## ### Context Switching

\*\*1. ISRs vs. Tasks:\*\*

- \*\*ISRs (Interrupt Service Routines):\*\* Brought into context by the hardware in the MCU (Microcontroller Unit).

- \*\*Tasks:\*\* Brought into context by the FreeRTOS kernel.

\*\*2. Interrupt Configuration:\*\*

- Hardware generates interrupts.

- Different configurations and masking options exist for interrupts.

## ### Execution Behavior

\*\*1. Task Execution:\*\*

- Tasks are typically set up like an infinite while loop.

- Synchronized with the system using primitives like queues and semaphores.

- Switched into context based on priority.

\*\*2. ISR Execution:\*\*

- ISRs should exit as quickly as possible to keep the system responsive.

- Fast exit prevents missing other interrupts by avoiding CPU hogging.

## ### Function Parameters

\*\*1. Task Functions:\*\*

- Tasks can take input parameters.

\*\*2. ISR Functions:\*\*

- ISR functions do not take input parameters.

- ISRs respond to hardware states by reading memory-mapped registers and performing actions accordingly.

- Example: An ISR for UART might read a status register, store received data in a static variable, and clear the interrupt.

## ### FreeRTOS API Access

\*\*1. ISR-Specific API Access:\*\*

- ISRs can only access a limited subset of the FreeRTOS API.

- The API for ISRs includes functions for operating on queues but with limited functionality (e.g., no blocking—calls return immediately).

- This limitation enhances safety by preventing blocking calls inside ISRs, which could cause system issues.

\*\*2. Error Handling:\*\*

- Calling a non-ISR API function from within an ISR will cause FreeRTOS to trigger `configASSERT`.

## ### Independent Operation

\*\*1. ISR Independence from RTOS:\*\*

- Some ISRs operate at a low level, completely independent of the RTOS.

- Such ISRs execute as they would in a system without an RTOS, with no kernel involvement and no task interruptions.

- This approach allows for blending high-performance ISRs with RTOS-managed tasks, offering flexibility in design.

## ### Stack Management

\*\*1. System Stack for ISRs:\*\*

- All ISRs share the same system stack.

- When writing ISRs, ensure sufficient stack space is reserved for potentially nested ISRs running simultaneously.

\*\*2. Dedicated Stack for Tasks:\*\*

- Each FreeRTOS task has its own dedicated stack, separate from the shared system stack used by ISRs.