

# Lecture 15: FreeRTOS

Vikram Iyer

# Announcements

<p>9:30-11:30 OH (Deeksha) ECE 345 20</p> <p>11:30-13:30 OH (Alex) ECE 345</p>	<p>13:00-15:00 OH (Deeksha) ECE 345 21</p>	<p>9:30-11:30 OH (Deeksha) ECE 345 22</p> <p>12:30-14:20 Lecture MOR 230 <i>Lecture 15: Intro to FreeRTOS</i></p> <p>14:00-15:00 OH (Vikram) ECE 345</p> <p>23:59 Lab 3 due</p>	<p>10:00-12:00 OH (Zach) ECE 345 23</p> <p>12:30-14:30 OH (Alex) ECE 345</p>	<p>12:30-14:20 Lecture MOR 230 <i>Lecture 16: Intro to Critical Sections and Semaphores</i></p> <p>14:00-15:00 OH (Vikram) ECE 345 24</p>
<p>Memorial Day 27</p>	<p>13:00-15:00 OH (Zach) ECE 345 28</p>	<p>9:30-11:30 OH (Deeksha) ECE 345 29</p> <p>12:30-14:20 Lecture MOR 230 <i>Lecture 17: FreeRTOS Examples</i></p> <p>12:30-15:30 Quiz (20 min during lecture) MOR 230 (Lecture room)</p> <p>14:00-15:00 OH (Vikram) ECE 345</p> <p>14:00-15:00 Quiz (DRS, during OH) ECE 345 (Lab room)</p>	<p>10:00-12:00 OH (Zach) ECE 345 30</p> <p>12:30-14:30 OH (Alex) ECE 345</p>	<p>12:30-15:00 OH (Vikram) ECE 345 31</p>
June				
Monday	Tuesday	Wednesday	Thursday	Friday
03	04	05	<p>23:59 Lab 4 due 06</p>	07

Finals week OH times will be up soon

# Announcements

Lab 3 due today

- Thursday = 1 late days, Friday = 2 late days

Lab 4 will be up soon

- Due during finals week to give you max time/flexibility to work on it
- If you are out of town etc please see me after lecture

Quiz on Wed 5/29 during lecture (20 min)

- In class, written, multiple choice (15 pts)
- Goal is to test concepts introduced after the midterm
- List of concepts will be posted by Friday

Assignment	Points
C prog 1	20
C prog 2	20
Quiz	15
Lab 1	70
Lab 2	70
Lab 3	70
Lab 4 / Project	90
Midterm	45
No Final	
<b>TOTAL</b>	<b>400</b>

# FreeRTOS

Open Source small Real Time OS

Main functions:

- A pre-emptive, priority based, scheduler
- Queues for inter-task communication

[Documentation Link](#) [ [API Reference](#) ]

# FreeRTOS

## Pre-emptive:

Scheduler uses interrupts to “break-in” to running tasks

Task model:

```
while(1) {  
    Do stuff;  
    vTaskDelay(int ticks); // optional!!!  
}
```

# FreeRTOS

## Priority-Based:

Each task can be assigned a [priority](#) level:

0 = lowest priority

`configMAX_PRIORITIES` = highest priority (default 4)

## Configuration:

`../libraries/FreeRTOS/freeRTOSConfig.h`

`../libraries/FreeRTOS/freeRTOSVariant.h`

# FreeRTOS

## Scheduler Ticks:

Lab2, Lab3: 1ms/2ms

FreeRTOS fastest option: 15ms (!) (`portTickRateHz`)

For faster things do ISRs with timer interrupt (e.g. every 1ms)

