

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

“ Комп'ютерна схемотехніка ”

для студентів спеціальності 8.091501 091501 «Комп'ютерні системи та мережі» та 8.091503 «Спеціалізовані комп'ютерні системи» всіх форм навчання

Частина II

Програмування ARM-мікроконтролерів.

2016

Методичні вказівки до лабораторних робіт з дисципліни "Комп'ютерна схемотехніка" для студентів спеціальності 8.091501 091501 «Комп'ютерні системи та мережі» та 8.091503 «Спеціалізовані комп'ютерні системи» всіх форм навчання.

Частина II. Програмування ARM-мікроконтролерів. /Укл.: С.О.Сгадов – Запоріжжя: ЗНТУ, 2016. – 38 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 №8 від 18.10.2016

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

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ.....	4
ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ.....	4
1 ЛАБОРАТОРНА РОБОТА №1 ВИВЧЕННЯ ПОРТІВ ВВЕДЕННЯ-ВИВОДУ.....	5
2 ЛАБОРАТОРНА РОБОТА №2 ВИВЧЕННЯ ТАЙМЕРІВ МІКРОПРОЦЕСОРА STM32F100.....	8
3 ЛАБОРАТОРНА РОБОТА №3 ГЕНЕРАЦІЯ ШИМ.	12
4 ЛАБОРАТОРНА РОБОТА №4 ПРОГРАМУВАННЯ ІНДИКАТОРУ WH1602.	15
5 ЛАБОРАТОРНА РОБОТА №5 ОБРОБКА РЕАКЦІЇ НА ЗОВНІШНІ ПОДІЇ.	21
6 ЛАБОРАТОРНА РОБОТА №6 ВИКОРИСТАННЯ семіхостинга ЧЕРЕЗ ПОСЛІДОВНИЙ ПОРТ.	25
7 ЛАБОРАТОРНА РОБОТА №7 ВИВЧЕННЯ ГОДИН РЕАЛЬНОГО ЧАСУ МІКРОПРОЦЕСОРА STM32F100.....	29
8 ЛАБОРАТОРНА РОБОТА №8 ВИЗНАЧЕННЯ ПРОДУКТИВНОСТІ DMA.	33
ДОДАТОК А. Схема стенда.....	38

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 C);

висновки з лабораторної роботи з роздрукованими результатами.

Виконаний звіт пред'являється наприкінці заняття для затвердження викладачу.

1 ЛАБОРАТОРНА РОБОТА №1

ВИВЧЕННЯ ПОРТІВ ВВЕДЕННЯ-ВИВОДУ.

Мета роботи: вивчити режими портів процесора STM32, навчитися управляти режимами виводів процесора STM32 і писати прості програми мовою C для зміни стану портів.

1.1 Теоретичні відомості

1.1.1 Ініціалізація порту

Ініціалізація порту відбувається у такий спосіб:

```
//Включаємо тактування порту GPIOB
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
//Конфігуруємо на вивід ніжки 0 і 1
GPIO_InitTypeDef GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin =GPIO_Pin_0|GPIO_Pin_1;
//Вибираємо виводи 0,1
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
//Максимальна швидкість роботи
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
//Вихід Push-Pull
GPIO_Init(GPIOB, &GPIO_InitStructure);
//Задаємо задані налаштування в регістри порту
```

1.1.2 Керування станом ніжок.

Можна встановлювати й очищати прямо через регістри BSRR і BRR.

```
GPIOB->BSRR = 0x00000005; // Установка 2 та 0
GPIOB->BRR   = 0x00000005; // Очищення бітів 2
                        //та 0:
```

або ODR:

```
GPIOB->ODR =0x00000005; //Установка стану порту
```

Або процедурами

```
GPIO_WriteBit(GPIOB,GPIO_Pin_8,Bit_SET);
GPIO_WriteBit(GPIOB,GPIO_Pin_8,Bit_RESET);
GPIO_SetBits(GPIOB,GPIO_Pin_8)
```

1.2 Порядок виконання роботи

1.2.1 Відкрити середовище розробки CoIDE і створити новий проект. При створенні проекту вкажіть директорію проекту й тип контролера (STM32F100C8). Урахуйте, що кожний проект розташовується в окремій папці!.

1.2.2 Скласти програму згідно з індивідуальним завданням.

1.2.3 Відкомпілювати програму із середовища й усунути помилки.

1.2.4 Отриманий hex файл завантажте в стенд за допомогою пакетного файлу load.bat ім'я.hex. Ви також можете прописати його виклик у конфігурації проекту для автоматичного виклику після компіляції.

1.3 Завдання до лабораторної роботи

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 0.5 секунд 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 1 сек послідовно запалювати жовтий, червоний, синій, зелений
6	Мигати світлодіодами, розташованими по одній і по іншій діагоналі.
7	Послідовно запалювати світлодіоди за годинниковою стрілкою
8	Послідовно запалювати світлодіоди проти годинникової стрілки

1.4 Контрольні питання

- 1.4.1. Перелічіть внутрішні шини STM32F100.
- 1.4.2. Перелічіть зовнішні паралельні порти STM32F100C8.
- 1.4.3. Як подати тактування на даний порт?
- 1.4.4. Яким образом задається режим ніжки порту?
- 1.4.5. Перелічіть можливі режими ніжок порту.
- 1.4.6. Регістри порту вводу-виводу.
- 1.4.7. Зміна стан ніжок порту.

