

Real-Time Systems / Real-Time Operating Systems
EE445M/EE380L.6, Spring 2015

Final Exam

Date: May 14, 2015

UT EID: _____

Printed Name: _____
Last, First

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signature: _____

Instructions:

- This exam has 10 pages. Please make sure that you have all sheets.
- Open book and open notes.
- No calculators or any electronic devices (turn cell phones off).
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided. *Anything outside the boxes will be ignored.*
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.

Problem 1	15	
Problem 2	15	
Problem 3	15	
Problem 4	10	
Problem 5	10	
Problem 6	20	
Problem 7	15	
Total	100	

Name: _____

Problem 1 (15 points): Hold and wait

- a) In class we discussed deadlock prevention by ensuring that hold and wait conditions can never occur. Show the C implementation of the spinlock realization of *OS_Wait2()* and *OS_Signal2()* functions that simultaneously acquire and release two semaphores. Your implementation must ensure that neither routine is ever waiting (spinning) for one semaphore while holding the other. Also, both functions must only return once both semaphores have been acquire or released, respectively.

```
void OS_Wait2(long *s1, long *s2)
{
}
}
```

```
void OS_Signal2(long *s1, long *s2)
{
}
}
```

- b) How could your OS ensure that programmers never accidentally create a hold-and-wait condition, i.e. wait for two semaphores by simply calling the normal *OS_Wait()* twice? Just describe the idea, no code needed.

Name: _____

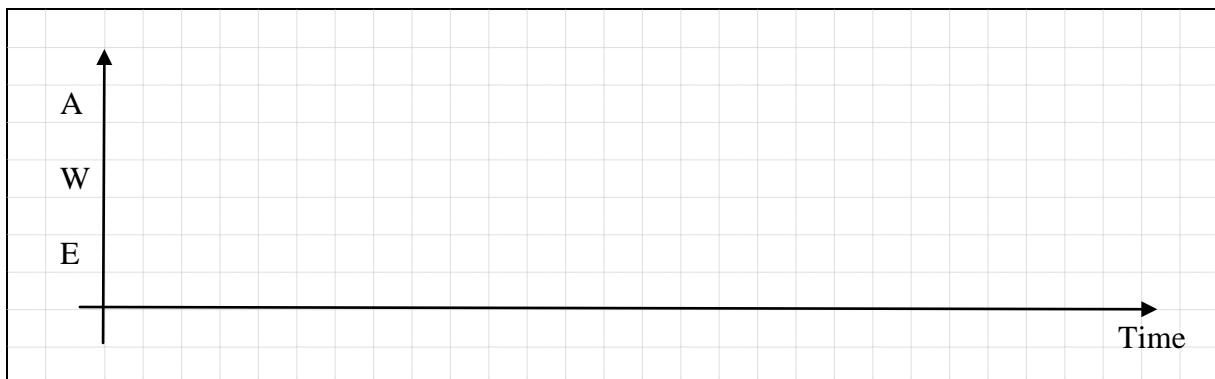
Problem 2 (15 points): Real-Time Scheduling

Consider a priority scheduled real-time system running three periodic tasks with the following priorities and execution times. You can assume zero context switch and interrupt overhead.

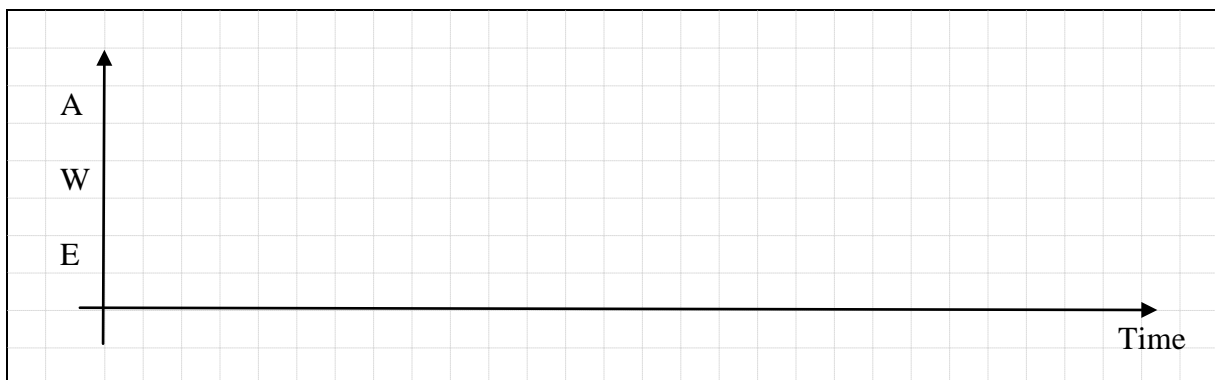
Task	Priority	Execution Time	Period
Airbag (A)	High	10 μ s	30 μ s
Warning (W)	Medium	20 μ s	60 μ s
Engine (E)	Low	30 μ s	120 μ s

- a) Does this task set follow a rate monotonic scheduling (RMS) strategy? If so, why? If not, why not?

