Jonathan W. Valvano First Name: _______________ Last Name:____________________
May 14, 2011, 2-5pm
Closed book part

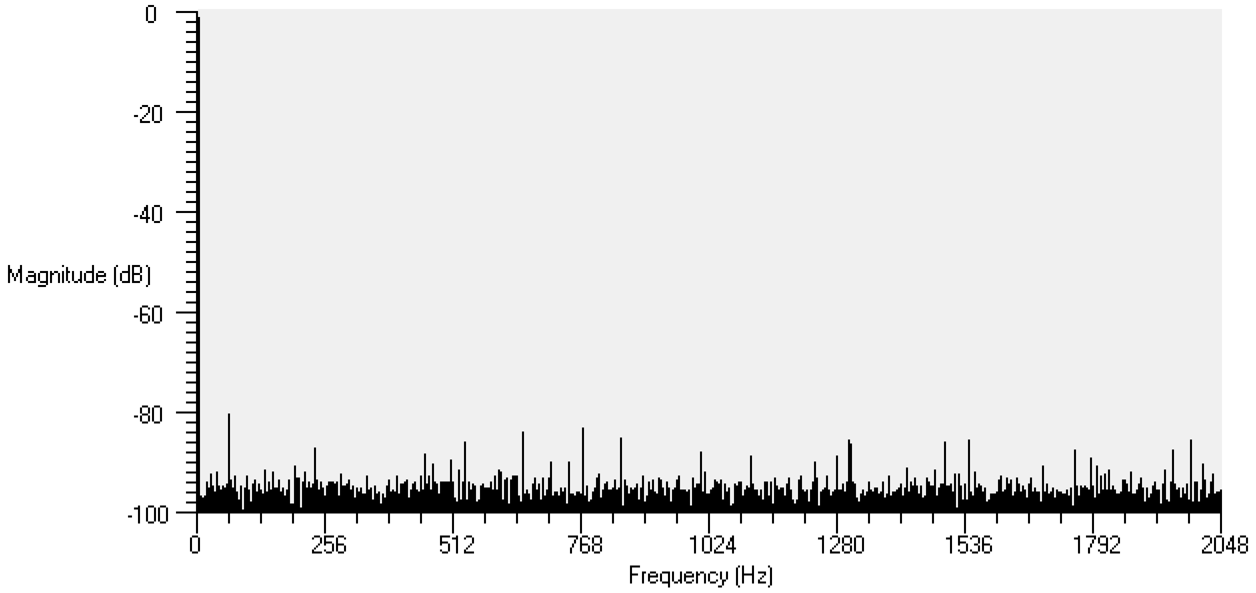
(30) Question 1. For each definition, select the term that best describes it. Not all words will be used. Aging, Aliasing, Anti-Reset-Windup, Bank-Switched Memory, Big Endian, Board Support Package, Bounded Waiting, Burst DMA, Cooperative Multi-Tasking (Nonpreemptive Scheduler) , Crisp Input, Critical Section, Cycle Steal DMA, Deadlock, Q, Dual Address DMA, Dual Port Memory, Stepper Motor, External Fragmentation, Full Duplex Channel, Half Duplex Channel, Hook, Internal Fragmentation, Little Endian, Membership Sets, Minimally Intrusive, Mutual Exclusion, Path Expression, Phase Response, Pole, Priority Inversion, Reentrant, Priority Scheduler, Requirements Document, Rotor, Semaphore, Servo, Simplex Channel, Single Address DMA, Slew Rate, Stabilization, Stator or Frame, Stuff Bits, Velocity Factor, Zero

	A condition when the largest file or memory block that can be allocated is less than the total amount of free space on the disk or memory.
	A DC motor with built in controller. The microcontroller specifies desired position and the motor adds/subtracts power to move the shaft to that position.
	A formal description of what the system will do in a very complete way, but not including how it will be done. It should be unambiguous, complete, verifiable, and modifiable.
	A memory module that interfaces to two separate address/data buses, and allows both systems read/write access the data.
	A memory module with two memory chunk that interfaces to two separate address/data buses. At any given time one memory chunk is attached to one address/data bus the other chunk is attached to the other.
	A place in the frequency domain where the filter gain is zero.
	A place in the frequency domain where the filter gain is infinite.
	A scenario that occurs when two or more threads are all blocked each waiting for the other with no hope of recovery.
	A scheduler that can not suspend execution of a thread without the thread's permission. The thread must suspend itself.
	A set of software routines that abstract the I/O hardware such that the same high-level code can run on multiple computers.
	A software module that can be started by one thread, interrupted and executed by a second thread.

