EECS 373
Exam 2
Winter 2005
March 24, 2005
(SOLUTIONS)

Name:

University of Michigan uniqname:

Open book, open notes. Calculators are permitted, but no laptops, PDAs, cell phones, etc. This exam has seven problems on ten pages. Problems vary in difficulty; it is strongly recommended that you do not spend too much time on any one problem.

The rules of the Honor Code of the University of Michigan College of Engineering apply for this exam.

Honor code pledge: I have neither given nor received aid on this examination, nor have I concealed any violations of the Honor Code.

Signature:
(examinations without a signed honor pledge will not be graded)

1. Interrupts *(9 points)*:

When programming an interrupt service routine, one thing your code must do is to save the MSR into SRR1.

TRUE ***FALSE***

When programming an interrupt service routine, one thing your code must do is to disable external interrupts at the beginning of the ISR to prevent nested interrupts.

TRUE ***FALSE***

Even a very simple interrupt handler that does not support nested interrupts must always re-enable interrupts before returning with an `rfi` instruction.

TRUE ***FALSE***

2. Control system behavior: *(14 points)*:

As you vary the gain of a proportional controller from zero up to a little over one, you observe all of the following behaviors at different gain values:

(a). output oscillates with increasing amplitude

(b). output oscillates forever with constant amplitude

(c). output does absolutely nothing (stays constant forever)

(d). output settles to correct value quickly (with no overshoot)

(e). output overshoots the correct value then settles to it fairly quickly

(f). output very slowly heads toward correct value and eventually settles there

(g). output overshoots the correct value and oscillates for a long time (and gradually settles down to the correct value)

Complete the table below to match each behavior (a) through (g) with a gain value. This is a one to one mapping; each behavior should be associated with exactly one gain value. Hint: think about the order in which these behaviors will occur as you vary the gain.

Gain	Behavior
0.0	(c)
0.2	(f)
0.4	(d)
0.6	(e)
0.8	(g)
1.0	(b)
1.2	(a)

3. Control system debugging: *(8 points)*:

For your EECS 373 final project, you decide to build a custom cooling system for a heavily overclocked gaming PC. You don't like a lot of noise, so you do not want to run the cooling fan at full speed. You use a temperature sensor on the CPU and a simple proportional feedback control system to control the fan speed. The user can pick a specific CPU temperature, and your control system will adjust the fan speed to maintain that temperature. When you initially test your system, you find that it basically works, but whenever you change the desired CPU temperature, the actual CPU temperature oscillates above and below this temperature and it takes several minutes for it to settle down.

What should you try first to improve the controller?

(a). add an integral term into your control function

(b). add a derivative term into your control function

(c). introduce some time delay into your control loop

(d). increase the proportional gain

(e). *** decrease the proportional gain ***

You get the above problem solved, but you observe that there are steady state errors in the temperature. When the user requests a temperature change, the system goes smoothly to a new temperature near the desired one. However, it never quite reaches the desired temperature even after waiting for a very long time.

Now what should you try to improve the controller?

(a). *** add an integral term into your control function ***

(b). add a derivative term into your control function

(c). introduce some time delay into your control loop

(d). increase the proportional gain

(e). decrease the proportional gain

4. PPC programming and ABI *(8 points)*:

Write an ABI compliant function named **whereami** that returns the program counter value for the next instruction AFTER the call to **whereami**. The function takes no arguments. Example usage:

```
main:    some_code
         bl whereami
         next_line_of_code
```

The return value of **whereami** should be the address of next_line_of_code. Your job is to write the code for the **whereami** function in the space below. Hint: keep it simple!

```
whereami:    mfspr r3, LR
             blr
```

5. Error correcting codes *(13 points)*:

Global Warming Motors, Inc. has a problem with their extra large sport utility vehicles. The transmissions tend to inadvertently shift gears, such as from "Park" into "Reverse", often crushing large tracts of old growth forest as a result. The source of problem is tracked down to bit errors in the electronic communication from the shift lever to the transmission. Their engineers design the following error correcting code:

Codeword	Gear Selection
0000000	Park
0001111	Reverse
0110011	Neutral
1010101	Drive
1011010	Low
1100110	Bulldoze

The next eight questions apply to the particular code above.

(a) What is the Hamming distance for this code? **4**

(b) This code can be used to detect single bit errors. ***TRUE*** FALSE

(c) This code can be used to detect double bit errors. ***TRUE*** FALSE

(d) This code can be used to detect triple bit errors. ***TRUE*** FALSE

(e) This code can be used to correct single bit errors. ***TRUE*** FALSE

(f) This code can be used to correct double bit errors. TRUE ***FALSE***

(g) This code can be used to correct triple bit errors. TRUE ***FALSE***

(h) This code can be used to correct single bit errors and detect double bit errors at the same time. ***TRUE*** FALSE

The next two questions are about Hamming codes in general, not the particular code above.

(i) What is the minimum Hamming distance for a code to detect four errors? **5**

(j) What is the minimum Hamming distance for a code to correct four errors? **9**

7. Timers *(24 points)*:

You are to implement a timer with the timer 1 channel of the PPC CPM module that has a source clock of 2 MHz. The timer must time intervals as long as 2 minutes. The timer must provide the best possible resolution that is a whole number of milliseconds. **Be sure to show your work for all parts of this question!** For these requirements:

(a). What is the resolution of the timer in milliseconds?

