ECE 2049 A Term 2022

Sakshi Gauro
sgauro@wpi.edu

Problem 1 (20 pts)

a) What are the exact frequencies of XT1 and XT2 on our lab board, the MSP430F5529 Launchpad? What are the default frequencies for ACLK, MCLK and SMCLK? What are the range of possible clock frequencies for MCLK and SMCK? What is each clock (ACLK, MCLK and SMCLK) used for?

→ XT1 is 32768 Hz and XT2 is 4 MHz. The default frequencies for ACLK, MCLK and SMCLK are 1.048576 MHz. ACLK and SMCLK are used by peripherals and MCLK is used by the CPU. Range = 10 kHz − 25 MHz for MCLK & SMCLK.

b) The following line is executed at the beginning of configDisplay() during system initialization by our demo project code. We are working with the clock system not digital IO. Why is it necessary? What does it do?

```
P5SEL |= (BIT5|BIT4|BIT3|BIT2);
```

→ If we want to assign pins to do something other than I/O we need to set them in P5El to 1. Then it will act accordingly. This function set pins 5-2 to XOUT, XIN, XT2OUT and XT2IN respectively. We need to do this because the because the crystal inputs/outputs are multiplexes with those pins.

c) Below are the values of the Unified Clock System registers for two different MSP430 applications. Use the MSP430x55xx User's Guide Chapter 5 and class notes to decode the bit fields for each register. What are the resulting values of ACLK, MCLK, and SMCLK? (*Hint*: see Users Guide page 164 for DCO eqs.)

UCSCTL0 = 0x12A8
UCSCTL1 = 0x0040
UCSCTL2 = 0x18F6
UCSCTL3 = 0x0001
UCSCTL4 = 0x0034
UCSCTL5 = 0x1110
UCSCTL6 = 0x01CC
UCSCTL7 = 0x0000
UCSCTL8 = 0x0707

→ 0: DCO = 18

1. DCORSEL = 4

2. FLLD = 1 → D = 2
FLLN = 011110110 = 246

3. SELREF = XT1CLK = REFOCLK
FLLREFDN = 2

4. ACLK source = XT1CLK
SMCLK source = DCOCLK
MCLK source = DCOCLKDIV

5. ACLKDIV

d) Write a C function that expressly turns on both XT1 and XT2 (see UCSCTL6) and sets the frequency of MCLK and SMCLK to as close to 12 MHz as you can. Explain whether you used XT1 or XT2 as source for DCO REFCLK and why. Assume all other clock settings are to be left in their default state. You may still assume that P5.5-2 are set as shown in (b) at the top of configDisplay() and do not have to be configured by your function. Your code must be typed to be graded.

```
void func(){
UCSCTL6 = 0xC0C0;
UCSCTL3 = 0x0050;
UCSCTL2 = 0x0002;
UCSCTL4 = 0x0033;
}
```

I used XT2CLK because it was 1/3 of 12 MHz so it was easy to convert to 12 MHz.

Problem 2

Time standards are very important to the scientific, engineering and computing communities. However, precisely keeping track of time all over the world is a deceptively tricky business. Watch this video and do a bit of Googling and tell me the following. (30 pts)

a) What is Coordinated Universal Time (UTC)? What time zone is Germany in? What is that in terms of UTC, (e.g., Minsk, Belarus is UTC + 2:00). Scientific times are often written something like 19:00Z, read as 19 hundred Zulu. What is Zulu time?

In computing, UTC based time scales usually store the current time as the number of seconds that have elapsed since the beginning of some "epoch". For the Network Time Protocol (NTP) this is defined as midnight January 1st (GMT) 1900. The elapsed seconds are stored in a 32-bit unsigned integer.

However, in many systems it is more practical to just consider the current year. The time format is often modified such that 0 corresponds to midnight January 1 of the current year. One benefit of using a UTC-style timescale is that a year's time record, to within one second, can readily be represented in a single unsigned 32-bit integer. (Actually, much longer than 1 year of seconds can be contained in a 32-bit unsigned integer.). The year can then be stored in a separate (16-bit) integer.

→ UTC is the time standard by which the world regulates clocks and times. Germany is in Central European Summer time, or UTC +2:00 and Central European Time, or UTC +1:00 Zulu time, or GMT is a time at the zero-median. GMT and UTC both at the 0° longitude line.

b) If a UTC-style clock counting whole seconds was initialized to 0 at midnight 1 Jan 2020 currently has a count of 10,300,000 seconds what day (i.e. date) and time is it? Like Feb 3, 8:43:02 am. *Hint*: Remember that days are counted from 1.

$$10{,}300{,}000 \text{ sec}/60/60/24 = 119 \text{ days}$$

$$\frac{(10\,300\,000 - 119 \cdot 60 \cdot 60 \cdot 24)}{60^2} = 5 \text{ hours}$$

$$\frac{(10\,300\,000 - 119 \cdot 60 \cdot 60 \cdot 24 - 60^2 \cdot 5)}{60} = 6 \text{ minutes } 40 \text{ seconds.}$$

APRIL 30 05:06:40 AM.

c) A data recording package is to be placed on the ocean floor for 9 months during which it will automatically listen and record acoustic data. It will also take various oceano-graphic measurements and must be able to measure time to within 5 millisecond throughout its operational life. The engineers have designed the device to record for a maximum of 300 days in case there was a delay in device retrieval. Can the engineers use a 32-bit unsigned integer to store time to within 5 ms inside this data recorder? Or, must they use a 64-bit variable? Explain your answer.

