Memo to: Randy Larimer
From: Erik Andersen
Date: February 3, 2015
Regarding: EELE 465, Lab0 Project – Heartbeat LED

Summary:

The purpose of this lab was to use the MC9S08QG8 microprocessor to flash an LED at 0.5 Hz (1 second on, 1 second off). This assignment was completed three different ways: first without using device initialization with a loop counter, second with device initialization and modifying the pre-scalar in order to get the same flash rate as before, third with device initialization and setting the pre-scalar back to its defaults by using the timer interrupt instead to get the required flash rate. The completed program could flash an external LED on the MC9S08QG8 microprocessor.

Preliminary Solutions:

The flowcharts given in Appendix A, were useful in giving us a place to start from. In all three solutions the code needed for turning the LED on and off needed to be determined and the LED pin needed to be initialized as an output. The first solution according to the flowchart shown in figure A.1 involved turning on the LED and creating a counter that determines when one second has gone by and then reusing that block of code before the LED is turned on again.

The second solution included using code warrior version 10.2 instead of 10.4 to generate code in order to set up the timing interrupt. After the previously mentioned initialization steps, the timer will trigger every one second, which will turn the LED on or off.

The third solution involved no modification of the pre-scalar to set the timing interrupt to one second so once the period of the interrupt was determined a counter can be used to see when one second has gone by.

Setup:

The setup simply involved interfacing the DEMO9S08QG8 board with the desktop computer by USB cable to be used for downloading the firmware to the processor for debugging and wiring the LED1 on the SCH to the PTB6 pin. Next we set the Data Direction register on Port B to a one allowing the use of PTB6 as an output. Writing a zero to PTB6 would turn on the LED. To toggle the LED the state of Port B was loaded into the accumulator; then one's complemented, which only toggle's bit 6 in this case; and the result was stored back into Port B.

First Solution – Counter Loops:

In order to use counters one had to determine how long it took for one second to pass, we simply guessed until we got it right. By determining the frequency of the bus clock to be 4MHz the number of counter loops was determined. The final design resulted in three nested loops using the 8-bit accumulator to count down in our case from 65 (decimal) to zero in order to achieve a one second delay. It is also worth noting that we "fed" the COP watch dog at the beginning of each of the three nested loops so that the processor would not reset. The predefined macro in the derivatives.inc file was used to accomplish this.

Second Solution – Device Initialization with Timer Interrupts:

For the second solution, device initialization was used to generate code using code warrior version 10.2 to set up the timer interrupt from the timer pulse width module (TPM). With the pre-scalar set to 64 and the overflow interrupt enabled, the timer would trigger an interrupt every 1.046 ms. The specified frequency was 0.5 Hz and the timing achieved was slightly greater than a 0.5 Hz flash from the LED.

Third Solution – Combination of Solutions 1 & 2:

For the third solution the pre-scalar had to be set back to one according to the assignment. This required that a counter be added to take the place of the pre-scalar function. The LED flashed every 16.046 ms, so a counter looping 100 times we found resulted in just the right delay achieved in solution two. The counter was created as a 3-byte variable in RAM, was initialized to 100, and was decremented by the timer. When the counter became zero, the LED state would be switched and the counter would be reset to 100.

Final Comments:

While performing this lab, many problems and errors in the code were encountered. Some took multiple hours to troubleshoot and debug. The most bothersome and frustrating was figuring out and understanding how in solution one too design the counter. For example, how many nested loops were required two or three? How does it decrement down to zero?

Appendix A – Figures

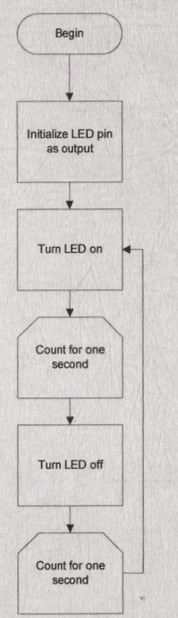

Figure A.1) Flowchart for the first solution, counter-based delay.

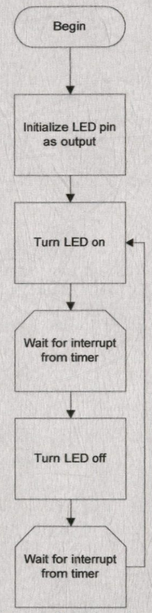

Figure A.2) Flowchart for the second solution, timer interrupt-based delay.

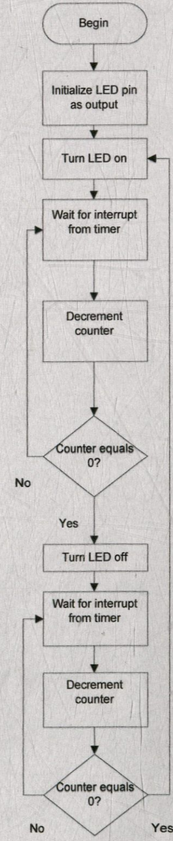

Figure A.3) Flowchart fo third solution, a mix of the two solutions.

"From the EELE465 website, used with permission"