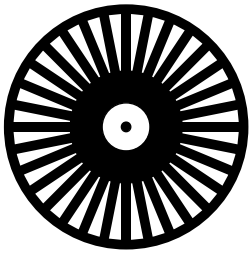
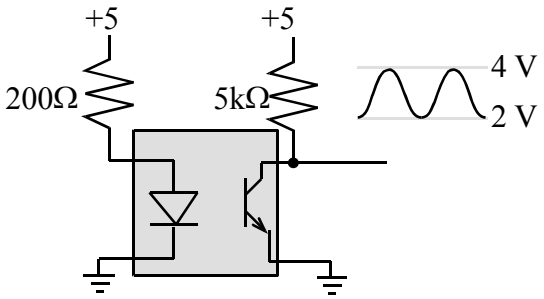
Jonathan W. Valvano First Name: _____ Last Name:_____
December 18, 2006, 9am-12n

This is an open book, open notes exam. You must put answers on these papers; please don't turn in any extra sheets.



(20) Question 1. The goal is to measure the speed of a rotating shaft with a resolution of at least 0.1 rpm using input capture on PT7. A white/black disk is placed on the shaft and an optical sensor is located near the disk. When the motor is spinning, an oscillating waveform is measured so that there is a minimum voltage of about 2V when the white part is visible and a maximum voltage of about 4V when the black part is visible. There are 32 lines.

Part a) Interface this sensor to the 9S12C32. Specify chip numbers and resistor values as needed. The 74HC14 can not be used, because the voltage range is too high, instead consider using a rail-to-rail op amp to create a CMOS logic compatible signal.



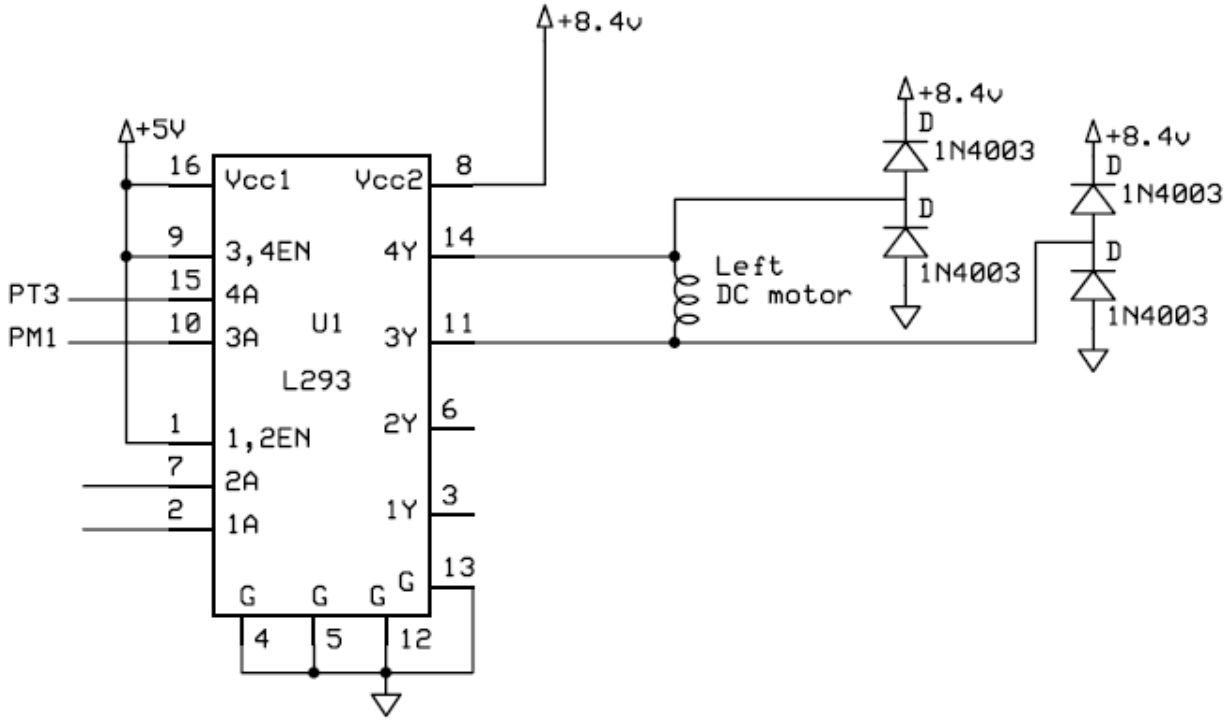
Part b) Write the software required to measure shaft speed. You may assume the maximum speed is 120 RPM or 2 rps. Include an initializing ritual and ISR(s) which measure the shaft speed quickly and accurately. Maintain a global variable with the most recent speed measurement.

(20) Question 2. The current to a DC motor depends on both the applied voltage and its mechanical load. The following data was collected from the DC motors we used in Lab 30

Condition	Voltage	Current	Speed
No load	9V	35 mA	150 RPM
Normal load	9V	100 mA	90 RPM
Stalled	9V	over 200 mA	0 RPM

When the robot strikes a wall or another robot, either the wheels will spin or the motor will stall. Design a hardware/software system that detects a stalled motor, by measuring the delivered current to the motor in a minimally intrusive manner. The overall goal is to set a global variable, called **`bStall`** to 1 if the current exceeds 200 mA.

