Міністерство освіти та науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

ЛАБОРАТОРНА РОБОТА №2

з курсу «Операційні системи реального часу»

«Розробка операційної системи реального часу у стилі використання

переривань на базі системного таймеру і таймерів загального призначення»

Виконали:

студенти групи ІТ-81

Тищенко І. В.

Юхимчук А. В.

Ачілов А. В.

Перевірив:

Катін П. Ю.

доцент кафедри АУТС

Київ 2021

# Мета лабораторної роботи

Вивчити сутність програмного і апаратного переривання, вміти розробляти ПС на основі переривання. Вивчити структуру ОСРЧ на основі переривання. Використовувати їх для вирішення практичних завдань. Вивчити технологію використання стандартної бібліотеки CMSIS для обробки переривання. Навчитися формувати сигнал з точно визначеними часовими характеристиками

**Завдання:**

1. Створити і побудувати проект на базі MDK Keil на основі ядра CMSIS, у стилі Interrupt Driven Systems.

2. Налагодити програму для формування сигналу у стилі використання переривань на базі системного таймеру і таймерів загального призначення відповідно до завдання.

3. Налаштувати програму відповідно до завдання, записати програму до Flash МК, у покроковому режимі переконатися у коректному налаштуванні.

4. Протестувати програму на платі налагодження STM32F103C8.

5. (Підвищена складність) Доробити програму так, щоб можна було змінювати ширину імпульсів з кнопки під час виконання програми.

**Хід роботи**

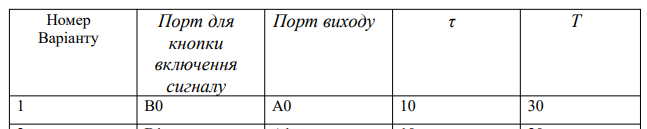
1. Розробити програму, що формує два вихідних сигнали на апаратних виходах МК.

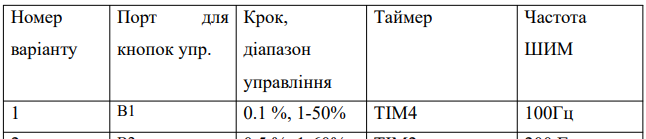
Перший імпульсний сигнал формується з використанням системного таймеру і запускається кнопкою 1, що підключена до апаратного виводу порту відповідно до завдання у таблиці. Характеристики сигналу і апаратні виходи МК, надані далі у таблиці 3.1.

Другий - сигнал ШІМ з керованими характеристиками. Він формується з використанням таймеру загального призначення і керується кнопкою 2, що підключена до апаратного виводу порту відповідно до завдання у таблиці Кнопка 2 дає можливість збільшувати і зменшувати інтенсивність сигналу. Характеристики сигналу і апаратні виходи МК, надані далі у таблиці.

1. Стиль розробки обрати самостійно з двох варіантів, а саме: використання функцій-драйверів CMSIS, використання бібліотек CMSIS для доступу до регістрів МК.
2. Під час захисту ЛР провести вимірювання і підтвердити характеристики сигналів. Або показати результат на налагоджувачі.

