CSE530: EOSI - ASSIGNMENT 3 – PART 2 REPORT

Sushant Trivedi (1213366971) and Ravi Bhushan (1214347783)

PROBLEM STATEMENT

Implement the functionality of Part 1 using bit-banging.

APPROACH TAKEN

Use of software and delay loops to control a gpio output signal to generate any waveform. We investigated two delay mechanisms, ndelay and hrtimer. We also investigated two methods to set bits to gpio gpio_set_value () and iowrite32(). We have used TSC counter and ktime to measure the overheads.

EXPERIMENTS PERFORMED

We performed the following experiments to determine the approach for bit banging:

1. Measure overhead of ndelay using tsc counter
2. Measure overhead of ndelay using ktime
3. Measure overhead of hrtimer using tsc counter
4. Measure overhead of hrtimer using ktime
5. Measure overhead of gpio_set_value () function using tsc counter
6. Measure overhead of gpio_set_value () function using ktime
7. Measure overhead of iowrite32() function using tsc counter
8. Measure overhead of iowrite32() function using ktime
9. Try bit banging, using ndelay and gpio_set_value ()
10. Try bit banging, using hrtimer and gpio_set_value ()
11. Try bit banging, using ndelay and iowrite32()
12. Try bit banging, using ndelay and iowrite32()

For overhead calculations, we have taken 1000 samples and averaged them to get average overhead. For bit banging, we have tried turning on all LEDs by sending all 1s and then turning them off by sending all 0s.

OBSERVATIONS

Please refer to ReadMe for instructions to run our test program (timer_delay_test.c)

The following logs get printed on putty once we run our test file.

root@quark:/home# insmod timer_delay_test.ko
[10166.838416] Configured pin gpio12 as ouput which cannot sleep
[10166.860207] Configured pin gpio28 as ouput which can sleep
[10166.873193] Configured pin gpio45 as ouput which can sleep
[10166.878868] Test Initialized
[10166.881919] Overhead Measurement. Sample Size 1000
*[10166.887638] For **ndelay** of 350ns **rdtsc()** recorded average delay of **611ns** with average overhead of **261ns***
*[10166.898456] For **ndelay** of 350ns **ktime_get()** recorded average delay of **942ns** with average overhead of **592ns***
*[10166.911847] For **Hrtimer** of 350ns **rdtsc()** recorded average delay of **2987ns** with average overhead of **2637ns***

*[10167.912497] For **Hrtimer** of 350ns **ktime_get()** recorded average delay of **1593ns** with average overhead of **1243ns***
[10168.920126] Resetting GPIOs using iowrite32()
*[10171.933638] Average time to set gpio using gpio_set_value() (rdtsc) : **1233ns***
*[10171.945055] Average time to set gpio using gpio_set_value() (ktime) : **1633ns***
[10174.960123] Resetting GPIOs using iowrite32()
[10174.964910] Resetting GPIOs using iowrite32()
*[10175.970516] Average time to set gpio using iowrite32() (rdtsc) : **113ns***
*[10175.978031] Average time to set gpio using iowrite32() (ktime) : **444ns***
[10178.990125] Resetting GPIOs using iowrite32()
[10178.994912] Resetting GPIOs using iowrite32()
[10180.000123] Bit Banging Test
[10180.003050] ndelay gpio_set_value: Trying to turn on all leds
[10182.020126] ndelay gpio_set_value: Trying to turn off all leds
[10185.030135] Resetting GPIOs using iowrite32()
[10186.040125] hrtimer gpio_set_value: Trying to turn on all leds
[10186.048198] hrtimer gpio_set_value: Trying to turn off all leds
[10192.060128] Resetting GPIOs using iowrite32()
[10193.070124] ndelay iowrite32: Trying to turn leds red
[10195.080129] ndelay iowrite32: Trying to turn off all leds
[10200.100138] Resetting GPIOs using iowrite32()
[10201.110124] hrtimer iowrite32: Trying to turn on all leds
[10201.116911] hrtimer iowrite32: Trying to turn off all leds
[10207.130129] Resetting GPIOs using iowrite32()

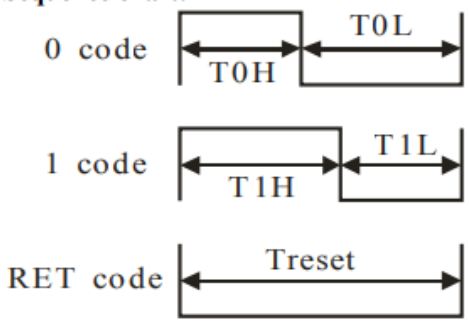
INFERENCES

https://cdn-shop.adafruit.com/datasheets/WS2812.pdf mentions the data transfer time as below:

