Nikola Stefanovski

Investigating FreeRTOS

What is FreeRTOS?

- Market-leading real-time operating system for microcontrollers
- Supports 40+ microcontroller architectures
- Focus on deterministic behavior for time-critical applications

FreeRTOS vs. General Purpose OS

- Single-user environment
- Predictable response times to events
- Strictly prioritized execution

- Maximize fairness
- Complex scheduling for many applications
- Extensive hardware abstraction
- Non-deterministic response times acceptable

FreeRTOS Applications

- IoT devices
- Medical devices
- Industrial control systems
- Automotive systems
- Applications with real-time requirements where failure is detrimental

Tasks

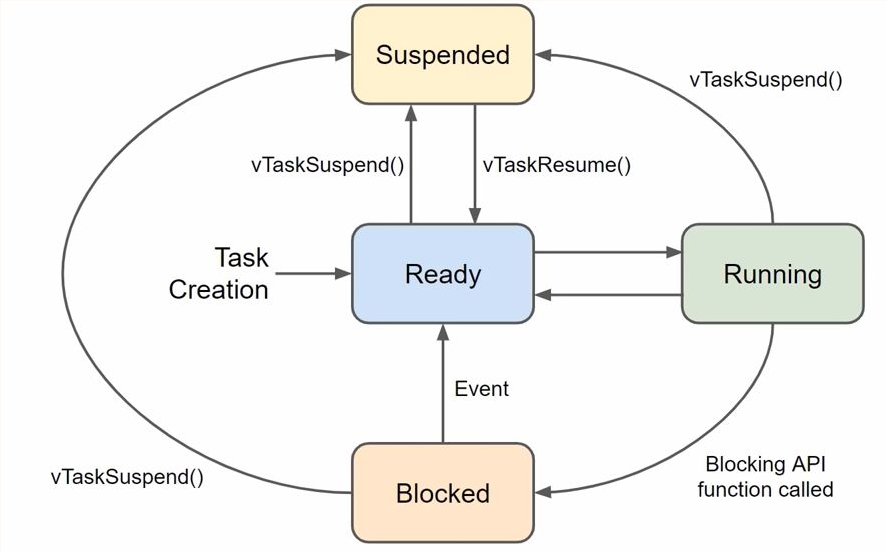
- Tasks: “threads” sharing memory space
- No address translation!
- No user/kernel mode separation

Task Control Block (TCB) structure:

- Stack pointer
- Priority value
- State information

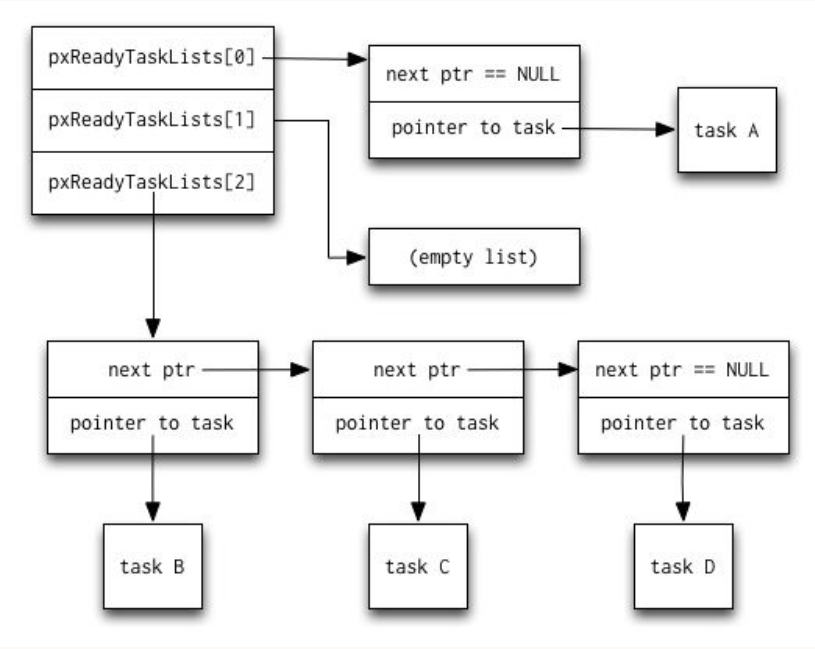
States

- Running
- Ready
- Blocked
- Suspended
  - Explicitly removed
  - Requires API call



