МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизация производственных процессов»

Операционные системы реального времени

Задания для контрольной работы

Ростов-на-Дону
ДГТУ
2023

УДК 681.5

Составитель: Быкадор В.С.

Методические указания. – Ростов-на-Дону : Донской гос. техн. ун-т, 2023. – 29 с.

Методические указания по дисциплине «Операционные системы реального времени. Задания для контрольной работы» предназначены для студентов заочной формы обучения по направлению подготовки 15.04.04 «Автоматизация технологических процессов и производств».

УДК 681.5

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

В печать ___.___.20__г.
Формат 60х84/16. Объем ____ усл. п. л.
Тираж___ экз. Заказ № ___.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

Выбор варианта задания

Задание состоит из трёх разделов. В первом разделе необходимо написать ответ на одни из теоретических вопросов, распределение вопросов приведено в таблице № 1. Во-втором разделе необходимо реализовать ряд программ с применением API-функций операционный системы реального времени (ОСРВ) FreeRTOS, связанных с базовыми механизмами работы ОСРВ. В третьем разделе необходимо выполнить реализацию программ с применением API-функций ОСРВ FreeRTOS, связанных с механизмами конкуренции задач. Второй и третий разделы, выполняются без распределения по вариантам - все задания для всех студентов.

ЗАДАНИЯ ПЕРВОГО РАЗДЕЛА

Таблица № 1. Распределение вопросов первого раздела контрольной работы по вариантам

	ПОСЛЕДНЯЯ ЦИФРА ЗАЧЁТНОЙ КНИЖКИ									
ПРЕДПОСЛЕДНЯЯ ЦИФРА ЗАЧ. КН.		0	1	2	3	4	5	6	7	8	9
	0	1	11	21	31	25	3	12	22	12	2
	1	2	12	22	32	26	4	13	23	13	3
	2	3	13	23	1	11	21	14	24	14	4
	3	4	14	24	2	12	22	15	25	15	5
	4	5	15	25	3	13	23	16	26	16	6
	5	6	16	26	4	14	24	3	13	31	7
	6	7	17	27	5	15	25	4	14	32	19
	7	8	18	28	6	16	26	5	15	13	20
	8	9	19	29	7	17	27	6	16	14	31
	9	10	20	30	8	18	28	7	17	15	32

Вопросы первого раздела контрольной работы:

1) Что такое переключение контекста задач планировщиком и для чего это необходимо в FreeRTOS?
2) Что такое базовый тип данных в FreeRTOS и для чего он был введен в данную ОСРВ?
3) Какие состояния задачи существуют в FreeRTOS, для чего необходимы эти состояния и каким образом задача переходит из одного состояния в другое.

4) Как выполняется передача параметра в задачу и восстановление параметра в задаче в FreeRTOS (покажите на примере передачи структуры)?
5) Что такое задача «Бездействия», как реализовать свою задачу «Бездействия» в FreeRTOS?
6) Какие две базовые схемы выделения памяти применяются в FreeRTOS?
7) Почему существуют разные схемы выделения памяти в FreeRTOS?
8) Какой алгоритм используется для поиска незанятых блоков памяти в FreeRTOS и как работает этот алгоритм?
9) Какие типы многозадачностей реализованы в FreeRTOS?
10) Приведите диаграмму переключения между задачами для вытесняющей и кооперативной многозадачности?
11) Как организовано хранение информации в очередях FreeRTOS?

12) Как организовать разбор сообщении в очереди FreeRTOS?
13) Что такое отложенная обработка прерываний в FreeRTOS?
14) Что такое двоичные семафоры и API-функции для работы с ними в FreeRTOS?
15) Что такое счетные семафоры и API-функции для работы с ними в FreeRTOS?
16) Как реализовать передачу данных через очереди в обработчиках прерываний в FreeRTOS?
17) Как повысить эффективность передачи данных через очереди в обработчиках прерываний в FreeRTOS?
18) Что такое вложенность прерываний в FreeRTOS?
19) Что такое механизм взаимного исключения в FreeRTOS?
20) Какие способы используются для организации взаимного исключения в FreeRTOS?
21) Что такое неатомарные операции и реентерабельность функций в FreeRTOS (приведите примеры)?
22) Что такое критическая секция в FreeRTOS ?
23) Какие способы организации критических секций применяются в FreeRTOS?
24) Что такое мьютексы в FreeRTOS?
25) Что такое рекурсивные мьютексы в FreeRTOS?
26) Поясните проблемы возникают при работе с мьютексами в FreeRTOS?
27) Для каких целей может потребоваться API-функция vApplicationTickHook() FreeRTOS (поясните на примере)?

