UNIVERSITY OF ALBERTA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CMPE 401 – Computer Interfacing

Final Examination Solutions

Instructor:	B. F. Cockburn
Exam date:	December 10, 2008
Exam duration:	120 minutes
Aids permitted:	A hardcopy of the course overheads can be freely consulted. Electronic calculators (all kinds) are permitted. Model solutions for past assignments and exams are **not** permitted.

Instructions:
1. Enter your printed name, signature and I.D. number on this cover page.
2. Verify that this booklet contains 10 pages (including this cover page).
3. Neatly enter your answers in the spaces provided.
4. Use the reverse sides of the pages for extra space or rough work.

Student name: ____________________,________________
Last name **First name**

Signature: ____________________________________

Student I.D.: ____________________________________

Question	Time	Worth	Mark	Subject
1.	12	10		Fundamental Concepts
2.	22	18		Real-time Multitasking
3.	18	15		UART and FEC Interrupts
4.	14	12		TCP/IP Stacks
5.	22	18		Layered Communications Interfaces
6.	18	15		Stepper Motor Control
7.	14	12		Direct Memory Access
Total	**120 mins**	**100**		---

Question #1 (Fundamental Concepts)

In your own words briefly define each of the following concepts. Be sure to explain why each concept is important in Computer Interfacing.

(a) Context switch [4 marks]

A context switch is the sequence of operations, performed by the kernel or operating system, that changes the task (or process) that is currently executing on the CPU. The operations include saving the context of the current task (e.g. the contents of CPU registers) into a data structure (e.g. a task control block), loading the context of the new task (loading the CPU registers from the TCB of the new task), and then restarting execution of the new task.

Context switches are important in Computer Interfacing because it is common to design large software systems as a collection of loosely interacting tasks. Context switches are required to allow the tasks to share the CPU (or the small number of available CPUs). The time to perform a context switch is a critical parameter that should be minimized since it represents overhead time (e.g. an implementation cost) that cannot be used by the tasks.

(b) Single-address DMA [3 marks]

Single-address direct memory access (DMA) is the process of transferring data between a source (or destination) at one fixed address in the memory map to a destination (or source) range of addresses. The single address usually corresponds to a data register in a peripheral controller. The range of addresses usually corresponds to addresses of locations in a buffer region in memory.

Direct memory access is important because it is a common way of transferring blocks of data within a computer system. DMA transfers are fast and efficient because they are not implemented as a large number of MOVE instructions that need to be separately fetched, decoded and executed. Instead DMA transfers are performed by a DMA controller circuit that produces the necessary system bus signals that move the data over the bus without requiring the execution of instructions on the CPU. (The CPU is still required to initialize and configure the DMA controller, and also to respond to hardware interrupts that might be produced by the DMA controller.)

(c) Determinism [3 marks]

Determinism is the property of having a relatively predictable response time (e.g. a response time that is restricted to a narrow range of possible values) to external events. The response time is ideally independent of system load.

Determinism is important in real-time systems since it makes is easier to ensure that real-time events will be handled sufficiently fast and in a predictable way.

Question #2 (Real-time Multitasking)

(a) Briefly explain what the major advantages and disadvantages of cooperative multitasking would be compared to pre-emptive multitasking. What requirements in the overall real-time problem would need to be considered when choosing between these two forms of multitasking? Comment on which of these two forms of multitasking would be more appropriate for (i) ensuring deterministic behaviour and (ii) making most efficient use of the available CPU clock cycles.

[2 marks] *Advantages of cooperative multitasking compared to pre-emptive multitasking:*

- *Each task, once it starts running, can run for a predictable block of time before giving up the CPU.*
- *The number of context switches can be effectively controlled by the way the tasks are designed.*
- *The kernel's design can be simplified a little bit since checks do not have to be included to ensure that task priorities are respected after every system event.*

[2 marks] *Disadvantages of cooperative multitasking compared to pre-emptive multitasking:*

- *The tasks are more tightly coupled to each other since each task, once it starts running, prevents all other tasks from running.*
- *It places a greater burden on the software designer to ensure that the system constraints (e.g. real-time response to external events) are met. The software design cannot rely on a relatively simple and independently enforced task priority mechanism.*