Data transfer time(TH+TL=1.25μs±600ns)

T0H	0 code ,high voltage time	0.35us	±150ns
T1H	1 code ,high voltage time	0.7us	±150ns
T0L	0 code , low voltage time	0.8us	±150ns
T1L	1 code ,low voltage time	0.6us	±150ns
RES	low voltage time	Above 50μs	

Sequence chart:



***Overhead Measurement Analysis*:**

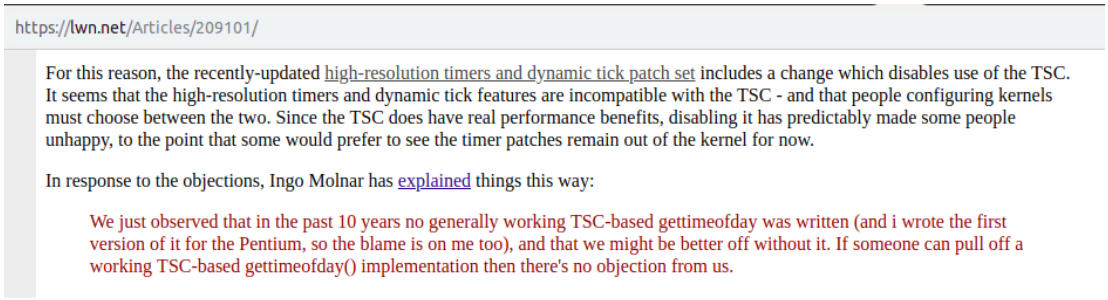
- **ndelay:**
  As we can infer from the logs, ndelay incurs an average overhead of around 250-280ns using rdtsc(), for a delay of 350ns.

  It incurred an average overhead of 550-600ns using ktime_get(). This overhead remains constant even if we decrease the delay. So, ndelay can be used for bit banging.

  We can reduce the T0H,T0L,T1H and T1L time to incorporate ndelay overheads.

- **hrtimer:**
  The logs show that hrtimer incur an average overhead of 2500-3000ns for a delay of 350ns using rdtsc(). It incurred an average delay of 1200-1400ns using ktimer_get(). As per this blog **https://lwn.net/Articles/209101/**, use of TSC has been disabled while using hrtimer. This explains the high value of delay registered by tsc.

  ktime_get() is not reliable as hrtimer internally uses ktime to keep track of time. So, we cannot conclusively determine from the experiment if hrtimer will be able to satisfy the timing constraints of bit banging.



- **gpio_set_value():**
  Average time to set one bit to gpio pin using this function was 1200-1300ns through rdtsc() and 1600-1700ns using ktime_get(). These delays are very high as compared to the timing constraints of WS2812 data sheet.
  Setting zero to LEDs has the strictest requirements of a high pulse in the range of 200-500ns. This requirement cannot be met by this function and the later experiments prove this fact.

- **iowrite32():**
  Average time to set a bit to gpio using this function was 80-120ns using rdtsc() and 400-500ns using ktime_get(). These delays are comparable to timing requirements of WS2812.
  Hence, we use this for our purpose

Bit-Banging Approach Analysis

- **ndelay and gpio_set_value ():**
  We were able to switch on all the LEDs but were unable to switch off the LEDs. We can infer that we are not able to send 0 bit to the LED receiver successfully. One possible

reason for this can be that gpio_set_value () cannot satisfy the strict timing requirements of 200-500ns.

- **hrtimer and gpio_set_value ():**
  We were able to switch on all LEDs but were unable to switch them off. The possible explanation can be same as that of ndelay.

- **ndelay and iowrite32():**
  We tried setting the LEDs to red and then turning them off. We were able to do this by tweaking the delay using ndelay.
  We have used this mechanism to reset the GPIOs after each experiment.
  In the Log file, we would see *"Resetting GPIOs using iowrite32()"* for this purpose

- **hrtimer and iowrite32():**
  We tried setting the LEDs to white and then turning them off. We were able to set the LEDs but were not able to reset them. We can infer that the overhead of hrtimer does not satisfy the strict timing requirements of setting zero.

CONCLUSION

- We came to the conclusion that bit banging is possible using ndelay() and iowrite32().
- We have created a driver (WS2812.c) for this which works with the test program of Part 1.
- We have used spinlock to synchronize the sent bits as we were facing synchronization issues using bit banging approach.