ECE/CS 5780/6780: Embedded System Design

Chris J. Myers

Review 2

Chapter 5 Topics

- Multithreaded preemptive schedulers
- Semaphores and their applications
- Fixed scheduling

Chapter 6 Topics

- Input capture
- Output compare
- Frequency measurement
- Pulse accumulator
- Pulse-width modulation

Chapter 7 Topics

- Serial communication basics
- Serial communication interface (SCI)
- Serial peripheral interface (SPI)

Chapter 8 Topics

- Relays, solenoids, and DC motors
- Stepper motors
- Input switches and keyboards
- Output LEDs
- Liquid crystal displays

Question 1(a)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Assuming that a thread is unable to obtain the semaphore, what should it do? Briefly, how could you modify the code to do this?

Question 1(a)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Assuming that a thread is unable to obtain the semaphore, what should it do? Briefly, how could you modify the code to do this?
- ANSWER: It should give up the rest of its time slice. This can be done with an `swi`. Either point the vector to the scheduler, or have the `swi` handler jump to the scheduler.

Question 1(b)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Explain in words how you could modify the thread scheduler to avoid rescheduling this thread until the semaphore is available.

Question 1(b)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Explain in words how you could modify the thread scheduler to avoid rescheduling this thread until the semaphore is available.
- ANSWER: Keep a blocked list in addition to a runnable list. If a thread wants a locked semaphore, put it on the blocked list. When the semaphore is released, move a blocked thread to the runnable list.

Question 1(c)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Assume that you have modified your scheduler such that threads are scheduled based upon a priority. Explain how priority coupled with spin-lock semaphores can cause a system to *deadlock* (i.e., stop making useful progress).

Question 1(c)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Assume that you have modified your scheduler such that threads are scheduled based upon a priority. Explain how priority coupled with spin-lock semaphores can cause a system to *deadlock* (i.e., stop making useful progress).
- ANSWER: (1) Low priority thread has the semaphore. (2) High priority thread gets scheduled, and starts to wait on semaphore. (3) Low priority thread never gets scheduled, and high priority thread cannot move on.

Question 1(d)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Adding priority can cause *starvation*. What is starvation and how does adding priority cause it?

Question 1(d)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- Adding priority can cause *starvation*. What is starvation and how does adding priority cause it?
- ANSWER: Starvation is when a thread never gets to make progress. Low priority threads can starve if there is always a high priority thread available.

Question 1(e)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- How can you modify a priority scheduler to avoid starvation?

Question 1(e)

In lab 6, you implemented a thread scheduler in which each thread would periodically attempt to obtain a shared resource which is protected using a semaphore.

- How can you modify a priority scheduler to avoid starvation?
- ANSWER: Give low priority threads a bump every time you skip them to get an “effective priority.” After a thread runs, set its effective priority to its actual priority.

Question 2(a)

In lab 7, you implemented a device to measure the frequency of a waveform. In this problem, you are to implement a frequency generator. Assume that the desired frequency is stored in a variable named `freq`.

- How could the code for generating a square wave shown in Section 6.2.3 be modified to generate a desired frequency?

Question 2(a)

In lab 7, you implemented a device to measure the frequency of a waveform. In this problem, you are to implement a frequency generator. Assume that the desired frequency is stored in a variable named `freq`.

- How could the code for generating a square wave shown in Section 6.2.3 be modified to generate a desired frequency?
- ANSWER: In this code, `Period`$= 1/2T$. Since $T = 1/f$, then if your desired frequency is f, then you can set `Period` to $1/2f$ in your main code.

Question 2(b)

In lab 7, you implemented a device to measure the frequency of a waveform. In this problem, you are to implement a frequency generator. Assume that the desired frequency is stored in a variable named `freq`.

- Using this code, what range of frequencies could be generated?

Question 2(b)

In lab 7, you implemented a device to measure the frequency of a waveform. In this problem, you are to implement a frequency generator. Assume that the desired frequency is stored in a variable named `freq`.

- Using this code, what range of frequencies could be generated?
- ANSWER: Assuming a 500ns TCNT clock and not taking into account the time to interrupt and time in the handler, then range is 15 Hz to 1 MHz (i.e., 1/(65535*2*500ns) to 1(2*500ns)).

Question 2(c)

In lab 7, you implemented a device to measure the frequency of a waveform. In this problem, you are to implement a frequency generator. Assume that the desired frequency is stored in a variable named `freq`.

- Explain in words how you could design a program to generate a wider range of frequencies.

Question 2(c)

In lab 7, you implemented a device to measure the frequency of a waveform. In this problem, you are to implement a frequency generator. Assume that the desired frequency is stored in a variable named `freq`.

