Jonathan W. Valvano First Name: _____ Last Name:_____
May 10, 2008, 9am-12n

Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). Please don't turn in any extra sheets.

(5) Question 1. If the CAN channel is noisy, it is possible that some bits will be transmitted in error. Assume there are four nodes, one is transmitting and three are receiving. What happens if a data bit is flipped in the channel due to noise being added into the channel?

(5) Question 2. Assume an analog input PAD0 is being sampled in the background. Data is piped through a FIFO to a foreground thread, and there is a real-time OS with blocking semaphores. We define time-jitter, δt, as the difference between when a periodic task is supposed to be run, and when it is actually run. The goal of a real-time DAS is to start the ADC at a periodic rate, $\Delta t = 1/f_s$. Let t_n be the nth time the ADC is started. In particular, the goal to make $t_n - t_{n-1} = \Delta t$. The jitter is defined as the constant, δt, such that

$$\Delta t - \delta t < t_n - t_{n-1} < \Delta t + \delta t \quad \text{for all } n.$$

Consider the situation where the time jitter is unacceptably large. Which modification to the system will cause the largest improvement in time jitter? Just circle your selection.

A) Run the ADC in continuous mode

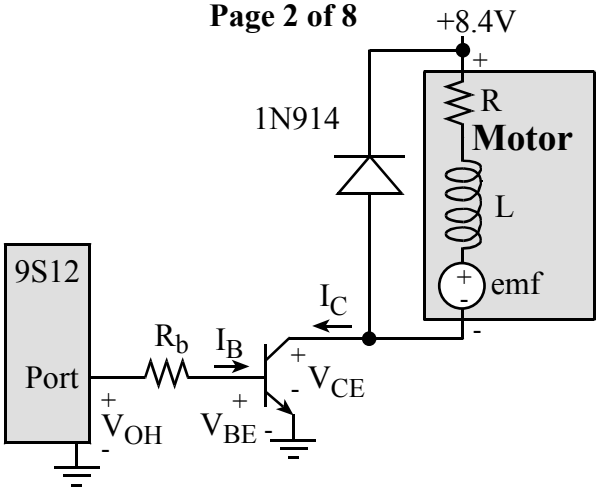
B) Convert from blocking semaphores to spinlock to semaphores

C) Change from round robin to priority thread scheduling

D) Set the ETRIG bit in ATDCTL2, and clear the ETRIGLE ETRIGP bits specifying a falling edge hardware triggered ADC. Use the PWM mode on PT3 of to create a squarewave with a period of Δt, and connect PT3 to PAD7.

E) Increase the time you run with interrupts disabled.

(15) Question 3. Consider the following motor interface. A series-wound DC motor, like the ones in the robot kit, can be modeled by a series combination of a resistor (the ohmic contribution of the long thin wires used to wrap the electromagnets), an inductor (physically resulting from the cylindrical manner in which the electromagnets are wound), and an emf (a voltage caused by the load torque applied from the shaft back into the electromagnet). The resistance is what you get when you measure the resistance of the motor with an ohmmeter (a stopped motor, disconnected from power). In this case, R = 50 Ω and L = 100 μH. Hint: L/R is 2 μsec.



Part a) Under maximum load, the emf becomes –40V and about 1A flows through the motor. What transistor do you choose for this interface (specify part number)? Why?

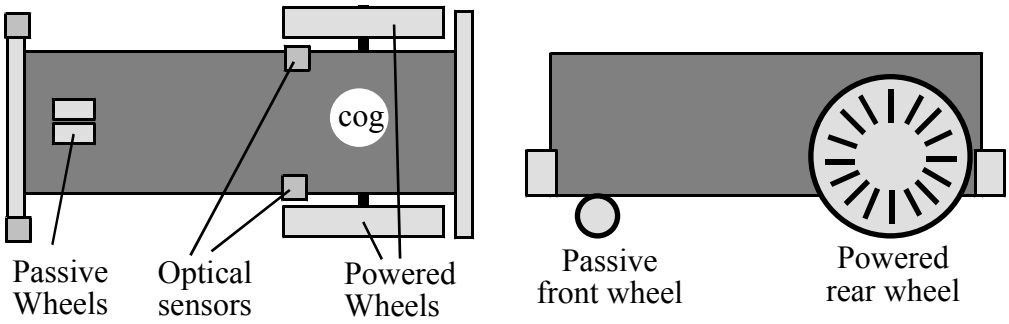
Part b) What value do you choose for R_b? Show your work using the symbols from the picture.

Part c) Initially the motor is off (no current to the motor). At time t=0, the digital port goes from 0 to +5 and transistor turns on. Assume for this section, the emf is zero (motor has no external torque applied to the shaft) and the transistor turns on instantaneously, derive an equation for the motor (I_c) current as a function of time.

