МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ»

Кафедра САУ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы реального времени»

Тема: «УПРАВЛЕНИЕ ЗАДАЧАМИ»

Вариант - 3

Студенты гр. 8491 Туленков К.А.

Саламахин А.

Преподаватель Гречухин М.Н.

Санкт-Петербург

2022

Цель работы: познакомиться с различными функциями управления задачами, алгоритмами планирования, реализованными в ОС FreeRTOS.

Задание:

1. Открыть IDE Keil MDK-ARM, создать новый проект по алгоритму из работы 1. В файле конфигурации FreeRTOSConfig.h необходимо разрешить использовать нужные в работе функции (с помощью директив вида #define INCLUDE_vTaskDelay 1).

2. Создать две функции-задачи. Одна из них будет мигать диодом на плате, вторая – реализовывать бегущий огонь на внешней диодной линейке. Варианты параметров частоты и внешнего вида бегущего огня приведены в табл. 2.3. Приоритет задачам поставить равный. Частоту мигания диода и задержку переключения бегущего огня реализовать приближённо, с помощью цикла for, подобрав подходящее число итераций.

3. Настроить использование вытесняющей многозадачности без разделения времени.

4. Скомпилировать проект. Загрузить его на плату, наблюдать работу с подключенной периферией. Наблюдать работу программы.

5. Настроить использование алгоритма вытесняющей многозадачности с разделением времени. Пересобрать проект, загрузить на плату, наблюдать работу системы.

6. Изменить поведение задач. Первая задача делает m (см. табл. 2.3) включений и выключений диода на плате с периодом 1 с, после чего приостанавливается на 100 тиков. Вторая – пробегает по линейке бегущим огнём m раз, после чего приостанавливается на 5 тиков. Приоритеты оставить равные. Загрузить проект на плату, наблюдать работу.

7. Изменить приоритеты задачи. Повысить приоритет второй задачи относительно первой. Загрузить проект на плату, наблюдать работу.

8. Изменить задачи, реализовав точную задержку с помощью API FreeRTOS вместо приближённой, реализованной с помощью for.

9. Изменить алгоритм задач. Первая задача в начале цикла приостанавливает вторую задачу, далее выполняет $m \times 2$ циклов включения/выключения диода на плате, после чего возобновляет вторую задачу и блокируется на две секунды. Время приостановки задачи изменить по необходимости. Загрузить проект на плату, наблюдать работу.

Таблица 1

Варианты заданий

Вариант	Частота мигания диода, Гц	Вид бегущего огня	m
1	1	Слева направо по одному	3
2	10	Справа налево по два	2
3	0.5	Сходящийся к центру	4
4	5	Расходящийся от центра	3

10. Изменить алгоритм задач. В начале работы создаётся только первая задача. Первая задача выполняет $m \times 2$ циклов включения\выключения диода, после чего создаёт вторую задачу с приоритетом, равным своему. После того, как первая задача выполнит в общей сложности $m \times 3$ циклов включения\выключения диода, она удаляет вторую задачу. Далее процесс повторяется циклически. Загрузить проект на плату, наблюдать его работу.

Код программы:

```
#include "stm32f4xx.h"                  // Device header
#include "FreeRTOS.h"                   // ARM.FreeRTOS::RTOS:Core
#include "FreeRTOSConfig.h"             // ARM.FreeRTOS::RTOS:Config
#include "task.h"                       // ARM.FreeRTOS::RTOS:Core
#include "queue.h"                      // ARM.FreeRTOS::RTOS:Core
#include "inttypes.h"
void vBlinkTaskFunction(void* pvParameters)
{
        while(1)
        {
                GPIOA->ODR |= GPIO_ODR_ODR_5; // led on A5
                vTaskDelay(1000);
                GPIOA->ODR &= ~GPIO_ODR_ODR_5; //led off
                vTaskDelay(1000);
        }
}
void vRunningTaskFunction(void* pvParameters)
{
        while(1)
        {
                for (int i=0;i<8;i++)
                {
                        GPIOC->ODR = (1 << (15-i)) | (1 << (i));
                        vTaskDelay(2000);
                }
        }
}
void InitHardware()
{
        RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;
        RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;
        GPIOA->MODER |= GPIO_MODER_MODER5_0;
}
int main ()
{
        InitHardware();
        xTaskCreate(vBlinkTaskFunction, "blink", configMINIMAL_STACK_SIZE, NULL,
4, NULL);
        xTaskCreate(vRunningTaskFunction, "runnig", configMINIMAL_STACK_SIZE,
NULL, 4, NULL);
        vTaskStartScheduler();
        while(1)
        {
        }
}
```

Описание кода программы:

vBlinkTaskFunction – мигает диодом, который подключен к выводу PA5 на плате, с частотой 1Гц.

vRunningTaskFunction – бегущий огонь, используются выходы порта GPIOC, к которым подключены 16 светодиодов.

