Experiment Results

	Experiment	**Period (ms)**			**Toggle Count after 1 Minute**		
	Description	Red	Yellow	Green	Red	Yellow	Green
1	WCET Red LED Toggle	1000	1000	1000	59	61	61
2	Timer RED LED	1000	1000	1000	60	59	59
3	With 90ms busy-wait loop in Green ISR	500	500	500	99	120	120
4	With 90ms busy-wait loop in Yellow ISR	500	500	500	13	119	119
5	With 110ms busy-wait loop in Green ISR	500	500	500	94	121	121
6	With 110ms busy-wait loop in Yellow ISR	500	500	500	NA	NA	NA
7	With 510ms busy-wait loop in Green ISR	500	500	500	NA	NA	NA
8	With 510ms busy-wait loop in Yellow ISR	500	500	500	NA	NA	NA
9	With 510ms busy-wait loop in Green ISR and sei call	500	500	500	NA	NA	NA
10	With 510ms busy-wait loop in Yellow ISR and sei call	500	500	500	NA	NA	NA

Questions

1) How good was your WCET analysis of the for loop?
   It was alright, not perfect. There was a lot of error, including the yellow and green timers throwing off the count as they interrupted the loop

2) Why should you not use the menu while running experiment 1?
   Because the serial line is slow, which could have slowed the red timer down even further (output to the serial line is done via a busy wait, so any menu entries would have waited for the output string to transfer over the 9600 baud serial line).

3) (With the 90ms busy-wait loop in one of the ISRs) What did you observe? Did the busy-wait disrupt any of the LEDs?

   The Red LED slowed down a little bit when the wait was in the green ISR, but significantly more when it was in the yellow ISR. The yellow and green toggle counts weren't really affected by the loop.

   The reason the red LED slowed down is because the interrupt handler is taking a significant amount of time to complete. In the case of the green LED, the interrupt handler is only called every 500ms, and it takes just over 90ms to execute, so the red LED has the other 410ms to

perform its toggle at the appropriate time.

When the delay was in the yellow ISR, the red LED slowed down much more significantly due to the higher frequency that the ISR is fired. The yellow ISR is executed every 100ms, and with the loop, it executes for 90ms. That gives the main program just 10ms (actually less, due to red and green ISRs and the time used to ping the serial line).

4) (With the 110ms busy-wait loop in one of the ISRs) What did you observe? Did the busy-wait disrupt any of the LEDs?

   The Red LED slowed down a little bit more when the wait was in the green timer, but both yellow and green LEDs remained synchronized. Things got more interesting when the loop was in the yellow LED ISR. With the loop in the yellow ISR, the menu no longer functioned. In this case, the ISR takes longer than the period of the interrupt, so it's often the case that the ISR is ready to fire as soon as the interrupt completes. This causes the green and yellow LEDs to no longer toggle at the same time. It also all but stopped the red LED from toggling (it did finally turn on close to the 1 minute mark). This is because the processor is spending almost all of its time inside the yellow ISR, and the main loop is barely able to execute a single instruction.

5) (With the 510ms busy-wait loop in one of the ISRs) What did you observe? Did the busy-wait disrupt any of the LEDs?

   Busy-wait in Green ISR
   The red LED stayed off the entire time, and the yellow LED stayed on the entire time. The green LED continued to flash as normal. This time, the busy-wait is longer than the green ISR period. The green LED continued to flash at the correct rate, since it is set to toggle in hardware rather than software. But the busy-wait in green ISR prevented the yellow ISR from executing (Timer 1 must have a higher priority interrupt than timer 3), and prevented the main program from executing to toggle the red LED.

   Busy-wait in Yellow ISR
   Again, the red LED stayed off the entire time and the green LED continued flashing at the correct rate. But this time, the yellow LED toggled about every 2.5 seconds. The yellow ISR routine takes almost 5 times it's execution period. This all but prevents the main program from executing. Since the green timer is toggling the green LED in hardware, it continues to toggle at the approriate interval. But since the busy wait is now in the yellow ISR, it gets to execute (almost) as soon as it completes. This means that the yellow tick count will increase about every 510ms, reaching the toggle threshold after 5 ISR calls (2550ms)

6) (With the 510ms busy-wait loop in one of the ISRs and calling sei at top) What did you observe? Did the busy-wait disrupt any of the LEDs?

   Note: For this step, I placed the busy wait at the end of the ISR. If it was placed at the

beginning, the yellow busy loop could cause the yellow LED from ever toggling, since the yellow interrupt would prevent the busy loop from ever terminating.

Busy-wait in Green ISR
The red LED stayed off the entire time, but the yellow and green LEDs continued to toggle at the correct rate. Since the green ISR's execution time is longer than its period, it continues to block the main program from executing (so the red LED never toggles). However, the sei call at the top re-enables interrupts on the processor. So when the yellow interrupt fires, it takes control of the processor and executes until completion. Presumably, eventually the stack would run out of room to hold the trail of interrupts that were interrupted and the program would crash.

Busy-wait in Yellow ISR
Like when the loop was in the green ISR, in this case, the green and yellow LEDs continued to flash in sync, and the red LED remained off. The sei call allowed the yellow timer to interrupt the yellow ISR while it was still running the busy loop. Whether the green ISR is ever called isn't decipherable, but since it likely has a higher interrupt priority than the yellow, it's likely it was called and executed successfully. Like when the timer was in the green ISR, presumably the stack will run out of room and the program will crash. Since the yellow ISR is executed 5 times as much as the green, it'll happen sooner.