Scheduling

- Fixed priority preemptive scheduling
  - Highest priority ready task always runs
  - Time-slicing only among equal priority tasks
  - Predictable interrupt latency

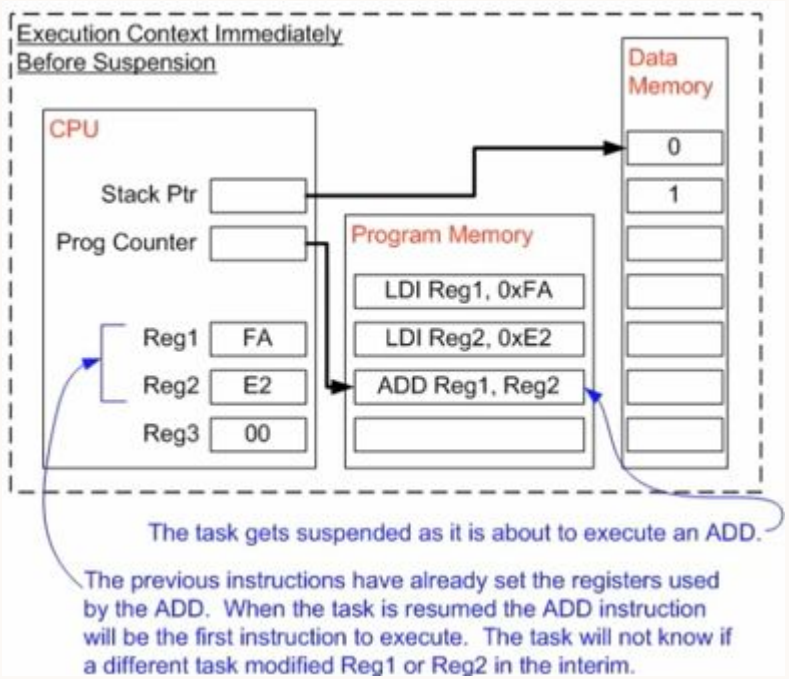


Contexts

Context switch process:

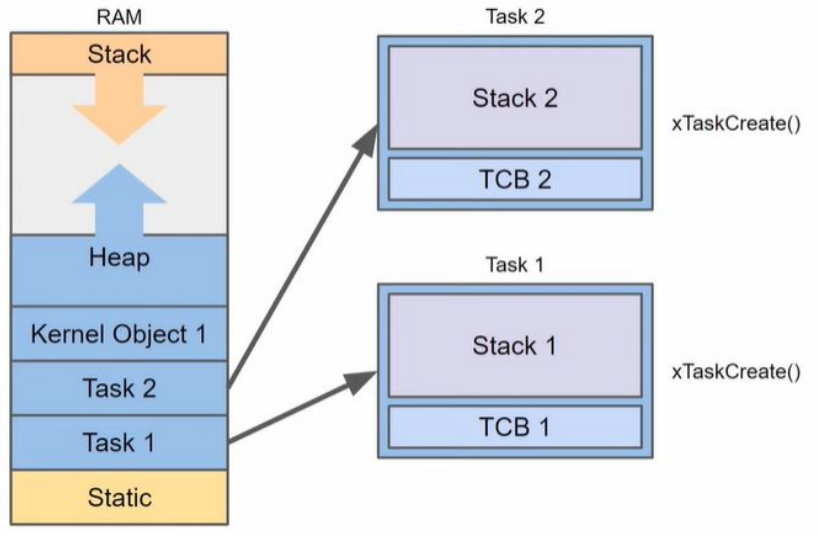
- Save current task context to its stack
- Update TCB pointer Restore context from new task's stack

What if another task modifies global variables?



Memory

- Heap 1: No freeing
- Heap 2: Best fit without coalescing
- Heap 3: C stdlib wrapper
- Heap 4: First fit with coalescing
- Heap 5: Multi-region memory support



No virtual memory system!