2 ЛАБОРАТОРНА РОБОТА №2 ВИВЧЕННЯ ТАЙМЕРІВ МІКРОПРОЦЕСОРА STM32F100

Мета роботи: вивчити таймери процесора STM32, навчитися управляти режимами таймерів процесора STM32 і обробляти переривання від таймера.

2.1 Теоретичні відомості

2.1.1 Ініціалізація таймера.

Розглянемо ініціалізацію таймера на прикладі Таймера 3.

2.1.1.1. Переконайтеся, що або у вашій програмі, або в `stm32f10x_conf.h` є раскомментовані рядки:

```
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_tim.h"
#include "misc.h"
```

2.1.1.2. Подайте тактову частоту на таймер

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE)
```

2.1.1.3. Задати конфігурація таймера

```
TIM_TimeBaseInitTypeDef TIM_BaseConfig;
// Запускаємо таймер на тактовій частоті в 4800 khz
TIM_BaseConfig.TIM_Prescaler = (SystemCoreClock /
4800000) - 1;
// Період - 150 тактів => 4800000/65535 = 73.2 Hz
TIM_BaseConfig.TIM_Period = 0xffff;
TIM_BaseConfig.TIM_ClockDivision = 0;
// Відлік від нуля до TIM_Period
TIM_BaseConfig.TIM_CounterMode = TIM_CounterMode_Up;
// Ініціалізуємо таймер №3
TIM_TimeBaseInit(TIM3, &TIM_BaseConfig);
```

2.1.1.4. Включаємо таймер

```
TIM_Cmd(TIM3, ENABLE);
```

2.1.2 Обробка переривання від таймера.

Для того щоб включити обробку переривання від таймера потрібно викликати процедури:

```
NVIC_EnableIRQ (TIM3_IRQn);
__enable_irq(); // глобальний дозвіл преривания.
```

А також сконфігурувати подію, по якому таймер видасть запит на переривання:

```
TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
```

Перехоплення переривання відбувається за допомогою приміщення в код процедури

```
void TIMномертаймера_IRQHandler(void){}
```

Важливо не забувати скидати прапор переривання. Наприклад:

```
void TIM3_IRQHandler(void)
{
    if(TIM3->SR & TIM_SR_UIF) {
        TIM3->SR &= ~TIM_SR_UIF;        //скинути прапор
    }
}
```

2.1.3 Затримка на таймері.

Настроїмо таймер, подавши тактування й забезпечивши подачу на вхід рахункового регістру таймера мілісекундних імпульсів:

```
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN; //подати
//тактування на TIM2
TIM2->PSC= 8000-1; //настроїти дільник для
формування мілісекунд
TIM2->CR1 |= TIM_CR1_OPM; //режим одного імпульсу
TIM2->EGR = 1; //TIM_PSCReloadMode_Immediate;
```

У другому рядку показане налаштування вхідного дільника таймера. 8000 — це частота тактування таймера в khz. Вона буде залежати від налаштування загальної системи тактування.

Далі текст функції, що виконує затримку на задану кількість мільсекунд:

```
//argument   тривалість затримки від 1 до 65535
мільсекунд
void delay_ms(uint16_t value)
{
    TIM2->ARR                                =                value;
//завантажити значення затримки
    TIM2->CNT = 0;
    TIM2->CR1 = TIM_CR1_CEN;                //запустити
таймер
    while((TIM2->SR & TIM_SR_UIF)==0){} //дочекатися
кінця затримки
    TIM2->SR &= ~TIM_SR_UIF;                //скинути прапор
}
```

Фрагмент програми, що використовує цю функцію:

```
/*миготіння   світлодіодом   із   частотою   1Hz
*/
while(1)
{
    delay_ms(500);                          //затримка тривалістю
500мс
    GPIOB->ODR ^= 0x0002; //проінвертувати вихід
}
```

2.2 Порядок виконання роботи

2.2.1 Скласти програму згідно з індивідуальним завданням, використовуючи або обробку переривання, або затримку на таймері.

2.2.2 Відкопіювати програму із середовища й усунути помилки.

2.2.3 Отриманий hex файл завантажте в стенд.

2.3 Завдання до лабораторної роботи 2

Таблиця 2.1 - Таблиця завдань до лабораторної роботи №2

№	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 перемикати синій і червоний світлодіоди. У це ж час моргати зеленим із частотою 2 Гц.
3	Мигати синім світлодіодом за допомогою переривання від таймера із частотою 4Гц.
4	Перемикати жовтий і зелений світлодіоди за допомогою переривання від таймера із частотою 1Гц.
5	Мигати зеленим світлодіодом за допомогою затримки на таймері 6, 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 перемикати світлодіоди за годинниковою стрілкою за допомогою переривання від таймера із частотою 2Гц.
8	Поступове плавне гасіння всіх світлодіодів протягом 5 сек..

2.4 Контрольні питання

- 2.4.1 Перелічите таймери STM32F100.
- 2.4.2 Чим відрізняються таймери базові, загального призначення й розширені?
- 2.4.3 Регістри таймера.
- 2.4.4 Ініціалізація таймера засобами Standart Periph Lib.
- 2.4.5 Режим одновібратора.
- 2.4.6 Програмування обробки переривань від таймера.

3 ЛАБОРАТОРНА РОБОТА №3 ГЕНЕРАЦІЯ ШІМ.

Мета роботи: вивчити таймери процесора STM32, навчитися управляти режимами таймерів процесора STM32 і обробляти переривання від таймера.

3.1 Теоретичні відомості

Для того щоб ініціалізувати ШІМ на виході мікропроцесора необхідно з'ясувати яким каналом компаратора цей вивід може управлятися.

Таблиця 3.1 – Відповідність виводів і каналів таймерів

Вивід	Альтернативна функція	Remap
PB0	ADC1_IN8/TIM3_CH3	TIM1_CH2N
PB1	ADC1_IN9/TIM3_CH4	TIM1_CH3N
PB5	I2C1_SMBA / TIM16_BKIN	TIM3_CH2 /SPI1_MOSI
PB6	PB6 I2C1_SCL/ TIM4_CH1 TIM16_CH1N	USART1_TX

Потім необхідно ініціалізувати відповідний до виводу таймер на частоту кілька тисяч герц, як це показано в л.р.2. Слід урахувати, що значення поля TIM_TimebaseInitTypeDef.TIM_Period визначає максимальне число відповідне 100% шпаруватості.

Щоб таймер міг генерувати ШІМ, необхідно запрограмувати канал компаратора

```
TIM_OCInitTypeDef OCInit;
OCInit.TIM_OCMode= TIM_OCMode_PWM1;
```

```

OCInit.TIM_OutputState = TIM_OutputState_Enable;
OCInit.TIM_Pulse          =0;//початкове значення
шпаруватості
OCInit.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OC3Init(TIM3,&OCInit);//третій канал
TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Enable);
TIM_ARRPreloadConfig(TIM3, ENABLE);
//Запускаємо таймер!
TIM_Cmd(TIM3, ENABLE);

```

Шпаруватість ШІМ можна буде управляти через регістр каналу таймера, наприклад,

```
TIM3->CCR2 = значення
```

або функцією `void TIM_SetCompareномерканала ( TIM_TypeDef * Timx, uint16_t Compare1 ),` наприклад,

```
TIM_SetCompare3(TIM3,20);
```

Примітка:

Щоб на вивід міг проектуватися канал компаратора режим GPIO повинен бути встановлений як `GPIO_Mode_AF_PP`.

Примітка:

Щоб використовувати `Tim3ch2` на `PB5`, необхідно включити remapping портів:

```

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|
RCC_APB2Periph_AFIO , ENABLE);
GPIO_PinRemapConfig(GPIO_PartialRemap_TIM3,ENABLE)

```

3.2 Порядок виконання роботи

- 3.2.1 Скласти програму згідно з індивідуальним завданням.
- 3.2.2 Відкомпілювати програму із середовища й усунути помилки.
- 3.2.3 Отриманий hex файл завантажте в стенд.

3.3 Завдання до лабораторної роботи

Таблиця 3.2 - Таблиця завдань до лабораторної роботи №3

№	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 – за годинниковою.
4	Плавно запалювати червоний і синій світлодіоди в протифазі.
5	Плавно запалювати всі світлодіоди відразу.
6	Плавно запалювати жовтий світлодіод. Моргати червоним із частотою 2Гц.
7	Імітувати синім світлодіодом процес запалювання люмінесцентної лампи..
8	Поступове плавне гасіння всіх світлодіодів у плинні 5 сек., використовуючи апаратний ШИМ

3.4 Контрольні питання

1. Що таке ШИМ?
2. Режим ШИМ таймерів STM32F100.
3. Блок захвата/порівняння.
4. Режими генерації ШИМ.
5. Порядок програмування виводу для ШИМ генерації.

4 ЛАБОРАТОРНА РОБОТА №4 ПРОГРАМУВАННЯ ІНДИКАТОРУ WH1602.

Мета роботи: вивчити програмування LCD індикатору HD44780.

4.1 Теоретичні відомості

4.1.1 Вивід на LCD.

Для того щоб здійснювати вивід на індикатор, сумісний із драйвером Hitachi HD44780, потрібно підключити бібліотеку `hd44780.h` і проініціалізувати його викликом функції `lcd_init( )`. Ініціалізацію портів при цьому робити спеціально не треба – усе це включене в `lcd_init( )`. Призначення інших функцій дане нижче.

Таблиця 4.1 - Список функцій для роботи з LCD дисплеєм.

void <code>lcd_cmd(int8u_t data, int16u_t loop);</code>	Функція “ lcd_cmd(data, loop) ”, є низкоуровневою функцією, яка використовується для передачі, як команд, так і даних (перед відправленням даних слід установити високий рівень на виводі RS, а після відправлення скинути рівень виводу RS). Оскільки для передачі даних використовується функції lcd_putc(data) / lcd_prints(“string”) / lcd_itostr(number) / lcd_numtostr(number, ndigit) , те функцію “ lcd_cmd(data, loop) ” рекомендується використовувати тільки для передачі команд дисплею. Параметр “loop” призначений для вказівки додаткової тимчасової затримки після виконання функції “ lcd_cmd ”. У більшості випадків додаткова затримка не використовується (“loop =0”).
void <code>lcd_clrscr(void);</code>	Очищає всю DDRAM пам'ять, тобто всю видиму й сховану області дисплея.
void <code>lcd_return(void);</code>	Повертає курсор у перше комірку першого рядка дисплея (DDRAM пам'яті), тобто у

	верхній лівий кут дисплея.
void lcd_goto(int8u_t line, int8u_t address);	За допомогою функції lcd_goto(line, address) можна переміщати курсор між гніздами одному рядка, між гніздами різних рядків, а також між гніздами різних областей пам'яті (DDRAM або CGRAM пам'яті). У якості параметра “line” слід указати номер рядка (1, 2, ..) на яку збирається перемістити курсор (аналогічно осі ординат Y), а в якості параметра “address” слід указати номер комірки (0, 1, 2, ..., 15, ..) на цьому рядку (аналогічно осі абсцис X).
void lcd_prints(const char *p)	Функція “lcd_prints” призначена для виводу символьного рядка (масив типу char, що закінчується нулем) на дисплей (DDRAM пам'ять). Також, для більшої зручності роботи з дисплеєм, можна активувати форматований вивід символів (системные символи : ‘\n’, ‘\t’, ‘\r’). Для цього слід привласнити макросу USE_FORMATTED_OUTPUT значення “1”. Щоб відключити використання форматowanego виводу, слід привласнити цьому макросу значення “0”. Символ переходу на новий рядок (‘\n’) використовується тільки для переходу з рядка #1 на рядок #2! Символ повернення каретки (‘\r’) аналогічний функції “lcd_return()” і повертає курсор у перше гніздо першого рядка дисплея (DDRAM пам'яті), тобто у верхній лівий кут дисплея! Символ табуляції (‘\t’) має те ж призначення що й системний символ ‘t’.

	Для керування кроком табуляції використовується макрос TAB_SPACE , значення якого можуть бути : 1, 2, 4 або 8.
void lcd_putc(int8u_t data);	Функція “lcd_putc” використовується для виводу символів на дисплей (запис DDRAM пам'яті).
void lcd_load(int8u_t *vector, int8u_t position);	Функція “lcd_load” використовується для завантаження користувацького символу в CGRAM пам'ять. Розмір символу залежить від конфігурації дисплея (макрос DEFAULT_DISPLAY_CONFIG), а також від значення макросу DRAW_CHAR_SIZE (8 або 11), які за замовчуванням відповідають символам 5x8 пікселів.
void lcd_drawchar(int8u_t *vector, int8u_t position, int8u_t line, int8u_t address);	Функція “lcd_drawchar” як і функція “lcd_load” записує користувацький символ (параметр vector) у деяку область CGRAM пам'ять (параметр position), але у відмінності від “lcd_load” відразу виводить цей символ на дисплей у зазначений рядок і комірку (параметри “line” і “address”).
void lcd_backspace(void);	Функція “lcd_backspace()” використовується для очищення комірки DDRAM пам'яті ліворуч від курсору.
void lcd_scroll(int8u_t direction);	Кожний виклик функції “lcd_scroll” зміщує видиму частину дисплея (DDRAM пам'яті) уліво/вправо (за допомогою макросів LEFT і RIGHT) на одну позицію. За допомогою даної функції можна одержати ефект “ рядка, що біжить, ”.
void cursor_shift(int8u_t direction);	Кожний виклик функції “cursor_shift” зміщує курсор уліво/вправо (за допомогою макросів LEFT і RIGHT) на одну позицію. Цю функцію можна використовувати для редагування в режимі реального часу

	помилок при введенні значень із клавіатури (наприклад кодовий замок).
void lcd_itostr(int32s_t value);	Функція “lcd_itostr” виводить цілі числа (у діапазоні +/- 2147483647) на дисплей.
void lcd_numtostr(int16u_t value, int8u_t ndigit);	Функція “lcd_numtostr” використовується для виводу декількох розрядів цілого числа (у діапазоні : 0 .. 65535) на дисплей. У випадку якщо у виведеного числа менше розрядів чому зазначене в параметрі ndigit, те розряди, що тоді залишилися, заповнюються нулями, тобто “виведене число” займає фіксоване число гнізд дисплея, що в деяких ситуаціях досить корисна властивість (див. нижче відсоток заповнення).

4.1.2 Програмування натискань на кнопки.

У даному стенді кнопки приєднані на PORTB.8 і PORTB.9. При натисканні кнопки заземлюють відповідний вивід, тому їх програмують як ввід з підтяжкою нагору - тобто при не натиснутій кнопці буде читатися "1":

```
GPIO_InitStructure.GPIO_Pin=GPIO_Pin_8|GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

