Міністерство освіти і науки України

Національний технічний університет України

«Київський Політехнічний Інститут імені Ігоря Сікорського»

Кафедра конструювання електронно-обчислювальної апаратури

Звіт

З лабораторної роботи №3

по курсу «Обчислювальні та МП засоби в РЕА-2»

Виконав:

cтудент III курсу

групи ДК-81

Шунь П.О.

Перевірив:

доц. Корнєв В.П.

cт. в. Бондаренко Н.О.

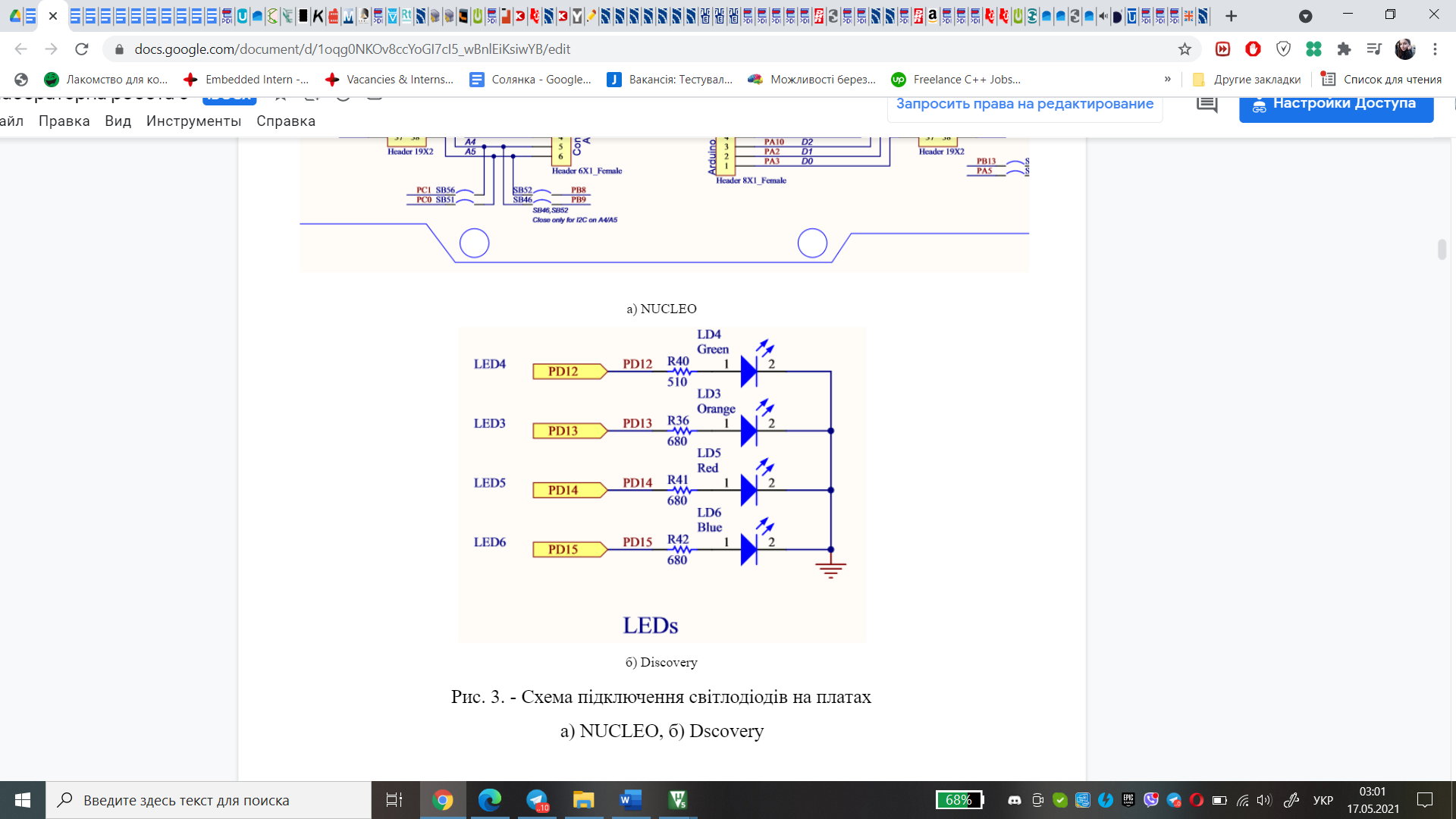
Київ – 2021

**Тема:** *Пряме програмування портів GPIO. Проект на мові асемблера і С.*

***Завдання:*** почергове увімкнення світлодіодів та зміна послідовності увімкнення натисненням кнопки – увімкнення почергового мерехтіння світлодіодів проти годинникової стрілки і зміна напрямку почергового мерехтіння по натисненню кнопки.

***Теоретична частина***

Щоб увімкнути світлодіод, який розташований на платі STM32F401 NUCLEO, або STM32F407 Discovery (далі просто NUCLEO i Discovery) необхідно дно відкрити інструкцію користувача (user manual UM1724 ( NUCLEO ) / UM1472 (Discovery) ) . Там написано, які піни за які компоненти на платі відповідають ( схема для світлодіодів плати Discovery приведена нижче ) . Керування відбувається за допомогою периферійних блоків GPIO (*General Purpose Input/Outputs*), які прийнято називати *портами вводу/виводу*. Вони позначаються англійськими буквами A,B,C, …,J,K . Кожен порт містить 16 виводів (pins - пінів). Отже, програмам користувача доступні 4 світлодіоди LD3 – LD6, світлодіодами на платі Discovery керують піни 12-15 порту «D» («PD12-15»), і щоб його увімкнути необхідно подати на його анод високий рівень.

