EE 3220 – System-on-Chip Design – 2021/22 Spring

Assignment 1 – Due date: Feb 14, 2020 (Monday) – 23:59pm

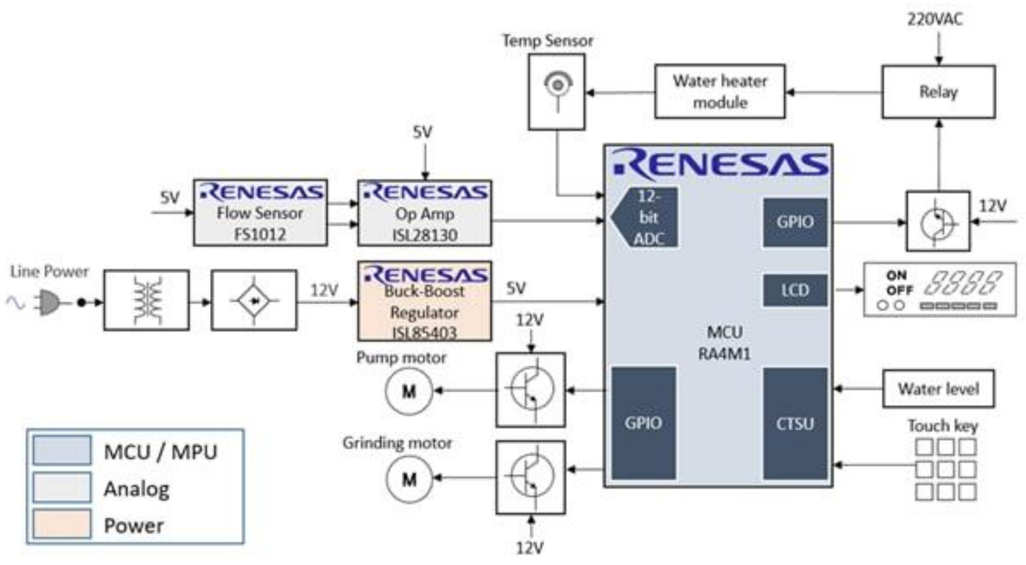
Submission method – Canvas online assignment collection box

Name: ________________________ **Date:** ______________
Student ID: ___________________ **Mark:** _____________

Submission guidelines:

- Please prepare your assignment in “PDF” format. (1%)
- Please rename your file to “EE3220 – assignment 1 – studentID.pdf”. (1%)
- For late submission, 10% deduction per day.

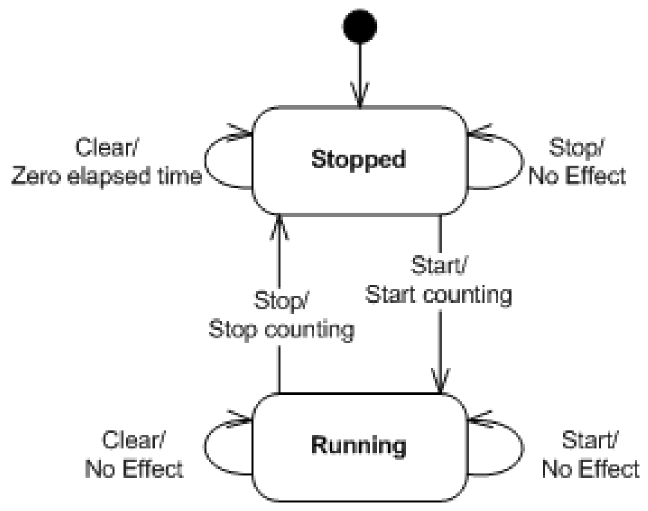
Q1: Consider a controller for a smart coffee machine (as shown below) as an embedded computer system.



A. What are the inputs?
   e.g. Touch key, water level and etc.
B. What are the outputs?
   e.g. pump motor, grinding motor and etc.
C. Name at least two safety features to include, and specify what hardware and/or software is needed for each.
   e.g. temperature, water level. Please also notice the relative modules.
D. Describe a useful feature which could you add in software without requiring additional hardware.
E. Describe a useful feature which could you add in software if the controller could keep track a user profile setting and add a new I/O component.
F. Describe a useful feature which could you add in software if the coffee machine included an internet connection.

For D,E,F, you can add your design or feature with reasonable instructions.

For Q2-Q4, consider a stopwatch as described here. The state machine below presents the desired behavior.



It has the following hardware.

- Buttons for Start, Stop, Lap and Clear functions.
    - Pressing **Start** starts the stopwatch running. If pressed multiple times, stopwatch continues running without resetting elapsed time.
    - Pressing **Stop** stops the stopwatch from counting.
    - Pressing **Clear** zeroes out the elapsed time if the stopwatch is not running. If it is running, the clear button is ignored.
    - Pressing **Lap** records the current time. If pressed multiple times, different laps data are recorded. If it is at the stop state, takes no effect.
- A timer which triggers an interrupt every 1 ms. The timer drives a counter which counts milliseconds since system start-up, and can be read as elapsed_time_counter.
- A display to show elapsed time with 1 ms resolution. The display must be updated 10 times per second.
- A "Lap" button, which records the time for a lap as shown in the above Apple Watch (as an example).

