ECE 2049: Homework #3

Due on Thursday, February 12, 2015

Ted Meyer

Problem 1

A)

busywork.

B)

busywork.

C)

This sets up the XT1 and XT2 crystal clocks, which are multiplexed on Port 5, pins (4,5) and (2,3) respectively. It is necessary to do this in main() to enable both clocks.

D)

Freq. XT1 = 32768 Hz.
Freq. XT2 = 4 MHz.

E)

```
void config() {
    UCSCTL0 = 0x1448;
    UCSCTL1 = 0x0040;
    UCSCTL2 = 0x10B7;
    UCSCTL3 = 0x0000;
    UCSCTL4 = 0x0044;
    UCSCTL5 = 0x0000;
    UCSCTL6 = 0xC0CC;
    UCSCTL7 = 0x0403;
    UCSCTL8 = 0x0707;
}
```

Problem 2

A)

UTC is a time standard based on GMT but is precisely defined by the IRCC to include such things as leap seconds. Most time zones are based on a UTC offset. EST is the same things as UTC - 5:00. ”Zulu” time is a 24 hour time based on UTC.

B)

There are 60*60*24*365=31,536,000 seconds in a normal year, and 31,622,400 seconds in a leap year. Considering that 1900 was NOT a leap year, this means that 3,099,621,045 seconds after UTC-0 is in fact 3,099,621,045 - (31,536,000 * 4) = 2,973,477,045 seconds after Jan 01, 1904. In every subsequent set of 4 years, there are a total of 126,230,400 seconds. 2,973,477,045 / 126,230,400 = 23 full sets of four years, and

Problem 2 continued on next page. . .

2,973,477,045 % 126,230,400 = 70,177,845 remaining seconds after 1996. Another two years is 31,536,000 + 31,622,400 = 63,158,400 seconds, which puts us at 1998 with 7019445 seconds left. Since there are 60*60*24 = 86400 seconds per day, this ammounts to 81 days, with a remaining 21045 seconds. the 81st Day in 1998 is March 22nd. 21045 seconds is 5 hours, 3045 seconds, or 5 hours, 50 minutes, and 45 seconds. The overall time/date is March 22nd, 1998, 5:50:45 AM (UTC).

C)

It can be demonstrated in a very simple manner by simply stating that the number of nanoseconds in a leap year is 1.3176e+15, while the number of representable integers in a 64 bit number is 1.8446744e+19.

D)

5132109 / (60*60*24) = 59 days and 34509 seconds. 34509 / (60*60) = 9 hours and 2109 seconds. 2109 / 60 = 35 minutes and 9 seconds. This would be March 1st, 2015, 9:35:09 AM.

E)

```
void runtimerA2() {
    TA2CTL = TASSEL_1 + CNTL_0 + ID_1 + MC_1;
    TA2CCR0 = 32767;
    TA2CCTL0 = CCIE;
}
long unsigned int timer = 0;
#pragma vector=TIMERA_2_VECTOR
_interrupt void Timer_A2_ISR() {
    timer++;
}
```

F)

```
void write_time(uint16_t *day, uint16_t *hour, uint16_t *min, uint16_t *sec, uint16_t time
    *day  = (timer/86400) + 1;
    *hour = (timer%86400) / 3600;
    *min  = (timer%3600)  / 60;
    *sec  = (timer%60);
}
```

Problem 3

A)

(2620+1) * (2 / 1.04857MHz) = 0.00499916076 seconds.

B)

after 11915 inturrupts, the clock will be .01 seconds slow.

C)

```
#pragma vector=TIMER_A0_VECTOR
_interrupt void Timer_A2_ISR(void) {
    if (leap_cnt < 11915) {
        timer++;
        leap_cntr++;
    } else {
        leap_cnt = 0;
    }
}
```

D)

theoretically, it could measure 0.00003051757 seconds. This is not feasable of course, because at that rate, the inturrupt would be triggered again before the inturrupt function had finished executing the first time, and nothing would ever actually get done.

E)

I would pick an SMCLK of 33, because 33 / 32768 is 0.00100708007, which is very close to a power of 10 while also being quite small.

Problem 4

A)

```
//4MHz -> 2199 cycles
void runtimerA2() {
    TA2CTL = TASSEL_2 + MC_1 + ID_1;
    TA2CCR0 = 2199;
    TA2CCTL0 = CCIE;
}

long unsigned int timer = 0;
#pragma vector=TIMER_A2_VECTOR
_inturrupt void Timer_A2_ISR() {
    timer++;
}
```

B)

No. The value of maximum is exact and no truncation occurs.

Problem 4 continued on next page. . .

C)

```
// assume timefmt is 8 chars, [ss.mmmm '\0']
void fill_time(char *timefmt, uint16_t timer) {
    timefmt[0] = '0'+(timer / 25000);
    timer %= 25000;
    timefmt[1] = '0'+(timer / 2500);
    timer %= 2500;

    timefmt[3] = '0'+(timer / 250);
    timer %= 250;
    timefmt[4] = '0'+(timer / 25);
    timer %= 25;
    timefmt[5] = '0'+(timer / 2);
    timer %= 2;
    timefmt[5] = '0';
}
```

D)

52 / 0.0004 = 130000
4MHz = 2200
2200 * (1/3999950) = 0.00055000687
0.00055000687*130000 = 71.5
Therefore, the timer will be super slow.