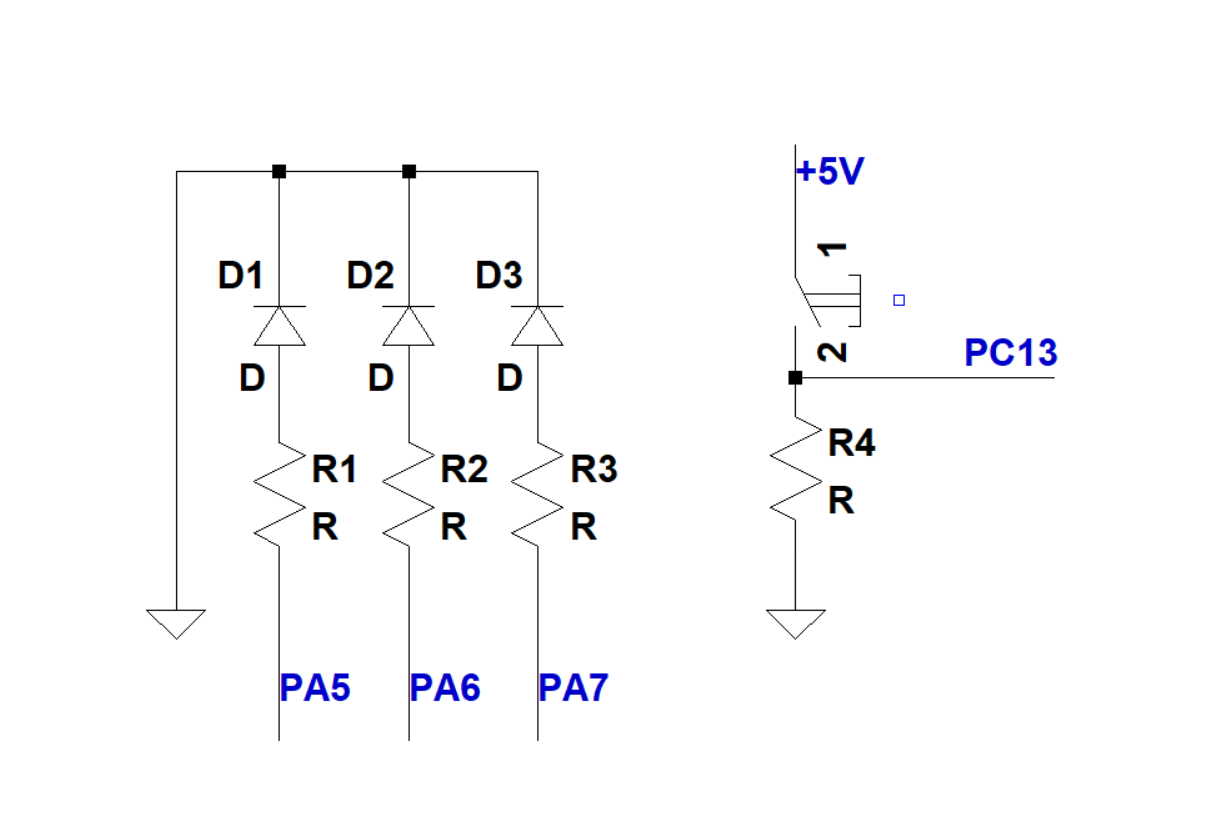


Для зниження електроспоживання мікроконтролеру (MCU), після його скидання (RESET), практично всі блоки периферійних пристроїв і утому числі всі порти вимкнуті. Увімкнення або вимкнення їх відбувається подачею або припиненням подачі на них тактуючого сигналу. Керування тактуючим сигналом відбувається за допомогою регістрів «RCC XXX peripheral clock enable» з загальної групи керування скиданням та синхронізацією  RCC (*Reset and clock control*) у якої знаходяться всі регістри керування налаштуванням генераторів, блоків ФАПЧ та шин. Замість «XXX» в позначення регістру має бути назва шини «AHB1», «AHB2» або «APB1», яка забезпечує доступ до певного ресурсу MCU ( інформація про відповідні за тактування порти є в «reference manual RM0368/RM0090» ) .

Тепер, коли порт «А» / «D» тактується, нам необхідно задати режим його роботи. Це робиться за допомогою регістру «GPIO port mode register» (**GPIOx\_MODER**) . Його зміщення становить «0х00», а щоб настроїти пін 5 / 12 на «вивід» даних («output mode») з відповідного порту **GPIOx** (**х=A,B,C,D,…,H/…,K**) необхідно записати в біти «11,10» / «25,24» значення «**01**».  