

UNIVERSITY OF ALBERTA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CMPE 401 – Computer Interfacing

Midterm Examination

Instructor: B. F. Cockburn
Exam date: October 27, 2008
Exam duration: 50 minutes
Aids permitted: A hardcopy of the course overheads can be freely consulted.
Electronic calculators are permitted.
Model solutions from this year's assignments are permitted.
Model solutions from previous years are **not** permitted.

- 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** _____
Last name First name

Signature: _____

Student I.D.: _____

| Question | Time | Worth | Mark | Subject |
|--------------|----------------|------------|------|--------------------------------|
| 1. | 8 | 16 | | Basic Acronyms |
| 2. | 12 | 24 | | Multitasking |
| 3. | 15 | 30 | | Computer Networking |
| 4. | 15 | 30 | | Semaphores and Synchronization |
| Total | 50 mins | 100 | | — |

Question #1 (Common Acronyms)

Consider the following four acronyms. For each one give the full spelling, and then briefly explain the significance of the corresponding concept in computer interfacing.

(a) FIFO

[5 marks] FIFO stands for "First-In First-Out" buffer. A FIFO is a memory unit with one write-only port and one read-only port. Data passes into the FIFO at the write port, and emerges later after some delay from the read port. The order of the data passing through a FIFO is preserved, hence the name first-in first-out. FIFOs are widely used at digital-to-digital interfaces to absorb small short-term fluctuations in the output data rate from the transmitting side of an interface, and the input rate into the receiving side. A FIFO allows the two sides of interface to operate in a less tightly coupled way, which simplifies their design.

(b) ADC

[5 marks] ADC stands for "Analog-to-Digital Converter". An ADC is used at analog-to-digital interfaces to allow analog signals (e.g., from sensors in the environment) to be accepted into a digital system (e.g., a microcomputer) for processing, storage and transmission in digital form. ADCs are required in many computer interfaces because the original signals may be created in analog form (say by sensors or gauges), but then those signals must be converted into digital form in order to be manipulated by a digital system.

(c) UART

[6 marks] UART stands for "Universal Asynchronous Receiver / Transmitter". A UART is used to interface a CPU with an RS-232 bidirectional asynchronous serial communications link. Data traveling in the transmit direction is loaded by the CPU into a memory-mapped transmit data register, and is then shifted out serially with the proper framing signals and bit timing. Data arriving serially in the receive direction is extracted out from the incoming frame using recovered timing, and then passed to a memory-mapped receive data register that can be read by the CPU. The UART unburdens the CPU from most low-level details, such as bit-timing, creation and checking of parity bits, detection of errors, etc.

Question #2 (Multitasking)

- (a) Briefly define what is meant by “hard real-time” in the context of Computer Interfacing.

[7 marks] “Hard real-time” refers to a computer interfacing problem where the response time specifications are sufficiently challenging that special operating system software is required. A lightly-loaded conventional computer system (e.g. a Windows PC) will not be adequate.

- (b) Briefly define what is meant by a “multitasking” software architecture.

[7 marks] A “multitasking software architecture” is one in which the application software is partitioning into loosely interacting tasks that share the CPU. Synchronization mechanisms must typically be provided to allow the multiple tasks to safely update and control shared hardware and shared data structures. Some mechanism is also required to determine which single task should be allowed to execute on the CPU at any one time.

- (c) Briefly explain why using a multitasking software architecture might be a useful strategy when designing a computer system that must handle multiple real-time control activities. In your answer, explain how three different activities (one requiring very fast response at unpredictable times, one requiring action at regular time intervals, and one with relaxed time constraints) could be handled successfully by the same system.

[4 marks] Using a multitasking software architecture allow a larger and more complex problem to be partitioned into multiple smaller and simpler problems. The timing constraints of each activity could handled directly by different tasks. Each task is responsible for handling a simpler part of the problem, and therefore the task should be easier to design and to verify on its own.

[6 marks] In the given example, the very fast response could be provided by an interrupt service routine (ISR), plus possible additional code implemented in a high-priority task. The high-priority task could “pend” or block on a semaphore that is “posted” or unblocked by the ISR. The second timed task could meet its timing requirement using programmed time delays, or (better still) by being unblocked to run by a timer-triggered ISR. The activity with relaxed time constraints might be handled by a lower priority task that gets to use up any idle time that is left over from the execution of the two higher priority tasks.