28) Что такое задача-сторож и как может быть использована задача-сторож в FreeRTOS?
29) Что такое сопрограмма в FreeRTOS и какие у нее преимущества?
30) Какие существуют состояния у сопрограммы в FreeRTOS?
31) Как выполняются сопрограммы и как работаю приоритеты у сопрограмм в FreeRTOS?
32) Какие существуют ограничения при использовании сопрограмм в FreeRTOS?

Базовый пример программы на языке программирования Си для микроконтроллера с ОСРВ FreeRTOS

Ниже приведен листинг программы на языке программирования Си для микроконтроллера с операционной системой реального времени (ОСРВ) FreeRTOS, позволяющей коммутировать одним светодиодом. Данный листинг программы может служить каркасом для заданий в данном методическом указании.

```
// подключаем библиотеку с ОСРВ FreeRTOS
#include <Arduino_FreeRTOS.h>
// подключаем стандартную библиотеку для МК AVR
#include <avr/io.h>

// создаем функцию для прикладной задачи
void vTask(void *pvParameters)
{
  // «бесконечная» коммутация светодиодом VD1
  while(1)
  {
    PORTB ^= (1 << PB5);
    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
  vTaskDelete(NULL);
}

void setup() {
  // настройка порта PORTB на вывод данных
  DDRB = 0xFF;
  PORTB = 0xFF;
  // создаём из функции прикладную задачу,
  // которую будет выполнять ОСРВ FreeRTOS
  xTaskCreate(
          vTask,
          "",
          configMINIMAL_STACK_SIZE,
          NULL,
          1,
          NULL
          );
  // запускаем планировщик ОСРВ FreeRTOS выполнение
  vTaskStartScheduler();
}
// данную функцию оставляем в коде пустой, она требуется для
компиляции проекта в Arduino IDE
void loop() {}
```

ЗАДАНИЯ ВТОРОГО РАЗДЕЛА

ЗАДАНИЕ № 1. Исследование программного управления коммутацией выводов порта МК под управлением ОСРВ FreeRTOS

Цели: 1) изучить организацию программы на языке программирования Си, при использовании API-функций прикладных задач операционной системы реального времени (ОСРВ) FreeRTOS;

2) изучить принцип псевдопараллельной работы прикладных задач в ОСРВ FreeRTOS на однокристальнном микроконтроллере Atmega328p.

Таблица № 1.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 1
Подключаемые библиотеки	• Arduino_FreeRTOS.h • avr/io.h • util/delay.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelete(NULL) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE
Ограничения	• нельзя использовать глобальные переменные • приоритеты задач одинаковые, например, равные 1

Описание

1) Реализовать **синхронную** коммутацию любых двух светодиодов, подключенных к выводам микроконтроллера. Под синхронной коммутацией подразумевается включение и выключение светодиодов в кратные моменты времени, например, 1 секунда, 2 секунды, 3 секунды и так далее. Рисунке 1.1. (а) показана синхронная коммутация светодиодов, как можно видеть, временные интервалы включенного состояния светодиодов VD1 и VD2 равны между собой.

2) Реализовать **асинхронную** коммутацию любых двух светодиодов, подключенных к выводам микроконтроллера. Под асинхронной коммутацией подразумевается включение и выключение светодиодов в некратные моменты времени. Рисунке 1.1. (б) показана асинхронная коммутация светодиодов, как можно видеть, временные интервалы включенного состояния светодиодов VD1 и VD2 неравны между собой не смотря на то, что периоды включения и выключения каждого из светодиодов VD1 и VD2 не изменяются. Но периоды включенного и выключенного состояния светодиодов не являются кратными.