Для того ж, щоб встановити на виході високий або низький рівень, треба записати значення «**1»** відповідно в біти «5»/«12» або «21»/«28» регістру «GPIO port bit set/reset register» (**GPIOx\_BSRR**).

***Хід роботи***

Схема, використана у лабораторній:



Код програми міститься в 2 файлах.

mystartup.s

Stack\_Size EQU 0x00000400

; stack area

AREA STACK, NOINIT, READWRITE, ALIGN = 3

Stack\_Mem SPACE Stack\_Size ; reserve stack memory

\_\_initial\_sp ; initial stack pointer value

; interrupt vector area

AREA RESET, DATA, READONLY

EXPORT \_\_Vectors

\_\_Vectors ; table of vectors

DCD \_\_initial\_sp

DCD Reset\_Handler

\_\_Vectors\_End

; code area

AREA |.text|, CODE, READONLY

EXPORT Reset\_Handler

Reset\_Handler PROC

B main

ENDP

EXTERN main\_loop

main PROC ; main progam entry point

BL main\_loop

B main ; endless loop

ENDP

ALIGN

END

З цього файлу ми викликаємо основний код програми, написаний на С  
Також ініціалізуємо ResetHandler.

main.c

typedef unsigned long uint32;

typedef unsigned long long uint64;

