Measuring software execution time

1 Introduction

It's often important in the development of embedded systems to be able predict the execution time of (parts of) the code. There is a variety of possible approaches to predicting software execution time. One approach that is not 100% reliable, but that can be readily applied in practice, is to develop the code and measure its execution time over a number of runs. When adopting this approach, care needs to be taken to ensure that the measurements are as accurate as possible, and to identify the likely size of any remaining inaccuracy.

2 In the lab

1. Clone the repository `cm0605_lab05.git` and checkout a solution to the week 5 lab.

   ```
   $ git clone https://github.com/DavidKendall/cm0605_lab05.git
   $ cd cm0605_lab05
   $ git checkout P09
   ```

2. Start up EWARM and load the workspace `cm0605_lab05/workspace.eww`.

3. Connect a `LPC-2378-STK` board to a USB port on your computer.

4. Download and debug the project. Run it and observe its behaviour. This is a solution to the lab exercise given in week 5, requiring the use of semaphores to protect access to a shared resource (LCD). Make sure that you understand the code in `main.c`, in particular how the semaphore is declared, created and used.

5. Comment out the lines that use `OSSemPend` and `OSSemPost`. Download and debug the program again. Observe its behaviour now when you run it. Start the lights flashing and decrease the flashing delay using the joystick. Why does this behaviour occur? Why does it not occur when the `OSSemPend` and `OSSemPost` lines are not commented out?

6. Now checkout `P10`. It improves on `P09` in a number of ways. What improvements do you notice?

7. Now clone `cm0605_lab07`. Open the workspace. Download and debug the project. This program is intended to investigate the execution time of code that uses the LCD. The software under test is the block of code between the comments `Software under test`. Run the program and observe its behaviour. You can re-run the experiment by pressing `BUT_1`. Make sure that you understand the code and how the experiment is constructed.

8. The peripheral clock for the timer is running at 15 MHz. The numbers shown in the experiment represent the number of ticks of this clock.

   (a) What is the resolution of this clock?

   (b) The timer counter is a 32 bit register. What is the longest period of time that can be measured at this resolution before the timer counter overflows?

   (c) Express in milliseconds the minimum, mean and maximum values that you observe for writing a formatted number to the LCD.

9. Modify this experiment in the following ways:

   (a) Uncomment the line that writes `Hello World` to the LCD and comment out the line that writes the formatted number string. Re-run the experiment. What can you conclude from the results about the length of time that a single number formatting operation takes?

   (b) Reduce the length of the original string by a single character. Re-run the experiment and note the results. Now increase the length of the original string by a single character. Re-run the experiment and note the results. What can you say about the length of time that it takes to write a single character to the LCD?

   (c) Turn off all optimisations, see
   `Project -> Options -> C/C++ Compiler -> Optimizations`
   Re-run the original experiment. What is the effect of the optimiser on the length of time that it takes to write a single formatted number string?

10. Now design some experiments of your own to investigate, e.g.

    (a) the accuracy and precision of the `dly100us` function;

    (b) the effect of using `i++`, `i += 1` and `i = i + 1` as the increment mechanism in the control parameters of a `for` loop;

    (c) the effect of the number of parameters in calling a function;

    (d) the effect of storing data in different types of memory, e.g. sum an array of ints stored in RAM (automatic); sum the same array stored in flash (static const);

(e) Use loop to obtain results for measurement less than clock resolution – remember to account for time taken by loop.

(f) the time taken by uC/OS-II to transfer control from the currently-running task to a different task. Implement the experiment and record the results. Identify the strengths and weaknesses of your solution.

Remember if there's anything you don't understand in the labs, please ask your tutor for help.