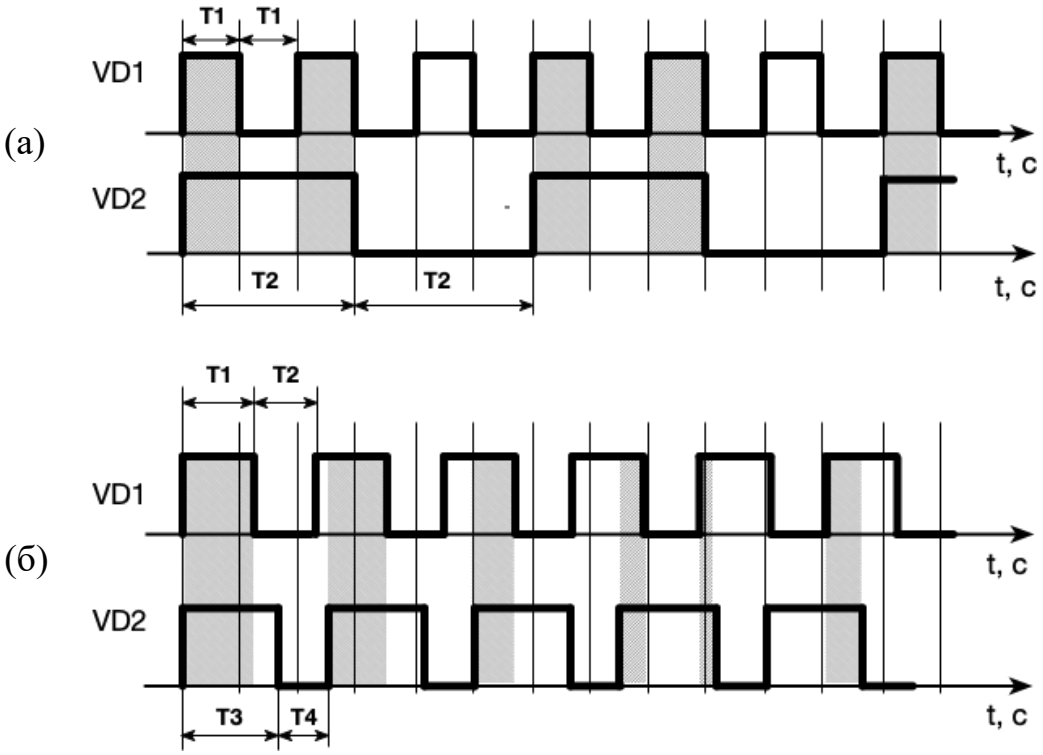

Рисунок 1.1 - Временная диаграмма синхронной (а) и асинхронной (б) коммутации светодиодов

Вопросы для самопроверки

1) Что такое ОС и в чем отличие между ОС мягкого и жесткого разделения времени?
2) Какие преимущества у ОСРВ при применения с МК?
3) Какие недостатки использования ОСРВ для МК?
4) Что такое переключение контекста задач планировщиком и для чего это необходимо в FreeRTOS?
5) Что такое базовый тип данных в FreeRTOS и для чего он был введен в данную ОСРВ?
6) Какие состояния задачи существуют в FreeRTOS, для чего необходимы эти состояния и каким образом задача переходит из одного состояния в другое.
7) Что означают параметры у API-функции, которая используется для создания задач в FreeRTOS?
8) Что означает константное значение configMINIMAL_STACK_SIZE?

ЗАДАНИЕ № 2. Исследование возможностей параметризации задач ОСВР FreeRTOS

Цели: 1) изучить принципы оптимизации программного кода, при применении задач ОСРВ FreeRTOS;

2) изучить методику передачи и получения параметров в задачу ОСРВ FreeRTOS.