$1/0.005 = 200 \text{ ticks/sec} \cdot 60 \cdot 60 \cdot 24 \cdot 30 = 518{,}400{,}000$

32 b. unsigned = 32 $\longrightarrow 2^{32} = 4294967296.$

No, the cannot. The unsigned 32 bit int will overflow, as it doesnot have enough states. They must use 64-bit.

d) How would you configure TimerA2 to implement a clock with 5 ms resolution assuming SMCLK has *already been configured* to equal 4 MHz? Write a C code segment (including the TimerA2 setup and TimerA2 ISR) implementing a clock that saves its time (count) to *a variable of the appropriate size*. You *do not* have to configure the UCS registers to set the value of SMCLK. Your timer should generate interrupts only once per 5 millisecond. Do you need leap counting? Explain.

You donot need leap counting because XT2CLK runs at 4MHz already, which goes evenly into zeroHz which is the frequency at 0.005 s/tick. You only need to divide the frequency by 2000.

```
unsigned long int timer = 0;
void setup(void){
TA2CTL = TASSEL_2 + ID_0 + MC_1
TA2CCR0 = 19999; //19999+1 = 20000 SMCLK tics = ~5/1000 seconds
TA2CCTL0 = CCIE; // TA2CCR0 interrupt enabled
_BIS_SR(GIE);
}
#pragma vector = TIMER2_A0_VECTOR
__interrupt void TimerA2_ISR(void){
timer++;
}
```

e.

e) Create a C function that converts the timer count from ***parts c & d*** to recorder mission day, hours, minutes and sec.sss. The count of days starts at 1 not 0 (e.g., Day 1 00:00:00.000 is midnight on the first deployed day). Your code must be typed to be graded. You *do not* have to convert your results to ascii for display.

```
//using an int array to store the values
double[] convertTimer(){
int t = timer/200;
double day = (t/ 86400) +1;
double hour = (t - (t*day)) /3600;
double minute = (t - (t*day*86400) - (t*hour*3600))/60;
double second = (timer/200.0)- ((t*day*86400) - (t*hour*3600) ;
return {day, hour, minute, second};
}
```

3.a.

Problem 3

For a certain application TimerA2 has been configured as shown below.

a) What is the ***exact time*** between interrupts. Assume that ACLK, SMCLK and MCLK are running at their *default settings*. (20 pts)

```
void runtimerA2(void)
{
  TA2CTL = TASSEL_2 | MC_1 | ID_1;
  TA2CCR0 = 499;
  TA2CCTL0 = CCIE;      // enable Timer A2 interrupt
}
```

default : @ 1048570Hz → divires by 2.
↳ 524285 Hz

500/524285 seonds/tick

b) If the system clock and TimerA2 settings from part (a) are used implement a timing system, how long until the time displayed by that system is off by 0.001 seconds? By 0.01 seconds? Will the system be fast or slow? How do you know?

$.001 = x(0.001 - 500/524288)$

$x = 21.58884$ ticks

$21.5888 \cdot 500 / 524285$ $\Rightarrow$ (off by 0.001 in 0.02058884 s or 21.5888 ticks)

$.01 = x(0.001 - 500/524285)$

$x = 215.8884$ ticks

$215.8884 \times 500/524285 \cdot 209884 = 0$

The system will be fast, because 1 tick $\approx 9.5 \cdot 10^{-4}$ s $< 1 \cdot 10^{3}$ s, so the clock is a little fast.

c. A complete measurement cycle takes 42 seconds. Will you need to use any leap counting to maintain the 0.40 ms accuracy of your timer over the whole measurement? Explain your answer.

```
__interrupt void TimerA2_ISR(void){
  if(leap_cnt < 216){
    leap_cnt++;
    timer++;
  }
  else{
    timer-=10;
    leap_cnt=0;
  }
}
```

The leap contains clock will be off by .01 second in 398.527 seconds.

$0.01/(0.216 - (216.000/524285)) = 1934.6476$ repititions.

$1934.6476 \cdot (216.000/524285) = 398.527$ seconds

d) Assume the MSP430 clocks have been configured to the following frequencies.

ACLK = 32,768 Hz MCLK = XT2 = 4.0 MHz SMCLK = XT2 = 4.0 MHz

What is the smallest time interval you could *theoretically* measure using TimerA2 assuming <u>*these*</u> settings for MCLK, SMCLK and ACLK? Could your MSP430 system actually function with the Timer set to that interval? Explain why or why not.

