**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СИКОРСЬКОГО»**

**КАФЕДРА КОНСТРУЮВАННЯ ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНОЇ АПАРАТУРИ**

**ЗВІТ**

з лабораторної робіти № 4

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

Виконав:

студент гр. ДК-82

Дмитрук О.О.

Перевірив:

ст. викладач

Бондаренко Н.О.

Київ – 2021

**Завдання**

1. Реалізувати виведення однорозрядного числа на семисигментному індикаторі з програмним перетворенням числа в код індикатора. Зменшення/збільшення числа натисканням кнопок. Використовувати бібліотеку CMSIS

**Теоретична частина**

Розглянемо теорію, необхідну для виконання лабораторної роботи.

Щоб працювати з портами вводу/виводу (GPIO) мікроконтроллеру STM32 потрібно виконати певні дії, а саме:

1. Ввімкнути тактування портів
2. Налаштувати порти Input / Output
3. Записувати в регістр ODR якісь значення (У випадку Output)  
   Зчитувати з регістру IDR певні записані в нього значення (Налашт. Input)

В кожного периферійного блоку є своя базова адреса, по якій можна звертатись до певних регістрів налаштування. На рис.1. зображена карта пам’яті мікроконтр. STM32F4, з яким ми будемо працювати.

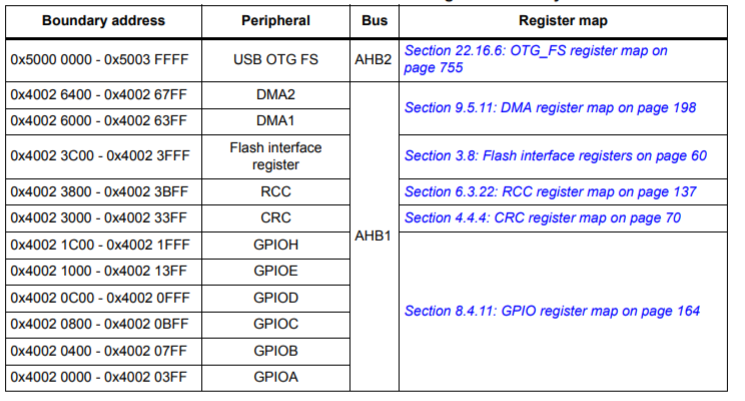
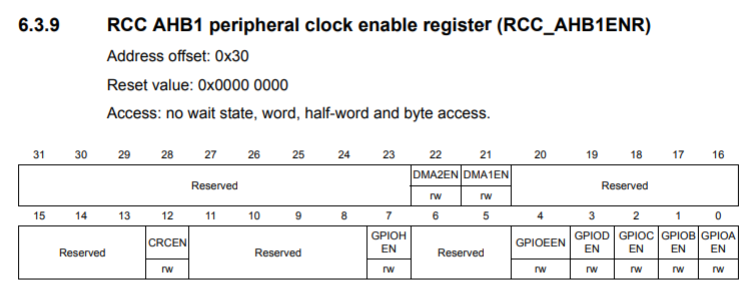


Рис.1. Карта пам’яті SET32F4

Всю інформацію про регістри налаштування даного мікроконтр. можна найти в reference manual-і.

Щоб ввімкнути тактування портів, необхідно звернутись до блоку RCC (Reset Clock Control) що має базову адресу 0x40023800

Тепер потрібно звернутись до регістру RCC\_AHB1ENR який відповідає за ввімкнення тактування GPIOx, який має зміщення відносно базової адреси 0x30.



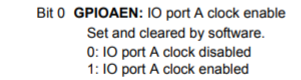


Рис.2. Регістр тактування GPIOx

Як бачимо, щоб ввімкнути тактування певного GPIO, необхідно встановити 1 в певному зарезервованому біті. Це можна легко зробити звернувшись по базовій адресі + зміщення, і виконати “побітове або” з маскою (1 << BIT) де BIT – номер біту певного порту.