Memory Allocation

- malloc()/free() are not thread safe nor deterministic
- Different RTS have varying requirements for RAM
- Alternative implementation is often required

Solution:

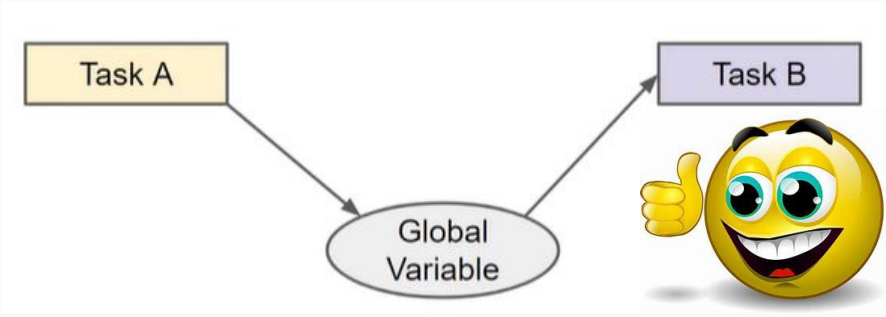
- Keep Allocation API in portable layer
  - Sits outside of source files
  - Allows for application specific implementation

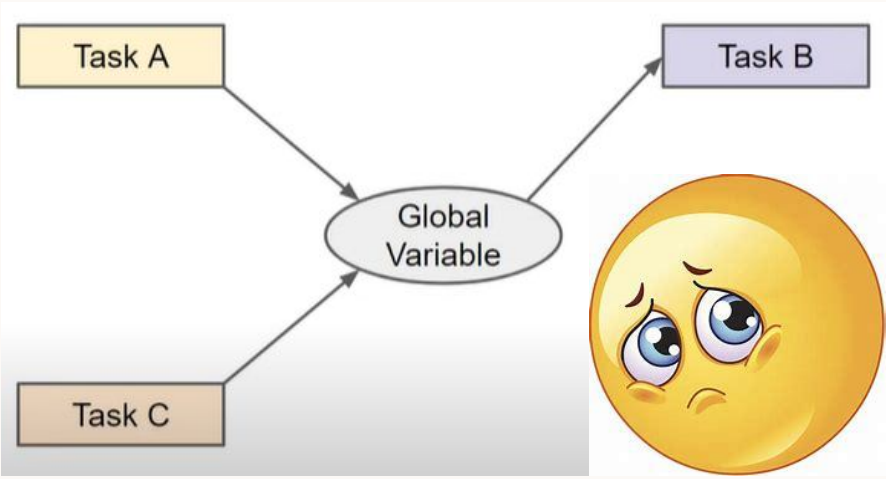
Inter Task Communication

- Simple write/read between two tasks can be feasible

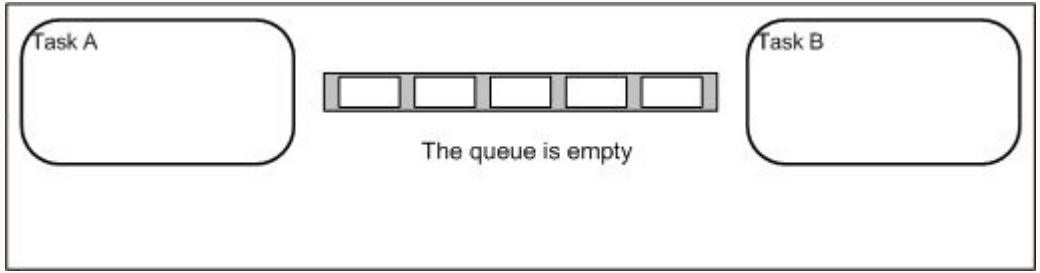
- Issues arise when multiple tasks attempt to write

How does FreeRTOS combat this?





Queues!



Primary inter-task communication mechanism:

- Fixed-size FIFO data structure
- Safe by design (atomic)
- Status codes for if queue is empty or full

What about variables such as counters?

Mutexes

- Special type of binary semaphore
- Implemented using queues
- Stores handle of owning task