**Terminology note:** “port” in FreeRTOS = a version for a specific chip

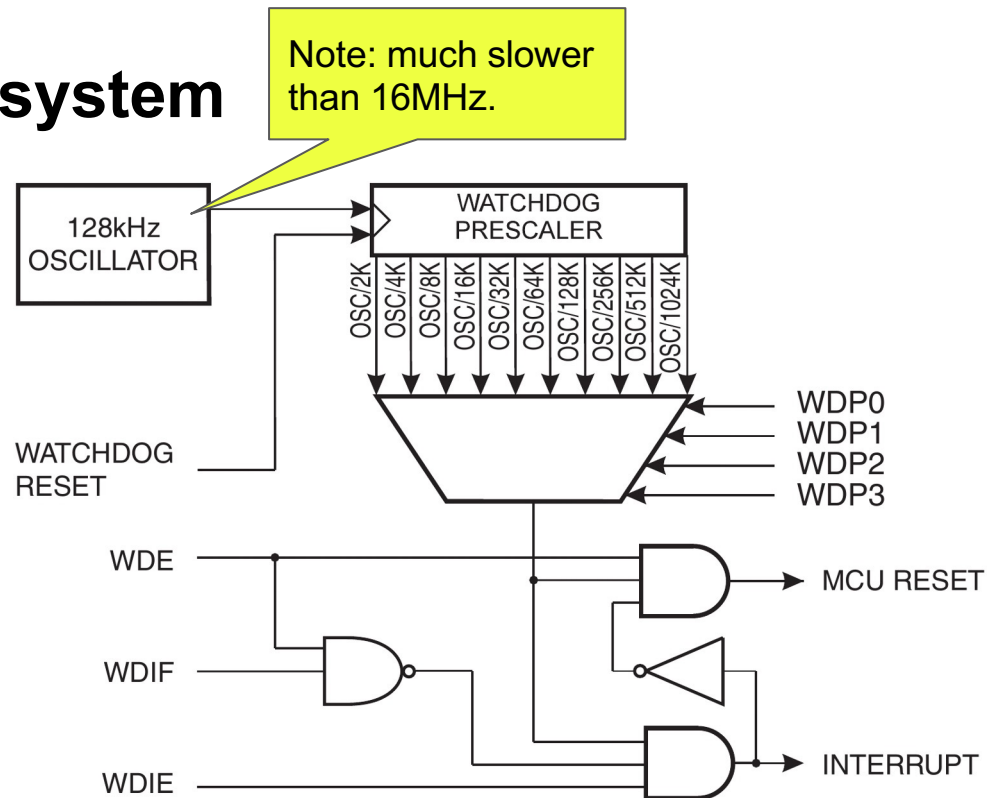
# FreeRTOS scheduler tick system

Uses built-in watchdog timer (all arduinos)

- + Doesn't use main timers
- Can't have watchdog functionality

```
/* Watchdog period options:
   WDTO_15MS
   WDTO_30MS
   WDTO_60MS
   WDTO_120MS
   WDTO_250MS
   WDTO_500MS
   WDTO_1S
   WDTO_2S
*/
```

Special Fuse WDTON to prevent tampering in software!



# Watchdog Timer Background

- Goal: protect a system from software crash or hung state.
- Timer controls system reset input (“reset button”)
- Must be cleared periodically by software.
- If software fails to reset watchdog, system does a restart
- External hardware watchdog can be used
- Example:



# ISR -- “down in the weeds”

```
#define portSAVE_CONTEXT()
    __asm__ __volatile__ ( "push    _tmp_reg\n\t" \
                          "in      _tmp_reg_, __SREG\n\t" \
                          "cli\n\t" \
                          "push    _tmp_reg\n\t" \
                          "in      _tmp_reg_, 0x3B\n\t" \
                          "push    _tmp_reg\n\t" \
                          "in      _tmp_reg_, 0x3C\n\t" \
                          "push    _tmp_reg\n\t" \
                          "push    _zero_reg\n\t" \
                          "clr     _zero_reg\n\t" \
                          "push    r2\n\t" \
                          "push    r3\n\t" \
                          "push    r4\n\t" \
                          "push    r5\n\t" \
                          "push    r6\n\t" \
                          "push    r7\n\t" \
                          "push    r8\n\t" \
                          ". . .\n\t" \
                          "push    r27\n\t" \
                          "push    r28\n\t" \
                          "push    r29\n\t" \
                          "push    r30\n\t" \
                          "push    r31\n\t" \
                          "lds     r26, pxCurrentTCB\n\t" \
                          "lds     r27, pxCurrentTCB + 1\n\t" \
                          "in      _tmp_reg_, __SP_L\n\t" \
                          "st     x+, _tmp_reg\n\t" \
                          "in      _tmp_reg_, __SP_H\n\t" \
                          "st     x+, _tmp_reg\n\t" \
                          );
```

# FreeRTOS References

FreeRTOS library/API: <https://www.freertos.org/a00106.html>. (Specifically, Task Creation, Task Control will be especially helpful.)

Here is also an interesting article about the “tick” for the FreeRTOS system:

<http://www.learnitmakeit.com/freertos->

[tick/#:~:text=At%20the%20simplest%20level%2C%20the,and%20overhead%20of%20task%20switching.](http://www.learnitmakeit.com/freertos-tick/#:~:text=At%20the%20simplest%20level%2C%20the,and%20overhead%20of%20task%20switching.)

# FreeRTOS example with ISR

## Tasks:

- 1) Blink an LED (250ms ON, 100ms OFF, ~3Hz)
  - a) On-board vs. Off board LED
  - b) Controlled by `volatile int blinkerpin: 13=onboard, 10=offboard`
- 2) Analog input
  - a) Read Analog input (from thumbstick)
  - b) Send its value on Serial to laptop
  - c) sleep(for a while)
- 3) Change Blink
  - a) Change `blinkerpin` between offboard (10) <-> onboard (13).
  - b) Alternate 2 second each

# FreeRTOS example with ISR

**Versions:** (2 ways to change the blink output pin)

1) Change Blink Task uses delays: `vTaskDelay(2000/portTICK_PERIOD_MS)`

2) ISR-driven

a) Set up Timer5 to cause interrupt at 50Hz

b) `ISR(TIMER5_COMPA_vect) { intcount++; }` (that's it!)

c) ChangeBlink counts up to 100 interrupts (2 sec).

# FreeRTOS example with ISR

## Challenges:

- “Starvation” - one task does not get enough cycles
- Uneven periodicity (tasks do not meet deadlines every time)
- Implicit interrupt blocking
  - Serial functions disable interrupts at times
  - Scheduler ???
- Set priorities carefully.