МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи № 3
з дисципліни: *"Основи програмування вбудованих систем"*
на тему: *" Робота з потоками у CMSIS-RTOS "*

Варіант 1

Лектор:
доц. каф. ПЗ
Марусенкова Т. А.

Виконав:
ст. гр. ПЗ-32
Бурець В.В.

Прийняв:
доц. каф. ПЗ
Крук О.Г.

« ____ » _____ 2021 р.

∑= _____ .....

Львів – 2021

Тема роботи: Робота з потоками у CMSIS-RTOS.

Мета роботи: Навчитися управляти потоками у операційній системі CMSIS-RTOS у середовищі Keil uVision.

ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Як включити у проект CMSIS-RTOS? Який заголовний файл слід підключити у файлах проекту для використання функцій і констант CMSIS-RTOS?

Щоб застосувати CMSIS-RTOS у проекті, потрібно при створенні проекту в діалозі «Manage Run-Time Environment» відмітити CMSIS=>RTOS(API)=>Keil RTX. Утім, це ж налаштування не пізно зробити для вже існуючого проекту (діалог викликається вибором пункту меню Project=>Manage=>Run-Time Environment…)

Щоб працювати з RTOS, слід підключити заголовний файл cmsis_os.h: #include "cmsis_os.h"

ЗАВДАННЯ

1. Ознайомитися з теоретичним матеріалом.
2. Проаналізувати коди трьох проектів-прикладів (проекти додаються).
3. Виконати індивідуальне завдання (номер завдання відповідає порядковому номеру студента у підгрупі).

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Варіант 1

1. Створити 4 потоки, 2 з яких виконуються завжди з нормальним пріоритетом і вмикають червоний і синій світлодіоди, а ще два – виконуються при встановленні сигнального прапорця з callback-функції таймера і засвічують два інші світлодіоди відповідно.

ХІД РОБОТИ

Для реалізації завдання я виконав наступні дії:

1. Сконфігурував порти GPIO аналогічно до попередньої роботи: порт D – для роботи зі світлодіодами. Відповідні налаштування реалізував окремими функціями у допоміжному файлі LEDInit.c.
2. У файлі LEDTask.c реалізував функції роботи для кожного з чотирьох потоків. Потоки для роботи з синім та червоним світлодіодами виконуються завжди з нормальними пріоритетами, а потоки для роботи

з зеленим і помаранчевим світлодіодами працює відповідно до роботи таймера.

3. Написав callback-функції для роботи із зеленим та помаранчевим світлодіодами.

Для використання CMSIS-RTOS вибрав такі налаштування проекту:

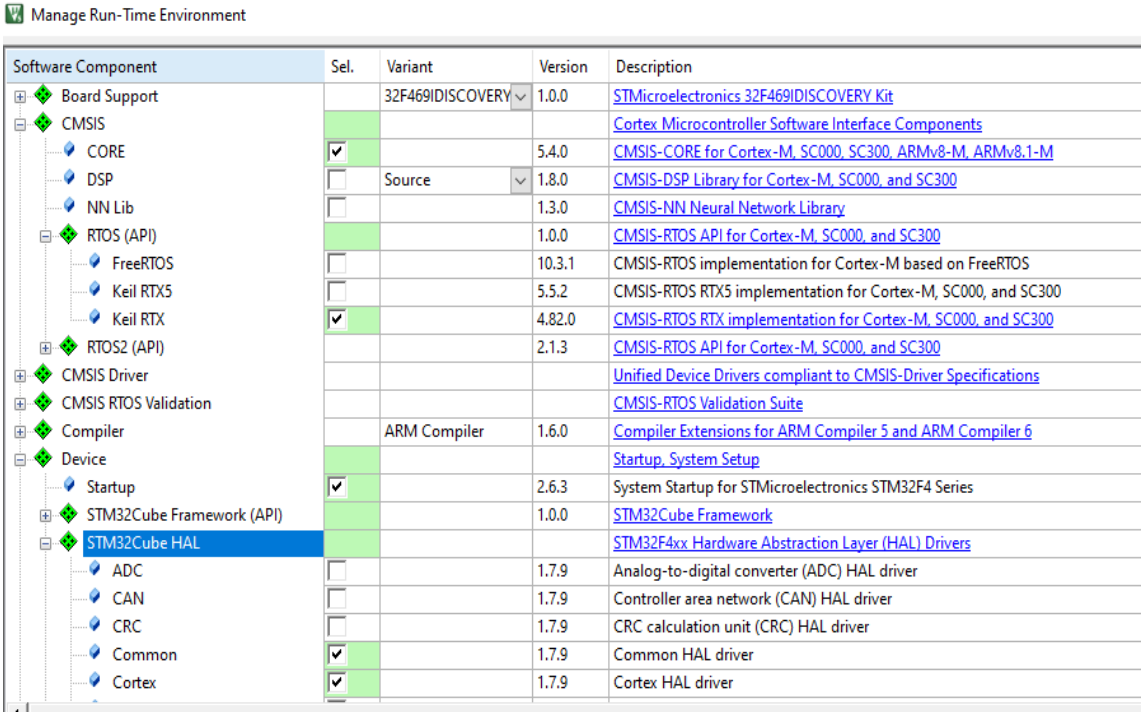

Рис. 1. Компоненти, необхідні для роботи з операційною системою CMSIS-RTOS

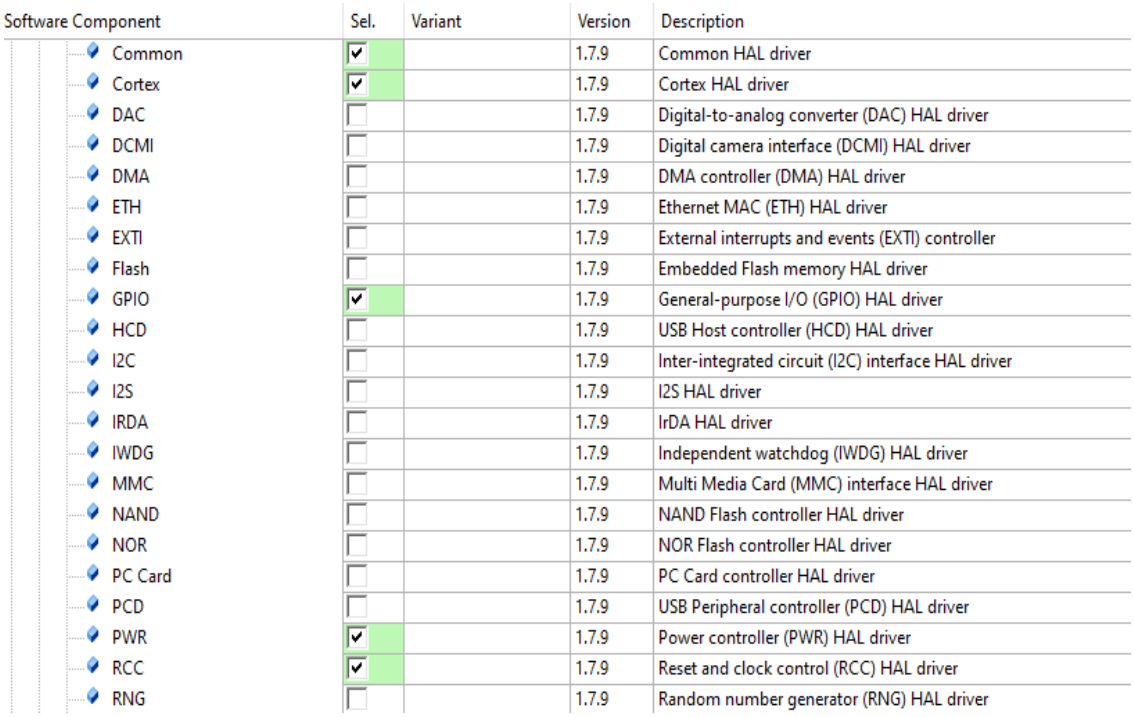

Рис. 2. Компоненти, необхідні для роботи з операційною системою CMSIS-RTOS

КОД ПРОГРАМИ

1) Індивідуальне завдання виконане з використанням бібліотеки CMSIS

Main.c

```
#include "cmsis_os.h"
#include "LEDTask.h"
#include "LEDInit.h"
#include "stm32f4xx_hal_gpio.h"

 // прототип для таймерів
void TimerGreenLED_Callback (void const *arg);
void TimerOrangeLED_Callback (void const *arg);

// оголошення для таймерів
osTimerDef (TimerGreenLED, TimerGreenLED_Callback);
osTimerDef (TimerOrangeLED, TimerOrangeLED_Callback);

//створення потоків
osThreadDef(redLEDTask, osPriorityNormal, 1, 0);
osThreadDef(blueLEDTask, osPriorityNormal, 1, 0);

osThreadDef(greenLEDTask, osPriorityNormal, 1, 0);
osThreadDef(orangeLEDTask, osPriorityNormal, 1, 0);
//--------------

int main(){
      //аргументи дял таймерів
      uint32_t exec1;
      uint32_t exec2;


      osTimerId id1;
      osTimerId id2;

      //ініціалізація світлодіодів
      ConfigGPIO();
      ConfigEXTI();

      //  призупинка ядра, щоб ініціалізувати потоки
      osKernelInitialize();

      exec1 = 1;

//створення таймера для роботи із потоком який працює із зеленим світлодіодом
      id1 = osTimerCreate (osTimer(TimerGreenLED), osTimerPeriodic, &exec1);
      if (id1 != NULL)  {
            if(osTimerStart(id1, 10) != osOK) {

            }
      }
```

```
	exec2 = 2;

//створення таймера для роботи із потоком який працює із помаранчевим
світлодіодом
	id2 = osTimerCreate (osTimer(TimerOrangeLED), osTimerPeriodic, &exec2);
	if (id2 != NULL)  {
		if(osTimerStart(id2, 10) != osOK) {

		}
	}

	// створення потоків для таймерів
	tid_greenLEDTask = osThreadCreate (osThread(greenLEDTask), NULL);
	tid_orangeLEDTask = osThreadCreate (osThread(orangeLEDTask), NULL);
	//------

	// створення звичайних потоків
	tid_redLEDTask = osThreadCreate (osThread(redLEDTask), NULL);
	tid_blueLEDTask = osThreadCreate (osThread(blueLEDTask), NULL);
	//------

	// запуск ядра RTOS, усі потоки будуть виконуватися
	osKernelStart();
}

//лічильники для таймерів
short greenLEDCounter = 0;
short orangeLEDCounter = 0;