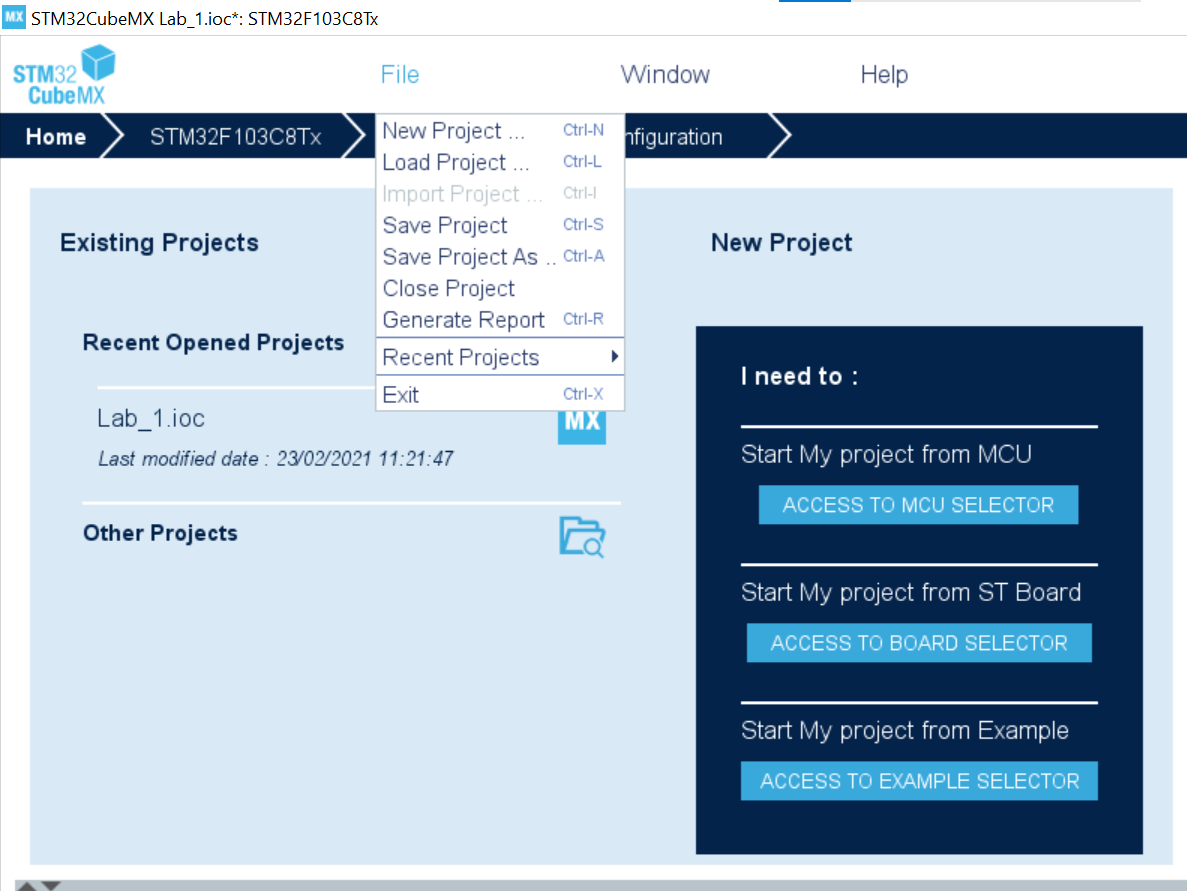
Варіант 1:



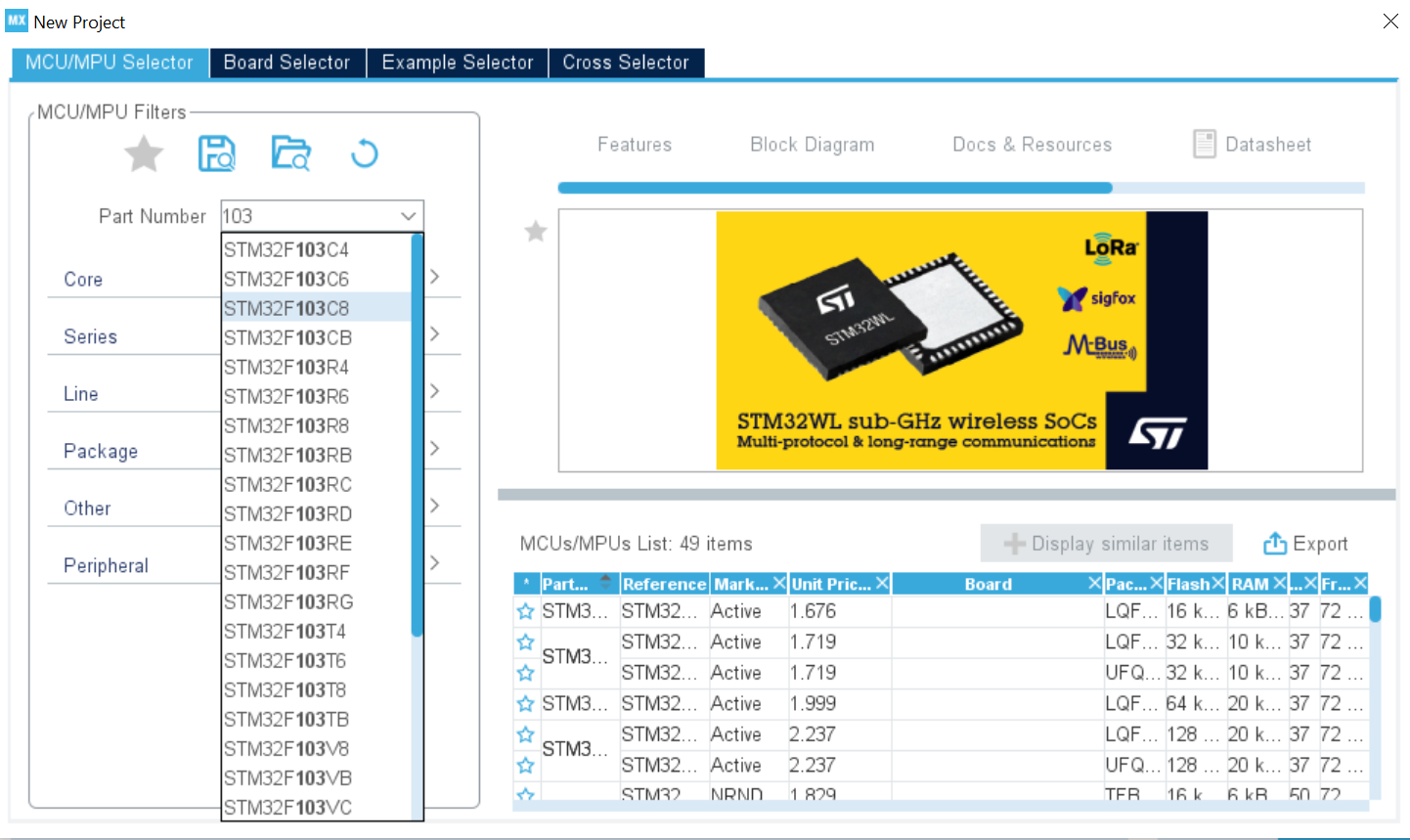


**Виконання роботи**

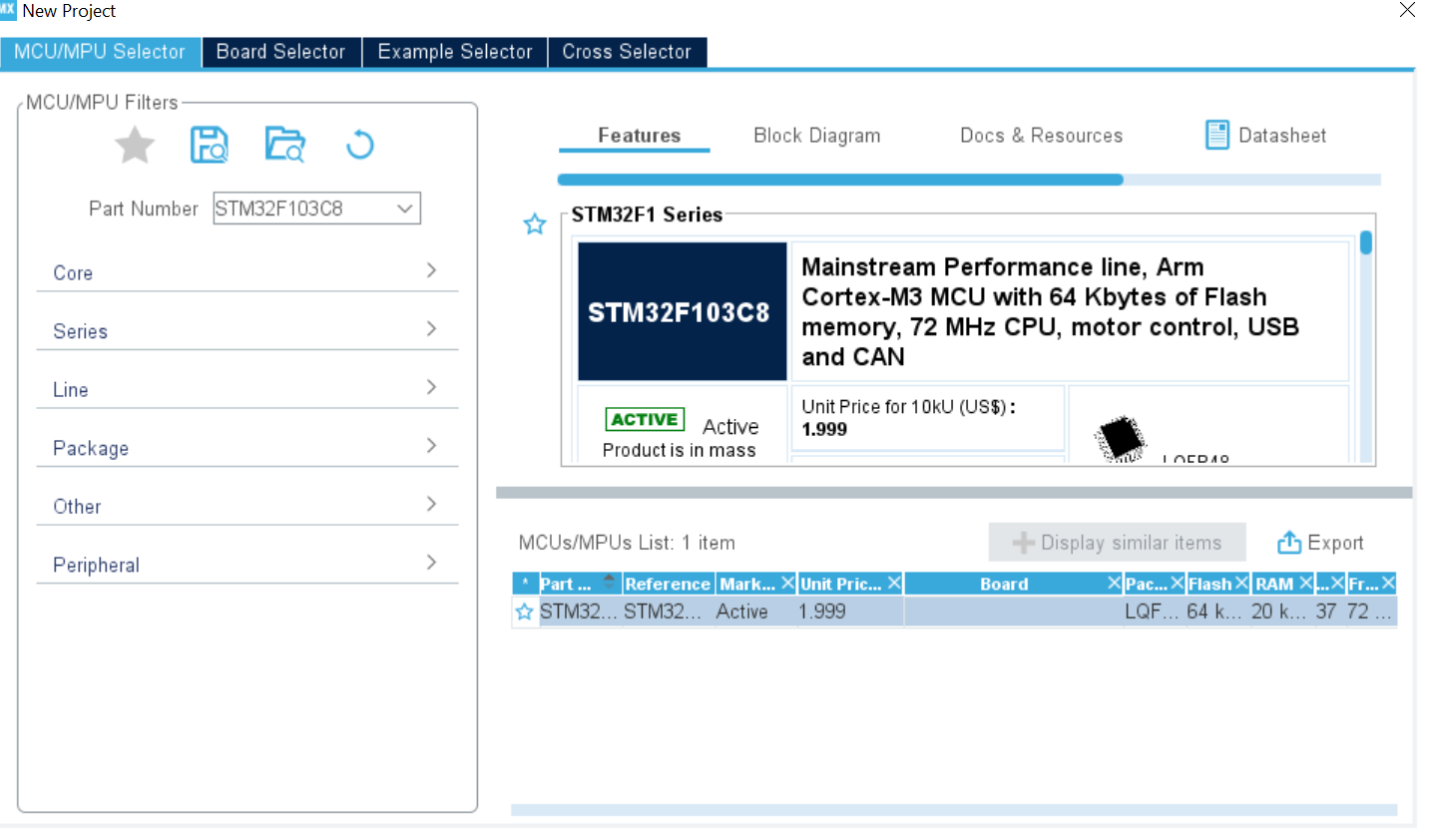
1.Використовуємо середовище CUBEMX і створюємо новий проект