Таблица № 2.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 1
Подключаемые библиотеки	• Arduino_FreeRTOS.h • avr/io.h • util/delay.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelete(NULL) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE
Ограничения	• нельзя использовать глобальные переменные • приоритеты задач одинаковые, например, равные 1
На что обратить внимание	• как определяются структуры в языке программирования Си • как определяются пользовательские переменные типа структуры • передача и получения параметров в функцию по ссылке • приведение типов

Описание

Реализовать коммутацию всех четырех светодиодов, подключенных к выводам микроконтроллера. В программе должна быть одна задача, которая будет получать параметр типа структуры, поля которой приведены ниже:

```
user_structure
{
        // номер вывода
        type_field pin_number;

        // время в отключенном или включенном
        // состоянии
        type_field period;
}
```

Время коммутации каждого из четырёх светодиодов разное. Может быть синхронным или асинхронным.

Вопросы для самопроверки

1) Как выполняется передача параметра в задачу и восстановление параметра в задаче в FreeRTOS (покажите на примере передачи структуры)?
2) Почему параметр задачи имеет тип void*?
3) В чём отличие операций доступа к полям структуры . («точка») и -> («стрелка»)?
4) Как можно реализовать операцию доступа к полям структуры -> («стрелка») через совместное использование операций . («точка») и приведение типа?

ЗАДАНИЕ № 3. Исследование API-функции задержки ОСВР FreeRTOS

Цель: изучить принципы организации временной задержки в ОСРВ FreeRTOS.

Таблица № 3.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 1
Подключаемые библиотеки	• Arduino_FreeRTOS.h • avr/io.h • util/delay.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • vTaskDelete(NULL) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE • portTICK_PERIOD_MS
Ограничения	• нельзя использовать глобальные переменные • приоритеты задач одинаковые, например, равные 1
На что обратить внимание	• как определяются структуры в языке программирования Си • как определяются пользовательские переменные типа структуры • передача и получения параметров в функцию по ссылке • приведение типов

Описание

1) Реализовать коммутацию любых двух светодиодов, подключенных к выводам микроконтроллера. Временную задержку включения и выключения светодиодов реализовать в виде цикла или при помощи функции стандартной библиотеки AVR - _delay_ms(...), при равных приоритетах задач.
2) Установить для одной из задач приоритет выше, чем у другой и проследить как поменяется работа микропроцессорной системы.
3) Реализовать коммутацию двух светодиодов, подключенных к выводам микроконтроллера. Временную задержку включения и выключения светодиодов реализовать через API-функцию vTaskDelay(...), при равных приоритетах задач.
4) Установить одной из задач приоритет выше, чем у другой и проследить как поменяется работа микропроцессорной системы.

Вопросы для самопроверки

1) Объясните что происходить с вызовом задач при реализации временной задержки через цикл?
2) Объясните что происходить с вызовом задач при реализации временной задержки через API-функцию vTaskDelay(...)?
3) Какой вариант реализации задержки эффективней с точки зрения расхода процессорного времени микропроцессорной системы и почему?
4) Для чего используется константное значение portTICK_PERIOD_MS?
5) Что делает API-функция vTaskDelete(NULL)и почему ей передается в качестве параметра значение NULL?

ЗАДАНИЕ № 4. Исследование API-функций ОСРВ FreeRTOS для организации выполнения прикладной задачи во время "Бездействия" МК

Цель: изучить принципы организации программы для выполнения прикладной задачи во время «Бездействия» микроконтроллера в ОСРВ FreeRTOS.

Таблица № 4.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 1
Подключаемые библиотеки	• Arduino_FreeRTOS.h • avr/io.h • util/delay.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • vTaskDelete(NULL) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE • portTICK_PERIOD_MS • vApplicationIdleHook(void)
Ограничения	• нельзя использовать глобальные переменные

Описание

1) Реализовать коммутацию любого одного светодиода в виде прикладной задачи ОСРВ FreeRTOS. Для лучшего наблюдения за

процессом работы микропроцессорной системы рекомендуется время включения и выключения светодиода выбрать в пределах 0,5 секунды.

2) Реализовать «перехват» бездействия микроконтроллера, во время которого переодически коммутировать любой другой свободный светодиод.

3) В начале реализовать задержку в задаче в виде цикла или функции стандартной библиотеки AVR _delay_ms(...), затем - через API-функцию vTaskDelay(...) и по наблюдать за работой системы.

4) Сделать выводы.

Вопросы для самопроверки

1) Что такое задача «Бездействия» в ОСРВ FreeRTOS?