[2 marks] *Requirements that would need to be considered:*

- *The experience of the software design team. Will they be able to meet the real-time constraints if they take over direct control of task context switching?*
- *The overall complexity of the software system. More complicated systems will likely be easier to design using prioritized pre-emptive multitasking. If the system is simple then co-operative multitasking will be able to benefit from less overhead from the reduction in the number of context switches.*

[2 mark] *To ensure deterministic behaviour it would be advisable, in most cases, to use pre-emptive multitasking because the kernel (or operating system) independently enforces a simple task priority scheme. Task priorities are a relatively simple way of ensuring that the system behaviour is deterministic.*

[2 mark] *To ensure the most efficient use of the available CPU cycles one would probably prefer to use cooperative multitasking since this software architecture would minimize the number of task context switches (and hence minimize one kind of time overhead).*

Question #2 (Real-time Multitasking, cont'd)

(b) Briefly explain how each of the following MicroC function calls can cause a context switch: (i) OSChangePrio(), (ii) OSTimeDly(), (iii) OSSemPost(), and (iv) OSSemPend(). Clearly indicate the conditions under which a context switch will occur in each case.

[8 marks, 2 marks for each function]

OSChangePrio() can cause a context switch if it lowers the priority of the currently running task to below the priority of another ready-to-run, but currently waiting, task. Once the priority of the currently running task is lowered, the kernel will perform a context switch to swap in the formerly waiting, high-priority task.

OSTimeDly() can cause a context switch whenever the time delay is greater than zero. This is because the function requests that the currently running task be suspended for the specified period of time (given as a number of hardware timer ticks).

OSSemPost() will possibly cause a context switch if at least one other task exists that is higher in priority and this is currently pending on the same semaphore. If the semaphore count is at least 1 (after being incremented) and there exists a higher priority task that was blocked on the semaphore, then a context switch will occur to the highest priority such task.

OSSemPend() will cause a context switch to the next highest priority ready-to-run task if the corresponding semaphore has a count value of zero or less (after being decremented).

Question #3 (UART and FEC Interrupts)

(a) Both the Universal Asynchronous Receiver Transmitter (UART) and the Fast Ethernet Controller (FEC) modules in the MCF5234 microcontroller contain interrupt status registers (UISRn and EIRn) and interrupt mask registers (UIMRn and EIMRn). Briefly describe how the status and mask registers are used together to determine when interrupt signals are sent back to the MCF5234 Interrupt Controller Modules (and hence back to the ColdFire CPU).

[4 marks] *In both the UART and FEC the contents of the interrupt status register and the interrupt mask register are bit-wise ANDed together. If the result of any of these AND operations is a 1, then the hardware interrupt signal is asserted and sent back to the Interrupt Controller Modules, and hence possibly back to the CPU.*

An active hardware interrupt will stay active until either (1) the mask bits for all active interrupt status bits are written to 0, or (2) all interrupt status conditions are cleared (often by writing 1's to the appropriate bits to the status register).

(b) In the EIRn there are separate bits, RXF and RXB, for two different receiver interrupt conditions. Briefly describe how these two conditions are different. Why do you suppose that these interrupts were provided separately instead of providing a single receive interrupt? Give a simple example to illustrate your answer.

[3 marks] *The Receive Frame Interrupt (RXF) bit is set when the last receive buffer in an incoming Ethernet frame has been fully received.*

[3 marks] *The Receiver Buffer Interrupt (RXB) bit is set when the next receive buffer, which is not the last in the frame, has been fully received.*

[5 marks] *These two interrupts have likely been provided separately to simplify the gathering of statistics (e.g. number of buffers filled vs. number of frames received) and also because the interrupt handling is likely to be different in the two cases. When the last buffer in a frame is received, then frame-level processing will likely occur, processing that would not be required for any of the earlier received buffers in the same frame.*

Question #4 (TCP/IP Stacks)

In the lectures we considered two different implementations of a TCP/IP stack for the MicroC multitasking kernel environment: a recent implementation from NetBurner and an older freeware stack called the Lightweight IP (lwIP). NetBurner's stack uses four separate tasks for the network hardware driver, the IP layer, the TCP layer, and the network application layer; by contrast, lwIP combines all four layers into one task and it uses its own complex buffer management system. Given that more recent microcomputers have much more available memory and run faster, give a reasonable justification for the stack configuration that was chosen by NetBurner. What are the major advantages of the stack architecture chosen by NetBurner?