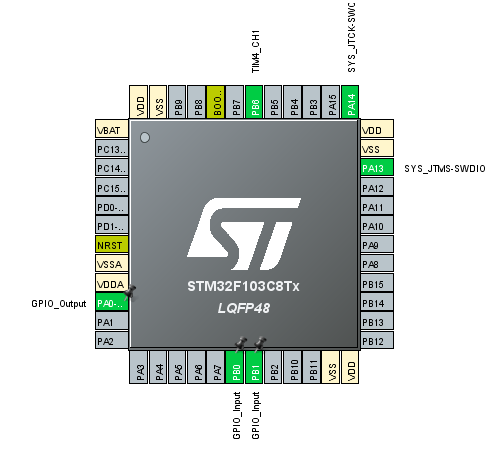
2.Обираємо тип SMTP103C8 мікропроцесора



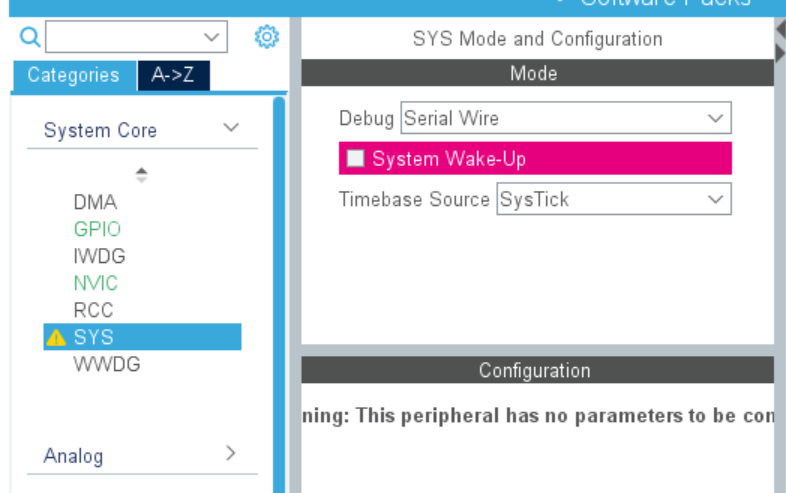
3.Зберігаємо та перходимо до наступного меню



