EECS149–Fall 2014 — Solutions to Homework 2

Aaron Feldman, SID 23983332, `ee149-ch`

September 25, 2014

Chapter 10 - Ex 2

(a) Strictly speaking, atomic means that it's the smallest computational element. That means that we would need the processor to be able to compute A, B, C in one line of assembly code and that the processor can compute it in 1 clock cycle. Consider the case where this wasn't true, but we disabled interrupts before entering A, B, or C, and re-enabled them after. If they weren't "atomic" where it took the processor 1 cycle to compute, then if the system loses power during execution of A, B, C, then we can't be guaranteed that once entering A, B, C, that they will be completed upon power up.

Chapter 10 - Ex 3

```
#include <stdio.h>

volatile uint timerCount = 0;
void countUp(void) {
    timerCount++;
}

int main(void) {
    timerCount = 0; // reset for system start
    SysTickPeriodSet(SysCtlClockGet() / 100);
    SysTickIntRegister(&countUp);
    SysTickEnable();
    SysTickIntEnable();

    return 0;
}
```

This clock will run for the max 32 int size / 100 seconds before overflowing.
$2,147,483,647/100 = 21,474,836.47s$

Chapter 10 - Ex 4

```
volatile static uint8_t alerted;
volatile static char* display;
void ISRA() {
    if (alerted ==0) {
```

```
5             display = "normal";
6         }
7 }
8 void ISRB() {
9     display = "emergency";
10    alerted = 1;
11 }
12 void main() {
13     alerted = 0;
14         ...
15 }
```

(a) We are not guaranteed that this program will function as designed. If the ISRB routine interrupts ISRA after line 4 then the system will display "emergency" from the ISRB routine then return to ISRA and continue execution on line 5 to then display "normal."

Chapter 10 - Ex 7

(a) Whether or not that it is possible for the ISR to update the value of sensor1 while the main function is checking whether sensor1 is faulty would depend on the ISA of the platform. So without knowing the platform, we won't be guaranteed that the ISR couldn't update the value of sensor1. That said, isFaulty(sensor1) is a function call and probably won't be executed atomically in most ISA's, leading us to suspect that it could be interrupted and have the value of sensor1 updated during a call in main.

(b) It's very possible that the faulty sensor wouldn't be reported. This is because the value may be updated to a non-faulty value before preceding the be checked in the main loop.

(c) If the timer was set for too short of an interval, the main function might not be able to exectute, hence the if statements checking for faulty sensors might never be executed.

(d) Yes, having multiple threads or not does not effect the possibility that the call to "isFaulty(sensor1/2)" isn't necessarily atomic. Hence during the call, the value for sensor1/2 might be updated before the call is returned. This then means we can't guarantee that the value wasn't changed and could result in undesirable behavior of the program.

Chapter 2 - Ex 4

This model is neither LTI nor BIBO stable.

Chapter 2 - Ex 5

(a) $\theta(t) = \theta(0) + \int_0^t \dot{\theta}(\tau)\, d\tau$

(b) No, the system is not BIBO.

(c) $= \int_0^t K(\psi(\tau) - \theta(\tau))\, d\tau$

$= \int_0^t Kau(\tau)\,d\tau \; - \; \int_0^t K\theta(\tau)\,d\tau$

$= Kat \; - \; K\int_0^t \theta(\tau)\,d\tau$

$= au(t)\left(1 \; - \; e^{-kt}\right)$

$\rightarrow \; \theta(0) \; = \; 0$

$\rightarrow \; \theta(\infty) \; = \; a$