості	43
5.2.1 Режими ШІМ.....	43
6 ЛАБОРАТОРНА РОБОТА №6 СТВОРЕННЯ БІБЛІОТЕКИ КЕРУВАННЯ ДИНАМІЧНОЮ ІНДИКАЦІЄЮ	48
6.1 Порядок виконання лабораторної роботи.....	48
6.2 Короткі теоретичні відомості	48
7 ЛАБОРАТОРНА РОБОТА №7 ГЕНЕРАЦІЯ ЗВУКА.....	51
7.1 Порядок виконання лабораторної роботи.....	51
7.2 Короткі теоретичні відомості	51

8 ЛАБОРАТОРНА РОБОТА №8 КЕРУВАННЯ СИСТЕМНОЮ ЧАСТОТОЮ ТА ВИЗНАЧЕННЯ ПРОДУКТИВНОСТІ МІКРОКОНТРОЛЛЕРУ	53
8.1 Порядок виконання лабораторної роботи.....	53
8.2 Короткі теоретичні відомості	53
ЛІТЕРАТУРА.....	53
Додаток А.....	57
Додаток Б.....	58

ЗАГАЛЬНІ ПОЛОЖЕННЯ

При виконанні лабораторного практикуму студент зобов'язаний:

- ознайомитися зі змістом майбутньої лабораторної роботи;
- повторити матеріал того розділу курсу “Комп'ютерна схемотехніка”, якому відповідає майбутня робота, і підготувати бланк звіту (заповнити титульний лист, написати алгоритм програми і підготувати відповіді на контрольні запитання);
- виконати роботу і відповісти викладачу на питання курсу “Комп'ютерна схемотехніка” даного розділу;
- у процесі самопідготовки оформити бланк звіту, якщо необхідно, прикласти листінг програми і зробити висновки про результати роботи;
- здати викладачу цілком підготовлений і ретельно оформлений бланк звіту.

При проведенні лабораторних робіт курсу “Комп'ютерна схемотехніка” усі бригади (по 1-5 чоловік) однієї групи виконують ту саму лабораторну роботу. Кожен студент повинен брати участь у виконанні роботи, оформити і здати викладачу звіт.

Працюючи в лабораторії, необхідно пам'ятати, що ви повинні виконати правила встановлені на кафедрі. У випадку несправностей обчислювальної техніки негайно відключити живлення і довести до відома викладача чи лаборанта.

ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

Звіт являє собою акуратно оформлені результати виконання завдання, представлені на встановленому бланку.

Звіт по роботі з підписом студента повинний бути зданий наприкінці лабораторного заняття. Він одночасно є допуском до виконання наступної роботи. На титульному листі звіту слід вказати номер і назву роботи, прізвище і групу студента.

Звіт повинний містити:

- алгоритм програми;
- короткий опис роботи алгоритму;
- програму реалізації (мова Асемблер);

висновки з лабораторної роботи з роздрукованими результатами.

Виконаний звіт пред'являється наприкінці заняття для затвердження викладачу.

1 ЛАБОРАТОРНА РОБОТА №1 ВИВЧЕННЯ СТЕНДА ТА КОМАНД ОДНОКРИСТАЛЬНОЇ ЕОМ AT90USB162

Мета роботи: Вивчення функціональних можливостей учбово-відладочного стенда, внутрішньої структури і системи команд ЕОМ AtMega. Вивчення команд пересилань, арифметичних, логічних команд, команд переходів, взаємодія внутрішніх вузлів ЕОМ.

1.1 Порядок виконання лабораторної роботи

1.1.1 Вивчити структурну схему стенда, розподіл пам'яті, призначення вузлів.

1.1.2 Вивчити структуру AT90USB162.

1.1.3 Вивчити електричну схему стенду.

1.1.4 Вивчити синтаксис команд пересилання, арифметичних команд, команд переходів.

1.1.5 Скласти програму на асемблері у середовищі Atmel Studio

1.1.5 Відповісти на контрольні питання викладача.

1.2 Теоретичні відомості

Структура типової програми

```
;*****
; название программы,
; краткое описание, необходимые пояснения
;*****
.device      AT90USB162
.include "usb162def.inc"
.include «имя_файла1.расширение» ; включение дополнительных
.include «имя_файла2.расширение» ; файлов
;*****глобальные константы
.equ         имя1 = xxxx ;
.equ         имя2 = nnnn
;*****глобальные регистровые переменные
.def         имя1= регистр
.def         имя2= регистр
;*****сегмент данных
.dseg
.org xxxx           ; адрес первого зарезервированного байта
label1: .BYTE 1      ; резервировать 1 байт под переменную
label1
label2: .BYTE m       ; резервировать m байт под переменную
label2
;***** сегмент EEPROM (ЭСППЗУ)
.eseg
.org          xxxx           ; адрес первого
```

```

зарезервированного байта
.db выражение1,выражение2,... ; записать список байтов в EEPROM.
.dw выражение1,выражение2,... ; записать список слов в EEPROM.
;*****сегмент кодов
.cseg
.org $0000 ; адрес начала программы в программной
памяти
;***** вектора прерываний (если они используются)
rjmp main ;прерывание по сбросу
.org $0002
rjmp INT0 ;обработчик прерывания IRQ0
.org $0004
rjmp INT1 ;обработчик прерывания IRQ1
.org adrINTx ;адрес следующего обработчика прерываний
rjmp INTx ;обработчик прерывания x
..... ;далее по порядку располагать
обработчики остальных ;прерываний

org 0x40 ;*****начало основной программы
;зависит от размера таблицы векторов прерываний
main: <команда> xxxx
...

;***** подпрограммы
;*****подпрограмма 1
subr1: <команда> xxxx
.....
ret
;*****подпрограмма 2
subr2: <команда> xxxx
.....
ret
.....
;***** программы обработчиков прерываний
INT0: <команда> xxxx
.....
reti
INT1: <команда> xxxx
.....
reti
INTx: <команда> xxxx
.....
reti
.....
; конец программы никак не обозначается.

```

Приклад програми, що вмикає усі світлодіоди

```
.device      AT90USB162
.include "usb162def.inc"
.def        tmp= r16
.equ        RedDiod=5
.equ YellowDiod=6
.equ        GreenDiod=0;PD
.equ        BlueDiod=2
.cseg
.org $0000                                ; адрес начала
програми в програмній пам'яті
;***** вектора прерываний (если они используются)
    rjmp reset                            ; прерывание по
сбросу
.org $0040
;*****начало основной программы
reset:
ldi tmp,High(RAMEnd) ;определение стека
    out      sph,tmp      ; с вершиной по адресу
RAMEnd
    ldi      tmp,Low(RAMEnd)
    out      spl,tmp
; конфігурація портів
ldi      tmp,$00      ; чистим порты В И С
    out      PortB,tmp
    out      PortC,tmp
    ldi      tmp,0b00010010
    out      PortD,tmp
    ldi      tmp,0b11111111      ; грузим признак
выхода
    out      DDRB,tmp            ; в регистр
управления
                                ; портом "В"
    ldi      tmp,0b11111100      ;
    out      DDRC,tmp           ; в регистр упр. портом
"С"
    ldi      tmp,0b01101001      ;
```

```

out    DDRD,tmp ; грузим в регистр упр. портом "D"

main:
ldi tmp, (1<<RedDiod)|(1<<YellowDiod)|(1<<BlueDiod)
    out    PortC,tmp
ldi tmp, (1<< GreenDiod)
    out    PortD,tmp
    end: jmp end
; конец программы

```

Формування затримки з допомогою циклів

```

delay:
    ldi r17,0xff; количество повторів
    ldi r18,0x7 ;   у r17-r20
    ldi r19,0
    ldi r20,0
_delay: subi r17,0x1
    sbci r18,0
    sbci r19,0
    sbci r20,0
    brcc _delay
    ret

```

Є доцільним зробити зовнішній файл з макросами для встановлювання то зброду бітів

```

.macro setpin
    in tmp,@0
    ori tmp,(1<<@1)
    out    @0,tmp
.endmacro

.macro Resetpin
in tmp,@0
andi tmp, ~(1<<@1)
out    @0,tmp
.endmacro

```

Ще корисними макросами є загрузка молодших регістрів константами (бо це неможливо за прямої спосіб)

```
.macro load
    ld tmp,@1
    mov @0,tmp
.endmacro
```

Приклад використання цих макросів

```
setpin PORTC,$64
resetpin PORTC,$64
load r1,$25
```

При цьому Ви зобов'язані определити tmp як регістр:

```
.def      tmp= r16
```

Щоб після компіляції автоматично запуслався загрузчик треба додати в опціях проекту стрічку в пункт "Post-build event command line"

```
"%ProgramFiles%\Atmel\Flip 3.4.7\bin\batchisp.exe"
-device AT90USB162 -hardware usb -operation erase
f memory flash blankcheck loadbuffer
"$ (OutputDirectory)\$ (OutputFileName).hex" program
verify start noreset 0
```

Нагодуюм, что мікроконтролер треба увести в стан завантажування коротким натисканням Reset (червона кнопка) при утриманні Boot(чорна кнопка).

Або можливо завантажити hex утилитой Flip у ручному режимі.

1.3 Контрольні питання

1.3.1 Час виконання команд (поняття такту, машинного циклу).

1.3.2 Типи команд. Формат команд. Команди прямої і непрямой адресації.

1.3.3 Команди зсуву. Арифметичні команди. Логічні команди.

1.3.4 Регістр ознак, команди, що викликають зміну регістру ознак.

1.3.5 Команди роботи зі стеком, послідовність дій (команд) при роботі зі стеком.

1.3.6 Призначення внутрішніх вузлів ОЕОМ.

1.3.7 Призначення і робота з регістрами даних ОЕОМ.

1.3.8 Система переривань ОЕОМ. Призначення портів ОЕОМ.

1.3.9 Фізичні характеристики вихідних сигналів ОЕОМ.