2) Какой приоритет имеет задача «Бездействия» в ОСРВ FreeRTOS и может ли прикладная задача иметь приоритет выше, ниже или равный задаче «Бездействия» в ОСРВ FreeRTOS?

3) Как реализуется перехват задачи «Бездействия» в ОСРВ FreeRTOS?

4) Для каких практических целей может использоваться перехват задачи «Бездействия»?

5) Какие есть ограничения на использование задачи «Бездействия» в ОСРВ FreeRTOS?

ЗАДАНИЕ № 5. Исследование API-функций динамического управления состоянием задач в программе с ОСРВ FreeRTOS

Цель: изучить принципы динамического управления состоянием прикладных задач в программе с ОСРВ FreeRTOS.

Таблица № 5.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 1
Подключаемые библиотеки	• Arduino_FreeRTOS.h • avr/io.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • vTaskDelete(NULL) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE • portTICK_PERIOD_MS • UBaseType_t uxTaskPriorityGet(TaskHandle_t xTask) • vTaskPrioritySet(TaskHandle_t xTask, UBaseType_t uxNewPriority)

Описание

1) Реализовать коммутацию двух любых светодиодов в виде отдельных (то есть для простоты реализации без передачи параметров в задачи) прикладных задач ОСРВ FreeRTOS.
2) Сохранить дескрипторы (ссылки) на созданные прикладные задачи в виде глобальных переменных.
3) Задать приоритет одной прикладной задачи выше, чем у другой.

4) Запустить программу на исполнение и по наблюдать за её работой.

5) Внести в программу изменения:

a. Добавить глобальные переменные, в которые записывать количество включений каждого из светодиодов в соответствующей прикладной задаче;

b. Когда количество включений светодиода в задаче с большим приоритетом будет равно, например, пяти, то выполнить динамическое, то есть во время выполнения программы на микроконтроллере, изменение приоритетов прикладных задач - задача с большим приоритетом должна получить меньший приоритет, а задача с меньшим приоритетом должна получить больший приоритет. То есть выполнить обмен приоритетами между двумя прикладными задачами. Для реализации динамического изменения приоритетов прикладных задач следует использовать соответствующие API-функции ОСРВ FreeRTOS:

- **UBaseType_t uxTaskPriorityGet(TaskHandle_t xTask)**
- **void vTaskPrioritySet(TaskHandle_t xTask, UBaseType_t uxNewPriority).**

6) Запустить программу на исполнение и по наблюдать за её работой.

Вопросы для самопроверки

1) Что из себя представляет дескриптор прикладной задачи и зачем он нужен?

2) Для каких целей используются API-функции ОСРВ FreeRTOS uxTaskPriorityGet(...) и vTaskPrioritySet(...)?

3) Могут ли одна задача уничтожить другую задачу и если может, то как это реализовать при помощи API-функций ОСРВ FreeRTOS?

4) Если ограничения на номера присваиваемых приоритетов и если есть, то какие?

5) Какое преимущество дает уничтожение неиспользуемых задач в FreeRTOS?

6) Какая память освобождается при удалении задачи в FreeRTOS?

7) Какие две базовые схемы выделения памяти применяются в FreeRTOS?

8) Почему существуют разные схемы выделения памяти в FreeRTOS?

9) Какой алгоритм используется для поиска незанятых блоков памяти в FreeRTOS и как работает этот алгоритм?

10) Какие типы многозадачностей реализованы в FreeRTOS?

11) Приведите диаграмму переключения между задачами для вытесняющей и кооперативной многозадачности?

ЗАДАНИЯ ТРЕТЬЕГО РАЗДЕЛА

ЗАДАНИЕ № 6. Исследование API-функций для работы с очередями в ОСРВ FreeRTOS

Цель: изучить принципы безопасной передачи сообщений между прикладными задачами в программе с ОСРВ FreeRTOS.