Part a) The following shows a typical DC motor interface. Modify it so that there is an analog signal that is proportional to the motor current can be measured. Connect that signal to PAD0. You only need to make it work spinning forward (4Y=8.4V and 3Y=0V). Be careful to limit the ADC input voltage to levels between 0 to +5V. You can use a 1 Ω ½ watt resistor.



Part b) Write the software required to detect a stalled motor. Use a periodic output compare ISR that measures the current every 10 ms and sets the **`bStall`** =1 if the current is greater than 200 mA. You may use **`ADC_Init`** and **`ADC_In`** without showing the code, but all other software is required.

(**5) Question 3.** Consider a situation where two 9S12C32 systems are connected with a CAN network. Computer 1 generates 8-bit data packets that must be sent to computer 2, and computer 2 generates 8-bit data packets that must be sent to computer 1. The packets are generated at random times, and the goal is to minimize the latency between when a data packet is generated on one computer to when it is received on the other. Describe the CAN protocol you would use

11-bit versus 29-bit ID
Number of bytes of data
Bandwidth

Clearly describe what is in the ID and how the data is formatted.

(5) Question 4. The following multithreaded system has a critical section. Modify these programs to remove the error. You may assume the RTI interrupts are enabled and are running. The **unsigned long** data type is a 32-bit integer.

```
unsigned long Time;

void main(void){
  while(1){
    if(Time == 1000000){
      SCI_OutString("done");
    }
  }
}
```

```
void interrupt 7 handler(){
  PTT |= 0x01;
  CRGFLG = 0x80;   // ack
  Time++;
  PTT &= ~0x01;
}
```

(20) Question 5. This problem investigates the design of an adaptive priority scheduler with exponential time slices. This is also called an **exponential Queue** or **multi-level feedback queue**. The CTSS system (MIT, early 1960's) was the first to use exponential queues. One of the difficulties in a priority scheduler is the assignment of priority. Typically one wishes to assign a high priority to threads doing I/O (which block a lot) so that the response to I/O is short, and assign a low priority to threads not doing I/O (which do not block a lot). However, in a complex system a particular thread may sometimes exhibit I/O bound behavior, but later exhibit CPU bound behavior. An adaptive scheduler will adjust the priority according to the current activity of the thread. Priority 1 threads will run with a time slice of 4000 (1ms), priority 2 threads will run with a time slice of 8000 (2ms), and priority 3 threads will run with a time slice of 16000 (4ms). Consider this blocking round-robin scheduler, with two new entries, shown in **bold**, added to the TCB.

```
struct TCB{
  struct TCB *Next;     // Link to Next TCB
  char *StackPt;        // Stack Pointer
  Sema4Type *BlockPt;   // 0 if not blocked, pointer if blocked
  short Priority;       // 1 (highest), 2, or 3 (lowest)
  unsigned short TimeSlice; // 4000,8000, or 16000
  char Stack[100];      // stack, size determined at runtime
};
typedef struct TCB TCBType;
typedef TCBType * TCBPtr;
```

Part a) Rewrite the **OS_Wait** function so that if a priority 2 or 3 thread blocks, its priority will be raised (decrement by 1) and its time slice will be halved. No changes to **OS_Signal** will be needed.

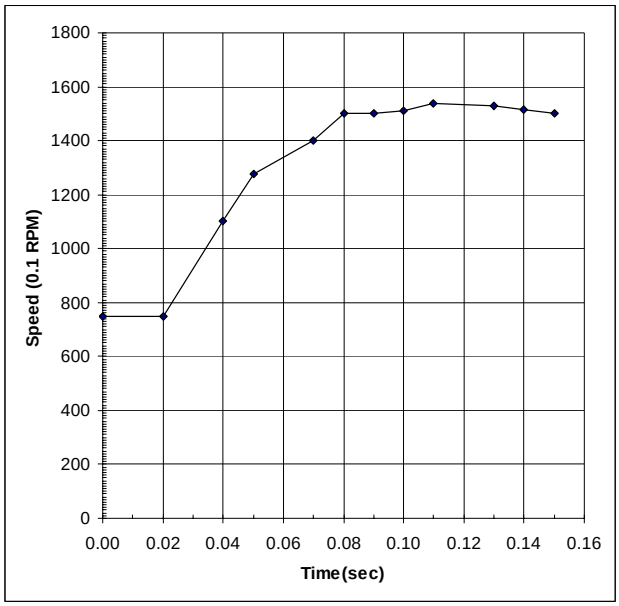
```
void OS_Wait(Sema4Type *semaPt){
  asm sei      // Test and set is atomic
  (semaPt->Value)--;
  if(semaPt->Value <=0){
    RunPt->BlockPt = semaPt; // block this thread
    TC3=TCNT+15;             // stop running this thread
  }
  asm cli
}
```

Part b) Rewrite the `threadSwitch` ISR so that if a priority 1 or 2 thread runs to the end of its time slice without blocking, its priority will be lowered (increment by 1) and its time slice will be doubled. In addition, implement priority scheduling with variable time slices.

