Homework 7: Answers

Profs. John McLeod & Arash Reyhani

ECE3375, Winter 2022

Please note that the particular homework assignment that you answered will have the same options as listed here, but the order of these options may have been randomized for each quiz. So your option (b) may instead be written here as option (c), etc. Please read the questions and options in full.
The answers to these questions are all found in the course notes, textbook, and video lessons.

1. Problem: You have a **16-bit,** $5\,\mathrm{MHz}$ **count-up timer**. You want it to count for a $3\,\mathrm{ms}$ interval. What value should be written to the timer's data register? (Please provide the answer as a **decimal** number.)

Answer:

$$(3\,\mathrm{ms})\,(5\,\mathrm{MHz}) = \left(3\times10^{-3}\,\mathrm{s}\right)\left(5\times10^{6}\,\mathrm{Hz}\right) = 15\times10^{3}.$$

Since this is a count-up timer, the timer stops when it overflows. Therefore we need to start the timer at $2^{16} - 15\,000 = 50\,536$.

2. Problem: You have a **16-bit,** $10\,\text{MHz}$ **count-down timer**. If the timer data register was previously set to `0x7A2B`, what is the timer interval?

Answer: This is a count-down timer, so the data register value is the same as the timer interval.

$$\frac{7 \times 16^3 + 10 \times 16^2 + 2 \times 16 + 11}{10\,\text{MHz}} = \frac{31275}{1 \times 10^7\,\text{Hz}} = 3.1275\,\text{ms}.$$

3. Problem: The interval timer on the DE1-SoC is set to **count-down and repeat**. How should you periodically check to see when each count-down is complete (i.e., a **timeout** occurred)?

(a). Read the status register. If the result is 1, a count-down was completed. Before proceeding, write an arbitrary number to the status register to clear the timeout flag.

(b). Read the status register. If the result is 3, a count-down was completed. Before proceeding, write an arbitrary number to the status register to clear the timeout flag.

(c). Read the status register. If the result is 2, a count-down is still in progress. Before proceeding, write an arbitrary number to the status register to clear the timeout flag.

(d). Read the control register. If the result is 8, the count-down has stopped. Before proceeding, write an arbitrary number to the control register to clear the timeout flag.

Answer: Option (b). The result should be 3 because the timer is still running and has timed-out (`0b0010+0b0001`).

- Option (a) will never happen while the timer is on "count-down and repeat" mode, as the timer will never stop running.
- There is nothing wrong with Option (c), except it doesn't tell you when a timeout has occurred — rather it tells you when a timeout has not occurred.
- Option (d) won't work, because the control register does not inform you about the current status of the timer. *Writing* 8 to the control register will stop the timer, but you can't know whether a timeout occurred without checking the status register.

4. Problem: What is the **longest** time interval you can count using only the ARM interval timer?

Answer: Using *only* the ARM interval timer means you can't store intermediate values in memory, so the maximum time is:

$$\frac{2^{32}}{100\,\mathrm{MHz}} = 42\,929.76\,\mathrm{ms}$$

5. Problem: You have constructed a coffee-maker peripheral for an ARM microcontroller. The coffee-maker takes 5 minutes to finish brewing. This is longer than the longest possible interval on the ARM interval timer.

Which of the options below is the **best** way to time a 5 minute interval?

(a). Set the timer interval for some integer subdivision of 5 minutes. Start the timer for count-down and repeat. Periodically add the timeout flag to some accumulating variable, then immediately clear the timeout flag. When the accumulating variable reaches the appropriate number of subdivisions, the coffee is done.

(b). Set the timer interval for some integer subdivision of 5 minutes. Start the timer for count-down and repeat. Periodically check the low period register. When that register reaches zero, increment a counter. When the counter reaches the appropriate number of subdivisions, the coffee is done.

(c). Define the full 5 minute count interval as a 64-bit number (i.e. spread across two registers in assembly). Set the timer interval to the maximum value, and subtract this value from the full count interval stored in memory. Start the timer for a single count-down. After the timer has counted down for a while, periodically subtract some value from the full count interval (stored in memory) and add that value to the current timer interval (i.e. "rolling it back"). When the full count interval in memory reaches zero, and the timer stops, the coffee is done.

(d). It is impossible: you need a 64 bit timer or a timer with a slower clock speed.

Answer: Option (a) is the best. From the previous question, we see the ARM timer can be set to $30\,\mathrm{s}$ (for example), so 10 count-down cycles is 5 minutes. Constantly adding the timeout flag to an accumulating variable is one way to accumulate this count of 10 cycles: whenever a timeout occurs, the timeout flag will be 1, incrementing the counter. If a timeout did not occur, the flag will be zero so adding it does nothing. Similarly, we can always clear the timeout flag after each read: either it will correctly clear the previous timeout, or it will do nothing (because the flag was already zero).

- Option (b) sort of works, but is a bad choice, because the low-period register will only be zero at the instant the timeout occurs. Immediately after that it will roll back to the full count interval. Therefore it is reasonably likely that this method will miss one or more timer intervals, resulting in a measuring longer than 5 minutes.

- Option (c) is a working algorithm, but is a poor choice for an ARM timer because it interferes with the period register while the timer is operating, leading to counting errors. (This is why you are supposed to check the counter register instead of the period register while the timer is operating.)

- Option (d) is not true.