UNIVERSITY OF ALBERTA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 315 – Computer Interfacing

Final Examination

Instructor: B. F. Cockburn
Exam date: April 27, 2017, 2:00 pm
Exam duration: 120 minutes
Aids permitted: A copy (paper or electronic) of the lecture slides can be freely consulted.
A two-sided 8.5" × 11" formula or summary sheet can be consulted.
No Internet access is permitted using any kind of device.
Electronic calculators (all kinds) and an English dictionary are permitted.

Instructions:
1. Enter your printed name, signature and I.D. number on this cover page.
2. Verify that this booklet contains 9 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: _____Model___,__Solutions____
Last name **First name**

Signature: _____

Student I.D.: _____

Question	Time	Worth	Mark	Subject
1.	15	12		Fundamental Concepts
2.	15	12		Multitasking Software
3.	15	12		Multiple Nonpreemptive Loops
4.	15	12		MicroC/OS
5.	15	13		Serial Communications
6.	15	13		Stepper Motors
7.	15	13		Microcomputer Busses & DMA
8.	15	13		TCP/IP Networking
Total	**120 mins**	**100**		---

Question #1 (Fundamental Concepts)

In your own words briefly explain each of the following concepts. Be sure to explain why each concept is important in embedded microcomputer systems.

(a) Deadlock

[4 marks] Deadlock is a condition that can occur in multitasking systems when a set of tasks gets blocked from proceeding because each task in the set holds at least one resource required by another task, and each task can't proceed because it still requires another resource that is held by another task in the set. Deadlock will typically cause serious system failure. Thus steps must be taken to prevent deadlock from occurring in the first place, or to detect deadlock and to resolve it after it occurs.

(b) Semisynchronous interfaces

[4 marks] A semisynchronous interface is an interface where the timing of each bus transaction is governed by a clock signal together with one or more handshake signals that determine how many whole clock cycles the bus transaction will last. A semisynchronous interface has both the advantages of synchronous timing (and thus simpler design) plus the flexibility in bus transaction duration (which allows data transfers with both faster and slow devices that are connected to the CPU). Most modern computer busses are semisynchronous.

(c) Direct memory access

[4 marks] Direct Memory Access (DMA) is a data transfer method in which a DMA controller circuit is used to efficiently and rapidly transfer blocks of data between two regions in the CPU's memory map. Once a DMA transfer has started, the transfer proceeds without any software interaction with the CPU. However, the system bus must be shared among the CPU and the one or more DMA controllers that may be active at the same time. CPU is the fastest method for transferring all but the smallest volumes of data. DMA controller hardware is required (an additional expense), but DMA hardware is now widely provided in modern microcontrollers.

Question #2 (Multitasking Software)

(a) Partitioning the application software into tasks is usually believed to increase the modularity of the software system. Briefly argue why modularity can be effectively increased by using tasks. Be sure to begin by listing some of the key properties that are associated with high modularity, and then consider how multitasking enhances (or decreases) each of those properties.

[7 marks] A software design has high modularity if the modules (in this case tasks) have the following properties:
(1) High cohesion: each module/task is designed to be concerned with one function on related data.
(2) Information hiding: each module/task should keep its implementation details to itself, and not disclose internal data widely to other modules/tasks.
(3) Loose coupling: modules/tasks interact with each other only when necessary, and they do so using simple mechanisms like semaphores and queues.
(4) Only essential parameters are passed between modules. Thus tasks exchange data using queues and well-protected (e.g., using semaphores) shared data structures.
(6) Global variables should be avoided since they lead to complex and hard to debug interactions. Global variables can be easily avoided by software that is structured into tasks.

(b) Briefly summarize some of the major disadvantages that can arise when partitioning the application software into multiple tasks.

Disadvantages of multitasking include:
(1) Time is wasted in the context switching between tasks.
(2) Memory space is required to store the kernel codes that is concerned with multitasking.
(3) Critical sections must be protected using mechanisms like semaphores.
(4) Deadlock must be avoided or at least detected & resolved.
(5) Debugging multitasking software can be challenging the interactions between tasks over time can be difficult to understand and/or recreate.
(6) If priorities are used, it may be challenging to select the best priorities for each task. Some experimentation may be required to select the priorities.

Question #3 (Multiple Nonpreemptive Loops)

(a) When the software architecture of a real-time embedded system uses multiple nonpreemptive loops, it is recommended that the timing of those multiple loops be produced by a single hardware timer and that the periods of those loops be harmonically related. Why, given that a typical modern microcontroller has four or more independently programmable interrupt-producing timers, would it still be preferable to only use one of those timers to control the timing of all of the preemptive loops? Are there an disadvantages to using only one of the hardware timers?

