Interrupts and FreeRTOS

Date	@November 11, 2024

Hardware interrupts

Hardware interrupts are an important part of many embedded systems because they allow events to occur asynchronously and notify the CPU that it should take some action. These types of interrupts can cause the CPU to stop whatever it was doing and execute some other function, known as an interrupt service routine (ISR).

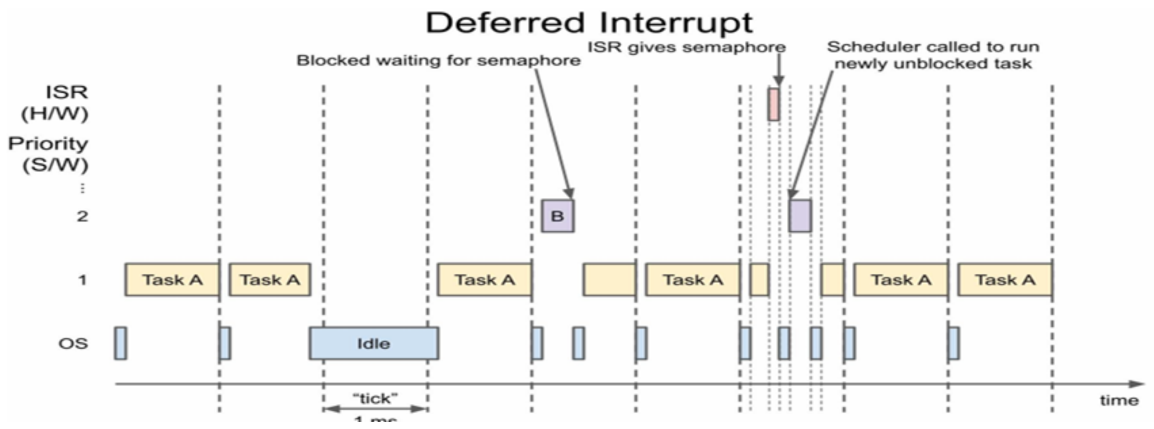
These hardware interrupts can include things like button presses, a hardware timer expiring, or a communication buffer being filled.

In FreeRTOS hardware interrupts have a higher priority than any task unless we disable them. When working with hardware interrupts, there are few things to keep in mind:

- an ISR should never block itself; they are not executed as part of a task, and as a result, cannot be blocked. You must only use FreeRTOS function calls that end in *FormISR inside an ISR and you cannot wait for a quque, mutex or semaphore
- you must keep ISRs as short as possible to avoid delays in your waiting tasks
- if a variable is updated inside an ISR, you likely need to declare it with the volatile qualifier. Marking a variable as volatile informs the compiler that it can change outside the current thread, preventing the compiler from optimizing it away due to assumed inactivity
- one of the easiest way to synchronize a task to an ISR is to use what's known as a deferred interrupt, in which we defer processing the data captured inside the ISR to another task. Whenever this data has been captured, we can give a semaphore to let some other task know that data is ready for processing

Deferred interrupts

Task B is blocked waiting for a semaphore. Only the ISR gives the semaphore. So, as soon as the ISR runs (and, say, collects some data from a sensor), it gives the semaphore. When the ISR has finished executing, Task B is immediately unblocked and runs to process the newly collected data.



ARM MPS2

the ARM MPS2 (Microprocessor System 2) platform is a development board designed by ARM to support the evaluation and prototyping of ARM-based systems. It serves as a flexible, scalable platform for testing ARM's Cortex-M and Cortex-A series processors. This platform typically provides hardware for software development, debugging and performance evaluation.

CMSIS

Common microcontroller software interface standard enables consistent device support and simple software interfaces to the processor and its peripherals, simplifying software reuse, reducing the learning curve for microcontroller developers, and reducing the time to market for new device.

MPS2 available interrupts

The list of available interrupts depends on the target board and is usually specified in the datasheet of the board

ISR handlers

In a FreeRTOS project, the `isr_vector[]` is not actually defined by FreeRTOS itself. It is typically part of the startup code provided by the compiler toolchain or device specific libraries. The interrupt vector table is set up by the microcontroller's startup file and is usually in assembly or C, depending on the toolchain and microcontroller.