1.3 Варіанти індивідуальних завдань

Таблиця 1.1 - Таблиця завдань до лабораторної роботи №1

№	Текст індивідуального завдання
1	У циклі запалити червоний, синій світлодіоди на 1 секунду з інтервалом в 0.5 секунд. Через секунду погасити.
2	З інтервалом 0.5з послідовно запалювати червоний, жовтий, синій, зелений.
3	Мигати всіма світлодіодами із частотою 2 Гц.
4	У циклі запалити жовтий, потім синій світлодіоди на 1 секунду через секунду погасити.
5	З інтервалом 1з послідовно запалювати жовтий, червоний, синій, зелений
6	Мигати світлодіодами, розташованими по одній і по іншій діагоналі с інтервалом 1 сек .
7	Послідовно запалювати світлодіоди за годинниковою стрілкою с інтервалом 0.5 сек.
8	Послідовно запалювати світлодіоди проти годинникової стрілки с інтервалом 2 сек .

2 ЛАБОРАТОРНА РОБОТА №2 ВИКОРИСТАННЯ ДИНАМІЧНОГО ІНДИКАТОРА

Мета роботи: Навчитися запускати прості програми на стенді.

2.1 Порядок виконання лабораторної роботи

2.1.1 Скласти програму для обчислення даного в індивідуальному завданні цілочисельного логічного виразу з виводом на індикатор результату.

2.1.2 Запустити програму на стенді.

2.2 Короткі теоретичні відомості

2.2.2 Системи відображення інформації

Найпростішими приладами відображення інформації в цифрових пристроях є світлодіоди і цифрові індикатори.

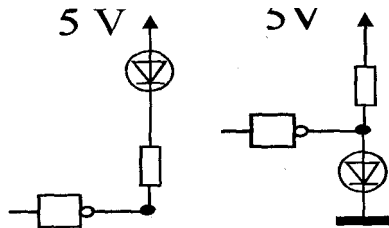


Рисунок 2.1 - Включення одиничних індикаторів

У напівпровідникових світлодіодах використовується властивість р-п переходу випромінювати світло у видимій частині спектра при протіканні через нього прямого струму ($I_p=5-20\text{ма}$, $U_p=2-3\text{В}$). Варіанти включення індикаторів на рис.2.1.

Для відображення цифрової інформації найбільше поширення одержали семисегментні індикатори, у яких зображення цифри складають із семи лінійних світлодіодних сегментів розташованих у виді цифри 8. На основі світлодіодів і семи сегментних індикаторів будуються підсистеми відображення інформації. При побудові

підсистем відображення інформації розрізняють два підходи - динамічна і статична індикація.

Статична індикація полягає в постійному підсвічуванні індикаторів HL1-n від одного джерела інформації рис.2.2.

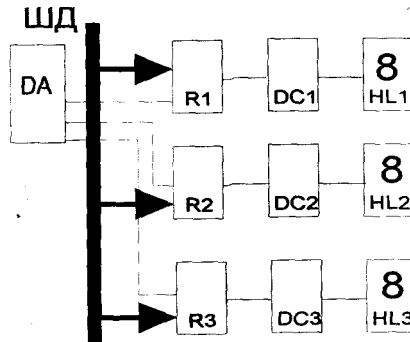


Рисунок 2.2 - Структурна схема статичної індикації

DA - дешифратор адреси, необхідний для вибірки відповідного регістра.

R1-R3- регістри в яких тимчасово зберігається значення коду числа для відображення (відповідний регістр вибирається DA).

DC1-DC3 – семисегментні дешифратори, що перетворюють двійковий код у семисегментний код.

HL1-HL3 - семисегментні індикатори.

ШД - шина даних, по ній здійснюється передача даних на індикацію.

У такій системі кожен індикатор HL1-n підключається через власний дешифратор DC1-n і регістр-клямку RG1-n до шини даних, вибірка регістрів RG1-n проводиться за допомогою селектора адреси СА. Апаратні витрати при такій організації складають n пар (регістр + дешифратор) при n десяткових розрядах індикатора.

У стенді адресація реалізована за допомогою виходів портів контролеру PC4,PC7,PD5,PD6.

Сутність динамічної індикації полягає в почерговому циклічному підключенні кожного індикатора HG1-n до джерела інформації через порт РВ.

При такому включенні значно зменшуються апаратні витрати, але необхідно забезпечити достатній час світіння одного індикатора, для того щоб не зменшувалася яскравість. Також необхідно забезпечити таку частоту перебору індикаторів, щоб не було помітне мерехтіння.

2.3 Приклад підпрограми декодування

Нехай є таблиця

dcode:

```
.db 0b00111111 ;"0"
.db 0b00000110 ;"1"
```

.....

```
.db 0b01110001 ;"f"
```

тоді можемо кодувати розряд у r2 через цю процедуру

decode:

```
lsl r2
ldi zh, high(dcode*2)
ldi zl, low(dcode*2)
ldi tmp, 0
add zl, r2
adc zh, tmp
lpm r0, z
out PORTB, r0; вивод кода индикации в порт
ret
```

2.3 Варіанти індивідуальних завдань

Таблиця 2.1 - Таблиця завдань до лабораторної роботи №2

№	Текст індивідуального завдання
1	Занести в регістр R4 ДД число 0X, у регістр R6 - X0, суму чисел відобразити на інд. HG1, HG0.
2	Занести в регістр R3 ДД число XX відобразити його на інд. HG3, HG0.
3	Занести в регістр B ДД число, з частотою 2 Гц виводити це число на інд. HG1, HG2.
4	Занести в R26 ДД число XX, у регістр R25 – X0, число з R26 відобразити на інд. HG1, HG0, число X з R25 відобразити на інд. HG2.

5	Занести в регістр R22 ДД число 0X, у регістр R15 – X0, суму чисел відобразити на інд. HG1, HG2
6	Занести в комірку з адресою 0Bh внутрішньої пам'яті ОЕОМ ДД число 0X, у регістр R3 – X0, суму чисел відображати на інд. HG2, HG1 з частотою 0,5Гц.
7	Занести в регістр R10 ДД число XX, поперемінно відображати молодшу і старшу тетраду на інд. HG3, HGO з частотою 1 Гц.
8	Занести у регістр R27 ДД число X0, у регістр R28 - XX, число X з R27 відображати на інд. HG0 з частотою 1 Гц, число з R1 відображати на інд. HG3, HG2 з частотою 0,5 Гц.
11	Занести в R2 ДД число X0, у регістр R3 - 0X, суму чисел відобразити на інд. HG3, HG0.
12	Занести в регістр R22 ДД число 0X, у регістр R5 – X0, два розряди суми (десятки і одиниці) по черзі відображати на інд. HG1, HG0.
13	Занести в регістр R21 ДД число 0X, віднімаючи від числа одиницю відображати на HG2 отримане значення до нуля з частотою 1 Гц.
14	Занести в регістр R23 ДД число XX, у регістр R5 XX, поперемінно відображати ці числа на інд. HG3, HG2, HG1, HG0.
15	Занести в регістр R1 ДД число 0X, у регістр R2 – X0, число X з R2 відобразити на інд. HG3, число X з регістру R1 відображати на інд. HG1 з частотою 0.5 Гц

Перелік використаних скорочень:

А – акумулятор;

ДД – двоїчно – десяткове;

X – будь – яка цифра від 1 до 9;

інд. – статичний індикатор.

2.4 Контрольні питання

2.4.1 Що таке динамічна індикація?

2.4.2 Дешифрація двійково-десяткових чисел при виводі на індикатори.

2.4.3 Програмне встановлення, скидання та маскування бітів засобами процесора AVR.

2.4.4 Арифметико-логічні операції процесора AtMega.

2.4.5 Організація часових затримок.

Розрахунок часу регенерації для динамічного методу відображення .

2.4.6 Обґрунтування необхідності застосування різних методів відображення даних.

2.4.7 Схемотехнічні рішення для побудови схем відображення інформації.

2.4.8 Схеми включення одиничних індикаторів.

3 ЛАБОРАТОРНА РОБОТА №3 СИСТЕМА ПЕРЕРИВАНЬ. ОПИТУВАННЯ ДИСКРЕТНИХ ДАТЧИКІВ

Мета роботи: Вивчення режимів роботи системи переривань AVR, програмна обробка дискретних сигналів.

3.1 Порядок виконання лабораторної роботи

3.1.1 Вивчити систему переривань AVR AT90USB162, особливості опитування дискретних датчиків з механічними контактами.

3.1.2 Розробити алгоритм для виконання індивідуального завдання.

3.1.3 Розробити програму для виконання індивідуального завдання до початку лабораторного заняття

3.1.4 Увести програму індивідуального завдання на ПК.

3.1.5 Проаналізувати виконання індивідуальної програми

3.1.6 Завантажити програму в стенд. Переконалися в правильному виконанні індивідуального завдання, при негативному результаті здійснити зміну алгоритму або програми. Повторити завантаження програми в стенд AVR.

3.1.7 Роздрукувати листінг правильно працюючої програми.

3.1.8 Відповісти на контрольні питання викладача.

3.2 Короткі теоретичні відомості

3.2.1 Опитування дискретних сигналів

Для введення інформації широко застосовуються кнопкові перемикачі і контактні клавіатури. Сигнал таких перемикачів формується шляхом замикання (розмикання) електричного ланцюга. Сигнал сформований контактною парою супроводжується деренчанням, тривалість якого складає ~8-12мс (рис.3.1)

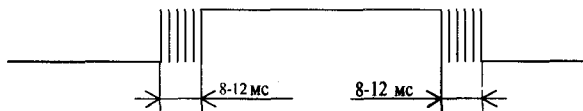


Рисунок 3.1 - Сигнал контактної пари

Для усунення деренчання в одержуваному сигналі на виході контактної пари встановлюють спеціальні формувачі. Приклад такого формувача заснованого на принципі безпосередньої установки RS-тригера приведений на рис. 3.2.

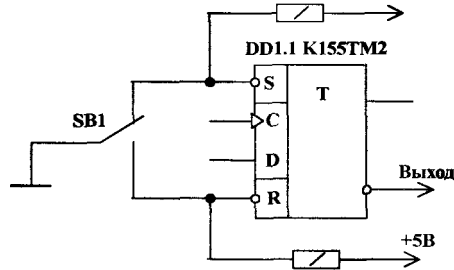


Рисунок 3.2 - Схема усунення деренчання за допомогою RS-тригера

Для зменшення апаратних витрат застосовують програмне усунення деренчання. Воно полягає в повторному опитуванні контактної пари з затримкою в 12 мс. Якщо результати опитування збігаються, то кнопка була натиснута, інакше в результаті першого опитування було зафіксовано деренчання.

3.2.2 Система переривань AVR AtMega

Як працює переривання в AVR мікроконтролерах?