Question #3 (Computer Networking)

- (a) The Internetworking Protocol (IP) governs how data packets, called “datagrams,” are formed and conveyed through the Internet from a source node to a designation node. IP provides a connectionless packet delivery service, but with no guarantee of delivery. IP has the right to break up datagrams in smaller datagrams, or to assemble a larger single datagram from component datagrams. In your own words briefly explain why it is convenient to define a service such as IP, with the above characteristics, when attempting to get a variety of different underlying hardware networks to work together as one network. What underlying complications are solved by the way IP has been defined?

[15 marks] It is challenging to get the various installed data networks to work together as an Internet. The different physical networks (including Ethernet, X.25, IBM’s token ring, and many others) have different packet lengths and formats, different header information, different protocols, and different guarantees on data integrity and delivery. IP provides a common and flexible intermediate layer that bridges the differences among the various underlying physical networks. It provides a new addressing scheme that can be used all over the Internet. IP is flexible in how long the packets need to be. IP packets can be split up or combined, as needed to suit the needs of the underlying networks. IP does not attempt to guarantee packet order or reliable packet delivery (leaving those problems to the transport layer to solve separately).

Question #3 (Computer Networking, cont'd)

- (b) Both the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) lie at an intermediate level, the so-called transport layer, between IP and the application layer. In your own words briefly explain what service enhancements are provided by each of TCP and UDP beyond what is already provided by IP. Why do you think there was a need to define both TCP and UDP instead of a defining a single transport layer protocol?

[10 marks] Both TCP and UDP add port numbers to the IP address. The port number is useful for identifying transport layer connections with applications running on servers and clients. UDP also adds a segment length and an optional segment checksum. TCP adds a much more complicated segment header than UDP. A sequence number and an acknowledgement number are used to verify that bytes traveling in both directions arrive in both directions. A mandatory segment checksum ensures that the data bytes are received without error. Various other flags and fields are provided to support specialized features in TCP (e.g. synchronizing the sequence numbers, provision for an optional urgent pointer to time-critical information in the segment).

[5 marks] A single protocol could have been used in place of both TCP and UDP, but a design decision was apparently made to provide a lightweight (i.e. simplified interface and reduced functionality) transport layer protocol for applications that can tolerate the occasional loss of data (e.g. streaming multimedia) or that involve only a short exchange of messages over a reliable short connection. UDP requires simpler and hence faster processing than TCP, and it is therefore more cost-effective in many applications than TCP.

Question #4 (Semaphores and Synchronization)

The binary semaphore can be used to synchronize events (e.g. interrupt-causing events) that occur in hardware with the activities of software tasks. Consider the following design problem: A communications interface is provided by the hardware that can generate an interrupt signal when a frame of incoming serial data has completely arrived. Assume that the frame contents are stored using a linked list of buffers (a chain of buffers, where each buffer has a pointer to the next buffer in the chain). Two tasks are to be designed. A first task, called BAILER, is to copy the entire contents of the frame into a packed array of bytes, called RXBUF, starting at a given base address. A second task, ADDER, is to arithmetically add up all of the bytes in the frame, and to store the sum in a global variable, called SUM. Describe, using words and point form pseudocode, how the actions of the interrupt service routine, task BAILER and task ADDER could be synchronized using binary semaphores. Be sure to indicate where and to what values the semaphores must be initialized.

[5 marks] Two binary semaphores, called FRAME_RECEIVED and ARRAY_COPIED, must be created and initialized to 0 by the start-up task.

[5 marks] The interrupt service routine (ISR) is executed after an entire frame of data has been received by the hardware. The ISR clears the interrupting condition and then posts to semaphore FRAME_RECEIVED to unblock BAILER.

[12 marks] Routine BAILER has an outer loop that starts off blocked by pending on semaphore FRAME_RECEIVED. Inside the outer loop, BAILER enters an inner loop that copies over the contents of each buffer in the linked list into the packed array with base address RXBUF. The pointer in each buffer must be dereferenced to locate the next buffer in the linked list. The last buffer is recognized by having a NUL pointer. BAILER exits the inner loop once it has copied over the last frame. At the end of the outer loop, BAILER posts to semaphore ARRAY_COPIED. It then goes back to blocking on FRAME_RECEIVED at the start of the outer loop.

[8 marks] Routine ADDER has an outer loop that starts off blocked by pending on semaphore ARRAY_COPIED. Inside the outer loop, an inner loop is entered that adds up the data in the array RXBUF, and then writes the resulting sum into the global variable SUM. It then goes back to blocking on ARRAY_COPIED at the start of the outer loop.

The consumer of the contents of SUM should be alerted some how about when the value has been updated. This detail was not specified in the problem statement.