[6 marks]
The NetBurner stack architecture is going to require greater computing resources than the lwIP stack since, unlike lwIP, multiple tasks are required in the implementation, and an optimized stack buffer management system is not used. More modern microcomputers have greater memory capacity and faster code execution speed. These greater performance characteristics can be sacrificed to gain greater quality with respect to the software architecture including the greater software modularity afforded by having multiple tasks, more strictly enforced boundaries between different protocol layers, and more strictly defined boundaries between the operating system, the stack, and the application tasks.

[6 marks]
The major advantages of the NetBurner stack architecture include:

- *Separate tasks to handle the network hardware driver, the IP layer, the TCP layer and the network application. As a result these layers can be designed, verified and modified relatively independently compared to an architecture where the layers are mixed together in one task.*
- *The priorities of the various tasks in the stack can be adjusted, just like regular tasks, to suit system performance objectives.*
- *A complex custom buffer data structure and buffer management system is not used.*

Question #5 (Layered Communications Interfaces)

(a) In the lectures we discussed the advantages and disadvantages of layering in a communications interface with respect to the stack implementation in the communicating computers. But there is also impact on the efficiency (that is, the faction of communicated bits that are actually useful application data bits) with which the communication medium is used. What do you suppose would be some of the main disadvantages of layering a communications interface with respect to this measure of efficiency? Consider the following two cases: (i) data is being produced in fixed size blocks, and (ii) data is being produced as a continuous stream of bytes.

[4 marks]
Layering a communications interface would reduce the efficiency with which the communication medium is used if each layer contributes its own overhead in terms of packet headers and/or trailers, and poor attention is paid to minimizing duplication of information in all of the various overhead fields. Layering would also likely introduce overhead since it would likely introduce more procedure calls, each adding their own overhead in terms of execution time and storage space.

[2 marks]
In case (i), when data is produced in fixed size blocks those blocks may not conveniently match the payload sizes of the available communication packets. If the data is produced in very small blocks, then space is wasted when a packet is sent with much of the payload area unused. If the data is produced in large blocks, then space may be wasted if the data does not exactly fit the payload space in a whole number of data packets.

[2 marks]
In case (ii), where data is being produced as a continuous stream of bytes, then the payload area in the available packets can be filled up to their full capacity, which would be achieving maximum possible efficiency. However, the communication link would experience an undesirable communication delay that was due to the need to fill up data packets instead of sending the data in smaller pieces as soon as it is received at the transmitting end.

Question #5 (Layered Communications Interfaces, cont'd)

(b) The Internetworking Protocol (IP) is allowed to split up a TCP segment into smaller segments before creating IP datagrams. This might be a useful thing to do in order to ensure that the IP datagrams are small enough to be handled by the physical layer protocol. Consider the problem of splitting up a given TCP segment into two equal-length shorter segments. How would the IP software determine the values to use in the following header fields: (i) the source port, (ii) the destination port, (iii) the sequence number, (iv) the acknowledgement number, (v) the segment checksum, and (vi) the data field for the two new segments?

