Jonathan W. Valvano
May 14, 2011, 2-5pm
Closed book part

(30) Question 1. For each definition, select the term that best describes it.

external fragmentation	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.
Servo	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.
requirements document	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.
dual port memory	A memory module that interfaces to two separate address/data buses, and allows both systems read/write access the data.
bank-switched memory	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.
zero	A place in the frequency domain where the filter gain is zero.
pole	A place in the frequency domain where the filter gain is infinite.
deadlock	A scenario that occurs when two or more threads are all blocked each waiting for the other with no hope of recovery.
cooperative multi-tasking nonpreemptive	A scheduler that can not suspend execution of a thread without the thread's permission. The thread must suspend itself.
Board Support Package	A set of software routines that abstract the I/O hardware such that the same high-level code can run on multiple computers.
reentrant	A software module that can be started by one thread, interrupted and executed by a second thread.
path expression	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.
aging	A technique used in priority schedulers that temporarily increases the priority of low priority treads so they are run occasionally.
burst DMA	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.
cycle steal DMA	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.
Hook	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.
crisp input	An input parameter to the fuzzy logic system, usually with units like cm, cm/sec, °C etc.

single address DMA	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.
dual address DMA	Direct memory access that requires two bus cycles to transfer data from an input device into memory, or from memory to an output device.
anti-reset-windup	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.
membership sets	Fuzzy logic variables that can take on a range of values from true (255) to false (0).
simplex channel	Hardware that allows bits (information, error checking, synchronization or overhead) to transfer only in one direction. Contrast with half duplex and full duplex channels.
critical section	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.)
little endian	Mechanism for storing multiple byte numbers such that the least significant byte exists first (in the smallest memory address). Contrast with big endian.
internal fragmentation	Storage that is allocated for the convenience of the operating system but contains no information. This space is wasted.
slew rate	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.
Stator or frame	The part of a motor that remains stationary.
velocity factor	The ratio of the speed at which information travels relative to the speed of light.
mutual exclusion	Thread synchronization where at most one thread at a time is allowed to enter.
aliasing	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. There is no noise because these signals are below the 3mV resolution of our ADC.

(5) Question 3. A real-time scheduler for this system. Use the Rate Monotonic Theorem. Add up maxTime/TimeInterval for each
20/1000 + 100/2000 + 10/500 + 100/1000 = 0.02+0.05+0.02+0.10 = 0.19 or 19%.
Since this is well below ln2, a solution should exist.

(3) Question 4. No Z-transform exists for this digital filter because it is nonlinear. There is a way to calculate this (but beyond this class). The Z-transform of x1(n)*x2(n) involves a closed path circular integration. It is not convolution.

(5) Question 5. Each process has its own **page table.** The page tables are switched as the OS suspends one process and launches another. This way one process cannot access the data in another process. If you wish to support sharing, then both processes will have page table entries for the shared information.

(12) Question 6. Write C code for mailbox used to pass 32-bit data between foreground threads.

```
long static  MailBoxValidData=0;
long static MailBoxBoxFree=1;
// ******** OS_MailBox_Send ************
// enter mail into the MailBox
// Inputs:  data to be sent
// Outputs: none
void OS_MailBox_Send(unsigned long data){
  OS_Wait(&MailBoxBoxFree);     // wait for the MailBox to be free
  MailBox = data;
  OS_Signal(&MailBoxValidData);
}
// ******** OS_MailBox_Recv ************
// enter mail into the MailBox
// Inputs:  none
// Outputs: data received
unsigned long OS_MailBox_Recv(void){ unsigned long mail;
  OS_Wait(&MailBoxValidData);        // wait for Mail
  mail = MailBox ;
  OS_Signal(&MailBoxBoxFree);         // box is free
  return mail;
}
```

(6) Question 7. An error occurs during transmission (one bit is reversed).
Part a) UART A) Neither transmitter or receiver notice
Part b) CAN D) Both the receiver and transmitter hardware notice, and the transmitter retransmits
Part c) Ethernet B) The receiver hardware notices, but the transmitter hardware does not notice

(10) Question 8. What is the smallest block size that allows the index to exist in a one block?
First simplification, each entry is a 32-bit number (4 bytes)
Let 2^n be the block size.
It can hold 2^{n-2} index entries.
Each index entry represents one block in the file, max file size is $2^n * 2^{n-2} = 2^{2n-2}$
Since there is no external fragmentation the max file size should be 2^{30}, $2^{2n-2} \geq 2^{30}$, $n \geq 16$
Double check first assumption, block size is 65536 bytes and there are 16384 blocks (no)

Second pass, each entry is a 16-bit number (2 bytes)
Let 2^n be the block size.
It can hold 2^{n-1} index entries.
Each index entry represents one block in the file, max file size is $2^n * 2^{n-1} = 2^{2n-1}$

Since there is no external fragmentation the max file size should be 2^{30}, $2^{2n-1} \geq 2^{30}$, $n \geq 15.5$
We get the same answer, n=16, block size is 65536 bytes and there are 16384 blocks

(4) Question 9. Consider the motor interface from the book. These are Darlington transistors, Vce is a function of current. The data sheet shows the Vce at maximum current 1A. We are running much less current, so the voltage drop is much less.

(20) Question 10. You will implement two functions
Part a) Make changes to this TCB to accommodate new feature.

```
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
  long Flags;          // 32 event flags
};
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) The spinlock implementation of `OS_Wait_Event_And`. It is similar to spinlock `OS_Wait`.

```
void OS_Wait_Event_And(unsigned long mask){
  OS_DisableInterrupts();     // Test and set is atomic
  while(RunPt->Flags&mask != mask){ // continue if all true
    OS_EnableInterrupts();
    OS_DisableInterrupts();
  }
  RunPt->Flags = RunPt->Flags&(~mask);       // disabled
  OS_EnableInterrupts();   // disabled
}
```

Part c) Spinlock implementation of `OS_Trigger_Event`. It is similar to spinlock `OS_Signal`.

```
void OS_Trigger_Event(TCBPtr Thread, unsigned long bits){
long sr;
  OSCRITICAL_ENTER()        // start Critical Section
  Thread->Flags |=  bits;   // set triggers
  OSCRITICAL_EXIT()         // end critical section
}
```