The smallest time interval you could theoretically measure with Timer A2 would be $1/4 \cdot 10^{6}$ s or about 2.5×10^{-7} seconds. The MSP430 system would likely not function with the timer set to that interval because the time it takes for the CPU to run the code in the interrupt would be longer than the tick at the interupt so the code wouldn't be able to run correctly.

Problem 4

An automated measurement process is controlled using an MSP4305529. The execution of the critical step in the measurement must be timed to within 0.40 millisecond. Variances greater than 0.40 ms will make the measurement results useless. Design a timing program suitable for timing this critical operation. (30 pts)

a. Write a function to configure the UCS registers to turn on XT2 and set SMCLK to use XT2 as it's clock source (i.e., set SMCLK = 4 MHz). Don't forget to set P5.5-2 as shown in 1b.

```
void configure(){
P5SEL |= (BIT5|BIT4|BIT3|BIT2);
UCSCTL6 = 0xC0CC;
UCSCTL4 = 0x0054;
_BIS_SR(GIE);
}
```

b. Write a function in C to setup TimerA2 using SMCLK divided by 2 as its clock to measure the interval. Also show your interrupt service routine.

```
void setup(){
TA2CTL = TASSEL_2 + ID_0 + MC_1;
TA2CCR0 = 1; // 2 SMCLK tics = 5/10000000 seconds
TA2CCTL0 = CCIE; // TA2CCR0 interrupt enabled
}
#pragma vector = TIMER2_A0_VECTOR
__interrupt void TimerA2_ISR(void){
timer++;
}
```

c. A complete measurement cycle takes 42 seconds. Will you need to use any leap counting to maintain the 0.40 ms accuracy of your timer over the whole measurement? Explain your answer.

No, because $1/(4 \cdot 10^6/2) = 5 \cdot 10^{-7}$
$0.004/5 \cdot 10^{-7} = 800$. since air timer goes into 0.40 ms evenly (every 800 ticks will be 0.4ms)

d. In Lab 3 you will need to display numeric readings to the LCD screen, but the LCD functions take arrays of char as input. Show how you'd convert your timer count to display seconds and milliseconds to 4 decimal places like *ss.mmmm* to the LCD. Do not have to include code to display the results just how you would fill the char array that would be displayed. Document you code well. Code should be typed to be graded.

```
char* convert(double* timein){
//day, minute, hour, second
//we need to convert each digit to ascii separately
  bool done = false;
  int fig = 1;
  while(!done){ //finds out how many digits the days                    //number
  has.(logbase10)
  if(time[0] / fig >= 10)
    fig*=10;
  else{
    done = true;
  }
  char time[((int)log10(fig)) + 2 + 2 + 2 + 3 + 4 + 1];
    //fig/10 is how many digits of days, the other nums     are digits for hours, /
    //min, sec and decimals of seconds. The 4 is the         dividers (':', ':', ':',
and '.')
    //then +1 for null character
int i;
int c = 0; //the pos of the array we want to fill with a number
int offset= 0; //the amount to subtract to only get the lower digits.
for(i = fig/10; i > 0; i--){
  int k = (((int) timein[0] - offset) / (pow(10, i))      //k gets the digit in the
ith spot
  offset+= (k)* pow(10, i);
  time[c] = k +48;
  c++;
}
int p = c+9 //run for the next 9 digits of the array (:00:00:00)
int count = 1; //keeps the position of timein
for(c=c; c<p; c+=3){
  time[c] = ':';
  time[c+1] = ((int) timein[count]/10) + 48;
  //takes first digit + puts it into char array
  time[c+2] = ((int) timein[count]%10) + 48; //takes       second digit
  count++;
}
time[c] = '.';
time[c+1] = ((int) (timein[3] - ((int)timein[3])) * 10) + 48; //gets the first digit
of the decimal
seconds
time[c+2] = ((int) (timein[3] - ((int)timein[3])) * 100)%10 + 48; //gets second digit
time[c+3] = ((int) (timein[3] - ((int)timein[3])) * 1000)%10 + 48; //gets third digit
time[c+4] = '\0'; //adds null character
return time;
}
```

e. Now assume that the XT2 crystal used with your MSP430F5529 is on the low end of its tolerance and ***actually oscillates at 3,999,980 Hz*** (i.e., it is 0.0005% off.) What would the error associated with your timer at the end of the 42 second long measurement be given that you designed your timer assuming SMCLK = 4 MHz exactly. Will your actual timer be fast of slow? Explain. Will this make your measurement system out of tolerance?

$42.5 \cdot 0.000005 = 0.00021s$ error.

The actual timer will be slow, because the clock is running at a lower(slow) frequency.

ECE2049 Homework #3

Submitted by: Sakshi Gaurv

Date: Sep 24th 2022

Question	Grade
1 - 20	
2 - 30	
3 - 20	
4 - 30	
Total (100):	