	A software technique to guarantee subfunctions within a module are executed in a proper sequence. For example, it forces the user to initialize I/O device before attempting to perform I/O.
	A technique used in priority schedulers that temporarily increases the priority of low priority treads so they are run occasionally.
	An I/O synchronization scheme that transfers an entire block of data all at once directly from an input device into memory, or directly from memory to an output device.
	An I/O synchronization scheme that transfers data one byte at a time directly from an input device into memory, or directly from memory to an output device.
	An indirect function call added to a software system that allows the user to attach their programs to run at strategic times. These attachments are created at run time and do not require recompiling the entire system.
	An input parameter to the fuzzy logic system, usually with units like cm, cm/sec, °C etc.
	Direct memory access that requires only one bus cycle to transfer data from an input device into memory, or from memory to an output device.
	Direct memory access that requires two bus cycles to transfer data from an input device into memory, or from memory to an output device.
	Establishing an upper bound on the magnitude of the integral term in a PID controller, so this term will not dominate, when the errors are large.
	Fuzzy logic variables that can take on a range of values from true (255) to false (0).
	Hardware that allows bits (information, error checking, synchronization or overhead) to transfer only in one direction. Contrast with half duplex and full duplex channels.
	Locations within a software module, which if an interrupt were to occur at one of these locations, then an error could occur (e.g., data lost, corrupted data, program crash, etc.)
	Mechanism for storing multiple byte numbers such that the least significant byte exists first (in the smallest memory address). Contrast with big endian.
	Storage that is allocated for the convenience of the operating system but contains no information. This space is wasted.
	The maximum slope of a signal. If the time-varying signal V(t) is in volts, it is the maximum dV/dt in volts/s.

	The part of a motor that remains stationary.
	The ratio of the speed at which information travels relative to the speed of light.
	Thread synchronization where at most one thread at a time is allowed to enter.
	When digital values sampled at $\mathbf{f_S}$ contain frequency components above ½ $\mathbf{f_S}$, then the apparent frequency of the data is shifted into the 0 to ½ $\mathbf{f_S}$ range.

(5) Question 2. In order to measure noise, the sensor on a data acquisition system is removed, and the inputs are grounded. The 10-bit LM3S8962 ADC is sampled at 4096 Hz, and the collected data are converted to the frequency domain by calculating the FFT. On this scale, -60 db is the same as about 3mV when converted to an equivalent voltage. Similarly, -80 db is the equivalent of 300 μV.



Is there any noise at all in this data? If no, justify your answer. If yes, what kind of noise

(5) Question 3. You are designing a real-time scheduler for this system. There are four periodic tasks that have minimal interaction with each other.

Task 1:	Executes 1000 times/sec, execution time varies from 5 to 20µs.
Task 2:	Executes 500 times/sec, execution time varies from 10 to 100µs.
Task 3:	Executes 2000 times/sec, execution time varies from 1 to 10µs.
Task 4:	Executes 1000 times/sec, execution time varies from 10 to 100µs.

Without actually writing the scheduler, you can determine whether or not a solution is likely. Is it possible to schedule these tasks? If no, prove it. If yes, justify your answer.

(3) Question 4. This digital filter estimates power from voltage measurements, assuming a constant impedance R: y(n) = x(n)*x(n-1)/R. Derive the Z-transform of this digital filter. This is a multiply not convolution. The ADC is 10 bits, and the sampling rate is 1 kHz. If it can't be done, explain why.

(5) Question 5. Explain how **paging** can improve security in a multi-process operating system.

(12) Question 6. Write C code for a mailbox that can be used to pass 32-bit data between foreground threads. None of the mailbox functions will be called from the background. Do NOT implement a FIFO. There will be a single producer and a single consumer both running in the foreground. You can define and initialize semaphores simply by adding globals:

```
long semaphore=0;
```

You may call the following two blocking semaphore functions without showing their implementation.

```
void OS_Wait(long *semaPt);
void OS_Signal(long *semaPt);
```

Add the `static` qualifier to private components, and no `static` for public components. A producer thread should block on full, and a consumer thread should block on empty.

Jonathan W. Valvano First Name: _______________ Last Name:_____________________

Open book part

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.

(6) Question 7. Consider three communication channels: UART with 1 start, 8 data bits and 1 stop bit, CAN, and Ethernet (not TCP/IP). An error occurs during transmission (one bit is reversed).

A) Neither transmitter or receiver notice

B) The receiver hardware notices, but the transmitter hardware does not notice

C) The transmitter hardware notices, but the receiver hardware does not notice

D) Both the receiver and transmitter hardware notice, and the transmitter retransmits automatically

Part a) For a UART, what happens? Select A B C or D

Part b) For a CAN, what happens? Select A B C or D

Part c) For Ethernet, what happens? Select A B C or D

(10) Question 8. Consider a file system with indexed allocation. The disk size is 2^{30} bytes, and there is an index table for each file. Assuming you wish to have no external fragmentation, what is the smallest block size that allows the entire index table to exist completely in a one block?

(4) Question 9. Consider the following motor interface from the book.