1. Припустимо відбувся “запит на обробку переривання” (IRQ).

До речі : якщо одночасно відбудуться кілька запитів на обробку переривання, те першим буде оброблене переривання з найвищим пріоритетом, усі інші запити будуть оброблені по завершенню високопріоритетного переривання.

2. Перевірка.

Якщо біт активації даного переривання встановлений (Interrupt enable bit), а також I-Біт (біт загального дозволу переривань) регістру стану процесора (SREG) встановлений, то процесор починає підготовку процедури обробки переривання, при цьому біт загального дозволу переривань (I-Біт регістру SREG) скидається, забороняючи в такий спосіб усі інші переривання. Це відбувається для того щоб ніяка інша подія не змогла перервати обробку поточного переривання.

До речі : якщо в процедурі обробки переривання встановити I-Біт у стан лог. одиниці, те будь-яке активоване переривання може у

свою чергу перервати обробку поточного переривання. Такі переривання називаються вкладені (Nested interrupts).

3. Підготовка.

Процесор завершує виконання поточної асемблерної команди, після чого поміщає адресу наступної команди в стек (PC->STACK). Далі процесор перевіряє яке джерело переривання подало “запит на обробку переривання” (IRQ), після чого скориставшись вектором даного джерела (посилання) з таблиці векторів (який залізно закріплений за кожним джерелом переривання), переходить у процедуру обробки переривання (інструкція JMP). На всі, про все процесор витрачає мінімум 4 такту! (залежно від моменту появи запиту й тривалість виконання поточної інструкції). Це дуже гарний час реакції на IRQ, у порівнянні з мікроконтролерами інших виробників.

До речі : якщо IRQ відбудеться, коли мікроконтролер перебуває в сплячому режимі (sleep mode), час реакції на IRQ збільшується ще на чотири такти, плюс час закладене у фьюз бітах SUT1 і SUT0 (Start-Up Time).

4. Виконання тіла ISR.

До речі : більшість Сі-компіляторів, перед початком і завершенням виконання тіла процедури обробки переривання, вставляють додаткові “підпрограми збереження/восстановлення регістрів” загального призначення (РОН) і регістру стану (SREG). Так що програмістові немає необхідності самостійно зберігати/восстановлювати РОН під час переривань. Просунуті Сі-компілятори можуть вирахувати скільки РОН знадобляться даному оброблювачеві переривання й вставити підпрограми збереження/восстановлення тільки потрібної кількості РОН.

5. Повернення керування

Після того як виконання процедури обробки переривання завершено, процесор витягає адресу повернення з переривання, який він зберіг у стеці (STACK->PC), додає до покажчика стека значення 2, що відповідає зменшенню стека на два байти, які раніше займав адресу повернення з переривання. Далі процесор уставляє I-Біт (біт загального дозволу переривань) регістру стану процесора (SREG). На всі, про все процесор витрачає рівно 4 такту. Після цього виконання програми триває рівно з того місця де вона була перервана.

Ради по створенню ISR

1. Намагайтеся якнайшвидше вийти із процедури обробки переривання, від цього залежить чуйність вашого додатка. Тобто якщо ви відносно довго будите перебуває в ISR, а в цей момент будуть з'являтися нові запити на обробку переривання, то почнеться нагромадження, а потім і втрата нових подій. Як ви самі розумієте таку поведінку бажане уникнути.

2. Обчислення в переривання повинні бути як можна простіше, від цього залежить число, що зберігаються /, що відновляють РОН.

Переривання Reset

У переривання Reset є 5 джерел переривання :

- Скидання при включенні живлення (Power-on Reset).
Мікроконтролер перебуває в стані скидання, поки напруга живлення перебуває нижче граничного значення VPOT.
- Зовнішнє скидання (External Reset). Мікроконтролер перебуває в стані скидання, поки на вивід RESET поданий низький рівень.
- Скидання сторожового таймера (Watchdog Reset).
Мікроконтролер скидається через деякий період часу, заданий у сторожовому таймері й коли сторожовий таймер включений.
- Скидання при виході за межу (Brown-Out Reset).
Мікроконтролер скидається, коли напруга живлення VCC опускається нижче граничного значення VBOT і коли Brown-Out детектор включений.
- JTAG скидання. Мікроконтролер перебуває в стані скидання, поки в однібітному регістрі Reset Register перебуває значення лог. одиниця.

3.2.3 Таблиця переривань

. Таблица 3.1-Таблиця адрес переривань AT90USB162

Вектор	Адреса	Источник	Определение
1	\$0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, USB Reset and debugWIRE AVR Reset
2	\$0002	INT0	External Interrupt Request 0
3	\$0004	INT1	External Interrupt Request 1
4	\$0006	INT2	External Interrupt Request 2
5	\$0008	INT3	External Interrupt Request 3
6	\$000A	INT4	External Interrupt Request 4
7	\$000C	INT5	External Interrupt Request 5
8	\$000E	INT6	External Interrupt Request 6
9	\$0010	INT7	External Interrupt Request 7
10	\$0012	PCINT0	Pin Change Interrupt Request 0
11	\$0014	PCINT1	Pin Change Interrupt Request 1
12	\$0016	USB General	USB General Interrupt request
13	\$0018	USB Endpoint	USB Endpoint Interrupt request
14	\$001A	WDT	Watchdog Time-out Interrupt
15	\$001C	TIMER1 CAPT	Timer/CounterM Capture Event
16	\$001E	TIMER1 COMPA	Timer/CounterM Compare Match A
17	\$0020	TIMER1 COMPB	Timer/CounterM Compare Match B
18	\$0022	TIMER1 COMPC	Timer/CounterM Compare Match C
19	\$0024	TIMER1 OVF	Timer/CounterM Overflow
20	\$0026	TIMER0 COMPA	Timer/CounterO Compare Match A
21	\$0028	TIMER0 COMPB	Timer/CounterO Compare match B
22	\$002A	TIMER0 OVF	Timer/CounterO Overflow
23	\$002C	SPI, STC	SPI Serial Transfer Complete
24	\$002E	USART1 RX	USART1 Rx Complete
25	\$0030	USART1 UDRE	USART1 Data Register Empty
26	\$0032	USART1TX	USART1 Tx Complete
27	\$0034	ANALOG COMP	Analog Comparator
28	\$0036	EE READY	EEPROM Ready
29	\$0038	SPM READY	Store Program Memory Ready

3.2.4 Зовнішні переривання.

Зовнішні переривання здійснюються через виводи INT7:0. Зверніть увагу, що після дозволу зовнішні переривання будуть генеруватися, навіть якщо лінії INT7:0 настроєні як виходи. Дана

особливість може використовуватися для програмної генерації переривання. Зовнішні переривання можуть генеруватися по падаючому або наростаючому фронту, а також по низькому лог. рівню. Одна із цих установок задається в регістрах керування зовнішніми перериваннями EICRA (INT3:0) і EICRB (INT7:4). Якщо зовнішнє переривання дозволене й настроєне на спрацьовування при низькому рівні, то переривання буде ініціюватися постійно поки на виводі буде залишатися низький рівень. Зверніть увагу, що для розпізнавання падаючого або наростаючого фронтів на INT7:4 необхідна наявність синхронізації введення-виводу. Переривання по низькому рівню й фронтам на INT3:0 визначаються асинхронно. Це означає, що дані переривання можуть використовуватися для пробудження мікроконтролера з режимів глибокого сну. Синхронізація введення-виводу зупиняється у всіх режимах сну за винятком режиму холостого ходу (Idle).

Зверніть увагу, що при використанні переривання за рівнем для пробудження мікроконтролера з режиму вимикання (Power-down), тільки після втримання рівня, що змінився, протягом певного часу генерується переривання. Це робить мікроконтролер менш чутливим до шумів. Оцінка зміни стану рівня виконується по двом його вибіркам з інтервалом рівним періоду сторожового таймера, який рівний 1 мкс (номінальне значення) при 5.0В. Частота сторожового таймера залежить від напруги. Пробудження мікроконтролера настає, якщо на вході присутній необхідний рівень у процесі вибірок або якщо він утримується до закінчення затримки при запуску синхронізації (виникає при виході з режимів сну). Час запуску визначається конфігураційними бітами SUT. Якщо двічі виконана вибірка рівня із синхронізацією сторожовим таймером, але після закінчення часу запуску цей рівень зник, то пробудження мікроконтролера настане, але переривання не буде генеровано. Для того щоб активізувати переривання за рівнем необхідно, щоб цей рівень утримувався протягом достатнього для пробудження мікроконтролера часу.

Таблиця 3.2-Регістр А керування зовнішніми перериваннями – EICRA

Розряд	7	6	5	4	3	2	1	0
	ICS31	ICS30	ICS21	ICS20	ICS11	ICS10	ICS01	ICS00

Даний регістр на доступний у режимі сумісності з Atmega103, але вихідна установка відповідає налаштуванню на переривання по низькому рівню на INT3:0, що також як і в Atmega103.

Розряди 7..0 – ISC31, ISC30 – ISC00, ISC00: Біти вибору умови генерації зовнішнього переривання 3 - 0

Зовнішні переривання 3 - 0 активізуються через зовнішні виводи INT3:0, якщо встановлені прапор I у регістрі статусу SREG і відповідна маска переривання в EIMSK. Вибір рівня або фронту для активізації зовнішнього переривання здійснюється відповідно до таблиці 48. Фронти на INT3..INT0 виявляються асинхронно. Переривання по вив. INT3:0 буде сгенеровано, якщо тривалість імпульсу буде більше мінімально необхідною (див. табл. 3.3). При виникненні імпульсів меншої тривалості генерація переривання не гарантується. Якщо обране переривання по низькому рівню, то для генерації переривання необхідно, щоб цей рівень залишався на колишньому низькому рівні до моменту завершення виконання поточної інструкції. Після дозволу переривання за рівнем воно буде генеруватися безупинно доти, поки на вході присутній низький рівень. При зміні біт Iscn може виникнути переривання. Тому, рекомендується спочатку відключити переривання Intn шляхом скидання біта дозволу переривання в регістрі EIMSK. Після цього, значення біт Iscn може бути змінено. І, нарешті, перед поновленням роботи переривань необхідно скинути прапор переривання Intn шляхом запису балка. 1 у прапор переривання (Intfn) у регістрі EIFR.