//callback-функція для роботи із зеленим світлодіодом
void TimerGreenLED_Callback (void const *arg){
	if(greenLEDCounter == 500){
		osSignalSet(tid_greenLEDTask, 1);
		greenLEDCounter = 0;
	}

	greenLEDCounter++;
}

//callback-функція для роботи із помаранчевим світлодіодом
void TimerOrangeLED_Callback (void const *arg){

	if(orangeLEDCounter == 1000) {
		osSignalSet(tid_orangeLEDTask, 1);
		orangeLEDCounter = 0;
	}

	orangeLEDCounter++;
}
```

LEDTask.h

```
#ifndef TASK_H
#define TASK_H
```

```
#include "cmsis_os.h"

extern osThreadId tid_redLEDTask;
extern osThreadId tid_blueLEDTask;
extern osThreadId tid_greenLEDTask;
extern osThreadId tid_orangeLEDTask;

void greenLEDTask(void const *argument);
void orangeLEDTask(void const *argument);
void redLEDTask(void const *argument);
void blueLEDTask(void const *argument);

#endif
```

LEDTask.c

```
#include "LEDTask.h"
#include "lightutils.h"

osThreadId tid_redLEDTask;
osThreadId tid_blueLEDTask;

osThreadId tid_greenLEDTask;
osThreadId tid_orangeLEDTask;


void greenLEDTask(void const *argument){
	osEvent evt;
	while(1){
		evt = osSignalWait(0x0001,osWaitForever);
		if(evt.status == osEventSignal){
			lightsOnOff(GREEN);
			osDelay(1000);
		}
		osThreadYield();
	}
}

void orangeLEDTask(void const *argument){
	osEvent evt;
	while(1){
		evt = osSignalWait(0x0001,osWaitForever);
		if(evt.status == osEventSignal){
			lightsOnOff(ORANGE);
			osDelay(1000);
		}
		osThreadYield();
	}
}


void redLEDTask(void const *argument){
	lightsOnOff(RED);
	osDelay(1000);
}
```

```
	void blueLEDTask(void const *argument){
		lightsOnOff(BLUE);
		osDelay(1000);
	}
```

LEDInit.h

```
#ifndef INIT_H
#define INIT_H

void ConfigGPIO(void);
void ConfigEXTI(void);

#endif
```

LEDInit.c

```
#include "LEDInit.h"

#include "stm32f4xx.h"

void ConfigGPIO()
{
  GPIOD->MODER = 0x55000000;
  GPIOD->OSPEEDR = 0;
  GPIOD->OTYPER = 0;
  GPIOD->PUPDR = 0x0001;
  GPIOA->MODER = 0x0000;
  GPIOA->PUPDR = 0x0001;

  RCC->AHB1ENR |= (RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_GPIODEN);
}

void ConfigEXTI()
{
  SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA;
  EXTI->IMR  |= EXTI_IMR_IM0;
  EXTI->RTSR |= EXTI_RTSR_TR0;

  NVIC_EnableIRQ(EXTI0_IRQn);
  NVIC_SetPriority(EXTI0_IRQn,1);
}
```