Тепер розберемось з налаштуванням GPIOx на ввід/вивід. Для цього їснує регістр GPIOx\_MODER (тобто в кожного порту є такий регістр)

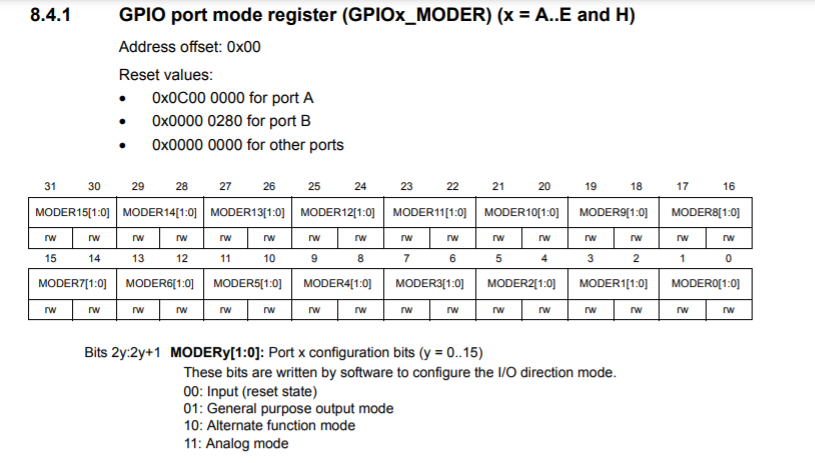


Рис.3. Регістр налаштування GPIOx

Записавши в даний регістр в певні біти певні данні, ми зможемо налаштувати певні піни на наступні режими:

1. 00 – Режим читання – тобто пін буде зчитувати сигнал і записувати його в регістр IDR по кожному активному фронту тактового сигналу
2. 01 – Режим запису – пін буде генерувати цифрову 1 або 0 (3.3в або 0в)
3. 10 – Режим, в якому пін буде налаштований під певні периферії, такі як SPI, I2C і т.п.
4. 11 – Аналоговий режим – пін налаштований на генер. ШИМ сигналу і т.п.

Розберемось з регістрами GPIOx\_ODR та GPIO\_IDR

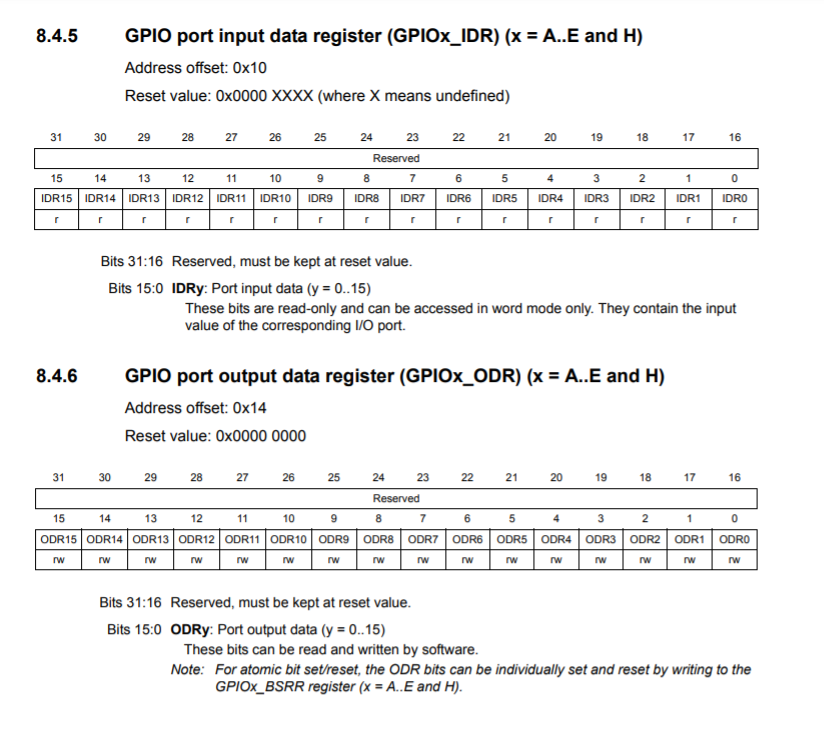


Рис.4. Регістри данних IDR та ODR

За допомогою регістру ODR ми можемо генерувати сигнали на піни GPIOx. Також з даного регістру можна зчитувати.

Регістр IDR працює лише на считування, і необхідний щоб в нього по кожному актив. фронту сигналу синхр. записувались дані з певних пінів GPIO. Можна ідентифікувати нажаття кнопки, що спричинить запис в певний біт регістру 0 або 1, в залежності від того як підключена кнопка.

Також існує досить багато регістрів, такі як BSSR, за допомогою якого також можна встановлювати та скидати певні біти регістру ODR (перші 16 біт відповідають за встановлення, старші 16 за скидання), GPIOx\_PUPDR, за допомогою якого можна підтянути пін до землі або до живлення, OTYPER , яким можна налаштувати пін як OpenDrain або Push-pull, за необхідності узгодження за напругою пінів. Сильно заглиблюватись в дані регістри не будемо, так як для виконання завдання вони не потрібні.