2 minutes = 120 seconds. Timer allows a maximum of 2^{16} counts.

$\frac{120 \text{ seconds}}{2^{16}} = 0.001831$ seconds $= 1.831$ milliseconds

This is the minimum timer resolution. Since a whole number of milliseconds were required, round up to 2 milliseconds.

Check: $\frac{120 \text{ seconds}}{2 \text{ milliseconds}} = \frac{120{,}000 \text{milliseconds}}{2 \text{ milliseconds}} = 60,000$

60,000 is less than 2^{16}, so this works.

(b). Given what you found in part (a), what is the longest possible time interval for the timer to the nearest second?

2 milliseconds $* 2^{16} = 131,072$ milliseconds $= 131.072$ seconds

Rounding to the nearest second, 131 seconds

(c). Based on what you found in part (a), what are the settings of the ICLK field and PS field of the TMR1 register? Express the ICLK field in **binary** and the PS field in **decimal**.

ICLK = 10 divide by 16

PS = 249

(d). Based on what you found in part (a), what is the setting of the TRR1 register for a timer interval of 2.1 seconds in **decimal**?

2.1 seconds = 2100 milliseconds

$\frac{2100 \text{ milliseconds}}{2 \text{ milliseconds}} = 1050$

(e). What fields must be set in the TMR1 register to provide a periodic interrupt for the time specified in the previous question and what are the values?

ORI = 1

FRR = 1

(f). Your sales people find out later that the customer needs a time resolution of 1 millisecond. As the field application engineer, what compromise must you make to the timer specification and what is the compromised value?

If the resolution must be changed from 2 milliseconds to 1 millisecond, the range will be limited.

New range: 1 millisecond * 2^{16} = 65,536 milliseconds = 65.536 seconds

(g). What other timer resources would you need to meet the customer's needs without compromise? You can answer in general terms, but specify the register and bit fields to enable this.

You will need to cascade with one of the other counters to obtain the resolution and range requirement.

TMR1, ICLK = 00

(h). For the new timer, what is the smallest possible time setting and the largest possible time setting?

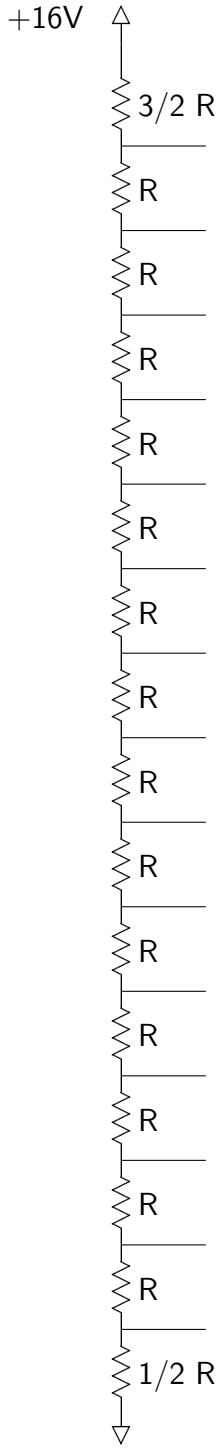
Smallest is 1 millisecond

Largest is 1 millisecond $* 2^{16} * 2^{16} = 2^{32}$ milliseconds $= 4,294,967,296$ milliseconds $= 4,294,967.296$seconds

$\frac{4{,}294{,}967.296 \text{ seconds}}{(60 \text{ seconds/minute})(60 \text{ minutes/hour})(24 \text{ hours/day})} = 49.7$ days

7. Analog to Digital Conversion *(24 points)*:

You are given an A to D converter similar to the ADC0808 except that it is a 4-bit device. The non-linearity error is +/- 1/2 LSB. It has a resistor ladder as follows with the reference voltage shown.



(a). What is the quantization error in volts?

Quantization error $= \pm\frac{1}{2}$LSB $= \pm\frac{1}{2}\frac{16 \text{ volts}}{2^4} = \pm$0.5 volts

(b). What are the transition voltages for the value 3 assuming the effects of non-linearity are **insignificant**?

Transition voltages = n $*$ LSB $\pm \frac{1}{2}$LSB

In this case, n = 3, LSB = 1 volt

Thus, transition voltages $= 3 * 1$ volt $\pm \frac{1}{2} * 1$ volt $= 3 \pm 0.5$ volts $=$ 2.5 volts and 3.5 volts

(c). What are the transition voltages for the value 3 assuming the effects of non-linearity are **significant**?

Transition voltages = quantization transition voltages $\pm\frac{1}{2}$ LSB
$=$ 2.5 $\pm$ 0.5 volts and 3.5 $\pm$ 0.5 volts
$=$ 2.0 to 3.0 volts and 3.0 to 4.0 volts

(d). In the lab you measure the transition voltage for count value 4 and 5. How large would you expect the difference between these two values to be before you suspected a problem with your measurement? Express your answer in volts.

The differential non-linearity = 2V
However, the maximum absolute difference between count 4:
3.5 ± 0.5 to 4.5 ± 0.5
and count 5:
4.5 ± 0.5 to 5.5 ± 0.5
$= |3.0 - 6.0| = 3$ volts
So, anything greater then 3 volts would be suspect.