# UNIVERSITY OF ALBERTA

## DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

## ECE 315 – Computer Interfacing

## Midterm Examination

| | |
|---|---|
| Instructor: | B. F. Cockburn |
| Exam date: | March 5, 2014 |
| Exam duration: | 50 minutes |
| Aids permitted: | A paper or electronic copy of the course lecture slides can be freely consulted.<br>The Internet cannot be accessed by any device.<br>Electronic calculators of any kind are permitted.<br>The model solutions for this year's Assignments #1 and 2 can be consulted.<br>Your own solutions to Assignment #1 can be consulted. |

Instructions:

1. Please enter your printed name, signature and I.D. number on this page.
2. Verify that this booklet contains 6 pages (including the cover page).
3. Neatly enter your answers in the spaces provided.
4. Use the reverse sides of the pages for rough work.
5. Take into account the marks per question when budgeting your time.

**Student name:** _____Model___,___Solutions____
**Family name** **Given name**

**Signature:** _____

**Student I.D.:** _____

| Question | Time | Worth | Mark | Subject |
|---|---|---|---|---|
| 1. | 8 | 15 | | Fundamental Concepts |
| 2. | 18 | 35 | | Multitasking Software |
| 3. | 12 | 25 | | Microcomputer Hardware |
| 4. | 12 | 25 | | Computer Networking |
| **Total** | **50 mins** | **100** | | **—** |

**Question #1 (Fundamental Concepts)**

(a) When is it safe to add two pointers, and when is it safe to subtract two pointers in a C program? Be sure to illustrate your answer with a short example.

[3 marks] It is never safe to add two pointers. Each pointer is a separate address offset into main memory, and adding two pointers produces a meaningless address. Access such an address could lead to catastrophic system behaviour.

[3 marks] It is safe to subtract two pointers only if both pointers are of the same type and if they point to objects of that type in a one 1-D array of those objects. Subtracting two such pointers will give the difference in the array indices of the two objects that are being pointed to. For example, one could have an array A[0], ..., A[99] of 100 int's. One could initialize int pointers intPtr1 and intPtr2 as intPtr1 = &A[5] and intPtr2 = &A[32]. Then the difference intPtr2 – intPtr1 would yield 32 – 5 = 27, regardless of the size of the objects.

(b) How is an interrupt status register used in conjunction with an interrupt mask register inside a microcomputer peripheral subsystem?

[3 marks] In peripheral systems that can produce interrupts, it is common to pair up an interrupt status register (say, ISR) with an interrupt mask register (say, IMR) of the same bit length. The bits in the ISR record which potentially interrupt-producing conditions are currently true. The bits in the IMR select which bits in the ISR are capable of producing interrupt signals that leave the peripheral subsystem. Essentially, ISR and IMR are bit-wise ANDed with each other, and if the result is nonzero then the external interrupt is asserted.

(c) What is the difference between "fast real-time response" and "determinism" in the context of externally generated input signals handled by an embedded system?

[3 marks] Fast real-time response refers to minimizing the delay between the moment when an external signal arrives at the embedded system, and the time that the embedded system changes external signals to respond to the input change.

[3 marks] Determinism refers to minimizing the variability in the real-time response to externally generated inputs. It is easier to design a predictable system, and thus it is easier to meet system response specifications, if the system has a greater degree of determinism. With greater determinism, relatively large response time delays are often more manageable.

**Question #2 (Multitasking Software)**

(a) What is a counting semaphore, and for which situations is it useful? What is the appropriate initial value of a counting semaphore?

[8 marks] A counting semaphore is an object that is used to control access to a pool of shared resources. A task requests access to one of the resources by pending on the semaphore, at which time it will be blocked if no resources are available. The task posts to the semaphore when it finishes using the assigned resource. The semaphore includes a count variable that records the presently available number of units of the shared resource. Negative values of the count are used to record the number of tasks that are currently blocked by the semaphore because of the present lack of free resources. The semaphore also includes a queue that records any tasks that are currently blocked on the semaphore.

[2 marks] The initial value of a counting semaphore should be the initial number of free resources in the pool.

(b) It is recommended that tasks in a pre-emptive multitasking system have a relatively simple structure, based either on an infinite loop or based on a series of simple interconnected states. In both cases, all possible execution paths through the loop or through each state must end with a blocking function. Briefly explain why these blocking functions must be present. What would happen if some of them were absent?

[10 marks]
All possible execution paths through each task's software must encounter a call to a blocking function that will sometimes cause the task to give up the CPU to allow other lower priority tasks to run.

If these blocking functions are not present in a task, or if there are execution paths that never have calls to blocking functions, then the CPU can be hogged by that task if it ever gets a chance to start running. If this were to happen, system failure would likely result because no lower priority tasks would get a chance to run, and the functions that they control would not get performed.

**Question #2 (Multitasking Software, cont'd)**

(c) Interrupts are provided in embedded systems to ensure that hardware events are handled promptly? However, in a multitasking system interrupts can be handled by either (i) using an interrupt service routine (ISR) only, or (ii) using a combination of an ISR and a task that is unblocked through a semaphore by the ISR. In what kinds of situations would you choose to use method (i) over method (ii)? On the other hand, in what kinds of situations would you prefer to use method (ii) over method (i)? Be sure to give the most relevant factors.

