How does dual core work?

Using the code in the file, two functions shall work simultaneously

The function xTaskCreatePinnedToCore() in ESP32 (FreeRTOS) is categorized under **task management** functions in FreeRTOS. It is specifically used for **creating tasks and assigning them to a specific core** in the ESP32's dual-core processor.

**Function Prototype**

BaseType\_t xTaskCreatePinnedToCore(

TaskFunction\_t pvTaskCode,

const char \* const pcName,

const uint32\_t usStackDepth,

void \* const pvParameters,

UBaseType\_t uxPriority,

TaskHandle\_t \* const pxCreatedTask,

const BaseType\_t xCoreID

);

**Parameter Breakdown**

1. **pvTaskCode** – Pointer to the function that implements the task.
2. **pcName** – Name of the task (useful for debugging).
3. **usStackDepth** – Stack size allocated for the task (in words, not bytes).
4. **pvParameters** – Pointer to parameters passed to the task function.
5. **uxPriority** – Priority of the task.
6. **pxCreatedTask** – Pointer to store the task handle (optional, can be NULL).
7. **xCoreID** – Specifies the core to which the task should be pinned.
   * 0: Run on Core 0
   * 1: Run on Core 1
   * tskNO\_AFFINITY: Run on any available core

**Code explanatory example:**

void myTask(void \*pvParameters) {

while (1) {

printf("Running on core: %d\n", xPortGetCoreID());

vTaskDelay(pdMS\_TO\_TICKS(1000));

}

}

void app\_main() {

xTaskCreatePinnedToCore(

myTask, // Task function

"My Task", // Name of the task

2048, // Stack size in words

NULL, // Task parameters

1, // Priority

NULL, // Task handle (not needed)

0 // Run on Core 0

);

}