**Scheduling** is done by time slicing which will be represented as a tick.

**Pre-emptive Scheduling** tasks are executed as per priority and they use the tick as delay not the hardware/processor interrupts.

**Task on different core** in case if processor runs the task on different cores it can run the tasks in parallel

**portTICK\_PERIOD\_MS** it is defined in FREERTOS the tick timer to 1ms as portTICK\_PERIOD\_MS 1 and we use vTASKDelay which accepts ticks and by dividing the desired period by portTICK\_PERIOD\_MS will give the numver of ticks.

**Diff** To do this on the ESP32, we use the xTaskCreatePinnedToCore() function instead and specify which core to use. If you are using vanilla FreeRTOS in your own build system, you will want to use xTaskCreate() instead.

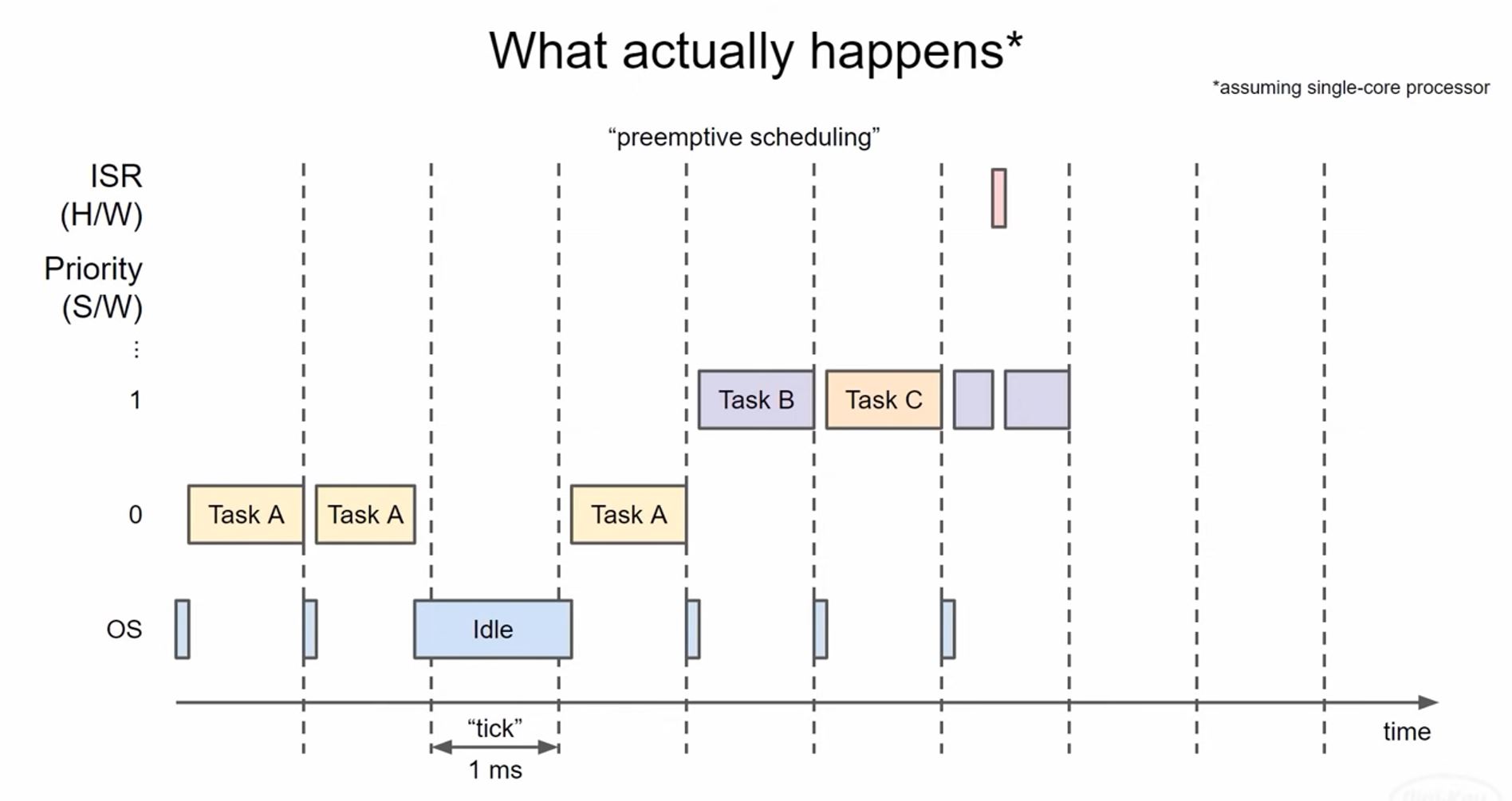
**A task** in FreeRTOS is similar to a thread in other multi-threaded environments (e.g. POSIX). It is a unit of CPU utilization designed to accomplish some goal. For our purposes, we just want to create a new thread and toggle an LED.

**Time slicing** slice is called a tick

Round robin fashion the task with same priority will be executed.

Pre-emptive scheduling cpu time takes from Task A to Task B then Task C which has higher priority then Task A.

If ISR is called



ESP IDF scheduler can perform Task B & C on other cores at the same time performing Task A on the other core.

**Context** as the tasks executes it use memory RAM ROM and all the related information

**Context Switching** saving the task and restoring the another task context.