[15 marks]

Situations where method (i) would be most appropriate:

If the event handling operation is relatively short, requires the fastest possible response, requires the greatest determinism, and does not require the use of routines in the task environment, then all of the event handling should be done in the interrupt service routine (ISR). This would give the fastest possible response and greatest possible determinism in the event handling, and it would minimize the time overhead of additional task context switches.

Situations where method (ii) would be most appropriate:

If the external event is relatively low in priority, if it does not need fast response time or the best possible determinism, or if it requires the use of functions that are best provided to a running task, then using the combination of an ISR that unblocks a task that handles most if not all of the event would be the preferred approach. This method is also attractive if there are many interrupt sources in the system, and you may want to ensure the fastest possible response for some of them by using method (i), and then use method (ii) for the less time-sensitive interrupts.

The most important factors:

- The need for the fastest possible response time to events
- The need for the best possible determinism
- The need to access functions from the task environment

**Question #3 (Microcomputer Hardware)**

(a) The different major blocks in the MCF5234 Microcontroller Unit (MCU) are connected together functionally using an on-chip system bus. In addition, the MCF5234 provides pins that provide an off-chip extension to the system bus. What is the purpose of this external system bus? In the laboratory microcomputer, how is this external system bus used?

[6 marks] The external system bus is used to effectively connect additional subsystems to the system bus to enhance the capabilities of the microcomputer. Those additional subsystems might include read-only memory (ROM), random-access memory (RAM), and additional peripheral interfaces.

[6 marks] In the MOD5324 microcomputer that is used in the course laboratory, the external system bus is used to connect a flash memory chip and a DRAM chip to the MCU. The flash memory chip is used to store the MicroC/OS operating system and the user code; the DRAM is used for data storage.

(b) The laboratory microcomputer uses a Micrel Ethernet PHY chip in combination with the Freescale MCF5234 Microcontroller Unit (MCU). Control and status information is exchanged between the two chips using the MDC (clock) and MDIO (data) lines. An unusual protocol is used where a 32-bit data frame is shifted serially from the MCU to the PHY, and then back from the PHY to the MCU. What is the reason why this data exchange protocol over two pins was used in this particular application? Why do you think the external extension bus to the MCF5234 was not used?

[6 marks] The data exchange protocol is used here to allow the MCU to access control and status registers inside the PHY. The control registers are configured at the time that the system is initialized, and might be updated later if the configuration needs to be altered for some reason. The status registers can be read to gather more detailed system status information. The serial interface is convenient because it requires only two pins. This reduces the cost and the slower serial operation has no performance impact on normal operation.

[7 marks] The extension bus was not used to connect the MCU to the PHY because the MCU-PHY connection is already standardized using the MDIO industry standard. The external MCU bus is specific to the MCU family, and it is not compatible with the MDIO industry standard. Using the extension bus to connect to the PHY would be expensive since the extension bus requires many more pins that the MDIO standard bus.

**Question #4 (Computer Networking)**

(a) The early computer network standards, such as X.25, used a combination of telephone connections (both digital and analog) to communicate data packets from source to destination nodes through a network of switching nodes. In these standards, the data link protocols provided features that guaranteed reliable data transmission for each node-to-node hop along a data packet's route. In modern computer networking protocols, such as TCP/IP, the end-to-end TCP protocol takes care of providing reliable data transport. Why was it necessary in the older data networks to provide reliable node-to-node packet transport, whereas in modern TCP/IP data networks, reliability at the node-to-node level is limited to the checksum that is provided in the IP header? What would happen if TCP/IP were to be used over the older networks?

[10 marks] The older networks used telephone connections, which are much less reliable than modern data network connections. It made sense to ensure reliability at the node-to-node level to avoid passing along corrupted data through the network. The cost of ensuring reliability from node-to-node was less than the cost of having very poor end-to-end data transport.

In modern TCP/IP networks, reliability is provided on an end-to-end basis because node-to-node errors are relatively rare. Providing reliability on a node-to-node basis would rarely find and correct errors, and so would be wasted effort most of the time. Checksums in the IP header increase chances of delivering the datagram, but avoid wasting time doing repeated checksums on the much larger IP payload.

[5 marks] If TCP/IP were to be used on older telephone line based networks, then the data throughput would be low since errors would been seen frequently, and the network would tend to get bogged down with repeated end-to-end retransmission attempts.

(b) What is the purpose of the 11-state finite state machine in a TCP interface? Why is it convenient to partition the TCP control software into states?

[10 marks] The 11-state finite state machine in implementations of TCP is conveniently used to structure the TCP control software according to the state machine in the TCP protocol standard. The state changes are triggered by discrete events, such as the arrival of TCP segments and commands received from the host processor. The eleven states conveniently capture these well-defined control states, and the state-to-state changes. Each state is simple in structure, has easily designed output behaviour. The code that defines the transition into the possible next state(s) is also easy to specify, to design and to verify.