lightutils.h

```
#ifndef LIGHTUTILS_H
#define LIGHTUTILS_H
#define ON 1
#define OFF 0

typedef enum {
	GREEN = 0x1000 ,
	ORANGE = 0x2000,
	RED = 0x4000,
```

```
	BLUE = 0x8000
} Color;

void lightsOnOff(Color color, short);

#endif
```

lightutils.c

```
#include "lightutils.h"
#include "stm32f4xx_hal_gpio.h"

void lightsOnOff(Color color, short state)
{
	if(state)
	{
		GPIOD->ODR |=color;
	}
	else
	{
		GPIOD->ODR &=(~color);
	}
}
```

РЕЗУЛЬТАТИ

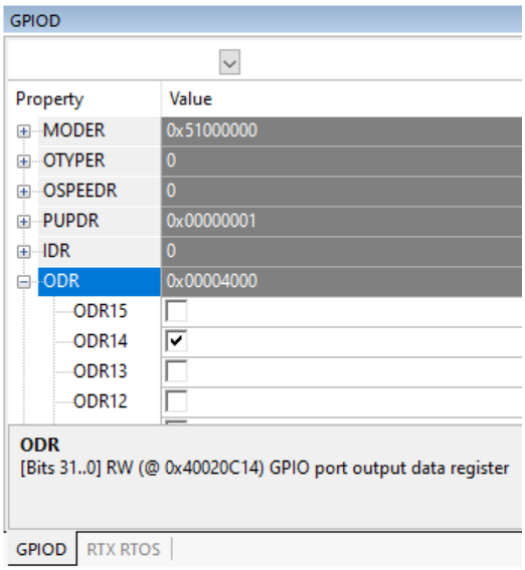

Рис.3.Результат роботи потоку який засвічує червоний світлодіод

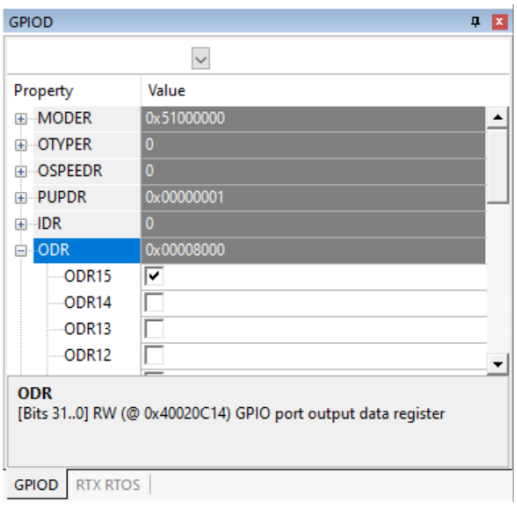

Рис. 4. Результат роботи потоку який засвічує синій світлодіод

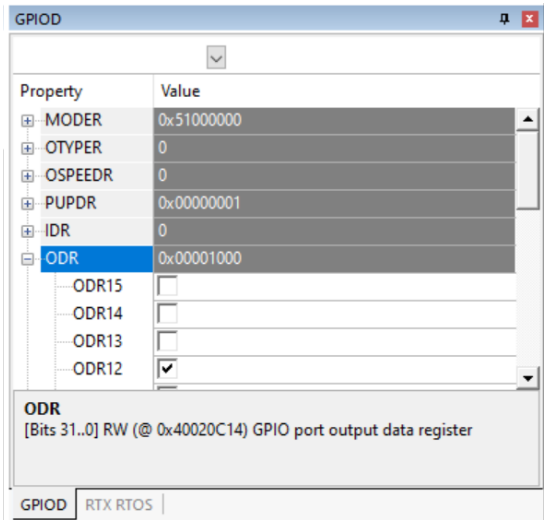

Рис. 5. Результат роботи потоку який засвічує зелений світлодіод

Рис. 5. Робота лічильників які працюють в callback-функціях

ВИСНОВКИ

Виконуючи цю лабораторну роботу, я ознайомився та навчився керувати потоками у операційній системі CMSIS-RTOS на платі STM32F4DISCOVERY. Написав програму для роботи зі світлодіодами з використанням окремих потоків та таймерів, та виконав завдання відповідно до свого варіанту, а саме створив 4 потоки, 2 з яких виконуються завжди з нормальним пріоритетом і вмикають червоний і синій світлодіоди відповідно, а інші два – засвічує зелений та помаранчевий світлодіоди при встановленні сигнального прапорця в callback-функції.