**Создание нового проекта FreeRTOS.**

**Введение.**

FreeRTOS построен таким образом, что его просто понять и легко использовать. Требуется только 3 файла исходного кода, которые являются общими для всех RTOS портов, и один аппаратно-зависимый файл, а также простой и интуитивный набор API.

FreeRTOS портируется на большое количество микроконтроллерных архитектур, и большое количество компиляторов. Каждый официальный порт постовляется с официальным демо-проектом, который (по крайней мере на момент его создания) компилируется и выполняется на аппаратной платформе, для которой он был разработан без какой либо модификации.

Демопроекты предоставляются для того, чтобы убедить пользователей начать работать с FreeRTOS за максимально быстрое время и с минимальными затратами.

Каждая архитектура, поддерживаемая FreeRTOS используется во многих различных микроконтроллерах, что означает, что FreeRTOS может быть использована буквально с тысячами различных микроконтроллеров. Умножаем это число на число поддерживаемых компиляторов, и затем еще раз умножаем на постоянно увеличивающееся число стартер китов и оценочных плат, которые поступают на рынок, и очевидно, что несмотря на наши максимальные усилия, мы можем только обеспечить официальные демонстрационные проекты, которые точно соответствует крошечной части от возможных комбинаций.

Рекомендуется новый FreeRTOS проект всегда создавать, начиная с одного из предварительно сконфигурированного демо-проекта, а затем его адаптировать. Выполнение этих рекомендаций гарантирует, что новый проект включает в себя все необходимые файлы исходного кода и заголовочные файлы, и установлены все необходимые обработчики прерываний, без усилий со стороны создателей проекта.

Некоторые FreeRTOS пользователи также хотят знать, как создавать FreeRTOS проекты способами, отличными от адаптации существующих проектов. Процедура, как это сделать, документирована ниже.

**Анатомия FreeRTOS проекта.**

FreeRTOS приложение стартует и выполняется также как не-RTOS приложение, до тех пор, пока не будет вызван vTaskStartScheduler(). Эта функция обычно вызывается из функции main() приложения. RTOS управляет последовательностью исполнения только после того, как будет вызван vTaskStartScheduler().

Настоятельно рекомендуется убедиться, что код выполняется корректно (корректный код старт-апа, корректно сконфигурирован линкер и т.д.) на выбранной платформе, прежде чем пытаться использовать какую либо функциональность RTOS.

**Файлы исходного кода.**

FreeRTOS представлен стандартными файлами исходного кода на C, которые собираются вместе с другими C-файлами в вашем проекте. Файлы исходного кода FreeRTOS распространяются в zip архиве. Страница организации исходного кода FreeRTOS описывает структуру файлов в zip архиве. Как минимум, следующие файлы исходного кода должны быть включены в ваш проект:

* FreeRTOS/Source/tasks.c
* FreeRTOS/Source/queue.c
* FreeRTOS/Source/list.c
* FreeRTOS/Source/portable/[compiler]/[architecture]/port.c.
* FreeRTOS/Source/portable/MemMang/heap\_x.c where 'x' is 1, 2, 3 or 4.

Если директория, которая содержит port.c файл также содержит файл исходного кода на ассемблере, то этот файл также должен быть использован.

Если вам требуется функциональность программного таймера, то включите FreeRTOS/Source/timers.c.

Если вам требуется функциональность сопрограммы, то включите файл FreeRTOS/Source/croutine.c.

**Заголовочные файлы.**

Как минимум, следующие директории должны быть добавлены в пути поиска заголовков компилятора (компилятору необходимо сообщить, чтобы он искал заголовочные файлы в этих директориях):

* FreeRTOS/Source/include
* FreeRTOS/Source/portable/[compiler]/[architecture].

В зависимости от порта, может потребоваться, чтобы эти же директории были прописаны в путях ассемблеров.

**Конфигурационный файл.**

Каждый проект также требует так называемый конфигурационный файл - FreeRTOSConfig.h. Этот файл адаптирует ядро RTOS для собираемого приложения. Таким образом, он специфичен для приложения, не для RTOS, и должен быть размещен в директории приложения, и ни в одном из каталогов исходного кода ядра RTOS.

Если heap\_1, heap\_2 или heap\_4 включены в ваш проект, то определение в файле FreeRTOSConfig.h configTOTAL\_HEAP\_SIZE будет определять размер кучи FreeRTOS. Ваше приложение не будет линковаться, если размер кучи установлен слишком большим.

Определение в файле FreeRTOSConfig.h configMINIMAL\_STACK\_SIZE устанавливает размер стека, используемого холостой задачей. Если это значение установлено слишком низким, то холостая задача произведет переполнение стека. Рекомендуется скопировать значение минимального размера стека из официального демо проекта FreeRTOS, предоставляемого для той же архитектуры микроконтроллера. Демо проекты FreeRTOS хранятся в подкаталогах директории FreeRTOS/Demo. Заметьте, что некоторые демо проекты устарели, и не содержат всех доступных опций конфигурации.

**Векторы прерываний.**

Cortex-M пользователям: информация об установке обработчиков прерываний представлена в разделе FAQ **"[The application I created compiles, but does not run](http://www.freertos.org/FAQHelp.html)".**

Каждый RTOS порт использует таймер для генерации периодических тик-прерываний. Множество портов используют дополнительные прерывания для управления переключением контекста. Прерывания, требуемые RTOS портом, обслуживаются файлами исходного кода RTOS.

Метод, используемый для установки обработчиков прерываний обеспечиваемый портом RTOS зависит от порта и используемого компилятора. Обратитесь к (и если необходимо скопируйте) предоставленным официальным демо-проектам для используемого порта. Также обратитесь к странице документации, на которой представлены официельные демо-проекты.