[3 marks] Usually all of the hardware timers will use clock signals that originate from the same crystal-stabilized oscillator, so multiple timers could be considered. But this would be wasteful of timer hardware. Also, it would be difficult to control the phase of each timer. By using one timer, the software can be designed to ensure that the executions of the different loops interleave efficiently without competing for the CPU.

[3 marks] The disadvantage of using only one hardware timer is that the one foreground task will have a rather complex structure that implements all of the harmonically related loops. This task will have rather poor modularity.

(b) A desirable property of an embedded real-time system is that it be designed so that it fails gracefully when subjected to too much work load, when critical real-time constraints are on the verge of not being met. Similarly, in battery-powered systems, it may be desirable to slow down the CPU's clock (but not the hardware timers) to save power when the battery is going low. How could the multiple nonpreemptive loop architecture be modified to offer performance that can be allowed to degrade in a controlled manner so that very important polling operations are still completed on time, with the processing time devoted to less important operations being reduced using software means.

[6 marks] Software flags could be used to identify which polling operations are critical to perform at the original frequency and which polling operations can be reduced in frequency with acceptable performance degradation. Then when an emergency situation is detected, a global flag could be set that would cause all of the less critical operations to be scanned only 1/X of the time, where X is an acceptable factor for each polling operation.

Question #4 (MicroC/OS)

(a) MicroC/OS has proven to be a popular real-time kernel for embedded systems for a number of reasons. A major reason for its success has been its portability, which is enhanced by the fact that it was implemented in well-structured and well-documented code written in ANSI C. What aspects of the software architecture of MicroC/OS, apart from the choice to use ANSI C, make MicroC/OS a portable kernel? What software changes would be required in order to provide an implementation of MicroC/OS on a new microcomputer?

[3 marks] Aspects that lead to greater portability:
- Most of the core software is written in standard ANSI C and is independent of the computer hardware.
- The "port" layer contains all of the CPU-dependent software. It is written in C and/or in assembly language.
- The MicroC/OS configuration, which is written in C, has software switches that can be used to retain or delete features of MicroC/OS.

[3 marks] Required software changes:
- A new context switch routine is required, which saves the CPU registers into the Task Control Block (TCB) for the swapped out task, and that loads the CPU registers with the saved contents for the newly running task.

(b) When a semaphore is created and initialized in a MicroC/OS system, an OS_SEM object is instantiated before all tasks, and then initialized either before all tasks or (preferably) by the UserMain task. Why is it necessary for OS_SEM to be instantiated outside all tasks? Why is it preferable for semaphores and message queues to be initialized inside UserMain?

[3 marks] The OS_SEM object must be outside the scope of all tasks, and so it must be instantiated outside the scope of all tasks.

[3 marks] Initialization for semaphores and message queues is best done within a task, and UserMain is the designated task for doing initialization. Its high priority ensures that it will finish its initialization activities before any other created tasks start to run.

Question #5 (Serial Communications)

(a) Briefly explain how the use of noise margins and differential signaling are two simple ways of enhancing a receiver's ability to receive a correct digital signal despite the addition of random noise during the communication of the signal from the transmitter.

[4 marks] Noise margins are the difference between the transmitted signal levels (e.g. lowest valid high and highest valid low) and the weakest acceptable received signals (e.g. lowest valid high and highest valid low). The noise margins allow a small amount of noise to be added to the signals during transmission without affecting their digital value.

[3 marks] In differential signaling, a digital signal and its complement are transmitted together on two conductors that pass through the same electro-magnetic environment. The receiver recovers the original digital value by subtracting the two received signals. The subtraction operation causes the common noise signal component to be subtracted away. This method assumes that (1) the noise that is added to the two transmitted signals is roughly the same, and (2) the two signals are attenuated during transmission by roughly the same amount.

(b) Briefly explain the purpose of the CRC-32 cyclic redundancy calculation in the Ethernet standard. How is this calculation used to improve the quality of the communication provided to the applications that are using an Ethernet connection?

[3 marks] The CRC-32 calculation is used in Ethernet to detect the presence of errors in received data frames. This error-detection capability gives the receiver the option of not acknowledging the data frame, which would force retransmission of the frame if a time-out mechanism (like that provided in TCP) is present in higher level protocol.

[3 marks] The CRC-32 error detection mechanism, plus retransmission to replace corrupted data frames (either from time-out or from a retransmission request from the receiver), effectively increases the reliability of the communications channel.

