

CMPE-250 Assembly and Embedded Programming

Laboratory Exercise 10

Timer driver

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students. Other than code provided by the instructor for this exercise, all code was developed by me.

Andrei Tumbar

Submitted: 11-10-20

Lab Section: 5

Instructor: Gordon Werner

TA: Tianran Cui
Anthony Bacchetta

Lecture Section: 1

Lecture Instructor: Melton

Abstract

In this laboratory exercise, a timer driver was implemented. The driver utilized the an interrupt driver timing design where a hardware driven peripheral device would interrupt the running program. The interrupt would increment a counter if the timing functionality was enabled. This exercise was successful as the goal to produce a timer accurate to the nearest 10ms was achieved.

Procedure

A command prompt was written that accepted commands: T, P, D, and C respresented timer start, pause, display, and clear respectively. The timer start command would enable a counter to increment every time the timing interrupt was run. This interrupt was setup to fire every 10ms. The pause command would stop the timing interrupt from incrementing the counter. The pause however, does not stop the interrupt from firing. The display command simply displayed the value of the counter in decimal representation. Finally the clear command cleared the counter to zero.

The counter was initialized to zero at the start of the program. We are using a 32-bit counter to allow the timer to time a long duration.

Results

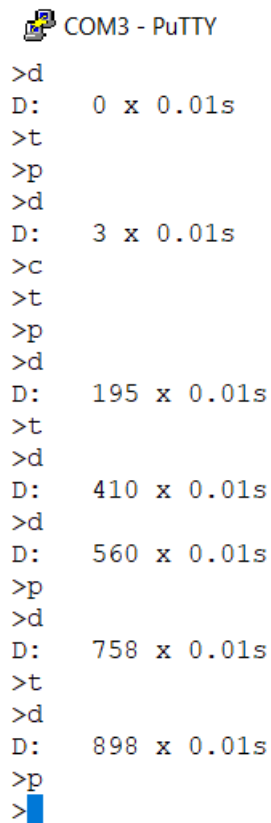
Memory ranges for various parts of the code were found.

Table 1: Code section offsets and endings.

Subroutine	Address	Ending Address
Executable Code	0x00000410	0x00000923
PIT ISR	0x00000753	0x00000768
Constants in ROM	0x000001c4	0x0000029b
RAM	0x1ffffd00	0x1ffffd6f

As seen in Table 1, the total size of the executable code was around 1.3k. The PIT_ISR was just 22 bytes as it was very simple subroutine. The constants contained some of the test prompts used in other labs making the total size 216 bytes. Finally the RAM section was reduced to a size of 112 bytes by decreasing the size of the print queues. The RxQueue could be reduced to just 5 bytes as input in this program could be processed faster than the user could provide it.

After the program was written and compiled, the executable was loaded into a KL-05 board to be tested.



```
COM3 - PuTTY
>d
D: 0 x 0.01s
>t
>p
>d
D: 3 x 0.01s
>c
>t
>p
>d
D: 195 x 0.01s
>t
>d
D: 410 x 0.01s
>d
D: 560 x 0.01s
>p
>d
D: 758 x 0.01s
>t
>d
D: 898 x 0.01s
>p
>
```

Figure 1: Terminal screen capture

Figure 1 shows the output of the terminal after each prompt command was used in various steps. The screen capture mainly shows the timer increasing its counter as time goes on. Meanwhile, the user can print the value of the counter or pause it as needed.

Conclusion

This laboratory exercise was useful in exploring hardware driven timing functionality. It was important to keep in mind performance of subroutines so that the timing functionality could be as accurate as possible. This exercise was successful as the desired functionality was implemented as demonstrated in Figure 1.