Таблица № 6.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO
Подключаемые библиотеки	• Arduino_FreeRTOS.h • queue.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • vTaskDelete(NULL) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE • QueueHandle_t xQueueCreate(UBaseType_t uxQueueLength, UBaseType_t uxItemSize) • BaseType_t xQueueSendToBack(QueueHandle_t xQueue, const void * pvItemToQueue, TickType_t xTicksToWait) • BaseType_t xQueueReceive(QueueHandle_t xQueue, void *pvBuffer,TickType_t xTicksToWait)

Описание

1) Создайте глобальную переменную для очереди типа QueueHandle_t.
2) Создайте очередь при помощи API-функции FreeRTOS xQueueCreate(...).
3) Создайте две прикладные задачи.
4) Одна прикладная задача будет загружать в очередь данные при помощи API-функции FreeRTOS xQueueSendToBack(...).
5) Вторая прикладная задача будет считывать данные из очереди при помощи API-функции FreeRTOS xQueueReceive(...).
6) Для отображения работы программы, задача, которая должна записывать данные в очередь может выполнять инкремент локальной целочисленной переменой и каждое новое значение записывать в очередь. Задача, которая должна считывать данные из очереди может выводить их в монитор порта для чего можно использовать статический класс из стандартной библиотеки Arduino - Serial.

Сложность заключается в выборе размера очереди, тайм-аутов и задержек. Все должно работать на приемлемой скорости без потери данных.

Вопросы для самопроверки

1) Для чего используются очереди в FreeRTOS?
2) Как организовано хранение информации в очередях FreeRTOS?
3) В каких случаях возникает блокировка при чтении и записи в очередь?
4) Какие API-функции используются для работы с очередями FreeRTOS?
5) Как организовать разбор сообщении в очереди FreeRTOS?

ЗАДАНИЕ № 7. Исследование API-функций для работы с двоичными семафорами в ОСРВ FreeRTOS

Цель: изучить принципы синхронизации прикладных задач при возникновении прерывания в ОСРВ FreeRTOS.

Таблица № 7.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 2
Подключаемые библиотеки	• Arduino_FreeRTOS.h • semphr.h • avr/interrupt.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • taskYIELD(...) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE • SemaphoreHandle_t xSemaphoreCreateBinaryStatic(StaticSemaphore_t *pxSemaphoreBuffer) • xSemaphoreGiveFromISR(SemaphoreHandle_t xSemaphore, signed BaseType_t *pxHigherPriorityTaskWoken) • xSemaphoreTake(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait)

Описание

1) Создайте глобальную переменную для двоичного семафора типа SemaphoreHandle_t.
2) Создайте двоичный семафор при помощи API-функции FreeRTOS xSemaphoreCreateBinaryStatic(...).
3) Создайте обработчик внешнего прерывания INT0 микроконтроллера Atmega328p.
4) В обработчике прерывания будет инкрементироваться глобальная переменная count после чего «выдавается» двоичный семафор API-функцией FreeRTOS xSemaphoreGiveFromISR(...).
5) Создайте две прикладные задачи.
6) Одна прикладная задача будет выводить строку «It is ordinary task» в монитор порта и иметь приоритет равный 1.
7) Вторая прикладная задача будет являться отложенными обработчиком прерывания и выводить данные из глобальной переменной count, но после того как будет «захвачен» двоичный семафор при помощи API-функции FreeRTOS xSemaphoreTake(...). Эта задача должна иметь приоритет равный 2.
8) Для отображения работы программы в мониторе порта можно использовать статический класс из стандартной библиотеки Arduino - Serial.

Вопросы для самопроверки

1) Что такое отложенная обработка прерываний в FreeRTOS и чем она может быть полезна?
2) Что такое двоичные семафоры и API-функции для работы с ними в FreeRTOS?
3) Как двоичные семафоры реализуют синхронизацию задач при доступе к ограниченным ресурсам (механизм работы двоичных семафоров)?
4) Для чего используется API-функция taskYIELD(...)?

ЗАДАНИЕ № 8. Исследование API-функций для работы со счетными семафорами в ОСРВ FreeRTOS

Цель: изучить принципы синхронизации прикладных задач при возникновении прерывания в ОСРВ FreeRTOS.