- b) Draw the task executions over time. Assume that all tasks become ready to execute, i.e. start their first period at time zero. Draw one iteration of the schedule until it starts repeating. Is the task set schedulable, i.e. do all task finish their execution before the start of their next period (=deadline)? Is there any priority inversion? If so, mark the duration of the inversion.



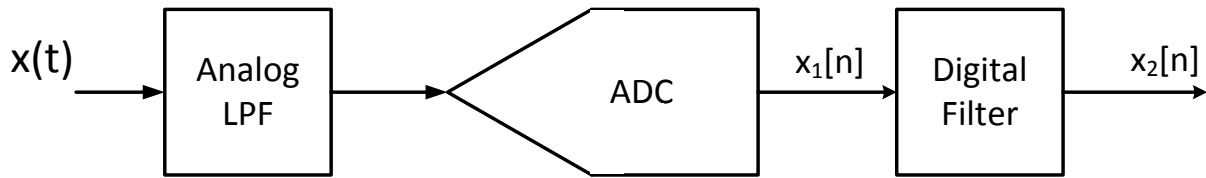
- c) Now assume that tasks A and E share a mutex (binary semaphore) that they acquire at the beginning of each execution and hold for the whole duration of their execution. Draw one iteration of the schedule. Is the task set schedulable? Is there any priority inversion? If so, mark the beginning and end of the inversion.



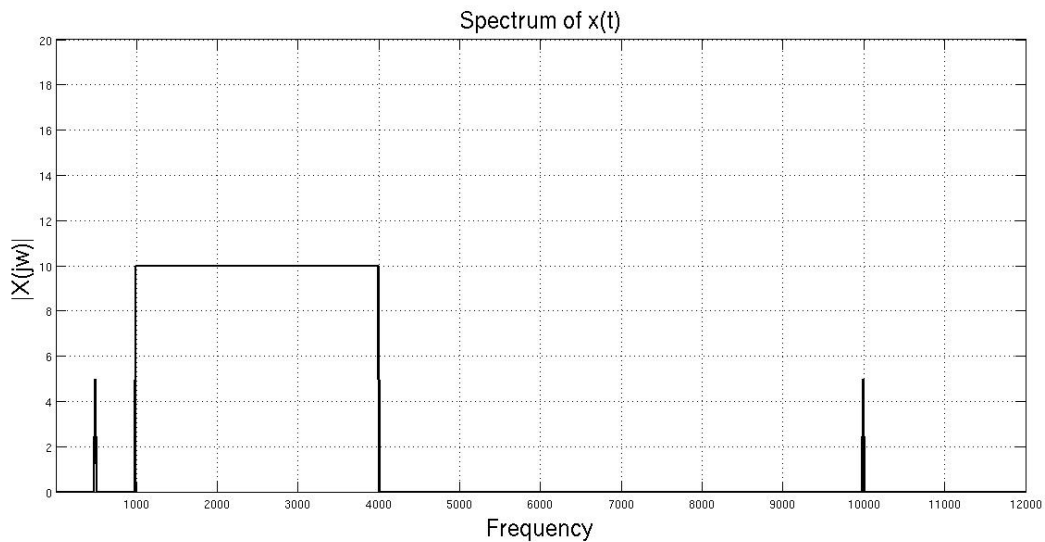
Name: _____

Problem 3 (15 points): Signal Processing

You are asked to design a signal processing chain with typical blocks as given below:



The maximum sampling rate of the ADC is limited to 12kHz. The frequency domain spectrum of the input $x(t)$ is as shown below. The desired signal is in the 1kHz - 4kHz band. There are interferers at 500Hz and 10kHz. You can assume that there are no other signals (noise, etc).



- a) Choose a sampling rate (f_s) for the ADC ($< 12\text{kHz}$) and give the cut-off frequencies of the analog low-pass filter (LPF) and of a digital filter of your choice such that $x_2[n]$ has minimal components of the interferers.

ADC f_s	
Analog LPF cutoff	
Digital filter response type & cutoff	

- b) Why do we need the analog filter?