Question #6 (Stepper Motors)

(a) When a microcontroller is used to control a stepper motor with an open loop control architecture, why is it important to produce step signals that start out at a slow step rate and that speed up only gradually through a series of faster and faster step rates until the fastest step rate (the slew rate) is reached? What would likely go wrong if there were too few different step rates?

[3 marks] Step rates must only increase gradually to avoid slippage between the intended rotor position and its actual position. If slippage occurs and the control system is open loop, then the controller will have inaccurate control over rotor position and system performance will be degraded (the rotor position will become inaccurate). System failure could result.

[3 marks] If there were too few step rates, then slippage would possibly occur in the transition between the step rates. The acceleration of the rotor's rotational speed would not be smooth, possibly causing excessive wear-and-tear on the equipment or (in the worst case) failure.

(b) Briefly explain the difference between open loop and closed loop control for stepper motors. What are the major advantages of closed loop control over open loop control?

[4 marks] In an open loop controller, the controller sends step commands to the motor, but the controller has no way of measuring the rotor position and then correcting it if the actual position is different from the intended position.
In a closed loop controller, there is the means to measure the actual position of the rotor. This information can be included in the calculations of the algorithm that produces the step signals to ensure that the rotor reaches its intended final position.

[3 marks] Closed loop control permits more accurate control over rotor position. The initial position measurement and compensation for slippage during rotation can be performed automatically without the intervention of a human operator or another system.

Question #7 (Microcomputer Busses & DMA)

(a) Multiplexed address/data busses are fairly common. For example, they are used in the Freescale FlexBus and the PCI expansion bus. Briefly describe the major advantages and disadvantages of such multiplexed busses. What are they considered reasonably choices for busses of moderate performance that might transfer bulk data in longer blocks.

[3 marks] The main advantage of a multiplexed bus is the reduced number of wires. System cost increases when the number of wires is increased. The disadvantage of a multiplexed bus is that more clock cycles are required to complete bus transactions.

[3 marks] Multiplexed busses are suitable for low-cost and (to some extent) size-constrained systems. Burst transfers are efficient over a multiplexed bus since a burst just needs one address cycle followed by possibly a large number of data cycles.

(b) Briefly explain the difference between single-address and dual-address direct memory access (DMA). Why are data transfers in single-address mode usually driven by DMA request signals originating in a peripheral, whereas in dual-address DMA the transfer is usually initiated when the CPU writes to a DMA control register.

[3 marks] Single-address DMA involves transfers to/from a memory from/to a peripheral. The single address refers to the address that selects locations in a memory buffer. Dual-address DMA (also called two-address DMA) involves data transfer from one memory (using one address) to a second memory (using a second address).

[4 marks] Dual-address DMA cannot be started by either memory buffer since a memory system is a passive system (a slave or target). The CPU must start such a data transfer. When a peripheral is involved, the peripheral can take the initiative and start each data transfer. For example, an transmitting communication port could initiate data transfers as data is sent and its transmit data buffer empties out.

Question #8 (TCP/IP Networking)

(a) TCP uses a window-based flow control method for ensuring that TCP datagrams are eventually received correctly at the far end of a TCP connection. What is the "window" in this method? What factors should be considered when deciding how big to make the window size?

[3 marks] The TCP window is the number of bytes that can be transmitted without being acknowledged from the other end of the TCP connection. The TCP window allows more data to be transmitted into the connection as earlier data is still moving over the connection.

[3 marks] Factors determining window size include:
(1) The quality of the connection. If the error rate is low, a larger window size is more efficient at overcoming the delays that can be expected in transit (e.g., propagation delays, time spent waiting in buffers at routers, detours in the routed path, etc.)
(2) The memory capacity of the receiver. If the capacity of the receiver data buffer is small, then the window size should also be small so that the receiver is not suddenly swamped by the arrival of a large amount of sent data.

(b) Whenever the TCP entity in a networked node attempts to transmit a TCP segment, it keeps a copy of the segment and then associates with it a timer with a specified time-out. What is the purpose of this timer and how might one go about choosing a good time-out value? What should happen to the copy of the segment and the timer object when all of the bytes in the segment have been acknowledged by the receiver?

[4 marks] The time-out of the timer is used to detect when it has taken too long for sent bytes to be acknowledged by the receiver. A typical time-out value is twice the expected round-trip delay.

[3 marks] When all of the bytes in a withheld segment copy have been acknowledged, then the segment buffer and its timer (probably just a timer data structure) must be recycled.