

**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.

Σ = ____

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:

Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Board Support		32F469IDISCOVERY	1.0.0	STMicroelectronics 32F469IDISCOVERY Kit
CMSIS				Cortex Microcontroller Software Interface Components
CORE	<input checked="" type="checkbox"/>		5.4.0	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M
DSP	<input type="checkbox"/>	Source	1.8.0	CMSIS-DSP Library for Cortex-M, SC000, and SC300
NN Lib	<input type="checkbox"/>		1.3.0	CMSIS-NN Neural Network Library
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
FreeRTOS	<input type="checkbox"/>		10.3.1	CMSIS-RTOS implementation for Cortex-M based on FreeRTOS
Keil RTX5	<input type="checkbox"/>		5.5.2	CMSIS-RTOS RTX5 implementation for Cortex-M, SC000, and SC300
Keil RTX	<input checked="" type="checkbox"/>		4.82.0	CMSIS-RTOS RTX implementation for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.3	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS RTOS Validation				CMSIS-RTOS Validation Suite
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Startup	<input checked="" type="checkbox"/>		2.6.3	System Startup for STMicroelectronics STM32F4 Series
STM32Cube Framework (API)			1.0.0	STM32Cube Framework
STM32F4xx HAL				STM32F4xx Hardware Abstraction Layer (HAL) Drivers
ADC	<input type="checkbox"/>		1.7.9	Analog-to-digital converter (ADC) HAL driver
CAN	<input type="checkbox"/>		1.7.9	Controller area network (CAN) HAL driver
CRC	<input type="checkbox"/>		1.7.9	CRC calculation unit (CRC) HAL driver
Common	<input checked="" type="checkbox"/>		1.7.9	Common HAL driver
Cortex	<input checked="" type="checkbox"/>		1.7.9	Cortex HAL driver

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

Software Component	Sel.	Variant	Version	Description
Common	<input checked="" type="checkbox"/>		1.7.9	Common HAL driver
Cortex	<input checked="" type="checkbox"/>		1.7.9	Cortex HAL driver
DAC	<input type="checkbox"/>		1.7.9	Digital-to-analog converter (DAC) HAL driver
DCMI	<input type="checkbox"/>		1.7.9	Digital camera interface (DCMI) HAL driver
DMA	<input type="checkbox"/>		1.7.9	DMA controller (DMA) HAL driver
ETH	<input type="checkbox"/>		1.7.9	Ethernet MAC (ETH) HAL driver
EXTI	<input type="checkbox"/>		1.7.9	External interrupts and events (EXTI) controller
Flash	<input type="checkbox"/>		1.7.9	Embedded Flash memory HAL driver
GPIO	<input checked="" type="checkbox"/>		1.7.9	General-purpose I/O (GPIO) HAL driver
HCD	<input type="checkbox"/>		1.7.9	USB Host controller (HCD) HAL driver
I2C	<input type="checkbox"/>		1.7.9	Inter-integrated circuit (I2C) interface HAL driver
I2S	<input type="checkbox"/>		1.7.9	I2S HAL driver
IRDA	<input type="checkbox"/>		1.7.9	IrDA HAL driver
IWDG	<input type="checkbox"/>		1.7.9	Independent watchdog (IWDG) HAL driver
MMC	<input type="checkbox"/>		1.7.9	Multi Media Card (MMC) interface HAL driver
NAND	<input type="checkbox"/>		1.7.9	NAND Flash controller HAL driver
NOR	<input type="checkbox"/>		1.7.9	NOR Flash controller HAL driver
PC Card	<input type="checkbox"/>		1.7.9	PC Card controller HAL driver
PCD	<input type="checkbox"/>		1.7.9	USB Peripheral controller (PCD) HAL driver
PWR	<input checked="" type="checkbox"/>		1.7.9	Power controller (PWR) HAL driver
RCC	<input checked="" type="checkbox"/>		1.7.9	Reset and clock control (RCC) HAL driver
RNG	<input type="checkbox"/>		1.7.9	Random number generator (RNG) HAL driver

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 для 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) {

        }
    }
}
```

```

    ехес2 = 2;

//створення таймера для роботи із потоком який працює із помаранчевим
світлодіодом
    id2 = osTimerCreate (osTimer(TimerOrangeLED), osTimerPeriodic, &ехес2);
    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_RTISR_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

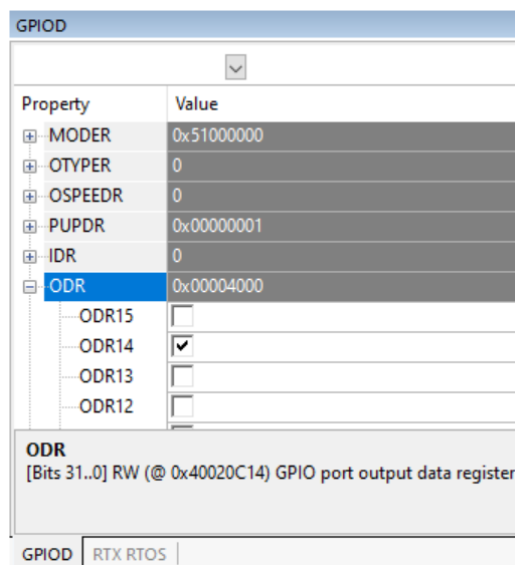


Рис.3.Результат роботи потоку який засвічує червоний світлодіод

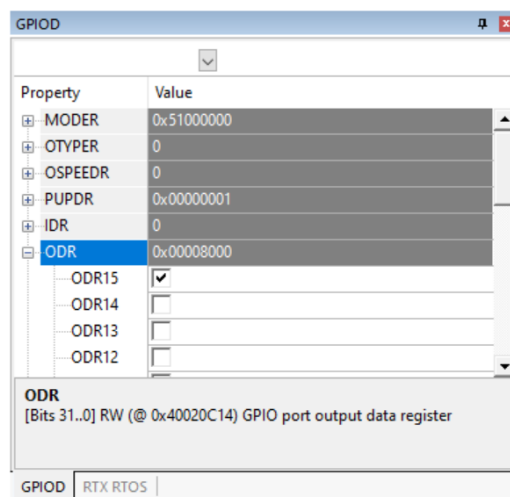


Рис. 4. Результат роботи потоку який засвічує синій світлодіод

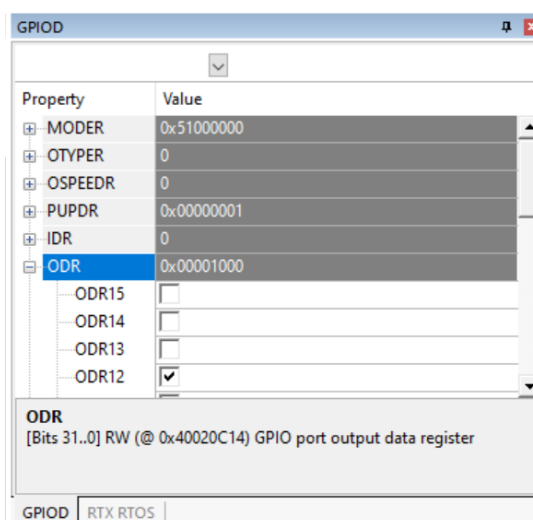


Рис. 5. Результат роботи потоку який засвічує зелений світлодіод

greenLEDCounter	144	short
orangeLEDCounter	644	short

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.