Таблиця 3.3– Завдання умови генерації запиту на переривання

Iscn1	Iscn0	Опис
0	0	Низький рівень на Intn генерує запит на переривання
0	1	Зарезервоване
1	0	Падаючий фронт на Intn генерує асинхронно запит на переривання
1	1	Наростаючий фронт на Intn генерує асинхронно запит на переривання

Перед зміною біт Iscn1/Iscn0 необхідно заборонити роботу переривання шляхом очищення біта дозволу переривання в регістрі EIMSK. А якщо ні, то переривання може виникнути після зміни даних біт.

Таблиця 3.4-Регістр В керування зовнішніми перериваннями – EICRB

Розряд	7	6	5	4	3	2	1	0	EICRB
	ICS71	ICS70	ICS61	ICS60	ICS51	ICS50	ICS41	ICS40	

Розряди 7..0 – ISC71, ISC70 - ISC41, ISC40: Біта вибору умови генерації зовнішнього переривання 7 - 4

Зовнішні переривання 7 - 4 активізуються через зовнішні виводи INT7:4, якщо встановлені прапор I у регістрі статусу SREG і відповідна маска переривання в регістрі EIMSK. Умова, по якому генерується переривання, вибирається виходячи з даних таблиці 3.8. Для визначення фронтів на виводах INT7:4 здійснюється вибірка їх станів. Якщо обране переривання по фронту або зміні рівня, то переривання буде сгенеровано, якщо на вході з'являється імпульс, тривалість якого більше одного періоду синхронізації. При дії на вході більш коротких імпульсів генерація переривання не гарантується. Зверніть увагу, що частота синхронізації ЦПУ може бути нижче чому частота XTAL, якщо дозволена робота дільника частоти XTAL. Якщо обране переривання по низькому рівню, то переривання генерується, якщо до моменту закінчення виконання поточної інструкції на вході по колишньому є присутнім низький рівень. Якщо дозволене переривання за рівнем, то воно буде генеруватися безупинно доти, поки на вході присутній низький рівень.

Таблиця 3.5 – Завдання умови генерації запиту на переривання

Iscl1	Iscl0	Опис
0	0	Низький рівень на Intn генерує запит на переривання
0	1	Будь-яка зміна логічного стану на Intn генерує запит на переривання
1	0	Падаючий фронт, виявлений по двом вибіркам на Intn, генерує запит на переривання.
1	1	Наростаючий фронт генерує запит на переривання.

Прим. 1: $n = 7, 6, 5$ або 4 .

Перед зміною біт $Iscn1/Iscn0$ необхідно заборонити роботу переривання шляхом очищення біта дозволу переривання в регістрі EIMSK. А якщо ні, то переривання може виникнути після зміни даних біт.

Таблиця 3.6-Регістр маски зовнішнього переривання – EIMSK

Розряд	7	6	5	4	3	2	1	0
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Розряди 7..0 – INT7 – INT0: Дозвіл запиту на зовнішнє переривання 7 - 0

Якщо в біт INT7 – INT0 і в біт I регістру статусу SREG записати 1, то дозволяється робота зовнішнього переривання по відповідному до виводу. Біти вибору умови генерації переривання в регістрах керування зовнішніми перериваннями EICRA і EICRB визначають по якій умові генерується переривання: по наростаючому фронту, по падаючому фронту або за рівнем. Кожної з даних виводів зберігає активність, навіть якщо він настроєний на вивід. Дана особливість може використовуватися для програмної генерації переривання.

Таблиця 3.7 EIFR: Прапори зовнішніх переривань 7 - 0

Розряд	7	6	5	4	3	2	1	0
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0

Якщо фронт або зміна логічного стану на INT7:0 викликає переривання, то відповідний прапор INTF7:0 ухвалює одиничний стан. Якщо встановлені біт I регістру статусу SREG і відповідний біт дозволу переривання INT7:0 у регістрі EIMSK, то мікроконтролер виконає перехід на вектор переривання. Прапор скидається апаратно після виконання процедури обробки переривання. Альтернативно прапор може бути скинутий програмно шляхом запису 1 у відповідний біт. Якщо INT7:0 настроєні на генерацію переривання за рівнем, то прапори постійно перебувають у скинутому стані. Зверніть увагу, що при переході в режим сну з відключеними перериваннями INT3:0 вхідні буфери цих виводів будуть відключеними. У свою чергу це може викликати зміну внутрішнього стану сигналів, який приведе до установки прапорів INTF3:0.

3.3 Приклад налаштування INT1 та INT5 на падаючий фронт

cli

```
ldi tmp, (1<<ISC11)
STS EICRA,tmp ;по спаду
ldi tmp, (1<<ISC51)
STS EICRB,tmp ;по спаду
in tmp,EIMSK
ori tmp, (1<<INT1) | (1<<INT5)
out EIMSK,tmp
sei
```

3.4 Варіанти індивідуальних завдань

Таблиця 3.5-Таблиця завдань до лабораторної роботи №3

№	Текст індивідуального завдання
1	При послідовному натисканні будь-яких кнопок, відображати номер кнопки на Д. інд.
2	Після натискання кнопки S1, здійснити періодичне миготіння з частотою 2 Гц на інд.числа 43.
3	У двійковому виді відображати на одиничних індикаторах D1-D4 номер натиснутої кнопки.
4	Здійснити ввімкнення/вимикання одиничних індикаторів D1-D4 у кількості, відповідній номеру натиснутої кнопки, відобразити номер кнопок на інд.
5	На інд. відобразити число 12. По натисканню кнопки S2, здійснювати зменшення на одиницю відображуваного числа.
6	По натисканню кнопки S1 на клавіатурі включити секундомір з відображенням його значення на інд. При натисканні кнопки S2 здійснити скидання секундоміра.
7	Натиснути кнопку S1 відобразити на Сінд. число 10. Натиснути S2, відобразити на інд. число 11.
8	Відобразити на інд. число 4321. По натисканню кнопки S1 зменшувати на одиницю відображуване число, по натисканню S2 збільшувати на одиницю відображуване число.

3.5 Контрольні питання

3.5.1 Структура системи переривань AVR ATmega.

3.5.2 Призначення, приклади застосування системи переривань.

3.5.3 Переривання від таймерів, послідовного прийомопередавача.

3.5.4 Апаратне усунення деренчання контактів для схем з ТТЛ і КМОП.

3.5.6 Програмне усунення деренчання контактів.

3.5.7 Необхідність застосування апаратного або програмного усунення деренчання контактів.

4 ЛАБОРАТОРНАЯ РАБОТА №4 ИЗУЧЕНИЕ ТАЙМЕРОВ МИКРОКОНТРОЛЛЕРА AVR

Мета роботи: вивчити таймери мікроконтролера AT90USB162, навчитися керувати режимами таймерів мікроконтролера AT90USB162 і обробляти переривання від таймера.

4.1 Теоретичні відомості

Таймери в 8-розрядних мікроконтролерах AVR являють собою 8- або 16-розрядні лічильники імпульсів. Джерелом рахункових імпульсів може бути тактовий генератор ОМК або зовнішній датчик або генератор імпульсів. Частота рахункових імпульсів, що надходять у таймер, може бути зменшена в попередньому дільнику частоти (предделителе) таймера. Предделитель у сутності також є лічильником з фіксованим набором меж рахунку, що визначають коефіцієнти розподілу, і видає в таймер імпульс наприкінці кожного періоду свого рахунку.

Перед складанням програми необхідно задатися значенням тактової частоти мікроконтролера й розрахувати необхідні для відліку часу коефіцієнти розподілу й модулі рахунку таймерів, що забезпечують генерування сигналів заданих частот..

Якщо потрібно сформулювати звук, то зручно використати режим СТС таймерів або який-небудь із режимів PWM з керованим модулем (межею) рахунку й шпаруватістю імпульсів $Q = 2$. Крім цього, у ряді випадків зручно використовувати переривання таймерів для керування процесом виводу імпульсів (наприклад, для запуску, зупинки або перепрограмування іншого таймера).

Нехай потрібно вивести імпульси із частотою $f = 666$ Гц, а системна тактова частота мікроконтролера $f_{clk\ I/O} = 4$ МГц. У режимі СТС таймерів частота імпульсів визначається вмістом регістру порівняння OCR:

$$f = \frac{f_{clk\ I/O}}{2N(1 + OCR)}, \quad (1)$$

де N – коефіцієнт розподілу предделителя таймера.

Задамо значення $N = 64$ з ряду припустимих значень. Виразимо з формули (1) значення OCR:

$$OCR = \frac{f_{clk/O}}{2Nf} - 1 = \frac{4000000}{2 \cdot 64 \cdot 666} - 1 = 45,92 \quad (2)$$

Отримане значення не перевищує максимальне число, що вміщається в розрядності таймерів (255 для 8-розрядних і 65534 для 16-розрядних), і його можна використовувати, округливши до найближчого цілого $OCR = 46$. Помилка завдання частоти при цьому буде

$$\Delta f = \frac{|f_{окр} - f|}{f} \cdot 100 \%$$

де $f_{окр}$ – значення частоти, розраховане після округлення значення OCR.

Якщо помилка Δf виявиться більше припустимої, а розрахункове значення OCR при цьому буде відносно малим (менш 10 % від максимального), то слід вибрати менше значення N .

4.1.1 Timer/Counter0

Таймер-Лічильник T0 використовує два виводи мікроконтролера Atmega16. Вивід T0 - це вхід зовнішнього тактового сигналу. Він може застосовуватися, наприклад, для підрахунку імпульсів. Вивід OC0 - це вихід схеми порівняння таймера-лічильника. На цьому виводі за допомогою таймера може формувати меандр або ШИМ сигнал. Також він може просто міняти свій стан при спрацюванні схеми порівняння.

Виводи T0 і OC0 задіюються тільки при відповідних налаштуваннях таймера, у звичайному стані це виводи загального призначення.

4.1.1.1 Регістри таймера-лічильника T0

Таймер T0 має у своєму складі три регістри:

- рахунковий регістр TCNT0,
- регістр порівняння OCR0,

- конфігураційні регістри TCCR0A, TCCR0B.

Крім того, є ще три регістри, що ставляться до всіх таймерів Atmega:

- конфігураційний регістр TIMSK,
- статусний регістр TIFR.
- регістр спеціальних функцій SFIOR

TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Це 8-мі розрядний рахунковий регістр. Коли таймер працює, по кожному імпульсу тактового сигналу значення TCNT0 змінюється на одиницю.