В бібліотеці CMSIS задекларовані адреси всіх регістрів, згаданих вище, і всіх інших, які будуть необхідні нам в подальшому вивчені роботи з мікропроцесорами Cortex-M. Бібліотека являє собою опис структур, вказівник яких вказують на базові адреси певних підрозділів карти памяті (таких як RCC, GPIOx, CRC, DMA і т.п.). В тілах структур містяться зміщення до певних регістрів відносно базових адрес, таких як GPIOA-> ODR, GPIOB->IDR, RCC->AHB1ENB і т.п. Також бібліотека містить в собі всі необхідні маски для зручної роботи з регістрами.

Також зазначу, що в STM32F4 є можливість дебагу в Real time, де можливо перевірити вміст всіх регістрів, і взагалі коректність програми. Для виводу регістрів на екран (в фреймворці Keil) при виконанні дебагу, потрібно перейти   
viev 🡪 system viewer , і там вибрати необхідний регістр.

Базову теорію розглянули, тепер можемо приступити до роботи.

**Виконання роботи**

Щоб реалізувати поставлене завдання, необхідно попередньо визначитись з портами вводу / виведення , на які будемо подавати лог. сигнали для семисигментного індикатору. Для цього було обрано перші 7 пінів порту GPIOB. Для кнопок було обрано 13 і 4 піни порту GPIOC.

Перейдемо до написання програми.

Для початку підключимо бібліотеку CMSIS, прописавши наступну строчку в файлы header.h:

#include "stm32f4xx.h"

Тепер ми можемо не оприділять адреси регістрів так як всі вони є в бібліотеці CMSIS

Далі визначимо константи, котрі містять код для формування сигналів для семисигм. індикатору.

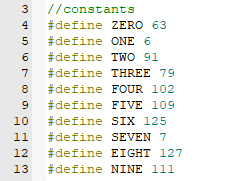


рис.5. Визначення констант для форм. сигнл. семисиг. індикатору

Визначимо прототипи всіх необхідних функцій:

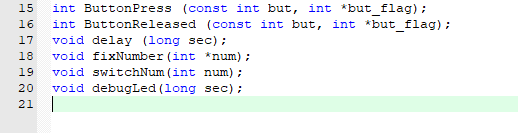


Рис.6. Визначення прототипи функцій обробки кнопок, і т.п.

Перейдемо до головної функції main.

При налаштуванні проекту, ми додали файл startup.s, який являє собою asm файл, який виконує стартове налаштування периферії, оприділення векторів переривань і т.п. Після налаштувань виконується перехід на мітку головної функції main:

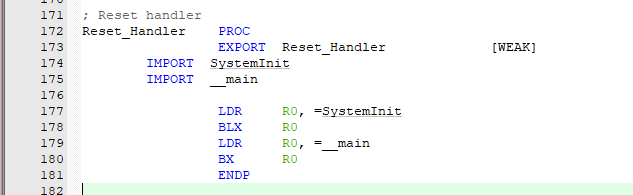


Рис.7. Виконання переходу на мітку main

Перейдемо до основного файлу main:

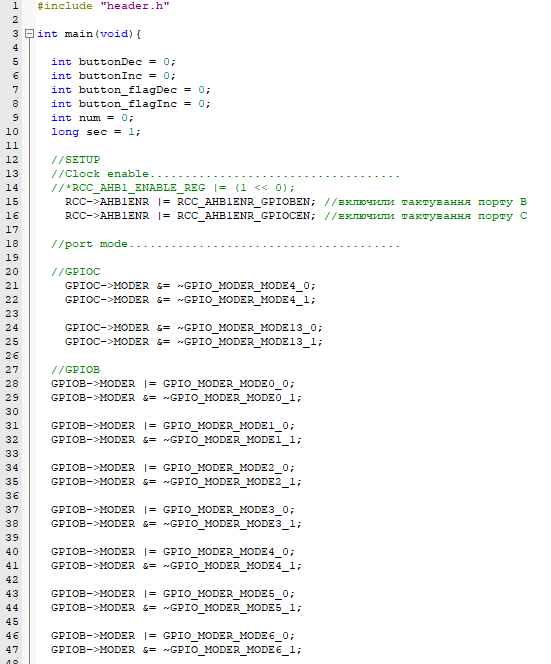


Рис.8. Ініціалізація змінних, ввімкнення тактування портів, їхнє налаштування

На рис.8. представлено початок виконання програми, де проводиться ініціалізація змінних, необхідних для обробки натискання кнопок, збереження числа і т.п. Починаючи зі строчки 14 виконується ввімкнення тактування портів B та C. Виконуємо побітове \*або\* вмісту регістру(RCC->AHB1ENR) з маскою одиниці, здвитуної вліво на певну кількість біт (RCC\_AHB1ENR\_GPIOBEN).