(20) Question 4. Consider a robot powered by two DC motors on the rear wheels, but the wheels are not perfectly matched. One wheel has more friction than the other. Experimental analysis shows the left wheel is slower than the right. The goal of this design is to go as fast as possible in a straight line. There are optical sensors on both wheels. There are an equal number of stripes on each wheel. Let **`NumLeft`** be the number of stripes counted on the left wheel. Let **`NumRight`** be the number of stripes counted on the right wheel. The approach will be to operate the left wheel at full speed and adjust the power to the right wheel so that the difference

`error` = **`NumRight`** - **`NumLeft`**

is driven to zero. If **`error`** is positive (right wheel is ahead of left wheel) then reduce power to the right wheel. If **`error`** is negative (right is behind of left) then increase power to the right wheel.



Part a) Assume the left wheel optical sensor is connected to PT0 and the right wheel optical sensor is connected to PT1. The hardware is given. Write an initialization routine and two input capture ISRs that measure **`NumRight`** and **`NumLeft`** in the background.

Part b) Assume the left wheel motor is interfaced to PT2 and the right wheel motor is interfaced to PT3. Essentially there are two copies of the interface circuit shown in Question 3. Again, the hardware is given. You will set PT2 high and use 8-bit PWM on PT3 to adjust power to the right motor. Write an initialization routine that sets PT2=1, and initializes a 100 Hz 8-bit PWM on PT3. Write a separate function that can be called to set the duty cycle of PT3 (0 to 255). Assume a 4 MHz E clock. PT3 does not have to be exactly 100 Hz, just approximately 100 Hz.

Part c) Let **`U`** be the 8-bit duty cycle to the right motor (0 to 255). Assume the time constant of the motor is 500 ms. Develop equations (using integer math) you could use to calculate **`U`** from the measured **`error`**. How often would you execute this equation? If you need some controller constants, define them as **`K1 K2 K3`** etc., specifying the sign of the constant without specifying the actual value. Explain your rationale.

(15) Question 5. Consider the following implementation of a spinlock semaphore used with a round robin preemptive scheduler.

```
void OS_Wait(char *semaPt){
  asm sei       // Test and set is atomic
  while(*semaPt <= 0){  // disabled
    asm cli             // disabled
    asm nop             // enabled
    asm sei             // enabled
  }
  (*semaPt)--;          // disabled
  asm cli               // disabled
}                       // enabled
void OS_Signal(char *semaPt){
  (*semaPt)++;}
```

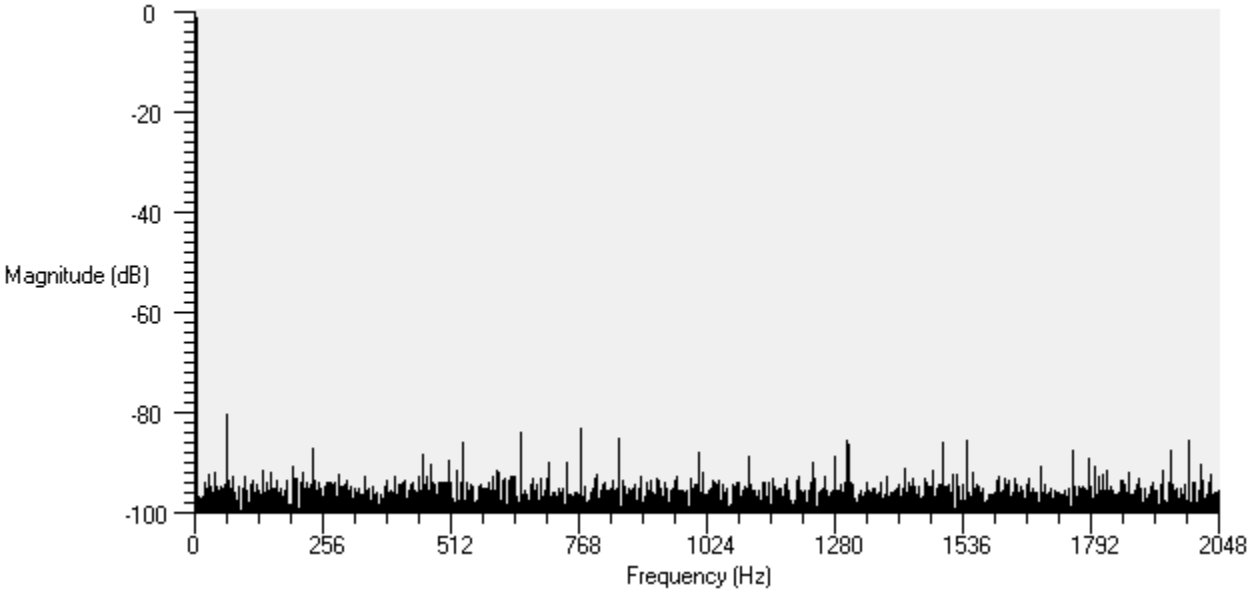
Part a) The compiler generated this code for `OS_Signal`

```
  0000 b745          [1]     TFR   D,X
  0002 6200          [3]     INC   0,X
  0004 3d            [5]     RTS
```

Are there any critical sections in **`OS_Signal`**? I.e., can two threads both call **`OS_Signal`**? Justify your answer.