Таблица № 8.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 2
Подключаемые библиотеки	• Arduino_FreeRTOS.h • semphr.h • avr/interrupt.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • taskYIELD(...) • vTaskStartScheduler() • configMINIMAL_STACK_SIZE • SemaphoreHandle_t xSemaphoreCreateCounting(UBaseType_t uxMaxCount, UBaseType_t uxInitialCount) • xSemaphoreGiveFromISR(SemaphoreHandle_t xSemaphore, signed BaseType_t *pxHigherPriorityTaskWoken) • xSemaphoreTake(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait)

Описание

1) Создайте глобальную переменную для счётного семафора типа SemaphoreHandle_t.
2) Создайте счётный семафор при помощи API-функции FreeRTOS xSemaphoreCreateCounting(...).
3) Создайте обработчик внешнего прерывания INT0 микроконтроллера Atmega328p.
4) В обработчике прерывания будет инкрементироваться глобальная переменная count после чего «выдавается» счётный семафор API-функцией FreeRTOS xSemaphoreGiveFromISR(...). Для того чтобы смоделировать множество быстро идущих друг за другом внешних прерываний можно несколько раз (например, три раза) продублировать вызов API-функции FreeRTOS xSemaphoreGiveFromISR(...), то есть несколько раз «выдать» семафор.
5) Создайте две прикладные задачи.
6) Одна прикладная задача будет выводить строку «It is ordinary task» в монитор порта и иметь приоритет равный 1.
7) Вторая прикладная задача будет являться отложенными обработчиком прерывания и выводить данные из глобальной переменной count, но после того как будет «захвачен» двоичный семафор при помощи API-функции FreeRTOS xSemaphoreTake(...). Эта задача должна иметь приоритет равный 2.
8) Для отображения работы программы в мониторе порта можно использовать статический класс из стандартной библиотеки Arduino - Serial.

Вопросы для самопроверки

1) Что такое отложенная обработка прерываний в FreeRTOS и чем она может быть полезна?
2) Чем отличаются счётные семафоры от двоичных семафоров?
3) Как создать в программе счётный семафор при помощи API-функций FreeRTOS?

ЗАДАНИЕ № 9. Исследование API-функций работы с критическими секциями и мьютексами в ОСРВ FreeRTOS

Цель: изучить механизмы взаимного исключения прикладных задач при доступе к ограниченным ресурсам микропроцессорной системы в ОСРВ FreeRTOS.

Таблица № 9.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO
Подключаемые библиотеки	• Arduino_FreeRTOS.h • semphr.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • vTaskDelay(...) • vTaskStartScheduler() • configUSE_MUTEXES 1 (в FreeRTOSConfig.h) • SemaphoreHandle_t xSemaphoreCreateMutex() • xSemaphoreTake(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait) • xSemaphoreGive(SemaphoreHandle_t xSemaphore)

Описание

1) Создайте глобальную переменную для мьютекса типа volatile SemaphoreHandle_t.

2) Создайте мьютекс при помощи API-функции FreeRTOS xSemaphoreCreateMutex(...).

3) Создайте две прикладные задачи.

4) Одна прикладная задача будет выводить по-символьно строку «**********» в монитор порта и иметь приоритет равный 1.

5) Вторая прикладная задача будет выводить по-символьно строку « - » в монитор порта и иметь приоритет равный 2.

6) Для отображения работы программы в мониторе порта можно использовать статический класс из стандартной библиотеки Arduino - Serial.

7) В начале выполните по-символьный вывод строк без добавления программного кода в прикладные задачи с «захватом» и «освобождением» мьютекса. Сделайте выводы.

8) Теперь добавьте в программный код прикладных задач «захват» и «освобождение» мьютекса. Сделайте выводы.

Вопросы для самопроверки

1) Что такое механизм взаимного исключения в FreeRTOS?
2) Какие способы используются для организации взаимного исключения в FreeRTOS?
3) Что такое неатомарные операции и реентерабельность функций в FreeRTOS (приведите примеры)?
4) Что такое критическая секция в FreeRTOS ?
5) Какие способы организации критических секций применяются в FreeRTOS?
6) Что такое мьютексы в FreeRTOS?
7) Какие API-функции применяются в FreeRTOS для работы с мьютексами?
8) Что такое рекурсивные мьютексы в FreeRTOS?
9) Какие API-функции применяются в FreeRTOS для работы с рекурсивными мьютексами?
10) Поясните проблемы возникают при работе с мьютексами в FreeRTOS?
11) Для каких целей может потребоваться API-функция vApplicationTickHook() FreeRTOS (поясните на примере)?
12) Что такое задача-сторож и как может быть использована задача-сторож в FreeRTOS?

