What is the use of Stack Size in RTOS

Stack Size and Task Needs:

The stack is used by a task to store:

- **Function Call Information**: Each function call uses some stack space for parameters, local variables, and return addresses.
- **Task-specific Data**: Any local variables or temporary data within the task function.
- **Function Call Depth**: The more functions you call (especially recursively), the more stack space is needed.

Calculating Stack Size:

1. **Determine Task Requirements**:
    - **Function Calls**: Analyse how many levels of function calls are used.
    - **Local Variables**: Consider the size of local variables and arrays.
2. **Test and Monitor**:
    - **Empirical Testing**: Often, stack size is determined empirically by testing. You start with a reasonable estimate and adjust based on observed behavior. If a task causes a stack overflow, you may need to increase the stack size.

Example Scenario:

If you choose 128 words and each word is 1 byte:

- **Stack Size**: 128 bytes.

If each word is 4 bytes (common on 32-bit processors):

- **Stack Size**: 128 words × 4 bytes/word = 512 bytes

RTOS Task Explaination: **xTaskCreate(ledblink, "blink", 128, NULL, 1, &ledblink_h)**

Explanation of Parameters:

1. **pvTaskCode**:
    - This is the function pointer to the task code.
    - Example: ledblink
2. **pcName**:
    - A name for the task (optional, for debugging purposes).
    - Example: "blink"

3. **usStackDepth**:
    - The stack size (in words, not bytes) for the task.
    - Example: 128
4. **pvParameters**:
    - A pointer to parameters passed to the task function (can be NULL if no parameters are needed).
    - Example: NULL
5. **uxPriority**:
    - The priority of the task, with 0 being the lowest. Higher numbers indicate higher priority.
    - Example: 1
6. **pxCreatedTask**:
    - A pointer to a variable where the task handle will be stored, allowing control of the task (e.g., suspend, delete).
    - Example: &ledblink_h