## FreeRTOS порт PIC32MZ

**ISR\_Support.h** содержит описание функций на ассемблере сохранения и восстановления контекста, включая FPU регистры:

* portSAVE\_FPU\_REGS
* portLOAD\_FPU\_REGS
* portSAVE\_CONTEXT
* portRESTORE\_CONTEXT

**port.c** содержит описание базовых функций запуска/остановки диспетчера, а также инициализации стека:

* xPortStartScheduler. Настраивает прерывание от таймера и запускает первую задачу из очереди.
* vPortEndScheduler
* pxPortInitialiseStack

В целом, функции, которые нужно переопределить описаны в файле portable.h в исходных кодах FreeRTOS.

**portmacro.h** содержит функции включения/выключения прерываний, входа/выхода из критической секции:

* portDISABLE\_INTERRUPTS
* portENABLE\_INTERRUPTS
* portENTER\_CRITICAL
* portEXIT\_CRITICAL
* portYIELD

Определение configASSERT находится в FreeRTOSConfig.h. Этот файл надо добавить в директорию с исходниками Demo.

Выделить основные функции и затем их переписать. Все необходимое появится, само собой. Ненужное не повторится.

*Через какое время вызывается прерывание от таймера?*

Обработчик прерывания от таймера называется vPortTickInterruptHandler, описан в **port\_asm.S** . Он вызывает функцию vPortIncrementTick (**port.c**), которая вызывает уже платформонезависимую функцию FreeRTOS xTaskIncrementTick.

Функция vPortYieldISR привязана к программному прерыванию номер 0. Вызывается в функции portYIELD.

Замены:

* \_CP0\_STATUS на C0\_SR
* \_CP0\_CAUSE на C0\_CR
* \_CP0\_EPC на C0\_EPC

AC0, AC1, AC2, AC3 registers from the DSP - мы не будем использовать DSP расширение ЦОС (MIPS DSP Application-Specific Extension (ASE)), поэтому не нужно сохранять регистры аккумуляторы для данной подсистемы.