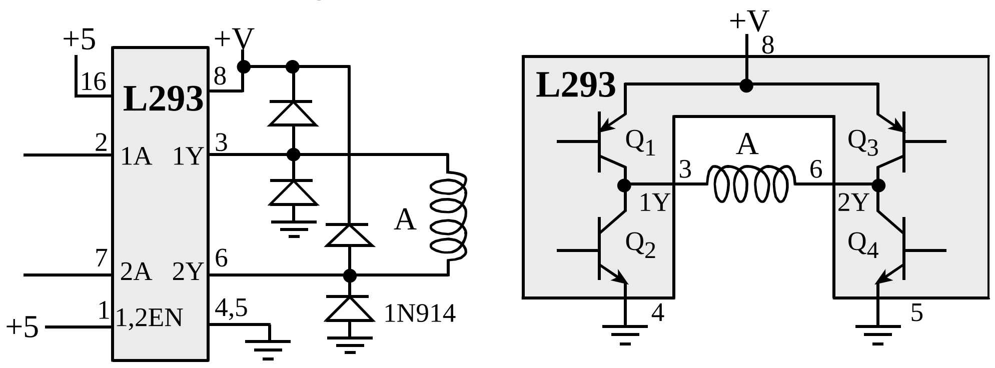


Figure 8.12. An H-bridge can drive current in either direction (the actual L293 uses all N-channel devices).

electrical characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = −1 A		V_{CC2}−1.8	V_{CC2}−1.4		V
V_{OL}	Low-level output voltage		I_{OL} = 1 A			1.2	1.8	V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	µA
		EN				0.2	±10	
I_{IL}	Low-level input current	A	V_I = 0			−3	−10	µA
		EN				−2	−100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

Assume input 1A is high. Assume Vcc2 (pin 8) is 8.4V. From the L293 data sheet the output voltage 1Y will be 8.4-1.4, which is 7V. Assume input 2A is low. From the data sheet the output voltage 2Y will be 1.2 V. This means the applied voltage across the motor will be 7-1.2, which is only 5.8V. Actual measurements with a L293 and the motors from the robot kit show a voltage drop across the DC motor much higher than 5.8V (close to 7V). Why?

(20) Question 10. Consider an operating system like the solution to Lab 2 (round robin scheduler with spinlock semaphores). You will implement the following two functions

```
void OS_Wait_Event_And(unsigned long mask);
void OS_Trigger_Event(TCBPtr Thread, unsigned long bits);
```

Each thread has 32 flag bits that it can use to synchronize with other threads. Assume we are thread 2, and our TCB is `tcbs[2]`. A simple example application of this feature uses one bit creating a binary semaphore. To wait on an event, we call

```
  OS_Wait_Event_And(0x000000001); // wait until bit 0 becomes 1
```

Another thread can signal us by calling

```
  OS_Trigger_Event(&tcbs[2], 0x00000001); // set bit 0
```

In this usage, our call to `OS_Wait_Event_And` will wait for the bit 0 to be set, then we clear bit 0 and continue. The call to `OS_Trigger_Event` by the other thread will set bit 0.

A more complicated usage involves multiple bits. To wait on three events, we call

```
  OS_Wait_Event_And(0x00000111); // wait until bits 8,4,0 become 1
```

Thread3 can signal us by calling

```
  OS_Trigger_Event(&tcbs[2], 0x00000001); // set bit 0
```

Thread4 can signal us by calling

```
  OS_Trigger_Event(&tcbs[2], 0x00000010); // set bit 4
```

Thread5 can signal us by calling

```
  OS_Trigger_Event(&tcbs[2], 0x00000100); // set bit 8
```

In this usage, our call to `OS_Wait_Event_And` will wait for bits 8,4,0 to be set, then we clear the three bits and continue. Each call to `OS_Trigger_Event` will set one of the bits. However it is possible for one of the other threads to trigger all three bits by executing

```
  OS_Trigger_Event(&tcbs[2], 0x00000111); // set bits 8,4,0
```

Part a) Each thread has its own TCB and stack. The threads are in a circular linked list. This list is static and all threads are initially ready to run. Make changes to this TCB to accommodate the new feature. You may assume the `OS_AddThread` function will initialize this new field to zero when the thread is created.

```
struct TCB {
  long *stackPointer;  // pointer to top of stack
  unsigned long Id;    // thread number, zero if this TCB is free
  struct TCB *Next;    // TCBs are in a circular linked list



};
typedef struct TCB TCBType;
typedef TCBType * TCBPtr;
TCBPtr RunPt;     // Pointer to tcb of thread currently running
TCBType tcbs[10]; // static allocation of tcbs
```

Part b) Show the spinlock implementation of **`OS_Wait_Event_And`**. To improve efficiency you should execute **`OS_Suspend`** during the spin loop. After a completion of the wait, clear the bits being waited for (**`mask`**). Call **`OS_DisableInterrupts() OS_EnableInterrupts()`** as needed.

```
void OS_Wait_Event_And(unsigned long mask){
```

Part c) Show the spinlock implementation of **`OS_Trigger_Event`**. Since this is a spinlock implementation, simply make changes to the TCB. Do not spin, block, suspend, or kill in this routine. Each of the 32 bits in **`bits`** specifies a separate trigger. Another thread could trigger all our flags by calling with the **`bits`** field 0xFFFFFFFF. Notice because we pass a thread parameter, the trigger event is sent to a specific thread (not broadcast to all threads like an **`OS_Signal`**).

```
void OS_Trigger_Event(TCBPtr Thread, unsigned long bits){
```