Залежно від режиму роботи таймера, рахунковий регістр може або збільшуватися, або зменшуватися. Регістр TCNT0 можна як читати, так і записувати. Останнє використовується коли потрібно задати його початкове значення. Коли таймер працює, змінювати його вміст TCNT0 не рекомендується, тому що це блокує схему порівняння на один такт.

TIFR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TOV0 – прапор переповнення

OCF0A, OCF0B – прапор події рівності відповідно OCR0A, OCR0B

OCR0A, OCR0B

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

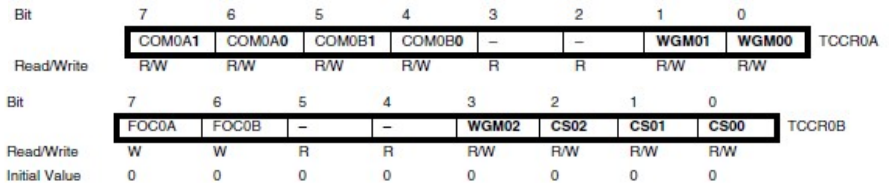
Це 8-мі розрядний регістр порівняння. Його значення постійне рівняється з рахунковим регістром TCNT0, і у випадку збігу таймер

може виконувати якісь дії - викликати переривання, міняти стан виводу OC0 і т.д. залежно від режиму роботи.

Значення OCR0 можна як читати, так і записувати.

TCCR0A, TCCR0B

Це два конфігураційний регістр таймера-лічильника T0, формати яких майже повністю збігаються з регістрами TCCR1A, TCCR1B :



Біти CS02, CS01, CS00 (Clock Select) - визначають джерело тактової частоти для таймера T0 і задають коефіцієнт подільника. Усі можливі стани описані в таблиці нижче.

Таблиця 4.1 – Біти CS

CS02	CS01	CS00	Описание
0	0	0	Источника тактирования нет. Таймер остановлен.
0	0	1	Тактовая частота МК
0	1	0	Тактовая частота МК/8
0	1	1	Тактовая частота МК/64
1	0	0	Тактовая частота МК/256
1	0	1	Тактовая частота МК/1024
1	1	0	Внешний источник на выводе T0. Срабатывание по заднему фронту
1	1	1	Внешний источник на выводе T0. Срабатывание по переднему фронту

Біти WGM10, WGM00 (Wave Generator Mode) - визначають режим роботи таймера-лічильника T0. Усього їх може бути чотири - нормальний режим (normal), скидання таймера при збігу (CTC), і два режими широтно-імпульсної модуляції (Fastpwm і Phase Correct PWM). Усі можливі значення описані в таблиці нижче.

Таблиця 4.2 – Біти WGM

WGM01	WGM00	Режим работы таймера/счетчика
0	0	Normal
0	1	PWM, Phase Correct
1	0	CTC
1	1	Fast PWM

Біти COM01, COM00 (Compare Match Output Mode) - визначають поведінка виводу OC0. Якщо хоч один із цих битов установлений в 1, то вивід OC0 перестав функціонувати як звичайний вивід загального призначення й підключається до схеми порівняння таймера лічильника T0. Однак при цьому він повинен бути ще настроєний як вихід.

І останній біт регістру TCCR0B - це біт FOC0 (Force Output Compare). Цей біт призначений для примусової зміни стану виводу OC0. Він працює тільки для режимів Normal і CTC. При установці біта FOC0 в одиницю стан виводу міняється відповідно значенням битов COM01, COM00. FOC0 біт не викликає переривання й не скидає таймер в CTC режимі.

TIMSK (Timer/Counter Interrupt Mask Register)

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Загальний регістр для всіх трьох таймерів Atmega16, він містить прапори дозволу переривань. Таймер T0 може викликати переривання при переповненні рахункового регістру TCNT0 і при збігу рахункового регістру з регістром порівняння OCR0. Відповідно для таймера T0 у регістрі TIMSK зарезервовано два біти - це TOIE0 і OCIE0. Інші біти ставляться до інших таймерів.

TOIE0 - 0-е значення біта забороняє переривання по події переповнення, 1 - дозволяє.

OCIE0 - 0-е значення забороняє переривання по події збіг, а 1 дозволяє.

Природно переривання будуть викликатися, тільки якщо встановлений біт глобального дозволу переривань - біт I регістру SREG.

TIFR (Timer/Counter0 Interrupt Flag Register)

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Загальний для всіх таймерів-лічильників регістр. Містить статусні прапори, які встановлюються при виникненні подій. Для таймера T0 - це переповнення рахункового регістру TCNT0 і збіг рахункового регістру з регістром порівняння OCR0.

Якщо в ці моменти в регістрі TIMSK дозволені переривання й установлений біт I, то мікроконтролер викличе відповідний оброблювач.

Прапори автоматично очищаються при запуску оброблювача переривання. Також це можна зробити програмно, записавши 1 у відповідний прапор.

TOV0 - установлюється в 1 при переповненні рахункового регістру.

OCF0 - установлюється в 1 при збігу рахункового регістру з регістром порівняння

SFIOR (Special Function IO Register)

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

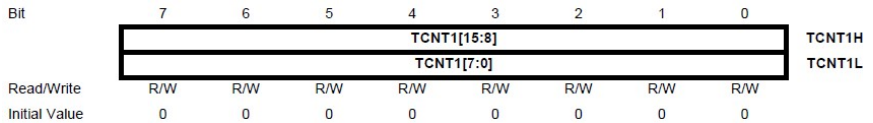
Початківцеві про цей регістр у принципі можна й не знати, один з його розрядів сбросує 10-ти розрядний двійковий лічильник, який ділить вхідну частоту для таймера T0 і таймера T1.

Скидання здійснюється при установці біта **PSR10 (Prescaler Reset Timer/Counter1 і Timer/Counter0)** в одиницю.

4.1.2 Timer/Counter1

Докладно розглянемо 16 розрядний регістр (16-bit Timer/Counter1) у нашому МК

4.1.2.1 Регістри таймера-лічильника T1



TCNT1(L/H) – це два 8 бітні регістри які в сумі дають один 16 бітний регістр. Коли таймер працює, по кожному імпульсу тактового сигналу значення TCNT1(L/H) змінюється на одиницю. Залежно від режиму роботи таймера, рахунковий регістр може або збільшуватися, або зменшуватися. Регістр TCNT1(L/H) можна як читати, так і записувати. Останнє використовується коли потрібно задати його початкове значення. Коли таймер працює, змінювати його вміст TCNT1(L/H) не рекомендується, тому що це блокує схему порівняння на один такт.

Запис у старший регістр (Tcntxh) ведеться спочатку в регістр TEMP. Цей регістр чисто службовий, і нам ніяк недоступний.

Записуємо старший байт у регістр TEMP (для нас це Tcntxh), а потім записуємо молодший байт. У цей момент, у реальний Tcntxh, заноситься раніше записане нами значення. Тобто два байти, старший і молодший, записуються одночасно! Міняти порядок не можна!

Виглядає це так:

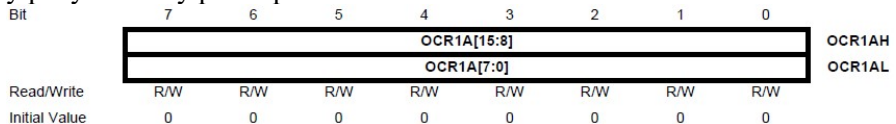
CLI ; Забороняємо переривання, в обов'язковому порядку!

STS TCNT1H, R16 ; Старшої байт спочатку в TEMP

STS TCNT1L, R17 ; А тепер записалося й у старший і молодший!

SEI ; Дозволяємо переривання

Читається всі також, тільки у зворотному порядку. Спочатку молодший байт (при цьому старший іде в TEMP), потім старший. Це гарантує те, що ми вважаємо саме той байт який був на даний момент у рахунковому регістрі.



OCR1A (L/H) – це два 8 бітні регістри які в сумі дають один 16 бітний регістр. Його значення постійне рівняється з рахунковим

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	—	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Біти ICNC1 і ICES1. ICNC1 включає/виключає схему придушення перешкод. Якщо він скинутий в 0, то схема пригнічення перешкод виключена й захват проводиться по першому ж активному фронту на виводі ICP1. Якщо біт установлений в 1, схема пригнічення перешкод включена й захват проводиться тільки у випадку 4-х однакових вибірок, відповідних до активного фронту сигналу. ICES1 визначає активний фронт сигналу, тобто фронт по якому блок захвата буде виконувати збереження рахункового регістру. Якщо біт скинутий в 0, активним є спадаючий фронт. Якщо встановлений в 1, то активним буде наростаючий фронт.

РЕГІСТР ЗАХВАТА: ICR1- це регістр захвата. У ньому зберігається значення рахункового регістру при сигналі на ніжці ICP1.

Біти COM1(A /B)0 і COM1(A/B)1 - визначають поведінку виводу OC1(A/B). Якщо хоч один із цих битов установлений в 1, то вивід OC1(A/B) перестав функціонувати як звичайний вивід загального призначення й підключається до схеми порівняння таймера лічильника T1. Однак при цьому він повинен бути ще настроєний як вихід. Поведінка виводу OC1(A/B) залежить від режиму роботи таймера-лічильника T1. У режимах normal і CTC вивід OC1(A/B) поводиться однаково, а от у режимах ШІМ його поведінка відрізняється.

Таблиця 4.3 –Значення битов COM для режимів Normal і CTC

COM1A1/COM1B1	COM1A0/COM1B0	Описание
0	0	Ножка отключена от T1
0	1	Состояние вывода меняется на противоположное
1	0	Вывод OC1(A/B) сбрасывается "0"
1	1	Вывод OC1(A/B) устанавливается в "1"

Таблиця 4.4 –Значення бітів COM для режиму Fast PWM

COM1A1/COM1B1	COM1A0/COM1B0	Описание
0	0	Ножка отключена от T1
0	1	Зарезервировано
1	0	Сбрасывается в "0" при прямом счете, устанавливается "1" при обнулении.
1	1	Устанавливается в "1" при прямом счете, сбрасывается в "0" при обнулении.

