# FreeRTOS Имплементации

# Мутекс

Мутекс е специјален тип на бинарен семафор каде што се користи за менаџирање на ресурси помеѓу таскови. Во мултитаскинг системите, постои голема можност повеќе од еден таск да користат еден ресурс за да го завршат нивното извршување. Но, споделувањето единствен ресурс помеѓу повеќе таскови може да предизвика повеќекратни грешки како што се ќор-сокак, оштетување на податоците, инверзија на приоритети итн. Мутексот ни овозможува да не настанат вакви грешки додека еден таск има престап до одредени информации.

# Код

Креирање на мутекс.

A picture containing text, font, screenshot, white

Description automatically generated

Креирање на два таска со различни приоритети. Вториот таск има поголем приоритет од првиот и кога скеџулер-от ќе започне тој ке биде прв што ќе се испринта.

Повеќе објаснување за аргументите на xTaskCreate.

Првиот аргумент – Покажувач то таск функцијата

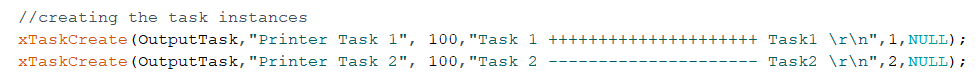
Вториот аргумент – Дескриптивно име на таскот

Третиот аргумент – Колку зборови да алоцираме од стекот

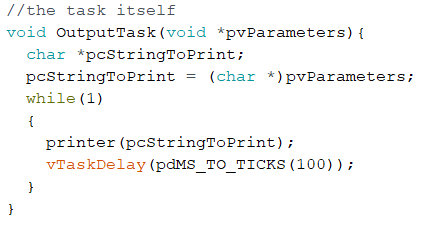
Четвртиот аргумент – Стрингот што сакаме да го испечатиме

Петтиот аргумент – Приоритетот на таскот

Последниот аргумент – Овој аргумент е опционален и може да е NULL



Тука го праќаме стрингот и потоа тој стринг се праќа до Printer функцијата. Кога ќе пратиме чекаме 100мс за да токенот биде вратен. Ова праќање се извршува бесконечно број пати меѓу двата таска при што имаме наизменично печатење .



Во Printer функцијата го употребуваме мутексот. Прво проверуваме дали токенот е слободен (xSemaphoreTake). Исто така имаме ставено време колку да чекаме доколку тој не е слободен. Во овој случај чекаме колку што е максимално возможно(portMAX\_DELAY). Кога ќе го земеме токенот ние на Serial Monitor го печатиме стрингот. На крај го даваме токенот на мутексот за да можат дригу таскови да добијат пристап то ресурсот(xSemaphoreGive).

A picture containing text, font, screenshot

Description automatically generated

Можеме да видиме дека стринговите не се корумпирани при печатење иако имаме два таска со различни приоритети што значи таскот со поголем приоритет не му го одзима времето на извршување на таскот со помал приоритет.

A screenshot of a computer program

Description automatically generated with medium confidence

# Користење на прекини за читање и запишување на информации од редица

Прекините имаат повисоки приоритети од другите таскови. Затоа, тие не треба да чека за мутекс, семафор и други ресурси и треба да се изврши веднаш штом ќе се појави. Одложување на обработката на прекините преку други задачи е можна контрамерка за минимизирање на времето за обработка на ISR што е можно поскоро. Накратко, кодот и времето на извршување на рутината на услугата за прекин треба да бидат што е можно помали.

# Код

Во овој код генерирам оверфлоу на timer1. Секој пат кога ке настане тој оверфлоу се повикува ИСР и тука на редицата се запишуваат елементи и на крај принтер таскот ги печати тие елемнти на serial monitor.

Креирање на таск за печатење со референца до принтер функција, кеирање на редица со 5 алоцирани слотови, и започнување на прекини .

A picture containing text, font, screenshot

Description automatically generated

Функцијата за печатење. Овде бесконечно ќе ги чита поставените информации од редицата и ќе ги запишува на serial monitor.

A screen shot of a computer code

Description automatically generated with low confidence

Прекинска сервисна рутина ISR каде на секој оверфлоу на timer1 се повикува и запишува 5 елементи на редица.

A screenshot of a computer code

Description automatically generated with low confidence

Функцијата која што генерира оверфлоу за да ISR биде повикана. Без оваа функција ISR нема да биде повикана.

A picture containing text, screenshot, font, number

Description automatically generated

# Промена на периодата на часовникот за време на извршување

Во оперативните системи во реално време, ние користиме софтверски тајмери за извршување на некоја задача во одредена стапка со помош на функцијата за повратен повик. Стапката на повратен повик на извршување на функцијата се одредува според периодот на софтверскиот тајмер. Но, понекогаш, треба да го промениме периодот на софтверскиот тајмер за време на извршувањето или откако ќе започне FreeRTOS распоредувачот.

# Код

Во овој код прво се извршува тајмерот со најмалата периода. После 10 итерации периодата се дуплира и уште 10 итерации пак се дуплира и останвува со оваа периода бесконечно.

Дефинирање на периодите кој ќе ги користиме. Во овој код ние двапати ја дуплираме периодата така што на почетокот е 500 ticks а на крај 2000 ticks.

A picture containing text, font, screenshot

Description automatically generated

Креирање на тајмерот преку xTimerCreate функција. Повеќе информации за аргументите на оваа функција.

Првиот аргумент – дескриптивно име на тајмерот

Вториот аргумент – периодата на тајмерот

Третиор аргумент – сетирање на тајмерот та е од AutoReload тип. Ако ја пратиме вредност pdTRUE значи дека е AutoReload а pdFALSE значи е OneShot тајмер

Четвртиот аргумент – давање на идентификатор на тајмерот

Последниот аргумент – која функција ќе биде повикана кога ќе помине рокот на тајмерот

Проверуваме дали тајмерот е успешно креиран. Доколку е креиран започни го тајмерот и зачувај ја вредноста во xTimerStartedFlag (вредноста ќе биде TRUE ако тајмерот е започнат или FALSE ако имало некој проблем). Ако тајмерот е успешно започнат, започни го и распоредувачот.

A screen shot of a computer program

Description automatically generated with low confidence

Функцијата што се повикува кога ќе помине рокот на тајмерот. За да знаеме колку итерации поминале на извршување на тајмерот, ние го гледаме ИД-то, така што го зимаме на почетокот на итерацијата, го зголемуваме за 1, и го сетираме тоа да биде новото ИД. Бидејќи ИД-то е од VOID туп ние треба да го кастираме во int32 и потоа пак во VOID. Кога ќе поминат првите 10 итерации се пикува xTimerChangePeriod каде што кажуваме на кој тајмер периодата ќе ја смениме, која е новата периода и колку време да чекаме доколку timer command queue е полн. Истото го правиме и за наредните 10 итерации.

A screenshot of a computer code

Description automatically generated with medium confidence