[10 marks]

(i) *The source port: The two new segments should use the same source port value as the original segment.*

(ii) *The destination port: The two new segments should use the same destination port value as the original segment.*

(iii) *The sequence number: The sequence number of the first new segment should be the same as the sequence number of the original segment, but the sequence number of the second new segment should be calculated as the sum of (1) the sequence number of the original segment and (2) the number of data bytes in the first new segment. This sum will be the sequence number of the first data byte in the second new segment.*

(iv) *The acknowledgement number: The acknowledgement number used in the original segment should be used as the acknowledgement number in both of the two new segments.*

(v) *The segment checksum: New segment checksums will need to be calculated for the two new segments. The checksum of the original segment is of no use.*

(vi) *The data field: The data field of the original segment should be partitioned into two equal halves which are then used as the data fields of the two new segments.*

Question #6 (Stepper Motor Control)

(a) A stepper motor is conveniently controlled by purely digital signals, where the stator windings are either electro-magnetized in one of two possible directions, or are not magnetized at all. Stepper motors are in fact a type of synchronous motor. The ideal current waveform in a stepper motor would have the shape of a sinusoid (e.g. sine or cosine) over time. Using sinusoidal-shaped current waveforms would avoid the strong vibration experienced by stepper motors when they are operating. In the space below, draw the full-step current waveforms (with current on the vertical axis and time increasing along the horizontal axis) for a two-phase stepper motor. Forward current is to be shown with positive values, and reverse current is to be shown with negative values. Show how these waveforms are actually rough approximations to two sinusoidal waveforms that are 90 degrees out of phase.

[5 marks]

```
     ^  Forw              Forw              Forw
Ph A +--------+        +--------+        +--------+
     |        |        |        |        |        |
     +        +--------+        +--------+        +----> t
                  Rev               Rev             Rev

     ^For            Forw              Forw
Ph B +---+        +--------+        +--------+        +->t
     |   |        |        |        |        |        |
     +   +--------+        +--------+        +--------+
            Rev               Rev               Rev
```

(b) Now show that the half-step control waveforms are also rough approximations to two sinusoids that are 90 degrees out of phase. Indicate on your waveforms the current configurations that correspond to the new half-stepped positions. With the aid of your diagram, explain why the total rotor current varies with the position of the stepper motor rotor.

[5 marks]

```
     ^  Forw                    Forw
Ph A +--------+Off        Off+--------+Off        Off+-
     |        +--+        +--+        +--+        +--+
     +           +--------+              +--------+
                     Rev                     Rev

     ^For                 Forw
Ph B +--+Off        Off+--------+Off        Off+----> t
     |  +--+        +--+        +--+        +--+
     +     +--------+              +--------+
               Rev                     Rev
```

[5 marks] Total rotor current is lower when Ph A or Ph B is off.

Question #7 (Direct Memory Access)

Consider the scenario where CPU-requested dual-address DMA is going to be performed using one of the three DMA controllers in an MCF5234 microcontroller. The source and destination devices are external memories, connected to different chip selects, that happen to have different data port widths. Specifically, the source device is 16 bits wide (connected to bits #31-16) and the destination device is 32 bits wide. Briefly explain the origin, the connecting path, and the destination of each of the following signals: (i) the source address; (ii) the destination address; (iii) the read/write signal; (iv) the chip select signals; and (v) the data signals.

(i) [2 marks] *The source address originates at the one designated DMA controller, where it is incremented by two for every 16-bit read operation from the source memory. The address is driven out of the MCF5234 over the external address bus to the address inputs of 16-bit source memory.*

(ii) [2 marks] *The destination address originates at the one designated DMA controller, where it is incremented by four for every 32-bit write operation to the designation memory. The address is driven out of the MCF5234 over the external address bus to the address inputs of the 32-bit destination memory.*

(iii) [2 marks] *The read/write signal originates at the one designated DMA controller. It alternates in value between being high (read signal) during read operations from the source memory and being low (write signal) during write operations to the destination memory. The signal is connected to the R/W inputs of the source and destination memories over the R/W line of the external control bus.*

(iv) [3 marks] *The two chip select signals (one for each memory device) are produced by the Chip Select Module in the MCF4234, which monitors the address on the system bus to determine when to assert (low) the two active-low chip select (CSn) signals. During initialization the two used CS signals are programmed to be associated with the two ranges of addresses in the microcontroller's memory map that correspond to the two memories.*

(v) [3 marks] *The data signals originate at the data port of the source memory, and during the read operation they travel over the external and internal data busses to the data buffer of the designated DMA controller. During the write operation the data signals travel from the data buffer over the internal and external data busses to the data port of the destination memory.*