Таблиця 4.5 –Значення бітів COM для режимів Phase і Frequency Correct

COM1A1/COM1B1	COM1A0/COM1B0	Описание
0	0	Ножка отключена от T1
0	1	Зарезервировано
1	0	Сбрасывается в "0" при прямом счете, устанавливается "1" при обратном.
1	1	Устанавливается в "1" при прямом счете, сбрасывается "0" при обратном.

Біти WGM10, WGM11, WGM12, WGM13 (Wave Generator Mode) - визначають режим роботи таймера-лічильника T1. Усього їх може бути чотири нормальний режим (normal), скидання таймера при збігу (CTC), і два режими широтно-імпульсної модуляції (Fastpwm і Phase Correct PWM). Усі можливі значення описані в таблиці нижче.

Таблиця 4.6 –Значення бітів WGM

Режим	WGM				Режим работа таймера-счетчика	Верхний предел	Обновление OCRnх	Установка флага TOVnх:
0	0	0	0	0	Normal	0xFFFF	Сразу после записи	МАКС
1	0	0	0	1	8-битныйPhase Correct PWM	0x00FF	На вершине счета	Нижнем пределе
2	0	0	1	0	9-битныйPhase Correct PWM	0x01FF	На вершине счета	Нижнем пределе
3	0	0	1	1	10-битныйPhase Correct PWM	0x03FF	На вершине счета	Нижнем пределе
4	0	1	0	0	CTC	OCRnA	Сразу после записи	МАКС
5	0	1	0	1	8-битныйFast PWM	0x00FF	На вершине счета	На вершине счета
6	0	1	1	0	9-битныйFast PWM	0x01FF	На вершине счета	На вершине счета
7	0	1	1	1	10-битныйFast PWM	0x03FF	На вершине счета	На вершине счета
8	1	0	0	0	Phase & Frequency Correct PWM	ICRn	На нижнем пределе	Нижнем пределе
9	1	0	0	1	Phase & Frequency Correct PWM	OCRnA	На нижнем пределе	Нижнем пределе
10	1	0	1	0	Phase Correct PWM	ICRn	На вершине счета	Нижнем пределе
11	1	0	1	1	Phase Correct PWM	OCRnA	На вершине счета	Нижнем пределе
12	1	1	0	0	CTC	ICRn	Сразу после записи	МАКС.
13	1	1	0	1	(резерв)	-	-	-
14	1	1	1	0	Fast PWM	ICRn	На вершине счета	На вершине счета
15	1	1	1	1	Fast PWM	OCRnA	На вершине счета	На вершине счета

4.1.1 Ініціалізація таймера.

Ініціалізація таймера 0

```
ldi tmp, Prescaler
out TCCR0B, tmp
clr tmp
out TCCR0A, tmp
ldi tmp, Initcounter
out TCNT0, tmp
```

Ініціалізація переривання по переповненню таймера 0

```
lds tmp, TIMSK0
ori tmp, 1<<TOIE0
sts TIMSK0, tmp
sei
```

4.1.2 Обробка переривання від таймера.

Ініціалізація переривання по переповненню таймера 0

```
lds tmp,TIMSK0
ori tmp,1<<TOIE0
sts TIMSK0,tmp
sei
```

На початку програми слід розташувати код:

```
.org $002A          ;timer0 ovr
rjmp TIM0Int
```

і наприкінці – процедуру обробки переривання

```
TIM0Int:
;дії по перериванню
.....
reti
```

4.1.3 Затримка на таймері.

Приклад процедури затримки на таймері 0:

```
timdel:
;затримка на таймері 0
; максимально на 1/3.81 сек
    push tmp
    ldi tmp,Initcounter; початкове значення
    out TCNT0,tmp      ; лічильника
    sbi TIFR0,TOV0; чистимо прапор переповнення

read:
    in    tmp,TIFR0;read the Timer Flag Register
    sbrs  tmp,TOV0;test the overflow flag
    rjmp  read ; якщо не встановлений - на read
TOV0_is_set :
    sbi TIFR0,TOV0; скидання overflow flag
    pop tmp
    ret
```

4.2 Порядок виконання роботи

4.2.1 Скласти програму згідно індивідуальному завданню, використовуючи для формування тимчасових інтервалів або обробку переривання, або затримку на таймері.

4.2.2 Відкомпілювати програму із середовища й усунути помилки.

4.2.3 Отриманий hex файл завантажите в стенд.

4.3 Завдання до лабораторної роботи 4

Таблиця 4.7 - Таблиця завдань до лабораторної роботи №4

№	Текст індивідуального завдання
1	Використовуючи приклад у процедурі затримки організувати перемикання червоного, жовтого, зеленого світлодіодів як у світлофорі з інтервалом 1с.
2	За допомогою переривання від таймера перемикає синій і червоний світлодіоди. У це ж час моргати зеленим із частотою 2 Гц у основній програмі.
3	Мигати синім світлодіодом за допомогою переривання від таймера із частотою 4Гц.
4	Перемикає жовтий і зелений світлодіоди за допомогою переривання від таймера із частотою 1Гц.
5	Мигати зеленим світлодіодом за допомогою затримки на таймері, а червоним – із частотою 10Гц за допомогою переривання.
6	Мигати по три рази червоним, потім синім, потім зеленим світлодіодом за допомогою переривання від таймера із частотою 2Гц.
7	По черзі перемикає світлодіоди за годинниковою стрілкою за допомогою переривання від таймера із частотою 2Гц.
8	Мигати зеленим світлодіодом за допомогою переривання від таймера 0 із частотою 2 Гц, а синім світлодіодом за допомогою переривання від таймера 1 із частотою 4 Гц.

4.4 Контрольні питання

4.4.1 Счетные регистры таймеров.

4.4.2 Управляющие регистры таймеров.

4.4.3 Тактирование таймеров.

4.4.4 Настройка и перехват прерываний от таймеров.

5 ЛАБОРАТОРНА РОБОТА №5 ГЕНЕРАЦІЯ ШІМ

Мета роботи: Навчитися керувати апаратним ШІМ.

5.1 Порядок виконання лабораторної роботи

5.1.1 Скласти програму на мові С.

Створіть новий проект у Atmel Studio (File->New Project) та виберіть C/C++ -> GCC C Executable Project

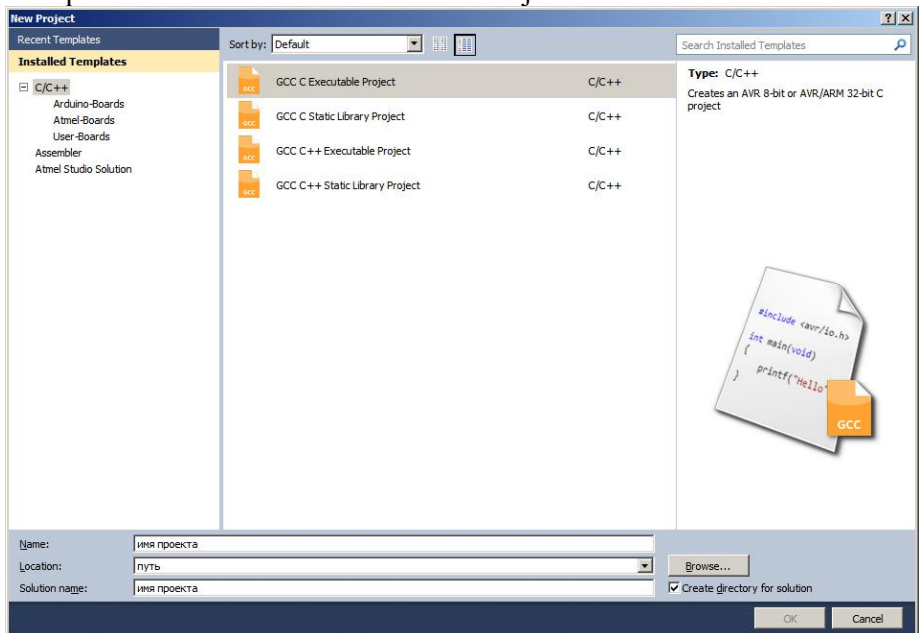


Рисунок 5.1 – Створення нового проекту

Шаблон програми може бути ось яким:

```
#define F_CPU 1000000 // використана частота CPU
#include <avr/io.h>
#include <avr/wdt.h>
#include <util\delay.h>

int main(void)
{
```

```

wdt_reset();
wdt_disable();
PORTB = 0b00000000;
DDRB=0xff;
PORTC = 0b00000000;
DDRC = 0b11111100;
PORTD = 0b00010010;
DDRD = 0b01101001;
}

```

5.1.2 Напишіть програму, відкомпілюйте. Запустить програму на стенді.

5.2 Короткі теоретичні відомості

Таблиця 5.1 –Відповідність виводів і каналів таймерів

Вивід	Канал таймера	Пристрій
PC5	OC1B	Червоний св.діод, динамік
PC2	Тільки програмний ШІМ	Синій св.діод
PC6	OC1A	Жовтий св.діод
PD0	OC0B	Зелений св.діод

5.2.1 Режим ШІМ

Режим Normal

Це найбільш простий режим роботи таймерів/лічильників. У цьому режимі рахунковий регістр функціонує як звичайний підсумовуючий лічильник. По кожному імпульсу тактового сигналу clk_{tn} інкрементується рахунковий регістр. При переході через значення $0xffff$ виникає переповнення, і рахунок триває зі значення $0x0000$. У тому ж такті сигналу clk_{tn} , у якому обнуляється регістр $Tcnt_{tn}$. Блоки порівняння таймерів у цьому режимі можуть використовуватися як для генерації переривань, так і для формування сигналів. Стан виходів Ocp_{ix} кожного із блоків порівняння 16-бітних таймерів-лічильників визначається вмістом бітів

Comna1...0, Omnb1...0, Comnc1...0 регістрів Tscna, як показано в таблиці 16-2.

Режим CTC (скидання при збігу)

У цьому режимі рахунковий регістр теж функціонує як звичайний підсумовуючий лічильник, інкрементирование якого здійснюється по кожному імпульсу тактового сигналу clk_{tn}. Однак максимально можливе значення рахункового регістру й, отже здатність, що дозволяє, лічильника визначаються або регістром порівняння блоку A Ocr_{na}, або регістром захвата Icr_n. Після досягнення максимального значення рахунок триває зі значення 0×0000.