Q2: Design pseudocode for the software using event-triggered scheduling with interrupts. Assume that each button can generate an interrupt.

- Use a variable called state to indicate whether the stopwatch is stopped or running or recording
- Use a variable called **elapsed_time** to track how much time has elapsed since the start button was pressed.
- Use a variable called **display_delay** to track how many milliseconds remain until the display needs to be updated again.

- Use a variable called **lap_time** to record the current lap time.

```
Main thread:
   state = stopped
   display_delay = 100
   elapsed_time = 0
   lap_time =0
   lap = 0
Start ISR:
   state = running Timer
   ISR: if state == running
   elapsed_time += 1 ms
   lap_time += 1ms
   display_delay -= 1
   if display_delay == 0 { display_delay = 100 display elapsed_time }
Stop ISR:
   state = stopped
Clear ISR
   if state == stopped
   elapsed_time = 0
```

Q3: Now design pseudocode for the software using a **static scheduler without using any interrupts**. Assume that the timer updates a hardware register called elapsed_time_register every millisecond.

- Use a variable called state to indicate whether the stopwatch is stopped or running
- Use a variable called start_time to record when the start button was pressed.
- Use a variable called stop_time to record when the stop button was pressed.
- Use a variable called lap_time to record when the lap button was pressed.
- Use a variable called next_display_update to indicate when the display needs to be updated next.

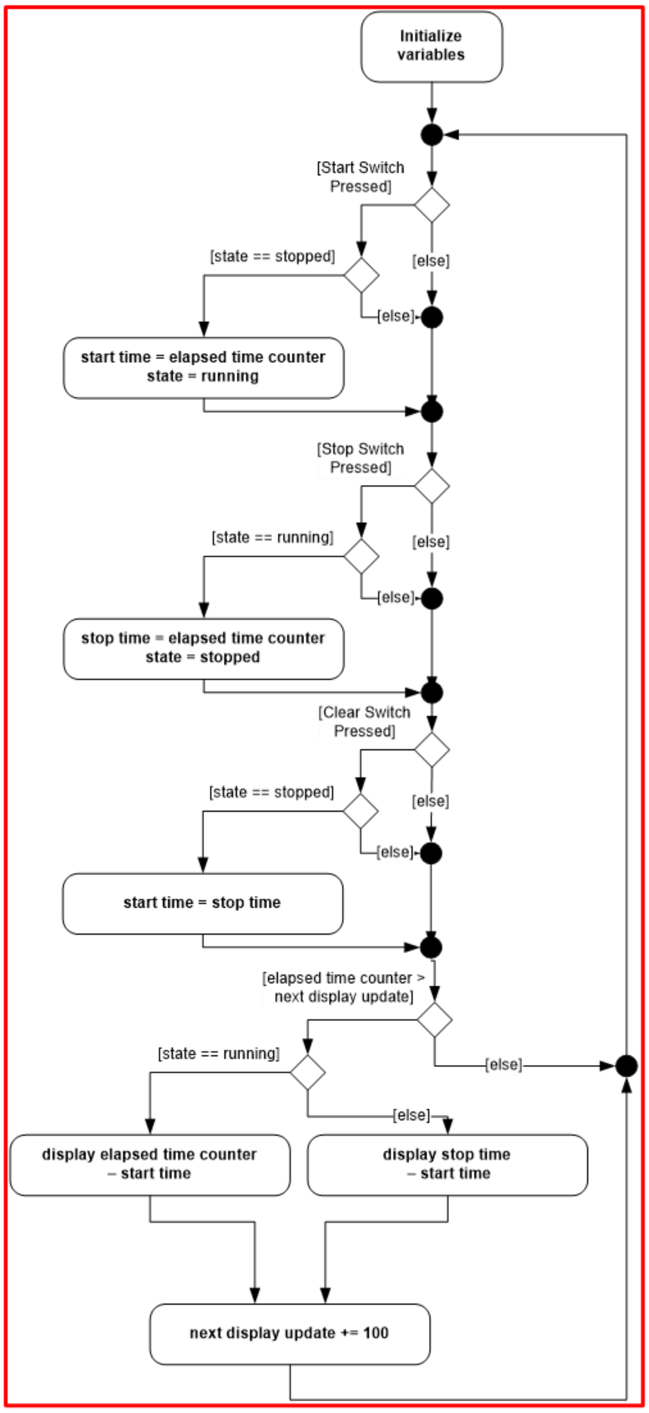
```
state = stopped
display elapsed_time_counter
next_display_update = elapsed_time_counter + 100
while (1) {
    if start switch pressed {
            if state == stopped {
                    start_time = elapsed_time_counter
                    state = running
            }
    }
    if stop switch pressed {
            if state == running {
                    stop_time = elapsed_time_counter
                    state = stopped
            }
    }
    if clear switch pressed {
            if state == stopped {
                    start_time = stop_time
```

```
                }
        }
        if lap switch pressed {
                if state == running {
                        display lap_time
                        lap_time = 0
                        lap +=1

                }
        }

        if elapsed_time_counter > next_display_update {
                if (state == running)
                        display elapsed_time_counter - start_time
                else
                        display stop_time – start_time
                next_display_update = next_display_update + 100
        }
}
```

Q4: Create a flowchart to represent your solution to the previous question.