Part b) Design a new op code for the 9S12, and use it to rewrite **`OS_Wait`** so the spin lock semaphore can be executed without critical sections and without disabling interrupts. Be VERY specific about what the new instruction does (e.g., the `INC 0,X` instruction reads the 8-bit memory value pointed to by Register X, adds one to the value, stores the result=value+1 back into memory, sets the Z bit is the result is 0, sets the N bit if the result is negative, and sets the V bit on a 127+1=128 operation). The 16-bit pointer, **`semaPt`**, will be in Register D when **`OS_Wait`** is called.

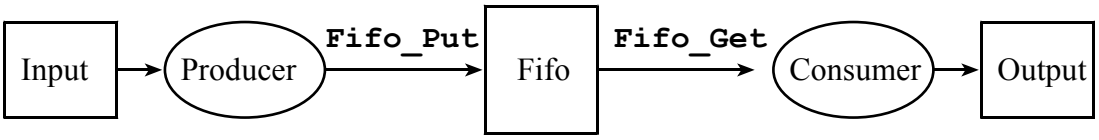
(10) Question 6. In order to measure noise, the sensor on a data acquisition system is removed, and the inputs are grounded. The 10-bit 9S12 ADC is sampled at 4096 Hz, and the collected data are converted to the frequency domain by calculating the FFT. On this scale, -54 db is the same as about 5mV when converted to an equivalent voltage. Similarly, -80 db is the equivalent of 250 μV.



Part a) Is there any noise at all in this data? Justify your answer.

Part b) How would you suggest we redesign the system to improve signal to noise ratio?

(10) Question 7. Consider a producer/consumer problem linked by a FIFO queue. Both the producer thread and the consumer thread operate in the background using interrupt synchronization. The input device is a CAN receiver, and the output device is a SCI transmitter. When the CAN input is ready an interrupt-38 is generated, and the producer thread (CAN input ISR) reads the data and puts them into a FIFO. When the SCI output is idle, an interrupt-20 is generated, and the consumer thread (SCI output ISR) gets data from the FIFO and writes them to the output device.



The CAN protocol is 200,000 bits/sec, 11-bit ID, 8 bytes of data. The SCI protocol is 10000 bits/sec, 8-bit data, 1 start bit, and 1 stop bit. Each CAN message contains 8 bytes of data, and the CAN frames occur according to this repeating pattern (every 1 sec):

Time = 0, CAN message received
Time = 1ms, CAN message received
Time = 2ms, CAN message received
Followed by a 0.998 second pause

Part a) What is the **average input bandwidth** (that actually occurs)? Show your work.

Part b) What is the **maximum output bandwidth** (maximum possible)? Show your work.

Part c) What is the smallest size you could you make the FIFO and not lose data? Justify your answer.

(20) Question 8. Consider a 9S12 system with 1 Mbyte of RAM stored in the $8000 to $BFFF window and accessed using the PPAGE paging system. Each page is 16 kbytes. In this OS, all global variables are allocated into this 1 Mbyte external RAM and shared amongst the user threads. The RunPt, TCB/stacks and other OS variables are located in the internal $3800 to $3FFF RAM (e.g., **`RunPt`** points into $3800-$3FFF). You are designing a preemptive thread scheduler for this system (like Lab 18). What would you do with the PPAGE register on the thread switch? In particular, make edit changes to this scheduler. Justify your answer. The foreground threads use **`Mem_Read`** and **`Mem_Write`** to access global variables

```
struct addr20
{ unsigned char msb;    // bits 19-14
  unsigned short lsw;   // bits 13-0
};
typedef struct addr20 addr20Type;
char Mem_Read(addr20Type addr){ char *pt;
  PPAGE = addr.msb;  // set address bits 19-14
  pt = (char *)(0x8000+addr.lsw); // set addr bits 13-0
  return *pt;        // read access
}
void Mem_Write(addr20Type addr, char data){ char *pt;
  PPAGE = addr.msb;  // set address bits 19-14
  pt = (char *)(0x8000+addr.lsw); // set addr 13-0
  *pt = data;        // write access
}

struct TCB{

  struct TCB *Next;             // Link to Next TCB

  unsigned char *StackPt;       // Stack Pointer

  unsigned char TheStack[96];   // stack

};
typedef struct TCB TCBType;
typedef TCBType * TCBPtr;
TCBPtr RunPt;  // Pointer to thread currently running
interrupt 11 void threadSwitchISR(void){
asm ldx RunPt

asm sts 2,x

  RunPt = RunPt->Next;

  TC3 = TCNT+1000;  // Thread runs for a unit of time

  TFLG1 = 0x08;     // acknowledge by clearing TC3F

asm ldx RunPt

asm lds 2,x

}
```