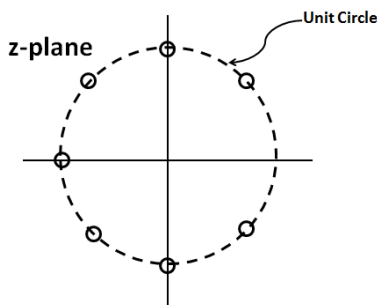
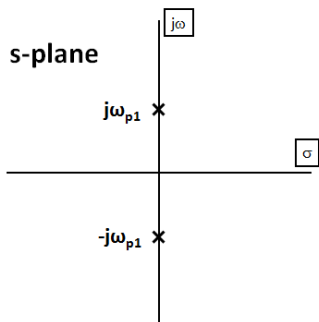
Name: _____

- c) Is there a way to design the system without using an analog filter? If so, give a range of ADC sampling frequencies that can be used so that your interferer doesn't destroy your signal. Pick a sampling frequency and give the cut-off frequencies for the digital filter such that there are minimal interferer components in $x_2[n]$. Hint: From the sampling theorem, it follows that after sampling, $X(f+nf_s) = X(f)$, where n is an integer. Hence, $X(f_s/2 + f) = X(f_s/2 - f)$, i.e. the sampled signal is symmetrical around $f_s/2$.

Possible f_s range	
ADC f_s and Digital cutoff	

Problem 4 (10 points): Filter Design and Analysis

- a) Given the following pole-zero plots, sketch the frequency response of the corresponding filters, i.e. the magnitude of the gain of the filter's transfer function over the frequency f . Also indicate whether the filters are analog or digital, and whether they are IIR or FIR.

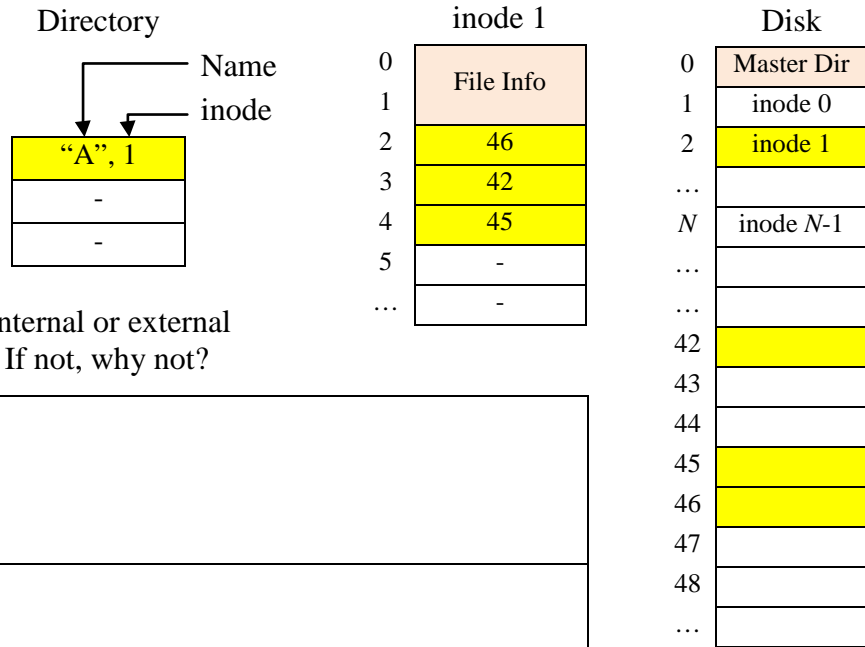


- b) Draw the pole-zero plot of a digital 60Hz notch filter that uses an ADC sampling frequency of 240Hz.

Name: _____

Problem 5 (10 points): Filesystem

In class, we discussed the Unix/Linux filesystem, which uses a form of indexed allocation that stores separate index tables for each file in so-called *inodes* on disk. Each file is associated with exactly one inode given in its corresponding directory entry. Each inode contains meta-information about the file (such as permissions) next to the index table of allocated blocks. Assume that the master directory is stored in disc block 0, followed by up to N inodes. Furthermore, assume that disc blocks have a size of 1kB, that 8 bytes in each inode are used to store meta-information, and that each block index number in the inodes has 32 bits.



a) Does this filesystem have internal or external fragmentation? If so, why? If not, why not?

Internal fragmentation	
External fragmentation	

b) What is the largest supported disk size? What is the largest file that can be created?

Largest disk size	
Largest file size	

c) How could the file system be changed to support larger file sizes?