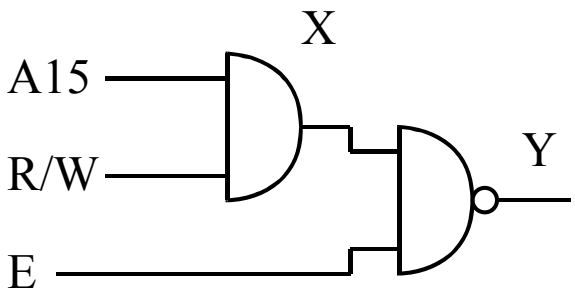
```
// this is a round robin scheduler with fixed timeslices
interrupt 11 void threadSwitch(void){
asm ldx RunPt
asm sts 2,x
  RunPt = RunPt->Next;
  while(RunPt->BlockPt){
    RunPt = RunPt->Next;  // don't run blocked threads
  }
  TC3 = TCNT+4000;       // Thread runs for a unit of time
  TFLG1 = 0x08;          // acknowledge by clearing TC3F
asm ldx RunPt
asm lds 2,x
}
```

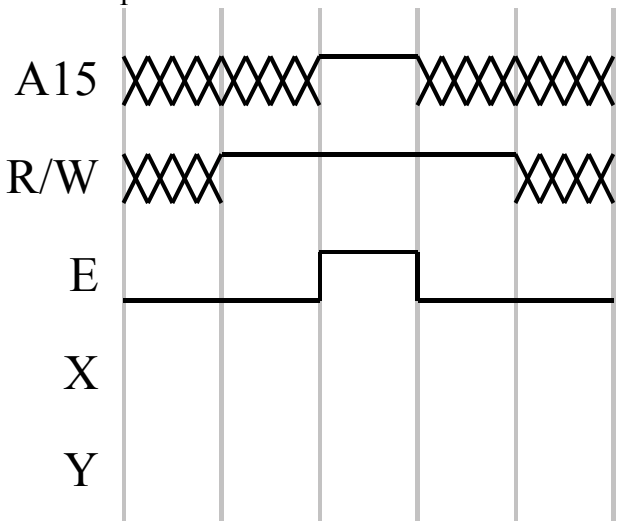
(10) Question 6. The Lab 30 DC motor will be controlled using a 9S12C32 system. The speed is measured using 16-bit input capture and has a measurement resolution of 0.1 rpm. The input capture device driver repeatedly updates a global variable, called **`Speed`**. This 16-bit unsigned variable has units of 0.1 rpm and a range of 0 to 1600 (meaning 0.00 to 160.0). The 9S12C32 uses 8-bit pulse-width modulation to control power to the motor. The controller software writes to a global variable, called **`Duty`**, which ranges from 0 (0%) to 250 (100%). The following plot shows an experimental measurement obtained when **`Duty`** is changed at time=0 from 100 to 250. The desired speed is stored in the global variable, **`Desired`**, which has the same units as **`Speed`**. Design a fixed-point PI controller, using process reaction curve approach, first proposed by Ziegler/Nichols and Cohen/Coon in 1953, that takes **`Speed`** and **`Desired`** as inputs and calculates **`Duty`** as an output. Show your work. How often should the controller be executed? Show just the equations (no software or hardware is required), calculating **`Duty`** as a function of **`Speed`** and **`Desired`**. This data was collected from the DC motors we used in Lab 30

Time (sec)	Speed (0.1 RPM)	Slope (0.1RPM/sec)
0.00	750	
0.02	750	0
0.04	1100	17500
0.05	1275	17500
0.07	1400	6250
0.08	1500	10000
0.09	1500	0
0.10	1510	1000
0.11	1537	2683
0.13	1530	-347
0.14	1514	-1597
0.15	1503	-1063



(5) Question 7. Consider the following digital circuit connected to a microcomputer bus in expanded mode. Assume a 10 ns gate delay through each gate. You may assume the signal E, A15, and R/W as shown in the timing diagram are actual outputs from the microcomputer. Draw the timing signals for X,Y. Use arrows to signify causal relationships. Each vertical line is 100ns.





(5) Question 8. The 6812 is running in expanded mode with 512K of extended program page RAM. One particular virtual address is page number $5, byte number $126.
Part a) Fill in the two boxes in the following software that reads this location.

```
PPAGE = [          ];
data = *((char *)([          ]));
```

Part b) What is the physical memory address of this byte?

(10) Question 9. A D flip-flop clocks its **Data** input on the rising edge of its **Clk**. For this chip, the setup time is 50 ns, and the hold time is 20 ns. A system using this flip-flop provides valid information to the **Data** input during this interval:

Data = ([80,120], [220,240])

and the rising edge of the **Clk** occurs at

↑**Clk** = [160,190]

Will the data be properly stored? Use timing analysis to justify your answer.