ЗАДАНИЕ № 10. Исследование API-функций для работы с сопрограммами в ОСРВ FreeRTOS

Цели: 1) изучить организацию программы на языке программирования Си, при использовании API-функций прикладных сопрограмм ОСРВ FreeRTOS;

2) изучить принцип псевдопараллельной работы прикладных сопрограмм в ОСРВ FreeRTOS на однокристальнном микроконтроллере Atmega328p.

Таблица № 10.1 Ресурсы для выполнения практической работы.

Название категории ресурса	**Ресурс**
Оборудование	• контроллер Arduino UNO • стенд № 1
Подключаемые библиотеки	• Arduino_FreeRTOS.h • croutine.h • avr/io.h
API-функции и константы ОСРВ FreeRTOS	• xTaskCreate(...) • configUSE_IDLE_ HOOK 1 (в FreeRTOSConfig.h) • configUSE_CO_ROUTINES 1 (в FreeRTOSConfig.h) • configMAX_CO_ROUTINE_PRIORITIES 3 (в FreeRTOSConfig.h) • vTaskStartScheduler() • BaseType_t xCoRoutineCreate(crCOROUTINE_CODE pxCoRoutineCode, UBaseType_t uxPriority, UBaseType_t uxIndex) • void crSTART(crCOROUTINE_CODE xHandle) • void crEND()

	• void crDELAY(CoRoutineHandle_t xHandle, TickType_t xTicksToDelay) • void vCoRoutineSchedule(void) • void vApplicationIdleHook(void)
Ограничения	• нельзя использовать глобальные переменные • приоритеты прикладные сопрограмм одинаковые, например, равные 1

Описание

1) Создайте две сопрограммы, каждая из которых будет выполнять периодическую коммутацию одного из светодиодов VD1 и VD4.
2) Каждая сопрограмма должна будет выполнить 3...5 коммутаций соответствующим светодиодом, затем перейти в блокированное состояние (API-функция crDELAY(...)).
3) Реализуйте правильный запуск сопрограмм используя API-функции ОСРВ FreeRTOS.
4) Запустите программу на исполнение и проанализируйте её выполнение.
5) Закомментируйте вызов API-функции crDELAY(...) внутри каждой задачи и запустите программу на исполнение. Проанализируйте её работу.

Вопросы для самопроверки

1) Что такое сопрограмма в FreeRTOS и какие у нее преимущества?
2) Какие существуют состояния у сопрограммы в FreeRTOS?
3) Как выполняются сопрограммы и как работают приоритеты у сопрограмм в FreeRTOS?
4) Как реализовать сопрограмму в FreeRTOS (покажите на примере)?
5) Какие существуют API-функции для создания и запуска сопрограмм в FreeRTOS, где и когда их нужно вызывать в программе?
6) Как необходимо настроить ОСРВ FreeRTOS для возможности использования сопрограмм?
7) Какие существуют ограничения при использовании сопрограмм в FreeRTOS?

Учебно-методическое и информационное обеспечение

1. Курниц А. FreeRTOS - операционная система для микроконтроллеров, Компоненты и технологии №2 - №10, 2011.
2. Описание функций ОСРВ FreeRTOS https://www.freertos.org/a00106.html
3. Керниган, Б.В., Ричи, Д.М. Язык программирования С: учебное пособие, М.: Интернет- Университет Информационных Технологий (ИНТУИТ), 2016.
4. Куль, Т.П. Операционные системы: учебное пособие, Минск: РИПО, 2015.
5. Солдатенко, И.С., Попов, И.В. Практическое введение в язык программирования Си: учебное пособие, Лань, 2018.
6. Фридман, А.Л. Язык программирования Си++: учебное пособие, М.: Интернет- Университет Информационных Технологий (ИНТУИТ), 2016.

Содержание