Name: _____

Problem 6 (20 points): Distributed Barrier

In the midterm, we developed an implementation of a *OS_Barrier()* synchronization between *N* tasks running on the same microcontroller. You are now asked to implement a distributed barrier that synchronizes *N* microcontrollers connected via a single, shared CAN bus. You can assume that the CAN bus will not be used for any purpose other than realizing the distributed barrier. Make any additions or changes to the CAN starter code shown below. Then show the C code for a spinlock realization of the *OS_Barrier()* function. Clearly indicate modifications that are specific to the microcontroller the code is running on. Hint: keep in mind that the CAN is a shared bus that supports broadcast operations where all connected computers, including the sender itself, can simultaneously receive any desired message on the bus.

```
#define N ...

// Message IDs
#define RCV_ID      2
#define XMT_ID      4

// setup object in CAN controller message RAM
void static CAN0_Setup_Message_Object( uint32_t MessageID,
    uint32_t MessageFlags, uint32_t MessageLength,
    uint8_t * MessageData, uint32_t ObjectID, tMsgObjType eMsgType) {
    ...
}

// Initialize CAN port
void CAN0_Open(void) {
    ...

    CANInit(CAN0_BASE);
    CANBitRateSet(CAN0_BASE, 8000000, CAN_BITRATE);
    CANEnable(CAN0_BASE);
    CANIntEnable(CAN0_BASE, CAN_INT_MASTER | CAN_INT_ERROR | CAN_INT_STATUS);

    // Set up filter to receive 4-byte message with RCV_MSG_ID
    CAN0_Setup_Message_Object(RCV_ID, MSG_OBJ_RX_INT_ENABLE, 4,
        NULL, RCV_ID, MSG_OBJ_TYPE_RX);

    NVIC_EN1_R = (1 << (INT_CAN0 - 48)); //IntEnable(INT_CAN0);
    return;
}
```

Name: _____

```
// send 4 bytes of data to other microcontroller
void CAN0_SendMessage(uint8_t *data){
    CAN0_Setup_Message_Object(XMT_ID, NULL, 4, data, XMT_ID,
                              MSG_OBJ_TYPE_TX);
}

// The CAN controller interrupt handler.
void CAN0_Handler(void){
    uint32_t ulIntStatus, ulIDStatus;
    int i;
    tCANMsgObject xTempMsgObject;
    xTempMsgObject.pucMsgData = data;
    ulIntStatus = CANIntStatus(CAN0_BASE, CAN_INT_STS_CAUSE); // cause?
    if(ulIntStatus & CAN_INT_INTID_STATUS){ // receive?
        ulIDStatus = CANStatusGet(CAN0_BASE, CAN_STS_NEWDAT);
        for(i = 0; i < 32; i++){ //test every bit of the mask
            if( (0x1 << i) & ulIDStatus){ // if active, get data
                CANMessageGet(CAN0_BASE, (i+1), &xTempMsgObject, true);
                if(xTempMsgObject.ulMsgID == RCV_ID){

                }
            }
        }
    }
    CANIntClear(CAN0_BASE, ulIntStatus); // acknowledge
}
```

```
void OS_Barrier( )
{
```