// AHB1 bus enable register

#define RCC\_AHB1\_ENABLE\_REG (uint32\*)0x40023830

// mode register

#define GPIOA\_MODER (uint32\*)0x40020000

#define GPIOC\_MODER (uint32\*)0x40020800

//#define GPIOA\_IDR (uint32\*)0x40020010

#define GPIOA\_IDR (uint32\*)0x40020810

// bit set/reset register (atomic)

#define GPIOA\_BSRR (uint32\*)0x40020018

#define RCC\_AHB1\_GPIOA\_CLOCK 0

#define GPIOA\_LED\_PIN\_BLUE 5

#define GPIOA\_LED\_PIN\_RED 6

#define GPIOA\_LED\_PIN\_YELLOW 7

#define GPIOA\_PIN\_BUTTON 13

void delay() {

volatile uint32 i = 500000; // Затримка приблизно 0.5 с

while ( i!=0) i--;

}

void wait\_5ms () {

volatile uint32 j = 1250; // для затримки ~5 мс.

while ( j!=0) j--;

}

// Організація кнопки з захистом від дребезгу

volatile uint32 BUTTON (char state){

volatile uint32 k = 0;

volatile uint32 counter = 0;

if ((\*GPIOA\_IDR & 0x2000) == 0x0){

counter = 0;

state = 0;

} else if ((\*GPIOA\_IDR & 0x2000) == 0x1)

{

for (k = 0; k <10; k++)

{

wait\_5ms();

if ((\*GPIOA\_IDR & 0x2000) == 0x0){

counter = 0;

} else {

counter += 1;

}

if (counter >= 4){

state = 1;

}

}

}

return state;

}

void main\_loop()

{

volatile uint32 state = 0;

char led\_on = 0;

\*RCC\_AHB1\_ENABLE\_REG |= (1 << RCC\_AHB1\_GPIOA\_CLOCK);

\*GPIOA\_MODER &= ~(1 << (GPIOA\_LED\_PIN\_BLUE \* 2 + 1));

\*GPIOA\_MODER |= (1 << (GPIOA\_LED\_PIN\_BLUE \* 2));

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_BLUE);

\*GPIOA\_MODER &= ~(1 << (GPIOA\_LED\_PIN\_RED \* 2 + 1));

\*GPIOA\_MODER |= (1 << (GPIOA\_LED\_PIN\_RED \* 2));

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_RED);

\*GPIOA\_MODER &= ~(1 << (GPIOA\_LED\_PIN\_YELLOW \* 2 + 1));

\*GPIOA\_MODER |= (1 << (GPIOA\_LED\_PIN\_YELLOW \* 2));

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_YELLOW);

\*GPIOC\_MODER &= ~(0 << (GPIOA\_PIN\_BUTTON \* 2 + 1));

\*GPIOC\_MODER |= (0 << (GPIOA\_PIN\_BUTTON \* 2));

while (1) {

state = BUTTON(state);

if (state > 0) {

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_BLUE);

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_RED);

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_YELLOW);

}

if (led\_on == 0) {

\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_BLUE);

//\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_RED);

//\*GPIOA\_BSRR |= (1 << GPIOA\_LED\_PIN\_YELLOW);

} else {

\*GPIOA\_BSRR |= (1 << (GPIOA\_LED\_PIN\_BLUE + 16));

\*GPIOA\_BSRR |= (1 << (GPIOA\_LED\_PIN\_RED + 16));

\*GPIOA\_BSRR |= (1 << (GPIOA\_LED\_PIN\_YELLOW + 16));

}

delay();

if (led\_on == 0)

{

led\_on = 1;

} else led\_on = 0;

}

}

***Висновок .***

У ході виконання лабораторної роботи ми розглянули структуру і технологію створення проекту, в якому основна програма, написана на мові С, викликає функції (підпрограми), написані на асемблері. Було виконане початкове завдання з реалізації почергового перемикання світлодіодів.