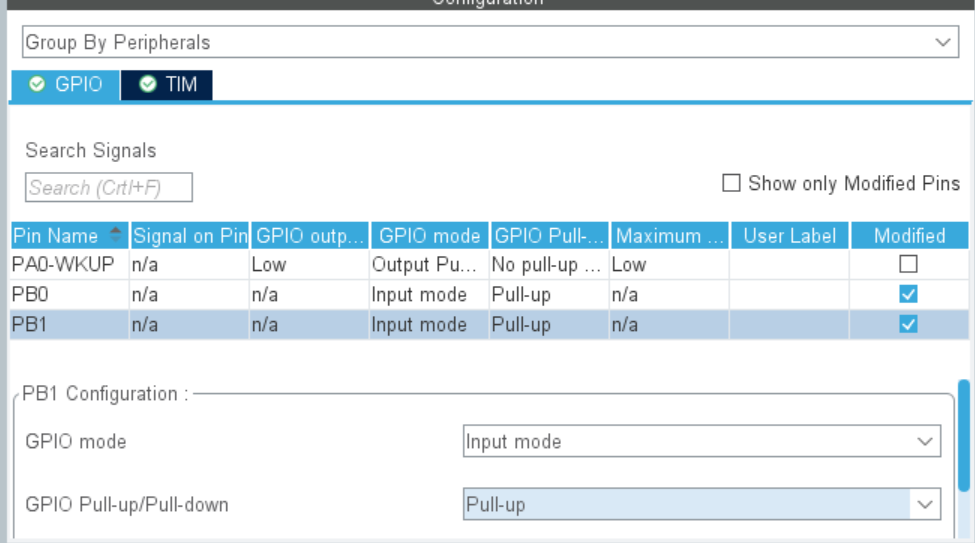
4.Бачимо зображену схему мікропроцесора. По варіанту завдання обираємо пін A0 на вивід, а B0, B1 – на вхід. Пін B6 налаштовуємо на таймер TIM4.