InitHardware – инициализация портов.

Работа программы:

Задание 1 и 2:

Разрешаем использовать функцию прописывая в конфиге #define INCLUDE_vTaskDelay 1 и создаем требуемые функции-задачи.

```
void vBlinkTaskFunction(void* pvParameters)
{
	while(1)
	{
		GPIOA->ODR |= GPIO_ODR_ODR_5; // led on A5
		vTaskDelay(1000);
		GPIOA->ODR &= ~GPIO_ODR_ODR_5; //led off
		vTaskDelay(1000);
	}
}
void vRunningTaskFunction(void* pvParameters)
{
	while(1)
	{
		for (int i=0;i<8;i++)
		{
			GPIOC->ODR = (1 << (15-i)) | (1 << (i));
			vTaskDelay(2000);
		}
	}
}
```

Задание 3 и 4:

Настраиваем использование вытесняющей многозадачности без разделения времени

```
1 #define configUSE_PREEMPTION  1
2 #define configUSE_TIME_SLICING 0
```

Задание 5:

Настраиваем использование вытесняющей многозадачности с разделением времени

```
1 #define configUSE_PREEMPTION  1
2 #define configUSE_TIME_SLICING 1
```

Задание 6:

Изменяем функции-задачи соответственно заданию. Приоритеты равные.

```
1  void vBlinkTaskFunction(void* pvParameters)
 2 {
 3         while(1)
 4         {
 5                 for (int m=0;m<4;m++)
 6                 {
 7                         GPIOA->ODR |= GPIO_ODR_ODR_5; // led on A5
 8                         vTaskDelay(1000);
 9                         GPIOA->ODR &= ~GPIO_ODR_ODR_5; //led off
10                         vTaskDelay(1000);
11                 }
12                 vTaskDelay(100);
13         }
14 }
15
16 void vRunningTaskFunction(void* pvParameters)
17 {
18         while(1)
19         {
20                 for (int m=0;m<4;m++)
21                 {
22                         for (int i=0;i<8;i++)
23                         {
24                                 GPIOC->ODR = (1 << (15-i)) | (1 << (i));
25                                 vTaskDelay(1000);
26                         }
27                 }
28                 vTaskDelay(5);
29         }
30 }
```

Задание 7:

Повышаем приоритет второй задачи относительно первой

```
1 xTaskCreate(vBlinkTaskFunction, "blink", configMINIMAL_STACK_SIZE, NULL, 4, &blink);
2 xTaskCreate(vRunningTaskFunction, "runnig", configMINIMAL_STACK_SIZE, NULL, 8,
  &running);
```

Задание 8 и 9:

Изменяем функции-задачи соответственно заданию.

```
void vBlinkTaskFunction(void* pvParameters)
{
        while(1)
        {
                vTaskSuspend(running);
                for (int m=0;m<8;m++)
                {
                        GPIOA->ODR |= GPIO_ODR_ODR_5; // led on A5
                        vTaskDelay(1000);
                        GPIOA->ODR &= ~GPIO_ODR_ODR_5; //led off
                        vTaskDelay(1000);
                }
                vTaskResume(running);
                vTaskDelay(2000);
        }
}
```

Задание 10:

Изменяем функции-задачи соответственно заданию.

```
void vBlinkTaskFunction(void* pvParameters)
{
        while(1)
        {
                for (int j=0;j<12;j++)
                {
                        GPIOA->ODR |= GPIO_ODR_ODR_5; // led on A5
                        vTaskDelay(1000);
                        GPIOA->ODR &= ~GPIO_ODR_ODR_5; //led off
                        vTaskDelay(1000);
                        if (j==7) xTaskCreate(vRunningTaskFunction, "running",
configMINIMAL_STACK_SIZE, NULL, uxTaskPriorityGet(blink), &running);
                }
                vTaskDelete(running);
        }
}
```

Вывод: выполнив лабораторную работу, мы познакомились с различными функциями управления задачами, алгоритмами планирования, реализованными в ОС FreeRTOS.