Lecture 16: FreeRTOS

Vikram Iyer

Adapted from material by Blake Hannaford

Announcements

- Lab 3 due Friday
- Short quiz on Canvas due during Finals week
- Lab 4 is up, due in 2 parts
  - Part 1 is just running an example and coming up with your project idea
  - Finals week OH posted soon

Monday	Tuesday	Wednesday	Thursday	Friday
22 09:00-11:00 OH (Vicente) ECE 345 13:30-15:30 OH (Joe) ECE 345	23 09:00-11:00 OH (Kyle) ECE 345	24 09:00-11:00 OH (Vicente) ECE 345 12:30-13:50 Lecture MOR 230 *Lecture 16: Intro to FreeRTOS* 14:00-15:00 OH (Vikram) ECE 345	25 09:00-11:00 OH (Kyle) ECE 345 13:30-15:30 OH (Joe) ECE 345	26 12:30-13:50 Lecture MOR 230 *Lecture 17: Intro to Critical Sections and Semaphores* 14:00-15:00 OH (Vikram) ECE 345 23:59 Lab 3 due
29 Memorial Day	30 09:00-11:00 OH (Kyle) ECE 345	31 09:00-11:00 OH (Vicente) ECE 345 12:30-13:50 Lecture MOR 230 *Lecture 18: FreeRTOS and Lab 4 Info* 14:00-15:00 OH (Vikram) ECE 345	01 09:00-11:00 OH (Kyle) ECE 345 13:30-15:30 OH (Joe) ECE 345	02 12:30-13:50 Lecture MOR 230 14:00-15:00 OH (Vikram) ECE 345 23:59 Lab 4.1 due
05	06	07	08	09 23:59 Lab 4.2 due

June

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

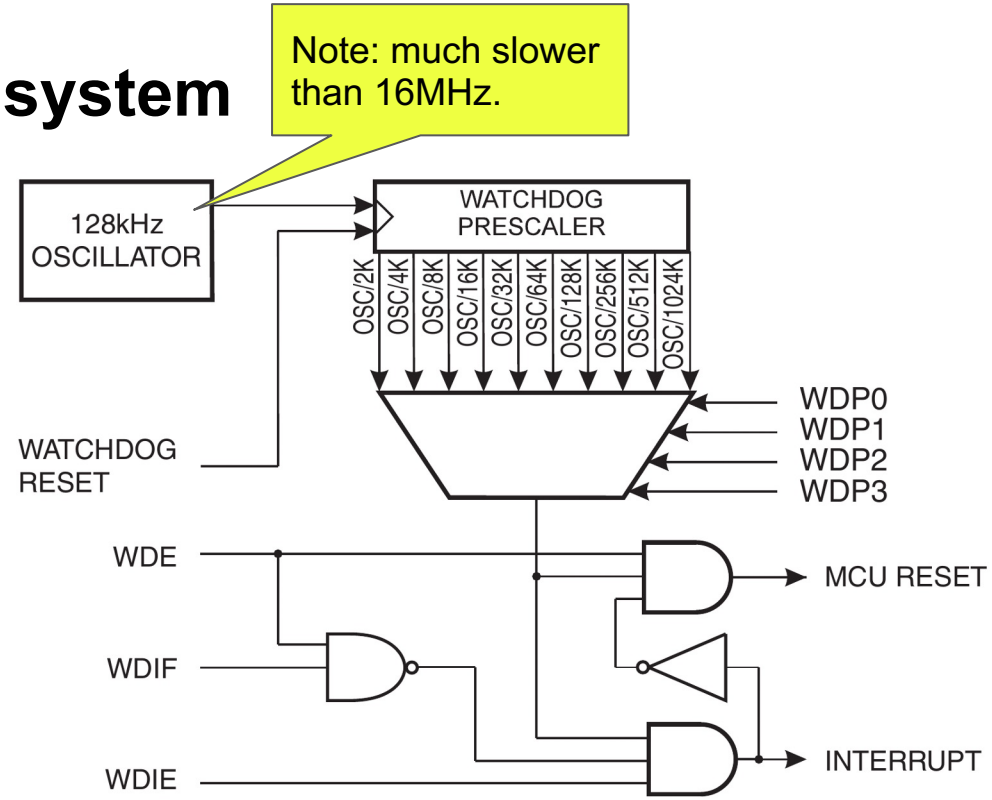
FreeRTOS scheduler tick system

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"   \
                                                . . .
                                "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"   \
                              );
```

Source: FreeRTOS `port.c`

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.

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 interupts (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.