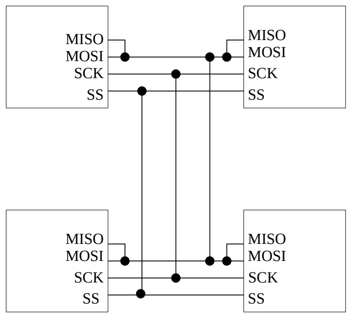
- Explain in words how you could design a program to generate a wider range of frequencies.
- ANSWER: Upper range is limited by the internal clock. Optimizing the handler would increase the upper range. The lower range can be increased either using multiple output compares or timer overflow.

Question 3

In lab 8, you implemented a simple network using SCI. Assume that you want to now implement a network using SPI instead.

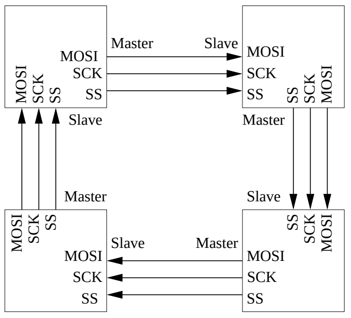
- Show a diagram of how you could connect four microcontrollers using SPI to form a network.
- Describe in words the procedure to get a packet from one microcontroller to another.

Question 3: Solution 1



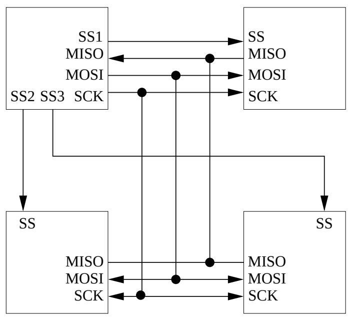
- ANSWER: Whan a microcontroller wishes to send a packet, it attempts to become master and set $\overline{SS}$. It then sends its packet on MOSI and checks the echo on MISO to watch for collisions.

Question 3: Solution 2



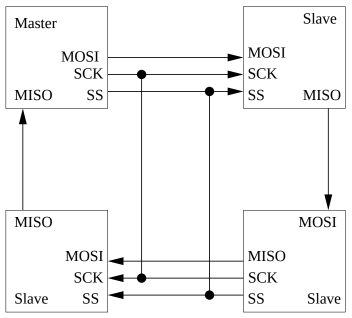
- ANSWER: Each microcontroller sends packets using SPI port on which it is master to neighbor who is slave on that port. Upon receiving packet on slave port, checks address. If not intended for this microcontroller, forwards packet on port in which it is master.

Question 3: Solution 3



- ANSWER: Master communicates with each slave individually in a round-robin fashion (i.e., SS1, SS2, and SS3). If slave wants to send a packet to another slave it must do so by sending the packet to the master who forwards it to the slave. No collisions problems.

Question 3: Solution 4



- ANSWER: Master continuously keeps packets moving around. If master wants to communicate, stores valid packet to SPDR, else “empty” packet. Same for slaves. No collision problems.

Question 4(a)

In lab 9, you designed a stepper motor interface in which you specified a number of steps and a period between steps. In this problem, we will consider using a DC motor instead.

- Why is it more difficult to use a DC motor?

Question 4(a)

In lab 9, you designed a stepper motor interface in which you specified a number of steps and a period between steps. In this problem, we will consider using a DC motor instead.

- Why is it more difficult to use a DC motor?
- ANSWER: It is difficult to determine the position and speed of the motor in an open loop fashion.

Question 4(b)

In lab 9, you designed a stepper motor interface in which you specified a number of steps and a period between steps. In this problem, we will consider using a DC motor instead.

- What do number of steps and period between steps correspond to for a DC motor?

Question 4(b)

In lab 9, you designed a stepper motor interface in which you specified a number of steps and a period between steps. In this problem, we will consider using a DC motor instead.

- What do number of steps and period between steps correspond to for a DC motor?
- ANSWER: Steps equal angle of rotation or number of cycles in PWM signal. Period equals speed of motor or duty cycle of PWM signal.

Question 4(c)

In lab 9, you designed a stepper motor interface in which you specified a number of steps and a period between steps. In this problem, we will consider using a DC motor instead.

- What would you need to add to your design in order to move a particular “number of steps” and to have a particular “period between steps”?

Question 4(c)

In lab 9, you designed a stepper motor interface in which you specified a number of steps and a period between steps. In this problem, we will consider using a DC motor instead.

- What would you need to add to your design in order to move a particular “number of steps” and to have a particular “period between steps”?
- ANSWER: Sensors for position and/or speed which are used as feedback in a closed loop solution.