**CMPE-250 Laboratory Exercise Ten**

**Timer Driver Input Timing**

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; however, other than code provided by the instructor for this exercise, all code was developed by me.



Dean Trivisani

Performed November 6, 2017

Submitted November 12, 2017

Lab Section 01L2

Instructor: Melton

TA: Donald

Michael Baumgarten

Matthew Toro

Lecture Section 01

Professor: Melton

**Abstract**

This exercise utilized the Keil MDK-ARM Microcontroller Development Kit to write a Cortex-M0+ assembly language program (for the NXP Freedom KL46Z microcontroller with interrupt-based serial I/O and the universal asynchronous receiver/transmitter (UART) module) that implemented a periodic interrupt timer (PIT) that measured the time taken for the user to respond to prompts in the terminal. The purpose of this exercise was to improve familiarity with interrupt-based serial I/O, as well as explore how interrupts can be used to create a timer driver. A project was created in Keil µVision and subroutines for initializing the PIT interrupt requests and the associated interrupt service routine (ISR) were written and integrated with subroutines used for interaction with the microcontroller and the command prompt using transmit and receive queues that were written in a previous exercise. The main program code was then written to implement these subroutines in a program that would display three prompts to the user, and output the time taken for the user to respond to each prompt, before outputting a notification thanking the user and quitting the program. The program was then run on the KL46Z board using the µVision debugger, and the timing was determined to be accurate by comparing the program’s time values with an externally measured time value, which the program was congruent with. Therefore, the program functioned correctly.

**Procedure**

A project directory and project files were created in µVision using the NXP MKL46Z256VLL4 microcontroller on the NXP FRDM-KL46Z Freedom development platform, and subroutine Init\_PIT\_IRQ for initialization of the PIT to generate an interrupt every 0.01s from PIT channel 0 was implemented. Subroutine PIT\_ISR was written such that on a PIT interrupt, if the byte variable RunStopWatch was not equal to zero, the word variable Count would be incremented, and left unchanged otherwise. The PIT\_ISR was then installed by adding it to the vector table. Each of three text prompts asking (one asking for the user’s name, the second asking for the date, and the last asking for the name of a 250 lab TA), as well as a thank you message for the user and text to specify the format of the time output were defined as constants in the MyConst AREA, and the space for the queue buffers and the queue records for the receive and transmit queues were allocated using constants defined with EQUates, while the RunStopWatch byte variable and Count word variable were allocated with immediate values. Additional subroutines from previous laboratory exercises were used in addition to those written for this exercise to implement correct queue operation and formatting of output. The main program code was then written such that a prompt asking for the user’s name would appear, and the time the user has taken to input a response would then be output once input was entered. The program would then repeat with a prompt asking for the date and use the same time output. The program then looped a final time with a prompt for a name of a 250 lab TA. After all three prompts were answered, a thank you message would be output and the program would exit. The program was then tested on the KL64Z board using the µVision debugger, where correct operation was ensured by comparing the times output by the program to the times as measured by an online stopwatch. These values were found to be in agreement, and as such the program functioned correctly and produced accurate timer functionality.

**Results**

The program executed as intended, and the full functionality of a PIT was able to be implemented and used as a timer for inputs. The memory ranges given in Table 1 show the memory addresses and sizes of the executable main program portion of the code, the PIT\_ISR code, the constants in ROM (the string constants used for printing to the command line), and the RAM.

Table 1: Memory Ranges for Timer Driver Code AREAs

|  |  |  |  |
| --- | --- | --- | --- |
| AREA | Start | End | Size (Bytes) |
| Executable Code | 0x00000410 | 0x0000087F | 1136 |
| PIT\_ISR | 0x0000074F | 0x00000768 | 26 |
| Constants in ROM | 0x000001FC | 0x0000025F | 100 |
| RAM | 0x1FFFE100 | 0x1FFFE19F | 160 |

The command prompt text shown in Figure 1 consists of testing the timing on the program by entering inputs at a normal pace, then resetting and entering single character inputs as quickly as possible for each entry, then resetting a final time and entering input after approximately 10 seconds. The output times given by the program show that these times were measured accurately.

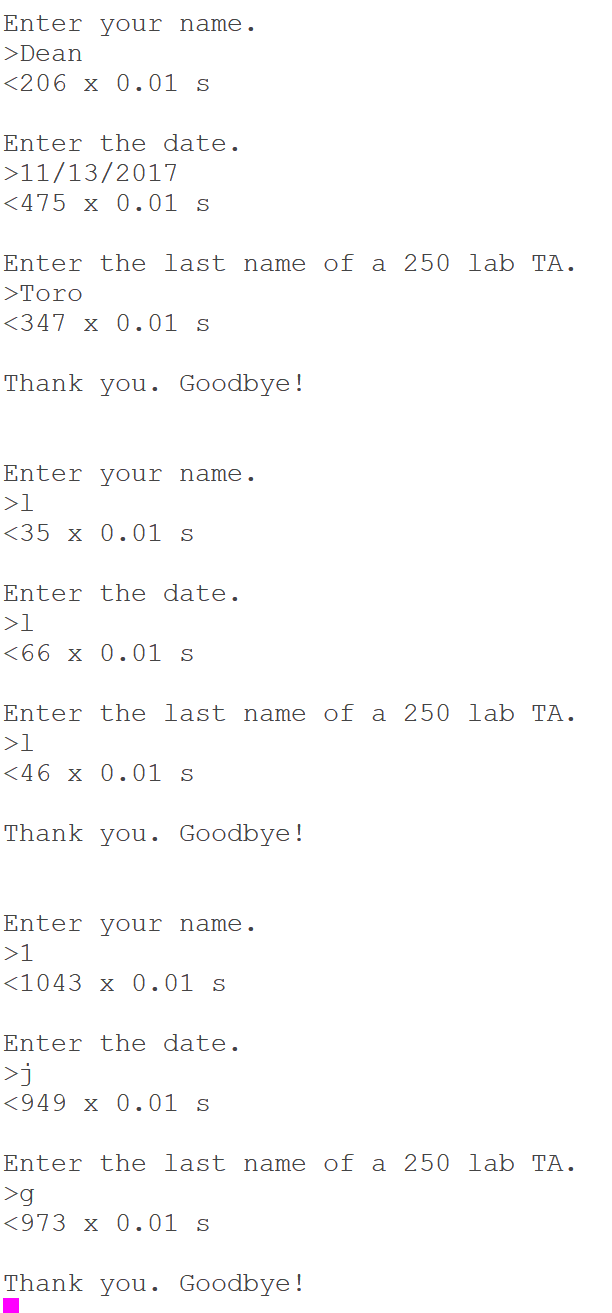


Figure : Timing Test for Program

**Conclusion**

This exercise gave insight into the functionality of periodic interrupt timers, as well as greatly improved familiarity with interrupt-based serial I/O. In addition, this exercise built specific knowledge of the utilization of timing hardware events, which is essential to device programming, especially where functionality is not based solely on responding to user interaction. Such knowledge is integral to computer engineering as a whole, as it forms the basis on which time-dependent systems can be designed and built.