FreeRTOS debugging tips. **CSE/ECE 474** **Spring 2023**

Arduino Mega 2560 Stats:

- 8K bytes of SRAM (your working memory) (e.g. 4096 ints total)
- FreeRTOS takes up about ½ of your SRAM! [LINK]
- 256K bytes of Flash memory (your code and constants)

FreeRTOS Programming/Debugging tips:

- Most important tip: Start small, and make SMALL changes in between test runs of your code. You cannot debug code that crashes due to stack overflows etc. Make sure you can revert each change back to your working version.
- `Serial` calls should be used VERY SPARINGLY (but when your stacks are OK they work)
- Stack space is limited. Try to set the smallest stack space value for each task that works, but test each task's stack size separately. The default setting of 128 from FreeRTOS examples is not enough on the AtMega2560.
- FreeRTOS indicates crashes with the ON-BOARD LED, so use off-board LEDs for your blink task(s). FreeRTOS signals:
  - **SLOW BLINK = STACK OVERFLOW (at least one task stack is too small)**
  - **FAST BLINK = FreeRTOS is out of available memory (you've allocated too much RAM or stack space).**
- The above light signals are NOT PERFECT. You can have stack overflows which lead to flakey behavior but are not detected by FreeRTOS.
- Even **one or two flashes** of the on-board LED at startup can indicate stack problems.
- Try this trick to stagger the startups of your tasks: place

  ```
  vTaskDelay(d/portTICK_PERIOD_MS);
  ```

  as the first line of each task function. Use a different `d` for each task from the list of {0,50,100,150….}. This might spread out the "pain" of tasks starting up and use less stack depth overall.
- When programming tasks remember:
  - `int d[100] = {0};` allocates 200 bytes ON THE STACK
  - `static int d[100] = {0};` allocates 200 bytes in your SRAM
  - BOTH of these are part of your 8k total, however.
- Use the `uxTaskGetStackHighWaterMark(NULL)` function to see how much stack depth you may be "wasting" by setting stack size too high. Allow some safety margin though. If the High Water Mark (HWM) is 0 you are too close to the edge. Maybe shoot for a margin of 50. [LINK] is an example blink task you can try this with.
- Be sure to experiment with task **priorities**. We've seen glitchy operation in some scenarios when all tasks have low or the same priority. Use the full range 0(low)-3(high).
- Retest your code after the initial upload by pressing the reset button. This verifies your exact startup behavior which may be obscured by the end of the upload process.
- Try `delay(500);` in between declaring your tasks and before starting up the scheduler.

CSE/ECE 474 ArduinoMEGA Lab 4 Specific advice:

- The big challenge is managing memory layout and stack space. Memory use depends strongly on the size of the FFT you will perform. We suggest the following procedure:
    a. Start with a couple of blink tasks or other very simple tasks.
    b. Using the technique of the linked code above, explore your stack size setting so that your blink tasks have a safety margin ("HighWaterMark" / HWM) of 30-60 or about 10% of your stack size setting.
    c. Start with 500 stack size and work your way down. The default 128 is not big enough. More complex tasks should start even higher, like 1500.
    d. Units of HWM are words, which are 2 bytes in our processor. You are also using words when you set the stack size in TaskCreate().
    e. Now add a basic FFT task calling the library. Since this is a more complex task, set the FFT size (`samples`) to 64. FFT size must be a power of two.
    f. Start FFT Task stack size at least 1500. Work your way down. The recommended FFT library uses doubles to store input and output. These take up a lot of SRAM for big FFT size so do not declare more than the minimum needed (one array for Real and one array for Imag).
    g. Finally, you can try increasing the FFT size. FFT sample# has to be a power of two so 64-->128-->256-->512.
    h. The system seems to break at a stack size of 256. The symptom is a rapidly flashing on-board LED. This is FreeRTOS signal for out of SRAM. Minimum FFT size for the assignment is 128 samples.
    i. Carefully keep an eye on your stack HWM's, watch for main board LED flashing (FreeRTOS signals), and **add free-RTOS features ONE AT A TIME** to see if they push you over the edge on memory.