Властиво читання стану входу здійснюється функцією

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx,
uint16_t GPIO_Pin )
```

Наприклад, GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_8) .

4.2 Порядок виконання роботи

4.2.1 Скласти програму згідно з індивідуальним завданням.

4.2.2 Відкомпілювати програму із середовища й усунути помилки.

4.2.3 Отриманий hex файл завантажте в стенд.

4.3 Завдання до лабораторної роботи

Таблиця 4.2 - Таблиця завдань до лабораторної роботи №4

№	Текст індивідуального завдання
1	Вивести число 100. Через кожні 2 сек збільшувати його.
2	Вивести число 30. Через кожні 1 сек зменшувати його. Коли досягнеться нуля – зупинити рахунок і почати моргати всіма світлодіодами із частотою 2Гц.
3	По скиданню запускати секундомір. (Щосекунди оновляти екран, виводячи на нього минуле зі старту час у секундах)
4	Виводити в першому рядку натуральне число. Щосекунди збільшувати його й виводити в другому рядку його квадрат.
5	Вивести час у форматі годинник:хвилини:секунди. Змінювати його на зразок електронних годин.
6	З інтервалом 0.5 сек виводити через кому квадрати натуральних чисел від 1 до 10.
7	Виводити в першому рядку натуральне число. Щосекунди збільшувати його й виводити в другому рядку його куб.
8	Вивести в середині екрана "Hello!". Зміщати його вправо-уліво зі швидкістю 2 символу в секунду.
9	Вивести число 255. Через кожні 3 сек зменшувати його..

4.4 Контрольні питання

- 4.4.1 Схема підключення індикатору й призначення сигналів.
- 4.4.2 Архітектура бібліотеки hd44780.h і призначення її функцій.
- 4.4.3 Опитування стану виводів і натискань на кнопки.

5 ЛАБОРАТОРНА РОБОТА №5 ОБРОБКА РЕАКЦІЙ НА ЗОВНІШНІ ПОДІЇ.

Мета роботи: вивчити програмування реакції на зовнішні події.

5.1 Теоретичні відомості

5.1.1 Опитування стану виводів (кнопки).

У даному стенді кнопки приєднані на PORTB.8 і PORTB.9. При натисканні кнопки заземлюють відповідний вивід, тому їх програмують як уведення з підтяжкою нагору - тобто при не натиснутій кнопці буде читатися "1":

```
GPIO_InitStructure.GPIO_Pin=GPIO_Pin_8|GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

Читання стану входу здійснюється функцією

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef *GPIOx,
uint16_t GPIO_Pin )
```

Наприклад:

```
GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_8).
```

5.2. Використання переривань.

5.2.1 Ініціалізуйте переривання від зовнішньої лінії згідно наступного прикладу

```
__enable_irq();//дозволяємо обробку переривань
GPIO_InitTypeDef GPIO_InitStructure;
EXTI_InitTypeDef EXTI_InitStructure;
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
NVIC_InitTypeDef NVIC_InitStructure;
    // Конфігурація лінії входу
```

```

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_AFIO, ENABLE);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOB, &GPIO_InitStructure);
/* Connect Button EXTI Line to Button GPIO Pin */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOB,
GPIO_PinSource8);

/* Configure Button EXTI line */
EXTI_InitStructure.EXTI_Line = EXTI_Line8;
EXTI_InitStructure.EXTI_Mode= EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger=EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);
/* Enable and set Button EXTI Interrupt to the lowest priority */
NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

```

5.2.2 Перехопимо оброблювач переривань

```

void EXTI9_5_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line8)==SET) перевірка прапора даної лінії
    {
        // дія
        //*****
EXTI_ClearITPendingBit(EXTI_Line8); // Скидання прапора переривання
    }
}

```

5.3 Порядок виконання роботи

- 5.3.1 Скласти програму згідно з індивідуальним завданням.
 5.3.2 Відкомпілювати програму із середовища й усунути помилки.
 5.3.3 Отриманий hex файл завантажте в стенд.

5.4 Завдання до лабораторної роботи

Таблиця 5.1 - Таблиця завдань до лабораторної роботи №4

№	Текст індивідуального завдання
1	Вивести число 100. Натискаючи S1, S2 збільшувати/зменшувати його, використовуючи механізм переривань.
2	Вивести поточну дату у форматі число/№місяця/рік. Редагувати, натискаючи S1, S2 його на зразок електронних годин, використовуючи механізм переривань.
3	Натискаючи S1 запускати й зупиняти секундомір. Натискаючи S2 – скидати. Використовуйте механізм переривань
4	Редагувати в першому рядку за допомогою S1 натуральне число. При натисканні на S2 вивести в другому рядку його квадрат.
5	Вивести час у форматі годинник:хвилини. Редагувати, натискаючи S1, S2 його на зразок електронних годин, використовуючи механізм переривань.
6	Редагувати в першому рядку за допомогою S1, S2 натуральне число, використовуючи механізм переривань. При цьому в другому рядку виводити його куб.
7	Вивести в середині екрана "Hello!". Зміщати його вправо-уліво, натискаючи S1, S2.

5.4 Контрольні питання

5.4.1 Переривання й події.

5.4.2 Настроювання подій від зовнішніх сигналів.

5.4.3 Перехоплення переривань від зовнішніх сигналів і аналіз їх джерела.

6 ЛАБОРАТОРНА РОБОТА №6 ВИКОРИСТАННЯ семіхостинга ЧЕРЕЗ ПОСЛІДОВНИЙ ПОРТ.

Мета роботи: навчитися виводити отладочную інформацію методом семіхостинга, навчитися перенаправляти стандартні потоки Із через послідовний порт.

6.1 Теоретичні відомості

Семіхостинг – це отладочный приймання, при якому оператори стандартного введення – виводу перенаправляються через порт на хост комп'ютер, де можна взаємодіяти із програмою на мікроконтролері через віконце термінала. Отже, за допомогою стандартних `scanf-printf` можна добитися взаємодії з консоллю при мінімальних апаратних витратах. Багато IDE передбачають вбудовані засоби для семіхостинга, але можна скористатися програмами-терміналами від сторонніх виробників.

6.2 Порядок виконання роботи

6.2.1 Створіть новий проект у середовищі Cooox Coide. У репозиторії відзначте Semihosting, Retarget printf, CMSIS boot, USART.

6.2.2 Додайте новий файл **Serial.c** у проект:

```
#include "stm32f10x.h"                                     /*
STM32F10x.h definitions                                  */

/*-----
  Initialize UART pins, Baudrate
  *-----*/
void SER_init (void) {
  int i;
#define USARTx  USART1
  /* Configure USART1 for 115200 baud                      */
  RCC->APB2ENR |= (1UL << 2); /*Enable GPIOA clock
  */
  GPIOA->CRH    &= 0xFFFFF00F;
```

```

GPIOA->CRH      |= 0x000004B0;                /* PA9 =
USART1_TX, PAa0 = USART1_RX */
RCC->APB2ENR    |= ( 1UL << 14/* Enable USART#1
clock*/
USARTx->BRR= 0x00D0; /* Configure 115200 baud @
24MHz*/
USARTx->CR3      = 0x0000;
/*          8 bit, 1 stop bit,          */
USARTx->CR2      = 0x0000;                  /*
no parity          */
    for (i = 0; i < 0x1000; i++) __NOP(); /* avoid
unwanted output */
    USARTx->CR1 = 0x200C;
}
/*-----
    Write character to Serial Port
-----*/
int sendchar (int c) {
    while (!(USARTx->SR & 0x0080));
    USARTx->DR = (c & 0xFF);
    return (c);
}
/*-----
    Read character from Serial Port (blocking read)
-----*/
int getkey (void) {
    while (!(USARTx->SR & 0x0020));
    return (USARTx->DR);
}

```

6.2.3 Створіть хедер **Serial.h** до нього

```

#ifndef _serialh
#define _serialh
void SER_init (void);
int sendchar (int c);
int getkey (void);
#endif

```

6.2.4 Відкрийте файл `prinfo.c`. Знайдіть і відредагуйте наступні функції таким манером:

```
signed int puts(const char *pStr)
{
    signed int i = fputs(pStr, stdout);
    fputc('\n', stdout);
    return i+1;
}

void PrintChar(char c)
{
    sendchar ( c ) ;
}
```

6.2.5 Напишіть основну програму, вона може мати вигляд:

```
#include "stm32f10x.h"
#include <stdio.h>
#include "serial.h"
// #include "system_stm32f10x.h"
int main(void)
{
    SystemInit();
    SER_init(); // ініціалізація послідовного порту
    printf("\n");
    printf("Hello World!\n");

    while(1)
    {
    }
}
```

6.2.6 Складіть програму згідно з варіантом, відкомпілюйте її.

6.2.7 Відкрийте вікно термінала й настройте на послідовний порт і виставте його параметри.

6.2.8 Запустите на стенді програму й перевірте вивід на термінал.

6.3 Завдання до лабораторної роботи

Таблиця 6.1 - Таблиця завдань до лабораторної роботи №6

№	Текст індивідуального завдання
1	Вивести числа від 0 до 100 і їх квадрати у дві колонки на терміналі.
2	Вивести числа від 30 до 1 через кожні 1 сек. Коли досягнеться нуля – почати моргати всіма світлодіодами із частотою 2Гц.
3	По натисканню на кнопки виводите на термінал повідомлення з її назвою.
4	Виводити в першому рядку натуральне число. Щосекунди збільшувати його й виводити в другому рядку його квадрат.
5	Роздрукувати перші n простих чисел (p - просте число, якщо $p \geq 2$ і ділиться тільки на 1 і на себе).
6	З інтервалом 0.5 сек виводити через кому квадрати натуральних чисел від 1 до 10.
7	Щосекунди збільшувати й виводити натуральне число, а також його куб.
8	Дане натуральне число n . Знайти значення числа, отриманого в такий спосіб: із запису числа n викинути цифри 0 і 5, залишивши колишнім порядок інших цифр. Виводити на термінал n і результат перетворення, починаючи з 10, з інтервалом 1 сек.
9	Підрахувати кількість парних цифр у десятковому записі цілого ненегативного числа n для n від 10 до 100.

6.4 Контрольні питання

- 6.4.1. Що таке семіхостинг?
- 6.4.2. Настроювання послідовного порту.
- 6.4.3. Як задається швидкість послідовного порту?
- 6.4.4. Як перенаправляти `printf` у послідовний порт.

7 ЛАБОРАТОРНА РОБОТА №7 ВИВЧЕННЯ ГОДИН РЕАЛЬНОГО ЧАСУ МІКРОПРОЦЕСОРА STM32F100

Мета роботи: вивчити основи роботи блоку RTC.

7.1 Теоретичні відомості

Функції, що обслуговують годинник реального часу (RTC).

Функція	Опис
RTC_ITConfig	Включає/виключає переривання від RTC.
RTC_EnterConfigMode	Входить у режим конфігурації RTC.
RTC_ExitConfigMode	Виходить із режиму конфігурації RTC
RTC_GetCounter	Отримує лічильник RTC.
RTC_SetCounter	Встановлює лічильник RTC.
RTC_SetPrescaler	Встановлює дільник RTC
RTC_SetAlarm	Встановлює будильник RTC
RTC_GetDivider	Отримує дільник RTC.
RTC_WaitForLastTask	Чекає закінчення операції модифікації регістрів RTC
RTC_WaitForSynchro	Чекає поки RTC не буде синхронізований з RTC APB clock.
RTC_GetFlagStatus	Перевіряє прапор RTC.
RTC_ClearFlag	Чистить прапор RTC.
RTC_GetITStatus	Перевіряє прапор переривання RTC.
RTC_ClearITPendingBit	Чистить прапор переривання RTC.

7.2 Порядок виконання роботи

7.2.1. Переконаєтеся що або у вашій програмі, або в `stm32f10x_conf.h` є раскоментовани рядкі:

```
#include "stm32f10x_rtc.h"
#include "stm32f10x_pwr.h"
```

7.2.2. Подати тактову частоту на RTC і вузол backup

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR |
    RCC_APB1Periph_BKP, ENABLE);
```

7.2.3. Задати конфігурація RTC

```
PWR_BackupAccessCmd(ENABLE);
//RCC_Lseconfig(RCC_LSE_ON);
RCC_LSICmd(ENABLE);
while(RCC_GetFlagStatus(RCC_FLAG_LSIRDY) == RESET);
RCC_RTCCLKConfig(RCC_RTCCLKSource_LSI);
RCC_RTCCLKCmd(ENABLE);
RTC_WaitForSynchro();
RTC_WaitForLastTask();
RTC_ITConfig(RTC_IT_SEC, ENABLE);
RTC_WaitForLastTask();
RTC_SetPrescaler(32767);
/*    RTC    period    =    RTCCLK/RTC_PR    =    (32.768
KHz)/(32767+1) */
RTC_WaitForLastTask();
```

7.2.4. Настроюємо переривання від RTC

```
NVIC_InitTypeDef NVIC_InitStructure;
/* Configure one bit for preemption priority */
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
/* Enable the RTC Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = RTC_IRQn;
```

```

NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

```

7.2.5. Перехоплюємо переривання

```

void RTC_IRQHandler(void)
{
    if (RTC_GetITStatus(RTC_IT_SEC) != RESET)
    {
        RTC_ClearITPendingBit(RTC_IT_SEC);
        RTC_WaitForLastTask();
    }
}

```

7.2.6 Скласти програму згідно з індивідуальним завданням.

7.2.7 Відкомпілювати програму із середовища й усунути помилки.

7.2.8 Отриманий hex файл завантажте в стенд

Таблиця 7.1 - Таблиця завдань до лабораторної роботи №7

№	Текст індивідуального завдання
1	Використовуючи RTC організувати перемикання червоного, жовтого, зеленого світлодіодів як у світлофорі з інтервалом 1с.
2	Запрограмувати виникнення події RTC Alarm через 10 з послуги скидання, по якому запалити всі світлодіоди.
3	Мигати синім світлодіодом за допомогою переривання від RTC із частотою 1/4Гц.
4	Перемикаєти жовтий і зелений світлодіоди за допомогою переривання від RTC з інтервалом 1с.
5	Мигати зеленим світлодіодом за допомогою переривання від RTC, кожні 10 секунд запалювати червоний світлодіод і виводити вміст лічильника RTC .
6	Мигати по три рази червоним, потім синім, потім зеленим світлодіодом за допомогою переривання від RTC із частотою 1Гц.
7	По черзі перемикаєти світлодіоди за годинниковою стрілкою за допомогою переривання від таймера з інтервалом 1с.
8	Виводити вміст лічильника RTC щосекунди.

7.4 Контрольні питання

- 7.4.1 Регістри блоку годиннику реального часу.
- 7.4.2 Програмування RTC.
- 7.4.3 Програмування Alarm-Ов від RTC.
- 7.4.4 Перехоплення переривань від подій RTC.

8 ЛАБОРАТОРНА РОБОТА №8 ВИЗНАЧЕННЯ ПРОДУКТИВНОСТІ DMA.

Мета роботи: вивчити контролер DMA процесора STM32f100.