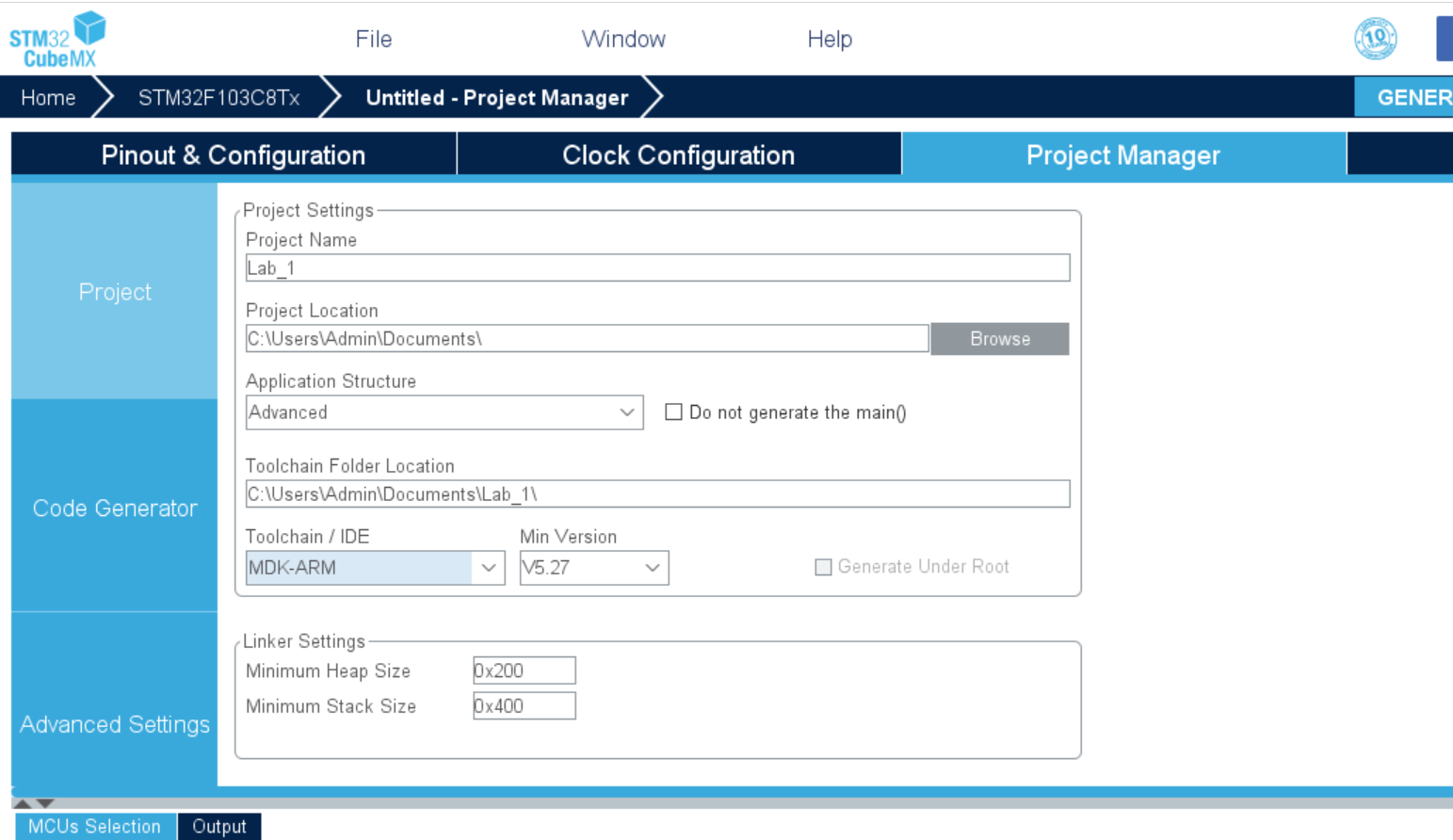
5.Оберемо тип дебагу Serial Ware



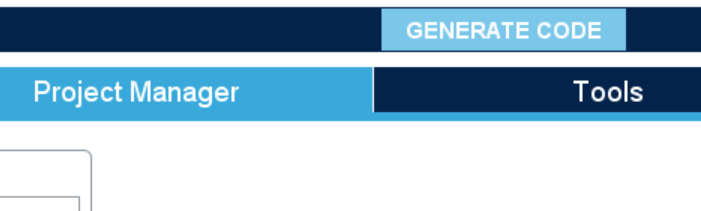
6. Встановлюємо піни B0 і B1 у режим Pull up - коли натискатимемо накнопку, буде генеруватись 0, а ініціалізуватись 1.

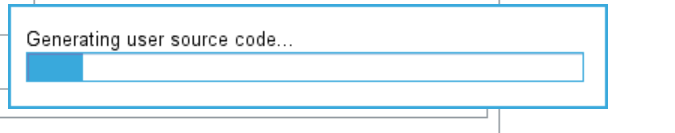


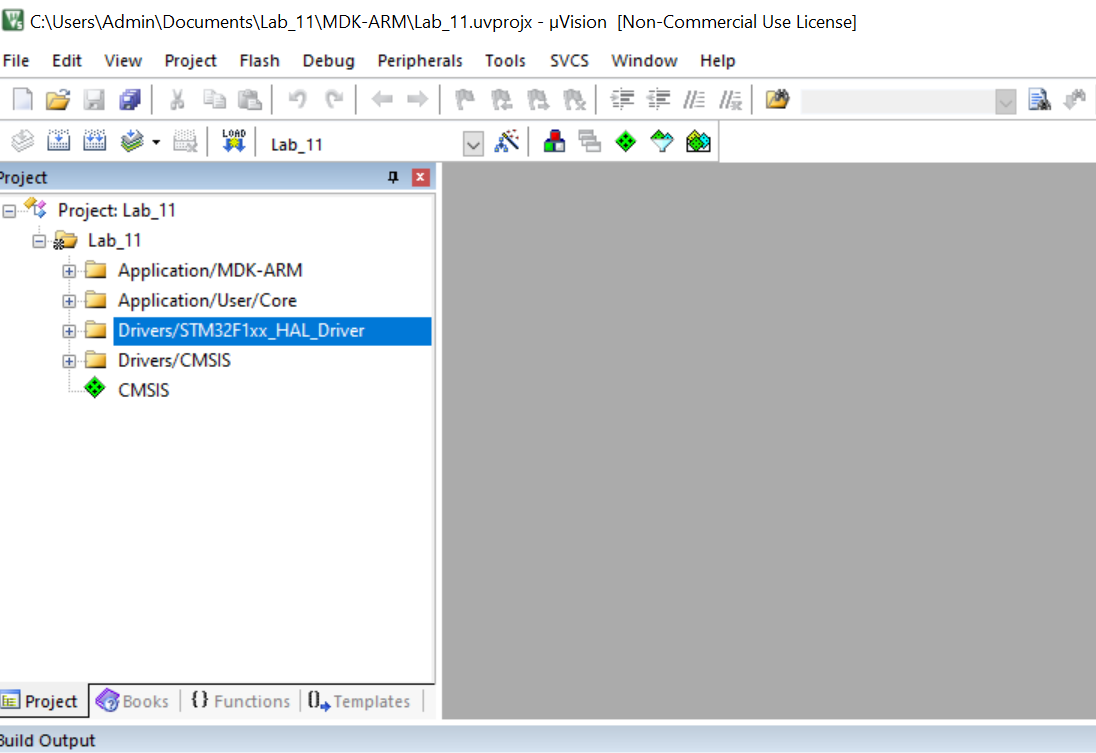
7.Обираємо IDE Weil та обираємо розміщення програми та її назву.



