# Real-time Operating System

BY

**DHWAJ VANDANKUMAR JANI**

# PROJECT:

RTOS solution for an Arm M4F controller that implements a preemptive RTOS solution with support for semaphores, yielding, sleep, priority scheduling, priority inheritance, and a shell interface.

# Function Description:

**Scheduler**:

Each time the scheduler is called, it will look at all ready processes and will choose the next process.Implements prioritization up-to 8 levels. The method used to prioritize the relative importance of the tasks is implicit in the assignment of the prioritization when createThread() is called.

**Kernel Functions:**

- **yield()** will yield execution back to the kernel that will store the return address and save the context necessary for the resuming the process later.
- **sleep(time_ms)** will call PendsvcIsr which saves context before it switches the task. The process is then delayed until a kernel determines that a period of time_ms has expired. Once the time has expired, the process that called sleep(time_ms) will be marked as ready so that the scheduler can resume the process later.
- **wait(semaphore)** causes a process to block until a resource or resources is available. The waiting process will be recorded in the semaphore process queue.
- **post(semaphore)** will signal waiting tasks that semaphore is free to use.

- **createThread()** stores the process name and initializes the process stack as needed.
- **destroyThread()** removes a process from the TCB and cleans up all semaphore entries for the process that is deleted.
- **systickIsr()** handles the sleep timing and kernel functions.
- **pendSvcIsr()** will store the return address and save the context necessary for the resuming the process later after which it performs task-switch.
- **shell** process that hosts a command line interface to the PC. The command-line interface supports the following commands (borrowing from UNIX):
  **ps:** The PID id, process name, and % of CPU time
  **ipcs:** the semaphore usage is displayed.
  **kill <PID>:** This command allows a task to be killed by referencing the process ID.
  **reboot:** The command restarted the processor.
  **pidof <Process_Name>:** returns the PID of a process <Process_Name> & starts a process running in the background if not already running. Only one instance of a named process is allowed.