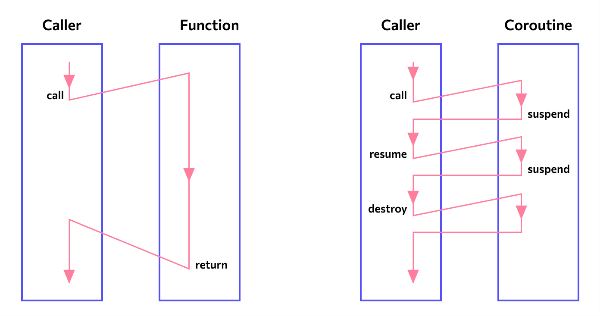
## Терминология

**Сопрограмма(coroutine)**- программные модули, которые работают асинхронно, то есть по очереди. В нужный момент исполнение такого модуля приостанавливается с сохранением всех его свойств, чтобы запустился другой. Когда управление возвращается к первому модулю, он продолжает работу. В результате программа выполняет несколько функций одновременно.

## Общие заметки

Задача (task=поток и т.п) может быть бесконечна.

## FreeRTOS

Создание минимального проекта [тут](https://www.freertos.org/Creating-a-new-FreeRTOS-project.html) и [тут](https://www.freertos.org/Hardware-independent-RTOS-example.html).

Создание задачи- TaskCreate(). При создании задачи в памяти создаются две сущности типа stack и TCB- task control block (содержит в себе информацию о задаче, по типу дескриптора и т.п.). Блокирование задачи (Blocked status), т.е. ее отложенную работу\работу через интервалы, можно реализовать через API функции TaskDelay() или TaskDelayUntil(). Прототип задачи простоя (если она не является фоновой, т.е. configUSE\_IDLE\_HOOK != 0) типа “ApplicationIdleHook()” можно использовать для получения информации о простое процессора, реализовав инкрементацию переменной в ее теле. Либо можно отправлять процессор в сон, например.

Запуск планировщика- TaskStartScheduler(). Алгоритм планирования выбирает задачу из пула задач имеющих статус Ready по приоритетам, по умолчанию планировщик работает на “упреждение” (configUSE\_PREEMPTION = 1), кроме того для задач с равными приоритетами используется разделение времени (configures\_TIME\_SLICING = 1).

Очереди используются для передачи данных между задачами и имеют определенный тип доступа\хранения. С их помощью возможно блокировать задачи, например, если нету команд\данных (типо паузы). Так же можно выстраивать последовательность команд, выполняемых задачей (например, направления движения). Очереди можно собирать в группы. Как вариант очередь можно формировать из указателей (для экономии выделяемой памяти, однако необходимо правильно с ними работать).

Возможность организации программных таймеров. Можно организовать перезапуск таймера из задачи, например, однократно пере включая подсветку (на заданное время) при нажатии на клавишу.

Про память. Для работы с динамическим выделением памяти используются, в том числе, ф-ии PortMalloc() и PortFree(). Ядро ОС поддерживает несколько моделей памяти типа heap\_1.c, heap\_2.c и т.п. Подробнее [тут](https://www.freertos.org/a00111.html).

Для работы вне ОС, в прерываниях например, нежелательно пользоваться стандартными функциями API. Лучше использовать специальные, по типу тех, что имеют окончание …ISR()

Семафор (подробнее в semphr.h) можно рассматривать как очередь длинной в один элемент, т.е. семафор может блокировать и запускать задачу. Его преимущество в экономии памяти и малом кол-ве функций API. Семафор снимается автоматически после вызова ф-ии xSemaphoreTake() в теле задачи. Возможен вариант организации счетного семафора xSemaphoreCreateCounting(), т.е. можно задать необходимое кол-во итераций задачи (как пример- управление шаговым двигателем).

События (подробнее в event\_groups.h) объединяют флаги в группы. Задача может быть настроена на обработку конкретного флага события.   
*На основе событий построена OSAL от Ti, там последовательно (по приоритетам), ОС проверяет группы событий задачи (простое 16и битное число), если флаг есть, то передает управление задаче, если флага нету, то проверяет следующую группу. Однако за один квант времени может быть выполнена лишь одна задача (возможно может быть выполнена следующая задача с более низким приоритетом, не проверял). Очереди для передачи данных/команд так же организованы через события (точнее есть зарезервированный флаг на событие получения сообщения.. от одной задачи для задачи, что то вроде нотификации)*

Нотификация. Функции по типу TaskNotify…(). Работает через TCB.

Касаемо сборки проекта. Отдельное внимание стоит уделить используемым библиотекам и файлам по типу startup\_xxx.s приоритеты и обработчики прерываний в разных библиотеках определены по-разному. Например, при использовании “stm32f10x.h” (большая) + startup\_stm32f10x\_md.s необходимо в FreeRTOSConfig.h определить макросы:  
#define vPortSVCHandler SVC\_Handler

#define xPortPendSVHandler PendSV\_Handler

#define xPortSysTickHandler SysTick\_Handler  
Про тактирование, которое может быть где то сконфигурировано, тоже забывать не стоит