Біти FOC1(A/B) (Force Output Compare). Цей біти призначені для примусової зміни стану виводів OC1(A/B). Біти працюють тільки для режимів Normal і CTC. При установці бітів в одиницю стан виводу міняється відповідно значенням бітів COM1(A/B)1, COM1(A/B)0. FOC1(A/B) біти не викликають переривання й не скидає таймер в CTC режимі.

Режим Fastpwm («Швидкодійний ШІМ»)

Режим Fast PWM дозволяє генерувати високочастотний сигнал із широтно-імпульсною модуляцією. Рахунковий регістр у цьому режимі функціонує як підсумовуючий лічильник, інкрементування якого здійснюється по кожному імпульсу тактового сигналу clk_{tn}. Стан лічильника змінюється від 0×0000 до максимального значення, після чого рахунковий регістр скидається й цикл повторюється. Залежно від установок бітів Wgmn3..0 максимальне значення лічильника (дозвіл ШІМ-Сигналу) або є фіксованим значенням (режими 5, 6, 7 з таблиці 7), або визначається вмістом певних регістрів таймера-лічильника (режими 14 і 15).

Режим Phase Correct PWM («ШІМ з точною фазою»)

Режим Phase Correct PWM, як і режим Fast PWM, призначений для генерації сигналів із широтно-імпульсною модуляцією. Однак у цьому режимі рахунковий регістр функціонує як реверсивний лічильник, стан якого спочатку змінюється від 0×0000 до максимального значення, а потім назад до 0×0000. Відповідно, максимальна частота сигналу в цьому режимі в 2 рази нижче максимальної частоти сигналу

в режимі Fast PWM. Залежно від установок бітів Wgmn3...0 максимальне значення лічильника (дозвіл ШІМ-Сигналу) або є фіксованим значенням, або визначається вмістом певних регістрів таймера/лічильника. Як і в режимі Fast PWM, при роботі з якими-небудь фіксованими значеннями модуля рахунку для завдання модуля рекомендується використовувати регістр захвата.

Режим Phase and Frequency Correct PWM («ШІМ з точною фазою й частотою»)

Режим Phase and Frequency Correct PWM дуже схожий на режим Phase Correct PWM. Єдина принципова різниця між ними — момент відновлення вмісту регістру порівняння. Максимальне значення лічильника (дозвіл ШІМ-Сигналу) у цьому режимі може визначатися тільки регістрами Icrna або Ocrna таймера-лічильника. У режимі Phase and Frequency Correct PWM кожний період сигналу є повністю симетричним. Цей наслідок того, що відновлення вмісту регістру порівняння відбувається в момент досягнення лічильником мінімального значення. Тому час прямого рахунку завжди дорівнює часу зворотного рахунку, вихідні імпульси симетричні, і відповідно частота генеруемого сигналу залишається постійною.

Таблиця 5.2 – Значення предделителя таймера

CSn12	CSn11	CSn10	Описание
0	0	0	Нет синхронизации. Таймер-счетчик остановлен.
0	0	1	clk/1 (без предделения)
0	1	0	clk/8 (с предделением)
0	1	1	clk/64 (с предделением)
1	0	0	clk/256 (с предделением)
1	0	1	clk/1024 (с предделением)
1	1	0	Внешний тактовый источник с выв. Тп. Синхронизация по падающему фронту.
1	1	1	Внешний тактовый источник с выв. Тп. Синхронизация по нарастающему фронту.

5.2.2 Приклад налаштування таймерів

```
// ініціалізація й запуск таймера 1
TCNT1 = Initcounter;
/* Restore global interrupt flag */
SREG = sreg;
TCCR1A = 0;
TCCR1C = 0;
TCCR1B = (7 & Prescaler);
//Fast PWM, 8-bit, канали A і B
TCCR1A|=(1<< COM1A1) | (1<< COM1B1) | (1<<WGM10);
TCCR1B|=(1<<WGM12);
// керування шпаруватістю (плавне запалювання)
for (volatile u8 i=0;i<255;i++)
{
    OCR1B=i;
    _delay_ms(10);
}
```

5.3 Завдання до лабораторної роботи 5

Таблиця 5.3 - Таблиця завдань до лабораторної роботи №5

№	Текст індивідуального завдання
1	Плавно запалювати червоний, жовтий, зелений світлодіоди з інтервалом 1с.
2	Плавно запалювати всі світлодіоди за годинниковою стрілкою. Гасити - проти.
3	Плавно запалювати всі світлодіоди проти годинниковий стрілки. Гасити – по.
4	Плавно запалювати червоний і зелений світлодіоди в протифазе.
5	Плавно запалювати червоний, жовтий, зелений світлодіоди відразу.
6	Плавно запалювати жовтий світлодіод. Моргати червоним із частотою 2Гц.
7	Імітувати зеленим світлодіодом процес запалювання /гасіння люмінесцентної лампи.
8	Поступове плавне гасіння червоного, жовтого, зеленого світлодіодів у продовж 5 сек.

5.4 Контрольні питання

5.4.1 Що таке ШИМ?

5.4.2 Які режими ШИМ є в Atmel?

5.4.3 Регістри AVR, які управляють режимами ШИМ.

5.4.4 Керування ШИМ у програмі на C.

5.4.5 Відповідність каналів компаратора виводам ШИМ.

6 ЛАБОРАТОРНА РОБОТА №6 СТВОРЕННЯ БІБЛІОТЕКИ КЕРУВАННЯ ДИНАМІЧНОЮ ІНДИКАЦІЄЮ

Мета роботи: Навчитися складати мовою С бібліотеки для керування периферією мікроконтролера.

6.1 Порядок виконання лабораторної роботи

6.1.1 Скласти програму на мові С.

6.1.2 Запустити програму на стенді.

6.2 Стисли теоретичні відомості

Налаштування таймера на генерацію переривань:

```
TCCR1B = (1 << CS12);
TCNT1 =TIMCOUNINTVALUE;
TIMSK1= (1<<TOIE1);
sei(); // enable interrupts
```

Щоб у програмі мовою С можна було перехоплювати оброблювач переривання, використовується макрос ISR з параметром. Так, для переривань від таймера оброблювач може виглядати як:

```
ISR(TIMER0_OVF_vect) //переповнення таймера 0
{
}

ISR(TIMER1_OVF_vect) //переповнення таймера 1
{
}
```

При цьому оброблювач переривання може перебувати в будь-якому *.c файлі проекту, що можна використовувати у своїй бібліотеці динамічної індикації.

Константи макросу ISR перераховані в бібліотечному хедері iosubxx2.h, який підключається автоматично при підключенні <avr/interrupt.h>

6.3 Завдання до лабораторної роботи 6

Напишіть бібліотеку мовою С, яка реалізує за допомогою переривання від таймера відображення масиву цифр числа **unsigned**

char HG[4] на динамічному індикаторі. Оголосите його глобальної змінної в хедері.

Напишіть реалізацію функцій:

Printhex(u16 word); // виводить слово в шестнадцятирічному
// виді

Printword(u16 word); // виводить десяткове беззнакове

Printbyte(u8 byte); // виводить байт у шестнадцятирічному
// виді

Printint(short int i); // виводить байт як десяткове знакове

З використанням даної бібліотеки виконаєте завдання згідно з таблицею 6.1.

Таблиця 6.1 - Таблиця завдань до лабораторної роботи №6

№	Текст індивідуального завдання
1	Занести в змінну I байт X, у змінну R6 - Y, суму чисел відобразити на індикаторах.
2	Занести в змінну слово та відобразити його на індикаторах.
3	Занести в змінну В ДД число, з частотою 2 Гц виводити це число на індикаторах.
4	Занести в змінну А ДД число XX, у змінну В – X0, число з R26 відобразити на інд. HG1, HG0, число X з R25 відобразити на інд. HG2.
5	Занести в змінну А ДД число 0X, у змінну Y – X0, суму чисел відобразити на індикаторах
6	Занести в змінну В байт , у регістр Z – байт, суму чисел відображати на індикаторах частотою 0,5Гц.
7	Занести в змінну А ДД число XX, поперемінно відображати молодшу і старшу тетраду на інд. HG3, HG0 з частотою 1 Гц.
8	Занести у змінну А ДД число X0, у змінну С - XX, число X з С відображати на інд. HG0 з частотою 1 Гц, число з А відображати на інд. HG3, HG2 з частотою 0,5 Гц.
11	Занести в X ДД число X0, у змінну R3 - 0X, суму чисел відобразити на інд. HG3, HG0.
12	Занести в змінну R22 байт, у змінну R5 – байт, по черзі відображати на R22, R5 та їх додаток індикаторах .
13	Занести в змінну R21 ціле число 0X, віднімаючи від числа одиницю відображати на індикаторах отримане значення до нуля з частотою 1 Гц.

14	Занести в змінну R23 ДД число XX, у змінну R5 XX, поперемінно відобразити ці числа на інд. HG3, HG2, HG1, HG0.
15	Занести в змінну R1 ДД число 0X, у змінну R2 – X0, число X з R2 відобразити на інд. HG3, число X з регістру R1 відобразити на інд. HG1 з частотою 0.5 Гц

6.4 Контрольні питання

- 6.4.1 Керування динамічним індикатором.
- 6.4.2 Перехоплення переривання із програми на С.
- 6.4.3 Налаштування таймера на генерацію переривань у програмі на С.
- 6.4.4 Модульне програмування на С (використання *.c і *.h файлів).

7 ЛАБОРАТОРНА РОБОТА №7 ГЕНЕРАЦІЯ ЗВУКА

Мета роботи: Навчитися генерувати сигнали з допомогою мікропроцесору.

7.1 Порядок виконання лабораторної роботи

7.1.1 Скласти програму на мові C.

7.1.2 Запустити програму на стенді.

7.2 Короткі теоретичні відомості

Прямокутні імпульси можна одержати, подаючи на вихід мікроконтролера по черзі високий і низький рівень із затримкою, яка залежить від частоти імпульсу.

```
for(int i=0; i<N;i++)
{
    _delay_us(T);
    PORTx|=1<<Nbit;
    _delay_us(T);
    PORTx&=~(1<<Nbit);
}
```

Таблиця 7.1 Частота звучання музикальних нот.