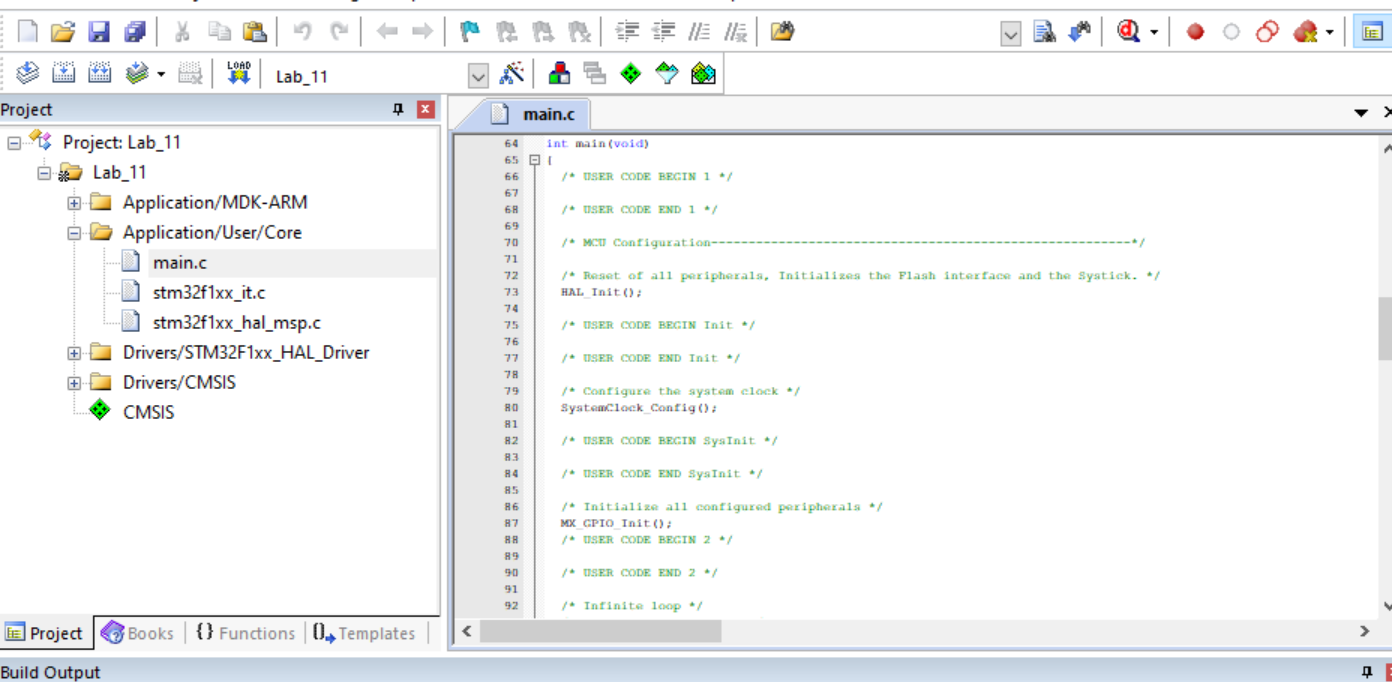
8.Натискаємо згенерувати код



9.Очікуємо певний момент часу поки код генерується і відкриваємо IDE із згенерованим кодом.