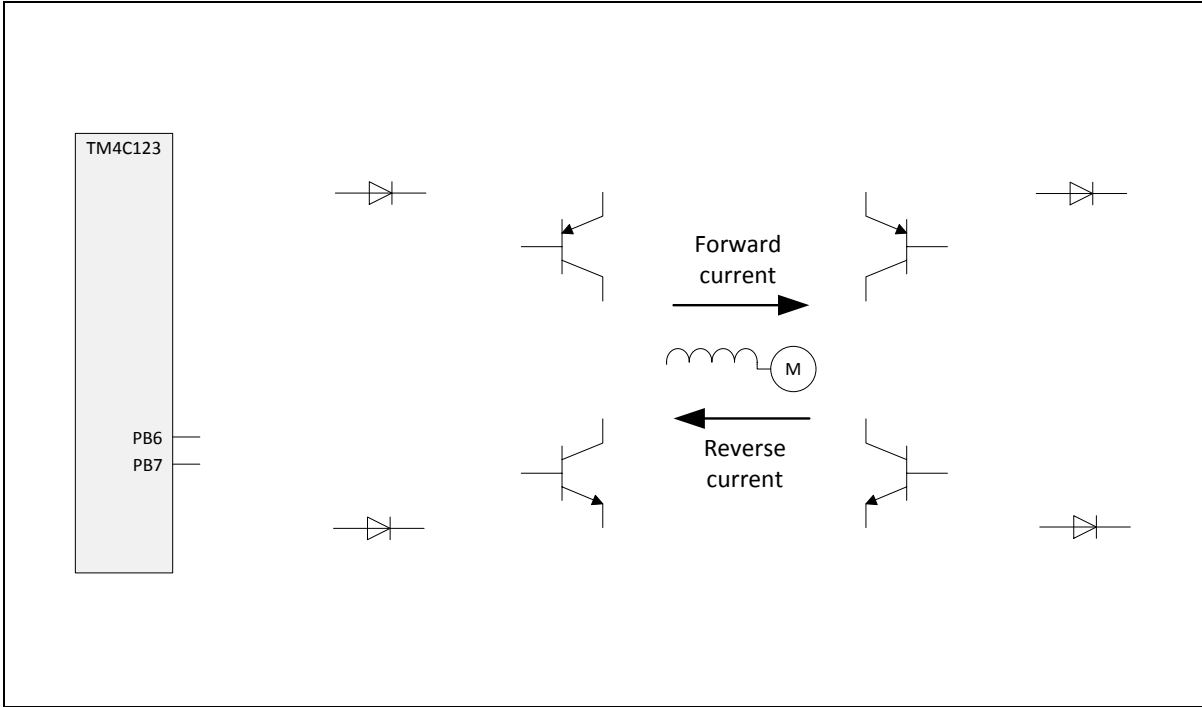
```
}
```


Name: _____

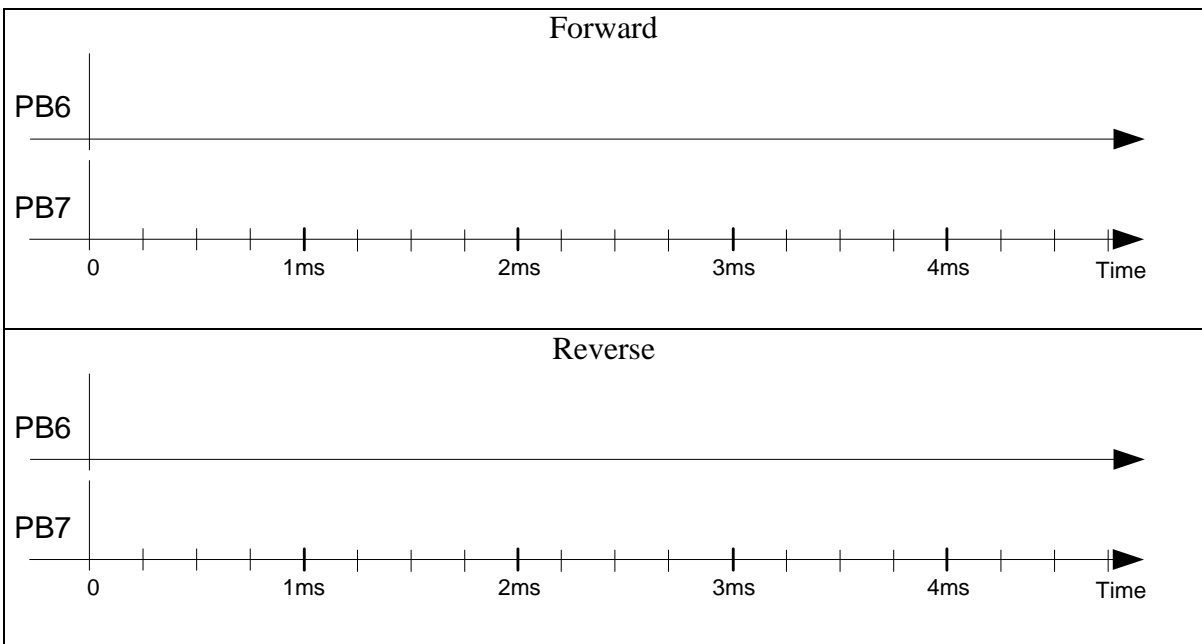
Problem 7 (15 points): Motor Interface

You are asked to design the interface for a DC motor that has a time constant of 40ms.

- a) Design and interface an H-bridge using N- and P-type power BJT/Darlington transistors. Add resistors as necessary, but do not add any other components not already shown.



- b) Show the output signals you need to create on the microcontroller's PB6 and PB7 ports to drive the motor in forward and backward direction at 1/4 of its maximum power. Also indicate (e.g. using dashed lines) how the signals would have to change to go to 3/4 power in either direction.



Name: _____

- c) For any PWM port that you use, what divider should the clock prescaler be set to ($/2$, $/4$, $/8$, etc.) and what period and duty cycle values should the PWM_LOAD and PWM_CMPA registers be loaded with, respectively, to generate the desired signals with maximum resolution? Assume a 80Mhz bus clock.

	$\frac{1}{4}$ speed ahead	$\frac{3}{4}$ speed ahead	$\frac{1}{4}$ speed astern	$\frac{3}{4}$ speed astern
PWM Divider				
PWM_x_LOAD				
PWM_x_CMPA				