This figure shows one solution of stopwatch with basic functions. You can design your process based on your own ideas. For the lap time, you can place this process at proper level in this figure.

Q5: As stated in the lecture, we have introduced the "Qualcomm Snapdragon 8 Gen 1".

- What is the ARM Architecture used in this SoC?
- What kind of new instructions are introduced in this Architecture?
- Please introduce the basic components used in this SoC.
- This SoC can achieve high performance and power saving. In your own words, how this SoC is able to achieve these outstanding features as compared to State-of-the-art products?

Please refer to the content of Qualcomm Snapdragon 8 Gen 1 and answer with reasonable instructions.

Q6: Assume a wafer size of 18 inches, a die size of 2.5cm^2, defect is 1 / cm^2. Alpha is 3. Please use the equations provided in the lecture notes, determine the die yield of this CMOS process.

$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha}\right)^{-\alpha}$$

Die yield = 0.16

Q7: A modern desktop processor may contain 1 billion transistors in a chip area of 100 mm^2. If Moore's Law continues to apply, what would be the chip area for those 1 billion transistors after 8 years (assume the cycle is 24 months)? What percentage is that area of the original area?

According to Moore's Law: The number of transistors on a chip will roughly double every two years. So, roughly 100 mm^2 can contain 16 billion transistors. The area will be 6.25mm^2

Q8: Consider a system with the following tasks. We wish to **minimize** the response time for task E. For each type of scheduler, describe the sequence of processing activities which will lead to the minimum and the maximum response times for task E. Assume that each task is ready to run and there are no further task releases.

Task	Duration
A	3
B	1
C	4
D	2
E	5

a. Static, non-preemptive scheduler
b. Dynamic, non-preemptive scheduler
c. Dynamic, preemptive scheduler

a. Best case: Task E starts at time 0.
Worst case: Task E starts at last after ABCD, T_r = 0+3+1+4+2 = 10

b. Best Case: Task E starts immediately (at time 0).
Worst Case: Longest task (C) just started running ε time units ago, so C won't run until it finishes. $T_r = 0 + 4 - \varepsilon + 5 = 9 - \varepsilon$

c. Best Case: Task E starts immediately (at time 0).
Worst Case: Longest task (C) just started running ε time units ago, but it is preempted by E.
$T_r = 0 + 5 = 5$

d.

Q9: An embedded operating system uses the Shortest Job First scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process Burst time Arrival time

P1	20	0
P2	25	15
P3	10	30
P4	15	45

Please draw the figure of all the process with burst time and wait time.

Q10: Consider the following table of arrival time and burst time for four processes P0, P1, P2 and P3.

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4 ms
P2	2 ms	5 ms
P3	4 ms	3 ms

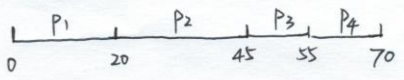
Now, the Shortest Remaining Time (SRT) scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the four processes? Please draw the figure of all the process.

Q9 and Q10

Q9.

Process	Burst Time	Arrival Time
P_1	20	0
P_2	25	15
P_3	10	30
P_4	15	45

Shortest Job First (non-preemptive)



Wait time:

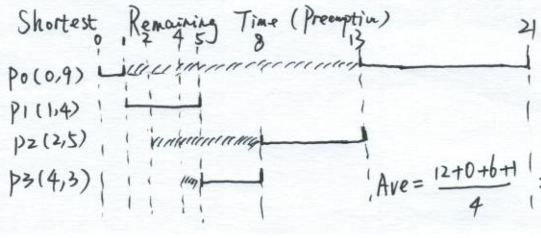
$P_1 = 0$ $P_2 = 20-15 = 5$ $P_3 = 45-30 = 15$ $P_4 = \cancel{70}$ $55-45=10$

$$\text{Ave} = \frac{0+5+15+10}{4} = 7.5$$

Q10.

Process	~~Burst~~ Arrival Time	~~Arrival~~ Burst Time
P0	0ms	9ms
P1	1ms	4ms
P2	2ms	5ms
~~P~~P3	4ms	3ms

Shortest Remaining Time (Preemptive)



$$\text{Ave} = \frac{12+0+6+1}{4} = 4.75$$

~ END ~