Починаючи з строки 21 виконується налаштування регістру MODERx.

Аналогічна процедура, що і з регістром RCC->AHB1ENR, але тепер потрібно змінити 2 біти, для пінів порту C на зчитування, тобто \*00\*, для пінів порту B на запис \*01\*

Далі буде представлений нескінченний цикл, в якому виконується обробка кнопок, формування сигналів на семис. індикатор та перевірка числа.

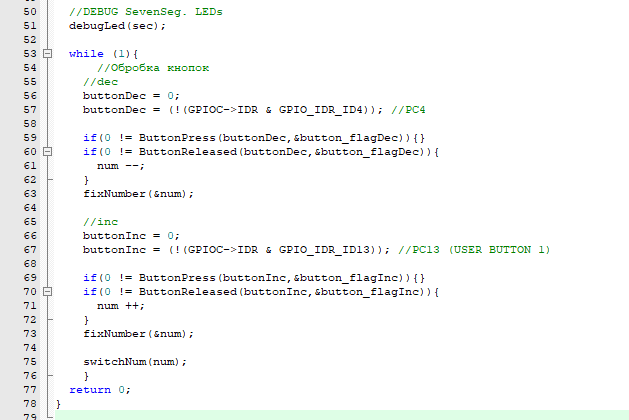


Рис.9. Нескінченний цикл для формування сигналів симисигм. інд. і т.п.

На строці 57 та 67 виконується зчитування бітів регістру IDR порту С, за допомогою побітового і, з маскою (1 здвинутої вліво на к-сть біт, номер якого необхідно зчитати) і порівнюючи отримане значення з нулем. Цей алгоритм реалізований під варіант, коли при натисканні кнопки записується в певний біт регістру IDR нуль, а не одиниця. Тобто коли у нас натиснута кнопка, в змінні записується 1. а коли ні , 0. На строчках 59,60,69,70 виконуються функції обробки натискання кнопки, які боряться з дрязькітом контактів. Також у випадку віджимання кнопок які підкл. до пінів 4 та 13 GPIOC виконується декрементування і інкрементування числа, яке виводиться на семисигм. іникатор, відповідно.

На строчках 63 та 73 виконується функція виправлення числа. У випадку якщо число більше за 9 , то воно обнуляється, а якщо менше за 0, воно стає рівне 9.

На строчці виконується функція запису в регістр ODR значень для формування сигналів семисигм. індикатору.

На рис.10. представлена реалізація всіх вище згаданих функцій.

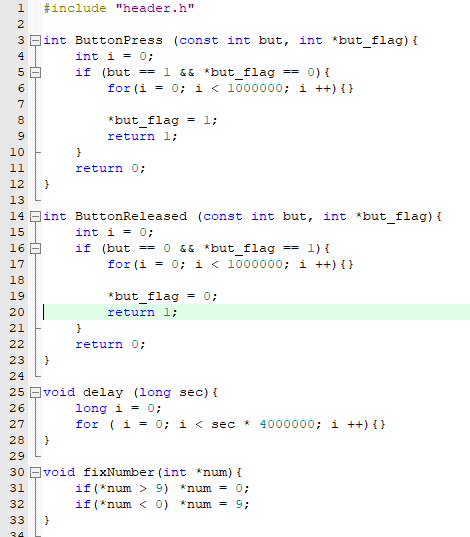


рис.10.а. Тіла функцій обробки натискання кнопок, виправлення числа.

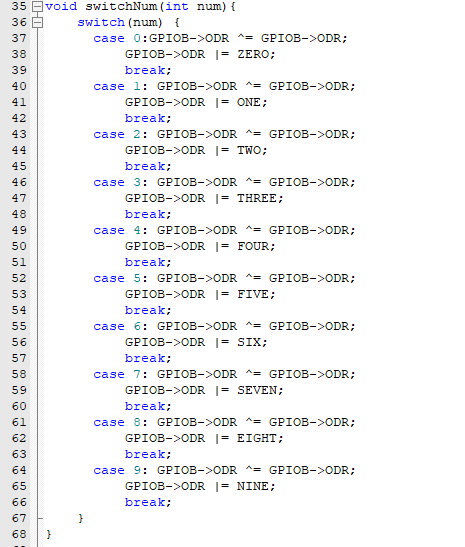


Рис.10.б. Тіло ф-ції формування сигналів для семисигм. індикатору