8.1 Теоретичні відомості

Настроювання DMA на копіювання блоку з однієї адреси пам'яті на іншій:

```
DMA_InitTypeDef DMA_InitStructure;
//reset DMA1 channel to default values;
DMA_DeInit(DMA1_Channel1); //channel will be used
for memory to memory transfer
DMA_InitStructure.DMA_M2M=DMA_M2M_Enable;//setting
normal mode (non circular)
DMA_InitStructure.DMA_Mode=DMA_Mode_Normal;
//medium priority
DMA_InitStructure.DMA_Priority=DMA_Priority_Medium
;
//source and destination data size word=32bit
DMA_InitStructure.DMA_PeripheralDataSize=DMA_Perip
heralDataSize_Word;
DMA_InitStructure.DMA_MemoryDataSize=DMA_MemoryDat
aSize_Word; //automatic memory increment enable.
Destination and source
DMA_InitStructure.DMA_MemoryInc=DMA_MemoryInc_Enab
le;
DMA_InitStructure.DMA_PeripheralInc=DMA_Peripheral
Inc_Enable;
//Location assigned to peripheral register will be
source
DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
//chunk of data to be transfered
DMA_InitStructure.DMA_BufferSize = розмір;
//source and destination start addresses
DMA_InitStructure.DMA_PeripheralBaseAddr=(uint32_t
) адреса початку джерела;
DMA_InitStructure.DMA_MemoryBaseAddr=(uint32_t)ціл
ьова адреса;
```

```

DMA_Init(DMA1_Channel1, &DMA_InitStructure);
//send values to DMA registers
// Enable DMA1 Channel Transfer Complete interrupt
DMA_ITConfig(DMA1_Channel1, DMA_IT_TC, ENABLE);
NVIC_InitTypeDef NVIC_InitStructure; /*Enable DMA1
channel IRQ Channel */
NVIC_InitStructure.NVIC_IRQChannel=DMA1_Channel1_I
RQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriori
ty = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure); //LED on before
transfer LEDToggle(LEDG);

//Enable DMA1 Channel transfer
DMA_Cmd(DMA1_Channel1, ENABLE);
while(status==0) {};
DMA_Cmd(DMA1_Channel1, DISABLE);
status=0;

```

Оброблювач переривання від DMA:

```

void DMA1_Channel1_IRQHandler(void) {
//Test on DMA1 Channel1 Transfer Complete
    if(DMA_GetITStatus(DMA1_IT_TC1))
    {
        status=1;
        //Clear DMA1 Channel1 Half Transfer, Transfer
        DMA_ClearITPendingBit(DMA1_IT_GL1);
    }
}

```

8.2 Хід роботи.

- 8.2.1 Користуюсь теоретичними відомостями написати функцію копіювання фрагмента пам'яті в іншій за допомогою DMA.
- 8.2.2 Завести в програмі два глобальні масиви


```
uint32_t source[ARRAYSIZE];
uint32_t destination[ARRAYSIZE];
```

 і проініціалізувати source початковими значеннями.

У файлі `cmsis_boot/startup/startup_stm32f10x_md_vl.c` зменште розмір стека

```
#define STACK_SIZE          0x00000050          /*!< Stack
size      (in      Words)                      */
```

8.2.3 Визначити тривалість N кратного копіювання масиву `source` в `destination` як за допомогою DMA, так і за допомогою циклів, користуючись даними з варіанта завдання.

8.2.4 Програмно расчитать і вивести продуктивність на дисплей стеда.

8.2.5 По закінченню кожного тесту запалювати світлодіод.

8.3 Завдання до лабораторної роботи

Таблиця 8.1 - Таблиця завдань до лабораторної роботи №8

№	Текст індивідуального завдання
1	ARRAYSIZE=800; N=10000;
2	ARRAYSIZE=900; N=2000;
3	ARRAYSIZE=500; N=10000;
4	ARRAYSIZE=700; N=10000;
5	ARRAYSIZE=950; N=5000;
6	ARRAYSIZE=650; N=15000;
7	ARRAYSIZE=150; N=25000;
8	ARRAYSIZE=800; N=2000;

8.4 Контрольні питання

8.4.1. Регістри блоку DMA.

8.4.2. Програмування DMA.

8.4.3. Перехоплення переривань від DMA.

ЛИТЕРАТУРА

1. Белов А.В. Самоучитель по микропроцессорной технике. – СПб.: Наука и Техника, 2003. – 224 с.
2. Коган Б.М. Электронные вычислительные машины и системы. – М.: Высш. шк., 1991. – 590 с.
3. Касаткин А.И., Вальвачев А.Н. Профессиональное программирование на языке Си: От Turbo C Borland C++. - М.: Высш. шк., 1992. – 240 с.
4. Майоров С.А., Кириллов В.В., Приблуда А.А. Введение в микро-ЭВМ. - Л.: "Машиностроение", 1988. – 304 с.
5. Нестеров П.В. и др. Микропроцессоры (в 3 кн.) т.1 под ред. М.Н. Преснухина. - М.: "Высшая школа", 1986. – 495 с.
6. Новиков М.Н. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. - М.: Мир. – 2001. – 379 с.
7. Рудометов Г.А. Материнские платы и чипсеты. 2-е издание. - СПб.: Питер Пресс, - 2001. – 352 с.
8. Соловьев В.В. Проектирование цифровых систем на основе программируемых логических интегральных схем. - М.: Горячая Линия – Телеком., 2001. – 636 с.
9. Фромберг Г. Конструкции на элементах цифровой техники. - М.: Горячая Линия – Телеком., 2002. – 264 с.
10. Фрунзе М.А. Микроконтроллеры? Это же просто! Том 1 М.: Додэка – XXI, - 2002 – 336 с.