10.Сам файл main.c буде згенерований за допомогою CUBEMX та всіх налаштувань, що ми виконали.



11. Вихідний код, який відповідає за циклічну зміну ширини смуги таймера та вмикання/вимикання ШІМ:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under BSD 3-Clause license,

\* the "License"; You may not use this file except in compliance with the

\* License. You may obtain a copy of the License at:

\* opensource.org/licenses/BSD-3-Clause

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#define FALSE 0x00

#define TRUE 0x01

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

TIM\_HandleTypeDef htim4;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM4\_Init(void);

void \_delay\_ms(uint32\_t ms);

char checkButtonClick(uint32\_t pin);

void incrTIM4PulseWidth(void);

void SysTick\_Init(void);

void ChangeSysTickStatus(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM4\_Init();

HAL\_TIM\_PWM\_Start(&htim4, TIM\_CHANNEL\_1);

SysTick\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

if (checkButtonClick(GPIOB->IDR & GPIO\_IDR\_IDR1)){ //if btn on PB1 pressed

incrTIM4PulseWidth(); //increment TIM4 PWM pulse width

\_delay\_ms(50);

}

if (checkButtonClick(GPIOB->IDR & GPIO\_IDR\_IDR0)){ //if btn on PB0 pressed

ChangeSysTickStatus(); //toggle SysTick

\_delay\_ms(1000);

}

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief TIM4 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM4\_Init(void)

{

/\* USER CODE BEGIN TIM4\_Init 0 \*/

/\* USER CODE END TIM4\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_OC\_InitTypeDef sConfigOC = {0};

/\* USER CODE BEGIN TIM4\_Init 1 \*/

/\* USER CODE END TIM4\_Init 1 \*/

htim4.Instance = TIM4;

htim4.Init.Prescaler = SystemCoreClock/100000U - 1;

htim4.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim4.Init.Period = 999;

htim4.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim4.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim4) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_Init(&htim4) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 500;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

if (HAL\_TIM\_PWM\_ConfigChannel(&htim4, &sConfigOC, TIM\_CHANNEL\_1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM4\_Init 2 \*/

/\* USER CODE END TIM4\_Init 2 \*/

HAL\_TIM\_MspPostInit(&htim4);

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : PA0 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pins : PB0 PB1 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_PULLUP;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

}

char checkButtonClick(uint32\_t pin)

{

if (!pin)

{

\_delay\_ms(20);

if (!pin)

return TRUE;

}

return FALSE;

}

void \_delay\_ms(uint32\_t ms)

{

for(uint32\_t i=0; i<ms; i++)

for(uint32\_t j=0; j<9550; j++) {}

}

void incrTIM4PulseWidth(void)

{

if (TIM4->CCR1 == ((TIM4->ARR+1)/100\*50)) //if pulse width is already 50% (in TIM4->CCR1 value 500)

TIM4->CCR1 = 1; //set pulse width to 0.1%

else

TIM4->CCR1 += 1; //else increment pulse width by 0.1%

}

void SysTick\_Init(void)

{

HAL\_SYSTICK\_Config(SystemCoreClock / (100U / uwTickFreq)); //set SysTick reload period to 10 ms

SysTick->CTRL &= ~SysTick\_CTRL\_ENABLE\_Msk; //Disable SysTick

}

void ChangeSysTickStatus(void)

{

if ((SysTick->CTRL & SysTick\_CTRL\_ENABLE\_Msk) == SysTick\_CTRL\_ENABLE\_Msk) //if SysTick enabled,

SysTick->CTRL &= ~SysTick\_CTRL\_ENABLE\_Msk; //disable it

else

SysTick->CTRL |= SysTick\_CTRL\_ENABLE\_Msk; //else enable

}

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

**Висновок**: у ході виконання лабораторної роботи ми створили проект у середовищі CubeMX, і відредагували його код в IDE Keil. ми вивчили сутність програмного і апаратного переривання, спробували розробити ПС на основі переривання. Також, ми дізнались, як формувати сигнал з точно визначеними часовими характеристиками. Після написання інструкцій, ми записати програму до Flash МК та протестували її на платі налагодження STM32F103C8.