Нота	Частоти звучання по октавам			
	Малая октава	Первая октава	Вторая октава	Третья октава
до	130.810	261.630	523.250	1046.500
до #	138.590	277.180	554.370	1108.730
ре	146.830	293.660	587.330	1174.700
ре #	155.560	311.130	622.260	1244.510
ми	164.810	329.630	659.260	1318.500
фа	174.610	349.230	698.460	1396.900
фа #	185.000	369.990	739.990	1479.980
соль	196.000	392.000	783.990	1568.000
соль #	207.650	415.300	830.610	1661.220
ля	220.000	440.000	880.000	1760.000
ля #	233.080	466.160	932.330	1864.660
си	246.940	493.880	987.770	1975.000

7.3 Задання к лабораторной работе 7

Таблиця 7.2 - Таблиця завдань до лабораторної роботи №7

№	Текст індивідуального завдання
1	При натисканні на кнопку S1 видавати гудок 0.5 сек частотою 400Гц.
2	При натисканні на кнопку S2 видавати трель тривалістю 3 сек зі звуків частотою 100 Гц і 1000 Гц.
3	Виводити на індикатор зворотний відлік від 10 сек. По закінченню відліку видати 3 коротких гудка й мигати синім світлодіодом.
4	При натисканні на кнопку S1 видавати гудок 1сек частотою 100 Гц, а при натисканні на кнопку S2 - гудок 1сек частотою 440 Гц.
5	За допомогою ШІМ імітувати наростаючу й убутну сирену.
6	Виводити на індикатор зворотний відлік від 5 сек. По закінченню відліку видати 3 довгих гудка й мигати зеленим світлодіодом.
7	Зіграти: до, сіль, до, фа, до, си, ля, сіль, фа, мі, ре, до
8	У циклі: зіграти ля й запалити синій діод, потім мі й синій діод.

7.4 Контрольні питання

7.4.1 Як згенерувати прямокутні імпульси на виході МК?

7.4.2 Як змінювати гучність звуку на МК?

7.4.3 Як згенерувати імпульси довільної форми?

8 ЛАБОРАТОРНА РОБОТА №8 КЕРУВАННЯ СИСТЕМОЮ ЧАСТОТОЮ ТА ВИЗНАЧЕННЯ ПРОДУКТИВНОСТІ МІКРОКОНТРОЛЕРУ

Мета роботи: Навчитися складати тестові послідовності. Навчитися оцінювати ефективність мікропроцесора та керувати його системною частотою.

8.1 Порядок виконання лабораторної роботи

8.1.1 Скласти тестову послідовність команд та програму на мові С або асемблері яка використовує зовнішній генератор частоти згідно завданню.

8.1.2 Вимірити час виконання тестової послідовності за допомогою секундоміра. Для цього необхідно вставити тестову послідовність у цикл як затримку між миготінням індикатора і вимірити період миготіння. Потім обчислити продуктивність. Виміри зробити 5-6 разів, обчислити середнє і похибку.

8.1.3 Виміряти продуктивність програми при використанні внутрішнього осцилятора й записати у звіт.

8.2 Стисли теоретичні відомості

Приклад функції, яка перемикає контролер на зовнішній генератор тактирування (кварц+PLL):

```
void Clockswitch2EXT()
{
    CLKSEL0 |= (1<<EXTE) ;
    while ( (CLKSTA & (1<<EXTON) ) == 0 ) ;
    CLKSEL0 |= 1<<CLKS ;
    PLLCSR |= 1<<PLLE ;
    CLKSEL0 &= ~(1<<RCE) ;
    //while ( (PLLCR & (1<<PLOCK) ) == 0 ) ;
}
```

Установка дільника:

```
void Clocksetdivisor(unsigned char divisor)
{ //divisor ціле від 0 (f/1) до 8 (f/256)
    divisor = 0xf & divisor ;
    CLKPR = (1<<CLKPCE) ;
```

```

        CLKPR =divisor;
    }

```

Приклад функції, яка перемикає контролер на внутрішній генератор тактування

```

void Clockswitch2RC(void)
{
    PLLCSR&=~(1<<PLLE);
    CLKSEL0|=1<<RCE;
    while( (CLKSTA&(1<<RCON) ) ==0 );
    CLKSEL0&=~(1<<CLKS);
    CLKSEL0&=~(1<<EXTE);
}

```

8.3 Завдання до лабораторної роботи №8

Таблиця 8.1 - Таблиця завдань до лабораторної роботи №8

№	Текст індивідуального завдання
1	Виміряти продуктивність мікроконтролера на частоті 16МГц.
2	Виміряти продуктивність мікроконтролера на частоті 8МГц.
3	Виміряти продуктивність мікроконтролера на частоті 4МГц.
4	Виміряти продуктивність мікроконтролера на частоті 2МГц.
5	Виміряти продуктивність мікроконтролера на частоті 1МГц.
6	Виміряти продуктивність мікроконтролера на частоті 0.5МГц.
7	Виміряти продуктивність мікроконтролера на частоті 0.25МГц.
8	Виміряти продуктивність мікроконтролера на частоті 0.125МГц.

8.4 Контрольні питання

8.4.1 Як влаштована схема тактування AT90USB162?

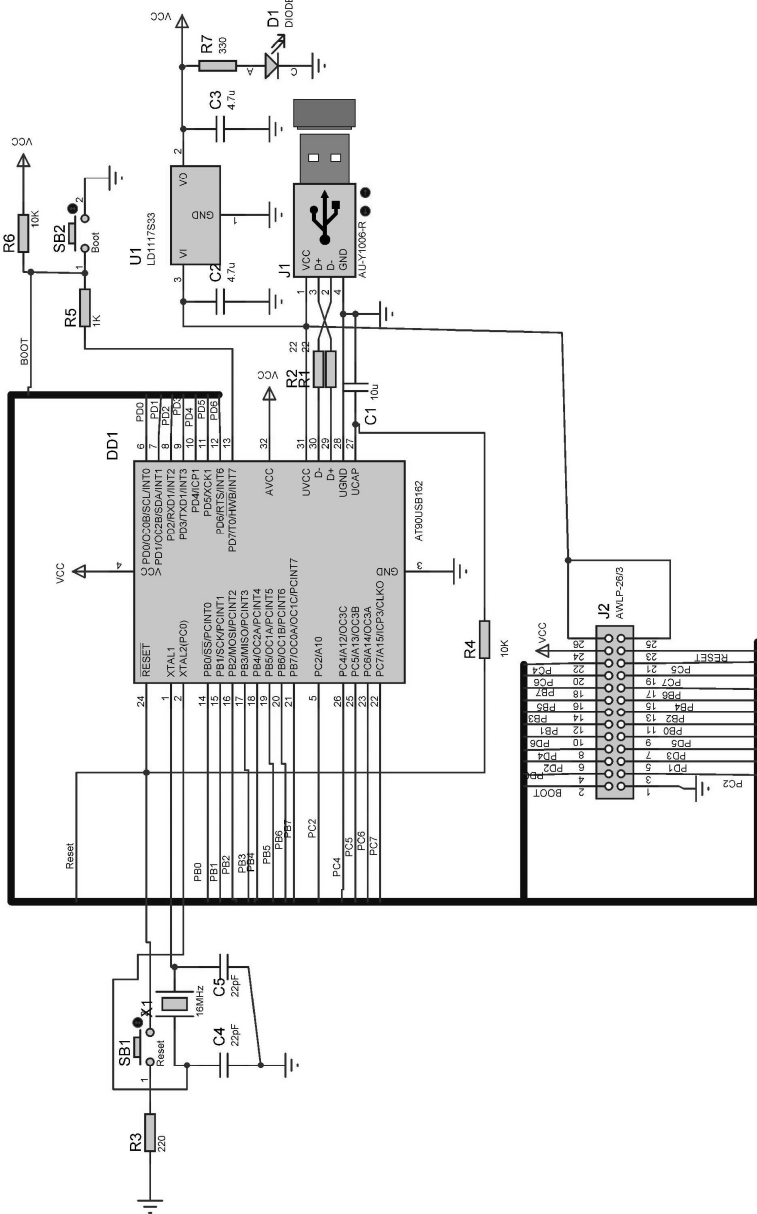
8.4.2 Які регістри беруть участь у завданні режиму тактування?

8.4.3 Які режими тактування існують в AT90USB162?

ЛИТЕРАТУРА

1. Аппаратные средства персональных компьютеров. Самоучитель. - СПб.: BHV, 2003. – 512 с.
2. Белов А.В. Самоучитель по микропроцессорной технике. – СПб.: Наука и Техника, 2003. - 224 с.
3. Бибило П.Н. Основы языка VHDL. - М.: Солон-Р, 2002. – 224 с.
4. Гук М. Аппаратные средства IBM PC. Энциклопедия. 2-е издание. – СПб.: Питер Пресс, 2001. – 928 с.
5. Евстифеев А.М. Микроконтроллеры AVR семейства Classic фирмы ATMEL. - М.: Додэка, 2002. – 288 с.
6. Каган Б.М. Электронные вычислительные машины и системы. – М.: Выш. шк., 1991. -590 с.
7. Касаткин А.И., Вальвачев А.Н. Профессиональное программирование на языке Си: От Turbo C Borland C++. - М.: Выш. шк., 1992. – 240 с.
8. Майоров С.А., Кириллов В.В., Приблуда А.А. Введение в микро-ЭВМ. - Л.: "Машиностроение", 1988. – 304 с.
9. Нестеров П.В. и др. Микропроцессоры (в 3кн.) т.1 под ред. М.Н. Преснухина. - М: "Высшая школа", 1986. -495 с.
10. Новиков М.Н. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. - М.: Мир. – 2001. – 379 с.
11. Рудометов Г.А. Материнские платы и чипсеты. 2-е издание. - СПб.: Питер Пресс, - 2001. – 352 с.
12. Соловьев В.В. Проектирование цифровых систем на основе программируемых логических интегральных схем. - М.: Горячая Линия – Телеком., 2001. – 636 с.
13. Стешенко В.Б. ПЛИС фирмы ALTERA: элементная база, система проектирования и языки описания аппаратуры. - М.: Додэка - XXI, 2002. -576 с.
14. Тавернье И.А. PIC-микроконтроллеры. Практика применения. - М.: ДМК, 2002. – 272 с.
15. Фромберг Г. Конструкции на элементах цифровой техники. - М.: Горячая Линия – Телеком., 2002. – 264 с.

16. Фрунзе М.А. Микроконтроллеры? Это же просто! Том 1 М.: Додэка – XXI., - 2002 – 336 с.



Додаток Б

