1, в чем отличие режимов работы планировщика с разделением времени и без.как часто запускается планировщик в этих двух режимах?

Предварительно:

В режиме с разд. Времени планировщик вызывается каждые 2 тика. Тогдв вопрос: как планировщик будет вызываться в конце выполнения задачи IST ,чтобы перейти к другой задаче?проверить :вызывается ли планировщик в конце выполнения задачи?

В конце выполнения задачи планировщик не вызывается.

ПРЕДВАРИТЕЛЬНО : ЕСЛИ ЗАДАЧИ РАВНОГО ПРИОРИТЕТА, То не ВЫЗЫВАЕТСЯ , если задачи разного, то да.

Первая вызвавшаяся задача будет бесконечно выполняться, проверено в первой и второй задаче. Так как

В режиме без разд. Времени вызывается в конце выполнения задачи и по вызову функции yeld . чтобы это проверить можно в 4 почмотреть, будет ли бесконечное выполнение однной задачи при отсутствии функциив taskdelay , потому что она ывзывает планмировщик (то есть быдет ли вызываться планировщик)?

Первая вызвавшаяся задача будет бесконечно выполняться, так как вызов планировщика не выполняется.

В обоих проверках можно в дебагере построчно отследить выполнение команд и посмотреть переходит ли выполнение к другой задаче в конце выполнения данной задачи ?

Также вопрос: если внутри задачи была вызвана функция блокировки на заданное время, то после ее отработки время перед переходом ко 2 задаче обновляется и становится снова равным 2 тикам ?

Доля проверки нужно после задержки вставить цикл , который будет выполняться гарантированно больше 2 тиков и отследить в дебагере переход к выполнению другой задачи.

2, что именно мы настраиваем при перелючении режимов с раздел. Времени и без ?

Что регулирует каждый из двух параметров ?

configUSE_PREEMPTION Значение configUSE_PREEMPTION, равное 1, дает предписание ядру FreeRTOS работать в режиме вытесняющей многозадачности

configUSE_TIME_SLICING с разделением времени или без

Вытесняющая многозадачность без разделения времени Ее идея заключается в том, что вызов планировщика происходит только в обработчиках прерываний. Задача выполняется до тех пор, пока не произойдет какое-либо прерывание. После чего она вытесняется задачей, ответственной за обработку внешнего события, связанного с этим прерыванием. Таким образом, задачи не сменяют друг друга по прошествии кванта времени, это происходит только по внешнему событию.

Сколько системных квантов времени уделяется задаче при выт. Многзад. при разделении времени? 2 или 1 ? 1

Через сколько тиков будет ли вызываться планировщик ? Для проверки в 4 лабе можно убрать в одной из задач блокировки и вызов планировщика , почмотреть через сколько времени вызовется вторая задача?

Для того, чтобы отследить момент последнего вызова планировщика , необходимо определить когда он возникает. Этот момент совпадает с моментом начала выполнения задачи, так как при каждом вызове планировщика происходит переход к другой задаче. Таким образом нужно поставить брейкпоинты на началала каждой задачи и отследить время между ними.

Время перехода от 2задачи к 1 1009 мкс, а переход от 1 ко 2 почему то происходит 1032.

Таким образом теперь можно однозначно сказать ,что время на выполнения равноприоритетных задач равно 1 тик.

При выполнении задач код перед бесконечным циклом выполняется только при первом вызове задачи, в дальнейшем от выполнится, только тогда когда задача закончится и вызовется снова. А так благодаря циклу задача крутиться бесконечно.

Доля проверки нужно после задержки вставить цикл , который будет выполняться гарантированно больше 2 тиков и отследить в дебагере переход к выполнению другой задачи.

3, Если задача, захватившая мьютекс пытается захватить его снова, акое значение вернет функция семафортэйк.

~~Предварительно: она вернет тру, т.к. данная задача уже захватила данный мьютекс, то для нее он как бы свободен. И к тому же в 4 лабораторной если не особождать мьютекс, то задача другая никогда не сможет егол захватить и данная задача будет выполняться бесконечною.~~

Проверку сделать так: в 4 лабе убрать освобождение мьютекса и проследить что происходит дальше.

При выполнении функции xSemaphoreTake она возвращает значение ложное, то есть мьютекс уже занят и задача не может повторно его захватить. То есть в конце работы с ресурсом мьютекс необходимо обязательно освобождать.

Сможет ли вторая задача захватить мьютекс. Предварительно: не сможет, так как сначала его должна освободить 2 задача. Да она действительно не может захватмить мьютекс.

4, как происхъодит захват семафора в задаче IST ?как работает функция захвата ?

Нужно ли освобождать семафор. Потому что по логике семафора : если его освободит,. то задача будет выполняться бесконечно, ~~хотя она будет выполняться бесконечно если его и не освобождать. Ведь данный семафор теперь всегда будет у этой задачи. Так как она единственная кто работает с ним помимо ISR~~ . На самом деле из предыдущего опыта стало известно, что повторно захватить семафор, который уже захвачен, одна и та же задача не может. Сначала нужно его освободить. Таким образом, чтобы обработчик перывания не работал, освобождать семафор запрещено.

5, как организовать использование переменной одной задачей, если эта переменная была изменена в другой задаче ? Как передавать через очередь команду первой задаче о переключении варианта визуализации ?

6, как настроить вызов функции ISR по прерыванию на ножке МК.

Прерывание

Для организации прерывания используется гибридная многозадачность во FreeRTOS

Гибридная многозадачность сочетает в себе автоматический вызов планировщика каждый квант времени, а также возможность принудительного, явного вызова планировщика. Полезной гибридная многозадачность может оказаться, когда необходимо сократить время реакции системы на прерывание. В этом случае в конце тела обработчика прерывания производят вызов планировщика, что приводит к переключению на задачу, ожидающую наступления этого прерывания.

Какого-либо специального действия для включения режима гибридной многозадачности не существует. Достаточно разрешить вызов планировщика каждый квант времени (макроопределение configUSE_PREEMPTION в файле FreeRTOSConfig.h должно быть равным 1) и в явном виде вызывать планировщик в функциях, реализующих задачи, и в обработчиках прерываний с помощью API-функций taskYIELD() и portYIELD_ FROM_ISR() соответственно.

Вызывает ли планировщик функция vTaskDelay

Задача сразу же блокируется, планировщик отдаст управление другой задаче, а вызывающая задача вернется в состояние готовности к выполнению, только когда счетчик квантов достигнет нужного значения.